ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ АВТОМАТИЗОВАНИХ СИСТЕМ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «РОЗРОБКА WEB-ДОДТАКУ ДЛЯ КЛІЄНТІВ АВТОСАЛОНУ НА БАЗІ МОВИ ПРОГРАМУВАННЯ PHP»

на здобуття освітнього ступеня магістра
зі спеціальності 126 Інформаційні системи та технології
освітньо-професійної програми Інформаційні системи та технології

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ *Олександр ТЕПЛЮК*
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав:
здобувач вищої освіти Олександр ТЕПЛЮК
група ІСДМ-62

Керівник: Оксана ТКАЛЕНКО
науковий ступінь, вчене звання к.т.н., доцент

Рецензент:
науковий ступінь, вчене звання _____
Ім'я, ПРІЗВИЩЕ

Київ 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра Інженерії програмного забезпечення автоматизованих систем

Ступінь вищої освіти Магістр

Спеціальність 126 – Інформаційні системи та технології

Освітньо-професійна програма Інформаційні системи та технології

ЗАТВЕРДЖУЮ
Завідувач кафедрою ІПЗАС

_____Каміла СТОРЧАК
«____» _____20___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ СТУДЕНТУ
Теплюку Олександру Владиславовичу
(прізвище, ім'я, по батькові здобувача)

1.Тема кваліфікаційної роботи: «Розробка Web-додатку для клієнтів автосалону на базі мови програмування PHP»

керівник кваліфікаційної роботи Оксана ТКАЛЕНКО, к.т.н., доцент
(ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від «19» жовтня 2023 року № 145.

2. Строк подання кваліфікаційної роботи: 28 грудня 2023 року.

3. Вихідні дані до кваліфікаційної роботи: Мова програмування PHP;
Бази даних MySQL;
Протоколи HTTP, HTTPS, SSH;
Фреймворк Laravel;
Науково-технічна література з питань, пов'язаних з наукою про веб-розробку, програмування на мові PHP та керуванням базами даних.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Дослідження особливостей веб-додатків та їх різновиду.
2. Аналіз сучасних технологій веб-розробки.
3. Дослідження та аналіз фреймворку Laravel.
4. Практична реалізація розробки веб-застосунку мовою програмування PHP з використанням фреймворку Laravel та бази даних MySQL.

5. Перелік ілюстративного матеріалу: *презентація*
   1. Огляд технологій і тенденцій у веб-розробці.
   2. Аналіз фреймворку Laravel та його особливості.
   3. Демонстрація робочого веб-застосунку та його функціоналу.
   4. Рекомендації по навчанню та дослідженням в області веб-розробки.

6. Дата видачі завдання: 19 жовтня 2023 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури	19.10 – 26.10.23	Виконано
2	Дослідження сучасних методів та технологій веб-розробки	27.10 – 02.11.23	Виконано
3	Аналіз видів веб-додатків	03.11 – 04.11.23	Виконано
4	Аналіз та дослідження фреймворку Laravel	05.11 – 10.11.23	Виконано
5	Реалізація та розробка веб-додатку	11.11 – 11.12.23	Виконано
6	Тестування готового проекту	12.12 – 13.12.23	Виконано
7	Оформлення роботи: вступ, висновки, реферат	14.12 – 22.12.23	Виконано
8	Розробка демонстраційних матеріалів	23.12 – 27.12.23	Виконано
9	Подання роботи в деканат	29.12.23	Виконано

Здобувач вищої освіти _____ Олександр ТЕПЛЮК
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник роботи
кваліфікаційної роботи _____ Оксана ТКАЛЕНКО
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 93 стор., 37 рис., 7 табл., 34 джерел.

Мета роботи – проведення комплексного аналізу сучасних методів і технологій веб-розробки, а також розробка веб-додатку для клієнтів на базі мови програмування PHP.

Об'єкт дослідження – використання мови програмування PHP для розробки web-додатків.

Предмет дослідження – web-додатки на базі мови програмування PHP з використанням фреймворку Laravel та бази даних MySQL.

Короткий зміст роботи: Досліджені сучасні методи та технології веб-розробки. Проаналізовано різновид веб-додатків та їх особливості. Досліджено технології SEO-оптимізації. Проаналізовано та досліджено фреймворк Laravel та його основні пакети для веб-розробки. Досліджено то проаналізовано використання баз даних MySQL для розробки веб-додатків. Здійснено реалізацію проекту, а саме, розроблено повноцінний веб-додаток для клієнтів автосалону з використанням мови програмування PHP, фреймворку Laravel та бази даних MySQL. Редактором коду виступав застосунок Notepad++. Для реалізації серверної частини та автономного розгортання було використано середовище Open Server. Використано панель PHPMyAdmin для керуванням базою даних. Виконано візуалізацію даних з використанням пакетів PHP Laravel Blade та JavaScript. Застосовано стилі CSS та HTML розмітку. Визначено роль PHP та фреймворків у веб-розробці.

КЛЮЧОВІ СЛОВА: ВЕБ-РОЗРОБКА, ВЕБ-ДОДАТКИ, ДАНІ, ІНФОРМАЦІЙНА СИСТЕМА, МОВА ПРОГРАМУВАННЯ PHP, БАЗИ ДАНИХ MYSQL, ТЕХНОЛОГІЯ, ФРЕЙМВОРК LARAVEL, ІНТЕРАКТИВНЕ СЕРЕДОВИЩЕ

ABSTRACT

Text part of the master`s qualification work: 93 pages, 37 pictures, 7 table, 34 sources.

The purpose of the work is to carry out a comprehensive analysis of modern methods and technologies of web-development, as well as the development of a web-application for clients based on the PHP programming language.

Object of research is using the PHP programming language to develop web applications.

Subject of research is web applications based on the PHP programming language using the Laravel framework and the MySQL database.

Summary of the work: Modern methods and technologies of web development are studied. The types of web applications and their features are analyzed. Researched SEO optimization technologies. Analyzed and researched the Laravel framework and its core packages for web development. The use of MySQL databases for the development of web applications has been studied and analyzed. The project was implemented, namely, a full-fledged web application was developed for car dealership customers using the PHP programming language, the Laravel framework, and the MySQL database. The code editor was the Notepad++ application. The Open Server environment was used to implement the server part and autonomous deployment. Used the PHPMyAdmin panel to manage the database. Performed data visualization using PHP Laravel Blade and JavaScript packages. CSS and HTML markup styles are applied. The role of PHP and frameworks in web development is defined.

KEYWORDS: WEB–DEVELOPMENT, WEB–APPLICATIONS, DATA, INFORMATION SYSTEM, PHP PROGRAMMING LANGUAGE, MYSQL DATABASES, TECHNOLOGY, LARAVEL FRAMEWORK, INTERACTIVE ENVIRONMEN.

ЗМІСТ

ВСТУП

Актуальність теми: сьогодні, так чи інакше, кожна людина використовує інтернет щоб задовольнити свої потреби. Замовлення продуктів, ліків, господарчих товарів або більш серйозні покупки, такі як, техніка, автомобілі, нерухомість та інші послуги. В Україні – ця тенденція вже давно набула великих обертів, та не збирається їх скидати. Для наших громадян, які є працьовитими, освіченими та діловими людьми – витрачання часу на оффлайн шопінг не завжди є раціональним рішенням. Людині зручніше скористуватится веб-додатком, зробивши пару кліків в своєму компьютері та придбати продукт, замовивши доставку або інший варіант покупки.. Все це поєднується у веб-застосунки, завдяки яким, люди можуть вільно та в зручний для них час, робити певні замовлення товарів або послуг. Тому розробка web-додатків для клієнтів набуває як ніколи найбільшої популярності в Україні та сучасному світі.

Мета і завдання дослідження: розробка web-додатку длі клієнтів на базі мови програмування PHP. У роботі передбачено проаналізувати сучасні тенденції та технології у web-розробці на мові програмування PHP. Створення зручно веб-додатку для клієнтів автосалону, яким в теорії буде зручно користуватися людям, які планують придбати автомобіль. Проаналізувати існуючи фреймоври та бази даних, зосередитись на певних технологіях та розробтити веб-застосунок. Визначити основні потреби клієнтів під час взаємодії з web-додатками.

Об'єктом дослідження є: використання мови програмування PHP, для розробки web-додатків.

Предмет дослідження: web-додатки на базі мови програмування PHP з використанням фреймворку Laravel та баз даних MySQL.

Методика досліджування: аналіз сучасних технологій веб-розробки на мові програмуваня PHP. Вибір для розробки фреймворку Larvel та бази даних MySQL та обгрунтування вибору. Розробка веб-додатку та демонстрація отриманих результатів.

Наукова новизна: з’ясовано, що на сьогодні PHP зберігає за собою місце однієї з найкращих мов для web-розробки. Обгрунтовано переваги технологій фреймворку Laravel та бази даних MySQL для веб-розробки. Робота з серверами та базами даних. Розроблено повноцінний веб-додаток для клієнтів, який включає в себе: зручніший функціонал, який забезпечує коректну взаємодію клієнта в web-додатком, повний стек розробки (фронт-енд та бек-енд розробка). (б-рівень наукової новизни: розширення і доповнення відомих даних без зміни їх суті).

Практична значущість результатів: результати даної розробки можуть використовуватися, як зразок або база для написання будь-якого веб-додатку, що спрямований на надання певних послуг та товарів для клієнтів. Розроблений веб-застосунок, можливо будуть використовувати у будь-яких компаніях, які займаються web-розробкою.

Апробація результатів та публікації:

1. Теплюк О.В. «Особливості розробки web-додатків для клієнтів». Тези доповіді на науково-технічній конференції «Технологічні горизонти: дослідження та застосування інформаційних технологій для технологічного прогресу україни і світу» – Київ, 28 листопада 2023 р.
2. Теплюк О.В. «Аналіз сучасних технологій розширення реляційних субд для роботи з часовими рядами». Наукова стаття у загальногалузевому науково-виробничому журналі «Зв’язок», м.Київ - №6, листопад-грудень 2023 р.

РОЗДІЛ 1. ДОСЛІДЖЕННЯ ТЕХНОЛОГІЙ WEB-РОЗРОБКИ

1.1 Розвиток web-розробки та розгляд її сучасних тенденцій

Розвиток веб-розробки є постійним і динамічним процесом, що відбувається під впливом різних технологічних змін. Сьогодні людство постійно взаємодіє з мережею інтернет тим чи іншим способом. Навчальні заклади, державні структури, підприємства та інші складові сучасного світу мають свій куточок на просторах інтернету. Цифровізація являє собою одну з найвеличезніших та найвпливовіших тенденцій сучасного світу, що спрямована на прискорення обробки інформації та поліпшення взаємодії людини з нею. Наукові та розважальні портали, сервіси надання послуг, соціальні мережі, тощо. Все це поєднує в собі інтернет. Існує безліч різноманітних веб-сайтів та веб-додатків, що спрямовані на покращення взаємодії людини з різною інформацією.

PHP є однією з найпопулярніших мов програмування, що використовуються для веб-розробки. Це мова, яка пройшла величезний шлях з 1995 року запуску й по сьогоднішній день. Починаючи з випуском PHP 3 та додаванням у нього підтримки об’єктно-орієнтованого програмування (ООП) і продовжуючи оновленням на PHP 8 з підтримкою JIT-компілятора та інших нових функцій і покращень продуктивності. PHP не втрачає своєї актуальності та не поступається іншим мовам веб-розробки в сьогоденні, таким як: Python, C#, Java та інші. Ця мова настільки оптимізована й адаптивна, що багато найсучасніших веб-додатків та серверів досі працюють на 5их версіях PHP, які були розроблені ще в далекому 2004 році. Вважається, що сьогодні ця версія є найпопулярнішою у веб-просторі. Звісно, час йде і з’являються нові версії мови програмування. На сьогоднішній день версія 8.3.0 є найсвіжішою в середовищі PHP та починає набувати нових обертів. Але розробники не забувають про старіші версії, та досі продовжують підтримувати деякі з них.

PHP продовжує залишатися потужним та важливим інструментом для веб-розробки, і нові версії та технології допомагають поліпшувати його продуктивність та можливості.

Мережа Інтернет складається з великої кількості комп'ютерів, маршрутизаторів та іншого обладнання, що необхідне для ефективної роботи. Всім відомо, що в основі цієї концепції знаходяться веб-сервера. Так звані комп'ютери, на яких зберігається різні дані та інформація, що доступні користувачам через веб-додатки. Кожен елемент мережі Інтернет, також відомий як вузол, має власний описник, який називається IP-адресою. Знаючи IP-адресу вузла, можна спробувати підключитися до нього, а з певним досвідом можна дізнатися, кому належить ця адреса та в якому регіоні світу вона знаходиться. У мережі Інтернет є спеціальні сервера DNS, також відомі як Domain Name System, на яких зберігаються всі списки зіставлення IP-адрес і символьних імен. Завдяки цьому, використовуючи лише ім'я сторінки в браузері, користувач може завжди перейти до серверів по потрібній IP-адресі. Для того, щоб переглядати веб-додатки з будь-якого девайсу, у користувача має бути встановлена спеціалізована прикладна програма, відома як браузер, яка зазвичай доступна безкоштовно. Але важливо розрізняти веб-додатки та розуміти, якою інформацією володіють ті чи інші ресурси [1].

Веб-розробка – це створення та підтримка веб-додатків. Сфера охоплює широкий спектр завдань, включаючи все: від кодування до технічного дизайну та продуктивності веб-сайту чи застосунку, що працює в Інтернеті. Існує три основних типи веб-розробки: фронт-енд, бек-енд і повний стек.

Інтерфейсна (фронт-енд) розробка відповідає за ті аспекти веб-додатку, які користувачі бачать і з якими взаємодіють: інтерфейс користувача (UI). Інтерфейсні розробники добре знаються на HTML, CSS і JavaScript, часто співпрацюють із командами дизайнерів і UX, щоб відобразити запланований вигляд і відчуття сайту, а також створити якісну взаємодію з користувачем на різних типах пристроїв.

Бекенд-розробка відповідає за всі аспекти веб-застосунку, які користувачі не бачать. Це також відоме як серверна розробка, оскільки бекенд-розробники

зосереджуються насамперед на закулісній логіці, програмуванні функціоналу та взаємодії з базами даних, які забезпечують роботу веб-додатку.

Розробка повного стека — це більш цілісний підхід, коли розробники, відповідальні за сайт або програму, піклуються про весь стек розробки, від внутрішньої роботи, яка зазвичай виконується на серверній частині, до рівня презентації, яким зазвичай керують розробники зовнішньої частини.

1.2 Різниця між веб-сайтом та веб-додатком

Трактування веб-сайтів та веб-додатків дуже схожі між собою для звичайних користувачів, але важливо розуміти та розрізняти їх. Основна відмінність між цими двома поняттями полягає у можливостях користувача при їх використанні. Складність функцій та інтерфейсу у веб-додатках набагато більша ніж у веб-сайтах, які розрізняють на статичні та динамічні. Статичний веб-сайт представляє собою набір статичних HTML сторінок, що в структурі зв’язані посиланнями. Ці сторінки розроблюються вручну, завантажуються та зберігаються на віртуальних серверах. Редагування таких даних та інформації відбувається за допомогою безпосередньо ручного втручання спеціалістів та зміни HTML сторінок. Незважаючи на те, що статичні веб-сайти мають деякі переваги з боку мінімального навантаження на сервер та швидкого завантаження, їх основні недоліки включа.ть складність змін, оновлення та редагування інформації [2].

В свою чергу динамічні веб-сайти мають розширений функціонал та навіть дозволяють редагувати інформацію без знань у сфері веб-програмування. Наприклад, такі веб-сайти можуть включати можливість публікації певних постів та інформації. Проте, якщо користувач в основному використовує цей ресурс для читання та перегляду наданої інформації, його все одно вважають веб-сайтом, а не веб-додатком.

Динамічні веб-сайти набагто гнучкіші в експлуатації. Існує безліч иехнологій, які допомогають розробляти дані ресурси вручну або за допоомгою певни систем керування контентом (CMS). CMS (Content Management System) – система керування вмістом веб-сайту. В професійному середовищі можна почути

назву «двигун сайту». Основна перевага цієї технології в тому, що для розробки не обов’язково мати навички та знання у веб-розробці. CMS дозволяє використовувати при розробці готові модули та скрипти для створення різноманітних динамічних веб-сайтів. Але CMS має багато суттевих недоліків, таких як:

— Потрібно постійно слідкувати за оновленнями версій;
— Обмеженний фунціонал розробки для реалізації певних задач;
— Не підходить для креативних та нетипових задач;
— Зниження продуктивності при великому накопиченні доповнень та інформації.

Популярні CMS на сучасному ринку включають WordPress, Joomla, OpenCart та Drupal.

Основна та найголовніша особливість веб-додатку полягає у його інтерактивності, та можливості користувача виступати активним учасником процесу, а не лише пасивном споживачем інформації. Прикладом такого додатку є будь’яка сучасна соціальна мережа, в якій користувач взаємодіє з додатком за допомогою лайків, репостів, особистих постів, публікацій фотографій та іншої інформації, тим самим впливаючи на свій рейтинг та рейтинг інших учасників процесу.

Переваги веб-додатків:

— Можливість працювати на декількох платформах, незалежно від операційної системи або пристрою;
— Уникнення проблем сумісності у користувачів, шляхом наявності і дсотупу у кожного до однієї, зазвичай найсучаснішої, версії додатку;
— Не потребують завантажженя на жорсткий диск пристрою, що усуває обмеження обсягу;
— Оптимізація під більшість пристроїв, завдяки чому відбувається зменшення витрат як для бізнесу, так і для користувача;
— Зменшення піратства програмного забезпечення, наприклад шляхом використання підписок SaaS, PaaS або інших.

1.2.1 Актуальність застосування та структура Web-додатків

У сучасномі світі, веб-додтаки є невід'ємною складовою процесу взаємодії людии з мережею Інтернет. Веб-додатки розробляються з різною метою та для різних цілей призначення. Ці ресурси доступні для користування різним категоріям споживачів. Деякі з веб-додатків можуть вимагати певного пграмного забезпечення або браузеру, але зазвичай більшість з них доступні незалежно від використовуваного браузера. Веб-додатки дозволяють користувачам швидко та зручно знаходити потрібну інформацію, а головне поліпшують взаємодію з нею.

Веб-додатки також дозволяють зберігати дані безпосередньо в базі даних та створювати певні звіти, для подальшого аналізу отриманих результатив тих чи інших дій. Прикладами таких застосунків є всім відомі поштові сервіси в мережі Інтернет, текстові редактори, електронні магазини, соціальні мережі, інструменти для редагування фотографій та відео, конвертація файлів та інше. Існують також комерційні і не комерційні веб-додатки. Їх основна вімдінність полягає в цілях розробки певного застосунку. Наприклад, інтернет магазини є комерційними веб-додатками та спрямовані на поліпшення пошуку та покупки певних товарів для споживача. В той час як до не комерційних можна віднести соціальні мережі та поштові сервіси. Але всюди існують певні ньюанси та можливість внутрішніх покупок. Популярні веб-додатки включають в себе, такі сервіси як Gmail, Amazon, Facebook, Rozetka, та багато інших [3].

Web-додатки, як правило, включають в себе три основні компоненти:

а) Клієнтська частина веб-додатку – графічний інтерфейс, який користувачить бачить в своєму браузері. Взаємодія відбувається шляхом натискання певних кнопок клієнтом, які відповідають за ті чи інші процеси та дії;

б) Серверна частина веб-додатку – програма, що розташована на сервері та обробляє запити користувача, які надходять від браузера з графічного інтерфейсу. Зазвичай ця частина застосунку програмується на професійних мовах веб-розробки, таких як PHP або Java. Під час кожної дії користувача в межаї клієнтської частини, браузер надсилає запит до сервера. В свою чергу сервер оброблює цей

запит, викликає певну функцію з заздалегідь запрограмованого коду, який формує веб-сторінку, використовуючи HTML і JavaScript, та відправляє її клієнту через мережу. Результат одразу відображається у вигляді веб-сторінки додатку в браузері. Зазвичай, такий серйозний процес займає меньше 1 секунди реального часу;

в) База даних (БД, СУБД) – програмне забезпечення на сервері, що зберігає дані в певні структури та надає можливість з ними взаємодіяти при необхідності. Серверна частина веб-додатку взаємодіє з базою даних, витягуючи необхідні дані для формування сторінок, яку намагається отримати користувач. За допомогою мережі Інтернет браузер відправляє HTTP-запити до веб-сервера. Сервер викликає функцію або метод, написаний розробником веб-додатку, який в свою чергу може звертатися до бази даних. Реузльтатом роботи такої функції є веб-сторінка з певною інформацією, яку браузер відображає користувачу.

1.2.2 Використання нових стандартів HTTP/2 та HTTP/3

Нехай користувач хоче відкрити у браузері певну сторінку сайту. Для цього він або переходить за деяким посиланням або вбиває URL-адресу сторінки в адресну строку браузера. При цьому обидва способи технічно еквівалентні. Після цього браузер відправляє серверу запит (англ. request), у якому просить сервер віддати сторінку за вказаною URL-адресою. Сервер отримує запит браузера, формує відповідь (англ. response) і відправляє його у назад браузер. Запит та відповідь є просто рядками, оформленими спеціальним чином. Правила оформлення цих рядків як раз і регулює протокол HTTP. HTTP – це клієнт-серверний протокол передачи гіпертексту. Всі веб-сторінки містять в собі гіпертекст та гіперсилки, завдяки яким відбувається певна взаємодія користувача з сервером. Отже, клієнт та сервер обмінюються один з одним повідомленнями, оформленими спеціальним чином.

Кожне HTTP повідомлення складається з трьох частин, які передаються у вказаному порядку: стартовий рядок (англ. starting line), заголовки (англ. headers) та тіло повідомлення (англ. message body). Технічно стартовий рядок є першим рядком повідомлення, потім в кожному новому рядку розміщується по одному заголовку, далі йде порожній рядок і після нього розміщується тіло запиту. При цьому стартовий рядок визначає тип повідомлення, та включає в себе URL-адресу та метод запиту. Існують такі методи запиту, як Post, Get, Put, Delete, Head. Розшифровка даних методів зображена в табл.1.1.

Таблиця 1.1

Розшифровка методів запитів

Post:	(submit data to the server) – відправити дані на сервер
Get:	(retrieve information) – отримання інформації
Put:	(save object) – збереження об’єкта
Delete:	(delete the object) – видалення об’єкта
Head:	(retrieve resource headers) – отримання заголовків ресурсів

Стартовий рядок відповіді від серверу відрізняється від запиту. В ній передається трьох значний код стану обработки запиту. Наприклад, якщо все пройшло вірно ми отримаємо код 200 – Ok, або якщо якась помилка це зазвичай код 404 – Not Found. Всього існує 5 типів відповідей від сервера. Їх прийнято класифікувати структурою наведеною у табл.1.2.

Таблиця 1.2

Класифікація кодів відповідей сервера

1XX:	інформування про процес передачі запиту.
2XX:	інформація про успішну передачу запиту.
3XX:	інформування про перенаправлення.
4XX:	помилка клієнта, яка вказує на те, що клієнт шукає щось чого нема або задає неправильний запит.
5XX:	помилки вказуючи на несправний стан сервера.

Заголовки характеризують тіло повідомлення, параметри передачі та інші відомості, а тіло безпосередньо містить дані повідомлення. Стартовий рядок та заголовки є обов'язковими елементами, а тіло – ні. При цьому стартові рядки відрізняються для запиту та відповіді. Дуже важливий момент, що всі ці данні протокол HTTP передає в звичайному вигляді як відкритий текст. Тому, якщо десь на шляху до сервера зловмисник перехватить цей текст, він отримає відкритий доступ до всіх данних, які користувач хотів надіслати.

Саме через це було розроблено спеціальний протокол HTTPS, приставка S в якому означає – Secure (від англ. захищенний). Цей протокол шифрує всі данні передачі та не дає змогу їх прочитати. Основні технології такого шифрування включать в себе SSL та TLS. Але в сучасному світі всі працюють з новітнімі версіями TLS 1.2 та TLS 1.3, які мають більшу захищеність та швидкість ніж їх попередники. Очевидно, що новітня версія TLS 1.3 має бути кращою і вона виправдовує себе. Різницю між 1.2 та 1.3 зображено на рисунку 1.1 за допомогою наглядного зменшення «рукопотискань» та відповідного збільшення швидкості.

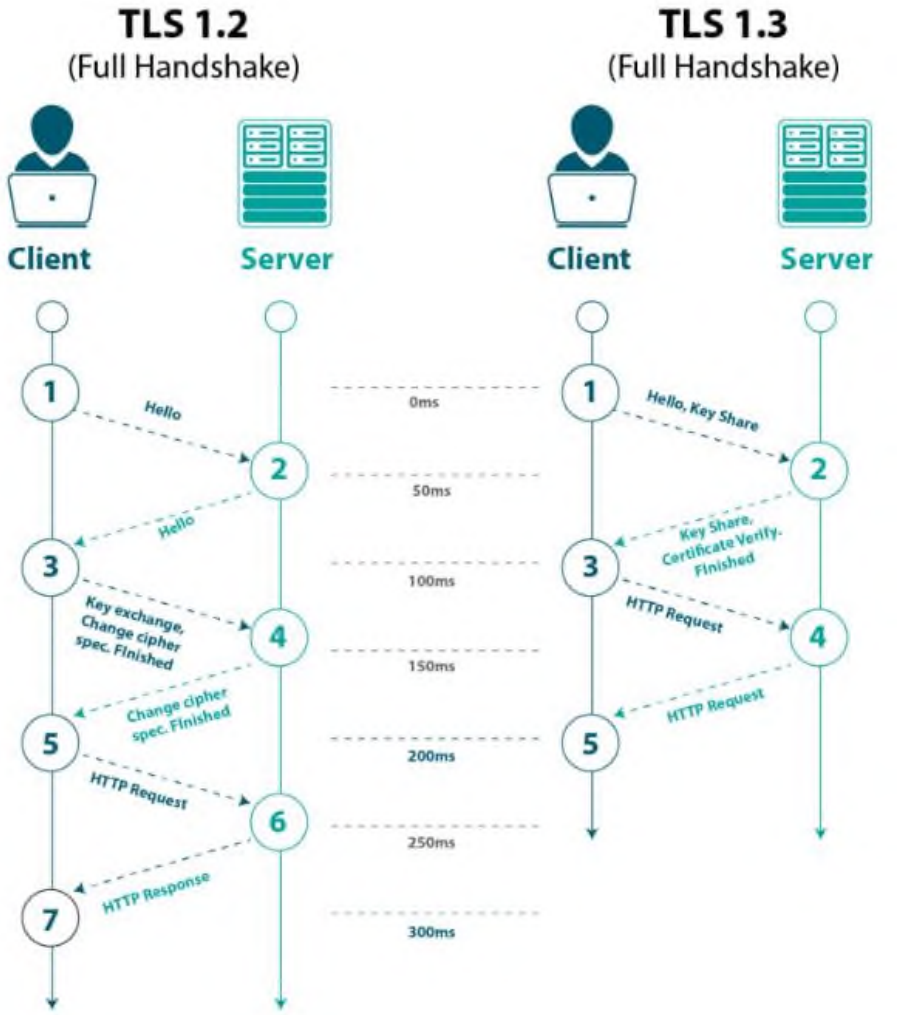


Рис. 1.1 Різниця TLS 1.2 та 1.3

TLS було розробленно для забезпечення трьох основних концепцій: конфеденційність, цілісніть та аутентифікація. Найбільш розповсюдженні дані – це веб-дані, якими користувач обмінюється з веб-сервісами: логіни, паролі, дані банківських карток та інші. TLS має можливість захисти голос, відео, пошту та майже будь'яку інформацію, яку наразу можливо передвати мережою Інтернет. Конфеденційність гарантує, що ніхто не зможе перехватити дані, які передає користувач або сервер. Навіть якщо зловмисник перехватить дані, вони будуть зашифровані за допомогою ключів та мати незрозумілий для простого читання вид. Цілісніть гарантує те, що дані які відправляютья не будуть змінені під час переправкі від клієнта до сервера [4]. Для того щоб забезпечити всі три вимоги, в TLS застосовуються такі механізми шифрування як симетричний та асиметричний. Симетричне шифрування являє собою використання одного ключа для зашифровки

та разшифровки повідомлення. Простота даного підходу змусила розробити додатковий, більш надійний підхід шифрування – асиметричний. Цей підхід містить в собі вже два ключа: публічний та приватний. Клієнт шифрує дані за допомогою свого приватного ключа, а відправляє їх до серверу вже шляхом застосування публічного ключа серверу [29]. Сервер в свою чергу отримаючі дані, розшифровує їх за допомогою свого приватного ключа. Таким чином, якщо зловмисник все ж таки отримає дані, він не зможе їх розшифрувати за допомогою тільки публічного ключа. Також, однією з головних задач цього протоколу, являється перевірка чи справжній цей веб-застосунок або сайт. Аутентифікація в свою чергу відповідає за безпечність ресурсу з яким хоче працювати користувач. Це відбувається шляхом запиту від сервера SSL сертифікату, який видається веб-ресурсам певними центрами сертифікації. Вони перевіряють власника та його права на ресурс. Після того як бразуер впевнився в його справжності - починаєтья обмін зашифрованими даними. Всі сучасні браузери занижують приорітет ресурсів без HTTPS при пошуку, а Google взагалі планує відмовитись від роботи з незахищенними ресурсами.

