

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,
Факультет комп'ютерних наук і технологій
(повне найменування інституту, назва факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавра

(ступінь вищої освіти)

на тему ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДОВГОСТРОКОВОГО
ПРОКАТУ АВТОМОБІЛІВ
LONG-TERM CAR RENTAL SOFTWARE

Виконав: студент(ка) 4 курсу, групи КНТ-117
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)
Інженерія програмного забезпечення

Полуміснко Д.О.

(прізвище та ініціали)

Керівник Льовкін В.М.

(прізвище та ініціали)

Рецензент Зеленьова І.Я.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ІРЕ, ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
“ ” 2021 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Полумієнка Дмитра Олександровича

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення довгострокового прокату автомобілів

Long-Term Car Rental Software

керівник проєкту (роботи) Льовкін Валерій Миколайович, к.т.н., доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “30” березня 2021 року № 103

2. Строк подання студентом проєкту (роботи) 17 травня 2021 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Проєктування програмного забезпечення. 3. Основні рішення щодо реалізації компонентів системи. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	Льовкін В.М., доцент		
Нормоконтролер	Камінська Ж.К., асистент		

7. Дата видачі завдання “10” березня 2021 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області	2-3 тижні	Розділ 1
3	Розробка архітектури програми	4 тиждень	Розділ 2
4	Розробка програми	5-6 тижні	Розділ 3
5	Тестування та експериментальне дослідження програми	7 тиждень	Розділ 4
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	8 тиждень	Додатки
7	Захист роботи	9 тиждень	

Студент(ка)

(підпис)

Полуміснко Д.О.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис)

Льовкін В.М.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 85 с., 31 рис., 1 табл., 3 дод., 15 джерел.

ПРОКАТ АВТОМОБІЛІВ, ДОВГОСТРОКОВИЙ ПРОКАТ, ІДЕНТИФІКАЦІЙНИЙ НОМЕР ТРАНСПОРТНОГО ЗАСОБУ, ASP.NET, MODEL-VIEW-CONTROLLER.

Об'єкт роботи – процес прокату автомобілів. Предмет роботи – програмне забезпечення довгострокового прокату автомобілів. Мета роботи – розробити програмне забезпечення довгострокового прокату автомобілів для підтримки роботи розподіленого агентства.

Матеріали, методи та технічні засоби: мова програмування C#, фреймворк ASP.NET MVC, система керування базами даних MS SQL Server.

Результати. Виконано огляд програмних аналогів, забезпечено проєктування програмного забезпечення з результатами у вигляді таблиці порівняння мов програмування, діаграм прецедентів, схеми бази даних, вибору фреймворку ASP.NET MVC, забезпечено реалізацію програми з результатами у вигляді опису рішень, які було прийнято під час реалізації програми, з узагальненням у вигляді структурної схеми, описом сценаріїв роботи користувачів з програмою.

Висновки. Розроблене програмне забезпечення призначене для виконання контролю і керування за замовленнями з прокату автомобілів зі сторони агентств з прокату та створення і моніторингу за замовленнями зі сторони клієнтів. Клієнти можуть переглядати всі наявні пропозиції, зручним чином отримувати інформацію про наявні тарифи та обирати потрібний.

Галузь використання. Агентства з прокату автомобілів, що надають послуги на період від 1 місяця з можливою розгалуженою системою пунктів прокату.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 85 pages, 31 figures, 1 table, 3 appendixes, 15 sources.

CAR RENTAL, LONG-TERM RENTAL, VEHICLE IDENTIFICATION NUMBER, ASP.NET, MODEL-VIEW-CONTROLLER.

The object of the work is the process of car rental. The subject of the work is software for long-term car rental.

The purpose of the work is to develop software for long-term car rental to support the work of a distributed agency.

Materials, methods and technical means: C# programming language, ASP.NET MVC framework, Microsoft SQL Server database management system.

Results. A review of software analogues was performed. Software was designed with results in the form of a table comparing programming languages, use case diagrams, database schema, selection of ASP.NET MVC framework. The software was implemented with results in the form of a description of decisions made during the program with its generalization in the form of the structure diagram, the description of scenarios of work of users with the program.

Conclusions. The developed software is designed to control and manage car rental orders for rental agencies and to create and monitor orders for customers. Customers can view all available offers, conveniently receive information about available rates and choose the right one.

Field of use. Car rental agencies which provide services for a period of 1 month and more with a possible distributed system of rental points.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	7
Вступ.....	8
1 Аналіз предметної області.....	9
1.1 Проблема довгострокового прокату автомобілів	9
1.2 Огляд програмних аналогів.....	12
1.3 Висновки за розділом 1	16
2 Проєктування програмного забезпечення	18
2.1 Вибір мови програмування	18
2.2 Вимоги до програмного забезпечення	19
2.3 Проєктування бази даних	23
2.4 Визначення фреймворку розробки програмного забезпечення	30
2.5 Висновки за розділом 2	32
3 Основні рішення щодо реалізації компонентів системи.....	33
3.1 Структура розробленого програмного забезпечення	33
3.2 Опис логіки реалізації.....	35
3.3 Висновки за розділом 3	40
4 Експлуатація, тестування та експериментальне дослідження програми	41
4.1 Призначення програми	41
4.2 Умови виконання програми	41
4.3 Тестування сценаріїв роботи з програмою.....	42
4.4 Висновки за розділом 4	54
Висновки	55
Перелік джерел посилання	56
Додаток А Технічне завдання	58
Додаток Б Текст програми	62
Додаток В Слайди презентації.....	78

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

CDW	– Collision Damage Waiver;
MVC	– Model-View-Controller;
PAI	– Personal Accident Insurance;
PEC	– Personal Effects Coverage;
SCDW	– Super Collision Damage Waiver;
SLI	– Supplemental Liability Insurance;
VIN	– Vehicle Identification Number.

ВСТУП

Розвиток економіки призводить до активізації необхідності оренди тих чи інших засобів для підтримки відповідної ринкової діяльності, адже не завжди підприємство може забезпечити собі власні такі засоби або не завжди їх потребує у власному господарстві, адже зокрема це пов'язано з додатковими витратами обслуговування, що може потребувати створення відповідного відділення, підтримки його діяльності. Одним з основних таких засобів є транспортні засоби. Ще однією складовою прокату автомобілів є потреба людей в забезпеченні підтримки розваг, потягу до забезпечення престижності. Це також є двигуном популярності прокату автомобілів.

Одночасно варто зазначити, що існує короткостроковий прокат автомобілів, коли автомобіль береться в оренду на строк від одного дня, каршеринг, коли автомобіль може братися в прокат на строк менше одного дня, тобто він буде взятий в одній точці міста і швидше за все повернутий в іншій, і довгостроковий, коли прокат відбувається на строк більше місяця, і в даному випадку швидше за все автомобіль береться і повертається в одному й тому самому місці. Останній варіант прокату направлений швидше на підтримку як раз бізнесу.

Тому довгостроковий прокат автомобілів є актуальною галуззю, розробка програмного забезпечення для якої є важливим практичним завданням, адже з розвитком економіки з'являються нові агентства з прокату автомобілів і їх робота без таких засобів неможлива.

Метою роботи стало розроблення програмного забезпечення довгострокового прокату автомобілів для підтримки роботи розподіленого агентства.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Проблема довгострокового прокату автомобілів

Під довгостроковим прокатом автомобілів розуміється оренда автомобіля на строк від 1 місяця.

Лізинг авто – це фінансовий інструмент, який є орендою транспортного засобу з подальшим правом викупу або поетапним придбанням автомобіля у власність [1].

Якщо людині потрібна машина на декілька місяців для роботи або тривалої поїздки в іншу країну, але при цьому важливо не замикатися на одному фіксованому контракті і плани можуть змінитися, то слід віддати перевагу саме довгостроковій оренді, при цьому компанії, які надають можливості довгострокової оренди, мають завжди обмеження максимальної тривалості, якщо автомобіль потрібний на більший строк, то потрібно звернутися до послуг лізингу (тривалість зазвичай при цьому починаючи від 2 років), загалом відносно довгострокової оренди та лізингу можна відзначити наступне:

а) довгострокова оренда характеризується тим, що витрати можуть бути більшими ніж під час лізингу (в залежності від тривалості прокату), а до переваг відносяться:

- 1) гнучкість: орендувати можна стільки часу, скільки потрібно;
- 2) керувати можна автомобілем нової моделі;
- 3) бронювання часто стає таким же простим, як звичайний прокат автомобілів;
- 4) зазвичай можна змінити машину під час оренди, якщо потрібно;
- 5) можна зателефонувати в службу обслуговування клієнтів, якщо виникли проблеми з автомобілем;
- 6) підходить для тих, хто довго потребує машину в іноземній країні;

б) до плюсів лізингу належать:

1) низькі витрати на обслуговування, більшість охоплюються договором лізингу;

2) керувати новим автомобілем можна не вносячи першого внеску;

3) лізинг може бути вигіднішим, якщо потрібна машина на термін більше 1 року;

в) мінуси лізингу:

1) потрібно робити платіж щомісяця протягом усього терміну лізингу;

2) обмежень пробігу може бути важко дотримуватися, якщо доводиться багато їздити;

3) якщо потрібний автомобіль на довгий термін в іноземній країні, лізинг, ймовірно, неможливий [2]-[3].

Основні компанії з прокату автомобілів працюють одразу в різних регіонах, однак управління логістикою в них в основному розділене відповідно до національних дочірніх компаній, такі організації керують мережею пунктів прокату (станцій), де клієнти можуть забрати і повернути автомобілі, типово національна орендна мережа має певну ієрархічну структуру: компанія з прокату автомобілів об'єднана з різними видами станцій, корпоративна станція функціонує за допомогою штатів та автомобілів, які входять до складу компанії з прокату автомобілів, станція з автономним штатом, але без окремого парку автомобілів називається корпоративним агентом, крім того, станції франчайзингових партнерів, а також іноземні станції, як правило, працюють з окремим парком, і в основному автономні щодо управління логістикою, при цьому компанія з прокату автомобілів, як правило, управляє приблизно 15 групами автомобілів, де кожна група містить різні машини зі схожою якістю (наприклад, щодо розміру та обладнання), кожна група представляє однорідний товар з базовою платою за оренду (ставкою) [4].

Ідентифікаційний номер транспортного засобу або vehicle identification number (VIN) – це унікальний код, який присвоюється кожному моторному транспортному засобу під час його виготовлення, це 17-символьний рядок літер і цифр без пробілів або літер Q (q), I (i) та O (o), вони пропущені, щоб уникнути плутанини з цифрами 0 та 1, кожен розділ VIN містить конкретну інформацію про транспортний засіб, включаючи рік, країну та фабрику виробництва, марку і модель та серійний номер, VIN, як правило, друкуються в одному рядку [5].

До основних типів страхування автомобіля належать:

– collision damage waiver (CDW) або відмова від збитків від зіткнення – це відмова від збитків, запропонована компанією, що займається прокатом автомобілів, яка означає, що якщо орендована машина пошкоджена або викрадена, не доведеться платити за її ремонт або заміну, однак ці відмови включають франшизу (хоча ваш звичайний поліс страхування автомобілів може включати покриття зіткнень для орендованих автомобілів, він, швидше за все, включає франшизу і може не оплатити всі витрати компанії, що орендує автомобіль, такі як втрата використання – плата за гроші, які компанія втрачає за час перебування машини неактивною);

– super collision damage waiver (SCDW) – це варіант страхування, який покриває всі можливі збитки без виключень, які можуть виявлятися за попереднього варіанту;

– supplemental liability insurance (SLI) – додаткове страхування відповідальності: страхування відповідальності, як правило, покриває збитки майну інших людей (наприклад, завдані їх автомобілям), а також медичні витрати за травми інших людей внаслідок нещасного випадку, за який водій несе юридичну відповідальність, при цьому покриття відповідальності може бути занадто низьким, для таких випадків і передбачено даний варіант страхування;

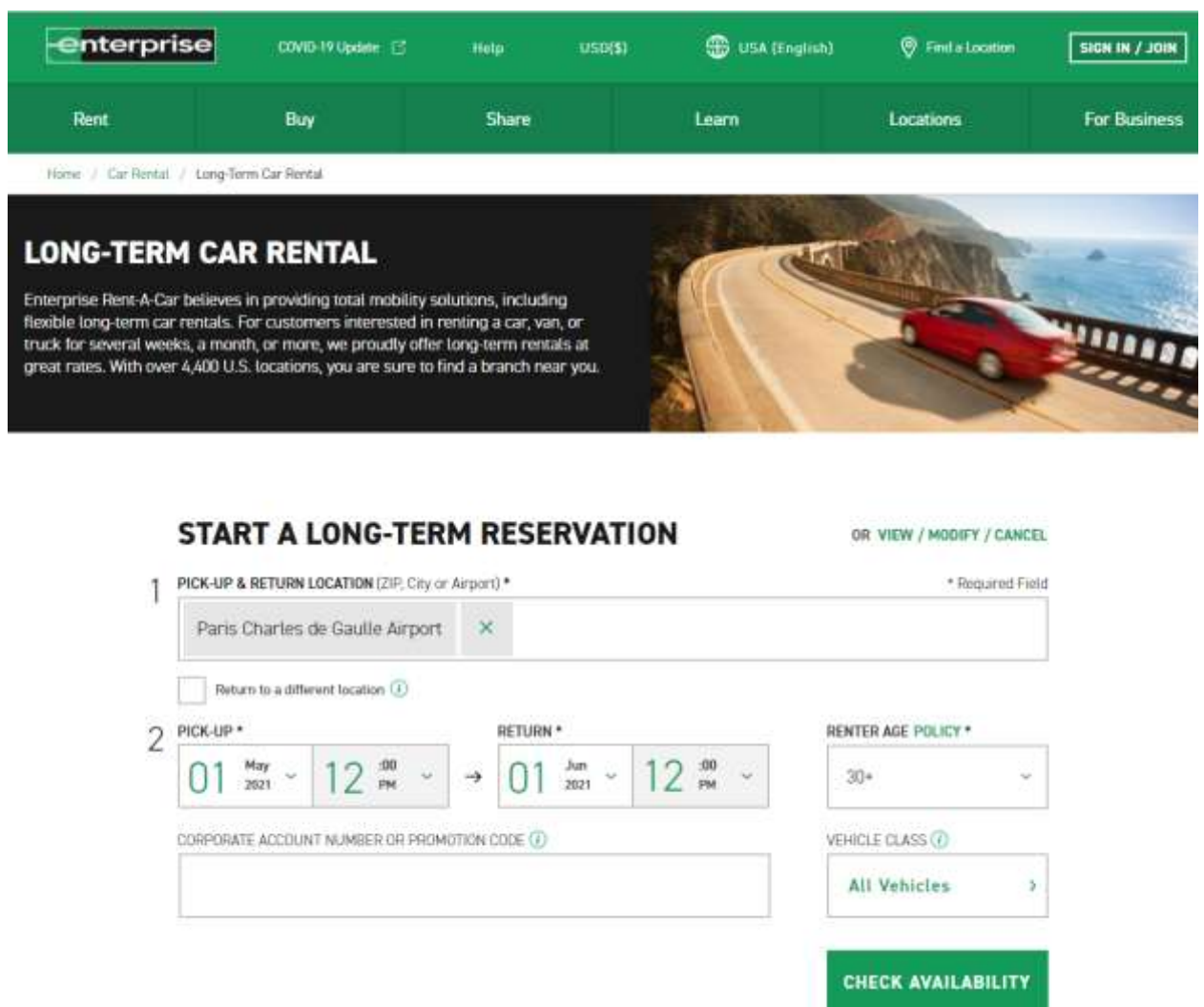
– personal accident insurance (PAI): хоча покриття відповідальності покриває пошкодження майна та травми інших людей внаслідок дорожньо-

транспортної пригоди, дане особисте покриття нещасних випадків включає медичну допомогу, допомогу швидкої допомоги та допомогу на випадок летального випадку для водія орендованого автомобіля та пасажирів;

– personal effects coverage (PEC) – покриття особистих речей охоплює викрадення майна з орендованого автомобіля до встановленого ліміту в доларах [6]-[7].

1.2 Огляд програмних аналогів

Одним з найвідоміших аналогів у даній сфері є американська компанія Enterprise Rent-A-Car [8] (рис. 1.1).



The screenshot displays the Enterprise Rent-A-Car website interface for starting a long-term reservation. The top navigation bar is green with the Enterprise logo, a COVID-19 update link, a Help link, a USD(\$), a USA (English) link, a Find a Location link, and a SIGN IN / JOIN button. Below this is a secondary navigation bar with links for Rent, Buy, Share, Learn, Locations, and For Business. The main content area features a 'LONG-TERM CAR RENTAL' section with a description of the service and a background image of a red car on a coastal road. Below this is a 'START A LONG-TERM RESERVATION' form with the following fields:

- 1 PICK-UP & RETURN LOCATION (ZIP, City or Airport) ***: A text input field containing 'Paris Charles de Gaulle Airport' and a close button (X). A checkbox for 'Return to a different location' is also present.
- 2 PICK-UP ***: A date and time selector showing '01 May 2021' at '12:00 PM'.
- RETURN ***: A date and time selector showing '01 Jun 2021' at '12:00 PM'.
- RENTER AGE POLICY ***: A dropdown menu showing '30+'.
- CORPORATE ACCOUNT NUMBER OR PROMOTION CODE**: A text input field.
- VEHICLE CLASS**: A dropdown menu showing 'All Vehicles'.
- CHECK AVAILABILITY**: A green button at the bottom right of the form.

Рисунок 1.1 – Інтерфейс сервісу Enterprise Rent-A-Car

Даний сервіс надає широкі можливості щодо прокату автомобілів, які включають оренду автомобілів, покупку автомобілів, лізинг, каршеринг, тобто усі розглянуті вище варіанти оренди автомобілів у даному випадку надаються.

Під час реєстрації облікового запису користувач одразу має внести не тільки загальні дані про себе як у будь-якому сервісі, але також і розширені контактні дані, і дані водійського посвідчення.

