

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування інституту, факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавра

(ступінь вищої освіти)

на тему ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ПРОКАТУ КЕМПЕРІВ
CAMPER RENTAL SOFTWARE

Виконав: студент(ка) 4 курсу, групи КНТ-138
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)
Інженерія програмного забезпечення

Клінкова А.Д.

(прізвище та ініціали)

Керівник Льовкін В.М.

(прізвище та ініціали)

Рецензент Скрупський С.Ю.

(прізвище та ініціали)

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

Інститут, факультет 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
“ ” 2022 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Клінкової Анастасії Дмитрівни
(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення прокату кемперів
Camper 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 «27» квітня 2022 року № 102

2. Строк подання студентом проєкту (роботи) червень 2022 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. Розробка програми. 4. Експлуатація та тестування 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 « 4 » квітня 2022 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 тиждень	Розділ 1
3	Проектування програми	3 тиждень	Розділ 2
4	Розробка програми	4-5 тижні	Розділ 3
5	Тестування та експериментальне дослідження програми	6 тиждень	Розділ 4
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	7 тиждень	Додатки
7	Захист роботи	8 тиждень	

Студент(ка)


(підпис)

Клінкова А.Д.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис)

Льовкін В.М.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 90 с., 26 рис., 3 табл., 3 дод., 13 джерел.

КЕМПЕР, ПРОКАТ КЕМПЕРІВ, ВЕБФРЕЙМВОРК, ASP.NET CORE, ENTITY FRAMEWORK CORE.

Об'єкт роботи – процес прокату кемперів. Предмет роботи – програмне забезпечення прокату кемперів. Мета роботи – розробити програмне забезпечення прокату кемперів для підтримки роботи агентств з прокату та індивідуальних власників кемперів.

Матеріали, методи та технічні засоби: мова програмування C#, вебфреймворк ASP.NET Core, система керування базами даних Microsoft SQL Server, модуль об'єктно-реляційного перетворення Entity Framework Core.

Результати. Отримані результати охоплюють проведений аналіз предметної області, включаючи класи кемперів, основні їх сучасні складові, програмні аналоги, вибір мови програмування, визначення структури системи, структури даних, моделювання вимог до програми, реалізацію складових системи, опис логіки використання і проведеного тестування створеної системи.

Висновки. Розробка програмного забезпечення прокату кемперів надала можливість організувати платформу для підтримки роботи окремого агентства, що може бути організовано в різних країнах світу, в тому числі в декількох одночасно, з можливістю підтримки роботи одразу декількох агентств, а також створення оголошень від індивідуальних власників кемперів.

Галузь використання. Прокат кемперів засобами агентств з прокату кемперів та індивідуальними власниками кемперів.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 90 pages, 26 figures, 3 tables, 3 appendixes, 13 sources.

CAMPER, CAMPER RENTAL, WEBFRAMWORK, ASP.NET CORE, ENTITY FRAMEWORK CORE.

The object of the work is the process of renting campers. The subject of the work is software for renting campers. The purpose of the work is to develop software for renting campers to support the work of rental agencies and individual owners of campers.

Materials, methods and technical means: C# programming language, ASP.NET Core web framework, Microsoft SQL Server database management system, Entity Framework Core object-relational transformation module.

Results. The obtained results cover the analysis of the subject area, including analysis of classes of campers, its main modern components, software analogues, choice of programming language, system structurization, data structurization, modeling of program requirements, implementation of system components, description of logic of use and testing of the created system.

Conclusions. The development of camper rental software has made it possible to organize a platform to support the work of a single agency, which can be organized in different countries around the world, including in several countries at the same time, with the ability to support multiple agencies and create ads from individual camper owners.

Field of use. Camper rental by means of camper rental agencies and individual camper owners.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	7
Вступ.....	8
1 Аналіз предметної області.....	10
1.1 Проблема прокату кемперів.....	10
1.2 Огляд програмних аналогів.....	15
1.3 Висновки за розділом 1	19
2 Проєктування програмного забезпечення	20
2.1 Вибір мови програмування	20
2.2 Визначення функціональних вимог до програми.....	21
2.3 Моделювання функціональних вимог	23
2.4 Проєктування бази даних	26
2.5 Визначення структури програми.....	32
2.6 Висновки за розділом 2	33
3 Розробка програми	34
3.1 Алгоритм функціонування програми.....	34
3.2 Опис реалізованих класів	35
3.3 Висновки за розділом 3	44
4 Експлуатація та тестування програми	45
4.1 Призначення програми	45
4.2 Умови виконання програми	45
4.3 Експлуатація програми.....	45
4.4 Тестування програми.....	50
4.5 Висновки за розділом 4	52
Висновки	53
Перелік джерел посилання	54
Додаток А Технічне завдання	56
Додаток Б Текст програми	60
Додаток В Слайди презентації.....	83

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

ASP – Active Server Pages;

БД – база даних;

ПЗ – програмне забезпечення.