Версія HTTP 1.1 була випущена в далекому 1999 році. На той момент веб-простір тільки набирав своїх обертів. Всі веб-ресурси мали примітивну структуру а про веб-додатки й взагалі мови не йшло. Через це веб-сторінки були невеликого розміру та пропускної здатності вистачало на той момент. Але з часом розвитку Web становився більш насиченим, та веб-ресурси відповідно зростали в об’ємі та накопиченні інформації. На фоні цього треба було передавати та обробляти більше даних. Це все послугувало для допрацювання існуючого HTTP 1.1 та розробки нової версії протоколоу – HTTP 2, яка вийшла в світ аж в 2015 році.

Тепер заголовки передаютья в зжатому вигляді, також є можливість об’єднати декілька з’єднань з сервером в один потік даних. В першій версіїї HTTP 1.1 була так звана конвеєрна передача даних. На рисунку 1.2 зображена наглядна різница між передачею даних в HTTP 1.1 та HTTP 2.

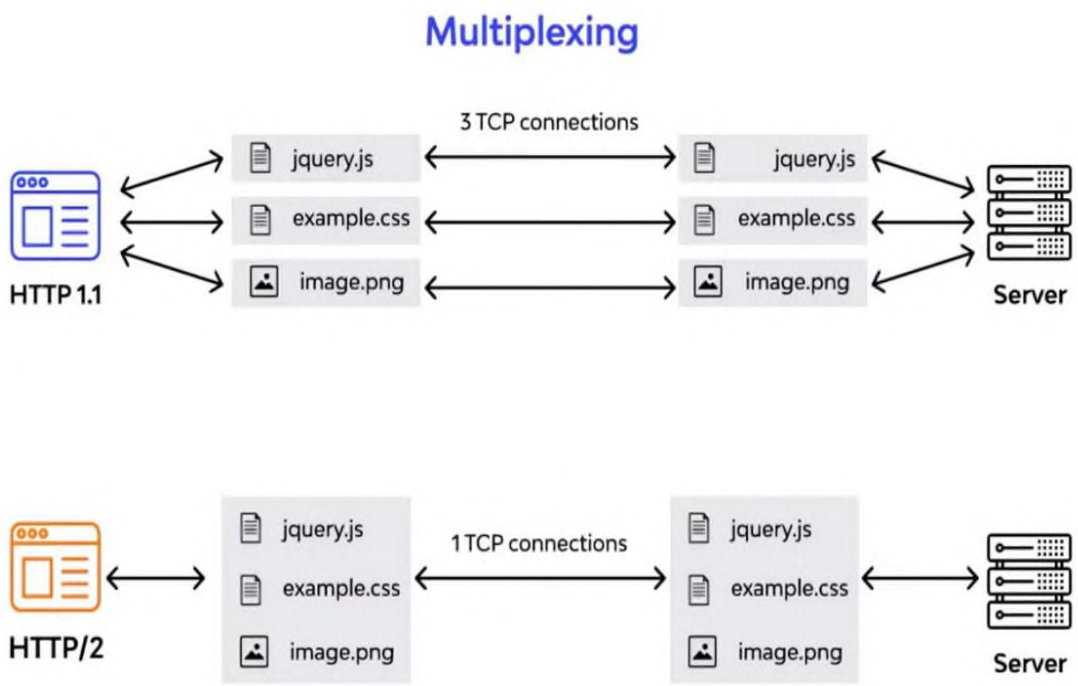


Рис. 1.2 Різниця передачі пакетів

Запити відправлялися по черзі та очікували поки попередній отримає відповідь від сервера. В новій версії цю структуру замінили на мультиплексову передачу даних, коли можлива одразу передача декількох запитів в режимі однієї сесії. Очевидно, що перша версія протоколу потребує зв'язок під кожний пакет даних, який потрібно передати [18]. В новітній версії це прибрали і дозволили передавати дані в межах одного зв'язка незалежно від їх розміру та формату. Цікавою особливістю оновлення є підтримка, так званих, push'ів від сервера. Сутність цієї концепції полягає в можливості сервера відправити певні дані, схожі за сенсом, клієнту ще до того, як він сам їх запросить. Наприклад, клієнт бажає придбати автомобіль та його цікавить не тильки зовнішній вигляд а ще й технічки характеристики або контакти автосалону. Це дозволяє попереднеьо закешувати певні дані та в момент коли вони знадобляться, просто витягнути їх з кешу без додаткових сесій та запитів на сервер. Також, HTTP 2 тепер працює з двійковими даними на відміну від першої версії, яка працювала з текствоими даними. Все це забезпечує зменшення час очікування від сервера та передачу набагато більшого об'єму інформації.

Третя версія протоколу передбачає собою ще більшу оптимізацію та покращення обробки даних у веб-сервісах. Розробка HTTP 3 почалась ще у 2019-2020 роках, але досі не набуло стандартизації та кінцевої точки реалізації. Але незважаючи на це, третю версію HTTP активно використовують всі сучасні браузери та дають змогу їй розвиватися. Основною відміною HTTP 3 від попередників є робота з протоколом UDP/IP + QUIC, в той час як 1.1 та 2 версіх працювали на TCP/IP. На рисунку 1.3 наглядно зображений змінений стек технологій, який використовується другою та третьою версіями протоколу відповідно до слоїв моделі OSI.

TCP являє собою структуровану обробку та відправку даних. Йому потрібно встановити зв’язок між двома вузлами передачі даних, отримати відповідь про готовність передачі інформації і вже тільки потім надсилати пакети даних. Але якщо якись пакет не відправився або був пошкоджений, доведеаться повторно робити пересилку та отримувати підтвердження. Очевидно, що такий підхід займає час, але він є надійним.

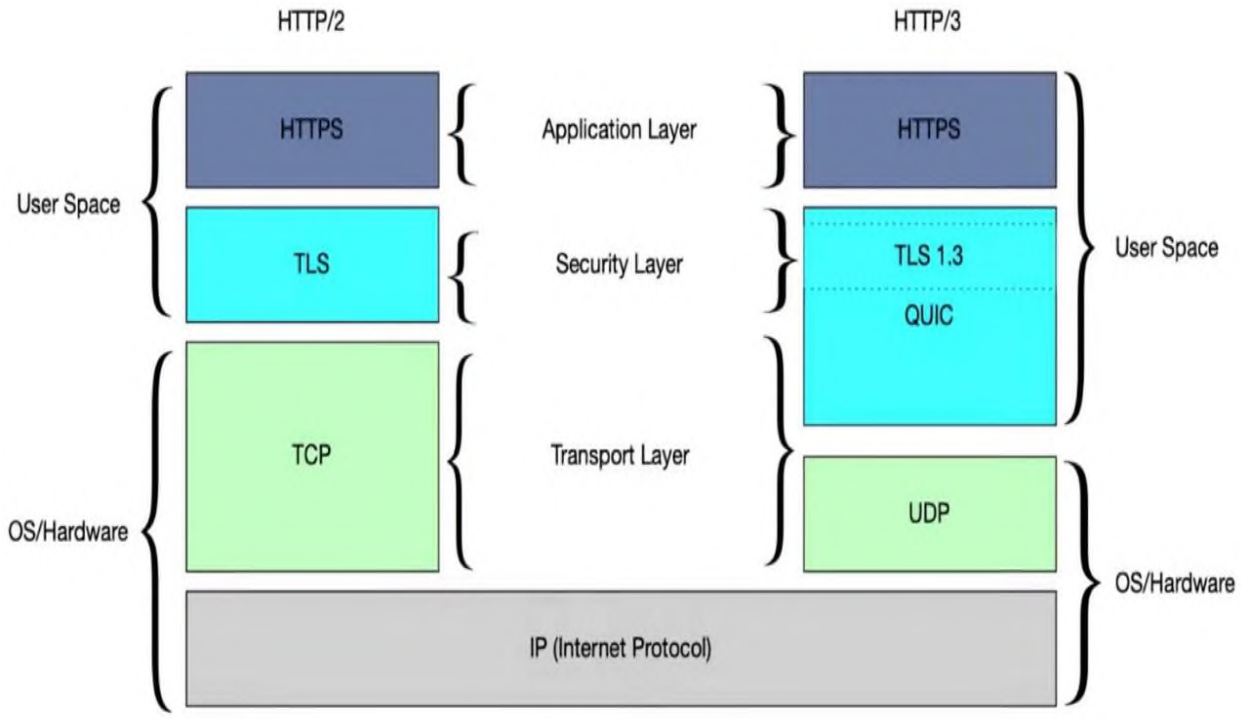


Рис. 1.3 Структурна різниця між 2 та 3 версіями HTTP

Особливість UDP полягає в тому, що йому не обов'язкове встановлення зв'язку між відправником та отримувачем. Інформація передається без попередньої перевірки готовності отримаючої сторони. Це робить протокол не таким надійним та впорядкованість даних не зберігається, деякі фрагменти могуть губитися й ненадходити. Можливий не послідовний прийом інформації отримувачем але швидкість збільшується в декілька разів. Зрозуміло, що втрачання даних це погано, але іноди цим можна знехтувати. Наприклад, коли ми дивимось якусь онлайн трансляцію або спілкуємсоь через інтернет телефонію. Якщо якись фрагмент відео або звуку на секунду втратится, користувач може це навіть не відчути але зрозуміє сутність передаваємої інформації по трансляції. В цих випадках TCP протокол відходить на другий план, бо довелося б отримувати відповідь при відправці кожного слова а це великий обсяг часу [18]. У випадку не отриманого пакету даних з сесії, UDP доведеться тільки повторити відправку саме цього пакету, в той час як TCP буде надсилати всю сессію повтороно.

Зазничимо те, що TCP – це точна та підтверджувана передача даних. Наприклад: відправка фотографій, листування в мережі між користувачами, банківські транзакції. Загалом це чутливий до втрат трафік. UDP в свою чергу потрібен для спілкування в голосовому форматі або для передачі потокового відео, наприклад трансляція з веб-камер або ip-камер. Підбиваючи висновки, можна сказати що кожна з версій протоколів має свої плюси та мінуси. HTTP 2 - це про надійність та швидкість порівняно з першою версією протоколу. HTTP 3 – це про технологічне майбутне, недивлячись на його постійне використання і в сучасному web-просторі. Об'єднує ці дві сучансі версії – їх оптимізація продуктивності та покращення обробки даних.

1.2.3 Класифікація веб-додатків

В сучасній веб-розробці існує три основні концепції створення веб-додатків: прогресивні (PWA), односторінкові (SPA) та багатосторінкові (MPA). Їх головне відокремлення від інших підходів полягає в простоті розробки та зручністю

взаємодії з користувачами. Також, наголошується на об’ємних та багато функціональних можливостях для інтгрецій і розвитку з бізнесом [19].

PWA (прогресивний веб-додаток)

PWA (Progressive Web Application) – представляє собою застосунок, створений за допомогою відомих усім веб-технологій, таких як HTML, CSS та JavaScript. Зазначимо, що прогресивний веб-додаток володіє функціональністю традиційного додатка. PWA включають у себе можливості, такі як push-повідомлення та можливість роботи в атономному режимі. Крім того, вони базуються на сучасних API, що спрощує надання розширених можливостей, забезпечує надійність та дозволяє їх встановлювати на будь-якому пристрої. Наведені додатки ефективно використовують переваги обширної веб-екосистеми, що включає в себе плагіни, спільноту та простоту розгортання і підтримки веб-застосунку. Така концепція іде в супереч звичайному додатку, який має складніший процес розробки та виходу на продакшн. Очевидно, що створення PWA є швидшим та простішим.

Деякі провідні компанії, такі як, Uber, Spotify, TikTok та інші, вже використовують PWA. Особливість цих продуктів полягає в можливості встановлення їх на домашньому екрані пристро, роботі в автономному режимі та наявності функціоналу, який зазвичай характерний для звичайних додатків, встановлених в операційну систему пристрою.

PWA розробляються як веб-додатки, що означає їх сумісність з будь-якою операційною системою та можливість використання користувачами в будь-якому браузері. Чимало організацій, як приватних, так і державних, обирають PWA не лише через їх вигідність з точки зори екномоії витрат на розробку, але й через активну взаємодію з користувачем.

Переваги PWA:

— Доступні і зручні у використанні;
— Низька вартість обслуговування;
— Автономний режим роботи: PWA кешуються в пам’яті браузера, через це не

втрачаючи працездатності оффлайн. Така концепція дуже підходить для компаній, які розробляють комерційні веб-додатки для продажу своїх товарів. Споживачі можуть офлайн продивлятись каталоги товарів та зручний для них час, що потенційно збільшує попит та продаж;

— Покращена працездатніть: Оскільки PWA використовує service worker, що містить файли JavaScript, які запускаються окремо від основної сесії браузеру та активно контролють кешування ресурсів. Це може забезпечити набагато кращу працездатність, ніж традиційні веб-додатки;

— Автономність оновлення додатку: розробник оновлює свій PWA та користувачам не потрібно робити це вручну. Все що треба це просто перезавантижит сторінку бразуеру, що займає секунду часу. Користувачу не потрібно заходити в App Store або Play Store та оновлювати застосунок власноруч. Це поліпшує сам процес і для розробників. Оновлення можливо робити постійно в незалежності від своїх користувачів;

— Незалежність від магазину застосунків: гарна новина для маленьких компаній з невиликим бюджетом. Не потрібно сплачувати додаткові податки Apple та Google за публікацію свого застосунку в їх магазинах.

Недоліки PWA:

— Проблеми зі старими пристроями: PWA являє собою доволі новітню технологію. Через це деякі пристрої зі старими версіями веб-браузерів можуть конфліктувати та не працювати з сучасними веб-додатками розробленими на PWA;

— Робота з iOS: погана співпрацця зі старими версіями операційної системи iOS, якщо точніше, то має можливість працювати тільки на останніх версіх iOS 12+.Також, варто зазначити що, Apple не дозволяє PWA отримувати доступ до деяких даних, в особливості до FaceID, Bluetooth та навіть інформацію про стан акумулятора;

Після проведеного аналізу PWA можна зрозуміти, що переваг набагато більше ніж недоліків. Технологія достатньо новітня, відповідно потребує більшої уваги та вдосконалень зі сторони розробників. Зазначимо, що PWA це та

технологія, яка буде як ніколи актуальна в епоху мобільний пристроїв та веб-розробки.

SPA (односторінковий веб-додаток)

SPA (Single Page Application) – ялвяє собою застосунок, який функціонує безпосередньо у браузері та не потребує постійного перезавантаження сторінки під час використання. Наведеними веб-додатками ми користуемось кожного дня. Приклади включають Facebook, Google Maps та GitHub. Односторінкові застосунки спрямовані на забезпечення відмінного користувацького досвіду (UX), намагаючись емулювати природнє середовище у браузері без перезавантажень сторінок та додаквих витрат часу на очікування.

Досвід користування/користувача (англ. User Experience, UX) – це той досвід та враження користувача, які він отримує при взаємодії з певними веб-застосунками. Сутність полягає в аналізі та забезпечення максимальної гармонії користувача з веб-додатком. Шляхом попереднього аналізу або опитувань паралельно роботи з додатком, розробники визначають, на скільки наданий функціонал є приемним до користувачів та наскільки він задовольняє їх потреби. Цей термін у веб-розробці більш притаманний для фронт-енд розробки, але бек-енд програмістам також варто знати про його існування. UX-розробник займається розробкою, так званого потоку додатку (application flow). Він визначає концепцію, за якою будуть розташовані всі елементи додатку, та будуть направляти користувача саме по тим запитам, які йому потрібні. Цей інтерфейс має максимально ефективно задовольнити потребу користувача в наданій йому інформації.

SPA представляє собою лише одну веб-сторінку, яку ми відвідуємо, і яка динамічно завантажує контент за допомгою JavaScript. Дані веб-додатки при розробці потребують використання сучасних фреймворків з більшим уклоном на фронт-енд розробку мовою JavaScript, такі як AngularJS, Meteor.js та Vue.js. Односторінкові додатки самостійно запитують розмітку та потрібні дані для формування сторінок безпосередньо у браузері. Це дозволяє забезпечити єдиний та

зручне веб-середовище для користувача, в той час як контент представленно у простій та зручній функціональній формі.

Переваги SPA:

— Односторінкові веб-додатки працюють швидко через їх простоту. Через те, що більшість файлів застосунку (HTML, CSS) завантажуются лише один раз, на початку роботи додатку, це забезпечує меньше навантаження на сервер та очікування відповідей. Залишаються лише унікальні дані, які передаються на сервер або користуву. Звідси випливає, що не потрібно програмувати та формувати сторінки на сервері. Очевидно, це дуже спрощує розробку. SPA майже примітивно налаштовується в браузері Google Chrome. З'являється можливість контролю мережевих сеансів та досліджування елементів і даних сторінки. Оптимізованість та простота коду дозволяє розробнику використовувати однакові напрацювання серверної бекенд частини для разгортання, як звичайної мобільної так і браузерної версії веб-додатку;

— Можливість SPA ефективного кешування великого обсягу даних в певні локальні сховища. На сервер не надходить постійного навантаження у виді запитів та відповедей, відповідно нема великого використання ресурсів веб-серверів. Це дозволяє використовувати меншу кількість серверів для одного й тогож трафіку. Застосунок робить лише один запит на сервер, зберігає всі дані та навіть дозволяє працювати з ними в автономному режимі.

Недоілки SPA:

— SPA має погану систему безпеки через його примітивність та односторінкову концепцію. Зловмисники з легкістью можуть перехоплювати дані, вставляти власні скрипти та видавати себе за користувачів. Але якщо виконувати захист кінцевих точок даних – цьому можна запобігти;

— Оптимізувати такі додатки за допомогою SEO – проблематично. Їх вміст завантажується через AJAX підхід (асинхронний JavaScript та XML);

— SPA система призначена працювати лише з однією веб-сторінкою. Через це, не відбувається перемикання між сторінками зі сторони браузеру. Натискання користувачем стрілки назад в інтерфейсі самого веб-браузеру ні до чого не

призведе. Відсутній «стан» веб-додатка, який користувач бачить на своєму пристрої. Для браузера існує тільки одна веб-сторінка застосунку, на якій має працювати користувач. Для цього розробникам потрібно відповідально підходити до внутрішнього функціоналу застосунку.

Angular, React та Vue – сучасні фреймворки, які допомагають розробникам ефективно та швидко створювати SPA за допомогою JavaScript.

MPA (багатосторінковий веб-додаток)

MPA (Multi Page Application) – мабуть, напопулярніша та найпотужніша концепція в сучасній індустрії веб-розробки. Традиційна технологія, насичена величзним функціоналом та можливостями, які допомагають розробляти веб-додатки, що потребуть роботи з масивним об'ємом даних. MPA складається з кількох веб-сторінок, завантаження з серверу яких відбувається тільки тоді, коли користувач захоче на них потрапати. Прикладом можуть послугувати всім відомі маркет-плейси та інтернет магазини, такі як Rozetka, Amazon або Steam. Переходячи по вкладкам цих веб-застосунків, користувач кожним натисканням відправляє запит до сервера за тією чи іншою веб-сторінкою. Сервер отримує запити та відправляє потрібні сторінки клієнту до веб-браузеру. Особливість полягає в тому, що кожна сторінка завантажується безпосередньо з серверною частини проекту. Як правило, даний тип веб-застосунків є складним, містить багато-рівненву архітектуру, різноманітні інтерфейси вкладені у веб-сторінки та потребує суттєвих навичок у веб-розробці. Але кінцевий продукт оправдовує очікування. Величезна багато-функціональність, яку включає в себе MPA, задовольняє всі потреби користувачів та допомагає їм досягти своїх цілей. Через це, багато сучасних компаній гігантів, які надають користувачам певні дані та послуги у великому обсязі, продовжують працювати з багатосторінковими веб-застосунками та не збираються змінювати вектор руху.

Як вже стало зрозуміло, MPA потребують багато уваги при розробці серверної частини та бекенду. Провідні технології розробки та мови програмування дозволяють виконувати найрізноматніші задачі в цьому векторі. PHP, Java, Python,

Ruby – все це про MPA та про об’єктно орінтовне програмування, без якого розробка сучасного багатосторінкового веб-додатку майже неможлива. На допомогу розробникам приходять різноманітні фреймворки, такі як Laravel, Django, Spring та Symphony.

Переваги MPA:

— Найбільшою та найсуттєвішою перевагою MPA над колегами, є необмежений обсяг та вміст інформації, яку додаток може обробляти та надавати. З огляду на те, що кількість сторінок не має обмежень, з’являється можливість відведення окремої сторінки під кожну з функцій, продукт або послугу. І найголовніше, що кожна така сторінка здатна обробляти будь-який обсяг інформації. Навідміну від односторінкового веб-додатка, де черезмірне завантаження сторінки інформацією може викликати великий час завантаження, а в деяких випадках при поганій оптимізації – «краш» застосунку;

— За допомогою MPA компанії та розробники мають можливість проводити аналіз ринку та діяльності, використовуючи аналітичні дані, що надходять з Інтернету. Існує велика кількість критичних аналітичних даних щодо активності в мережі, які підлягають можливості обробки. Аналітичні системи надають інформацію про відвідувачів та час, який вони проводять на певних сторінках веб-застосунку;

— Оскільки MPA сумісні з пошуковими системи, вони мають можливість легко взаємодіяти з SEO-оптимізаційними технологіями. Кожна сторінка в MPA має свою унікальну URL-адресу, що допомагає пошуковим систем краще індексувати, обробляти та розуміти вміст веб-застосунку.

Недоліки MPA:

— Великий обсяг роботи та час. Через свою багато-функціональність MPA потребують копіткої розробки з нуля. Через це ціна розробки таких веб-додатків може «кусатись»;

— Обслуговування та оновлення MPA застосунків можуть бути проблематичними через великий обсяг інформації. Але при правильній структурі та оптимізації розробником додатку, ця проблема може зникнути;

— Місце на сервері. Через великий об’єм інформації, яка потребує своїх баз даних

та їх керування, MPA потребують більш затратних рішень в цьому векторі; — MPA завантажують кожну сторінку знов, коли до неї надсилається запит, незважаючи на те, що хвилину назад вона вже була зідяна користувачем. Технічно, це збільшує час обробки інформації, але для користувача це мілісекунди часу.

1.3 SEO Оптимізація

Для більшості з нас сьогодні, коли нам щось потрібно — будь то відповідь, ідея, стратегія чи послуга — ми починаємо з запиту в пошукових системах. Тільки Google отримує 3,5 мільярда пошукових запитів на день. Так само, як пошукові системи стали невід'ємною частиною нашого життя, вони також стали невід'ємною частиною багатьох маркетингових стратегій бізнесу. Насправді 49% маркетологів вважають органічний пошук є найвищою рентабельністю інвестицій.

SEO (Search Engine Optimization) розшифровується як «оптимізація пошукових систем». Це набір технічних і контентних практик, спрямованих на узгодження сторінки веб-сайту з алгоритмом ранжування пошукової системи, щоб її можна було легко знайти, сканувати, індексувати та показувати в пошуковій видачі за відповідними запитами. Простими словами, SEO означає процес покращення вашого веб-додатку для збільшення його видимості в Google та інших пошукових системах щоразу, коли люди шукають певні матеріали, продукти або послуги [27]. На рисунку 1.4 наглядно зображено розшифровку даного поняття.

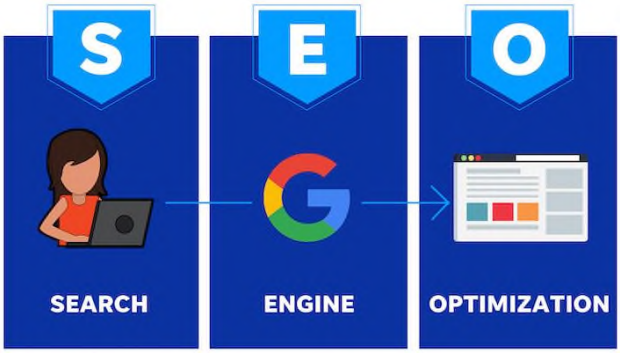


Рис. 1.4 Розшифровка SEO

Пошук (Search): Що робить користувач, коли хоче знайти відповідь на запитання або продукт чи послугу, які відповідають його потребам.

Пошукова система (Engine): браузер (наприклад, Google), на якому користувач може виконувати вказаний пошук.

Пошукова оптимізація (Optimization): що ви робите, щоб зазначена пошукова система з'єднала цей пошук із вашим веб-додатком.

Чим краща видимість ваших сторінок у результатах пошуку, тим більша ймовірність, що вас знайдуть і натиснуть. Зрештою, мета пошукової оптимізації полягає в тому, щоб допомогти залучити відвідувачів веб-сайту, які стануть клієнтами, клієнтами або аудиторією, яка постійно повертається.

SEO – це те, що ви робите, щоб підвищити рейтинг у Google і отримати більше трафіку на свій веб-застосунок. Так, Google – лише одна пошукова система з багатьох. Існують інші браузери та пошукові системи. Навіть Instagram є пошуковою системою. Але захоплюючи 92% частки ринку, терміни «Google» і «пошукова система» є синонімами в сучасному світі. Також, існує ще два розповсюджених терміна у сфері пошукової оптимізації – це SEM (Search Engine Marketing) та PPC (Pay Per Click). SEM розшифровується як двигун пошукового маркетингу або, як його більш відомо називають – пошуковий маркетинг.

Пошуковий маркетинг (SEM) являє собою різновид цифрового маркетингу. Це загальний термін для поєднання SEO та PPC, спрямованих на залучення трафіку через звичайний і платний пошук. SEM – це процес отримання трафіку та видимості в пошукових системах за допомогою як оплачуваних, так і неоплачуваних зусиль. Різниця між цими поняттями наведена в таблиці 1.3. Сьогодні багато розробників використовують SEM як взаємозамінні з PPC та SEO.

Таблиця 1.3

Різниця між SEO, PPC та SEM

SEO	PPC	SEM
Включає в себе залучення органічного трафіку з пошукових систем.	Включає в себе залучення платного трафіку з пошукових систем.	Являє собою абстракцію та об'єднання в собі цих двох попередніх понять та представляє залучення як органічного так і платного трафіку.

Органічний пошук – це безоплатні результати пошукової видачі, які вдображаються відповідно до запиту користувача. Пошукова система, така як Google, надає користувачеві перелік результатів пошуку, автоматично відбираючи відповідні варианті для надання бажаної інформації. Органічний пошук виступає основним джерелом трафіку для багатьох веб-додатків та веб-сайтів, що робить його надзвичайно цінним для їхніх власників, оскільки він є безкоштовним і стабільним. Однак, існують ньюанси в сенсі часу та скільки його знадобиться для ефективної SEO-оптимізації додатку.

PPC (pay-per-click) – тип цифрового маркетингу, де рекламодавці стягують плату щоразу, коли натискають одне з їхніх оголошень. В основному рекламодавці роблять ставки на певні ключові слова чи фрази, за якими вони хочуть, щоб їхні оголошення з’являлися в результатах пошуку. Коли користувач шукає одне з цих ключових слів або фраз, реклама рекламодавця з’являється серед результатів. Отже, якщо розглядати пошуковий маркетинг як монету, SEO та PPC – це дві сторони однієї монети: SEO – неоплачувана сторона, PPC – платна. Ще один ключовий момент: важливо ніколи не думати про це як про «SEO проти PPC» (тобто, який із них кращий), оскільки це взаємодоповнюючі канали. Це не питання «або-або» – завжди потрібно вибирати обидва варіанти (якщо дозволяє бюджет) [28].

SEO є критично важливим маркетинговим каналом. По-перше, і головне: звичайний пошук забезпечує 53% усього трафіку веб-додатку. Це одна з важливих причин, чому прогнозується, що світова індустрія SEO досягне приголомшливих 122,11 мільярдів доларів США до 2028 року. SEO забезпечує реальні бізнес-результати для брендів, підприємств і організацій будь-якого розміру. Щоразу, коли люди хочуть кудись піти, щось зробити, знайти інформацію, дослідити або купити продукт/послугу, їхня подорож зазвичай починається з пошуку в мережі Інтернет. Але сьогодні пошук неймовірно фрагментований. Користувачі можуть здійснювати пошук у традиційних пошукових системах (наприклад Google, Microsoft Edge або Safari), соціальних платформах (наприклад YouTube, TikTok, Instagram) або веб-сайтах роздрібних продавців (наприклад, Rozetka, Amazon, eBay,

Walmart). Аналізу, який був проведений в Сполучених Штатах Америки на прикінці 2022 року, наглядно зображено на рисунку 1.5. Зазначу, що в Україні ситуація приблзно однакова з усіма відомими інтернет магазинами та соціальними мережами, але деякі, такі як Walmart в нашому випадку замінюються на Rozetka та інші популярні інтернет магазини роздрібної торгівлі.