Детальніше було проаналізовано засоби Enterprise Rent-A-Car, якими надаються можливості довгострокової оренди (рис. 1.2). Під час організації пошуку потрібно задати пункт, де буде забрано і повернено автомобіль, дати прокату. У випадку короткострокового прокату автомобіля зазвичай задаються одразу окремо пункт відправлення і пункт прибуття. Окрім того можна задати промокод на послуги.

enterprise TERMS & CONDITIONS / POLICIES | total: \$1,342.⁵¹

RENTAL DETAILS → PICK-UP & RETURN → VEHICLE Select → EXTRAS

Sat, May 01, 12:00 PM
Tue, Jun 01, 12:00 PM

Paris Charles de Gaulle Airport

CHOOSE A VEHICLE CLASS 37 RESULTS SORT BY Featured

Filters

TRANSMISSION Total from:

- ☐ Automatic \$1813
- ☐ Manual \$970

MILEAGE Total from:

- ☐ Limited Mileage \$970

VEHICLE TYPE Total from:

- ☐ Cars \$1362
- ☐ SUVs \$1833
- ☐ Passenger Vans \$2189
- ☐ Vans \$970

NUMBER OF PASSENGERS Total from:

- ☐ 2+ \$970

CURRENT SELECTION

Mini
Renault Twingo or similar
Manual 4 People 1 Bags
FEATURES & PRICE DETAILS

PAY LATER
\$303.¹⁵ Per Week | \$1,342.⁵¹ total
SELECT

Economy
Fiat 500 or similar
Manual 4 People 2 Bags
FEATURES & PRICE DETAILS

PAY LATER
\$318.³¹ Per Week | \$1,409.⁶⁰ total
SELECT

Economy 5 door
Opel Corsa or similar
Manual 5 People 2 Bags
FEATURES & PRICE DETAILS

PAY LATER
\$348.⁴² Per Week | \$1,543.⁸⁹ total
SELECT

Compact
Peugeot 308 or similar
Manual 5 People 3 Bags
FEATURES & PRICE DETAILS

PAY LATER
\$394.¹⁰ Per Week | \$1,745.²⁹ total
SELECT

Feedback

Рисунок 1.2 – Інтерфейс замовлення в сервісі Enterprise Rent-A-Car

За результатами пошуку потрібно обрати один з автомобілів, що підпадають під задані параметри, а інформація про вартість прокату автомобілів розпочинається від 1 тижня (рис. 1.2). Для наданого списку автомобілів можна встановити різноманітні фільтри.

Після обрання автомобіля потрібно обрати варіант страхування, додаткове обладнання, що необхідне для прокату, а далі внести контактні дані, якщо їх не було внесено раніше (тобто робота відбувається без облікового запису) та погодитися з умовами договору.

У сервісі Sixt обособлення довгострокового прокату не відбувається [9] (рис. 1.3).

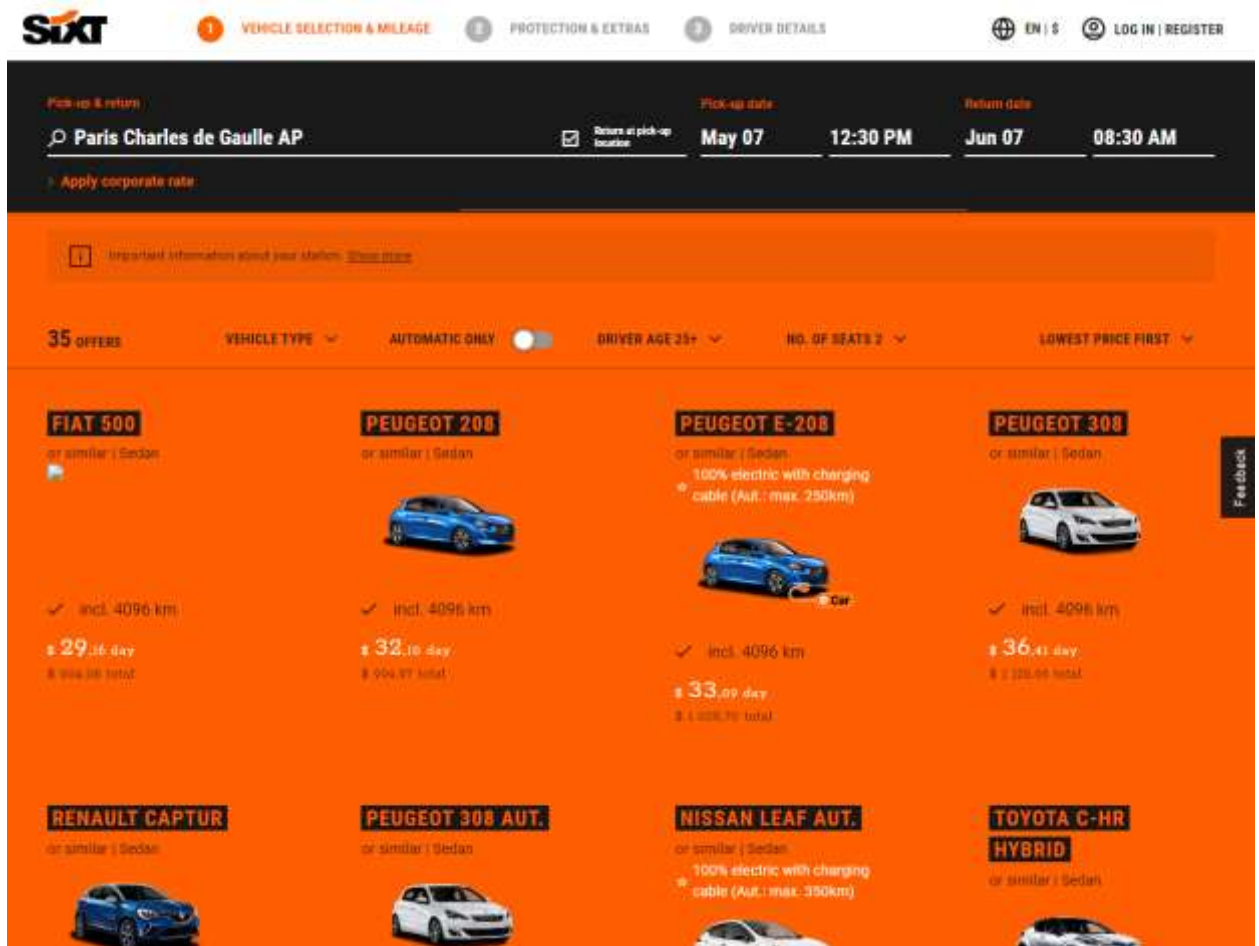


Рисунок 1.3 – Інтерфейс сервісу Sixt

Фактично під час пошуку можна задати період прокату, що не перевищує 365 днів. При цьому розрахунок вартості прокату буде все одно за

стандартними подобовими тарифами, що дозволяє, але фактично спотворює ідею довгострокового прокату автомобіля. Принцип організації замовлення подібний до попередньої альтернативи, але при цьому за результатами пошуку видається завжди подорова вартість прокату, на відміну від попереднього варіанту.

І Enterprise Rent-A-Car, і Sixt є міжнародними сервісами прокату автомобілів [10] (рис. 1.4).

Сервіс AVISLeasing направлений швидше як раз на довгострокову оренду автомобілів, засобів короткострокової оренди автомобілів сервіс не має. При цьому замовлення виконуються в закритому режимі. Тобто дізнатися фактичну або приблизну вартість прокату не можна попередньо.

ДОВГОСТРОКОВА ОРЕНДА МАЛЮГАБАРИТНИХ АВТО

ЗАЛИШТЕ СВОЇ КОНТАКТНІ ДАНІ І НАШ КОНСУЛЬТАНТ ЗВ'ЯЖЕТЬСЯ З ВАМИ

ІМ'Я

Е-МАЙЛ

НОМЕР ТЕЛЕФОНУ

НАЗВА ВАШОЇ КОМПАНІЇ

ВАС ЦІКАВИТЬ

КОМЕНТАР

Я ознайомлений(на) з Політикою конфіденційності та даю згоду на обробку моїх персональних даних.

НАДІСЛАТИ

Mini - Група А

Ваш вибір: Fiat 500 або подібний

Це компактний Fiat 500, ідеальний для пересування містом та паркуванням. До того ж цей автомобіль економічний на дорозі.

Вартість оренди: 1

ОБРАТИ

Economy - Група В

Ваш вибір: VW Polo, Ford Fiesta або Skoda Rapid

Ця група автомобілів, ви отримаєте змогу переконатися в практичності та задоволенні, що це за автомобіль.

Вартість оренди: 21

Вартість депозиту: 1

ОБРАТИ

Comfort - група С

Рисунок 1.4 – Інтерфейс вебсайту AVISLeasing

Вебсайт надає функціональність оформлення заявки на зв'язок з менеджером. Окрім безпосередньо лізингу надається також можливість покупки автомобіля.

Виконаний аналіз продемонстрував, що проблема розроблення програмного забезпечення довгострокового прокату автомобілів є актуальною. Існуючі сервіси в Україні або не направлені саме на виділення довгострокового прокату, а тому пропонують для нього добові тарифи, або виконують це в закритому режимі. Створення такого сервісу дозволить привернути увагу насамперед за допомогою відкритості.

Довгостроковий прокат дозволяє більше уваги приділяти, наприклад, сервісному обслуговуванню. Тоді клієнт, який бере автомобіль в оренду, може бути повністю впевненим у його стані. При короткостроковій оренді між роботою з клієнтами виконується тільки контроль загальних проблем, які швидше пов'язані з загальним станом автомобіля і виявленням великих дефектів. При довгостроковій оренді аналіз технічного стану можна виконати якісніше.

1.3 Висновки за розділом 1

Виконано аналіз проблеми довгострокового прокату автомобілів з виділенням понять прокату і лізингу, визначенням їх особливостей, визначенням супутніх понять до основних сутностей предметної області, зокрема включаючи VIN номер автомобіля, способи страхування прокату автомобілів. Виконано аналіз програмних аналогів, що забезпечують послуги прокату в Україні та світі. Виділено нестачу програмних засобів для відкритого довгострокового прокату автомобілів, на що і вирішено націлити виконувану розробку.

Для розв'язання у роботі і досягнення мети було виділено такі завдання:

- визначення вимог до програмного забезпечення довгострокового прокату автомобілів;
- вибір мови і засобів проєктування і розробки;
- проєктування програмного забезпечення;
- реалізація і налагодження програмного забезпечення.

2 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Вибір мови програмування

Програмне забезпечення має розроблятися у вигляді вебзастосунку довгострокового прокату автомобілів. Таке рішення дозволяє з одного боку створити бекенд, необхідний для роботи всієї майбутньої системи з охопленням мобільних застосунків зокрема, а з іншого боку – отримати повноцінне рішення, що можна буде використовувати на різних пристроях.

Грунтуючись на цьому, альтернативними рішеннями для реалізації програми було обрано мови програмування C# [11]-[12] та PHP [13] як найбільш популярне рішення для веброзробки на даний момент.

Основними критеріями порівняння було визначено:

- тип виконання програм – це важливий критерій, оскільки від цього залежить і швидкість виконання програми;
- велика кількість вбудованих функцій: наявність додаткових функцій дозволяє писати менше коду власноруч, більше використовувати готових рішень;
- масштаб розроблюваних рішень: сервіс довгострокового прокату автомобілів є досить масштабним рішенням, яке з часом має тільки розростатися;
- фреймворки веброзробки: сучасні вебзастосунки розробляються на основі вебфреймворків, від наявності ефективних засобів залежить якість створених рішень;
- призначення.

Результати порівняння мов програмування для веброзробки було зведено до табл. 2.1.

За результатами порівняння обрано мову програмування C# для створення програмного забезпечення, що розробляється, оскільки вона має призводити до створення коду, що швидше виконується, та дозволяє створювати рішення більшого масштабу.

Таблиця 2.1 – Вибір мови програмування для веброзробки

Критерій	PHP	C#
Тип виконання програм	Інтерпретатор	Компілятор
Велика кількість вбудованих функцій	+—	++
Масштаб розроблюваних рішень	Середні і малі	Середні і великі
Фреймворки веброзробки	+	+
Призначення	Веброзробка	Загальне

2.2 Вимоги до програмного забезпечення

Програмне забезпечення, що розробляється, повинно бути направленим на роботу окремих агентств, а також за необхідності – на підтримку роботи декількох агентств. У першому випадку в якості агентств по суті розглядаються пункти прокату.

Програма має підтримувати саме довгостроковий прокат, тобто взяття в прокат і повернення автомобіля протягом дня або навіть у строк менший за місяць не передбачається. Відповідно укладання договору прокату реалізується через представника агентства і не передбачається самостійно. Тобто клієнт може оформити замовлення на прокат, але його потім має підтвердити працівник агентства, тільки після взаємодії з працівником клієнт може отримати автомобіль. Окрім оформлення замовлення клієнт може також переглядати різного роду інформацію про власні замовлення. Також програма повинна дозволяти клієнту вносити дані про себе. Тобто за рахунок оформлення замовлення клієнтом і внесення даних про себе етап видачі автомобіля в прокат має скорочуватися. З іншого боку дана програма

агентствам дозволяє фіксувати прокати в електронному вигляді, керувати ними всіма одночасно, відповідно автоматизувати і прискорювати їх роботу.

Для забезпечення роботи користувачів з програмою в сфері організації довгострокового прокату автомобілів мають надаватися наступні функції:

- авторизація клієнтів і працівників агентства;
- реєстрація клієнтів;
- внесення даних автомобілів для прокату;
- керування автомобілями в прокаті;
- визначення вартості прокату автомобіля;
- керування прокатами автомобілів;
- створення замовлення на прокат автомобіля клієнтом або працівником агентства;
- підтвердження прокату автомобіля працівником агентства;
- повернення автомобіля;
- пошук автомобіля для прокату;
- визначення автомобілів у прокаті;
- перегляд і керування профілем користувача;
- перегляд історії прокату автомобілів клієнтом;
- пошук історії прокату автомобілів;
- перегляд власних замовлень прокату автомобілів.

Клієнт може взаємодіяти з програмою (рис. 2.1) шляхом виконання наступних функцій:

- авторизація;
- реєстрація;
- створення замовлення на прокат автомобіля;
- скасування замовлення на прокат автомобіля;
- перегляд власних замовлень прокату автомобілів;
- пошук автомобіля для прокату;

- перегляд і керування профілем;
- перегляд історії власних прокатів автомобілів.

Альтернативними сценаріями є сценарій, що включає скасування замовлення на прокат автомобіля, сценарій, що включає реєстрацію, та сценарій, що включає керування профілем користувача.



Рисунок 2.1 – Діаграма прецедентів роботи клієнта з програмою

Працівник агентства може взаємодіяти з програмою (рис. 2.2) шляхом виконання наступних функцій:

- авторизація;
- внесення даних автомобілів для прокату;
- керування автомобілями в прокаті;
- визначення вартості прокату автомобіля;
- керування прокатами автомобілів;
- створення замовлення на прокат автомобіля;
- підтвердження прокату автомобіля;

- повернення автомобіля;
- пошук автомобіля для прокату;
- визначення автомобілів у прокаті;
- перегляд і керування профілем користувача;
- перегляд історії прокату автомобілів клієнтом;
- перегляд історії прокату заданого автомобіля;
- пошук історії прокату автомобілів.



Рисунок 2.2 – Діаграма прецедентів роботи працівника агентства з програмою

Альтернативними сценаріями є сценарій, що включає керування профілем користувача, сценарії, що включають перегляд історії прокату

автомобілів за клієнтом і за заданим автомобілем, адже вони надають додаткові засоби, а стандартним є пошук історії прокату автомобілів, де визначаються дата, номер замовлення.

2.3 Проєктування бази даних

Системою керування базами даних було обрано Microsoft SQL Server через її здатність забезпечувати якісну взаємодію з застосунками корпоративного рівня, що максимально важливо для програмних систем такого типу.

За результатами аналізу предметної області було виділено основні сутності.

Сутність AutoMaker визначає виробника автомобіля.

Сутність AutoModel визначає конкретну модель автомобіля, з якою пов'язана основна інформація, що визначає параметри послуги прокату.

Сутність Auto визначає конкретний автомобіль, прокат якого відбувається. З цією сутністю пов'язаний сам факт надання конкретної послуги прокату.

Сутність Rental визначає прокат автомобіля.

Сутність RentalPrice визначає тариф, за яким визначає вартість прокату.

Сутність Insurance визначає параметри послуги страхування.

Сутність Promo визначає код на знижку.

Сутність Agency визначає агентство або пункт прокату (за підтримки роботи одного агентства)

Сутність Leaser визначає орендаря.

Сутність Worker визначає працівника сервісу прокату.

Сутність Service визначає додаткові послуги, які пропонуються прокатником під час виконання прокату.

Таблиця Auto містить поля:

- vin – VIN-номер;
- id_automodel – модель автомобіля;
- lastchekup – момент виконання останнього технічного огляду автомобіля;
- made – рік виробництва автомобіля;
- information – будь-яка додаткова інформація про автомобіль;
- state – стан автомобіля;
- rentals – кількість днів оренди автомобіля;
- id_agency – агенція, яка надає автомобіль в оренду;
- number – реєстраційний номер автомобіля (максимальна довжина була визначена в 10 символів, це довжина номерів у Індії);
- color – колір автомобіля.

Поле id_automodel є зовнішнім ключем до таблиці AutoModel.

Поле id_agency є зовнішнім ключем до таблиці Agency, тобто кожен автомобіль прив'язаний до конкретного пункту прокату, там береться автомобіль у прокат і туди повертається.

Таблиця AutoMaker містить поля:

- id – порядковий номер виробника;
- makename – назва виробника;
- country – країна виробника.

Таблиця AutoModel містить поля:

- id – порядковий номер моделі;
- modelname – назва моделі;
- id_automaker – виробник автомобіля;
- gearbox – тип коробки передач автомобіля;
- photo – фотографія автомобіля;
- engine – обсяг двигуна автомобіля;
- fuel – тип пального;
- fuelrate – витрата пального на 100 кілометрів (для електродвигуна кількість хвилин підзарядки на 100 кілометрів);

– cityfuelrate – витрата пального на 100 кілометрів за міського використання;

– passengernumber – максимальна кількість пасажирів;

– doornumber – кількість дверей;

– autotype – тип автомобіля;

– luggage – обсяг багажника;

– conditioner – тип система кондиціонування (або її відсутність);

– autoclass – клас автомобіля (економний, бізнес, представницький або преміум);

– inscoeff – коефіцієнт страхування даної моделі автомобіля, що може використовуватися для врахування престижності автомобіля у вартості страхування.