ВСТУП

Кемпери надзвичайно популярні серед американців: так понад 11% домогосподарств у США мають кемпери, що дає людям свободу подорожувати на 20-60% дешевше, ніж під час традиційного відпочинку, а в даний момент слід констатувати зліт запитів на кемпери серед міленіалів, так понад 15 мільйонів із них мають кемпер у США [1].

Кемпери – великий бізнес у Європі: хоча загальних даних немає, деякі оцінки промисловості вказують на те, що кемпери можуть становити до одного з трьох усіх нових транспортних засобів, що продаються щороку, крім того це величезний ринок секонд-хенду та ринок переобладнання, на якому кемпери перетворюються на всі види використання від офісів для подорожей до торговців вуличною їжею, деякі дилери навіть мають списки очікування до семи місяців для потенційних покупців [2].

Дані Outdoorsy показують, що кількість замовлень на оренду кемперів міленіалами зростає на 70 % з 2019 по 2020 рік, ця ж демографічна група також склала 35 % тих, хто вперше орендував кемпер на Outdoorsy, що склало 90 % замовлень платформи протягом року, до того ж згідно з опитуванням, проведеним Асоціацією промисловості автофургонів, це перший раз, коли міленіали більше прагнуть придбати кемпер, ніж будь-який інший віковий сегмент (з опитаних 30% міленіалів повідомили, що вони, ймовірно, придбають кемпер у наступному році, тоді як лише 26% представників покоління X і 8% представників покоління бумерів сказали те саме), до того ж цей інтерес міленіалів до покупки кемперів знайшов відображення в даних про продажі від Thor Industries: виробник кемперів вказав, що у 2020 році на 21,1 8% більше міленіалів придбало туристичний причіп Thor в порівнянні з 2019 роком [3].

Індустрія кемперів приносить економіці США понад 114 мільярдів доларів, задіюючи в ній понад 600 000 людей і вносить 6 мільярдів доларів у податках, тому слід зазначити, що це важлива частина економіки США [1].

Усі ці фактори вказують на актуальність діяльності в галузі прокату кемперів. У даному випадку можна констатувати простір для відкриття прокатів кемперів, що вимагають підтримки за допомогою відповідного програмного забезпечення (ПЗ). Дане ПЗ має реалізовуватись у роботі.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Проблема прокату кемперів

Стосовно прокату кемперів важливим є визначення класів кемперів, що є загально прийнятим і дозволяє в залежності від класу розуміти, з якими саме параметрами кемперів буде відбуватися робота.

Виділяють наступні класи кемперів:

- клас А (найбільші кемпери), що включає кемпери розміру від 26 до 45 футів вагою від 13 000 до 30 000 фунтів: найбільші кемпери схожі на гігантські будинки на колесах, вони можуть бути відносно простими або мати всілякі додаткові можливості, включно з двоспальними ліжками, пральними та сушильними машинами, а також мобільними гаражами, в яких можна помістити спортивний автомобіль, також дані кемпери можуть бути дуже дорогими, мати низьку паливну ефективність і обмеження в тому, куди вони можуть подорожувати через їх розмір і потреби в підключенні;

- клас С (варіант середнього розміру), що включає кемпери розміру від 20 до 30 футів вагою від 10 000 до 13 000 фунтів: не такий великий, як клас А, але не такий маленький, як клас В, спальні приміщення в класі С часто знаходяться над кабіною, можна знайти висувні виходи, подібні до тих, що поширені в більшому класі А, зазвичай є окрема ванна кімната, тоді як кемпер класу В може і не мати її, при цьому кемпер класу С зазвичай недостатньо великий, щоб буксирувати інший автомобіль;

- клас В, що включає кемпери розміру від 17 до 23 футів вагою від 4000 до 9000 фунтів: варіант середнього розміру, який набагато менший, ніж великі кемпери класу А, ці невеликі кемпери можуть мати базові можливості, включаючи, наприклад, ліжко і місце для зберігання, або мати додаткові можливості, як ванні кімнати, обідні зони, деякі такі кемпери представлені у варіанті повного приводу, мають хорошу економію палива, можуть поміститися на багатьох паркувальних місцях, ними легко керувати;

– туристичний трейлер, що включає кемпери розміру від 10 до 40 футів і вагою від 1100 до 12000 фунтів: фактично це бамперні причепа, які під'єднуються до стандартного зчіпного пристрою для причепа, і вони можуть варіюватися від легких до досить важких, це означає, що найменші туристичні причепа можна тягнути невеликим позашляховиком, а для інших може знадобитися велика вантажівка, причепа для подорожей підходять для туристів, які хочуть легше відкріплювати та зберігати кемпер, коли він не використовується, а велика кількість варіантів означає, що кожен знайде щось для себе;