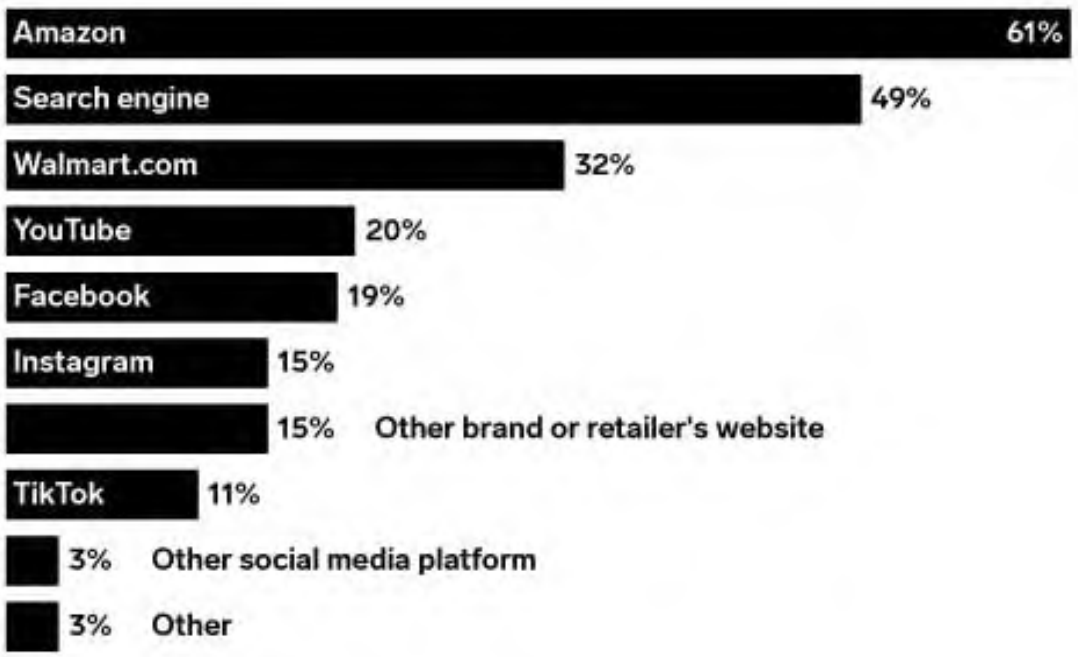


Рис. 1.5 Аналіз пошукових сервісів користувачів

Щороку проводяться трильйони пошуків. Пошук часто є основним джерелом трафіку для веб-сайтів, тому важливо бути «зручним для пошукових систем» та SEO-оптимізованим на будь-якій платформі, де люди можуть шукати ваш веб-додаток. Усе це означає, що покращення вашої видимості та вищі позиції в результатах пошуку, ніж ваші конкуренти, може позитивно вплинути на ваш прибуток. SEO також неймовірно важливе, оскільки сторінки результатів пошукових систем (або SERP) є суперконкурентними – вони наповнені функціями пошуку а також PPC-оголошеннями. Функції SERP включають: панелі знань, рекомендовані фрагменти, карти, зображення, відео, головні новини, «що цікавить людей?», тощо.

Пропоную більш детально розглянути поняття SEO-оптимізації. Отже, візьмемо наприклад інтернет магазин Rozetka. Щоб знайти його, користувач пише

в Google Chrome запит «інтернет магазин», «розетка», «магазин техніки», «Rozetka», тощо. Інтернет магазин Rozetka являє собою один з найпопулярніших магазинів техніки і інших товарів в Україні з багаторічни досвідом та гарною репутацією. Кожного тижня цей магазин збирає більше 100 тисяч кліків та запитів в свій адрес. Простіше кажучи, ці та інші фактори допомогли «Розетці» заробити добру репутацію в пошукових системах, що допомогло їм протягом багатьох років займати перші позиції в пошукових системах. Цей веб-застосунок накопичив сигнали, які демонструють, що він авторитетний і заслуговує довіри – і тому заслуговує на рейтинг, коли хтось шукає в мережі інтернет за допомогою подібних запитів. Загалом SEO працює завдяки поєднанню:

— Розробник: особа або команда, відповідальна за виконання або забезпечення завершення стратегічної, тактичної та оперативної роботи з оптимізації пошукових систем;

— Процесів: дії, вжиті для підвищення ефективності роботи;

— Технологій: використовувані платформи та інструменти;

— Діяльність: кінцевий продукт або вихід.

Багато інших факторів впливають на роботу SEO. Розглянемо чотири основних з них:

а) Розуміння того, як працюють пошукові системи. Простіше кажучи, якщо розробрник хоче, щоб люди знаходили його веб-додаток та компанію за допомогою пошуку на будь-якій платформі – йому потрібно розуміти технічні процеси, що стоять за роботою механізму, а потім переконатися, що він, як розробник, надаєте всі правильні «сигнали», щоб вплинути на видимість. Якщо говорити про традиційні веб-пошукові системи, такі як Google, існує чотири окремі етапи пошуку, що будуть наведені в табл. 1.4.

Таблиця 1.4

Етапи пошуку в Google

Сканування:	Пошукові системи використовують сканери, щоб знаходити сторінки в Інтернеті, переходячи за посиланнями та використовуючи карти сайту.
Візуалізація:	Пошукові системи визначають, як виглядатиме сторінка, використовуючи інформацію HTML, JavaScript і CSS.
Індексація:	Пошукові системи аналізують вміст і метадані сторінок, які вони виявили, і додають їх до бази даних (хоча немає гарантії, що кожна сторінка вашого сайту буде проіндексована).
Ранжування:	Складні алгоритми розглядають різноманітні сигнали, щоб визначити, чи є сторінка релевантною та достатньо високою якістю для відображення, коли шукачі вводять запит.

Але важливо зазначити, що оптимізація для пошуку Google відрізняється від оптимізації для пошуку на інших платформах, таких як YouTube або Amazon. Візьмемо, наприклад, Facebook, де важливі такі фактори, як залучення (лайки, коментарі, поширення тощо) і те, з ким люди пов’язані. Крім того, у Twitter важливі такі сигнали, як нещодавність, взаємодія чи довіра до автора.

б) Дослідження. Дослідження є ключовою частиною SEO. Деякі форми досліджень, які покращать ефективність SEO, включають:

1) Дослідження аудиторії: важливо розуміти свою цільову аудиторію або ринок. Хто вони, тобто їх демографічні та психографічні характеристики? На які питання розробник та його додаток можуть відповісти?

2) Дослідження ключових слів: цей процес допомагає розробнику визначити та включити релевантні та цінні пошукові терміни, які люди використовують, на сторінки додатку, а також зрозуміти, наскільки високий попит і конкуренція є для рейтингу цих ключових слів.

3) Дослідження конкурентів: що роблять конкуренти? Які їхні сильні та слабкі сторони? Які типи вмісту вони публікують?

4) Дослідження бренду/бізнесу/клієнтів: які їхні цілі та як ваш веб-додаток може допомогти їм досягти цих цілей?

5) Дослідження веб-додатку: різноманітні аудити SEO можуть виявити можливості та проблеми на веб-сайті, які перешкоджають успіху в звичайному пошуку. Деякі аудити, які слід розглянути: технічний SEO, контент, профіль посилань і інші.

6) Аналіз SERP: це допоможе розробнику зрозуміти мету пошуку для даного запиту (наприклад, чи є він комерційним, транзакційним, інформаційним чи навігаційним) і створити вміст, який з більшою ймовірністю заробить рейтинг або видимість.

в) Планування. Стратегія SEO — це довгостроковий план дій. Розробнику необхідно встановити цілий план того, як він їх досягатиме. Перероблення стратегії SEO як дорожньої карти. Шлях, який вибере розробник, ймовірно, з часом буде змінюватися і розвиватися, але пункт призначення повинен залишатися ясним і незмінним. План SEO може включати в себе такі речі, як: Встановлення цілей і очікувань; Визначення та узгодження значущих показників; Вирішення способу створення та реалізації проектів; Координація та спілкування з внутрішніми та зовнішніми зацікавленими сторонами; Вибір і впровадження інструментів/технологій; Наймання, навчання та структурування команди; Встановлення бюджету; Вимірювання та звітування про результати; Документування стратегії та процесу.

Коли всі дослідження завершені, настав час втілювати ідеї в дії. Це означає, що потрібно створювати новий контент та консультуватися командою розробників який контент потрібно створити. Рекомендація або впровадження змін або вдосконалень на існуючих сторінках. Це може включати: оновлення та вдосконалення вмісту, додавання внутрішніх посилань, включення ключових слів/тем/сутностей або визначення інших способів його подальшої оптимізації. Також, варто не забувати про видалення старого, застарілого або низькоякісного вмісту: типи вмісту, які погано позиціонуються, спонукають конвертувати трафік [27].

г) Моніторинг і обслуговування. Розробник повинен знати, коли в його веб-додатку щось йде не так або ламається. Моніторинг є критичним. Йому потрібно знати, якщо трафік на критичну сторінку падає, сторінки стають повільними, не реагують або випадають з індексу, весь веб-додаток переходить у режим роботи, посилання перериваються чи будь-які інші потенційно катастрофічні проблеми. Очевидно, що аналіз веб-додатку допоможе більш конкретно виявляти небажані фрагменти. Щоб приймати рішення щодо оптимізації пошукових систем на основі даних, розробник має використовувати пенві чинники, що наведені в табл. 1.5.

Таблиця 1.5

Чинники SEO-оптимізації

Аналітика веб-додатку:	Налаштування та використання інструментів таких як, Google Analytics, Google Search Console і Bing Webmaster Tools, для збору даних про продуктивність.
Інструменти та платформи:	Існує багато комплексних платформ або пакетів, які пропонують кілька інструментів, але розробник також може використовувати лише вибрані інструменти SEO для відстеження ефективності виконання конкретних завдань. Або, якщо у вас є ресурси, і жоден із інструментів на ринку не робить саме того, що потрібно, ви можете створити власні інструменти.
Звітування:	Після того, як будуть зібрані дані, потрібно звітувати про прогрес. Звіти можна створювати програмно або вручну. Звіт про ефективність повинен розповідати історію та складатися через значущі проміжки часу, як правило, у порівнянні з попередніми звітними періодами (наприклад, рік за роком). Це залежатиме від типу веб-сайту (як правило, це буде щомісяця, щокварталу чи інший інтервал).

Ще одна причина, чому SEO має вирішальне значення для брендів і компаній: на відміну від інших маркетингових каналів – хороша робота SEO є стабільною. Коли платна оптимізація (PPC) закінчується, трафік також закінчується. Трафік із соціальних мереж у кращому випадку ненадійний – і лише маленька частина того, яким він був раніше. SEO є основою цілісного маркетингу, де все, що робить ваша

компанія, має значення та актуальність в сучасному світі. Коли ви зрозумієте, чого хочуть ваші користувачі, ви зможете застосувати ці знання в:

— Оптимізації (SEO або PPC);

— Вмісті та структурі веб-додатку.

SEO також зміцнює довіру – веб-додаток або сайт із високим рейтингом зазвичай вважається авторитетним або заслуговує на довіру, що є ключовими елементами, які Google хоче винагородити кращими рейтингами. Існує три «стовпи», за допомогою яких SEO-оптимізація набуває своєї сили та потужності:

а) Технічна SEO: оптимізація технічних аспектів веб-додатку.

Оптимізація технічних елементів веб-додатку є важливою та фундаментальною для успіху SEO. Вона виконується на бекенд частині проекту. Все починається з архітектури – створення веб-застосунку, який можна сканувати та індексувати пошуковими системами. Розробнику потрібно полегшити пошуковим системам пошук і доступ до всього вмісту сторінок та контенту веб-додатку (тобто тексту, зображень, відео). Які технічні елементи тут важливі: структура URL-адреси, навігація, внутрішні посилання тощо. Досвід також є важливим елементом технічної оптимізації. Пошукові системи наголошують на важливості сторінок, які швидко завантажуються та забезпечують хорошу взаємодію з користувачем. Такі елементи, як Core Web Vitals (індекс швидоксті загрузки веб-додатку або веб-сайту), зручність для мобільних пристроїв і зручність використання, HTTPS і уникнення нав’язливих міжсторінкових оголошень – усе це має значення для технічного SEO [28].

Важлива сфера технічної оптимізації — структуровані дані (також відомі як схема). Додавання цього коду в веб-додаток може допомогти пошуковим системам краще зрозуміти його вміст і покращити вигляд застосунку у результатах пошуку. Крім того, послуги веб-хостингу і безпека сайту відіграють важливу роль у SEO. Структура URL-адреси також впливає на технічну складову. Організована структура додатку, як використання таких сегментів: /blog, /landing page, /product, полегшує Google сканування вашого сайту, користувачам – навігацію по ньому, а розробнику – аналіз та сегментацію даних у звітах. Архітектура веб-додатку, одна

з ключових складових: в ідеалі користувач повинен мати доступ до будь-якої сторінки веб-доадтку за три кліки або менше. Наведена структура зображена на рисунку 1.6. Ключовим тут є внутрішнє зв’язування.

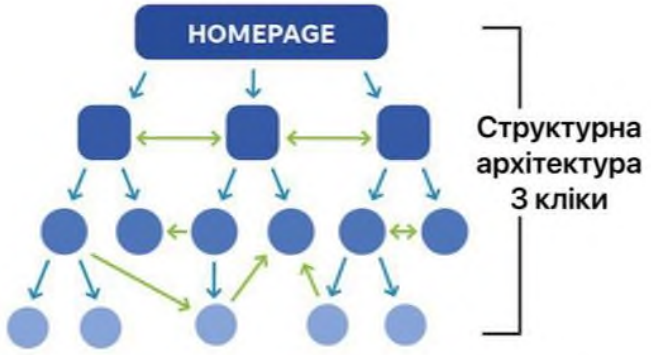


Рис. 1.6 Бажана SEO структура застосунку

Канонічні URL-адреси: канонічна URL-адреса – це URL-адреса, яку розробник хочете представити набору повторюваних сторінок. Google докладе всіх зусиль, щоб визначити канонічну URL-адресу для будь-якого заданого набору дублікатів, але разробник також може вказати це Google за допомогою канонічних тегів або переспрямувань 301. Наприклад:

—http://tepliukmagwork.com;

—http://tepliukmagwork.com/;

—https://www.tepliukmagwork.com;

—https://www.tepliukmagwork.com/;

—https://www.tepliukmagwork.com/.

Усі переспрямування на цю канонічну URL-адресу: https://www.tepliukmagwork.com

Розмітка схеми: розмітка схеми допомагає Google (та іншим пошуковим системам) зрозуміти типи вмісту веб-застосунку, дозволяючи відображати розширені результати, коли це можливо. Наприклад, схема посилання на сайт може надати більше можливостей в пошуковій видачі, приклад буде зображено на рисунку 1.7.

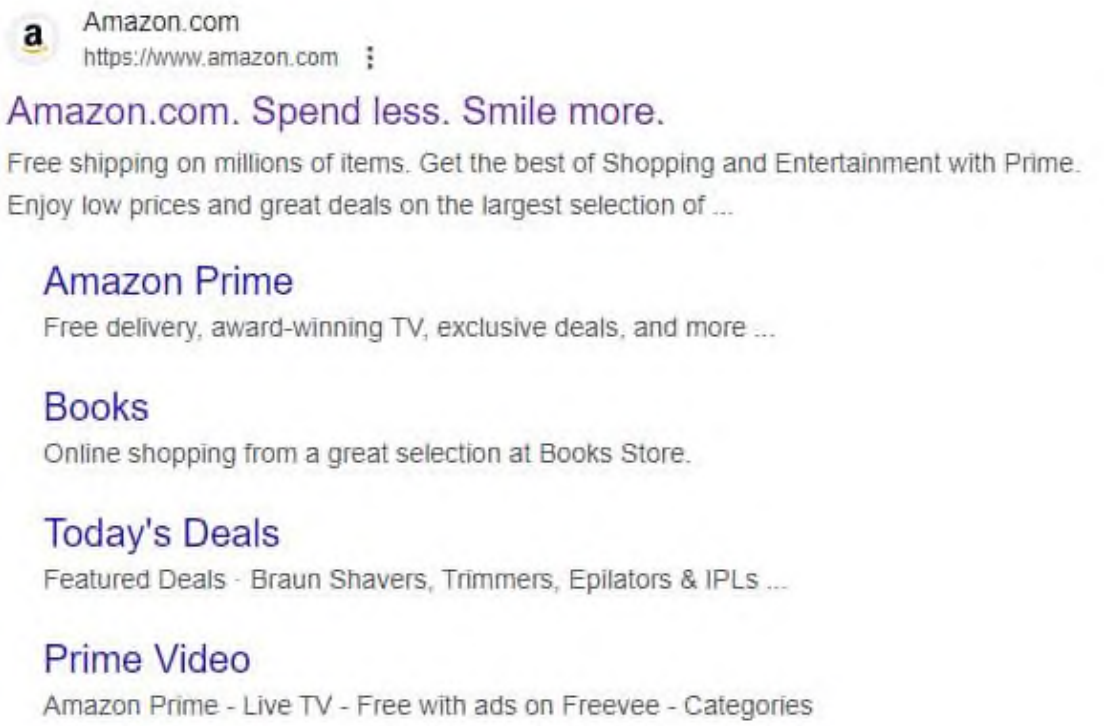


Рис. 1.7 Розширені результати пошуку

б) SEO оптимізація контенту.

У SEO контент потрібно оптимізувати для двох основних аудиторій: людей і пошукових систем. Це означає, що розробник оптимізуєте вміст, який будуть бачити користувачі (те, що насправді є на сторінці), а також те, що бачитимуть пошукові системи (код бекенду). Метою завжди є публікація корисного високоякісного вмісту. Розробник можете зробити це завдяки поєднанню розуміння бажань і потреб користувачів, даних і вказівок від Google. Оптимізуючи контент для людей, розробник повинен переконатися, що інформація:

— Охоплює відповідні теми, з якими у вас є досвід або знання;

— Включає ключові слова, які люди використовували б, щоб знайти вміст;

— Є унікальним або оригінальним;

— Написаний добре, без граматичних і орфографічних помилок;

— Є актуальною, містить точну інформацію;

— Включає мультимедіа (наприклад, зображення, відео);

— Краще, ніж мають конкуренти в SERP;

— Читабельний – структурований так, щоб людям було легко зрозуміти інформацію, якою розробник ділиться (підзаголовки, довжина абзацу, жирний/курсив, упорядковані/невпорядковані списки, рівень читання тощо).

Для пошукових систем деякі ключові елементи вмісту, для яких потрібно оптимізувати: теги заголовків, метаопис, теги заголовків (H1-H6), альтернативний текст зображення, відкрити графіки і метадані.

c) SEO зовнішня оптимізація.

Є кілька видів діяльності, які можуть не називатися «SEO» у строгому значенні, але, тим не менш, можуть узгоджуватися з успіхом SEO та опосередковано сприяти цьому. Побудова посилань (процес отримання посилань на веб-сайт) – це діяльність, яка найбільше пов’язана з пошуковою оптимізацією за межами сайту. Отримання різноманітних посилань на веб-застосунок із релевантних, авторитетних і надійних веб-сайтів може принести великі переваги (наприклад, рейтинги, трафік). Якість посилань перевершує кількість посилань, а велика кількість якісних посилань є метою. Існує безліч методів просування веб-сайтів, які поєднуються з зусиллями SEO. До них належать:

— Розбудова бренду та маркетинг бренду (методи, призначені для підвищення впізнаваності та репутації);

— Контент-маркетинг: Деякі популярні форми включають створення відео, електронних книг, дослідження, подкасти (або участь в інших подкастах) і гостьові публікації (або гостьові блоги);

— Маркетинг і оптимізація в соціальних мережах: представлення свого бренду на будь-якій і всіх відповідних платформах, його повна оптимізація та відповідний вміст;

— Управління лістингом: заявка, перевірка та оптимізація інформації на будь-яких платформах, де інформація про веб-додаток може бути перерахована та знайдена пошуковими користувачами (наприклад, каталоги, сайти оглядів, вікі);

— Оцінки та відгуки: їх отримання, моніторинг і реагування на них.

Загалом, говорячи про зовнішню оптимізацію, мова йде про діяльність, яка не матиме прямого впливу на здатність ранжуватись розробником із чисто технічної точки зору. Однак знову ж таки, усе, що робить бренд, має значення. Розробник хоче, щоб його бренд був знайдений скрізь, де люди можуть його шукати. Таким чином, деякі люди намагалися перейменувати «оптимізацію пошукової системи» на «оптимізацію пошуку» або «оптимізацію пошуку всюди».

Підбиваючи висновки з приводу SEO-оптимізації, можна затвердити що ця технологія одна з найголовніших, що застосовуються в сучасній веб-розробці. Будь-якия сучасна компанія, різних розмірів, має свій веб-додаток або сайт та займається його SEO-оптимізацією тим чи іншим способом. Якщо питання постає, що ж вибрати SEO або PPC, то краще вибирати обидва варінти. Оскільки, вони доповнюють один одного та надають більших результатів. Однак, особисто на мою думку, саме SEO-оптимізація все ж таки вириваєтся вперед. Тому що, це гарантований результат, який надасть свої плоди в майбутньому і ці плоди ніколи не зіпсуються. Пошукові системи, поведінка користувачів і ваші конкуренти постійно змінюються. Веб-сайти змінюються та їх вміст застаріває. Процеси оптимізації мають покращитися та стати ефективнішими. Завжди є те, що розробник можете контролювати, тестувати чи вдосконалювати. Або, як сказав Брюс Клей (засновник і призидент компанії Bruce Clay Inc., глобального агенства з SEO-оптимізації): «Пошукова оптимізація «помре» лише тоді, коли Google припинить щось змінювати, а ваша конкуренція зникне.». Все залежить від бюджету, можливостей та навичок розробника. SEO-оптимізація ніколи не стане застарілою технологією.

Інтернет речей та веб-розробка є двома ключовими напрямками, які взаємодіють між собою та набирають нових обертів в сучасному світі, що сприяє розвитку інноваційних рішень у багатьох галузях та сферах. Це передбачає розвиток та впровадження певних аспектів і новтніх тенденцій у майбутньому:

— Інтерфейс та взаємодія: Розробка зручних та зрозумілих веб-додатків та інтерфейсів для керування та моніторингу IoT-пристроїв;

— Розробка мобільних додатків для взаємодії з IoT-пристроями через смартфони

та інші мобільні пристрої;
— Створення систем аналітики та візуалізації даних для зручного представлення інформації зібраної IoT-пристроями;
— Використання хмарних технологій для розгортання та масштабування веб-додатків пов’язаних з IoT;
— Забезпечення безпеки веб-додатків, які взаємодіють з IoT-пристроями, враховуючи потенційні ризики та загрози;
— Штучний інтелект (AI) та машинне навчання (ML): Інтеграція із системами штучного інтелекту для вдосконалення аналітики та автоматизації прийняття рішень на основі забраних даних;
— 5G технології: Використання 5G для забезпечення швидкого та надійного підключення для великої кількості IoT-пристроїв.
— Більше уваги до екологічних рішень: Розробка технологій спрямованих на зменшення впливу IoT на довкілля та використання енергоефективних рішень.

РОЗДІЛ 2. АНАЛІЗ, ДОСЛІДЖЕННЯ І ВИБІР СЕРЕДОВИЩА РОЗРОБКИ LARAVEL ТА БАЗ ДАНИХ MYSQL

За допомогою певних бібліотек та фреймворків можна розробляти найсучасніші веб-додатки для клієнтів. Для програмування на PHP використовують такі найпопулярніші фреймворки як Laravel та Symfony. Вони надають структуру та певні рішення, для швидкого розгортання проектів. Це поліпшує розробку, та надає можливість програмістам не виконувати рутинні дії. В цьому розділі я проведу аналіз та закріплю, чому саме Ларавел є кращим фреймворком для розробки веб-застосунків. Також, буде проведено аналіз переваг Ларавел та кращих його пакетів технологій для програмування на PHP.

В минулому розділі я проаналізував сутність поняття веб-додатку та показав різницю між веб-застосунком та сайтом. Як відомо, для розробки веб-сайтів найчастіше використовують CMS програми та HTML-верстування фронтенду. В той час, як веб-застосунки розробляються за допомогою саме фреймворків та програмування на PHP. Важливо розуміти що саме дозволяє фреймворк при розробці, і чого просто технічно немає в CMS.

2.1 Різния між CMS та Фреймворком

Основна відмінність між CMS і Framework полягає в тому, що CMS – це система, яка створює та обробляє веб-контент, а Framework – це програма, яка включає стандартизований інтерфейс, що можна змінювати залежно від клієнта за допомогою спеціального коду, написаного користувачем. І CMS, і Framework є прикладними програмами. CMS – це система керування вмістом, а фреймворк – це модульне середовище, яке є частиною ширшого набору програмного забезпечення.

Загалом, розробку веб-продуктів на основі CMS можна порівняти зі складанням меблів із кількох готових елементів. Тим часом створення веб-рішень за допомогою фреймворку схоже на замовлення нестандартних елементів, розроблених відповідно до ваших конкретних вимог. В останньому випадку теслі будуть використовувати деякі дерев'яні матеріали, отримані від виробника, а не

виготовляти їх із сирих стовбурів дерев. Але ви все одно отримаєте меблі, виготовлені на замовлення.

Перш ніж почати проект веб-розробки, бізнес-менеджери розробники та власники компанії повинні вирішити, чи буде майбутнє рішення базуватися на CMS або фреймворку. Цей вибір має значний вплив на подальший процес розробки та кінцевий продукт. Він визначає, скільки часу знадобиться програмістам для завершення роботи, який бюджет власник витратить, які характеристики та можливості матиме веб-рішення [14]. Отже, життєво важливо зрозуміти, чим відрізняються ці варіанти та які наслідки приносить кожен із них. Базову різницю між цими поняттями зображено на табл. 2.1. Більш детально я досліджу поняття CMS та Фреймворку далі в цьому розділі.

Таблиця 2.1

Базова різниця між CMS та Framework

CMS	Framework
Використовується для створення та зміни вмісту.	Містить стандартизований інтерфейс, який можна змінити індивідуальним кодом, написаним розробником, на основі певних вимог.
Легше у використанні.	Фреймворк важче в освоєнні не кваліфікаційному спеціалісту
Допомагає керувати контентом.	Допомагає координувати код і робить процес створення програмного забезпечення простішим і масштабованішим.
Зазвичай CMS має багато обмежень у використанні і розробці.	Фреймворк немає обмежень і все залежить від розробника.

2.2.1 Фреймворк (Framework)

Фреймворк (англ. Framework) — це набір кодів, які використовуються як основа для створення веб-додатків. На відміну від CMS, він не має «готових до використання» інструментів для керування та оновлення вмісту. Однак фреймворк впроваджується разом зі «загальними кодами», які служать будівельними блоками для розробки вашого веб-сайту, що робить його відносно простішим варіантом порівняно з кодуванням повністю з нуля. Фреймворки також можна розширити за

допомогою «бібліотек», які є наборами функцій, встановлених разом із фреймворками, для створення нових функціональних можливостей шляхом розширення основних функцій фреймворку подібним чином, як плагіни для CMS.

Фреймворки дозволяють створити веб-доадток, налаштований відповідно до ваших потреб. Хоча це вимагатиме багато роботи, ви отримуєте можливість розробити систему на 100% так, як ви цього хочете – із функціями та функціями, яких немає в стандартній CMS. Сьогодні існує велика різноманітність веб-фреймворків, написаних різними мовами програмування. Найпоширеніші приклади включають:

— Microsoft пропонує .NET Framework. Він підтримує мови програмування C# і Visual Basic. Інфраструктура розробки програмного забезпечення .NET в основному використовується для створення та запуску програм на серверах Windows;

— CodeIgniter — це фреймворк PHP, відомий своєю невеликою площею та безпроблемністю. Початківці вважають за краще використовувати його для створення легких додатків;

— Laravel також є фреймворком PHP, якому розробники віддають перевагу через його розгалужену екосистему. Це фреймворк PHP із повним стеком, який використовує менше ресурсів і споживає менше пам’яті, ніж інші фреймворки;

— Symfony є першим вибором для багатьох розробників, особливо при створенні великомасштабного програмного забезпечення. Фреймворк PHP існує довше, ніж більшість інших фреймворків, і відомий своєю великою кількістю багаторазових компонентів і бібліотек.

Фреймворк, з іншого боку, – це набір попередньо написаних фрагментів коду, що можна багаторазово використовувати та які охоплюють деякі загальні функції. Використання фреймворку допомагає розробникам програмного забезпечення економити час на виконанні типових завдань програмування, оскільки їм не потрібно створювати всі функції з нуля. Фреймворк забезпечує стандартну форму створення та розгортання програми. Він містить усі файли зі стандартизованою функціональністю. Відповідно до вимог розробник може їх змінити та створити. У

використанні фреймворка також є багато переваг. Структура забезпечує методологію організації коду. Це також покращує повторне використання коду. Вимоги до програмного забезпечення можуть змінитися в будь-який час, але зміни можна легко внести за допомогою інфраструктури. Також доступні готові та перевірені інструменти. Фреймворк містить багато компонентів/розділів. Таким чином, багатьом розробникам легше працювати над різними аспектами проекту.

Переваги фреймворків:

— Фреймворки є дуже гнучкими і можуть бути розроблені відповідно до будь-яких ваших вимог;
— Легкі в оновлені без втрати даних;
— Веб-додатки на основі Framework мають кращу продуктивність, ніж застосунки на основі CMS.

Недоліки фреймворків:

— Створення веб-сайтів із фреймворків – це складний процес. Щоб створити якісний веб-сайт, ви повинні володіти великою мовою програмування;
— Панелі адміністратора немає. Сегмент редагування сайту потрібно прописати окремо, а це, по суті, створення іншого сайту. Розробка веб-сайту через фреймворк є досить трудомістким процесом.

2.2.2 Система управління контентом (CMS)

Система керування вмістом або CMS — це програма, що містить функції та функції, які дозволяють легко керувати вмістом і публікувати його без підтримки веб-розробника. Стереотипна CMS має два основні елементи:

— Програма для керування вмістом (CMA): CMA дозволяє додавати, змінювати та видаляти вміст на вашому веб-сайті;
— Програма доставки вмісту (CDA): CDA компілює ваші дані в CMA, оновлює їх і доставляє на ваш веб-сайт для перегляду відвідувачами вашого сайту.