Поле id_automaker є зовнішнім ключем до таблиці AutoMaker.

Таблиця Rental містить поля:

– id – порядковий номер прокату автомобіля;

– id_lease – користувач, який взяв автомобіль у прокат;

– id_worker – робітник, який оформив або підтвердив прокат;

– id_automodel – модель автомобіля, прокат якої виконується;

– created – момент початкового оформлення;

– date1 – момент початку прокату;

– date2 – момент завершення прокату;

– price – фактична вартість прокату;

– paid – поточна сплачена сума за прокат (або фінальна);

– state – стан прокату;

– id_promo – використаний промокод (може бути порожнім);

– id_rentalprice – тариф, використаний під час обчислення вартості прокату;

– id_insurance – замовлене страхування (може бути порожнім).

Поле id_lease є зовнішнім ключем до таблиці Leaser.

Поле id_worker є зовнішнім ключем до таблиці Worker.

Поле `id_auto` є зовнішнім ключем до таблиці `Auto`.

Поле `id_promo` є зовнішнім ключем до таблиці `Promo`.

Поле `id_rentalprice` є зовнішнім ключем до таблиці `RentalPrice`.

Поле `id_insurance` є зовнішнім ключем до таблиці `Insurance`.

Таблиця `AutoRental` визначає конкретний автомобіль, який закріплений за прокатом, і містить поля:

- `id` – порядковий номер;
- `problems` – виявлені проблеми;
- `information` – будь-який додатковий опис;
- `id_auto` – автомобіль;
- `id_rental` – прокат.

Поле `id_auto` є зовнішнім ключем до таблиці `Auto`.

Поле `id_rental` є зовнішнім ключем до таблиці `Auto`.

Таблиця `RentalPrice` містить поля:

- `id` – порядковий номер тарифу прокату;
- `maxduration` – максимальна тривалість прокату, за якої може бути використано тариф;

- `id_automodel` – модель автомобіля;
- `price` – вартість прокату;
- `period` – кількість днів, за яку встановлено відповідну вартість (одиниця вимірювання вартості);
- `deposit` – необхідний депозит для прокату.

Поле `id_automodel` є зовнішнім ключем до таблиці `AutoModel`, тобто вартість прокату ґрунтується саме на моделі автомобіля.

Таблиця `Insurance` містить поля:

- `id` – порядковий номер страхування;
- `name` – назва страхування;
- `information` – інформація про страхування;
- `period` – період часу, на який розповсюджується страхування;
- `price` – вартість страхування.

Таблиця Promo містить поля:

- id – порядковий номер промокоду;
- code – промокод (конкретна комбінація, яку має ввести користувач);
- pricescut – абсолютне значення зниження вартості прокату;
- percentcut – процентне зниження вартості прокату;
- date1 – момент, з якого діє промокод;
- date2 – момент завершення дії промокоду.

Таблиця Agency містить поля:

- id – порядковий номер агентства;
- name – назва агентства (або пункту прокату);
- city – місто;
- address – повна адреса;
- information – інформація про агентство;
- phone – контактний номер агентства.

Таблиця Leaser містить поля:

- id – порядковий номер орендаря;
- login – логін;
- password – пароль;
- surname – прізвище орендаря;
- name – ім'я орендаря;
- phone – контактний номер орендаря;
- information – інформація про орендаря;
- birth – дата народження орендаря;
- country – громадянство орендаря;
- passport – паспорт орендаря;
- code – ідентифікаційний код орендаря;
- driver – номер водійського посвідчення орендаря;
- driverend – строк дії водійського посвідчення орендаря;
- balance – баланс рахунку орендаря;
- costs – сумарні витрати на оренду даного користувача.

Таблиця Worker містить поля:

- id – порядковий номер працівника агентства;
- surname – прізвище;
- name – ім'я;
- login – логін;
- password – пароль;
- id_agency – агентство (пункт прокату автомобілів), в якому працює

працівник;

- phone – контактний номер робітника;
- position – посада робітника.

Поле id_agency є зовнішнім ключем до таблиці Agency.

Таблиця Service містить поля:

- id – порядковий номер додаткової послуги;
- name – назва послуги;
- information – інформація про послугу;
- price – вартість послуги.

До додаткових послуг можуть відноситися зокрема дитячі крісла, надання обладнання з відстеження геолокації автомобіля, захисного обладнання автомобіля, доставки автомобіля тощо.

Оскільки будь-яка додаткова послуга може входити до складу різних прокатів, а кожен прокат може охоплювати одразу декілька послуг, то для реалізації зв'язку між таблицями Service і Rental було створено таблицю RentalService.

Таблиця RentalService містить поля:

- id – порядковий номер;
- id_service – додаткова послуга;
- id_rental – прокат, до якого належить послуга;
- quantity – кількість послуг (або додаткових речей);
- price – вартість.

На рис. 2.3 представлена розроблена результуюча повна схема бази даних.

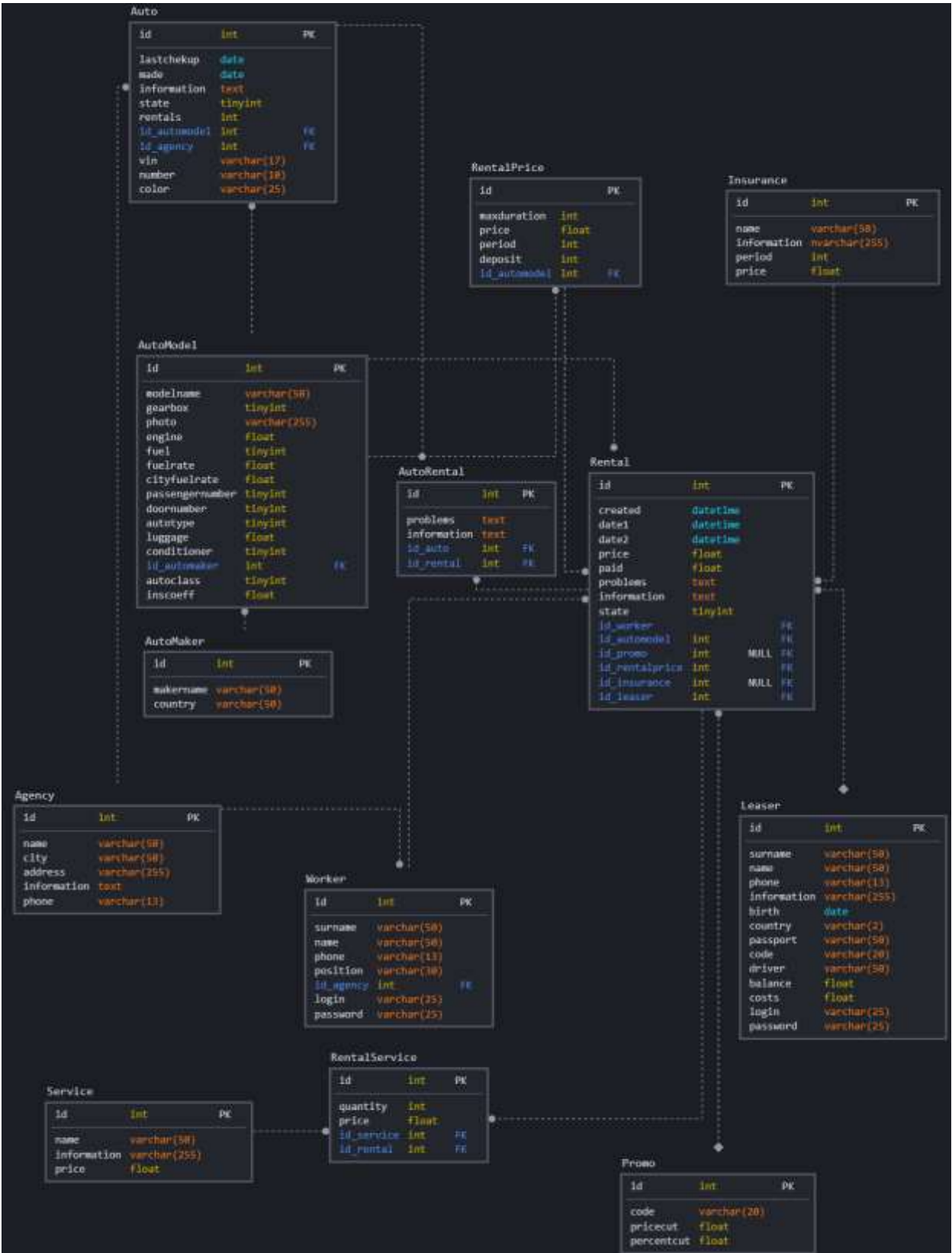


Рисунок 2.3 – Схема бази даних

2.4 Визначення фреймворку розробки програмного забезпечення

Для розробки вебзастосунку було вирішено використовувати вебфреймворк ASP.NET, що обумовлено наявністю у нього наступних позитивних характеристик:

- висока продуктивність: продуктивність є важливою особливістю будь-якого програмного забезпечення, з появою ядра ASP.NET та вебсервера Kestrel, ASP.NET є однією з найкращих платформ вебзастосунків з точки зору швидкості;

- міжплатформна та контейнерна підтримка: з моменту введення базової платформи .NET програми ASP.NET можна створювати та розгортати на таких платформах, як Windows, Linux та macOS, за допомогою Microsoft та спільноти Linux ASP.NET став першокласним кандидатом для безперебійної роботи служб розробки, окрім того ASP.NET Solutions надає розробникам можливість використовувати всі нові технології;

- уніфікований фреймворк Model-View-Controller (MVC): MVC був розроблений спеціально для створення вебзастосунків, які обслуговують HTML, основні рішення ASP.NET також розширюють структуру MVC, так що інкапсуляція аспектів контролера та моделі сторінки разом із двостороннім прив'язуванням є можливою;

- ASP.NET має ексклюзивну підтримку WebSockets, що може бути використано при запуску тривалих з'єднань та безперебійному спілкуванні вперед і назад із браузером, SignalR – це повний фреймворк, який також доступний для спрощення обробки загальних сценаріїв, SignalR може бути активно використаний під час перегляду поточних даних моніторингу одного з серверів (кожен раз, коли надходять нові дані, вони негайно передаються в браузер, щоб можна було побачити їх у реальному часі);

- прив'язка моделі – ізоляція вебформи від моделі: можливість прив'язки моделі до ASP.NET дозволяє розробляти вебформи, які повністю не залежать від моделі, яка заповнює подання;

– глобалізація та локалізація: ASP.NET робить локалізацію дат та тексту у вебзастосунку дуже простими, якщо потрібно застосовувати розробку в різних країнах світу, то локалізація є зручним рішенням, за допомогою ASP.NET налаштування програми для декількох мов можна виконати за допомогою файлів, які вважаються центральним сховищем, де зберігаються всі тексти, і вебсторінки можуть їх прочитати та заповнити мітку [14].

Фреймворк ASP.NET побудований на архітектурному шаблоні MVC, що складається з моделей, контролерів і подань, послідовність взаємодії яких представлено на рис. 2.4 у вигляді діаграми кооперації.



Рисунок 2.4 – Архітектурний шаблон MVC

2.5 Висновки за розділом 2

Обрано мову програмування C# для реалізації програми, якій було віддано перевагу порівняно з мовою PHP.

Визначено функціональні вимоги до програми з побудовою діаграм прецедентів для клієнтів та працівників агентства.

Виконано проєктування бази даних, за результатами чого описано структуру створених таблиць, відношення між ними та схему бази даних.

Прийнято рішення про вибір фреймворку розробки вебзастосунків ASP.NET для реалізації програмного забезпечення, в основі якого використовується архітектурний шаблон MVC.

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

3.1 Структура розробленого програмного забезпечення

Основна структура розробленої програми (рис. 3.1) включає такі одиниці, виражені фізично у вигляді відповідних папок і файлів:

- папка `Controllers` містить класи контролерів застосунку;
- папка `Data` містить клас `RentalDBContext`, який містить засоби роботи з множиною об'єктів створених моделей, зокрема автомобілів `Autos`, моделей автомобілів `AutoModels`, орендарів `Leasers`, працівників `Workers` тощо;
- папка `Models` містить класи моделей застосунку;
- папка `Views` містить класи подань, згрупованих за папками у відповідності з контролерами, які ними використовуються;
- папка `wwwroot` містить статичні файли, що включають зображення, каскадні таблиці стилів, javascript-скрипти;
- файл `appsettings.json` визначає загальні налаштування застосунку, що включають зокрема рядок з'єднання з базою даних;
- файл `Program.cs` визначає основну точку входу в вебзастосунок;
- файл `Startup.cs` визначає конфігурацію роботи хоста.

Контролери розділені на наступні:

- `AutoController` – контролер для роботи з автомобілями;
- `AutoModelController` – контролер для роботи з моделями автомобілів;
- `InsuranceController` – контролер для роботи зі страхуваннями;
- `LeaserController` – контролер для роботи з обліковими записами орендарів;
- `PromoController` – контролер для роботи з промокодами;
- `RentalController` – контролер для роботи з прокатами автомобілів;
- `ServiceController` – контролер для роботи з додатковими послугами.

Відповідним чином згруповані подання за папками, а яких міститься набір подань, що відповідають кожному контролеру.

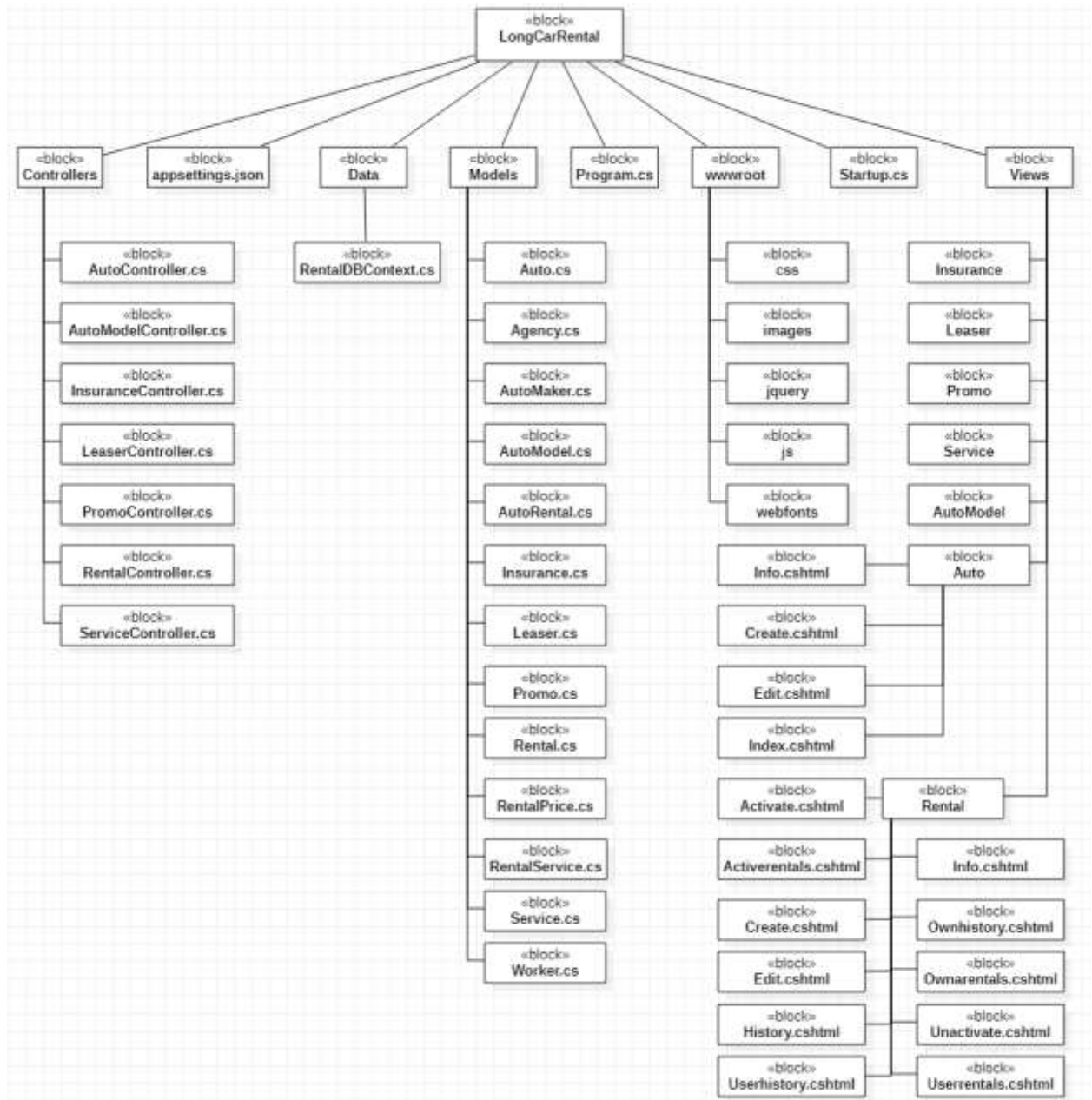


Рисунок 3.1 – Структурна схема програмного проєкту

На структурну схему винесено основний набір подань, оскільки він достатньо великий, при тому основний набір подань для кожної папки, що включають подання створення, редагування, індексування, повторюються.

3.2 Опис логіки реалізації

Для організації роботи з основними сутностями предметної області було створено моделі, що відповідають таблицям бази даних, що було розглянуто в підрозділі 2.3.