– переносний (pop-up) трейлер (компактний і простий у зберіганні), що включає кемпери розміром від 8 футів до 16 футів вагою від 700 фунтів до 4000 фунтів: ще називається складаним причепом, це легкий кемпер, який складається для зручності зберігання та буксирування, доречно розглядати як альтернативу намету, оскільки зазвичай такий клас представлений вітальною в центрі та двома спальними зонами з обох боків, окрім того багато з них також мають помірні ціни, при цьому складні причепа не так добре ізолювані для погодних умов;

– п'яте колесо, що включає кемпери розміром від 22 до 40 футів вагою від 7000 до 20000 фунтів: це найбільші причепа, які можна буксирувати позаду звичайного автомобіля, їх називають «п'ятим колесом», оскільки передня частина причепа простягається над кузовом, а кузов містить зчіпний механізм (U-подібний з'єднувальний компонент), який полегшує буксирування, оскільки в результаті такий кемпер легше повертати, такі кемпери мають ванні кімнати і навіть декілька спалень, але для них також потрібен пікап зі спеціально встановленим зчіпним механізмом;

– вантажні кемпери розміром на декілька футів довший за автомобіль, вагою від 1000 до 5000 фунтів: підходять для подорожей без мережі протягом тижнів, місяців або навіть років, це не кемпер, призначений для великої родини, він найкраще підходить для однієї, двох або максимум трьох осіб, можливості повного приводу та потужна підвіска означають, що ці

кемпери можуть їхати куди завгодно, але мають обмежене місце для зберігання та лише найосновніші зручності, хоча є і кемпери для вантажівок з більшими зручностями, призначеними для регулярного перебування в кемпінгу, вони часто включають верхню спальну кабіну, варіанти для кемперів для вантажівок включають довгі та короткі ліжка, жорсткі ліжка та висувні [4]-[5].

На даний момент сучасні кемпери мають багато внутрішнього обладнання і складових, які потрібно враховувати під час їх вибору, а відповідно інформація про них має бути доступна в застосунку, включаючи:

- кількість і тип ліжок: одна з перших речей, на що люди звертають увагу, – це спальне місце в кемпері, деякі люди вважають за краще мати ліжко, яке завжди залишається на місці, тому їм не потрібно заправляти його щовечора перед сном, тоді як інші люди не проти прибирати ліжко щоранку та ввечері, деякі люди навіть вважають за краще мати окрему спальню в кемпері, у багатьох кемперах також є ліжка, які чудово підходять для дітей або підлітків, але не завжди підходять дорослим, деякі з ліжок можуть бути перетворені на диван, який перетворює спальну зону в житловий простір;

- місця для сидіння є дуже важливими: важливо, скільки людей подорожуватиме, чи це буде компанія, яке любить розважатися, більшість кемперів мають якусь обідню зону, тому потрібно переконатися, що там зручно буде поміститися всій родині, перевірити, як складається стіл, щоб отримати додатковий простір, коли стіл не використовується;

- кухня: більшість кухонь будуть мати якийсь кухонний прилад, раковину та холодильник, розміри кухні та побутова техніка зазвичай збільшуються в розмірі разом із розміром кемпера, тож у більшості невеликих кемперів будуть крихітні кухонні зони та побутова техніка, а у великих кемперах – набагато розкішніші та великих розмірів, більшість сучасних кемперів також мають мікрохвильові печі, зазвичай плита працює на газі, як і духовка, холодильники часто працюють як на газі, так і на електриці, а в сучасних кемперах система автоматично перемикається між

режимами в міру необхідності, мікрохвильові печі часто вимагають підключення до електрики, щоб використовувати їх або запустити генератор, більш висококласні кемпери можуть мати посудомийні машини, проте вони часто займають дорогоцінний простір у шафі, невеликі за розміром і потребують води та електрики для роботи, окрім того кухня кемпера може мати або не мати місце для зберігання (важливо мати чашки, кухлі, столові прилади, обробні дошки, сірники, затискачі, контейнери для посуду, каструлі, сковорідки, кавники, паперові рушники);

– дуже маленькі кемпери або причепають можуть не мати ванну кімнату, але більшість кемперів буде її мати: у невеликих кемперах це може бути просто волога ванна – це, по суті, крихітне закрите приміщення, фактично туалет з можливістю одночасно приймати душ, перевага вологих ванн полягає в тому, що вони займають дуже мало місця, великі кемпери мають гарні ванні кімнати (у них буде унітаз, раковина, а також стоячий душ або невелика ванна з душем), тому потрібно подумати, якому типу ванної кімнати віддати перевагу;

– розважальні системи: багато кемперів мають стереосистеми та деякі типи підключення до телевізорів, старі кемпери можуть мати більші телевізори, тоді як більшість сучасних кемперів мають компактні телевізори з плоским екраном, окрім того можуть бути доступні зовнішні розважальні системи, важливим також є підключення до телевізора (багато кемперів мають антену, яку можна переміщати, щоб отримати сигнал, інші можуть мати портативну супутникову систему);