Приклади популярних систем керування вмістом включають:

— WordPress відповідає за приблизно 30% усіх веб-сайтів, які існують в Інтернеті. Це CMS на основі PHP і MySQL з відкритим кодом. Це одна з найпростіших і найлегших у використанні CMS;
— Magento — це CMS для електронної комерції з відкритим кодом на основі PHP. Його масштабованість і чудові функції роблять його дуже популярним серед веб-сайтів електронної комерції;
— Запущена в 2004 році Umbraco є однією з провідних систем CMS .NET framework. Його сумісна стратегія керування вмістом робить його дуже зручним для користувачів;
— Joomla є другою за популярністю CMS (після WordPress, звичайно). Як початківці, так і просунуті розробники можуть використовувати Joomla (хоча вона вважається більш складною, ніж WordPress). Його було розроблено з урахуванням високої продуктивності та хизується багатьма функціями, зручними для SEO-оптимізації;
— Drupal, як і Magento, є CMS з відкритим кодом. Здебільшого він використовується на сайтах соціальних публікацій із великим вмістом. Він пропонує широкі можливості для налаштування та має великий вибір модулів розширення.

По суті, CMS (або система керування контентом) — це тип програмного забезпечення, яке дозволяє користувачам створювати веб-рішення з набором готових будівельних блоків і керувати цифровим контентом за допомогою стандартної панелі адміністратора. Drupal, WordPress і Joomla є поширеними в серидовищі розробки на CMS. Завдяки своїй простоті та зручності використання вони досить популярні в суспільстві. Щоб створювати веб-додатки за допомогою системи керування контентом, користувачі повинні завантажити відповідне програмне забезпечення та налаштувати його відповідно до потреб свого бізнесу.

Основною метою використання CMS є керування вмістом веб-сайту. Він також містить функції для легкого керування веб-вмістом, таким як текст, відео, аудіо тощо. Розробники використовують CMS для різних цілей, наприклад для електронної комерції, блогів, веб-сайтів-каталогів, порталів новин, порталів роботи

та багато іншого. CMS допомагає виконувати різні операції. По-перше, користувач може легко форматувати макет, додавати, редагувати та видаляти вміст. Здебільшого він дозволяє публікувати веб-сторінки, створювати нові теми та використовувати вже існуючі теми, індексувати та здійснювати пошук. Отже, розглянемо їх переваги та недоліки.

Переваги CMS:

— Веб-сайти CMS можна розробити та запустити дуже швидко;
— Панель користувача/панель адміністратора на багатьох популярних CMS проста для розуміння та навігації;
— Системи керування вмістом пропонують низку плагінів і тем, які ви можете встановити, щоб швидко створити та розробити свій веб-сайт, тобто розробити їх легко. Вам не потрібно мати глибокі знання програмування або наймати будь-яких дорогих розробників програмного забезпечення, щоб створити ваш веб-сайт для вас. Просто завантажте шаблон CMS і створіть на його основі свій веб-сайт.

Недоліки CMS:

— Додавання користувацьких функцій і функціональних можливостей до CMS є дуже виснажливою роботою. Особливо, якщо плагіни ще не доступні. Його обмежені можливості налаштування є значною перешкодою для CMS;
— Розробники системи керування вмістом прагнуть створювати універсальні продукти з великою функціональністю. Однак багато з цих функцій є зайвими. Натомість вони витрачають ресурси, що негативно впливає на навантаження та швидкість роботи;
— Іноді, щоб виправити або змінити деякі функції сайту, потрібні глибокі знання про те, як працює CMS в цілому;
— Ефективність веб-додатку на основі CMS є більш низькою, ніж застосунок, побудований на фреймворку.

2.3 Основні аспекти різниці між CMS та Framework

Фреймворки і системи керування вмістом — два дуже різні інструменти. Щоб вирішити, який з них кращий, нам потрібно спочатку розглянути їх відмінності та

ключові фактори, які слід враховувати. Важливе базове розуміння того, чим системи керування вмістом відрізняються від фреймворків веб-розробки. Але для багатьох власників бізнесу набагато важливіше знати, як ця різниця впливає на процес розробки та основні якості веб-рішення. Отже, давайте розглянемо основні аспекти, на які вам слід звернути увагу, вибираючи між розробкою за допомогою фреймворку та CMS.

2.3.1 Гнучкість фреймворків

Рішення на основі CMS мають деякі функції за замовчуванням, які можна налаштувати та розширити за допомогою різноманітних доповнень. Наприклад, офіційний каталог плагінів WordPress пропонує понад 58 000 плагінів, які користувачі можуть встановити на додаток до основних функцій. Проте кількість потенційних модифікацій завжди обмежена наявними можливостями CMS. Це означає, що розробник може створити робочий веб-продукт за допомогою CMS. Але він зможе налаштувати його під свої потреби лише до певної межі. Іншими словами, такий тип розробки не дозволить реалізувати вимоги, які виходять за рамки обраної CMS та її розширень.

Вибір та рішення розробки веб-додатку на PHP за допомогою фреймворку, наділяє розробника можливості включити в веб-продукт будь-яку функціональність. Хоча розробники використовуватимуть деякі попередньо написані фрагменти коду, вони зможуть змінювати їх за потреби без жодних обмежень. Крім того, фреймворк – це лише основа для веб-продукту. Щоб створити деякі унікальні функції, програмісти напишуть абсолютно новий власний код. Отже, розробникам належить визначити вимоги до якості та кількості функцій, рівня масштабованості продукту та інтеграції сторонніх розробників. З CMS ви отримуєте попередньо визначений набір функцій, які можна додатково налаштувати, завантажуючи та встановлюючи різні плагіни. Фреймворки, з іншого боку, повністю настроюються. Ви повинні будувати та проектувати все з нуля. Хоча код, який ви використовуєте, забезпечує базу для створення, нічого не

визначено заздалегідь. Отже, ви можете формувати та формувати його відповідно до ваших вимог, незалежно від того, наскільки вони специфічні чи незвичайні.

З CMS ви маєте обмежені можливості налаштування та не можете змінити основні функції, в той час як з фреймворком таких перешкод немає.

2.3.2 Безпека та оновленість

Безпека – є однією з найбільших проблем для будь-кого, хто розробляє програму або веб-сайт, особливо якщо ви власник онлайн-бізнесу, якому довіряєте конфіденційну інформацію своїх клієнтів. Це ще один критерій, який ви повинні враховувати, вирішуючи, чи вибрати CMS або фреймворк для веб-розробки свого проекту. Веб-рішення на основі CMS є більш уразливими до кібератак і більш схильні до ризику витоку даних. Причина в тому, що багато популярних систем керування контентом є відкритими, тобто їхній код є загальнодоступним. Звичайно, хакерам легше знайти шляхи доступу до рішення, якщо вони знають, як воно побудоване. Розширення або їх оновлення, завантажені з ненадійних ресурсів, також можуть поставити під загрозу безпеку рівня рівня.

Веб-продукти на основі фреймворків важче атакувати, оскільки стороннім сторонам важче знайти в них недоліки безпеки. Крім того, більшість сучасних фреймворків забезпечують набір вбудованих функцій безпеки, які захищають рішення від найпоширеніших кіберзагроз. Наприклад, Laravel пропонує кілька пакетів безпеки, які включають захист CSRF, шифрування даних, захист від впровадження XSS і SQL тощо. Добре розроблений фреймворк є набагато безпечнішим, ніж звичайна CMS, але системи керування вмістом часто мають плагіни та модифікації для посилення їх безпеки.

Щоб забезпечити безперебійну роботу вашого веб-додатку, він завжди має бути оновленим, мати найновіші функції та не мати помилок. Регулярне оновлення веб-сайту також захищає його від хакерів та інших потенційних загроз безпеці.

Веб-додатки розроблені за допомогою фреймворків потребують постійного обслуговування, в той час як CMS можуть мати регулярні оновлення. Веб-додаток

на основі фреймворку надійно захищений під час створення та часто не потребує жодних оновлень для роботи. Але якщо ви хочете постійно оновлювати та додавати нові функції у свій веб-додаток з інших причин, ніж просто безпека, CMS — не буде найкращим варіантом. Оновити CMS відносно просто. Вам навіть не потрібно знати PHP або будь-які інші мови програмування, щоб оновити CMS. Але це спричинить певні зміни та зброси до первинних налагоджень вашого CMS рішення. В той час, як фреймворк може потребувати трохи більшого часу для оновлення, але всі розробки будуть збережні і програмістам не доведеться виконувати пенві фрагменти проекту знов.

2.3.3 Час розробки та бюджет

Створення веб-рішення на основі CMS дешевше та потребує менше часу, ніж розробка на замовлення. Процес встановлення CMS зазвичай такий же простий, як і встановлення звичайних веб-додатків. Можна навіть створити та налаштувати базовий веб-сайт за лічені дні. Оскільки більшість CMS орієнтовані на людей без технічної підготовки, багато завдань на етапах розробки та супроводу можна виконати без участі програмістів. Тим не менш, настійно рекомендується отримати допомогу від професійних розробників, якщо ви хочете створити більш складне рішення або внести деякі додаткові налаштування (наприклад, розширену оптимізацію SEO).

Використання фреймворків для веб-проекту завжди вимагає глибоких знань програмування та сильних навичок програмування. Крім того, цей вид розробки досить тривалий. Хоча IT-команда може використовувати деякі заздалегідь прописані функції, розробникам потрібно зробити значну частину роботи самостійно. Збільшений час розробки природно призводить до вищих витрат на розробку. Ось чому веб-проекти на основі фреймворків набагато дорожчі за проекти на основі CMS. Оновлення (тобто додавання нових функцій) і підтримка таких продуктів також вимагатиме більше фінансових ресурсів.

Бюджет завжди є важливим фактором при прийнятті будь-яких важливих рішень щодо веб-розробки. CMS значно дешевше, ніж Frameworks. Вища можливість налаштування має вищу ціну.

2.3.4 Адаптивність та управління контентом

Веб-додатки, створені за допомогою фреймворків, як правило, більш масштабовані, ніж CMS. Це тому, що під час використання Frameworks ви не обмежені попередньо встановленими темами, плагінами та архітектурою. Отже, якщо вам потрібен гнучкий додаток, який потребує унікальних функцій і має потенціал для значного зростання, скористайтеся фреймворками.

CMS надають стандартні функції для керування вмістом. Ці системи постачаються з готовими до використання панелями адміністратора, які дозволяють користувачам легко змінювати інформацію на веб-сайті. Додавання, видалення та редагування вмісту зазвичай не є проблемою навіть для новачків. Якщо ви хочете отримати більш чітке уявлення про те, як все працює, подивіться на інформаційну панель WordPress. Скажімо, вам потрібно опублікувати нову публікацію в блозі. Для цього вам потрібно просто вибрати вкладку «Публікації» на лівій бічній панелі та натиснути «Додати новий». Після заповнення відповідних полів (наприклад, заголовка, заголовків, основного тексту) вам потрібно буде натиснути кнопку «Опублікувати», і публікація з’явиться на вашому сайті.

Фреймворки не дозволяють напряму користувачам критично керувати вмістом — вони допомагають розробникам створювати рішення (тобто спеціальні панелі адміністратора), які можуть містити функції керування вмістом. Користувачі можуть вносити певні зміни у веб-застосунок, які підкріплює собою притаманна йому інтерактивність, але критично змінювати (наприклад видаляти користувачів та сторінки) вони не можуть. Якщо докласти достатньо зусиль до цього процесу, можна створити спеціальну панель адміністратора, яка буде такою ж зручною для користувача, як готові CMS або навіть краще. Але це практично цілий підпроект у вашому проекті веб-розробки. Крім того, менеджери вмісту

можуть редагувати інформацію у веб-додатках на основі фреймворку, вносячи зміни безпосередньо у файли застосунку. Але цей варіант вимагає хороших навичок програмування, тому він не підходить, якщо ви плануєте запустити веб-сайт із інтенсивним вмістом.

Тип веб-застосунку, який ви створюєте, і чому ви його створюєте, визначає, чи може він потребувати будь-яких унікальних функцій і функцій, відмінних від вашого типового проекту веб-розробки. Приклади унікальних функцій включають інтеграцію програми третьої сторони або поєднання з незалежними системами. Наприклад, великий інструмент електронної комерції, можливо, доведеться інтегрувати в інструмент CRM (Customer Relationship Management).

Хоча CMS може адаптуватися, щоб включити багато користувацьких функцій і функціональних можливостей за допомогою плагінів, вона не настільки адаптована, як Framework. Якщо ви хочете включити користувацькі функції в CMS, які недоступні в бібліотеках плагінів і тем, вам доведеться залучити аутсорсинг, щоб створити її на замовлення. Цей процес досить складний. Ви можете налаштувати фреймворк відповідно до своїх потреб. Однак ця адаптивність має свою ціну – буквально. Створення фреймворків набагато дорожче, ніж CMS.

2.4 Висновки до аналізу CMS та фреймворків

Обидва види розвитку мають свої переваги і недоліки. Якщо ви виберете варіант CMS, створення веб-рішення потребуватиме менше часу та грошей, але ви будете обмежені можливостями CMS. У той же час розробка на основі фреймворку дозволить вам створити багатофункціональне рішення, яке точно відповідає вимогам вашого бізнесу. Але вартість проекту буде значно вищою, а програмістам знадобиться більше часу, щоб доставити вам кінцевий продукт. Якщо веб-сайт, який розробник намагаєтеся створити, відносно простий і не потребує жодних складних чи унікальних функцій, вибір паде на CMS. Наприклад, CMS, така як WordPress, може безперешкодно працювати з блогом, невеликим веб-сайтом електронної комерції. Тим не менш, якщо потрібен веб-додаток із конкретними та детальними вимогами, вибір фреймворку — найкращий спосіб. Наприклад, якщо

очікується постійна величезна кількість користувачів у веб-додатку одночасно, вибір буде за Laravel. Великий обсяг трафіку перевантажить CMS, в той час як фреймворк має свої певні рішення щодо обробки великого трафіку користувачів.

З проведеного мною аналізу різниці між CMS та фреймворками, я зробив висновок, що як для мене – розробка за допомогою фреймворків є більш професійною та багато функціональною. Звісно, якщо проект не потребує серйозної реалізації, можливо обійтись і CMS рішеннями. Але за допомогою фреймворку, я зможу надати навіть простому проекту своїх певних родзинок та функціоналу, що зробить його більш зручнішим та оптимізованішим, ніж би цей проект був розроблений на CMS. Фреймворк – це про унікальні вимоги і унікальні рішення для проекту, багатофункціональність та міцна безпека. І все це має можливість вдосконалення. Розробка за допомогою фреймворку, на мою думку, є синонімом зі словом «професійність». Саме тому я обрав фреймворк Laravel для розробки на PHP. Він включає в себе безліч аспектів та функціоналу, завдяки якому програмування веб-застосунків перевторюється в певне мистестцто для розробників.

2.5 Дослідження фреймворку Laravel та його особливості

Laravel — це простий у використанні веб-фреймворк, який допоможе створювати розширювані веб-сайти та веб-додатки на основі PHP у масштабі. Перш ніж створювати веб-додаток або веб-сайт, розробнику потрібно прийняти фундаментальне рішення щодо того, яку технологію він збирається використовувати. Це одна з найскладніших частин процесу веб-розробки. Щоб створити щось просте, наприклад інтернет-блог або портфоліо, розробник може покластися на технології розробки веб-сайтів без коду. Якщо розробник хоче створити щось більш просунуте, рішення без коду може бути недостатньо. Натомість слід вибрати фреймворк і почати писати на ньому код. Laravel є хорошим вибором як простий у використанні фреймворк із відкритим вихідним кодом для створення сучасних веб-додатків у масштабі.

Перший стабільний випуск Laravel відбувся в 2011 році, але не привернув особливої уваги. Коли Laravel 3 вийшов у 2012 році, він почав набирати популярності та став найшвидше зростаючим фреймворком PHP на ринку завдяки новим функціям, таким як Artisan CLI. З аналізу проведеного за допомогою Google Trends, зображеного на рисунку 2.5.1, яскраво видно як Laravel швидко опереджував інші популярні фреймворки, такі як Symfony, CodeIgniter і Yii.

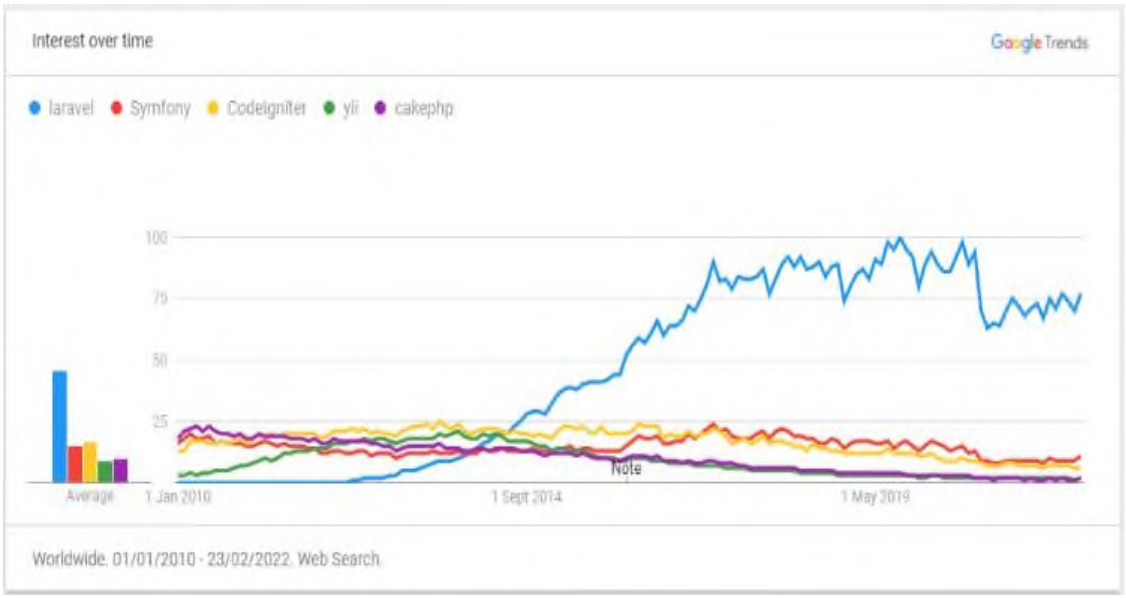


Рис. 2.1 Світова популярність фреймворків

У Laravel є такі корисні вбудовані функції, як інтерфейс командного рядка (CLI) Artisan, власна автентифікація та архітектура контролера модельного перегляду (MVC). Ці функції роблять фреймворк простим у використанні і є основною причиною його популярності. Він має понад 70 000 зірок на GitHub і став затребуваною навичкою на ринку праці.

Laravel — це фреймворк PHP із відкритим кодом, який використовується для створення веб-застосунків і програм. Він містить усі необхідні компоненти та функції, щоб допомогти веб-розробникам створити веб-додаток за допомогою мови програмування PHP. Важливо зазначити, що Laravel — не єдиний фреймворк, доступний розробникам PHP. Насправді інші популярні фреймворки PHP включають Symfony, CodeIgniter і Yii (це лише деякі з них). А якщо ми вийдемо за межі зони PHP, то буде ще більше доступних фреймворків, таких як Rails, який

базується на мові Ruby, .NET, який використовує C#, або Sails.js, фреймворк Javascript. Як і більшість популярних веб-фреймворків, Laravel дотримується підходу моделі-перегляду-контролера (MVC) до розробки [13]. Це просто означає, що процес розробки розбивається на три основні компоненти:

— Модель: як дані обробляються та зберігаються на сайті чи в додатку;
— Перегляд: це візуальне представлення даних, або, іншими словами, UI (інтерфейс користувача). Це здебільшого оброблятиметься HTML/CSS та JavaScript. Приклади елементів інтерфейсу можуть включати спадні списки, діаграми, текстові поля та таблиці;
— Контролер: як випливає з назви, він контролює дані, які показуються користувачеві. Коли користувач взаємодіє із веб-застосунком, він отримує відповідні дані з моделі, а потім відображає їх користувачеві за допомогою компонента перегляду.

Як ви можете собі уявити, цей підхід MVC до розробки веб-застосунків може мати багато переваг, включаючи швидший час розробки, простіше обслуговування та ще більшу масштабованість. Це робить такий фреймворк, як Laravel, бажаним вибором для спеціальних веб-проектів.

Laravel – це передусім бекенд-фреймворк. Він пропонує деякі незначні інтерфейсні функції, але найкраще працює з іншою інтерфейсною структурою. Коли справа доходить до вибору зовнішнього середовища для вашого проекту Laravel, є багато варіантів на вибір. В межах моєї магістерської роботи я буду використовувати доволі простий та зручний, як для мене, редактор коду Notepad++. В нього я встановив та інтегрував Laravel для подальшої розробки веб-застосунку. Notepad++ має підтримку багатьох мов програмування та технологій для веб-розробки, таких як: C++, PHP, Java, XML, HTML, CSS, JavaScript, Python, Ruby та запити SQL. В Інтернеті є багато документації та підтримки для інтеграції інших зовнішніх середовищ з інтерфейсними фреймворками.

Важливо розуміти, чим саме фреймворк Laravel відокремлюється від мови програмування PHP. Laravel — це фреймворк, певна платформа, що створена за допомогою мови сценаріїв PHP та певних бібліотек. PHP (або ядро PHP) — мова

програмування з відкритим кодом. Іншими словами, це серверна мова, яка відповідає за обробку даних у веб-додатку та його роботи з базами даних. PHP сам по собі не можна використовувати для створення веб-програми. Мови на стороні клієнта, такі як Javascript, HTML і CSS, також необхідні для формування інтерфейсу програми, поряд з PHP. Фреймворк — це набір мов програмування, бібліотек і компонентів. Особливістю Laravel є те, що він не потребує додаткової установки великої кількості бібліотек, пакетів та плагенів. Всі вони вже містяться в середовищі розробки на відміні від інших фреймворків.

Мета використання фреймворку – полегшити та пришвидшити розробку застосунку. Оскільки фреймворки вже мають вбудовані спільні функції та завдання. Для порівняння, якби ви використовували ядро PHP для розробки веб-додатку з нуля, вам знадобилося б багато років, щоб написати весь код. Єдина справжня подібність між Laravel і PHP полягає в тому, що обидва можна використовувати для створення веб-додатків. І, звісно, без знання PHP розробнику було б дуже важко використовувати структуру Laravel. Наголошую, що без знання мови програмування PHP та баз даних MySQL не можливо працювати з цим фреймворком. Це професіональний інструмент для веб-розробки а не конструктор сайтів.

Laravel має багато особливостей та переваг порівняно з іншими фреймворками. Наприклад, основні з них:

— Можливість оновлення на нові версії фреймворку через веб-документацію та опис оновлення. Розробники фреймворку надають програмістам маневрувати від старіших до новіших версій Laravel та навпаки, надаючи їм повну документації з описом до кожного оновлення програми. За допомогою певних команд можливе оновлення методів, середовища та іншого. За потреби, також існує можливість повернутися на стару версію для внесення певних коректив, тощо;

— Artisan – це інтерфейс командного рядка, що входить до складу Laravel. Artisan існує в корені вашої програми як сценарій artisan і надає ряд корисних команд, які можуть допомогти під час створення програми. Щоб переглянути список усіх доступних команд Artisan, ви можете скористатися командою «list». Кожна

команда також містить екран «довідки», який відображає та описує доступні аргументи та параметри команди. Щоб переглянути екран довідки, пропишіть команду «help». Laravel використовує Artisan як CLI, який допомагає веб-розробникам:

а) Перенести дані;

б) Керувати базами даних;

в) Створювати шаблонний код, контролери, моделі та інш. Artisan CLI спрощує веб-розробку завдяки функціям генерації коду та керування базами даних, доступними лише за кілька команд. Замість того, щоб писати шаблонний код або налаштовувати базу даних, розробник може зосередитися на створенні логіки програми.

— Особистий клас requset, який містить купу зручних методів для отримання та обробки input'ів, не важливо в якому форматі прийшов запит, завжди повернеться значення параметра в потрібному нам форматі. Можливість автоматичної конвертації нових значень в булеан, для зручнішого розуміння отриманих даних та їх обробки. А також конвертація різних даних в потрібні формати при відрправці на фронтенд;

— Localization – можливість розробляти мультиязичніть додатків за допомогою своїх папок та файлів з перекладом. Зручне керування та доступ до них, дозволяє ефективно використовувати свої функції та скрипти перекладів на інші мови, які допомагають поліпшувати процес локалізацій веб-застосунків під певні регіони;

— Фреймворк Laravel підтримує архітектурний шаблон MVC для створення веб-програм. Цей шаблон визначає набір правил, які вказують, як створювати та підтримувати масштабовані веб-програми. Шаблон Laravel MVC допомагає розробникам навести порядок і узгодженість неструктурованого коду. Підхід MVC також полегшує розробку як малих, так і великомасштабних веб-додатків.

— Вбудовані пакети, які можливо додавати як готові функції до своєї веб-програми без необхідності писати код з нуля. Навіть існує можливість створювати власні пакети окремо та додавати їх у розробку. Laravel підтримує можливість кастомізації. Мається на увазі, можливість разробляти кастомні консольні команді,

які потім можна використовувати по ходу розробки проекту. Їх застосування можливо при роботі з колекціями. Наприклад, має певни об’єкет в який передаємо певний масив. Надалі цей масив можливо обробляти певними функціями та методами, які доступні для колекцій. Також, костумізовати можна помилки (FindorFail) та певні пакети. Додатковий розділ Helpers включає глобальні вспоміжні функції та методи, які містить цей фреймворк для роботи з PHP, наприклад, робота зі строками та масивами;

— Автоматизоване тестування, що включає як модульне тестування, так і тестування функцій. Модульне тестування може тестувати невеликі фрагменти коду. У той час як тестування функцій можна використовувати для перевірки великих фрагментів коду або функцій на сайті;

— Обробка маршрутів: використання простих імен замість довгих імен шляхів полегшує керування великими програмами. Крім того, усі назви маршрутів можна змінити в одному місці за допомогою спеціального файлу, замість того, щоб змінювати назви маршрутів вручну кілька разів у програмі;

— Функції безпеки. Безпека – це питання номер один для більшості компаній. За допомогою Laravel розширені функції безпеки легко налаштувати на більшості веб-додатків для підвищення безпеки та захисту застосункі від хакерів і кіберзлочинців. Laravel використовує алгоритм хешування Bcrypt, що означає, що він ніколи не зберігає паролі в базі даних. Це забезпечує безпеку ваших даних і даних ваших клієнтів. Laravel пропонує низку функцій безпеки, таких як аутентифікація користувачів, авторизація ролей, підтвердження електронної пошти, хешування паролів тощо;

— Міграція бази даних: з контролем версій міграцією бази даних керувати набагато легше;

— На відміну від багатьох інших фреймворків, Laravel готово підтримує кешування веб-додатку, що чудово підходить для підвищення швидкості застосунку. Щоб ще більше підвищити продуктивність програми, Laravel робить інші методи оптимізації швидкості, такі як зменшення використання пам’яті та індексування бази даних, дуже простими у застосуванні. Це робить Laravel чудовим вибором для

проектів, якщо швидкість сайту та зручність для SEO-оптимізації є одними з ключових вимог;

— Зростання обсягу трафіку у веб-додатку – не проблема. Веб-застосунок створений на Laravel, може обробляти запити набагато швидше, ніж більшість інших фреймворків. Laravel використовує унікальну систему черги повідомлень, що означає, що ви можете відкласти певні завдання веб-додатку, наприклад надсилання електронних листів, на пізніший час. Можливість контролювати трудомісткі завдання означає, що ваш веб-застосунок може виконувати завдання швидше. І це не тільки підтримує працездатність сервера, але й може знизити витрати на хостинг у довгостроковій перспективі;

— Майже кожен сайт потребує інтеграції з будь-яким додатком третьої сторони. Це може включати платіжну систему або маркетинговий інструмент, який використовує ваша компанія. Незалежно від інтеграції, Laravel спрощує інтеграцію програм сторонніх розробників завдяки своїм чистим API для інтеграції.

2.6 PHP та Blade

Blade Template Engine — це потужний інструмент, який постачається разом із Laravel. Використання Blade дозволяє:

— Підключати моделі даних;

— Обробляйти код програми всередині тегів шаблону;

— Перенаправлення виводу в текстовий файл або інші потоки.

Завдяки цим функціям Blade робить розробку швидшою та зручнішою.

Laravel — це бекенд-фреймворк, який надає всі функції, необхідні для створення сучасних веб-додатків, як маршрутизація, перевірка, кешування, черги, зберігання файлів тощо. Але також, розробники Laravel запропонували програмістам повний стек фронтенl розробки, включаючи потужні підходи для створення інтерфейсу вашої програми. Існує два основні способи розробки зовнішнього інтерфейсу під час створення програми за допомогою Laravel:

створення свого зовнішнього інтерфейсу, використовуючи PHP або фреймворки JavaScript, такі як Vue і React.

У минулому більшість додатків PHP відтворювали HTML у веб-переглядачі за допомогою простих шаблонів HTML, які перемежовувалися операторами PHP echo, що відтворювали дані, отримані з бази даних під час запиту (зображено на рис. 2.2).

```
<div>
    <?php foreach ($users as $user): ?>
        Hello, <?php echo $user->name; ?> <br />
    <?php endforeach; ?>
</div>
```

Рис. 2.2 PHP в шаблоні HTML

У Laravel цей підхід до відтворення HTML все ще можна досягти за допомогою представлень і Blade. Blade — це надзвичайно легка мова шаблонів, яка забезпечує зручний і короткий синтаксис для відображення даних, ітерації даних тощо [26]. Зразково зображено на рис. 2.3, як шаблони Blade дозволяють зменшити обсяг коду.

```
<div>
    @foreach ($users as $user)
        Hello, {{ $user->name }} <br />
    @endforeach
</div>
```

Рис. 2.3 Приклад застосування Blade

Під час створення додатків у такий спосіб подання форм та інші взаємодії сторінок зазвичай отримують абсолютно новий HTML-документ із сервера, і вся сторінка повторно відображається браузером. Навіть сьогодні багато програм можуть ідеально підходити для того, щоб створювати їхні інтерфейси таким чином, використовуючи прості шаблони Blade.