Моделі створено у відповідності з розглянутою структурою таблиць, але з додаванням зручних засобів роботи з ними. Наприклад, під час реалізації моделі Rental у її складі визначено поле, що відповідає моделі AutoRental, це дозволяє потім в контролерах опрацьовувати це загалом як єдину сутність, у верхній частині якої зберігається інформація про орендовану модель автомобіля, а у внутрішньому AutoRental – інформація про конкретний автомобіль, який було орендовано. До моделей, що пов'язані зв'язком один-до-багатьох, з одного боку включено зовнішній ключ, як було показано на схемі бази даних, а з іншого набір моделей, тобто якщо RentalPrice містить в якості зовнішнього ключа поле Id_automodel, то модель AutoModel містить поле RentalPrices, що представляє собою список об'єктів відповідної моделі. Таким чином можна за моделлю автомобіля одразу дізнатися список тарифів на її прокат, а в результаті в контролері не переформовувати дані, а одразу їх передати в подання, де кожне значення на відповідною моделлю буде виведено в таблиці на сторінці.

Для реалізації роботи з даними використано фреймворк Entity Framework, що дозволило створювати в коді програми класи моделей і оперувати з даними у об'єктно-орієнтованій формі без необхідності реалізації конкретних запитів до бази даних, які генеруються автоматично на основі звернень до об'єктів у коді самим фреймворком. Це спростило виконану роботу і дозволило уникнути необхідності подвійної роботи з даними, коли потрібно реалізовувати і роботу з таблицями бази даних, і з моделями в об'єктній формі, і з'єднувати їх між собою.

Клас `RentalDbContext` створює контекст роботи з даними, адже він містить список всіх об'єктів, що відповідають кожній моделі предметної області, саме через них виконується вибірка даних в контролері.

Звичайно, що головною сутністю предметної області є модель замовлення прокату автомобілю. На прикладі контролерів та подань для неї розглянемо принципи організації їх та взаємодії між ними.

Будь-яке замовлення на прокат може знаходитися в одному з 4 станів:

- створене – створене орендарем і не підтверджене працівником агентства;
- виконуване – підтверджене працівником агентства, тобто автомобіль вже виданий орендарю у прокат;
- скасоване – скасоване орендарем або працівником агентства, неактивне замовлення;
- виконане – завершене замовлення, повернення автомобіля підтверджено працівником агентства.

Для реалізації роботи з прокатами автомобілів було створено наступні методи контролера `RentalController`.

Метод `Task<IActionResult> Ownhistory()` виконує оброблення запиту на отримання власних виконаних замовлень на прокат автомобіля.

Метод `Task<IActionResult> Info(long? num)` виконує оброблення запиту на виведення повної інформації про замовлення з номером `num`. Для цього спочатку реалізується підвантаження повних даних (даних відповідних таблиць) за зовнішніми ключами `Id_lease`, `Id_worker`, `Id_automodel`, `Id_promo`, `Id_rentalprice`, `Id_insurance` з вибором об'єкту з заданим номером, а далі ці дані передаються у відповідне подання:

```
var rental = await privateDC.Rentals.Include(a => a.Leaser).Include(a =>
a.Worker).Include(a => a.Automodel).Include(a => a.Promo).Include(a =>
a.Rentalprice).Include(a => a.Insurance).Include(a =>
a.AutoRental.Auto).FirstOrDefaultAsync(b => b.Id == num);
```

```

if (rental == null)
{
    return NotFound();
}
return View(rental);

```

Метод `Task<IActionResult> Activate(long? num, string regnumber)` виконує оброблення запиту на підтвердження замовлення на прокат автомобіля з номером `num` та обраним реєстраційним номером автомобіля `regnumber`, що виконує працівник агентства, задля чого витягаються відповідні дані, вноситься зміна у стан замовлення та реалізується додавання даних про автомобіль з заданим реєстраційним номером.

Метод `Task<IActionResult> Unactivate(long? num)` виконує оброблення запиту на скасування замовлення на прокат автомобіля з номером `num`.

Метод `Task<IActionResult> Userhistory(long? num)` виконує оброблення запиту на отримання даних про замовлення на прокат, які вже були повністю виконані і які належать орендарю з номером `num`:

```

var    connectedData    =    await    privateDC.Rentals.Include(a =>
a.Leaser).Include(a => a.Worker).Include(a => a.AutoModel).Include(a =>
a.Promo).Include(a => a.Rentalprice).Include(a => a.Insurance).Include(a =>
a.AutoRental.Auto);

var rentals = connectedData.Where(a => a.Id_leaser == num);
rentals = rentals.Where(a => a.State == 4);
rentals = rentals.OrderByDescending(a => a.Id);
return View(await rentals.ToListAsync());

```

Метод `Task<IActionResult> Userrentals(long? num)` виконує оброблення запиту на отримання даних про активні замовлення (створені або

виконувані) на прокат автомобілей, орендарем в яких є орендар з номером num.

Метод `Task<IActionResult> Ownarentals()` виконує оброблення запиту на отримання даних про активні замовлення авторизованого користувача.

Метод `Task<IActionResult> History(long? num, DateTime date1, DateTime, date2)` виконує оброблення запиту на отримання даних про історію замовлень, що може характеризуватися номером num (може бути і пропущений) та проміжком виконання з date1 до date2 і відповідно є виконаними за своїм станом.

Метод `Task<IActionResult> Activerentals(long? num, DateTime date1, DateTime, date2)` виконує оброблення запиту на отримання даних про виконувані в даний момент замовлення, що можуть характеризуватися номером num (може бути і пропущений) та проміжком виконання з date1 до date2.

Метод `IActionResult Create()` виконує оброблення початкового GET-запиту до сторінки створення прокату.

Метод `Task<IActionResult> Create([Bind("Id, Date1, Date2, Price, Paid, State, Id_insurance, Id_rentalprice")] Rental rental, string promo)` виконує оброблення POST-запиту, який передбачає саме збереження внесених даних прокату до бази даних, при цьому оскільки промокод від користувача отриманий у вигляді рядка, а його ще потрібно перевірити і перетворити за успішної перевірки на об'єкт відповідної моделі, то параметр promo передається у метод окремо, рядком, а усі інші параметри прокату напряму прив'язуються з форми до відповідних характеристик.

Метод `Task<IActionResult> Edit(long? num)` виконує оброблення початкового GET-запиту до сторінки редагування прокату з номером num.

Метод `Task<IActionResult> Edit(long num, ([Bind("Id, Date1, Date2, Price, Paid, State, Id_insurance, Id_rentalprice")] Rental rental)` виконує оброблення POST-запиту, який передбачає саме збереження внесених оновлених даних прокату з номером num до бази даних.

У поданні забезпечується виведення відповідних даних, наприклад, на сторінці історії замовлень авторизованого користувача в таблиці виводяться дані про номер замовлення, стан замовлення та виробника і модель автомобіля наступним чином:

```
@foreach (var item in Model) {
    <tr>
        <th scope="row"><b>@string.Format("#{0}", item.Id)</b>
    </th>
    <td>
        @if (Model.State == 3) {
            <div class="tm-status-circle cancelled">
            </div>Неактив
        }
        else {
            <div class="tm-status-circle done">
            </div>Історія
        }
    </td>
    <td>
        <b>>@string.Format("{0} {1}", item.Automaker.Makename,
item.Modelname)</b>
    </td>
}
```

Подання створено на основі використання безкоштовного HTML-шаблону Product Admin [15] з використанням набору інструментів Bootstrap версії 4.1.3.

3.3 Висновки за розділом 3

Описано структуру розробленого програмного проєкту, створеного на основі використання архітектурного шаблону MVC. Описано логіку створення моделей, контролерів та подань. Приведено принципи організації роботи з даними у цих складових.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення програми

Розроблене програмне забезпечення призначене для виконання контролю і керування за замовленнями з прокату автомобілів зі сторони агентств з прокату та створення і моніторингу за замовленнями зі сторони клієнтів. Клієнти можуть переглядати всі наявні пропозиції, зручним чином отримувати інформацію про наявні тарифи та обирати потрібний. Безпосередньо видачою автомобілів займаються працівники агентств.

Програма призначена для роботи з агентствами, які мають розподілену систему пунктів видачі, при цьому працівники агентства пов'язуються з конкретним пунктом, що дозволяє їм працювати тільки з автомобілями цього пункту. Автомобіль можна взяти і повернути тільки в одному конкретному пункті прокату.

Передбачається тільки довгостроковий прокат, тобто базовим періодом роботи є місяць. Прокат може виконуватися на строк від 1 місяця і більше.

4.2 Умови виконання програми

Технічні засоби серверної сторони повинні включати такий мінімальний набір:

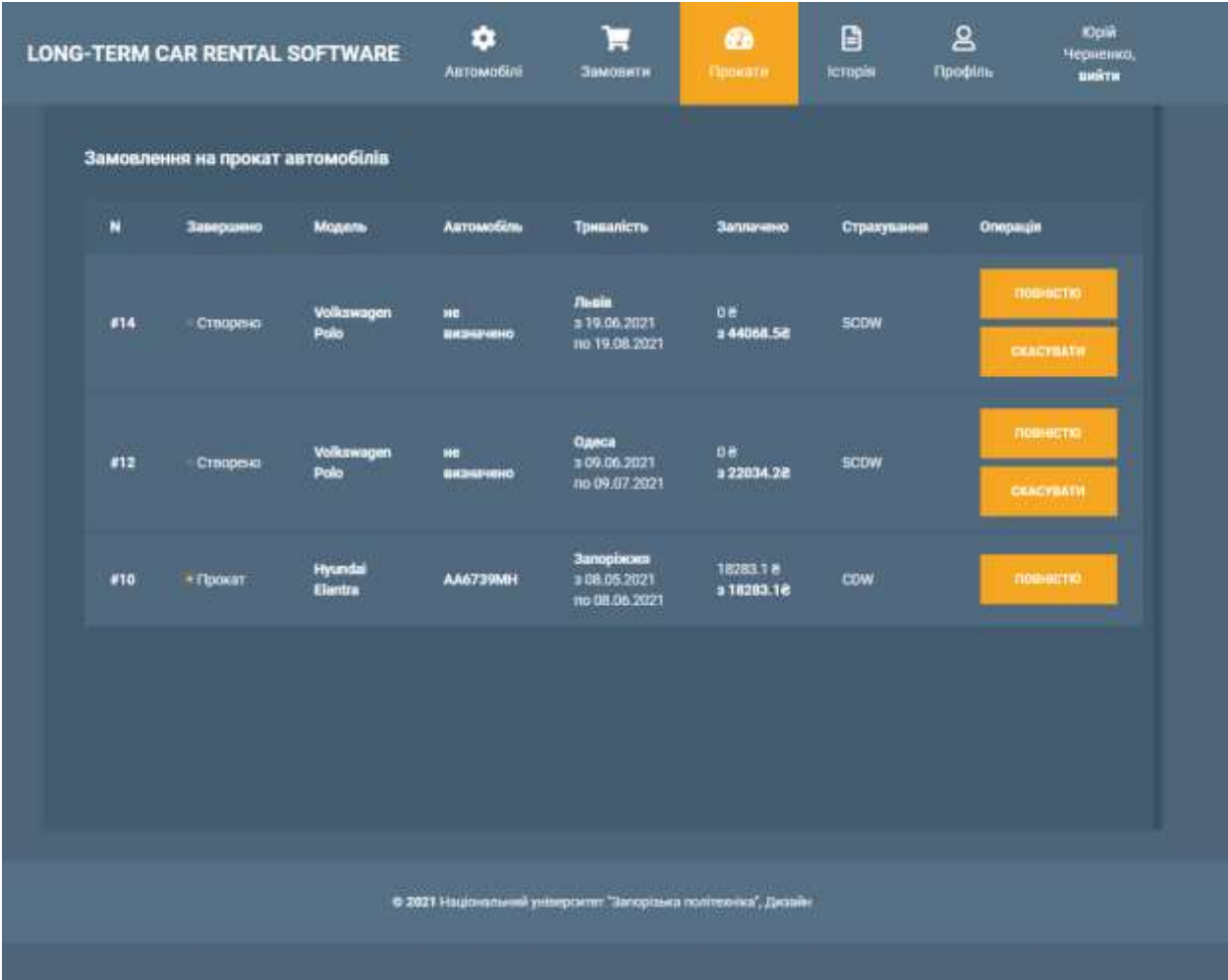
- процесор з 4 ядрами і тактовою частотою не менше 2,1 ГГц;
- оперативну пам'ять обсягом 8 Гб;
- 2 Гб доступного простору на жорсткому диску.

Обов'язковим є підключення до високошвидкісного інтернету (не менше ніж 100 Мбіт/с, бажано – 1 Гбіт/с).

4.3 Тестування сценаріїв роботи з програмою

Тестування роботи програми необхідно розпочати з обов'язкового запуску вебсервера Kestrel та сервера системи керування базами даних Microsoft SQL Server. Тоді сервер можна вважати активованим і можна виконувати запити з браузера клієнтської сторони.

Початково користувачу відображається сторінка авторизації клієнтського запису, після виконання якої реалізується відображення сторінки прокатів авторизованого користувача (рис. 4.1). На цій сторінці відображається перелік замовлень на прокат автомобіля.



LONG-TERM CAR RENTAL SOFTWARE							
Автомобілі Замовити Прокати Історія Профіль Юрій Черненко, вийти							
Замовлення на прокат автомобілів							
N	Завершено	Модель	Автомобіль	Тривалість	Заплачено	Страховання	Операція
#14	Створено	Volkswagen Polo	не визначено	Львів з 19.06.2021 по 19.08.2021	0 ₴ з 44068.58	SCDW	ПОВНІСТЬ СКАСУВАТИ
#12	Створено	Volkswagen Polo	не визначено	Одеса з 09.06.2021 по 09.07.2021	0 ₴ з 22034.28	SCDW	ПОВНІСТЬ СКАСУВАТИ
#10	Прокат	Hyundai Elantra	AA6739MH	Запоріжжя з 08.05.2021 по 08.06.2021	18283.1 ₴ з 18283.1 ₴	CDW	ПОВНІСТЬ

© 2021 Національний університет "Запорізька політехніка", Дизайн

Рисунок 4.1 – Перегляд власних активних прокатів автомобілів користувачем

Усі замовлення відсортовані за номером, починаючи з найбільшого.

До даної сторінки відносяться замовлення, які тільки створені клієнтом та підтверджені працівником агентства, тобто ті, для яких видано автомобіль у прокат.

Для кожного замовлення відображається його номер, стан (створений відображається сірим кольором, прокат відображається помаранчевим), модель автомобіля, номерний знак конкретного автомобіля (для замовлень, які вже знаходяться в прокаті), місто прокатного пункту, період активності замовлення, інформація про сплату замовлення (сплачену суму та вартість замовлення), інформацію про страхування та доступні операції.

Доступні операції для замовлення включають:

- кнопку «Скасувати», що дозволяє завершити замовлення (у такому випадку замовлення не буде відображено працівнику агентства для вибору підтвердження або скасування його і буде переміщено в історію замовлень);
- кнопку «Повністю», що дозволяє перейти до сторінки перегляду повної інформації про замовлення.

Пункти меню для клієнта включають:

- «Автомобілі», де можна переглянути доступні автомобілі за обраним пунктом видачі автомобілів (агентством) та тарифи на їх прокат;
- «Замовити», де можна створити нове замовлення на прокат автомобіля;
- «Прокати», де можна переглянути активні замовлення на прокат автомобілів;
- «Історія», де можна переглянути інформацію про замовлення, які вже виконано;
- «Профіль», де можна переглянути і відредагувати дані власного користувацького профілю.

На сторінці «Історія» (рис. 4.2) відображені замовлення, які вже не виконуються. До таких відносяться скасовані замовлення, позначені червоним кольором, та завершені замовлення, позначені зеленим кольором.

Щоб переглянути детальну інформацію про замовлення, потрібно натиснути кнопку «Повністю».

LONG-TERM CAR RENTAL SOFTWARE

Автомобілі Замовити Прокати **Історія** Профіль Юрій Черненко, вийти

Замовлення на прокат автомобілів

№	Завершено	Модель	Автомобіль	Тривалість	Заплачено	Страховання	Операція
#13	Неактив	Volkswagen Polo	не визначено	Львів з 09.05.2021 по 09.06.2021	0 ₪ з 22034.3₪	SCDW	повністю
#9	Історія	Toyota Corolla	AK2455AA	Запоріжжя з 07.05.2021 по 07.06.2021	22312.1 ₪ з 22312.1₪	SCDW	повністю
#8	Неактив	Toyota Corolla	не визначено	Запоріжжя з 04.05.2021 по 04.06.2021	0 ₪ з 22312.1₪	SCDW	повністю
#5	Історія	Renault Logan	AK3791AA	Запоріжжя з 01.05.2021 по 01.06.2021	10308.6 ₪ з 10308.6₪	немає	повністю

© 2021 Національний університет "Запорізька політехніка", Дизайн

Рисунок 4.2 – Перегляд історії прокатів автомобілів користувачем

Сторінка «Замовити» (рис. 4.3) дозволяє сформувати замовлення на прокат автомобіля.

У замовленні потрібно спочатку обрати дату початку та дату завершення прокату за допомогою календаря або введення відповідної дати. Далі потрібно обрати пункт прокату. За потреби необхідно задати виробника автомобіля. Якщо конкретного виробника не обрано, то активним є пункт «Усі», який дозволяє переглянути всі доступні моделі автомобілів на задані дати.

LONG-TERM CAR RENTAL SOFTWARE

Автомобілі **Замовити** Прокати Історія Профіль Олександр Звятицький, вийти

Створення замовлення на прокат автомобіля

Починаючи з дати: 18.05.2021

Пункт прокату: Хмель (Запоріжжя, пр. Соборний)

Завершуючи датою: 18.06.2021

Виробник автомобіля: Hyundai

ЗНАЙТИ

	Hyundai Accent Тип: Седан Клас: Економ Трансмісія: Автомат Двигун: 1.7 Паливо: Бензин Витрата в місті: 9.8 Витрата на трасі: 7.2	Пасажирів: 5 Дверей: 4 Кондиціонер Клімат-контроль Багажник: 268 л	Тариф до 1 міс.: 11447.8 ₴ / міс. Депозит: 42000 ₴ Тариф до 5 міс.: 10319.4 ₴ / міс. Депозит: 63000 ₴ Тариф до 12 міс.: 9762.9 ₴ / міс. Депозит: 84000 ₴
	Hyundai Elantra Тип: Седан Клас: Економ Трансмісія: Автомат Двигун: 2.0 Паливо: Бензин Витрата в місті: 10.7 Витрата на трасі: 8.3	Пасажирів: 5 Дверей: 4 Кондиціонер Клімат-контроль Багажник: 342 л	Тариф до 1 міс.: 12142.4 ₴ / міс. Депозит: 43000 ₴ Тариф до 5 міс.: 11718.2 ₴ / міс. Депозит: 64500 ₴ Тариф до 12 міс.: 11102.8 ₴ / міс. Депозит: 86000 ₴

Тариф до 1 міс.