– електрика та підключення є досить стандартними, хоча вони можуть відрізнятися на різних кемперах: хоча зазвичай можна підключитися до електрики, якщо кемпер перебуває на спеціалізованій парковці, проте якщо мережа не доступна, то знадобиться інший тип живлення, так зазвичай є якийсь тип акумулятора, який буде жити кемпер, а підзарядку можна виконувати різними способами, зокрема зараз багато людей встановлюють сонячні батареї на свій кемпер, щоб можна було легко заряджати батареї,

інші мають різні види систем перезарядки (наприклад, під час їзди на кемпері він автоматично заряджається від сонячних батарей, окрім того кемпер може працювати на бензині від основного бензобака або запустити генератор лише одним натисканням кнопки зсередини автобуса);

– резервуари для води та насоси: проточна вода є досить стандартною функцією більшості кемперів, у більшості кемперів є резервуар для прісної води, який можна наповнити, тоді просто вмикається водяний насос, щоб отримати свіжу воду з раковини, є можливість підключитися безпосередньо до води на вулиці за допомогою шланга, наприклад, коли людина перебуває на стоянці кемперів, менші фургони або кемпери можуть мати невеликі резервуари для прісної води або використовувати інший вид системи насосів;

– резервуари для відходів: зазвичай в кемпері є чорний і сірий баки, для простої стічної води з раковини або душу більшість фургонів мають сірий бак, щоб утримувати їх, для туалетних відходів є чорний бак, залежно від типу та розміру кемпера принцип роботи цих банків може дещо відрізнятися, деякі менші кемпери мають касети, які зберігають відходи, що потрібно спорожнити вручну, тоді як більші установки мають великі резервуари та шланги, які можна під'єднати до каналізації;

– нагрівач гарячої води у кемпері також є стандартною складовою: деякі з них просто працюють на пропані, а інші дозволяють використовувати як пропан, так і електроенергію, це важлива функція, оскільки при цьому не потрібно постійно використовувати свій дорогоцінний пропан для гарячої води, потрібно почекати деякий час, поки гаряча вода нагріється;

– кондиціонер стає більш стандартною функцією: кондиціонер окрім зручності пов'язаний також з додатковим споживанням електроенергії, тому багато кемперів можуть працювати з блоком кондиціонування повітря, лише якщо вони підключені до електрики або якщо вони вмикають генератор [6]-[7].

1.2 Огляд програмних аналогів

Огляд рішень для прокату кемперів на ринку дозволив виділити основні наявні рішення серед всього проаналізованого набору.

Одним з найвідоміших рішень є вебсайт Outdoorsy (рис. 1.1) [8].

The screenshot displays the Outdoorsy website interface. At the top, there is a navigation bar with the Outdoorsy logo, a search bar with a magnifying glass icon, and fields for 'Location', 'Dates', 'Guests', and a 'Search' button. Below the navigation bar, there are four images showing the interior of a camper van. The main listing is for 'Montana Lodge (2 personnes)' by 'Nomadism'. It features a star rating of 5 (3 reviews), location 'Montrouge', and details: '2 seatbelts • Sleeps 2 • 5.93 m • Year 2021'. The price is shown as '\$248 \$292 /night (15% off)' for the dates 'May 23, 2022' to 'May 25, 2022'. The total price is '\$743.58'. There is a 'Refundable security deposit' of '\$2,700.00'. A green 'Instant book' button is visible. The listing also includes a section for 'Meet your host, Nomadism' with a description and a 'Read more' link. A green button 'Ask Nomadism a question' is also present. The bottom of the listing shows a section for 'Enhanced clean' and a section for the host 'Alexandre M.' with a 5-star rating and a review from February 2022.

Рисунок 1.1 – Вебсайт Outdoorsy

Outdoorsy є платформою для взаємодії власників кемперів з орендарями. Надається можливість пошуку кемперів за великою кількістю параметрів, включаючи конкретні зручності, потрібні користувачу. Окрім того користувач має багато інформації за конкретним оголошенням, включаючи зокрема і перелік відгуків конкретних орендарів даного кемпера.

Вебсайт RVshare (рис. 1.2) [9] створений за подібною логікою, що і попереднє рішення, для якого він є конкурентом. Дане рішення включає пропозиції тільки всередині США. Проте серед негативних відмінностей слід відзначити відсутність можливості перегляду відгуків за власником. Тобто відгуки можна переглянути за оголошенням, але отримати більш узагальнюючу картину за власником не можна. Якщо якесь оголошення тільки створено, то отримати ширшу інформацію за власником не вдасться.