Однак у міру того, як очікування користувачів щодо веб-додатків зросли, багато розробників виявили потребу створювати більш динамічні інтерфейси з більш витонченою взаємодією. Зважаючи на це, деякі розробники вирішують

розпочати створення інтерфейсу своєї програми за допомогою фреймворків JavaScript, таких як Vue і React. Інші, віддаючи перевагу тій мові серверної програми, яка їм зручна, розробили рішення, які дозволяють створювати сучасні інтерфейси користувача веб-додатків, водночас переважно використовуючи мову серверної програми, яку вони вибрали. Наприклад, в екосистемі Rails це спонукало до створення таких бібліотек, як Turbo Hotwire і Stimulus. В екосистемі Laravel потреба створити сучасні динамічні інтерфейси, переважно використовуючи PHP, призвела до створення Laravel Livewire і Alpine.js.

Laravel Livewire — це фреймворк для створення інтерфейсів на базі Laravel, які виглядають динамічно, сучасно та живо, як і інтерфейси, створені за допомогою сучасних фреймворків JavaScript, таких як Vue і React. Використовуючи Livewire, ви створите «компоненти» Livewire, які візуалізують окрему частину вашого інтерфейсу користувача та відкривають методи та дані, які можна викликати та взаємодіяти з інтерфейсом вашої програми. Наприклад, на рис. 2.4 зображено, як виглядає простий компонент «Лічильник».

```php
<?php

namespace App\Http\Livewire;

use Livewire\Component;

class Counter extends Component
{
    public $count = 0;

    public function increment()
    {
        $this->count++;
    }

    public function render()
    {
        return view('livewire.counter');
    }
}
```

Рис. 2.4 «Лічильник» на PHP

Відповідний шаблон для лічильника з урахуванням Laravel Livewire, відображено на рис. 2.5.

```
<div>
    <button wire:click="increment">+</button>
    <h1>{{ $count }}</h1>
</div>
```

Рис. 2.5 «Лічильник» на Laravel Livewire

Як бачите, Livewire дає змогу писати нові HTML-атрибути, такі як wire:click, які з'єднують інтерфейс і серверну частину програми Laravel. Крім того, ви можете відобразити поточний стан вашого компонента за допомогою простих виразів Blade. Очевидно, що це дуже спрощує розробку та зменшує обсяг рутинних дій, а також робить код чистим і елегантним.

Для багатьох Livewire зробив революцію в розробці інтерфейсу за допомогою Laravel, дозволивши їм залишатися в комфорті Laravel під час створення сучасних динамічних веб-додатків. Як правило, розробники, які використовують Livewire, також використовуватимуть Alpine.js, щоб «всипати» JavaScript у свій інтерфейс лише там, де це необхідно, наприклад, щоб відобразити діалогове вікно.

2.7 Eloquent ORM та Query Builder в Laravel

Eloquent ORM являється основною фішкою Laravel та одним з найвжливіших особливостей цього фреймворку. Абстракція над базою даних. Eloquent object-relational mapper (ORM) — це інструмент керування базами даних, який є частиною фреймворку Laravel [26].

Eloquent полегшує взаємодію з базою даних. З Eloquent кожна таблиця бази даних має відповідну модель, яку можна використовувати для подальшої взаємодії з таблицею. Окрім отримання записів із таблиці, модель Eloquent дозволяє вставляти, оновлювати та видаляти записи.

Eloquent — це об'єктно-реляційний маппер (ORM), який за замовчуванням включений у структуру Laravel. ORM або Object Relational Mapper — це програмне забезпечення, яке полегшує обробку записів бази даних, представляючи дані як

об’єкти, працюючи як рівень абстракції поверх механізму бази даних, який використовується для зберігання даних програми.

ORM, що входить до складу Laravel, називається Eloquent і дає нам змогу працювати з об’єктами та зв’язками бази даних за допомогою виразного синтаксису. Це схоже на роботу з об'єктами в PHP. У Laravel кожна таблиця бази даних має відповідну «модель». Eloquent ORM забезпечує реалізацію Active Record, що означає, що кожна модель, яку ми створюємо в нашій структурі MVC, відповідає таблиці в нашій базі даних [17].

Створення моделі Eloquent подібне до створення класу. Усі файли моделі мають бути в папці app/models. Приклад створення моделі: «class Group extends Eloquent { }».

Усі моделі Eloquent походять від класу Eloquent. Ім'я класу в множині в нижньому регістрі використовуватиметься як ім'я таблиці, якщо інше ім'я не вказано явно. Eloquent також припускатиме, що кожна таблиця має стовпець первинного ключа з назвою «id», якщо не зазначено.

Ми можемо вказати таблицю наступним чином:

class Group extends Eloquent { protected $table = 'group_list'; } — вказуватиме на таблицю group_list, коли вона буде потрібна

Тут групова модель відповідатиме таблиці груп (за замовчуванням). Ми можемо отримати доступ до даних у таблиці груп за допомогою основних операцій CRUD (Create, Retrieve, Using, Delete):

а) Create: (створюємо групу)

```
$new_group = new Group;
$new_group->name = 'NewGroup';
$new_group->description = 'Awesome Group';
$new_group->save();
<read:< pre="">
```

б) Retrieve: (спробуємо отримати модель за первинним ключем, інакше виникне виняток)

```
$model = User::findOrFail(1);
```

```
$model = User::where('id', '>', 5)->firstOrFail();
$group = Group::find(1);
$group->name = ‘Group01’;
$group->save();
```

в) Using: (використання методу where)

```
Group::where('name', '=', ‘Group01’)-date(array('name' => ‘Group1’));
```

г) Delete: (видалимо один запис а потім видалимо декілька записів)

```
$group = Group::find(1);
$group->delete();
```

Group::destroy(1, 2, 3); – видалення декількох записів в певному діапазоні

Group::where('id', '<', 10)->delete();.

За замовчуванням моделі Eloquent матимуть ключі з автоматичним збільшенням. Eloquent полегшує завдання взаємодії з таблицями бази даних, забезпечуючи об’єктно-орієнтований підхід до вставки, оновлення та видалення записів бази даних, а також надаючи спрощений інтерфейс для виконання складних запитів SQL.

У Laravel конструктор запитів до бази даних надає простий інтерфейс для створення та виконання певних дій. Його можна використовувати для виконання всіх операцій з базою даних у вашій програмі, починаючи з базового підключення до БД, CRUD, агрегатів тощо, і він дуже продуктивно працює на всіх підтримуваних системах баз даних. Важливим фактором конструктора запитів є те, що, оскільки він використовує об’єкти даних PHP (PDO), розробнику не потрібно хвилюватися про атаки SQL-ін’єкцій (однак важливо переконайтеся, що ми ненавмисно не знімаємо цей захист) [30]. Розробник може уникнути всіх цих рядків коду, щоб очистити дані перед подачею їх до БД. Відображу це більш детально на прикладі створення простого запиту для отримання значень з таблиці користувачів: «$users = DB::table('users')->get()».

«DB::table» відповідає за початок вільного запиту до таблиці бази даних. Таблиця, з якої потрібно вибрати значення, згадується в дужках у лапках, і, нарешті, метод «get()» отримує значення. Так само, щоб отримати один рядок,

змінюємо наведений вище код, додавши пропозицію where: «$user = DB::table('users')->where('name', 'Oleksandr')->first()».

Тут отримаємо рядок, який має значення Oleksandr у стовпці імені. Метод «first()» поверне лише першу знахідку. Що робити, якщо нам потрібен лише ідентифікатор користувача Oleksandr. Замість того, щоб повертати весь масив результатів, ми можемо просто вилучити цей конкретний стовпець: «$user_id = DB::table('users')->where('name', 'Oleksandr')->pluck('id')».

Щоб вказати більше одного стовпця, ми можемо використати метод «select»: «$users = DB::table('users')->select('name', 'email')->get()».

Часто пишуться запити з певними умовами «where». Далі отримаємо список користувачів, чий «user_id» менше 10: «$users = DB::table('users')->where(id, '<', 10)->get()».

Так, я розділяю оператор і операнди на три параметри та передаю їх умовам «where». Тепер отримаємо всіх тих користувачів, чий «user_id» знаходиться між 10 і 20: «$users = DB::table('users')->whereBetween('id', array(10, 20))->get()».

Laravel має методи whereBetween(), whereNotBetween(), wherein() і whereNotIn(), яким можливо передавати значення у вигляді масиву.

На початку опрацювання Query Builder я згадав про SQL-атаки. Припустимо, що значення 10 і 20 беруться як введення користувача. Як програміст, я не можу довіряти тому, що користувач вводить у поле введення. Він може бути дійсним користувачем, який вводить правильні значення, або тим, хто намагається ввести хибні значення та призвести до збою моєї бази даних: «$users = DB::table('users')->whereBetween('id', array($from, $to))->get()».

Тут $from і $to — дані користувача. Якщо ми подивимося на клас з’єднання з базою даних Laravel для методу select(), цей масив обертається навколо з’єднання PDO, і він відповідає за очищення даних перед виконанням запиту. Отже, я отримаю чисті запити в мою базу даних тим самим передбачу SQL-іньекції моєї БД.

За допомогою конструктора запитів можна також писати необроблені запити SQL:

«DB::select(DB::raw(“SELECT * FROM `users` WHERE name = ‘$name’ ”))».

Тут $name отримано з введення користувача. $name може містити шкідливий код, тому потрібно змінити наведений вище код, щоб зробити його зручним для SQL.

«DB::select(DB::raw("SELECT * FROM `users` WHERE `name` = :username"), array('username' => $name))».

Отже, значення масиву під час проходження через з’єднання PDO очищається. Це пояснює чому значення передаються як масив, а не як параметри, розділені комами.

З дослідження стає зрозумілим, те що Eloquent ORM надає розробнику саме тих можливостей програмування та роботи з базами даних, яких дуже часто не вистачає в інших фреймворках [16]. За допомогою цієї технології можливо зручно розпреділяти бази даних на запис та читання. На приклад робити бази даних, з яких буде братися інформація, та надаватися користувачу. І в той час мати іншу базу даних, в яку буде записуватись інформація нових користувачів та інш. Також, завдяки Query Builder доступне використання джобів (певних методів) та баз даних для обробки інформації наданої користувачем. Наприклад, користувач оплачує замовлення. Його запит надсилається в чергу. Потім в базу даних те, що він його оплитв. В базі даних таблиця це бачить, та відправляє цей запит на джоб (метод обробки). З джобів вже робится повідомлення на пошту кристувача, що його замовлення дійсне, оплата пройшла та «дякуємо за замовлення!». Laravel має це все вбудованим в собі та надає можливість отпимізовано працювати з цими функціями. Існує бесплатний офіційний пакет Horizon, який допомагає керувати джобами, чергами та запитами.

Насправді фреймворк Laravel налічує багато офіційних платних та безкоштовних додаткових пакетів, які допомагають вирішувати певні задачі при розробці проектів різних розмірів. Розглянемо основні з них:

— Laravel Vapor. Безсерверна платформа для розгортання Laravel з автоматичним масштабуванням, яка працює на основі AWS Lambda. Керування своєю інфраструктурою Laravel на Vapor передбачає сприятливу масштабованість і простоту безсерверної роботи. Vapor абстрагує складність керування програмами Laravel на AWS Lambda, а також взаємодію цих програм із чергами SQS, базами даних, кластерами Redis, мережами, CloudFront CDN тощо;
— Laravel Envoyer. Envoyer це zero-downtime деплоєр для PHP та Laravel проектів. Це інструмент, який підключається до вашого сервера, щоб деплоїти, і який використовує серію інструментів, щоб забезпечити те, що вся підготовча робота для деплою (наприклад, composer install), виконується на тлі, в той час як попередня версія сайту все ще працює;
— Laravel Nova. Інтерфейс керування для веб-додатків Laravel. Він містить набір інструментів та інтерфейсів для швидкого створення індивідуальних панелей адміністратора, які можна легко адаптувати до ваших індивідуальних вимог. На мові розробників це «адмінка». Він базується на структурі Laravel і використовує її особливості та функціональність для створення плавного та інтуїтивно зрозумілого досвіду як для розробників, так і для користувачів. Фільтрація та пошук даних, користувацькі показники та візуалізації, а також проста взаємодія зі сторонніми інструментами та службами є одними з можливостей Laravel Nova. Цей пакет також має широкі можливості розширення з підтримкою спеціальних полів, дій і налаштувань, що робить його незамінним інструментом для будь-якого розробника, який спеціалізується на цій технології;
— Laravel Octane. Являє собою пакет з відкритим кодом, який значно поліпшить ефективність розробляємого веб-застосунку. Під капотом Octane використовуються Swoole та RoadRunner – два сервери програм, які відповідають за обслуговування та завантаження розробляємої програми. У звичайній PHP-програмі, яка працює, наприклад, через веб-сервер nginx, кожен вхідний запит породжує PHP-FPM воркер. Це означає, що кожен запит ініціює окремий процес PHP, який виконує всі необхідні завдання для обслуговування запиту. У випадку Laravel це означає, що кожен запит призводить до повторного завантаження

фреймворка, реєстрації служб від усіх сервіс-провайдерів, ініціалізації провайдерів, проходження через список middleware, виклику контролера, відтворення шаблону тощо, поки не отримаємо відповідь від сервера. У випадку використання Swoole або RoadRunner кожен вхідний HTTP-запит все ще створює воркер, але всі воркери використовують один і той же вже завантажений фреймворк. Це означає, що лише перший вхідний запит ініціює завантаження фреймворка (включаючи усі сервіс-провайдери і т.д.), в той час як кожен наступний запит використовує вже завантажений фреймворк. Це є ключовою особливістю, яка забезпечує високу швидкодію Octane;

— Laravel Flare. Flare відстежує ваші програми PHP і JavaScript і відстежує всі помилки, коли щось йде не так. Для типових проблем Flare пропонує рішення та відповідну документацію. У локальній розробці Flare може автоматично виправляти помилки за вас одним клацанням миші. Відстеження помилок Laravel ніколи не було зручнішим. Flare — це засіб відстеження помилок Laravel, який зберігає витрати на низькому рівні незалежно від того, скільки програмістів працює над проектом. Flare поєднується з Ignition — відкритою сторінкою помилок Laravel. Сервіс забезпечує цілеспрямований і надшвидкий інтерфейс із можливістю налаштування.

Насправді, це все ще далеко не кінець особливостей та переваг Laravel у веб-розробці. Наприклад, цей фреймворк також включає:

а) CSRF protection – Вбудований захист від кроссайтових атак;

б) HTTP client – додатковий вбудований зручний клієнт для розробки та робти з API;

в) Клас пошти, який допомагає надсилати пошту з багатим вмістом і вкладеннями з веб-програми.

З проведеного аналізу та дослідження, зрозуміло чому Laravel займає перші місця при виборі фреймворку для веб-розробки. Гігант, який надає будь-які можливості при розробці а також сприятливий інтерфейс. Структурований код та його чистота при розробці на Laravel не залишають байдужими веб-розробників вже декілька років. Саме через всі вище згадані аспекти та переваги цього

фреймворку, я вирішив обрати саме його для розробки веб-застосунку для клієнтів на базі мови програмування PHP.

2.8 Аналіз та вибір бази даних для розробки

Кожен сучасний веб-додаток має мати особисту базу даних для зберігання, роботи і обробки певної інформації та її різновидів. Існує певне різноманіття типів та структур БД, але найпопулярніший підхід сьогодні це реляційна база даних. Існує декілька видів реляційних баз даних. Наприклад, це може бути MySQL, SQLite, Maria DB, Oracle, Microsoft SQL Server та PostgreSQL. Особливість такої БД полягає в її чіткій структурі та впорядковоності. Для керування та взаємодії з реляційними базами даних використовують спеціальну, структуровану мову SQL, яка ялвяє собою певні запити та команди з своїм унікальним синтексисом і особливостями [21].

Реляційна база даних зберігає всю інформацію в собі завдяки таблицям, пов’язаниим між собою певними відносинами. Ці відносини довзоляють обробляти дані з декількох таблиць за допомогою вище згаданих SQL запитів. Дані зазвичай структуровані в кількох таблицях, які можна об’єднати за допомогою первинного або зовнішнього ключа. Ці унікальні ідентифікатори демонструють різні зв’язки, які існують між таблицями, і ці зв’язки зазвичай ілюструються за допомогою різних типів моделей даних. Приклад подібних зв’зків у реляційних базах даних наведено на рисунку 2.6. Розробники використовують SQL-запити, щоб комбінувати різні точки даних і підсумовувати ефективність, що дозволяє отримати розуміння, оптимізувати робочі процеси та виявити нові можливості. Наприклад, маємо таблицю бази даних із інформацією про клієнтів, яка містить дані компанії на рівні облікового запису. Також може бути інша таблиця, яка описує всі окремі транзакції, пов’язані з цим обліковим записом. Разом ці таблиці можуть надати інформацію про різні галузі, які купують певний програмний продукт. Стовпці (або поля) для таблиці клієнта можуть бути ідентифікатором клієнта, назвою компанії, адресою компанії, галуззю тощо. Стовпцями для таблиці транзакцій можуть бути

дата транзакції, ідентифікатор клієнта, сума транзакції, метод платежу тощо. Таблиці можна об’єднати за допомогою загального поля ідентифікатора клієнта. Таким чином, розробник може запитувати таблицю, щоб створювати цінні звіти, такі як звіти про продажі за галуззю чи компанією, які можуть інформувати потенційних клієнтів.

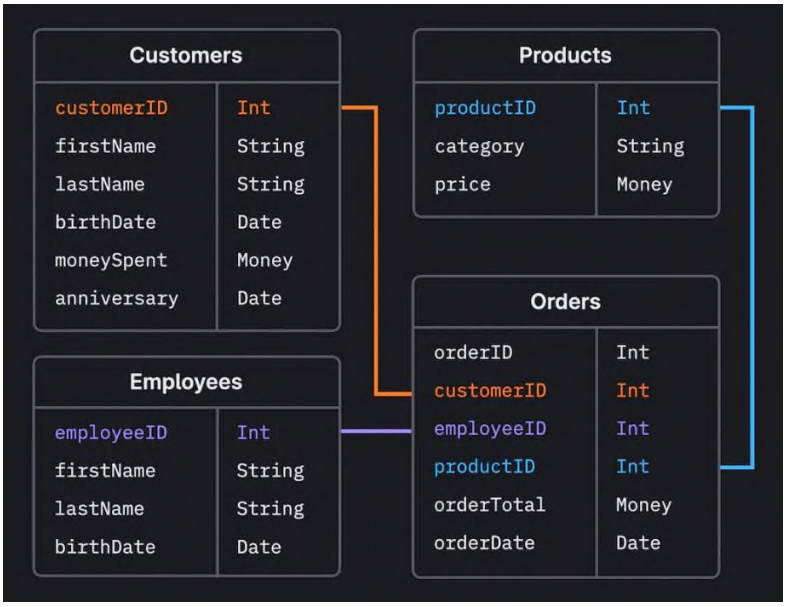


Рис. 2.6 Зв’язки в реляційних БД

Реляційні бази даних також зазвичай асоціюються з транзакційними базами даних, які спільно виконують команди або транзакції. Популярним прикладом, який використовується для ілюстрації цього, є банківський переказ. Визначена сума знімається з одного рахунку, а потім подається на інший. Загальна сума грошей знімається та вноситься, і ця транзакція не може відбутися в якомусь частковому сенсі. Транзакції мають специфічні властивості. Представлені акронімом ACID, властивості ACID визначаються як:

— Atomicity (Атомарність): усі зміни даних виконуються так, ніби це одна операція. Тобто виконано всі зміни або жодної з них;

— Consistency (Послідовність): дані залишаються в послідовному та узгодженому стані від стану до кінця, зміцнюючи цілісність даних;

— Isolation (Ізоляція): проміжний стан транзакції не видно іншим транзакціям, і в результаті транзакції, які виконуються одночасно, здаються серіалізованими;

— Durability (Довговічність): після успішного завершення транзакції зміни в даних зберігаються й не скасовуються навіть у разі збою системи.

У той час як реляційні бази даних структурують дані в табличному форматі, нереляційні бази даних не мають такої жорсткої схеми бази даних. Насправді нереляційні бази даних організовують дані по-різному залежно від типу бази даних. Незалежно від типу нереляційної бази даних, усі вони спрямовані на вирішення проблем гнучкості та масштабованості, властивих реляційним моделям, які не є ідеальними для неструктурованих форматів даних, таких як текст, відео та зображення. Ці типи баз даних включають:

— Сховище ключ-значення: ця модель даних без схеми організована у структуру пар ключ-значення, де кожен елемент має ключ і значення. Ключ може бути схожим на щось подібне в базі даних SQL, як ідентифікатор кошика для покупок, а значення — це масив даних, як кожен окремий товар у кошику для покупок цього користувача. Він зазвичай використовується для кешування та зберігання інформації про сеанс користувача, наприклад, про кошики для покупок. Однак це не ідеально, коли потрібно отримати кілька записів одночасно. Redis і Memcached є прикладами відкритих баз даних із цією моделлю даних;

— Сховище документів: як випливає з назви, бази даних документів зберігають дані як документи. Вони можуть бути корисними в управлінні напівструктурованими даними, а дані зазвичай зберігаються у форматах JSON, XML або BSON. Це зберігає дані разом, коли вони використовуються в програмах, зменшуючи кількість перекладу, необхідного для використання даних. Розробники також отримують більше гнучкості, оскільки схеми даних не повинні збігатися між документами (наприклад, ім’я та ім’я). Однак це може бути проблематичним для складних транзакцій, що призводить до пошкодження даних. Популярні випадки використання баз даних документів включають системи керування вмістом і профілі користувачів. Прикладом документно-орієнтованої бази даних є MongoDB, компонент бази даних стека MEAN;

— Зберігання з широкими стовпцями: ці бази даних зберігають інформацію в стовпцях, що дозволяє користувачам отримувати доступ лише до певних стовпців,

які їм потрібні, не виділяючи додаткової пам’яті для нерелевантних даних. Ця база даних намагається усунути недоліки сховищ ключ-значення та документів, але оскільки вона може бути складнішою системою для керування, її не рекомендується використовувати для нових груп і проектів. Apache HBase та Apache Cassandra є прикладами баз даних із відкритим кодом із широкими стовпцями. ApacheHBase створено на основі розподіленої файлової системи Hadoop, яка забезпечує спосіб зберігання розріджених наборів даних, який зазвичай використовується в багатьох програмах для великих даних. ApacheCassandra, з іншого боку, розроблено для керування великими обсягами даних на кількох серверах і кластеризації, яка охоплює кілька центрів обробки даних. Його використовували для різноманітних випадків використання, наприклад веб-сайтів соціальних мереж і аналізу даних у реальному часі; — Сховище графів: цей тип бази даних зазвичай містить дані з графа знань. Елементи даних зберігаються як вузли, ребра та властивості. Вузлом може бути будь-який предмет, місце чи людина. Ребро визначає відношення між вузлами. Графові бази даних використовуються для зберігання та керування мережею зв’язків між елементами в межах графа. Neo4j (посилання знаходиться за межами IBM), графічний сервіс бази даних на основі Java з версією спільноти з відкритим вихідним кодом, де користувачі можуть придбати ліцензії на онлайн-резервне копіювання та розширення високої доступності, або ліцензовану версію попереднього пакета з резервним копіюванням і розширеннями.

Основна концепція нереляційних баз даних полягає в тому, що NoSQL надають перевагу доступності над узгодженістю.

Мова структурованих запитів (structured query language – SQL) є стандартною мовою програмування для взаємодії з системами керування реляційними базами даних, що дозволяє адміністратору бази даних легко додавати, оновлювати або видаляти рядки даних. Запити SQL також дозволяють користувачам отримувати дані з баз даних, використовуючи лише кілька рядків коду. Враховуючи цей зв’язок, легко зрозуміти, чому реляційні бази даних іноді також називають «базами даних SQL». Можливість об’єднання даних у такий спосіб допомагає зменшити

надмірність у системах даних, дозволяючи групам обробки даних підтримувати одну основну таблицю для клієнтів, а не дублювати цю інформацію, якщо в майбутньому відбудеться інша транзакція. В таблиці 2.2 будуть наведені базові команди SQL для взаємодії з базами даних.

Таблиця 2.2

Базові команди SQL

Create	Генерація нової таблиці
Use	Вибір певної таблиці для подальшої роботи з нею
Source	Дозволяє виконувати одразу декілька команд, що знаходяться в файлі .sql
Insert	Добавлення даних в таблицю
Update	Оновлення даних в таблицях
Select	Отримання певних даних з вибраної таблиці
Join	Використовується для зв’язку двох або більше таблиць за допомогою спільних атрибутів
Drop	Команда для видалення бази даних

Основною перевагою підходу до реляційної бази даних є можливість створювати значущу інформацію шляхом об’єднання таблиць. Об’єднання таблиць дозволяє зрозуміти зв’язок між даними або спосіб з’єднання таблиць. SQL включає в себе можливість підраховувати, додавати, групувати, а також комбінувати запити. SQL може виконувати базові математичні функції та функції проміжного підсумку та логічні перетворення [25]. Програмісти можуть упорядкувати результати за датою, назвою або будь-яким стовпцем. Ці функції роблять реляційний підхід єдиним найпопулярнішим інструментом запитів у сучасному світі веб-розробки. Реляційні бази даних мають кілька переваг порівняно з іншими форматами баз даних:

— Простота використання. Через тривалість життя продукту існує більше спільноти навколо реляційних баз даних, що частково увічнює його подальше використання. SQL також дозволяє легко отримувати набори даних із кількох таблиць і виконувати прості перетворення, такі як фільтрація та агрегація. Використання індексів у реляційних базах даних також дозволяє їм швидко

знаходити цю інформацію без пошуку в кожному рядку вибраної таблиці. Хоча реляційні бази даних історично розглядалися як більш жорсткий і негнучкий варіант зберігання даних, прогрес у технології та опції DBaaS змінюють це сприйняття. Незважаючи на те, що розробка схем вимагає більше витрат порівняно з пропозиціями баз даних NoSQL, реляційні бази даних стають більш гнучкими, коли вони переходять у хмарне середовище;
— Зменшена надмірність. Реляційні бази даних можуть усунути надмірність двома способами. Сама реляційна модель зменшує надмірність даних за допомогою процесу, відомого як нормалізація. Як зазначалося раніше, таблиця клієнта повинна реєструвати лише унікальні записи інформації про клієнта, а не дублювати цю інформацію для кількох транзакцій. Збережені процедури також допомагають зменшити кількість повторюваної роботи. Наприклад, якщо доступ до бази даних обмежено певними ролями, функціями або командами, збережена процедура може допомогти керувати контролем доступу. Ці багаторазові функції звільняють бажаний час розробників додатків для виконання високопродуктивної роботи;
— Простота резервного копіювання та аварійного відновлення. Реляційні бази даних є транзакційними — вони гарантують узгодженість стану всієї системи в будь-який момент. Більшість реляційних баз даних пропонують прості варіанти експорту та імпорту, що робить резервне копіювання та відновлення тривіальним. Ці експорти можуть відбуватися навіть під час роботи бази даних, що спрощує відновлення після збою. Сучасні хмарні реляційні бази даних можуть здійснювати безперервне дзеркальне відображення, завдяки чому втрата даних під час відновлення обчислюється секундами або менше. Більшість хмарних служб дозволяють створювати репліки для читання, як у IBM Cloud® Databases для PostgreSQL. Ці репліки для читання дозволяють зберігати копію ваших даних, доступну лише для читання, у хмарному центрі обробки даних. Репліки також можуть бути підвищені до екземплярів читання/запису для аварійного відновлення.

У той час як реляційна база даних організовує дані на основі реляційної моделі даних, система керування реляційною базою даних (relational database management system – RDBMS) є більш конкретним посиланням на базове

програмне забезпечення бази даних, яке дозволяє програмістам та розробникам підтримувати її. Ці програми дозволяють користувачам створювати, оновлювати, вставляти або видаляти дані в системі, і вони забезпечують:

а) Структуру даних;

б) Багатокористувацький доступ;

в) Контроль привілеїв;

г) Доступ до мережі.

Приклади популярних систем RDBMS включають MySQL, PostgreSQL і IBM DB2. Крім того, система реляційної бази даних відрізняється від базової системи управління базами даних (СУБД) тим, що вона зберігає дані в таблицях, тоді як СУБД зберігає інформацію у вигляді файлів.

При розробці веб-додатку для клієнтів на базі програмування PHP я обрав саме MySQL базу даних і далі поясню чому.