Рисунок 4.3 – Створення замовлення прокату автомобіля користувачем

Після внесення даних потрібно натиснути на кнопку «Знайти», після чого нижче в таблиці буде відображено всі доступні моделі на вказані дати.

За кожною моделлю виводиться інформація про неї, а також інформація про доступні тарифи за цим автомобілем. За необхідності можна змінити дати, враховуючи отриману інформацію про тарифи.

У останньому стовпчику потрібно поставити галочку навпроти відповідної моделі, а внизу натиснути на кнопку «Обрати», щоб відзначити цю модель автомобіля для оформлення замовлення.

Після цього потрібно обрати варіант страховки автомобіля з випадуючого переліку. Далі потрібно сформуванати набір додаткових послуг. Для цього потрібно обрати конкретну послугу, вказати кількість для неї, та додати до таблиці.

У результаті для збереження замовлення потрібно натиснути на кнопку «Створити замовлення».

Для користувача працівника агентства реалізовано окремий інтерфейс авторизації. Після успішної авторизації працівнику агентства надається меню з наступних пунктів:

- «Автомобілі», де можна керувати внесенням даних щодо існуючих автомобілів у прокат;

- «Замовити», де можна створювати замовлення на прокат автомобілів зі сторони працівника агентства, тобто з фіксуванням факту видачі автомобіля в прокат;

- «Прокати», де можна переглядати інформацію про замовлення різного стану та керувати ними;

- «Пошук», де можна виконувати пошук автомобілів під час взаємодії з клієнтом;

- «Клієнти», де можна переглядати дані клієнтів та за потреби редагувати їх.

Розділ «Автомобілі» включає пункти меню, що дозволяють працювати з переліком наявних моделей (рис. 4.4), створювати нову модель, керувати варіантом страхувань, керувати наявними додатковими послугами. Фактично цей розділ створений для прямої взаємодії з базою даних щодо внесення нових даних і редагування існуючих.

Під час роботи з моделями автомобілів надається таблиця, у якій потрібно обрати одну з моделей автомобілів. У таблиці представлені усі наявні моделі. Для того щоб обрати модель автомобіля, потрібно натиснути галочку в першому стовпці. Далі необхідно натиснути на кнопку «Редагувати», якщо потрібно відкоригувати дані про цю модель.

Якщо потрібно змінити тарифи, за якими обслуговується дана модель в агентстві, то потрібно після обрання відповідної моделі натиснути на кнопку «Вартість».

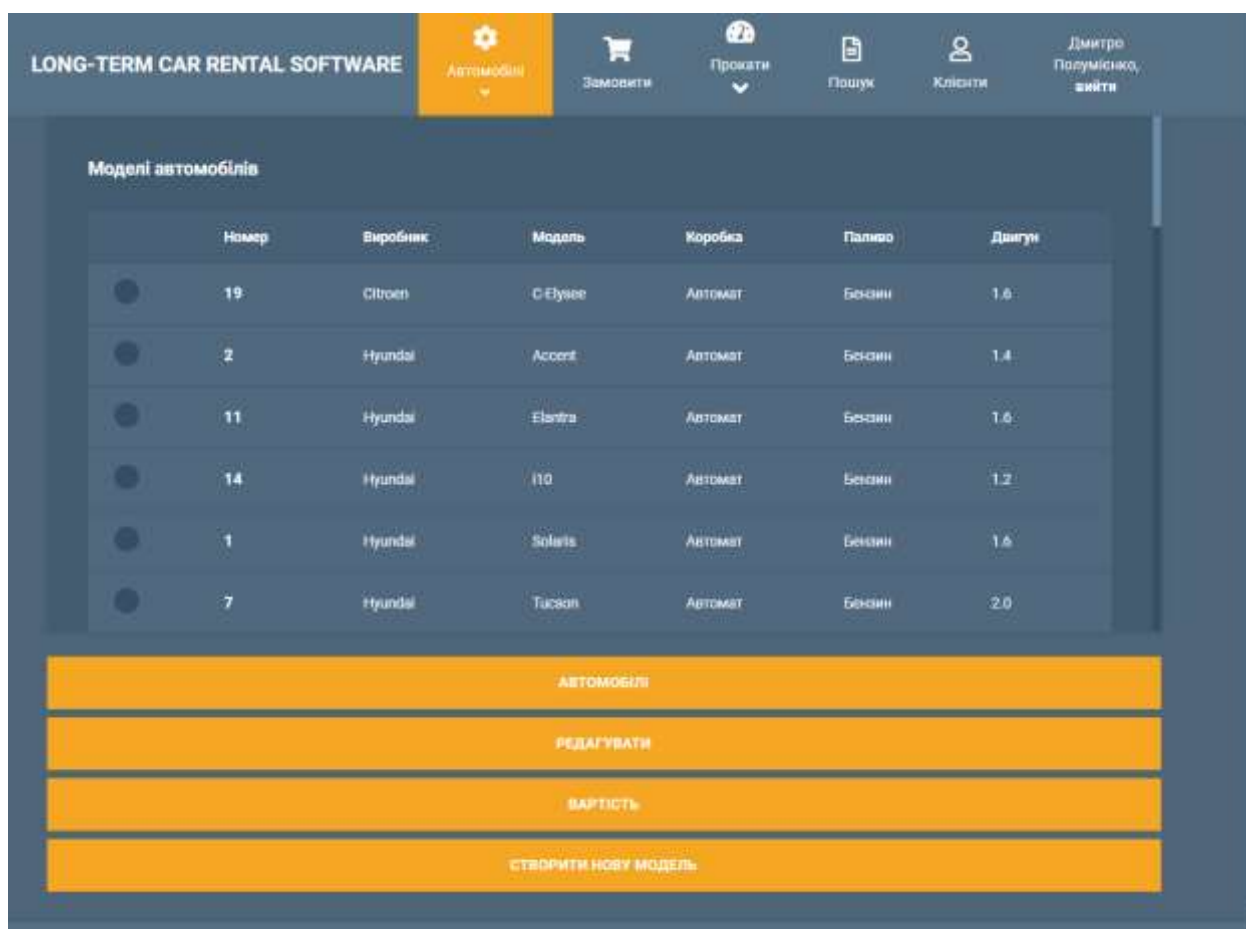


Рисунок 4.4 – Керування моделями автомобілів

Інформація про кожну модель в таблиці включає її номер в системі, виробника, саму назву моделі, коробку передач, використовуване паливо та двигун. Якщо потрібно переглянути детальнішу інформацію, то потрібно натиснути на назву моделі, тоді відкриється окрема сторінка з повною інформацією про модель (зображенням, кількістю пасажирів, дверей тощо).

Можна також керувати конкретними автомобілями цієї моделі в агентстві. Для цього потрібно обрати конкретну модель, а далі натиснути на кнопку «Автомобілі», тоді відкриється нова сторінка (рис. 4.5), в таблиці якої будуть відображені всі автомобілі цієї моделі, наявні в агентстві, за яким працює авторизований користувач. Тобто виводяться не всі автомобілі моделі, наявні в базі даних, а тільки ті, що закріплені за пунктом видачі, з яким працює авторизований користувач, усі інші автомобілі користувачу не видно. Кожен автомобіль виводиться за номерним знаком, роком

виробництва, станом, кількістю виконаних прокатів за його участю, VIN-номером.

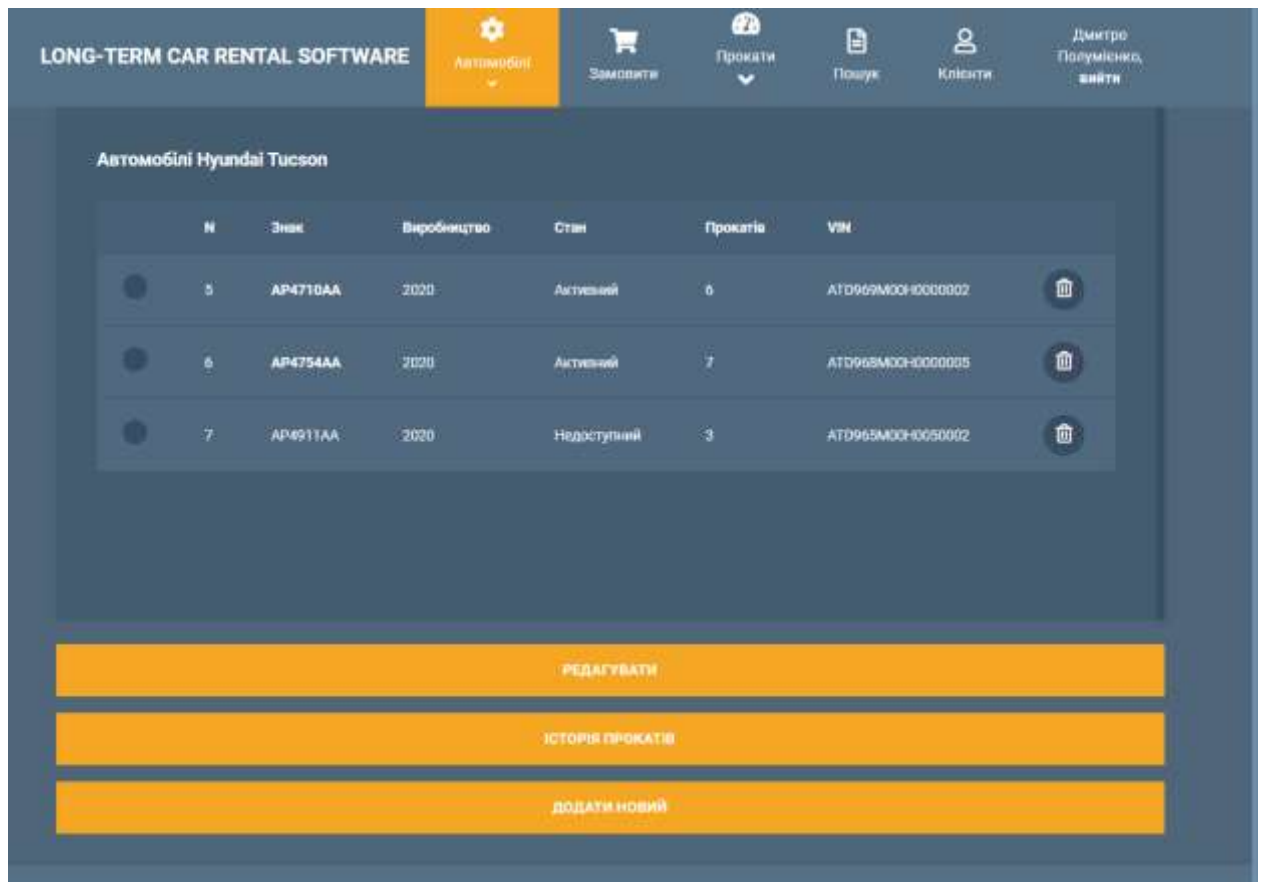


Рисунок 4.5 – Керування автомобілями

Для того щоб деактивувати використання певного автомобіля, потрібно натиснути на іконку з зображенням корзини біля номерного знаку. При цьому автомобіль не вилучається, а тільки переводиться до неактивного стану. Це значить, що під час створення замовлень прокату користувачі не зможуть обрати цей автомобіль.

Для того щоб переглянути повну інформацію про автомобіль, потрібно натиснути на номерний знак, тоді відкриється окрема сторінка.

Якщо обрати один з автомобілів за допомогою галочки в лівому стовпці таблиці, то далі можна натиснути на кнопку «Редагувати», щоб відредагувати внесені дані автомобіля. Якщо при цьому натиснути на кнопку

«Історія прокатів», то відкриється сторінка з історією виконання прокатів даного автомобіля на основі переліку відповідних замовлень.

Кнопка «Додати новий» дозволяє внести новий автомобіль обраної моделі до бази даних, при цьому він одразу буде прив'язаний до пункту прокату, з яким працює даний працівник.

Для додавання нової моделі в систему достатньо на цій сторінці натиснути на кнопку «Створити нову модель». Після цього відкриється окрема сторінка (рис. 4.6).

The screenshot displays the 'LONG-TERM CAR RENTAL SOFTWARE' interface. The top navigation bar includes icons for 'Автомобілі' (Cars), 'Замовити' (Book), 'Прокати' (Rentals), 'Пошук' (Search), and 'Клієнти' (Clients). The main content area is titled 'Створення нової моделі в прокаті' (Creating a new car model for rental). The form contains the following fields and options:

- Виробник** (Manufacturer): Lexus (dropdown)
- Назва моделі** (Model name): NX
- Двигун** (Engine): 2.0
- Коробка передач** (Transmission): Автомат (Automatic) (dropdown)
- Тип палива** (Fuel type): Бензин (Gasoline) (dropdown)
- Клас** (Class): Преміум (Premium) (dropdown)
- Тип** (Type): Кросовер (SUV) (dropdown)
- Пасажирів** (Passengers): 5
- Дверей** (Doors): 5
- Обсяг багажника, л** (Luggage capacity, L): [empty field]
- Кондиціонування** (Air conditioning): Клімат-контроль (Climate control) (dropdown)
- Пальне у місті** (Fuel in city): [empty field]
- Пальне на трасі** (Fuel on highway): [empty field]

There is a 'Завантажити зображення' (Upload image) button next to a placeholder image. At the bottom of the form is a large orange button labeled 'СТВОРИТИ МОДЕЛЬ' (Create model).

Рисунок 4.6 – Створення нової моделі автомобіля

Під час створення нової моделі потрібно спочатку обрати виробника автомобіля з випадаючого списку, далі ввести конкретну назву автомобіля, задати обсяг двигуна, обрати коробку передач з випадаючого списку

(автомат або механіка), обрати тип палива з випадаючого списку (бензин, дизель, газ або електрика), задати кількість пасажирів, кількість дверей, обсяг витрат пального в місті (кількість літрів на 100 кілометрів), обсяг витрат пального на трасі, обсяг багажника, обрати клас автомобіля з випадаючого списку, обрати тип автомобіля з випадаючого списку, обрати тип системи кондиціонування або її відсутність з випадаючого списку.

Окрім того можна завантажити зображення моделі, для цього потрібно натиснути на кнопку «Завантажити зображення».

Для збереження даних внесеної моделі потрібно натиснути на кнопку «Створити модель».

Розділ «Клієнти» (рис. 4.7) меню працівника агентства дозволяє отримати на окремій сторінці таблицю з переліком створених клієнтів. Інформація про кожного клієнта включає його прізвище, ім'я, номер, країну, номер телефону, баланс рахунку (обчислюється як різниця між сумами сплачених замовлень користувача та вартістю замовлень користувача) та витрати (обчислюється як абсолютне значення сплачених коштів за замовлення користувачем).

Для кожного клієнта можна натиснути на кнопку «Редагувати» і перейти до редагування даних клієнта. Це потрібно для коригування даних паспорта, ідентифікаційного коду тощо.

Для того щоб переглянути історію замовлень користувача, потрібно натиснути на кнопку «Історія» у рядку користувача в таблиці.

Клієнти реєструються в системі самостійно.

Для більш швидкого пошуку потрібного користувача необхідно скористатися полем номеру телефону під таблицею. Після введення потрібного номеру необхідно натиснути на кнопку «Знайти», тоді в таблиці буде відображено тільки тих користувачів, що мають зазначений номер телефону.

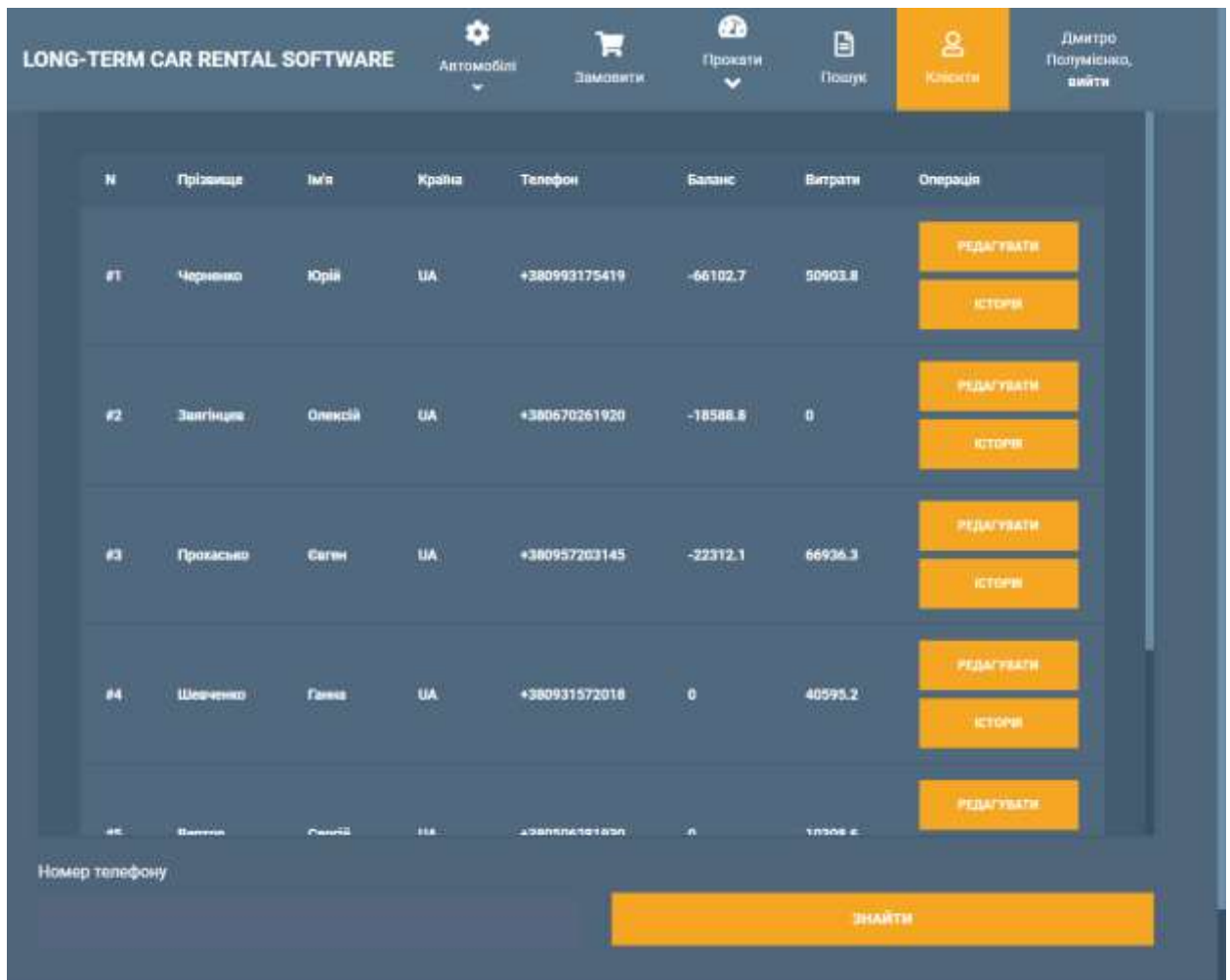


Рисунок 4.7 – Керування профілями користувачів

Розділ «Прокати» має набір пунктів меню, що дозволяють скористатися відповідною функціональністю програми. Підрозділи включають засоби перегляду історії прокатів, активних прокатів, замовлень.