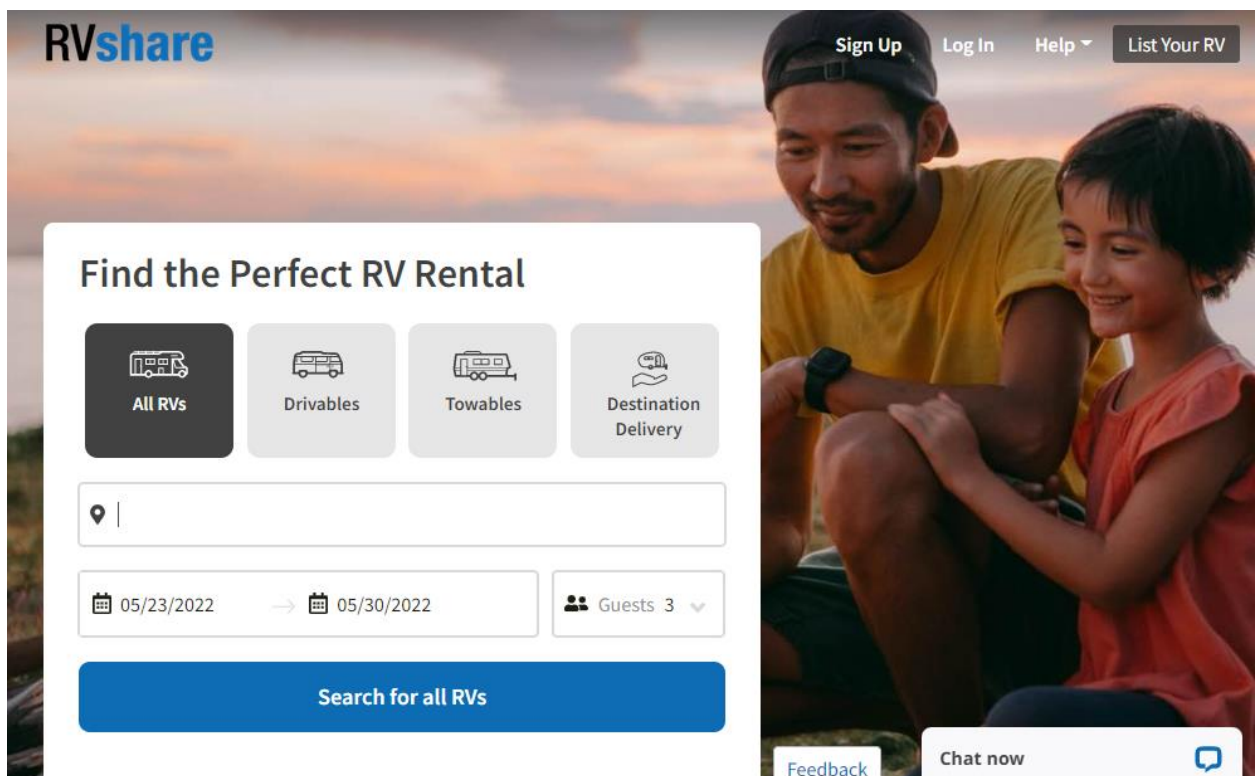
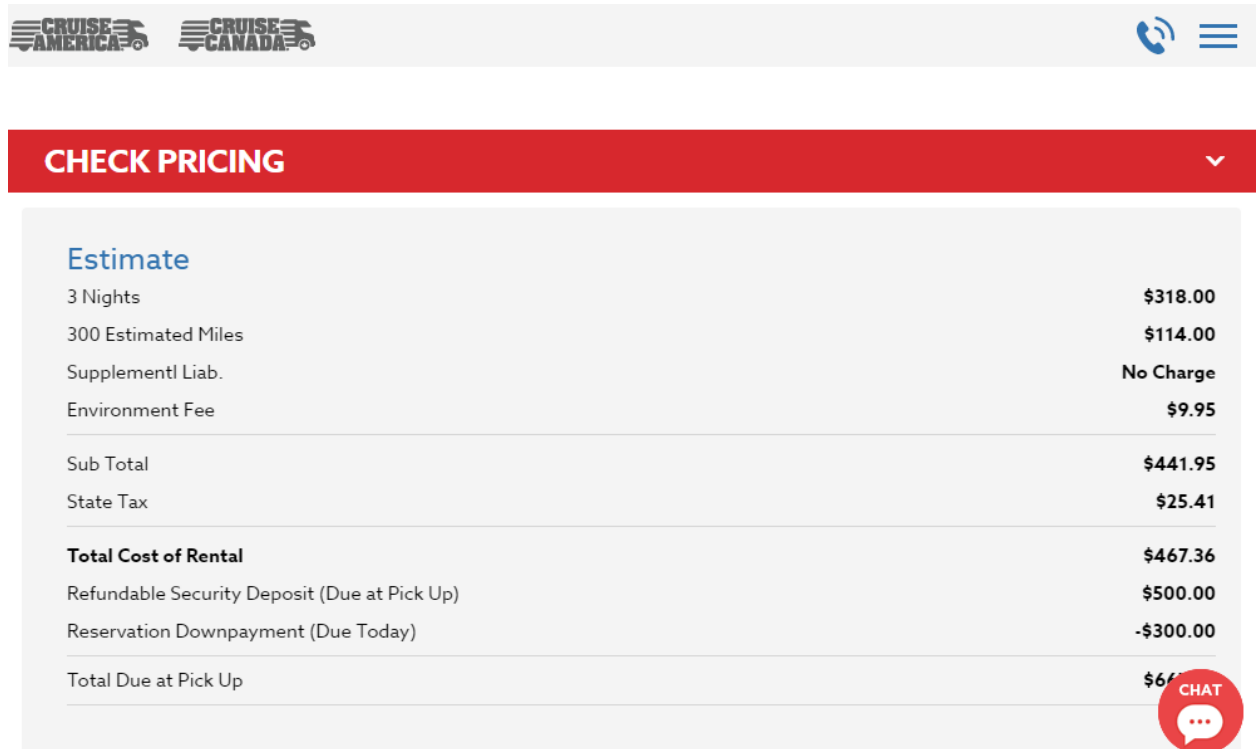