2.9 Бази даних MySQL

MySQL (My Structured Query Language) — це безкоштовна система керування реляційною базою даних (RDBMS), яка є безкоштовною з відкритим вихідним кодом і використовує різні пропрієтарні ліцензії, у тому числі GNU General Public License (GPL). Як інші RDBMS, MySQL використовує SQL для керування даними всередині бази даних. Ця взаємодія також відбувається безпосередньо завдяки мові програмування PHP. На рисунку 2.7 зображен зв’язок PHP з MySQL. Він організовує корельовані дані в одну або кілька таблиць даних, і ця кореляція допомагає структурувати дані. Це дозволяє програмістам використовувати SQL для створення, зміни та вилучення даних з реляційної бази даних. Нормалізуючи дані в рядках і стовпцях таблиць, MySQL перетворюється на масштабовану, але гнучку систему зберігання даних із зручним інтерфейсом, яка може керувати великою кількістю інформації. MySQL також контролює доступ користувачів до бази даних як додатковий захід безпеки, керуючи користувачами та надаючи доступ до мережі на основі правил адміністратора. І це полегшує перевірку цілісності бази даних і створення резервних копій [25]. Хоча доступ до MySQL зазвичай здійснюється за

допомогою SQL, він часто використовується з іншими програмами як компонент різноманітних стеків технологій, включаючи LAMP (Linux, Apache, MySQL і Perl/PHP/Python). У результаті навіть деякі CMS, які потребують можливостей реляційної бази даних, працюють на базах даних, які використовують MySQL, включаючи Drupal, Joomla, phpBB і WordPress. Деякі популярні веб-додатки також використовують MySQL, наприклад Facebook, Flickr, Twitter і YouTube.

Рис. 2.7 MySQL та PHP

MySQL працює в структурі клієнт-сервер. Кілька пристроїв підключаються до сервера через мережу, дозволяючи клієнтам запитувати дані через графічний інтерфейс користувача (GUI). Ключові процеси в середовищі MySQL можна підсумувати таким чином: MySQL створює та керує базами даних, визначаючи зв’язки між таблицями; Клієнти роблять запити, вводячи певні оператори SQL; Сервер обробляє ці запити, доставляючи запитані дані клієнтам.

Вибір графічного інтерфейсу користувача MySQL може значно вплинути на ефективність керування даними за допомогою таких опцій, як MySQL Workbench, SequelPro, DBVisualizer і Navicat DB Admin Tool. Вибір відповідного графічного інтерфейсу залежить від індивідуальних потреб, причому phpMyAdmin є улюбленим вибором для керування веб-базами даних. SQL і MySQL пов’язані, але не взаємозамінні терміни. Давайте детальніше дослідимо відмінності між ними:

SQL — це мова, яка використовується для керування реляційними базами даних і надсилання запитів. Це стандартна мова, яка надає набір команд і синтаксис для взаємодії з базами даних. SQL використовується для створення, оновлення, отримання та видалення даних із бази даних, а також для визначення та керування структурою бази даних (таблицями, індексами тощо). SQL — це не конкретна система баз даних, а мова, яка підтримується багатьма системами керування реляційними базами даних, включаючи MySQL.

Для ефективного керування базами даних за допомогою SQL існують чотири основні типи мов SQL, які служать різним цілям. Ці різні типи мов SQL разом сприяють ефективному управлінню даними та контролю в системі бази даних. Розглянемо їх:

— Мова запитів даних (DQL): DQL в основному використовується для запитів до баз даних для отримання інформації зі збережених даних. Наприклад, він використовується для таких завдань, як вибір і отримання максимального значення в певному стовпці таблиці;

— Мова визначення даних (DDL): DDL відповідає за визначення структур і схем бази даних. Він включає такі дії, як створення таблиць, визначення типів даних і встановлення загальної структури бази даних;

— Мова керування даними (DCL): DCL фокусується на управлінні доступом, авторизаціями та дозволами для користувачів і процесів, які взаємодіють з базою даних. Він охоплює такі завдання, як надання адміністративних привілеїв, а також обмеження певних користувачів доступом лише для читання;

— Мова обробки даних (DML): DML використовується для внесення змін до існуючих компонентів у базі даних. Загальні операції включають вставку нових записів, оновлення значень у клітинках і видалення даних із таблиць.

З іншого боку, MySQL — це спеціальна система керування реляційною базою даних (RDBMS), яка використовує SQL як мову запитів. MySQL зберігає дані в таблицях із рядками та стовпцями, а також підтримує SQL для взаємодії з даними. MySQL відомий своєю продуктивністю, надійністю та простотою використання,

що зробило його популярним вибором для веб-додатків, систем керування вмістом (CMS) та інших типів програмного забезпечення.

Сьогодні майже кожна веб-програма з відкритим кодом використовує MySQL. Він сумісний з усіма хостинг-провайдерами та надзвичайно простий у використанні. Але якщо веб-додаток або веб-сайт електронної комерції працює погано, пропоную три варіанти покращення роботи вашої MySQL бази даних:

а) Точне налаштування продуктивності. Можна покращити продуктивність веб-додатку, налаштувавши проксі-сервер високої доступності або екземпляри HAProxy. Крім того, використовувати оновлене програмне забезпечення балансування навантаження, щоб оптимізувати базу даних і пришвидшити сервер. Програмне забезпечення для балансування навантаження бази даних розроблено для забезпечення гнучкості та масштабованості для розширення можливостей у разі потреби. Він також може задовольнити незаплановані вимоги до продуктивності в майбутньому;

б) Аудити безпеки. Атаки типу «відмова в обслуговуванні» (DoS) і спам можуть завдати шкоди серверу бази даних. Але надійне програмне забезпечення для балансування навантаження допомагає легко запобігти проблемам з продуктивністю та збільшити час безвідмовної роботи. Він також забезпечує автоматичне перемикання після відмови та своєчасне оновлення безпеки;

в) Оптимізація запитів. Інструменти або методи оптимізації бази даних можуть допомогти виправити навантаження на сервер, лише якщо веб-сайти та програми добре закодовані. Але програмне забезпечення для балансування навантаження SQL-сервера може допомогти в більшій мірі. Це універсальне рішення для підтримки безвідмовної роботи, узгодженості даних, підвищення продуктивності та зниження витрат на обслуговування.

Крім того, це забезпечує постійну доступність для покращеного досвіду клієнтів [33]. Коротше кажучи, MySQL робить усе: від запуску перевірок працездатності до зменшення часу очікування запиту та рівномірного розподілу навантаження між декількома серверами. Отже, давайте розглянемо основні переваги MySQL та чому я зупинився саме на цьому рішенні:

— Відкритий код. MySQL є одним із найпопулярніших варіантів для організацій або компаній щодо програмного забезпечення як послуги. Його видання для спільноти є у вільному доступі для всіх, хто може використовувати та змінювати, пропонуючи чудову швидкість, масштаб і надійність. Це може бути надзвичайно корисним, особливо коли підприємства хочуть уникнути сплати ліцензійних зборів. Оскільки вихідний код доступний для перегляду та зміни будь-ким, розробники можуть вносити зміни у своє програмне забезпечення відповідно до своїх потреб. Ця гнучкість може стати в нагоді підприємствам із унікальними вимогами або якщо є потреба інтегрувати програмне забезпечення з іншими інструментами чи системами;

— Інтеграція з Laravel. Фреймворк Laravel має багато професійних пакетів та рішень для розробки з базами даних MySQL. «Generating Migrations», «Writing Seeders» саме ті технології, які Laravel пропонує програмістам для роботи з реляційними базами даних, а саме з MySQL;

— Безпека даних. MySQL є найбезпечнішою системою керування базами даних у світі. Остання версія MySQL пропонує захист даних і підтримку обробки транзакцій, що може значно принести користь будь-якому бізнесу, особливо підприємствам електронної комерції, які часто здійснюють грошові операції;

— Масштабованість на вимогу. Масштабованість за вимогою є відмінною рисою MySQL. Він керує глибоко вбудованими програмами, використовуючи невелику площу, навіть у базах даних, які зберігають терабайти даних. Крім того, MySQL пропонує індивідуальні рішення для підприємств електронної комерції з певними вимогами до баз даних;

— Більш висока ефективність. MySQL має кілька унікальних функцій, включаючи спеціальне програмне забезпечення системи зберігання. Це дозволяє системним адміністраторам налаштувати сервер бази даних MySQL для бездоганної роботи. Немає значення, чи це веб-додаток для електронної комерції, що отримує мільйон запитів щодня, чи високошвидкісна система обробки транзакцій. MySQL створено, щоб задовольнити зростаючі вимоги майже кожної програми та забезпечити повнотекстові індекси, оптимальну швидкість і чіткі кеші для покращеної

продуктивності;

— Час роботи сервера 24×7. MySQL гарантує безвідмовну роботу 24/7. Він також пропонує широкий спектр рішень для баз даних високої доступності, включаючи конфігурації реплікації головного/підпорядкованого та спеціалізовані кластери серверів;

— Повна підтримка транзакцій (Database Transactions MySQL). MySQL є рушієм транзакційних баз даних номер один у світі. Його функції включають повну атомарну, послідовну, ізольовану, довговічну та багатоверсійну підтримку транзакцій, а також необмежене блокування на рівні рядків. Завдяки цим унікальним функціям MySQL є універсальним рішенням для комплексної цілісності даних, яке забезпечує миттєву ідентифікацію взаємоблокувань за допомогою контрольованої сервером посилальної цілісності;

— Комплексний контроль робочого процесу. MySQL простий у використанні, середній час завантаження та встановлення становить менше 30 хвилин. Крім того, не має значення, чи є ваша платформа Microsoft, Macintosh, Linux або UNIX. MySQL — це комплексне рішення з функціями самостійного керування. Ці функції автоматизують усе: від конфігурації та розширення простору до проектування даних і адміністрування бази даних.

Також, фреймворк Laravel має суміжну технологію з базами даних MySQL під назвою «Seeder». Бувають випадки, коли розробник створює програму, і вона має конфігураційні дані, які дозволяють програмі працювати ефективно. Ці дані будуть розміщені як вихідні дані, щоб забезпечити її доступність під час встановлення. Seeder – це клас, який використовується для вставлення даних у базу даних за допомогою виконання команди. Нижче я наведу три основні причини, через які використання Seeder набуває сенсу:

а) Конфігурація або налаштування. Надійні додатки мають значення конфігурації, від яких залежить додаток, і тому для спрощення розгортання необхідно мати засівач;

б) Користувачі за замовчуванням. З міркувань безпеки деякі розробники створюють свій обліковий запис адміністратора програми на льоту за допомогою

засівачів;

в) Демо дані. Для тестування програми нам потрібні демонстраційні дані. Під час розгортання ми можемо запустити наші сівери, щоб заповнити програму ефективними/демонстраційними даними для тестування.

РОЗДІЛ 3. ОПИС РОЗРОБКИ ВЕБ-ДОДАТКУ ТА ДЕМОНСТРАЦІЯ РЕЗУЛЬТАТІВ. КООПЕРАЦІЯ З БАЗАМИ ДАНИХ

3.1 Визначення основних функціональних вимог клієнта до web-додатку автосалону

Розробка веб-додатку для автосалону передбачає врахування різних функціональних вимог, в першу чергу, для задоволення потреб користувачів. Грамотно розроблений додаток має включати в себе усі сучасні інтеграції та забезпечувати користувачам оптимізовану роботу та функціональність. В даному веб-додатку будуть застосовані такі інтерактивні аспекти:

— Каталог автомобілів: Перелік автомобілів, їх фільтрація за моделями та можливість зручно пересуватися між ними;

— Детальні характеристики автомобіля: Технічні характеристики, комплектації, фото та можливість надання певної цінки автомобілю, що згодом впливає на рейтинг авто в межах застосунку;

— Можливість пошуку авто, замовлення тест-драйву та переглядання авто в найближчому для клієнта автосалоні з чотирьох доступних у місті Київ (A, B, C, D).

— Кожен клієнт може зареєструватися в даному веб-застосунку та заходити в особистий кабінет. Це дозволить розробнику зручніше працювати з клієнтами, а клієнтам надасть можливість вибирати найзручніший для нього автосалон та замовлення в нього тест-драйву та інших привілегій. Історія покупок і збереження улюблених автомобілів.

— Контакти та опис компанії. Кожен клієнт може у вільному доступі скористатится інформацією про компанію розробника та знайти контактні дані автосалону.

— Мобільна сумісність. Розроблений веб-додаток налагоджено оптимізовується під різні популярні мобільні девайси та планшети.

3.2 Розробка та демонстрація результатів

Встановивши фреймворк Laravel та бази даних MySQL приступаємо до безпосереднього разгортання середовища розробки [34]. Структура являє собою безліч папок, з певними файлами коду і не тільки. Частина структури даного проекту буде наведена на рисунку 3.1.

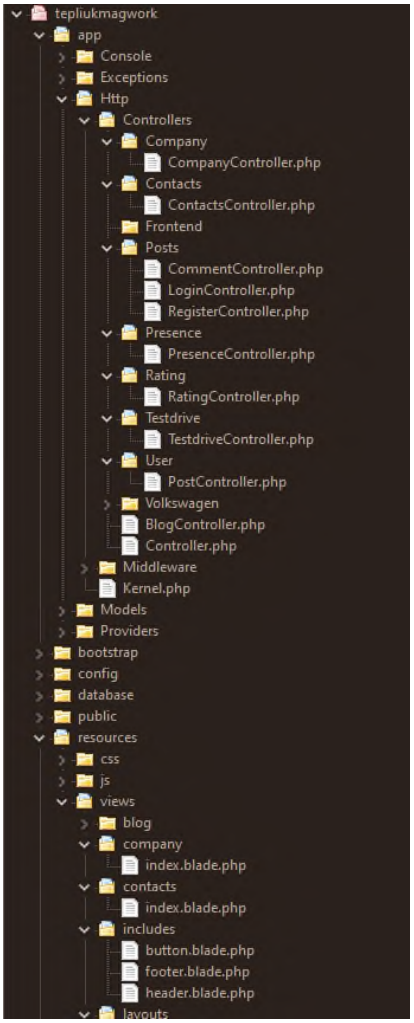


Рис. 3.1 Структура файлової частини Laravel

Структура веб-застосунку налічує в собі безліч папок, з певним функціоналом. Цей функціонал поділяється, як на фронт-енд так і на бек-енд. Зауважу, що в даній дипломній роботі я особисто займався і фронт-ендом і бек-ендом, в той час як зазвичай, при розробці веб-застосунків ці задачі розділяють на двох або більше людей, в залежності від складності. Тобто, розробка даного веб-

застосунку включає в себе повний стек технологій необхідних для веб-програмування.

Отже, розпочнемо з головної сторінки. За базу автосалону було взято уявну компанію «TepliukVagPremium», що класифікується на продажі преміальних моделей автівок Volkswagen, Audi та Porsche. На рисунку 3.2 зображено головну сторінку розробленого веб-застосунку, а саме його Header. Зазвичай Headr являє собою шапку будь-якого веб-додатку, яка включає в себе основні кнопки розроблені для взаємодії користувача з програмою.

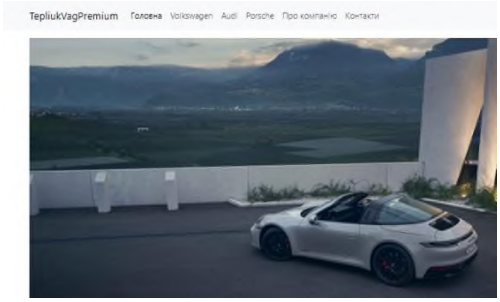


Рис. 3.2 Header

Головна сторінка веб-додатку автосалону включає в себе слайдер з усіма преміальними автівками в наявності а також інтерактивні новини, такі як «Акції» та «Анонси» нових автомобілів. На рис. 3.3 зображена повна верхня частина головної сторінки.

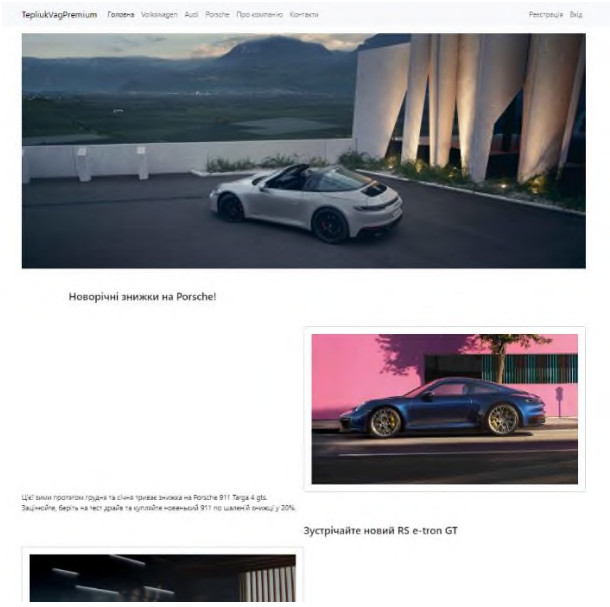


Рис. 3.3 Головна сторінка

Наведене розташування кнопок взаємодії в Header відбвується шляхом фронт-енд верстування додатку [22-24]. Створення та підв’язування кнопок до маршрутів та посилань зразково наведено на рис. 3.4. Також, на рис. 3.5 зображено футер головної сторінки з анонсом нового автомобіля та важливою інформацією.

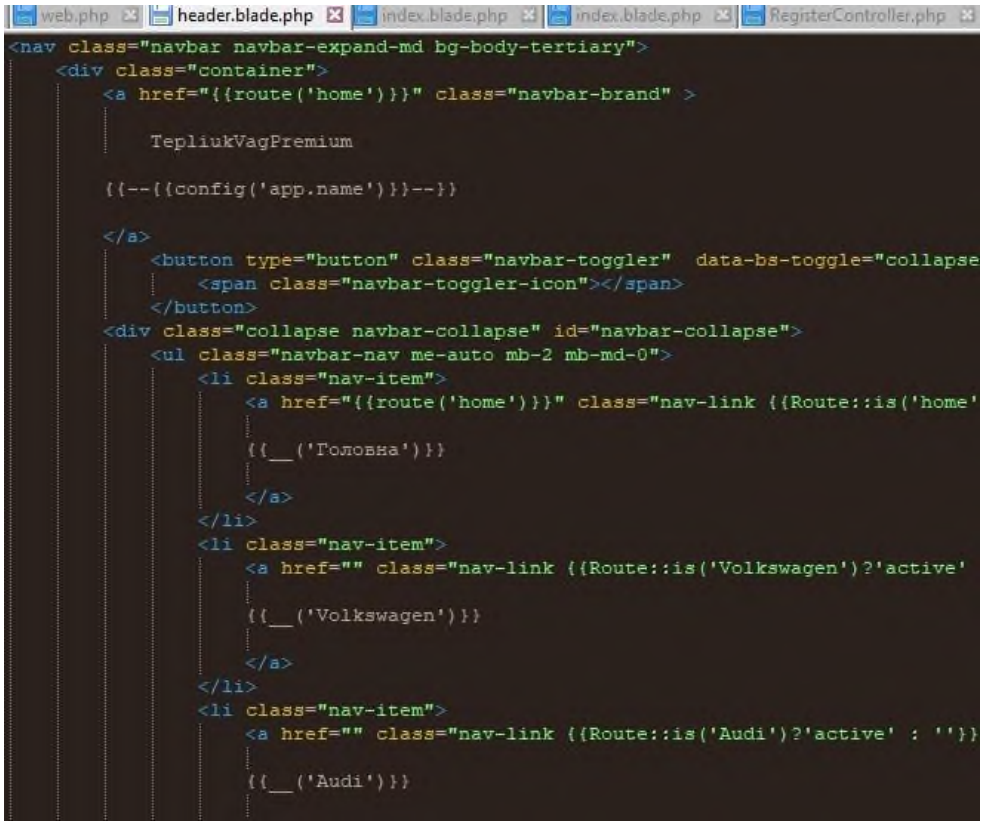


Рис. 3.4 Створення кнопок Header`у

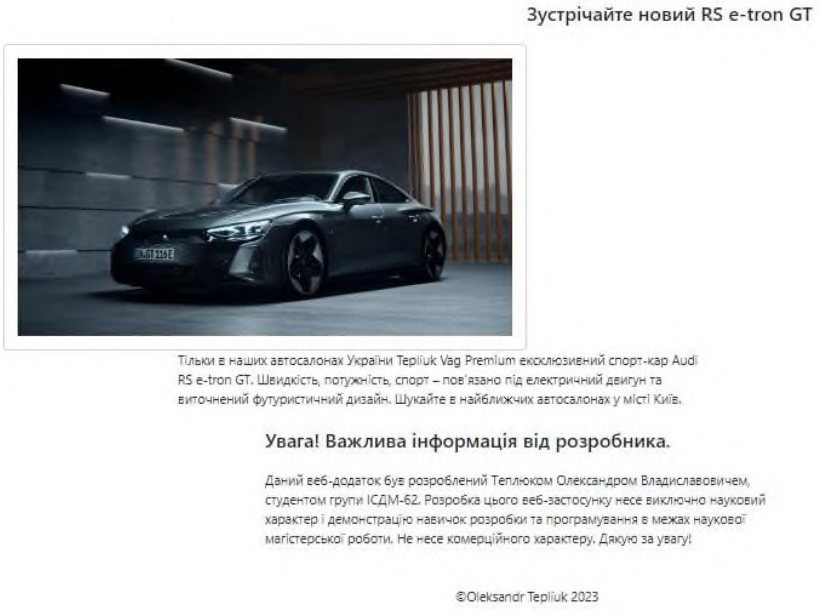


Рис. 3.5 Футер головної сторінки

Перейдемо безпосередньо до ствоерння та проектування бази даних MySQL [24]. На рис. 3.6 зображено побудовану базу даних під назвою «tepliukmagwork» та сгенерована в ній таблиця, що відноситься до автомобілів «volkswagen».

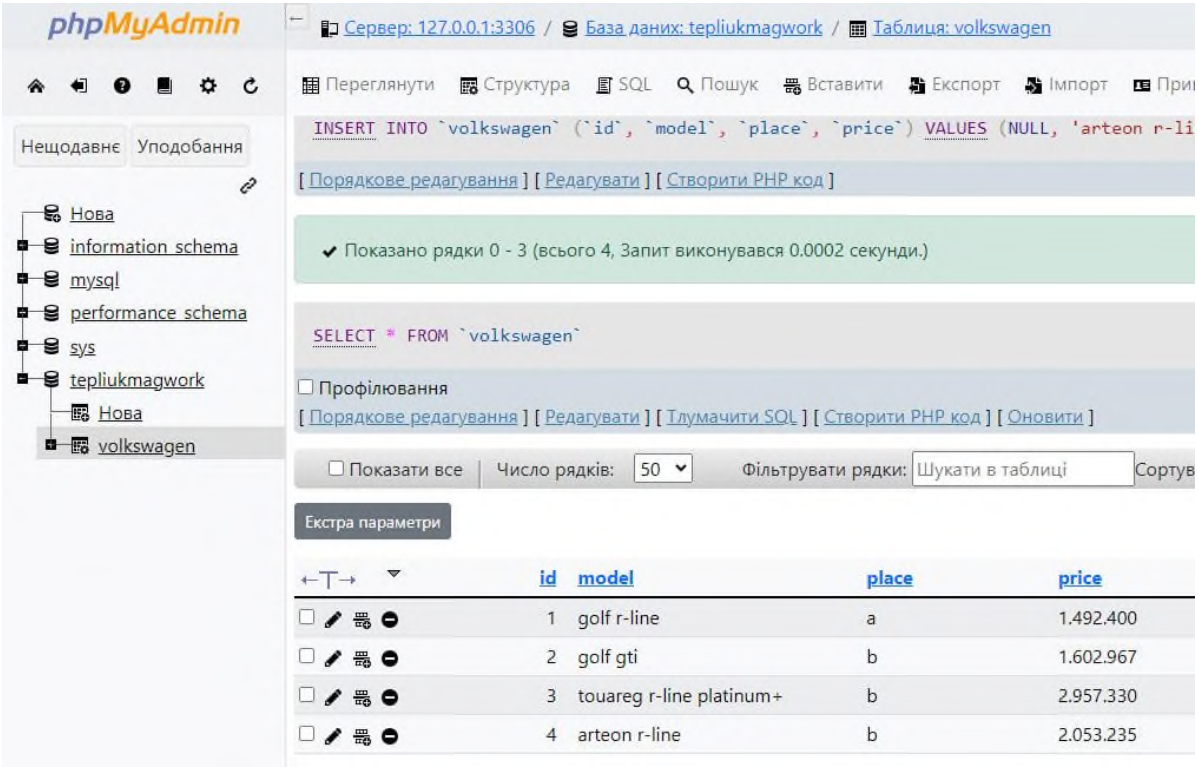


Рис. 3.6 База даних з однією таблицею

В даній таблиці бази даних наведено всю інформацію по автомобілям volkswagen. Отже, ми маємо:

— id – унікальний ідентифікатор автомобіля;

— model – назва моделі авто;

— place – місце розташування в наявності;

— price – ціна на певне авто.

Аналогічне проєктування таблиць проводиться для кожної марки Audi та Porsche.

Також, обов’язковим є генерація таблиці користувачів. Назову її «our-users» та надам полям «email» та «phone» індексу «UNIQUE», щоб уникнути повторних реєстрацій та дублювання персональних даних. Результат наведено на рис. 3.7.

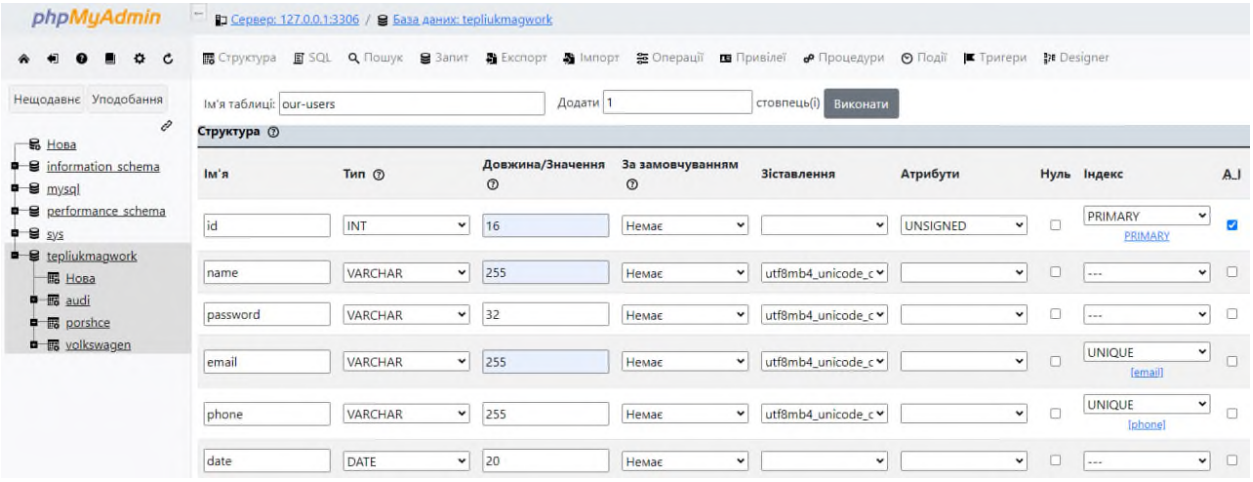


Рис. 3.7 Створення концепції таблиці користувачів

Важливий момент в заданні ідентифікатору «id» атрибуту «UNSIGNED» та індексу «PRIMARY» з включеним інкрементом. Це забезпечить те, що наш унікальний ідентифікатор завжди буде більше нуля та автоматично зростати на одиницю при додаванні нових полей. Це зручна, маленька особливість роботи з MySQL.

Сгенеруємо таблицю з розташуванням та адресами автосалонів. Ця таблиця буде зображена на рис. 3.8.

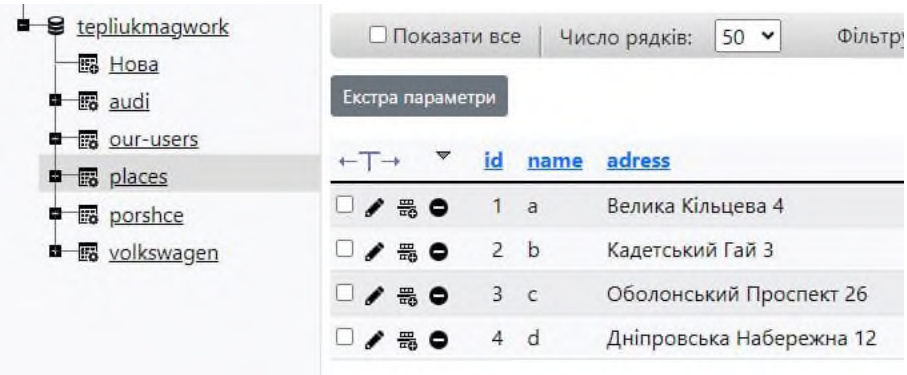


Рис. 3.8 Таблиця автосалонів

Завершуючи генерування таблиць баз даних, зазначу, що бази даних є одним з ключових елементів сучасного веб-додатку, без яких його функціонування неможливе. Отже, маємо кінцевий варіант нашої бази даних у вигляді зображеному на рис. 3.9.