До перших належать замовлення, завершення яких вже було підтверджено, тобто автомобіль за ними клієнт вже повернув.

До других належать прокати, які відбуваються в поточний момент, тобто клієнт забрав автомобіль, але ще не повернув. За цими замовленнями працівник може підтвердити їх повернення, таким чином зафіксувати остаточну вартість та визначити внесену користувачем суму.

До третіх належать замовлення, які вже були сформовані клієнтом, але ще не підтверджені працівником агентства (рис. 4.8).

LONG-TERM CAR RENTAL SOFTWARE

Автомобілі | Замовити | Прокати | Пошук | Клієнти | Дмитро Полумісний, вийти

Створені користувачами замовлення

N	Клієнт	Модель	Тривалість	Автомобіль	Страхування	Вартість	Операція
#12	Черненко Юрій	Volkswagen Polo	з 09.05.2021 по 09.06.2021	AP3220AB ▾	SCDW	22034.28	ПІДТВЕРДИТИ СКАСУВАТИ ПОВНІСТЬ
#13	Завітінська Олексій	Renault Logan	з 08.05.2021 по 08.06.2021	AP7318AA ▾	SCDW	18588.86	ПІДТВЕРДИТИ СКАСУВАТИ ПОВНІСТЬ
#14	Черненко Юрій	Volkswagen Polo	з 19.06.2021 по 19.08.2021	AP3220AB ▾	SCDW	44068.52	ПІДТВЕРДИТИ СКАСУВАТИ ПОВНІСТЬ

© 2021 Національний університет "Запорізька політехніка", Дизайн

Рисунок 4.8 – Керування замовленнями клієнтів

Під час перегляду створених клієнтом замовлень формується таблиця, що включає наступну інформацію: номер замовлення, дані клієнта, дані про модель автомобіля, тривалість прокату, обрану клієнтом страховку та вартість прокату. Якщо працівник погоджує прокат, то він має спочатку обрати з випадального списку в стовпці «Автомобіль» конкретний номер знак автомобіля цієї моделі (включаються тільки автомобілі, пов'язані з пунктом видачі, до якого належить працівник), а далі натиснути на кнопку «Підтвердити» для збереження даних та підтвердження факту видачі автомобіля клієнту.

Якщо користувач бажає переглянути детальну інформацію про замовлення, що включає як дані моделі, так і дані користувача, то потрібно

натиснути на кнопку «Повністю». Звідти можна побачити дані, що відображаються на сторінці «Клієнти», але за конкретним клієнтом.

Якщо працівник агентства не бажає погодити видачу автомобіля, то він має натиснути на кнопку «Скасувати», тоді стан замовлення буде змінено на відповідний, а клієнт автомобіль не отримає.

Підрозділ «Історія прокатів» (рис. 4.9) розділу «Прокати» надає засоби перегляду історію прокатів автомобілів, при цьому основними засобами відфільтровування замовлень є засоби визначення проміжку дат та визначення номеру замовлення. Якщо користувач знає точний номер замовлення, то він має ввести його в поле «Номер замовлення» та натиснути на кнопку «Знайти». Якщо користувач бажає відобразити не всі замовлення, а тільки за заданий проміжок дат, то він має ввести або обрати з календаря дати в полях «Починаючи з дати» та «Завершуючи датою». У вказаний проміжок дат має потрапляти увесь прокат повністю, тобто і дата початку прокату, і дата завершення прокату. Після вибору цих параметрів пошуку також потрібно натиснути на кнопку «Знайти».

Після натискання на кнопку «Знайти» в таблиці відображаються замовлення, які підпадають під критерії пошуку. Для кожного замовлення видається його номер, інформація про клієнта, тривалість замовлення, номер автомобіля, обраний варіант страхування та сплачену суму за замовлення разом з його вартістю. Для перегляду детальних даних замовлення потрібно натиснути на кнопку «Повністю».

Якщо замовлення було скасовано самим клієнтом, то відповідно конкретний автомобіль до нього не прив'язаний і в стовпці «Автомобіль» виводиться не конкретний номерний знак, а значення «не визначено».

Після завершення виконання потрібних дій користувач може вийти з вебзастосунку. Для цього йому потрібно натиснути на посилання «вийти» поруч з власним іменем у рядку меню.

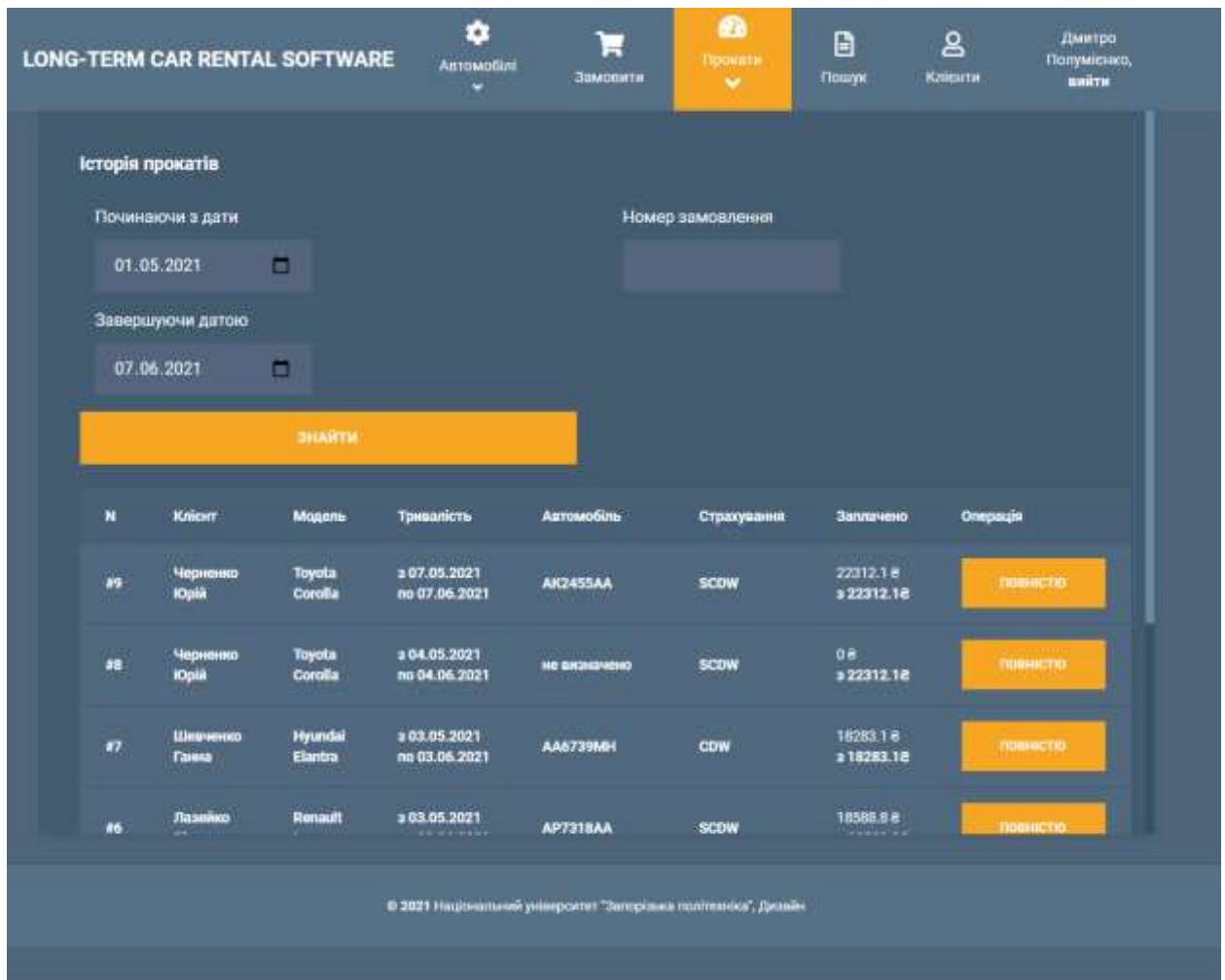


Рисунок 4.9 – Перегляд історії прокатів автомобілів

4.4 Висновки за розділом 4

Описано процес виконання функціональних можливостей програми для клієнтів та працівників агентства у вигляді відповідних ланцюжків взаємодії користувачів з програмою та представлено відповідні інтерфейси.

ВИСНОВКИ

Дипломну кваліфікаційну роботу бакалавра присвячено розробленню програмного забезпечення довгострокового прокату автомобілів для підтримки роботи розподіленого агентства. У якості довгострокового прокату розглядалась оренда автомобіля на строк не менше 1 місяця. За результатами аналізу предметної області та існуючих програмних аналогів визначено функціональні вимоги до програми. Виконано моделювання роботи користувача з програмою на основі використання діаграм прецедентів.

Розроблено програмне забезпечення у вигляді вебзастосунку на основі застосування фреймворку розробки вебзастосунків ASP.NET для платформи .NET та мови програмування C#, яку обрано для реалізації. Роботу з даними реалізовано на основі використання системи керування базами даних Microsoft SQL Server та модуля об'єктно-реляційного відображення Entity Framework, виконано проєктування схеми бази даних. Описано процес реалізації вебзастосунку за допомогою обраного фреймворку. Структуру вебзастосунку створено на основі використання архітектурного шаблону MVC.

Описано процес виконання функціональних можливостей програми для клієнтів та працівників агентства у вигляді відповідних ланцюжків взаємодії користувачів з програмою та представлено відповідні користувацькі інтерфейси. Розроблене програмне забезпечення призначене для виконання контролю і керування за замовленнями з прокату автомобілів зі сторони агентств з прокату та створення і моніторингу за замовленнями зі сторони клієнтів. Клієнти можуть переглядати всі наявні пропозиції, зручним чином отримувати інформацію про наявні тарифи та обирати потрібний під час планування.

Усі завдання, які було визначено у дипломній кваліфікаційній роботі бакалавра, виконано у повному обсязі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Лізинг авто [Електрон. ресурс] : що це й як він працює для приватних осіб? – Режим доступу : <https://7cars.com.ua/ua/shho-take-lizing-avto-yak-fizichnij-osobi-vzyati-avtomobil-v-lizing/>
2. Long Term Car Rental vs Leasing | SIXT rent a car Magazine [Electronic resource]. – Access mode : <https://www.sixt.com/magazine/tips/long-term-car-rental-vs-leasing/>
3. Miller, D. Leasing a car [Electronic resource] : How to do it and mistakes to avoid / D. Miller. – Access mode : <https://www.bankrate.com/loans/auto-loans/car-leasing-mistakes-to-avoid/>
4. Fink, A. Modeling and solving the short-term car rental logistics problem [Text] / A. Fink, T. Reiners // Transportation Research Part E Logistics and Transportation Review. – 2006. – № 42 (4). – Pp. 272-292.
5. Free Vehicle Identification Number (VIN) Decoder & Lookup [Electronic resource]. – Access mode : <https://driving-tests.org/vin-decoder/>
6. The Four Types of Rental Car Insurance, Explained [Electronic resource]. – Access mode : <https://www.allianztravelinsurance.com/travel/rental-cars/rental-car-insurance-explained.htm>
7. Оренда авто за кордоном [Електрон. ресурс] : як заощадити на страховці. – Режим доступу : <https://travelyourway.com.ua/ua/planirovanie-samostoyatelnyh-puteshestvij/avtomobilistam/arenda-avto-kak-sekonomit-na-strahovke/>
8. Rental Cars at Low, Affordable Rates | Enterprise Rent-A-Car [Electronic resource]. – Access mode : <https://www.enterprise.com/en/home.html>
9. SIXT оренда авто в Києві, Львові - Прокат автомобілів в Україні [Електрон. ресурс]. – Режим доступу : <https://www.sixt.ua/>
10. Лізинг авто Київ. Автомобіль в лізинг, Україна | AVIS [Електрон. ресурс]. – Режим доступу : <https://avisleasing.com.ua/>
11. Stellman, A. Head First C# [Text] : A Learner's Guide to Real-World

Programming with C# and .NET Core / Andrew Stellman , Jennifer Greene. – Sebastopol, CA : O'Reilly Media, 2021. – 800 p.

12. Price, M. J. C# 9 and .NET 5 – Modern Cross-Platform Development [Text] : Build intelligent apps, websites, and services with Blazor, ASP.NET Core, and Entity Framework Core using Visual Studio Code / M. J. Price. – Birmingham : Packt Publishing, 2020. – 822 p.

13. Tatroe, K. Programming PHP [Electronic resource] : Creating Dynamic Web Pages / K. Tatroe. – Sebastopol : O'Reilly, 2020. – 544 p.

14. Top Features that Make ASP.NET Technology Your Ultimate Choice [Electronic resource]. – Access mode : <https://medium.com/@ISHIRInc/top-features-that-make-asp-net-technology-your-ultimate-choice-d04829cc0846>

15. Free Template 524 Product Admin [Electronic resource]. – Access mode : <https://templatemo.com/tm-524-product-admin>

ДОДАТОК А
Технічне завдання

Вступ

Програма має бути призначена для роботи з агентствами, які мають розподілену систему пунктів видачі, при цьому працівники агентства пов'язуються з конкретним пунктом, що дозволяє їм працювати тільки з автомобілями цього пункту. Автомобіль можна взяти і повернути тільки в одному конкретному пункті прокату.

Передбачається тільки довгостроковий прокат, тобто базовим періодом роботи є місяць. Прокат може виконуватися на строк від 1 місяця і більше.

Безпосередньо видачею автомобілів займаються працівники агентств.

A.1 Підстави для розробки

Розробка має бути виконана в межах завдання на дипломну кваліфікаційну роботу, яке було поставлено за темою «Програмне забезпечення довгострокового прокату автомобілів» у відповідності з наказом № 103 від 30 березня 2021 р. за Національним університетом «Запорізька політехніка».

A.2 Призначення розробки

Програмне забезпечення має бути призначене для виконання контролю і керування за замовленнями з прокату автомобілів зі сторони агентств з прокату та створення і моніторингу за замовленнями зі сторони клієнтів.

Клієнти в цій системі можуть переглядати всі наявні пропозиції, зручним чином отримувати інформацію про наявні тарифи та обирати потрібний.

A.3 Основні вимоги до програми

A.3.1 Вимоги до функціональних характеристик

Програма має надавати такі функціональні особливості:

- авторизація клієнтів і працівників агентства;
- реєстрація клієнтів;
- внесення даних автомобілів для прокату;
- керування автомобілями в прокаті;
- визначення вартості прокату автомобіля;
- керування прокатами автомобілів;
- створення замовлення на прокат автомобіля клієнтом або працівником агентства;
- підтвердження прокату автомобіля працівником агентства;
- повернення автомобіля;
- пошук автомобіля для прокату;
- визначення автомобілів у прокаті;
- перегляд і керування профілем користувача;
- перегляд історії прокату автомобілів клієнтом;
- пошук історії прокату автомобілів;
- перегляд власних замовлень прокату автомобілів.

A.3.2 Умови експлуатації

Для експлуатації програми особливих вмінь від користувачів окрім вміння користуватися комп'ютером не висувається.

Для забезпечення роботи бази даних на сервері необхідні вміння системного адміністратора для підтримки роботи вебсервера сервера системи керування базами даних Microsoft SQL Server.

A.3.3 Вимоги до складу та параметрів технічних засобів

Технічні засоби серверної сторони повинні включати щонайменше:

- процесор з 4 ядрами і тактовою частотою не менше 2,1 ГГц;
- оперативну пам'ять обсягом 8 Гб;
- 2 Гб доступного простору на жорсткому диску, адже зокрема на сервер мають завантажуватися фотографії моделей автомобілів.

A.4 Порядок контролю та приймання

Принципи контролю виконання роботи визначаються календарним планом, розташованим у завданні на дипломну кваліфікаційну роботу, де вказано назви етапів, строки їх виконання та результат.