Рисунок 1.2 – Вебсайт RVshare

Значно відрізняється від цих аналогів вебсайт Cruiseamerica (рис. 1.3) [10]. Його відмінністю є те, що це вебсайт конкретного агентства з прокату кемперів. Тому діяльність відбувається тільки в США, що

обмеженням. Окрім того можна переглядати дані кемперів, можна шукати їх на потрібні дати, бронювати, але при цьому не можна отримати інформацію про відгуки орендарів, себто результат зворотної взаємодії не відзначений.



The screenshot shows the Cruise America website header with logos for 'CRUISE AMERICA' and 'CRUISE CANADA', a phone icon, and a menu icon. Below the header is a red 'CHECK PRICING' button. The main content area displays a pricing estimate table with the following items and costs:

Estimate	
3 Nights	\$318.00
300 Estimated Miles	\$114.00
Supplementl Liab.	No Charge
Environment Fee	\$9.95
Sub Total	\$441.95
State Tax	\$25.41
Total Cost of Rental	\$467.36
Refundable Security Deposit (Due at Pick Up)	\$500.00
Reservation Downpayment (Due Today)	-\$300.00
Total Due at Pick Up	\$667.36

A red 'CHAT' button is visible in the bottom right corner of the pricing section.

Рисунок 1.3 – Вебсайт Cruiseamerica

Результати порівняння програмних аналогів представлені в табл. 1.1.

Існують також сервіси для прокату в Україні, але вони не мають системи бронювання і їхні вебсайти несуть тільки загальну інформаційну функцію. Можливість бронювання прокату кемперів в Україні виділяє виконувану розробку. Також розробка має дозволяти працювати і власникам, і агентствам. Наявність засобів для індивідуальних власників звичайно передбачає, що і агентство може працювати з програмою, проте важлива наявність інструментів для введення подібних кемперів однієї й тієї самої моделі, щоб кожного разу не створювати окреме оголошення заново.

У аналогів зазвичай вартість прокату може бути встановлена в обмеженому інтервалі, тобто сервіс накладає певні обмеження, розробка таких обмежень не передбачає, тобто ціноутворення буде ринковим

повністю. У той же час аналоги дозволяють встановити особливу ціну доби на день, тиждень або місяць. У розробці мають бути визначені особливі інтервали (на 2 доби, на 5 діб, на тиждень – як власник вважає за потрібне).

Таблиця 1.1 – Порівняння аналогів з виконуваною розробкою

Критерій	Outdoorsy	RVshare	Cruiseamerica	Розробка
Країна дії	США, Канада, Мексика	США	США	Без обмеження на країни
Поширення	Індивідуаль- ні власники	Індивідуаль- ні власники	Окреме агентство	Агентства, індивідуаль- ні власники
Встановлення інтервалу цін для класів кемперів	Так	Так	Так	Ні
Ціноутворення	День, тиждень, місяць	День, тиждень, місяць	День, тиждень, місяць	Гнучке
Розширений пошук	Так	Так	Ні	Так
Відгуки за всіма оголошеннями власника	Так	Ні	Ні	Так
Відгуки про орендаря	Так	Так	Ні	Так
Відгуки про оголошення	Так	Так	Ні	Так

Вебсайт Cruiseamerica не дозволяє під час пошуку визначити конкретні потрібні зручності. Також він не має системи відгуків. Усе це враховано у виконуваний розробці, що робить її достатньо гнучкою у використанні.

1.3 Висновки за розділом 1

У розділі 1 виконано аналіз предметної області прокату кемперів, включаючи визначення класів кемперів, основні їх сучасні складові, а також окремо огляд програмних аналогів, за якими було визначено основні характеристики, які в результаті було використано під час виконання їх порівняння і виділення особливостей програми, що розробляється.

Мета роботи має бути досягнута через виконання наступних завдань роботи:

- аналіз предметної області прокату кемперів;
- визначення вимог до ПЗ прокату кемперів;
- проєктування ПЗ прокату кемперів;
- реалізація і налагодження ПЗ прокату кемперів.

2 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Вибір мови програмування

За результатами проведеного аналізу зрозуміло, що подібні застосунки найчастіше використовуються у вигляді вебзастосунків або окремо мобільних застосунків. Окрім того потенційно може використовуватися десктопне рішення для роботи всередині компанії. Тому було вирішено виконувати розробку програми як вебзастосунку, але враховувати потенційне розширення під час вибору мови програмування, який здійснювався між C# [11]-[12] та PHP як основними мовами веброзробки.