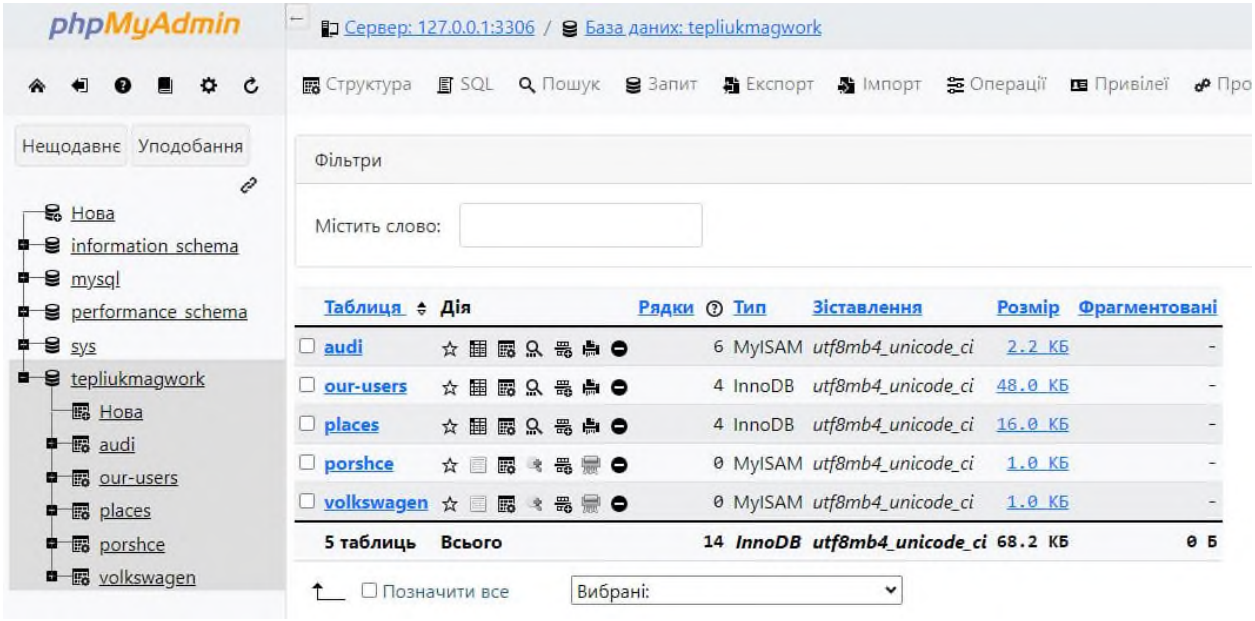


Рис. 3.9 Структура бази даних для програми

Продовжимо розробку сторінок веб-додатку. На слідуючій вкладці, пілся головної, було розроблено сторінки та функціонал пов’язаний з автомобілями volkswagen. Розроблено слайдер з фотографіями автомобілей в наявності, що зображено на рис. 3.10, а також нижче розташовані кнопки, які перекидують користувача на відповідні моделі автомобілів: Golf, Arteon та Touareg. При натисканні кнопки Golf, користувач переходить до наступної сторінки, де він може обрати потрібну йому класифікацію моделі Golf, що зображено на рис. 3.11, та далі знайти більш детальну інформації про автомобіль. При натисканні кнопки «R – Line» або «GTI» клієнт переходить в наступний розділ зображений на рис. 3.12, де відображається більш детальна інформація про обраний автомобіль.

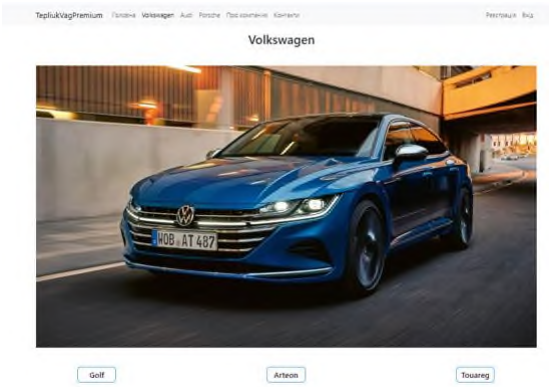


Рис. 3.10 Сторінка автомобілів volkswagen

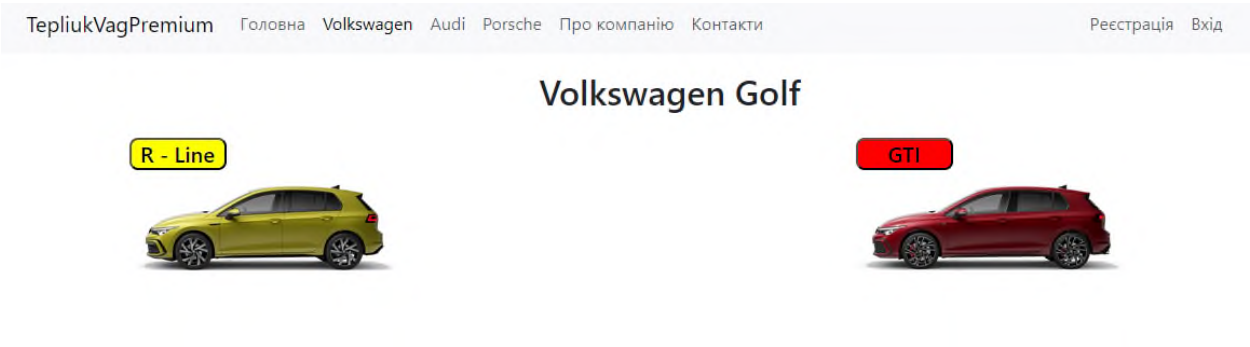


Рис. 3.11 Класифікація Golf

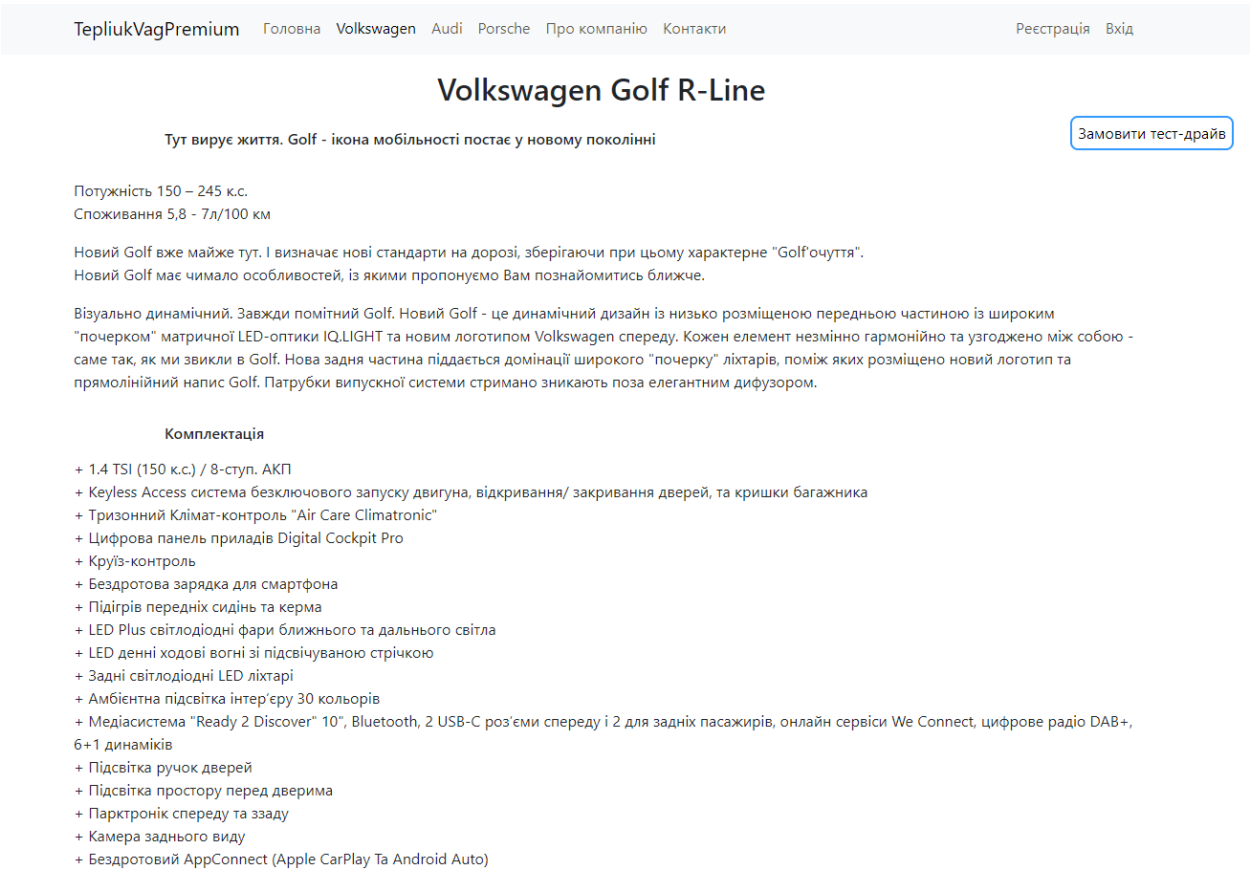


Рис. 3.12 Сторінка Golf R-Line

На рис. 3.13 зображена нижня частина сторінки з детальною інформацією про автомобіль. Також, на цій сторінці знаходяться дві кнопки «Дізнатись наявність» та «Оцінити авто». За допомогою них, клієнт може взаємодіяти з веб-додатком та у зручному форматі дізнаватись для себе важливої інформації. Звертаю увагу на кнопку «Замовити тест-драйф». Ця кнопка є мобільною та постійно знаходиться на екрані користувача, не залежачи від того в якій частині сторінки він знаходиться. Рис. 3.14 яскраво відображає, в яких автосалонах наразі знаходиться даний

автомобіль та в якій кількості. Кожен автосалон має свою адресу та свій унікальний ідентифікатор за літерою англійської абетки: A, B, C, D. Також, кількість вказується за для того, щоб якщо у випадку коли клієнтом виявиться крупна компанія, яка замовляє декілька авто для службових цілей, менеджери або замовники цієї організації змогли одразу побачити їх кількість в наявності. Аналогічна розробка сторінок була проводена для автомобілів Audi та Porsche.

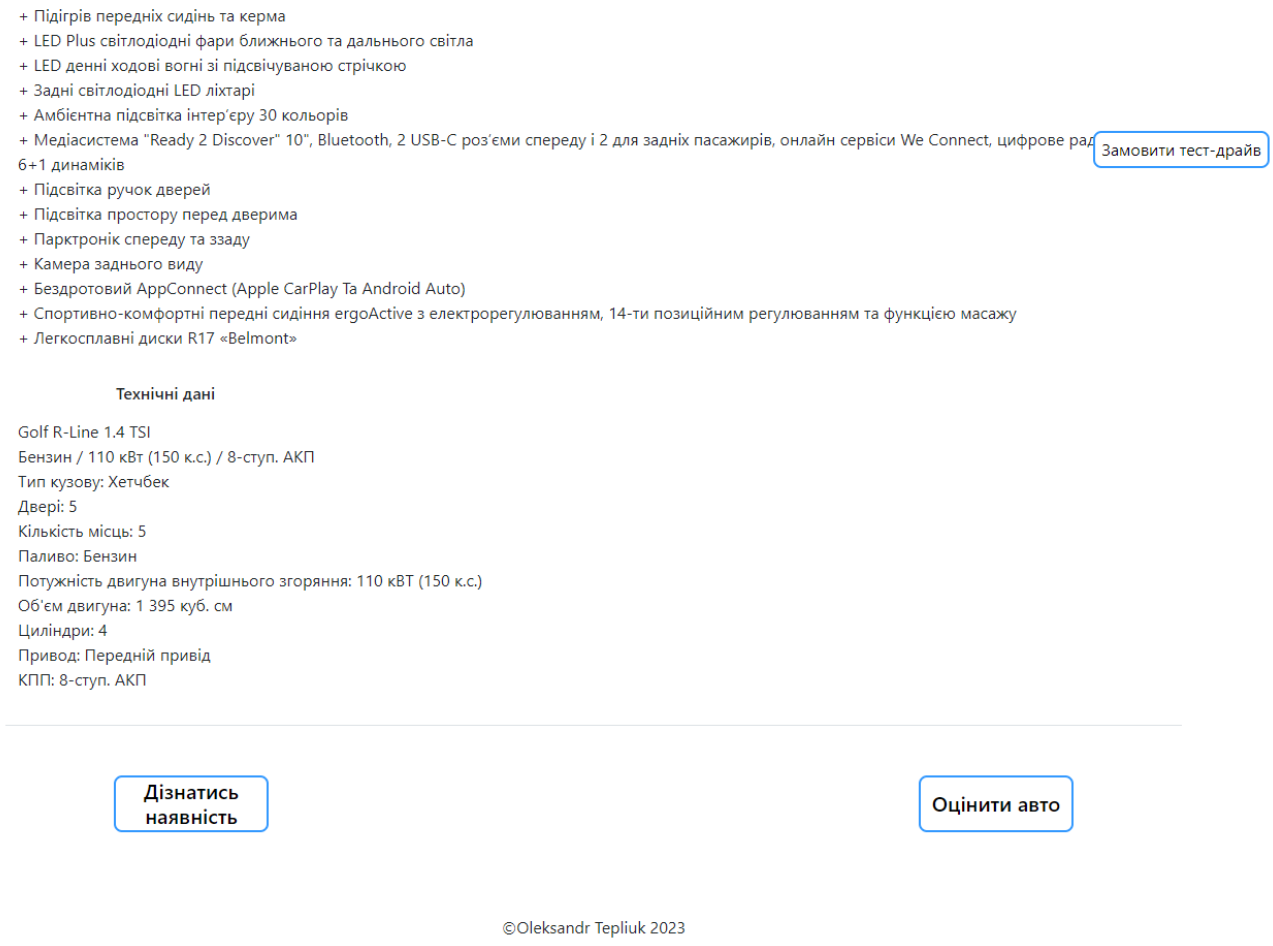


Рис. 3.13 Футер Golf R-Line

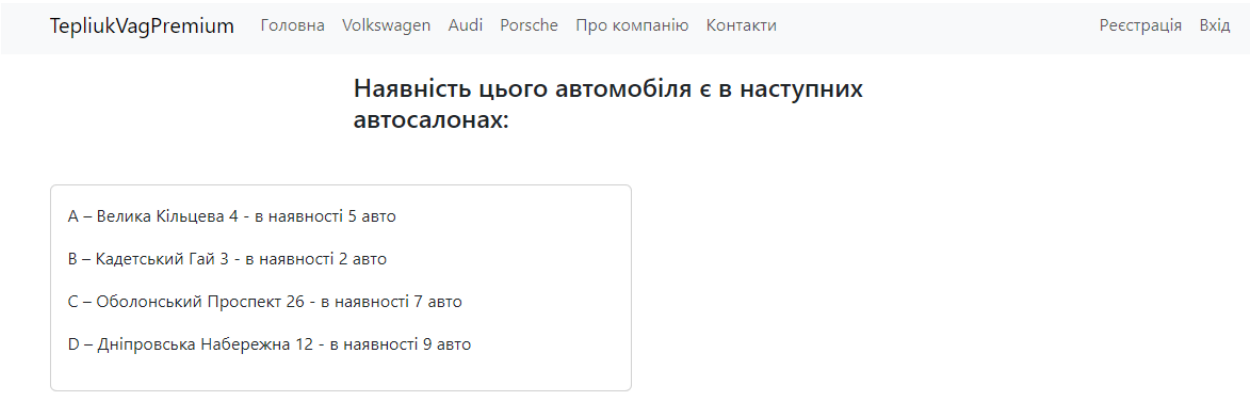


Рис. 3.14 Перевірка наявності

На рис. 3.15 та рис. 3.16 відповідно продемонстрована робота кнопок оцінювання та запису на тест-драйв. У випадку оцінки, користувач перетягує кружок виставляючи відповідну оцінку, яка змінюється з переміщенням курсору. При записі на тест-драйв, користувачеві одразу наглядно відкривається інтерактивний календар, завдяки якому клієнт може зручно підібрати під себе дату та вибрати її в одне натискання по обраному числу місяця.

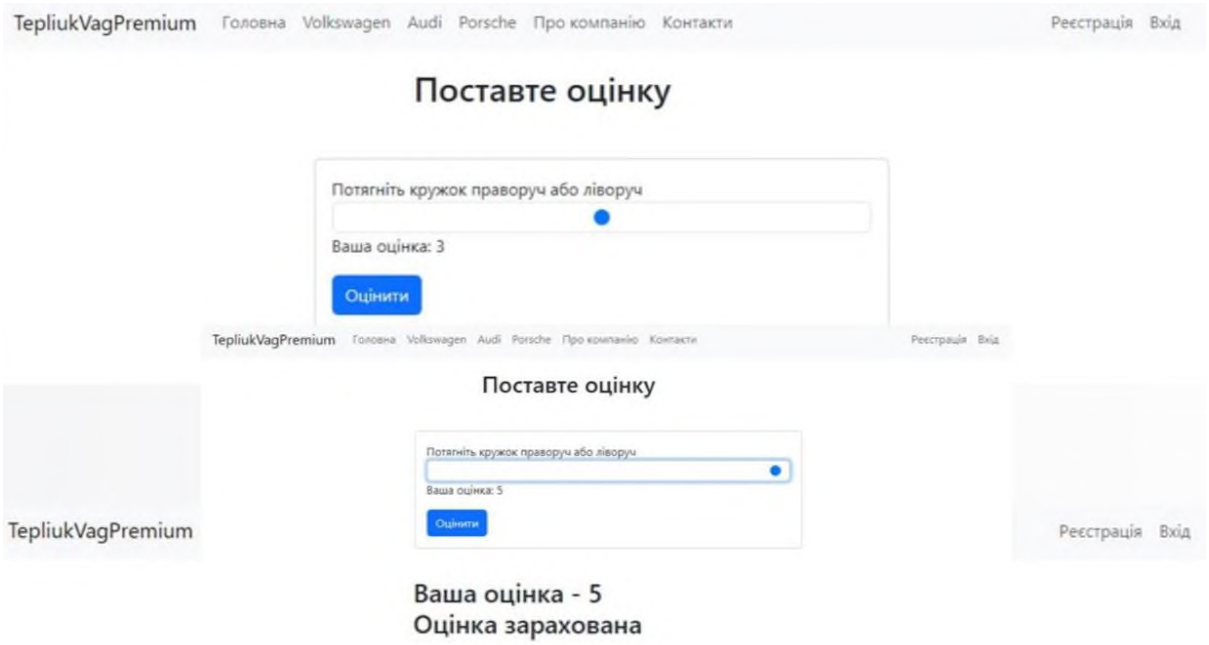


Рис. 3.15 Оцінка авто

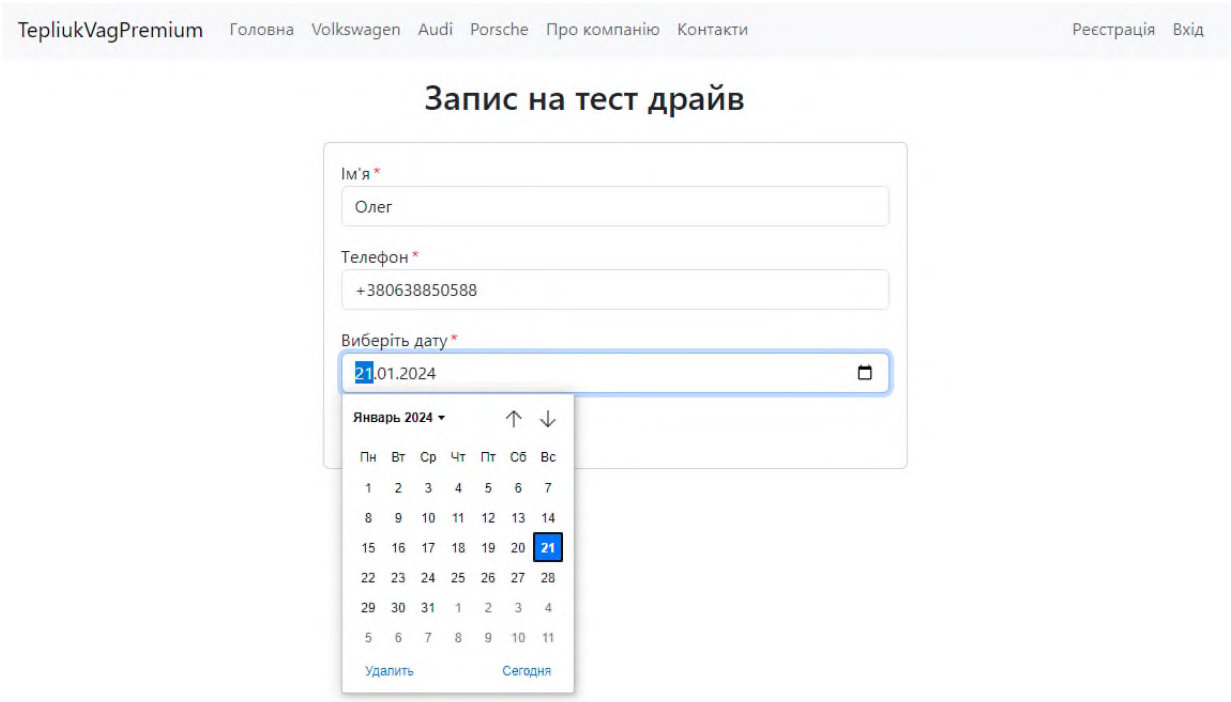


Рис. 3.16 Запис на тест-драйв

В даному додатку передбачено особистий кабінет клієнта. Це означає, що користувач може зареєструватися в розробленій системі, для подальшої зручності в роботі з програмою. Також, передбачено вхід в особистий кабінет. Ці дві кнопки «Реєстрація» та «Вхід» відображені на кожній сторінці веб-додатку, щоб кожен клієнт міг в будь-який час скористатися послугами особистого кабінету та здійснити вхід або реєстрацію. На рис. 3.17 та рис. 3.18 відповідно відображено роботу форм заповнення певних особистих даних користувача. Також, присутній маркер у вигляді галки про «надання згоди на обробки особистої інформації».

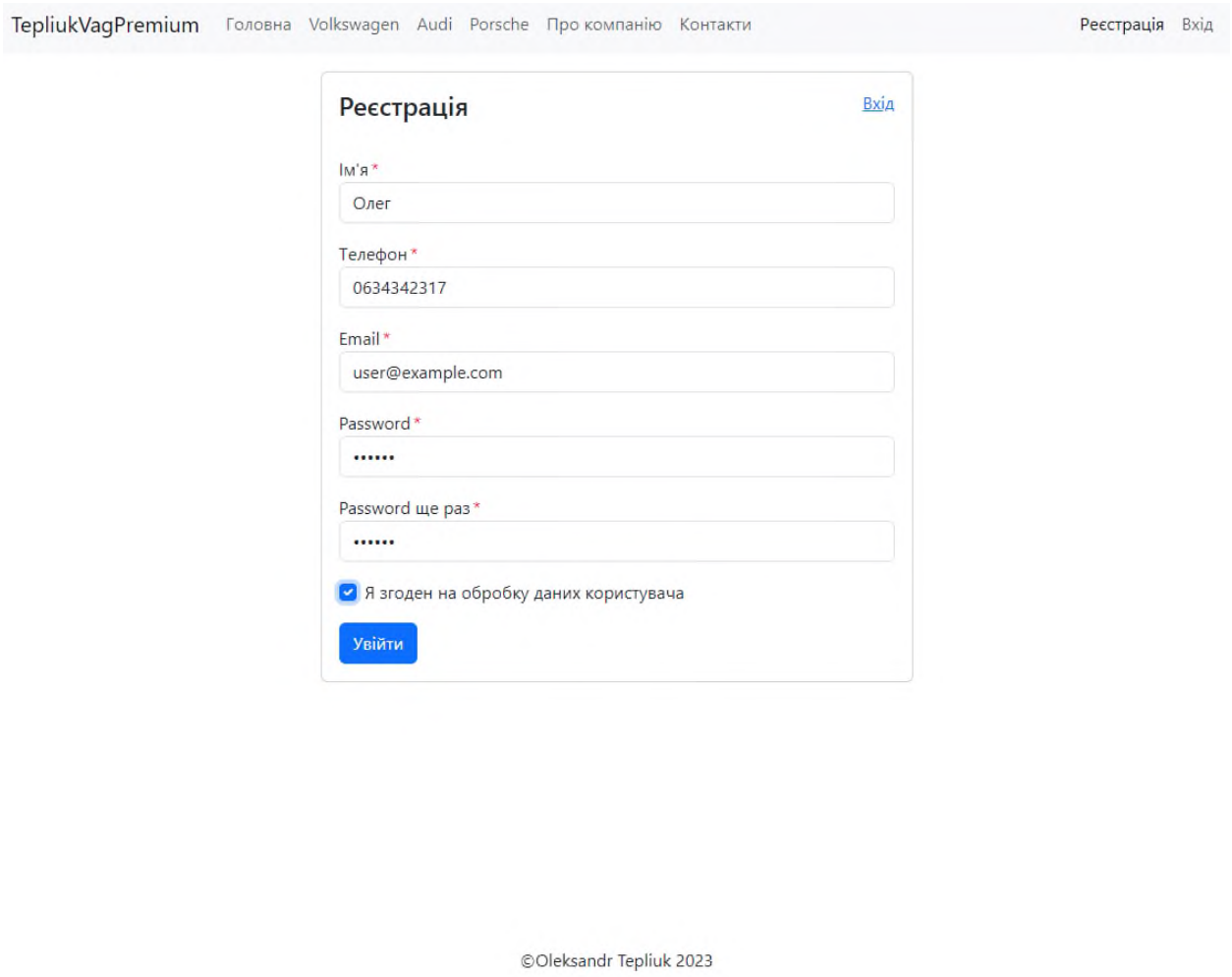


Рис. 3.17 Форма реєстрації

Шаблон для сторінки реєстрації, являє собою зразок динамічного шаблону, різного для кожної сторінки. В результаті цього нема дублювання коду на кожній сторінки та відбувається досягання принципу MVC.

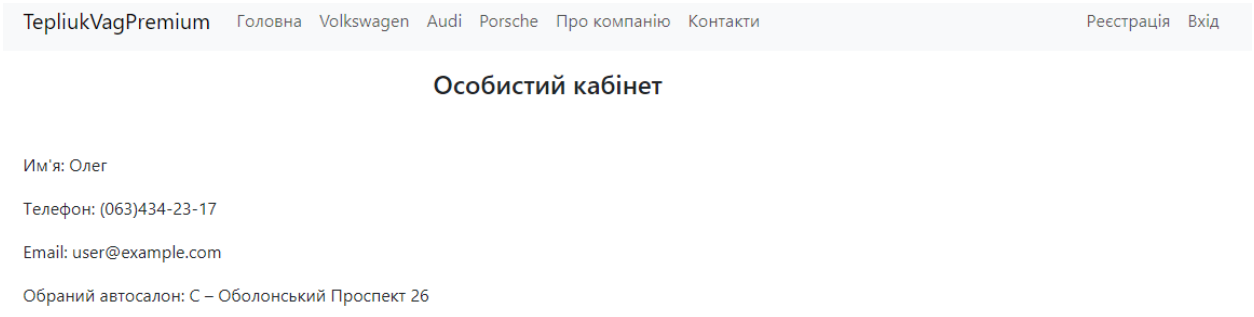
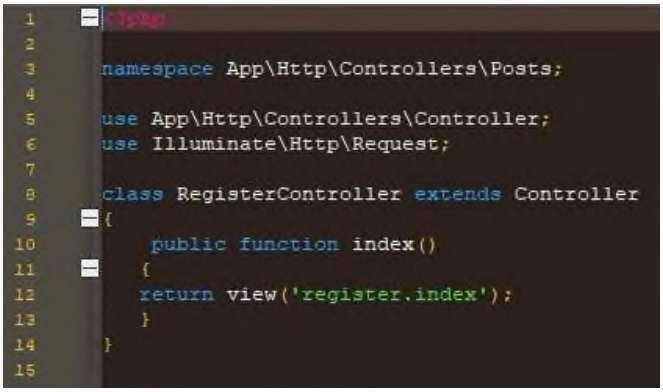


Рис. 3.18 Форма входу

Нижче на рис. 3.20 та рис. 3.21 будуть наведені фрагменти коду, до шаблону реєстрації. Це наглядна фронт-енд розробка, яка демонструє роботу зв’язків у

шаблоні реєстрації. Додатково, на рис. 3.19. буде зображено фрагменту контролеру програмованого за допомогою PHP та фреймворку Laravel.



```php

namespace App\Http\Controllers\Posts;

use App\Http\Controllers\Controller;
use Illuminate\Http\Request;

class RegisterController extends Controller
{
    public function index()
    {
    return view('register.index');
    }
}

```

Рис. 3.19 Контролер до реєстрації

Також, на рис. 3.21 відображені певні класи та маршрути, завдяки яким відбувається направлення певних даних з форми реєстрації далі на обробку та завантаження у бази даних. Більш детальний код до авторизації, реєстрації та роботи з базами даних буде наведено в додтаку А та Б.

```
@extends('layouts.auth')
@section('page.title','Registration')
@section('auth.content')
    <section>
                    <div class="card-body">
                        <div class="d-flex justify-content-between">
                            <div>
                                <h4 class="m-0">
                                    {{__('Реєстрація')}}
                                </h4>
                            </div>
                            <div>
                                <a href="{{route('login')}}">
                                    {{__('Вхід')}}
                                </a>
                            </div>
                        </div>
                    </div>
                    <div class="card-body">
                        <form action="{{route('register.store')}}" method="POST">
                        @csrf
                            <div class="mb-3">
                                <label class="required">{{__("Ім'я")}}</label>
                                <input type="text" name="name" class="form-control" autofocus>
                            </div>

                            <div class="mb-3">
                                <label class="required">{{__('Телефон')}}</label>
                                <input type="tel" name="tel" class="form-control">
                            </div>

                            <div class="mb-3">
                                <label class="required">{{__('Email')}}</label>
                                <input type="email" name="email" class="form-control">
                            </div>

                            <div class="mb-3">
                                <label class="required">{{__('Password')}}</label>
                                <input type="password" name="password" class="form-control">
                            </div>
                            <div class="mb-3">
                                <label class="required">{{__('Password ще раз')}}</label>
                                <input type="password" name="password_confirmation" class="form-control">
                            </div>
```

Рис. 3.20 Шаблон реєстрації (перша частина)

```
                        </div>

                        <div class="mb-3">
                            <label class="required">{{__('Email')}}</label>
                            <input type="email" name="email" class="form-control">
                        </div>

                        <div class="mb-3">
                            <label class="required">{{__('Password')}}</label>
                            <input type="password" name="password" class="form-control">
                        </div>
                        <div class="mb-3">
                            <label class="required">{{__('Password ще раз')}}</label>
                            <input type="password" name="password_confirmation" class="form-control">
                        </div>
                        <div class="mb-3">
                            <div class="form-check">
                                <input type="checkbox" name="agreement"  value="1" class="form-check-input"  id="remember">
                                <label class="form-check-label" for="agreement">
                                    {{__('Я згоден на обробку даних користувача')}}
                                </label>
                            </div>
                        </div>
                        <button  type="submit"  class="btn btn-primary">
                            {{__('Увійти')}}
                        </button>

                    </form>
                </div>


        </section>

    @endsection

```

Рис. 3.21 Шаблон реєстрації (друга частина)

Нагадучи про те, що даний веб-додаток розроблено з оптимізацію під мобільні пристрої, на рис. 3.22 буде відображення та сама форма реєстрації користувача, але вже на мобільному пристрої. За зразок було взято iPhone 14 Pro Max, враховуючи те, що цей смартфон є доволі сучасним та популярним.