ДОДАТОК Б
Текст програми

Б.1 Текст файла AutoModelController.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using LongCarRental.Data;
using LongCarRental.Models;

namespace LongCarRental.Controllers
{
    public class AutoModelController : Controller
    {
        private readonly RentalDBContext privateDC;

        public AutoModelController (RentalDBContext modelcont)
        {
            privateDC = modelcont;
        }

        [Authorize]
        public async Task<IActionResult> Index()
        {
            ViewData["AutoModels"] =
privateDC.AutoModels.Include(a => a.Automaker);
            IQueryable<Worker> workers = privateDC.Workers;
            var worker = await workers.FirstOrDefaultAsync(a =>
a.login == User.Identity.Name);
            var agency = await
privateDC.Agencies.FirstOrDefaultAsync(a => a.Id == worker.Id_agency);
            ViewData["Autos"] = privateDC.Autos.Where(i =>
i.Id_agency == agency.Id);
            return View();
        }

        [Authorize]
        public async Task<IActionResult> Info(long? num)

```

```

{
    if (num == null)
    {
        return NotFound();
    }

    var automodel = await privateDC.AutoModels
        .Include(a => a.Automaker)
        .FirstOrDefaultAsync(b => b.Id == num);
    if (automodel == null)
    {
        return NotFound();
    }

    return View(automodel);
}

[Authorize]
public IActionResult Create()
{
    ViewData["AutoMakers"] = new
SelectList(privateDC.AutoMakers, "Id", "Makername");
    return View();
}

[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Create([Bind("Id,
Modelname, Gearbox, Photo, Engine, Fuel, Fuelrate, Cityfuelrate,
Passengernumber, Doornumber, Autotype, Luggage, Conditioner,
Id_automaker, Autoclass, Inscoeff")] Automodel automodel)
{
    if (ModelState.IsValid)
    {
        privateDC.Add(automodel);
        await privateDC.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
    }
    ViewData["Id_automaker"] = new
SelectList(privateDC.AutoMakers, "Id", "Id", automodel.Id_automaker);
    return View(automodel);
}

```

```

[Authorize]
public async Task<IActionResult> Edit(long? num)
{
    if (num == null)
    {
        return NotFound();
    }

    var automodel = await
privateDC.AutoModels.FindAsync(num);
    if (automodel == null)
    {
        return NotFound();
    }

    ViewData["AutoMakers"] = new
SelectList(privateDC.AutoMakers, "Id", "Makername");
    return View(automodel);
}

[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Edit(long num,
[Bind("Id, Modelname, Gearbox, Photo, Engine, Fuel, Fuelrate,
Cityfuelrate, Passengernumber, Doornumber, Autotype, Luggage,
Conditioner, Id_automaker, Autoclass, Inscoeff")] Automodel automodel)
{
    if (num != automodel.Id)
    {
        return NotFound();
    }

    if (ModelState.IsValid)
    {
        try
        {
            privateDC.Update(automodel);
            await privateDC.SaveChangesAsync();
        }
        catch (DbUpdateConcurrencyException)
        {
            if (!AutoModelExists(automodel.Id))
            {

```

```

        return NotFound();
    }
    else
    {
        throw;
    }
}
return RedirectToAction(nameof(Index));
}
ViewData["Id_automaker"] = new
SelectList(privateDC.AutoMakers, "Id", "Id", automodel.Id_automaker);
return View(automodel);
}

```

Б.2 Текст файла AutoController.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using LongCarRental.Data;
using LongCarRental.Models;

namespace LongCarRental.Controllers
{
    public class AutoController : Controller
    {
        private readonly RentalDBContext privateDC;

        public AutoController (RentalDBContext modelcont)
        {
            privateDC = modelcont;
        }

        [Authorize]
        public async Task<IActionResult> Index(int amodel)
        {
            IQueryable<Worker> workers = privateDC.Workers;

```

```

        var worker = await workers.FirstOrDefaultAsync(a =>
a.login == User.Identity.Name);
        var agency = await
privateDC.Agencies.FirstOrDefaultAsync(a => a.Id == worker.Id_agency);
        var autos = privateDC.Autos.Where(i => i.Id_agency
== agency.Id);
        ViewData["Autos"] = autos.Where(a => a.Id_automodel
== amodel);
        return View();
    }

```

```

[Authorize]
public async Task<IActionResult> Info(long? num)
{
    if (num == null)
    {
        return NotFound();
    }

    var auto = await privateDC.Autos
        .Include(a => a.Automodel)
        .Include(a => a.Agency)
        .FirstOrDefaultAsync(b => b.Id == num);
    if (auto == null)
    {
        return NotFound();
    }

    return View(auto);
}

```

```

[Authorize]
public IActionResult Create()
{
    return View();
}

```

```

[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Create([Bind("Id,
Lastcheckup, Made, Information, State, Rentals, Id_automodel, Vin,
Number, Color")] Auto auto)
{

```

```

        if (ModelState.IsValid)
        {
            IQueryable<Worker> workers = privateDC.Workers;
            var worker = await workers.FirstOrDefaultAsync(a
=> a.login == User.Identity.Name);
            var agency = await
privateDC.Agencies.FirstOrDefaultAsync(a => a.Id == worker.Id_agency);
            auto.Id_agency = agency.Id;
            privateDC.Add(auto);
            await privateDC.SaveChangesAsync();
            return RedirectToAction(nameof(Index));
        }
        return View(auto);
    }
}

```

```

[Authorize]
public async Task<IActionResult> Edit(long? num)
{
    if (num == null)
    {
        return NotFound();
    }

    var auto = await privateDC.Autos.FindAsync(num);
    if (auto == null)
    {
        return NotFound();
    }

    return View(auto);
}

```

```

[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Edit(long num,
[Bind("Id, Lastcheckup, Made, Information, State, Rentals,
Id_automodel, Id_agency, Vin, Number, Color")] Auto auto)
{
    if (num != auto.Id)
    {
        return NotFound();
    }
}

```

```

        if (ModelState.IsValid)
        {
            try
            {
                privateDC.Update(auto);
                await privateDC.SaveChangesAsync();
            }
            catch (DbUpdateConcurrencyException)
            {
                if (!AutoExists(auto.Id))
                {
                    return NotFound();
                }
                else
                {
                    throw;
                }
            }
            return RedirectToAction(nameof(Index));
        }
        ViewData["Id_automodel"] = new
SelectList(privateDC.AutoModels, "Id", "Id", auto.Id_automodel);
        ViewData["Id_agency"] = new
SelectList(privateDC.Agencies, "Id", "Id", auto.Id_agency);
        return View(auto);
    }

```

Б.3 Текст файла RentalController.cs

```

using System;
using System.Collections.Generic;
using System.Globalization;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using LongCarRental.Data;
using LongCarRental.Models;

namespace LongCarRental.Controllers

```

```

{
    public class RentalController : Controller
    {
        private readonly RentalDBContext privateDC;

        public RentalController (RentalDBContext modelcont)
        {
            privateDC = modelcont;
        }

        [Authorize]
        public async Task<IActionResult> Ownhistory()
        {
            var connectedData = await
privateDC.Rentals.Include(a => a.Leser).Include(a =>
a.Worker).Include(a => a.AutoModel).Include(a => a.Promo).Include(a =>
a.Rentalprice).Include(a => a.Insurance).Include(a =>
a.AutoRental.Auto);
            var login = User.Identity.Name;
            IQueryable<Leaser> leasers = privateDC.Lesers;
            var leaser = leasers.FirstOrDefault(a =>
a.Login == login);
            var rentals = connectedData.Where(a => a.Id_leaser
== leaser.Id);

            rentals = rentals.Where(a => a.State == 4);
            rentals = rentals.OrderByDescending(a => a.Id);
            return View(await rentals.ToListAsync());
        }

        [Authorize]
        public async Task<IActionResult> Info(long? num)
        {
            if (num == null)
            {
                return NotFound();
            }

            var rental = await privateDC.Rentals.Include(a =>
a.Leser).Include(a => a.Worker).Include(a => a.Automodel).Include(a
=> a.Promo).Include(a => a.Rentalprice).Include(a =>
a.Insurance).Include(a => a.AutoRental.Auto).FirstOrDefault(b =>
b.Id == num);

            if (rental == null)

```

```

        {
            return NotFound();
        }

        return View(rental);
    }

    [Authorize]
    public async Task<IActionResult> Activate(long? num,
string regnumber)
    {
        if (num == null)
        {
            return NotFound();
        }

        var rental = await privateDC.Rentals.Include(a =>
a.Leser).Include(a => a.Worker).Include(a => a.Automodel).Include(a
=> a.Promo).Include(a => a.Rentalprice).Include(a =>
a.Insurance).Include(a => a.AutoRental.Auto).FirstOrDefaultAsync(b =>
b.Id == num);

        if (rental == null)
        {
            return NotFound();
        }
        var worker = await
privateDC.Workers.FirstOrDefaultAsync (a => a.Login ==
User.Identity.Name);
        rental.Id_worker = worker.Id;
        rental.AutoRental.Id_auto = privateDC.Autos.
FirstOrDefaultAsync(b => b.Number == regnumber);
        rental.State = 2;
        return View(rental);
    }

    [Authorize]
    public async Task<IActionResult> Unactivate(long? num)
    {
        if (num == null)
        {
            return NotFound();
        }
    }

```

```

        var rental = rental = await
privateDC.Rentals.Include(a => a.Lease).Include(a =>
a.Worker).Include(a => a.Automodel).Include(a => a.Promo).Include(a =>
a.Rentalprice).Include(a => a.Insurance).Include(a =>
a.AutoRental.Auto).FirstOrDefaultAsync(b => b.Id == num);
        if (rental == null)
        {
            return NotFound();
        }
        rental.State = 3;
        return View(rental);
    }

    [Authorize]
    public async Task<IActionResult> Userhistory(long? num)
    {
        var connectedData = await
privateDC.Rentals.Include(a => a.Lease).Include(a =>
a.Worker).Include(a => a.Automodel).Include(a => a.Promo).Include(a =>
a.Rentalprice).Include(a => a.Insurance).Include(a =>
a.AutoRental.Auto);
        var rentals = connectedData.Where(a => a.Id_lease
== num);

        rentals = rentals.Where(a => a.State == 4);
        rentals = rentals.OrderByDescending(a => a.Id);
        return View(await rentals.ToListAsync());
    }

    [Authorize]
    public async Task<IActionResult> Userrentals(long? num)
    {
        var connectedData = await
privateDC.Rentals.Include(a => a.Lease).Include(a =>
a.Worker).Include(a => a.Automodel).Include(a => a.Promo).Include(a =>
a.Rentalprice).Include(a => a.Insurance).Include(a =>
a.AutoRental.Auto);
        var rentals = connectedData.Where(a => a.Id_lease
== num);

        rentals = rentals.Where(a => a.State != 3 && a =>
a.State != 4);
        rentals = rentals.OrderByDescending(a => a.Id);
        return View(await rentals.ToListAsync());
    }

```

```

        [Authorize]
        public async Task<IActionResult> Ownarentals()
        {
            var connectedData = await
privateDC.Rentals.Include(a => a.Leaser).Include(a =>
a.Worker).Include(a => a.Automodel).Include(a => a.Promo).Include(a =>
a.Rentalprice).Include(a => a.Insurance).Include(a =>
a.AutoRental.Auto);
            var login = User.Identity.Name;
            IQueryable<Leaser> leasers = privateDC.Leaders;
            var leaser = leasers.FirstOrDefault(a =>
a.Login == login);
            var rentals = connectedData.Where(a => a.Id_leaser
== leaser.Id);
            rentals = rentals.Where(a => a.State != 3 && a =>
a.State != 4);
            rentals = rentals.OrderByDescending(a => a.Id);
            return View(await rentals.ToListAsync());
        }

        [Authorize]
        public async Task<IActionResult> History(long? num,
DateTime date1, DateTime, date2)
        {
            var connectedData = await
privateDC.Rentals.Include(a => a.Leaser).Include(a =>
a.Worker).Include(a => a.Automodel).Include(a => a.Promo).Include(a =>
a.Rentalprice).Include(a => a.Insurance).Include(a =>
a.AutoRental.Auto);
            if (num)
            {
                connectedData = connectedData.Where(a => a.Id ==
num);
                connectedData = connectedData.Where(a => a.State
4);
            }
            if (date1)
            {
                foreach (var rent in connectedData)
                {
                    if (rent.Date1 < date1 || rent.Date2 <
date1)
                        connectedData.Remove(rent);
                }
            }
        }
    }
}

```

```

        }
    }
    if (date2)
    {
        foreach (var rent in connectedData)
        {
            if (rent.Date1 > date2 || rent.Date2 >
date2)
                connectedData.Remove(rent);
        }
    }
    return View(await connectedData.ToListAsync());
}

[Authorize]
public async Task<IActionResult> Activerentals(long?
num, DateTime date1, DateTime, date2)
{
    var connectedData = await
privateDC.Rentals.Include(a => a.Leaser).Include(a =>
a.Worker).Include(a => a.Automodel).Include(a => a.Promo).Include(a =>
a.Rentalprice).Include(a => a.Insurance).Include(a =>
a.AutoRental.Auto);
    if (num)
    {
        connectedData = connectedData.Where(a => a.State
== 2);
    }
    if (date1)
    {
        foreach (var rent in connectedData)
        {
            if (rent.Date2 < date1)
                connectedData.Remove(rent);
        }
    }
    if (date2)
    {
        foreach (var rent in connectedData)
        {
            if (rent.Date2 > date2)
                connectedData.Remove(rent);
        }
    }
}

```

```

    }
    return View(await connectedData.ToListAsync());
}

[Authorize]
public IActionResult Create()
{
    ViewData["Insurances"] = new
SelectList(privateDC.Insurances, "Id", "Name", "Information",
"Period", "Price");
    var amodels = new SelectList(privateDC.AutoModels,
    "Id", "Modelname", "Gearbox", "Photo", "Engine", "Fuel", "Fuelrate",
    "Cityfuelrate", "Passengernumber", "Doornumber", "Autotype",
    "Luggage", "Conditioner", "Autoclass", "Prices");
    var automakers = privateDC.AutoMakers;
    var amodels2 = new List<Order>();
    foreach (var amod in amodels)
    {
        var amaker = automakers.FirstOrDefaultAsync(a =>
a.Id == amod.Id_automaker);
        amod.Modelname = amaker.Makename + " " +
amod.Modelname;
        amodels2.Add(amod);
    }
    ViewData["AutoModels"] = amodels2;
    ViewData["Services"] = new
SelectList(privateDC.Services, "Id", "Name", "Information", "Price");
    return View();
}

[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Create([Bind("Id,
Date1, Date2, Price, Paid, State, Id_insurance, Id_rentalprice")]
Rental rental, string promo)
{
    if (ModelState.IsValid)
    {
        IQueryable<Leaser> leasers = privateDC.Leasers;
        var leaser = await leasers.FirstOrDefaultAsync(a
=> a.Login == User.Identity.Name);
        rental.Id_leaser = leaser.Id;
        rental.Created = DateTime.Now;
    }
}

```

```

        var promos = privateDC.Promos;
        var promoobj = promos.FirstOrDefaultAsync(a =>
a.Code == promo);

        if (promoobj)
        {
            rental.Id_promo = promoobj.Id;
        }
        privateDC.Add(rental);
        await privateDC.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
    }
    return View(rental);
}

[Authorize]
public async Task<IActionResult> Edit(long? num)
{
    if (num == null)
    {
        return NotFound();
    }

    var rental = await privateDC.Rentals.FindAsync(num);
    if (rental == null)
    {
        return NotFound();
    }

    return View(rental);
}

[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Edit(long num,
[Bind("Id, Date1, Date2, Price, Paid, State, Id_insurance,
Id_rentalprice")] Rental rental)
{
    if (num != rental.Id)
    {
        return NotFound();
    }

    if (ModelState.IsValid)

```

```

{
    try
    {
        privateDC.Update(rental);
        await privateDC.SaveChangesAsync();
    }
    catch (DbUpdateConcurrencyException)
    {
        if (!RentalExists(rental.Id))
        {
            return NotFound();
        }
        else
        {
            throw;
        }
    }
    return RedirectToAction(nameof(Index));
}

ViewData["Id_auto"] = new
SelectList(privateDC.Autos, "Id", "Id", rental.Id_auto);
ViewData["Id_promo"] = new
SelectList(privateDC.Promos, "Id", "Id", rental.Id_promo);
ViewData["Id_rentalprice"] = new
SelectList(privateDC.Rentalprice, "Id", "Id", rental.Id_rentalprice);
ViewData["Id_insurance"] = new
SelectList(privateDC.Insurances, "Id", "Id", rental.Id_insurance);
return View(rental);
}