Обидві мови є відомими засобами веброзробки. У той же час мова PHP відома тільки в цій галузі, а розробка за її допомогою десктопних рішень якщо і можливо, то дуже ускладнена і потребує додаткового часу в обмежені строки, що є негативною стороною. У свою чергу C# у складі платформи .NET дозволяє розробляти як вебзастосунки на основі використання технології Active Server Pages (ASP) і фреймворку ASP.NET [13], так і десктопні застосунки, і навіть мобільні застосунки. Таким чином, у подальшому потенційно можна буде розширити доступні рішення за виконуваною розробкою, не змінюючи мови програмування і відповідно маючи менше проблем під час розробки.

Ще одним аргументом на користь C# є те, що застосунки, які створюються в результаті, мають більшу швидкість роботи порівняно з застосунками, створеними мовою програмування PHP.

Для мови C# доступний вебфреймворк ASP.NET, що є одним з найвідоміших рішень для масштабних застосунків. Для мови PHP є цілий набір вебфреймворків, проте не можна сказати, що якийсь із них переважає ASP.NET. Тому за даним критерієм можна за результатами порівняння визначити паритет.

У підсумку мова С# і вебфреймворки ASP.NET були визначені для виконання розробки програми.

Результати порівняння подані в табл. 2.1.

Таблиця 2.1 – Вибір мови програмування

Критерій	С#	PHP
Придатність до розробки вебзастосунків	Так	Так
Придатність до потенційної розробки десктопних, мобільних застосунків	Так	Ні
Швидкість роботи застосунків	Більше	Менше
Вебфреймворк	ASP.NET	Laravel, Zend, Yii, Zend

2.2 Визначення функціональних вимог до програми

Результатом проведення аналізу предметної області стало на даному етапі визначення функціональних вимог до програми, що включають:

- пошук варіантів прокату кемперів;
- перегляд власних запитів прокату кемпера;
- перегляд власних прокатів кемперів;
- оцінювання прокату;
- пошук варіантів транспортування кемпера;
- визначення витрат на прокат кемпера;
- перегляд запитів на прокат кемпера;
- підтвердження або скасування запита на прокат кемпера;

– перегляд власних вільних кемперів на задану дату або за заданою станцією;

- перегляд прокатів власником за календарем;
- визначення пропозицій з транспортування кемпера;
- скасування пропозиції з транспортування кемпера;
- перегляд графіка прокату заданого власного кемпера;
- перегляд популярності власних кемперів;
- створення відгуку про орендаря;
- перегляд популярних станцій прокату;
- перегляд популярних моделей кемперів;
- перегляд неоцінених власних прокатів;
- реєстрація орендарів, власників;
- авторизація;
- перегляд профілів орендарів і власників;
- видача кемпера в прокат;
- повернення кемпера з прокату;
- перенесення фактичного знаходження кемпера в задану станцію;
- додавання і редагування виробників кемперів;
- додавання і редагування моделей кемперів;
- додавання і редагування кемперів власником;
- додавання і редагування оголошення про прокат кемпера;
- визначення недоступних для прокату кемпера дат;
- пошук власником серед власних кемперів;
- створення запиту на прокат кемпера
- скасування запиту на прокат кемпера;
- створення запиту на транспортування кемпера;
- перегляд найкращих кемперів;
- архівувати власне оголошення.

2.3 Моделювання функціональних вимог

Під час моделювання функціональних вимог було використано вимоги, визначені в розділі 1. Моделювання здійснювалось для орендаря і власника, а також адміністратора.

Під власником розглядалися користувачі, які можуть представляти певну фірму, що надає на прокат кемпери, а також звичайні власники кемперів, які хочуть надати власний кемпер у прокат.

Власник під час роботи з програмою може використовувати її на основі наступних сценаріїв:

- керування запитами на прокат кемпера, що охоплює під час реалізації перегляд запитів на прокат, підтвердження запита на прокат кемпера, а в якості альтернативного сценарія – скасування запита на прокат кемпера;
- визначення пропозицій з транспортування кемпера;
- керування прокатом кемпера, що охоплює під час реалізації сценарії перегляду графіка прокату, видачі кемпера в прокат, повернення кемпера з прокату;
- керування власними кемперами, що охоплює під час реалізації сценарії перегляду власних кемперів, перегляду власних вільних кемперів і додавання кемпера, а в якості альтернативного сценарія – редагування кемпера;
- керування власними оголошеннями;
- перегляд популярності власних оголошень;
- перегляд популярних станцій прокату;
- перегляд популярних моделей кемперів;
- перегляд найкращих кемперів;
- створення відгуку про орендаря;
- перегляд профіля орендарів;
- авторизація.

Відповідну діаграму Use Case представлено на рис. 2.1.