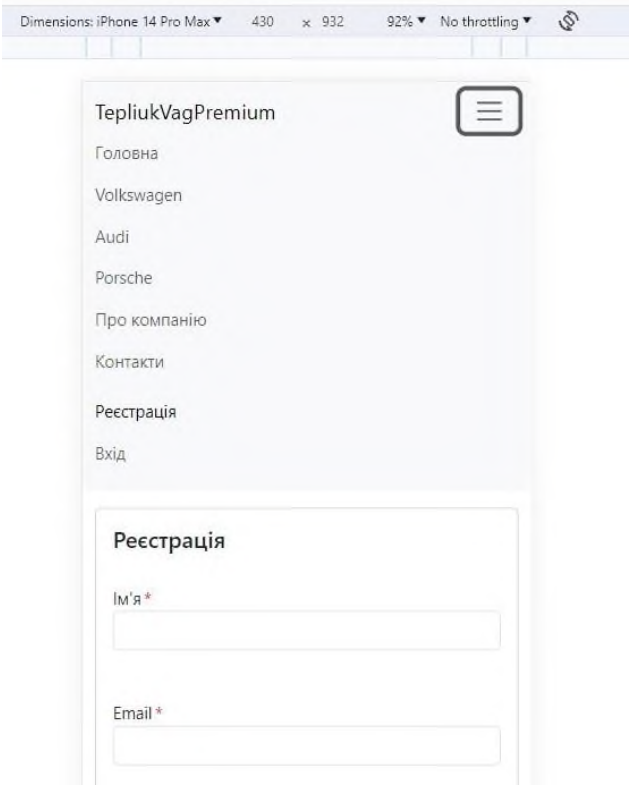


Рис. 3.22 Відображення на смартфоні

Веб-додаток для клієнтів, який розроблений як автосалон, має мати достовірну інформацію про свою компанію а також контактні дані. На сторінках «Про компанію» та «Контакти», зручно та доступно розташована ця інформація. Результат відпрацювання даних сторінок зображений на рис. 3.23 та рис. 3.24.

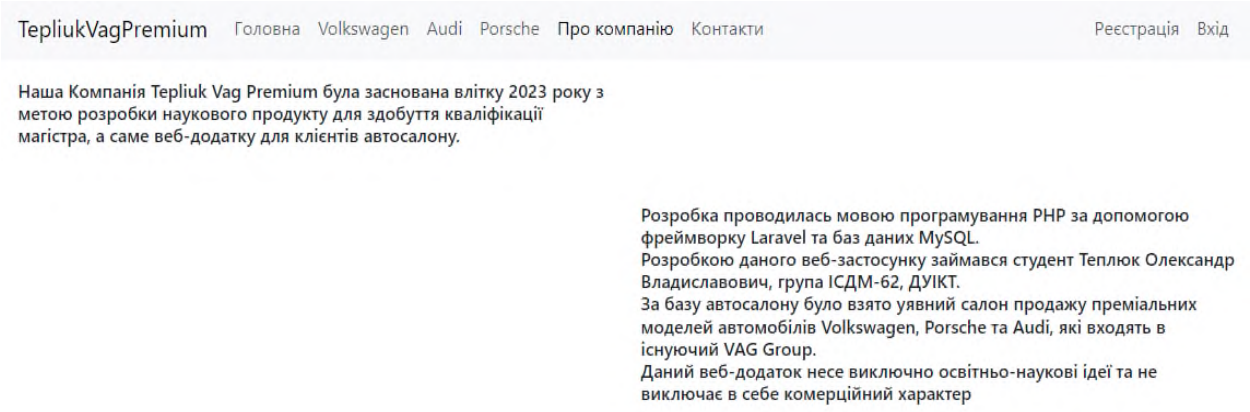


Рис. 3.23 Сторінка «Про компанію»

На сторінці «Про компанію» описано початок зародження ідеї розробки та основні технології застосовані в даному веб-застосунку. Сторінка «Контакти» слугує певним навігатором для клієнтів. На ній зображено мапу міста Київ, в якому знаходиться дана компанія, та ідентифікаційні літери відповідних автосалонів.

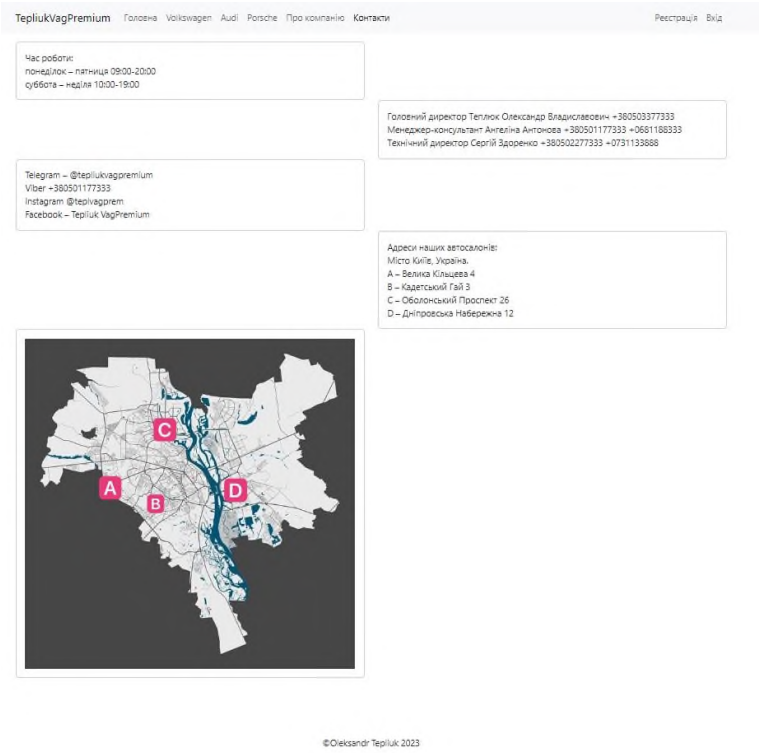


Рис. 3.23 Сторінка «Контакти»

Також, користувач може розшифрувати ту чи іншу літеру та дізнатися адресу зручного для нього автосалону. Наведено контактну інформацію співробітників та дані щодо соціальних мереж безпосередньо компанії. Можливо, деякі клієнти звикли спілкуватися через соціальні мережі та месенджери. Дана опція існує спеціально для такого типу користувачів, що поліпшить їх взаємодію з веб-застосунком та співпрацю з компанією в цілому.

ВИСНОВКИ

Під час аналізу, розробки та дослідження даної дипломної роботи було виконано всі поставлені задачі, розглянуто сучасні тенденції та технології веб-розробки на PHP, проведено аналіз існуючих методів та розробницьких рішень для програмування веб-додатків для клієнтів. Було обгрунтовано обрано фреймворк Laravel та базу даних MySQL, як технології для розробки на мові програмування PHP. Шляхом аналізу даних середовищ, було з'ясовано, що вони є найкращими варіантами для сучасної веб-розробки. Розроблено архітектуру програмного забезпечення. Для реалізації поставленої задачі було вирішено використовувати мову програмування PHP. В ході роботи було реалізовано інтерфейс користувача, спроектовано базу даних та розроблено серверну частину додатку для взаємодії з базою даних. Даний веб-додаток дозволяє клієнтам автосалону, з використанням браузеру, продивитися автомобілі, їх характеристики та зовнішний вигляд, зробити замовлення та скористуватися тест-драйвом. Також, було створено реєстрацію користувачів в особистий кабінет, який надає додаткові можливості взаємодії та систему рейтингу автомобілів. Усі призначені завдання на дипломний проект було вирішено. Розроблений веб-додаток має перспективи подальшого розвитку та вдосконалення.

У першому розділі було проаналізовано різноманітні типип веб-додатків, їх переваги та недоліки. Долсіджено відмінності між веб-сайтом та веб-додатком, наведено приклади популярних програм. Висвітлено той факт, що в сучасності веб-додатки використовуються в різних сферах нашого повсякденного життя. Було затверджено вибір багатосторінкового веб-додатку для подальшої розробки проекту, спираючись на проведений аналіз то дослідження. Також, проаналізовано протоколи http та http`s, технології TLS та детально дослідженно технологію оптимізації SEO. Наприкінці першого розділу були наведіні пропозиції щодо дослідження IoT на майбутнє та сучасні тенденції в цьому середовищі.

У другому розділі було детально проаналізовано фреймворк Laravel та бази даних MySQL, з'ясовано чому обрані саме ці технології та наведені їх ключові переваги та можливості розробки.

У третьому розділі було розроблено безпосередньо сам веб-додаток. За допомогою фреймворку Laravel, мови програмування PHP та JavaScript було спрограмовано фронт-енд і бек-енд даного проекту. Спроектована та реалізована інтерфейсна частина веб-додатку для клієнтів автосалону. Запропоновані певні моделі автомобілів, їх характеристи та можливість взаємодіяти з ними. Були створені та протестовані форми реєстрації, авторизації та функціонал рейтингу автомобілів.

Наприкінці можна впевнено сказати, що веб-додаток для клієнтів, розроблений з використанням фреймворку Laravel та мови програмування PHP, дозволить власнику та розробникам значно збільшити продажі товарів і послуг, розширити кількість клієнтів, скоротити час процесу від товару до його продажу. Також, концепція веб-застоссунків допоможе покращити онлайн-менеджмент магазинів, знижуючи витрати на оренду або купівлю торгових площ, приміщень та наймаючи велику кількість персоналу. Сьогодні існує маса технологій для розробки веб-додатків, і щоб розробити найякісніший з них, потрібно використовувати і застосовувати новітні сучасні інструменти для їх створення, а саме ті, що було проаналізовано та досліджено в даній дипломній роботі.

ПЕРЕЛІК ПОСИЛАНЬ

1. Олексій Васильєв. Програмування мовою PHP. Ліра-К, 2022. 233 с.
2. І.Л. Бородкіна, Г.О. Бородкін. Web-технології та Web-дизайн: застосування мови HTML для створення електронних ресурсів. Ліра-К, 2020. 111 с.
3. Роберт Мартін. Чиста архітектура: мистецтво розробки програмного забезпечення. Фабула, 2019. 201 с.
4. Патрік Дебуа, Джон Вілліс, Джин Кім. Посібник із DevOps. Фабула, 2023 рік. 304 с.
5. Роберт Мартін. Чистий кодер: Кодекс поведінки для професійних розробників. Фабула, 2023. 156 с.
6. Barry Pollard. HTTP 2 in Action. Maning Publications, March 2019. pp. 234– 401.
7. Kristina Halvorson, Melissa Rach. Content Strategy for the WEB, 2nd Edition. New Riders, February 2018. 104 c.
8. Eric Mann. PHP Cookbook: Modern Code Solutions for Professional Developers 1st Edition. O'Reilly, 2023. pp. 102– 343.
9. Robin Nixon. Learning PHP, MySQL & JavaScript 6th ed. O'Reilly Media, 2021. pp. 345– 730.
10. Matt Zandstra. PHP 8 Objects, Patterns, and Practice: Mastering OO Enhancements, Design Patterns, and Essential Development Tools 6th ed. Edition. Apress, April 2021. pp. 220– 833.
11. Bennie Albriton. Laravel Basics: How To Use Laravel To Build Your Own App. Independently published, December 2022. 34 c.
12. Mike McGrath. PHP and MySQL in Easy Steps 2nd Edition. In Easy Steps, 2018. 132 c.
13. A. Scholtens. Mastering Laravel. Sas155, February 2023. 54 c.
14. Jennifer Robbins. Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics 5th Ed. O'Reilly, 2018. pp. 344– 567.
15. Domain-Driven Laravel: Learn to Implement Domain-Driven Design Using Laravel. Jesse Griffin. 2021 by APress. 432 c.

16. Heli Helskyaho, Jean Yu, Kai Yu. Machine Learning for Oracle Database Professionals: Deploying Model-Driven Applications and Automation Pipelines. APress, 2021. 133 c.
17. Matt Stauffer. Laravel: Up & Running: A Framework for Building Modern PHP Apps 2nd Edition. O'Reilly, 2019. pp. 89– 511.
18. Hypertext Transfer Protocol Version 2 (HTTP/2) Documentation — [Електронний ресурс]. – Режим доступу URL: https://httpwg.org/specs/rfc7540.html (дата звернення 20.10.2023).
19. Що таке веб додаток? Різниця PWA та SPA — [Електронний ресурс]. – Режим доступу URL: https://webcase.com.ua/uk/ (дата звернення 22.10.2023).
20. Developing Web Applications with PHP — [Електронний ресурс]. – Режим доступу URL: https://www.zend.com/resources/developing-web-applications-php (дата звернення 24.10.2023).
21. The vital role of PHP in Web Application Development — [Електронний ресурс]. – Режим доступу URL: https://www.orchestrate.com/blog/the-vital-role-of-php-in-web-application-development/ (дата звернення 26.10.2023).
22. Notepad++ Manual — [Електронний ресурс]. – Режим доступу URL: https://npp-user-manual.org/ (дата звернення 28.10.2023).
23. PHP Manual Documentation — [Електронний ресурс]. – Режим доступу URL: https://www.php.net/manual/en/ (дата звернення 01.11.2023).
24. PHP MyAdmin Documentation — [Електронний ресурс]. – Режим доступу URL: https://www.phpmyadmin.net/docs/ (дата звернення 02.11.2023).
25. MySQL Documentation — [Електронний ресурс]. – Режим доступу URL: https://dev.mysql.com/doc/ (дата звернення 03.11.2023).
26. Laravel Documentation — [Електронний ресурс]. – Режим доступу URL: https://laravel.com/docs/ (дата звернення 04.11.2023).
27. SEO article — [Електронний ресурс]. – Режим доступу URL: https://www.tribeseo.com/ (дата звернення 06.11.2023).

28. Joe Williams abot SEO — [Електронний ресурс]. – Режим доступу URL: https://digitalmarketinginstitute.com/resources/ebooks/ebook-a-guide-to-seo-for-small-business?bloglink (дата звернення 09.11.2023).
29. The world's most popular network protocol analyzer — [Електронний ресурс]. – Режим доступу URL: https://www.wireshark.org/ (дата звернення 11.11.2023).
30. Learn about MySQL — [Електронний ресурс]. – Режим доступу URL: www.builtin.com (дата звернення 12.11.2023).
31. Command line tool and library for transferring data with URLs — [Електронний ресурс]. – Режим доступу URL: https://curl.se/ (дата звернення 23.11.2023).
32. GNU Operating Systems — [Електронний ресурс]. – Режим доступу URL: https://www.gnu.org/software/wget/ (дата звернення 24.11.2023).
33. Debugging and Troubleshooting — [Електронний ресурс]. – Режим доступу URL: https://www.telerik.com/fiddler (дата звернення 25.11.2023).
34. PHP The Right Way — [Електронний ресурс]. – Режим доступу URL: https://phptherightway.com/ (дата звернення 26.11.2023).

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

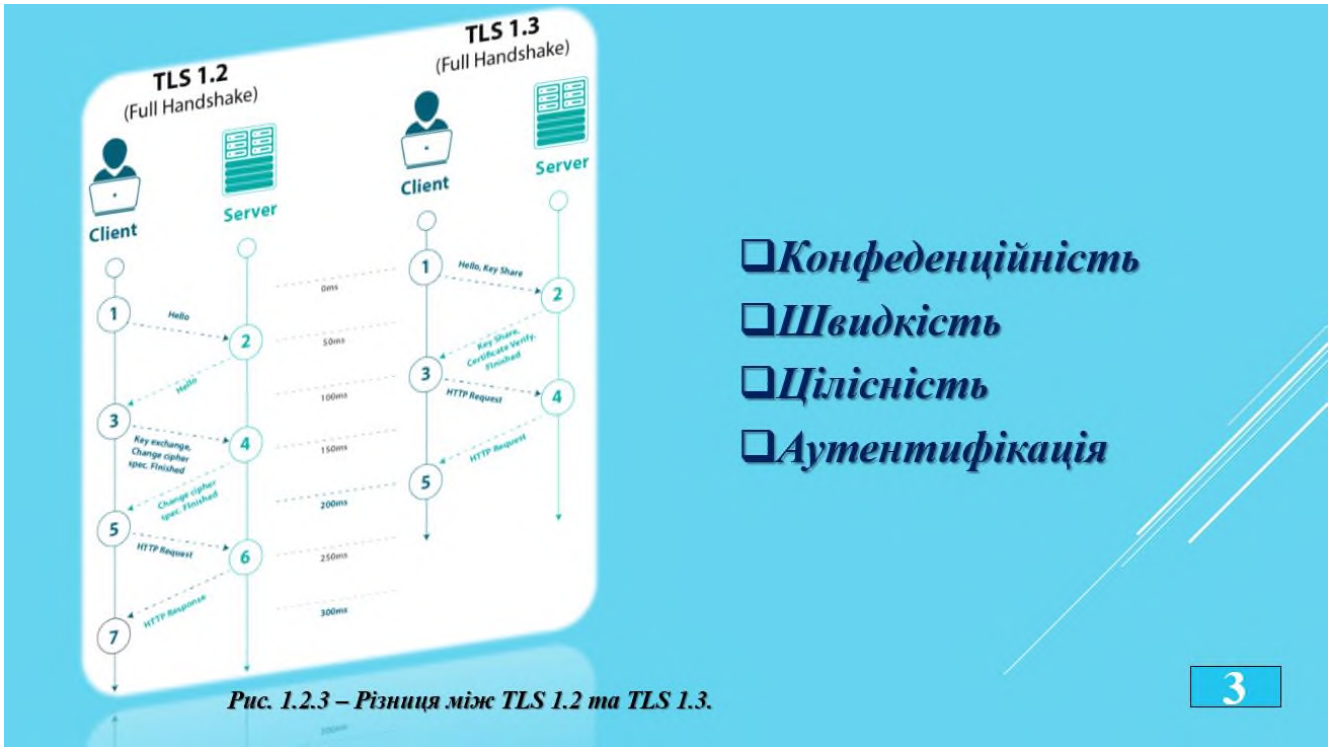


Рис. 1.2.3 – Різниця між TLS 1.2 та TLS 1.3.

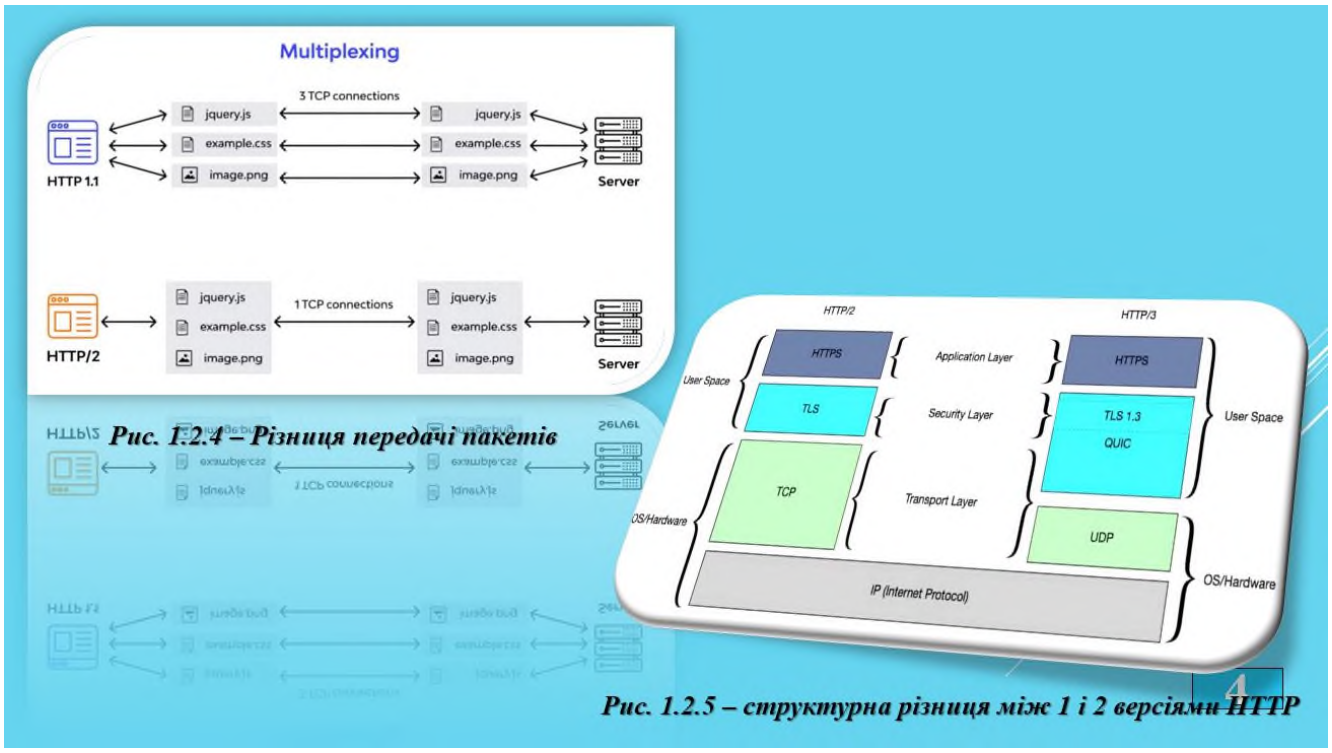


Рис. 1.2.4 – Різниця передачі пакетів

Рис. 1.2.5 – структурна різниця між 1 і 2 версіями HTTP

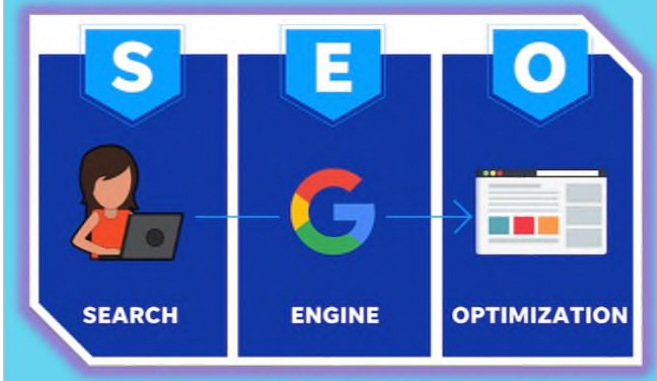


Рис. Рисунок 1.3.1 – Розшифровка SEO

Чому на це слід звернути увагу?

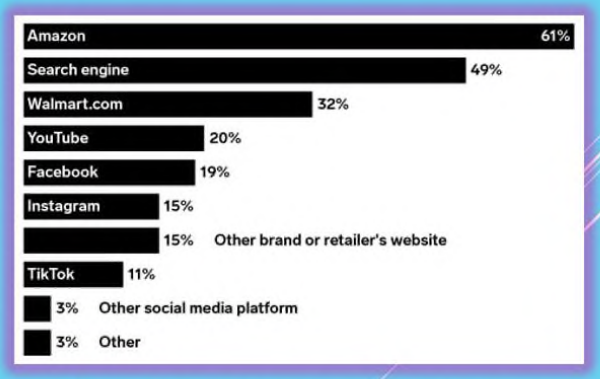


Рисунок 1.3.3 – Аналіз пошукових сервісів користувачів.

5

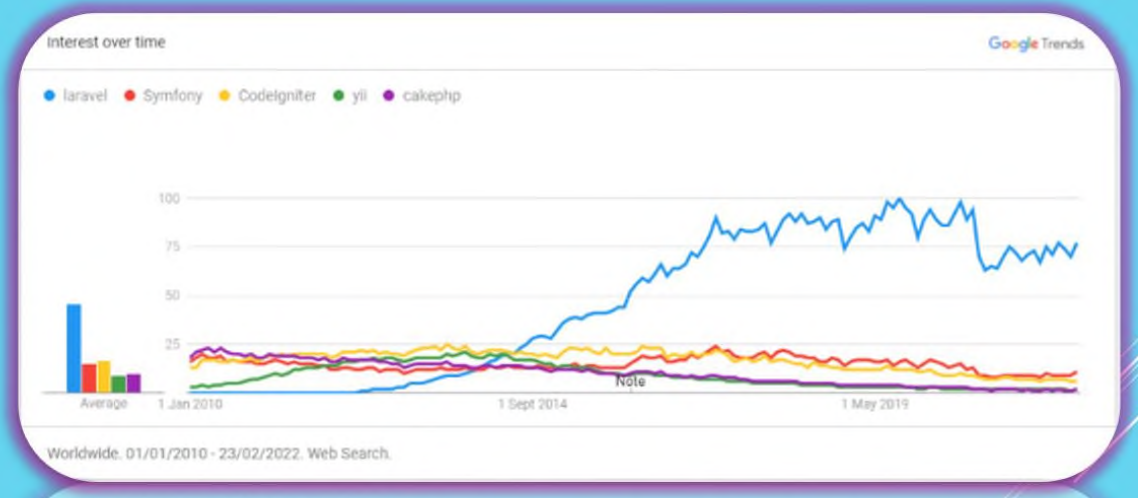


Рисунок 2.5.1 – Світова популярність фреймворків.

7

```php
<div>
    <?php foreach ($users as $user): ?>
        Hello, <?php echo $user->name; ?> <br />
    <?php endforeach; ?>
</div>
```

Рис. 2.6.1 – PHP в шаблоні HTML.

```blade
<div>
    @foreach ($users as $user)
        Hello, {{ $user->name }} <br />
    @endforeach
</div>
```

Рис. 2.6.2 – Приклад застосування Blade.

Blade та Livewire в Laravel

```php
<?php

namespace App\Http\Livewire;

use Livewire\Component;

class Counter extends Component
{
    public $count = 0;

    public function increment()
    {
        $this->count++;
    }

    public function render()
    {
        return view('livewire.counter');
    }
}
```

Рис. 2.6.3 – «Лічильник» на PHP.

```blade
<div>
    <button wire:click="increment">+</button>
    <h1>{{ $count }}</h1>
</div>
```

Рис. 2.6.4 – «Лічильник» на Laravel Livewire.

8

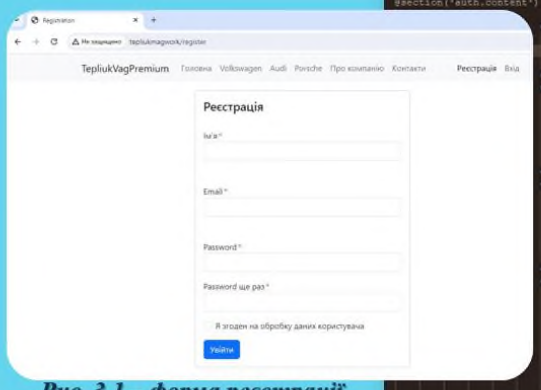


Рис. 3.1 – форма реєстрації

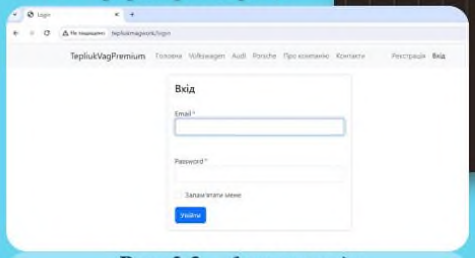


Рис. 3.2 – форма входу

```blade
@extends('layouts.auth')
@section('page.title', 'Registration')
@section('auth.content')

<div class="card-body">
    <h1 class="m-0">
    {{__('Реєстрація')}}
    </h1>
</div>
<div class="card-body">
    <form action="{{route('register.store')}}" method="POST">
        <div class="mb-3">
            <label class="required">{{__('Ім'я')}}</label>
            <input type="text" name="name" class="form-control" autofocus>
        </div>
    </form>
</div>
<div class="card-body">
    <form action="{{route('login.store')}}" method="POST">
        <div class="mb-3">
            <label class="required">{{__('Email')}}</label>
            <input type="email" name="email" class="form-control">
        </div>
    </form>
</div>
<div class="card-body">
    <form action="">
        <div class="mb-3">
            <label class="required">{{__('Password')}}</label>
            <input type="password" name="password" class="form-control">
        </div>
        <div class="mb-3">
            <label class="required">{{__('Password ще раз')}}</label>
            <input type="password" name="password_confirmation" class="form-control">
        </div>
        <div class="mb-3">
            <div class="form-check">
                <input type="checkbox" name="remember"  value="1" class="form-check-input"  id="remember">
                <label class="form-check-label" for="remember">
                    {{__('Я згоден на обробку даних користувача')}}
                </label>
            </div>
        </div>
        <button  type="submit"  class="btn btn-primary">
                {{__('Увійти')}}
        </button>
    </form>
</div>
```

Рис. 3.3 – код до реєстрації

10