```

ДОДАТОК В
Слайди презентації

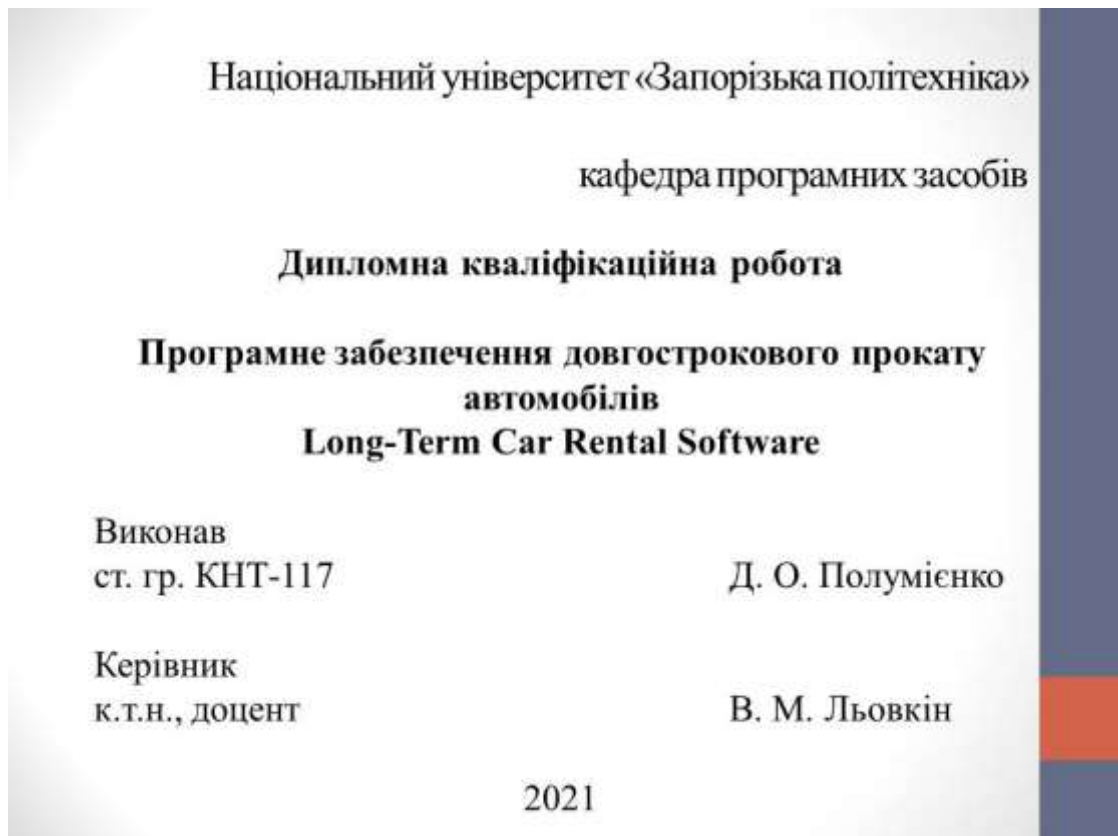


Рисунок В.1 – Слайд № 1

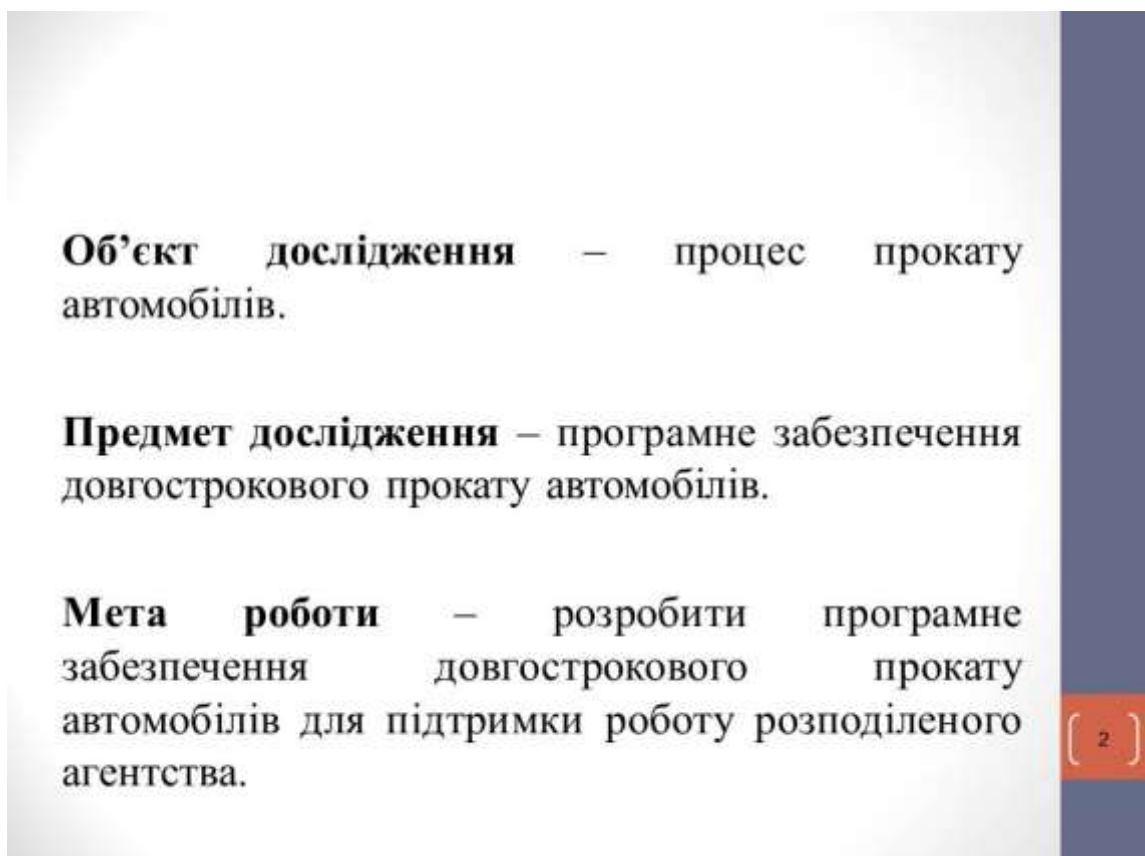


Рисунок В.2 – Слайд № 2

Завдання роботи

- визначення вимог до програмного забезпечення довгострокового прокату автомобілів;
- вибір мови і засобів проєктування і розробки;
- проєктування програмного забезпечення;
- реалізація і налагодження програмного забезпечення.

[3]

Рисунок В.3 – Слайд № 3

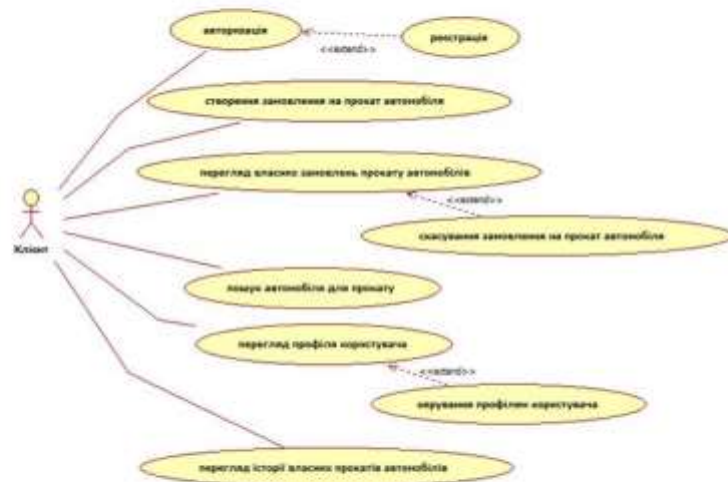
Вибір мови програмування

Критерій	PHP	C#
Тип виконання програм	Інтерпретатор	Компілятор
Велика кількість вбудованих функцій	+	++
Масштаб розроблюваних рішень	Середні і малі	Середні і великі
Фреймворки веброботки	+	+
Призначення	Веброботка	Загальне

[4]

Рисунок В.4 – Слайд № 4

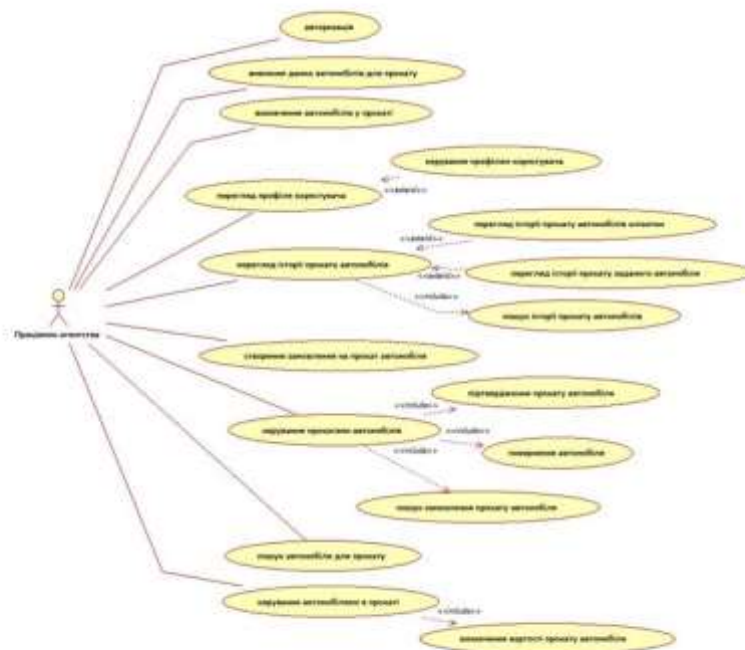
Діаграма прецедентів клієнта



{ 5 }

Рисунок В.5 – Слайд № 5

Діаграма прецедентів працівника агентства



{ 6 }

Рисунок В.6 – Слайд № 6



Рисунок В.8 – Слайд № 8



Рисунок В.8 – Слайд № 8



Рисунок В.9 – Слайд № 9

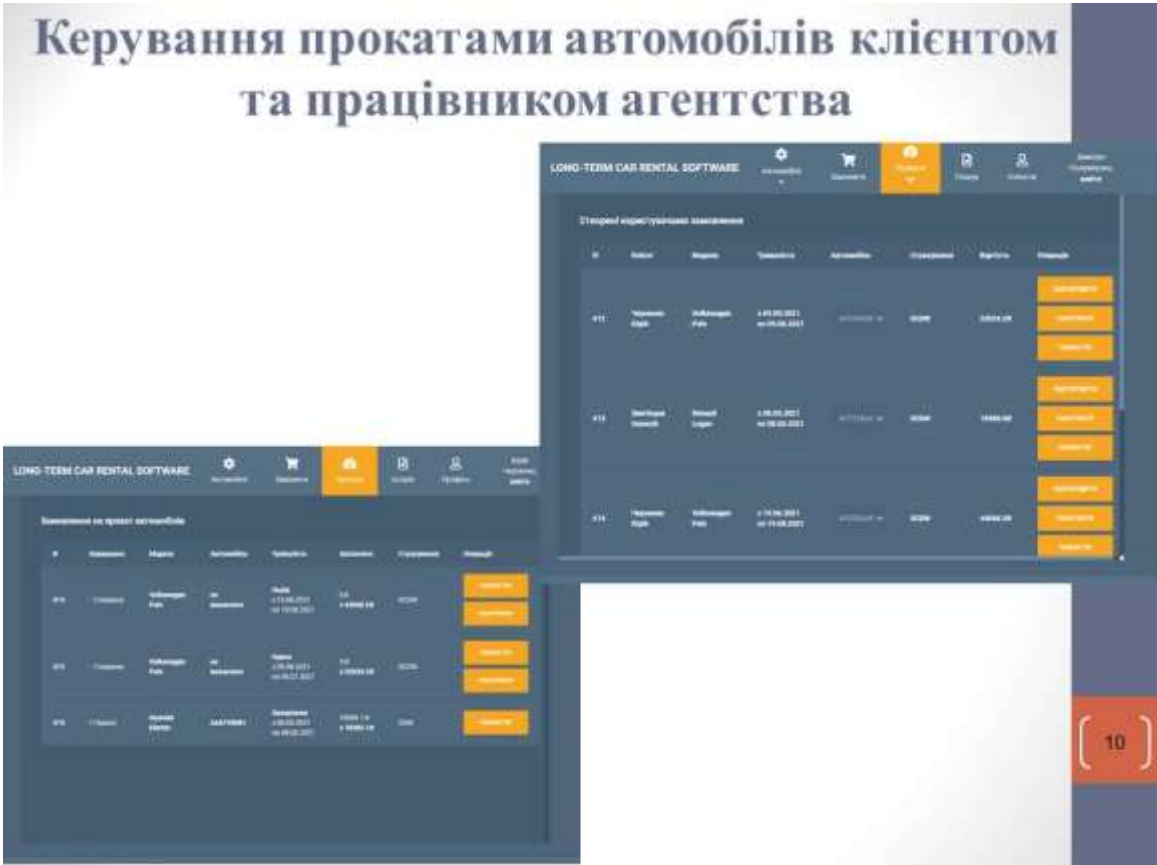


Рисунок В.10 – Слайд № 10

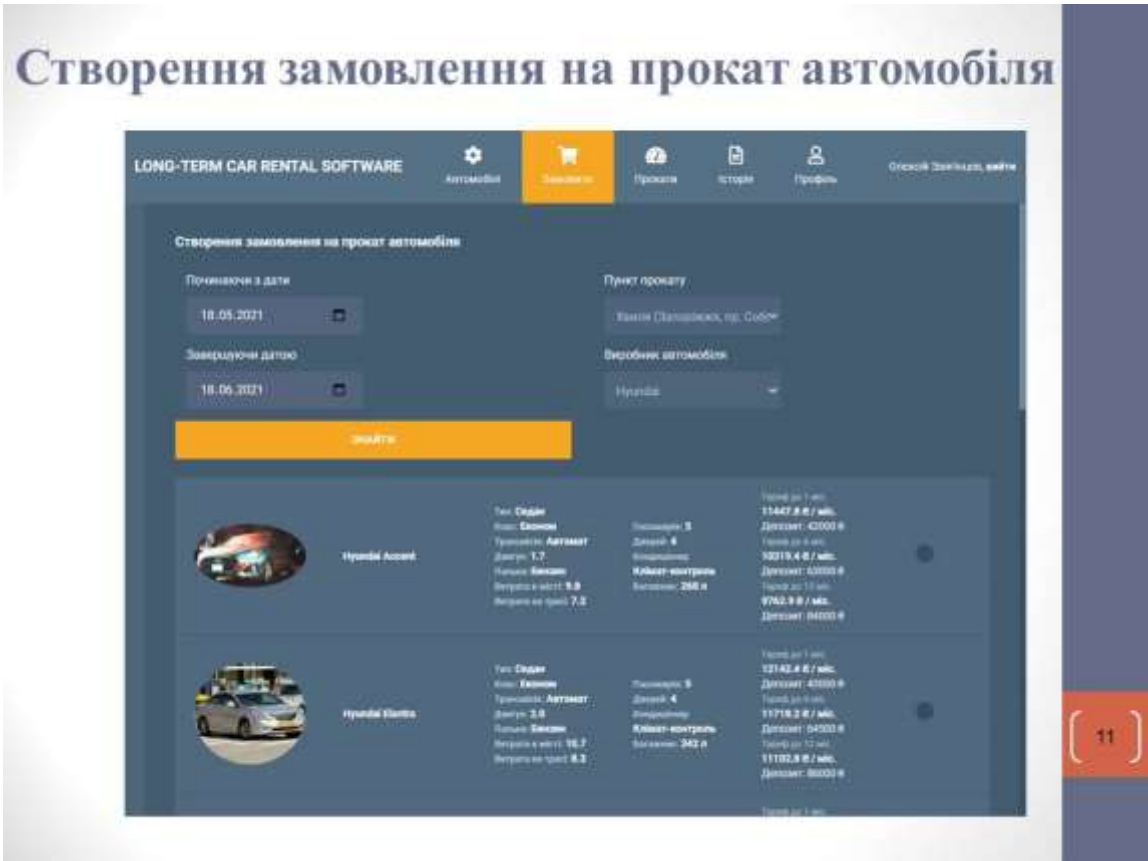


Рисунок В.11 – Слайд № 11

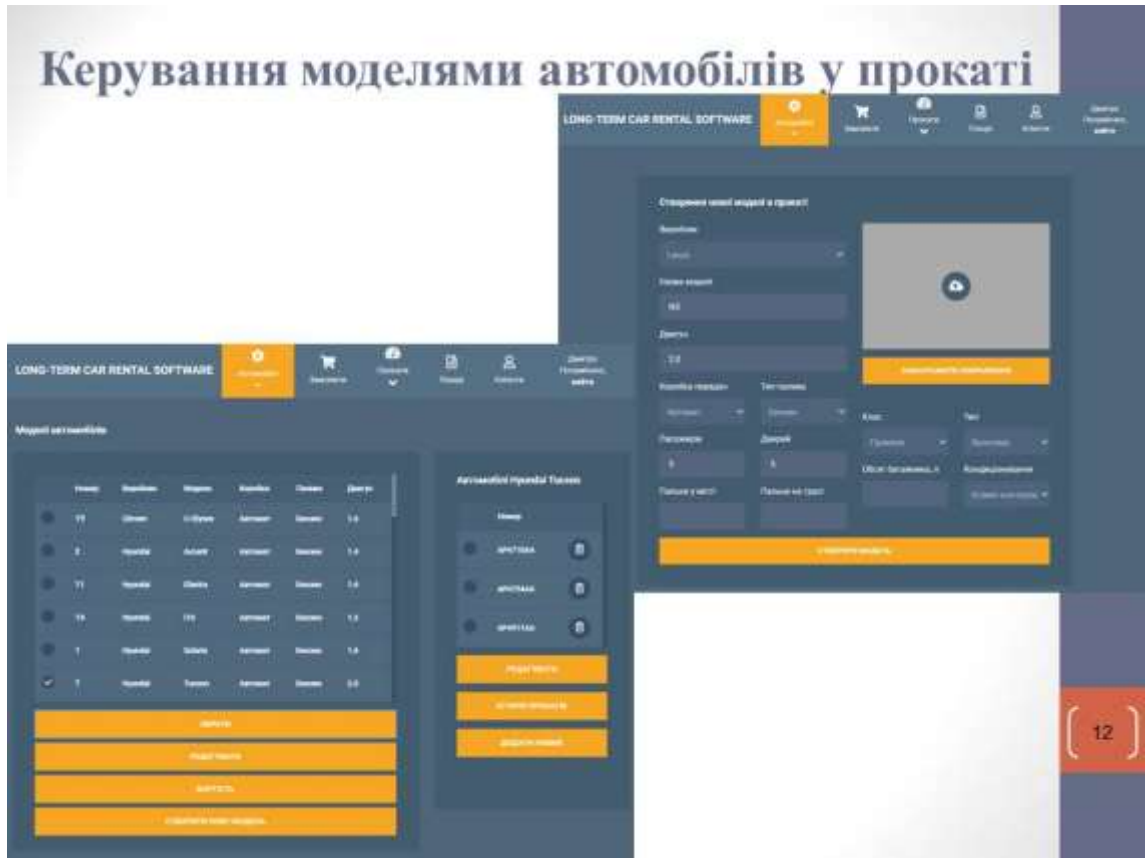


Рисунок В.12 – Слайд № 12

Висновки

Розроблено програмне забезпечення у вигляді вебзастосунку на основі застосування фреймворку розробки вебзастосунків ASP.NET для платформи .NET та мови програмування C#, яку обрано для реалізації. Роботу з даними реалізовано на основі використання системи керування базами даних Microsoft SQL Server та модуля об'єктно-реляційного відображення Entity Framework, виконано проєктування схеми бази даних. Описано процес реалізації вебзастосунку за допомогою обраного фреймворку. Структуру вебзастосунку створено на основі використання архітектурного шаблону MVC.

Описано процес виконання функціональних можливостей програми для клієнтів та працівників агентства у вигляді відповідних ланцюжків взаємодії користувачів з програмою та представлено відповідні користувацькі інтерфейси.

Усі завдання, які було визначено у дипломній кваліфікаційній роботі бакалавра, виконано у повному обсязі.