Рисунок 2.1 – Діаграма use case для власника

Сценарій керування власними оголошеннями охоплює під час реалізації сценарії перегляду власних оголошень, додавання оголошень, пошуку оголошень, визначення цінових пропозицій прокату кемпера, а в

якості альтернативних сценаріїв – редагування оголошення про прокат кемпера, визначення недоступних для прокату кемпера дат.

Орендар під час роботи з програмою може використовувати її на основі наступних сценаріїв (рис. 2.2):

- пошук варіантів прокату кемперів;
- створення запиту на прокат кемпера;
- перегляд власних прокатів кемперів;
- оцінювання прокату;
- перегляд профілю орендаря;
- перегляд популярних станцій прокату;
- перегляд популярних моделей кемперів;
- авторизація.

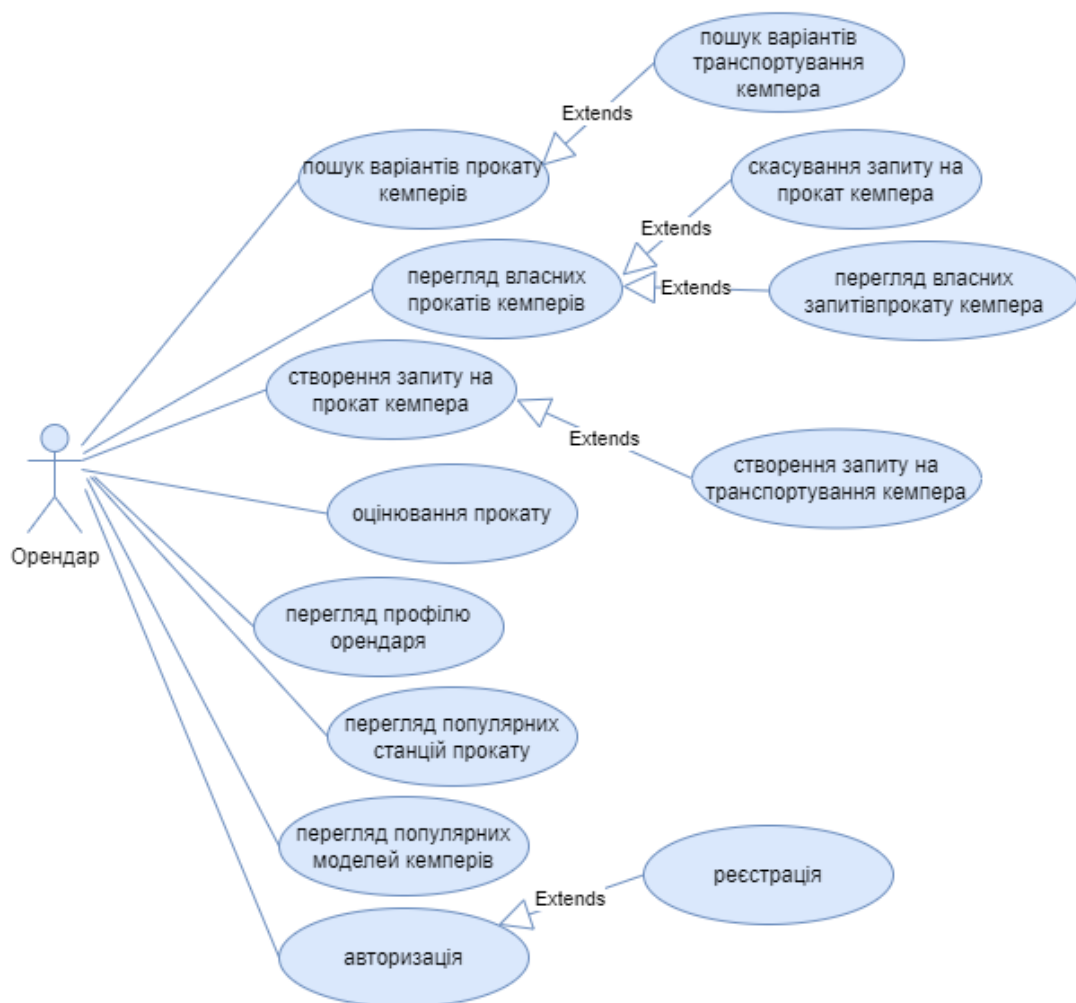


Рисунок 2.2 – Діаграма use case для орендаря

Альтернативними сценаріями є:

- скасування запиту на прокат кемпера, який реалізується як розширення сценарію перегляду власних прокатів кемперів у випадку, якщо орендар помилково створив запит на прокат кемпера;
- пошук варіантів транспортування кемпера, який є розширенням сценарію пошуку варіантів прокату кемперів і реалізується тоді, коли орендар готовий обмежити не тільки станцію початку прокату, але і завершення тією станцією, яка буде визначена пропозицією;
- створення запиту на транспортування кемпера, який реалізується як альтернатива сценарію створення запиту на прокат кемпера;
- перегляд власних запитів прокату кемпера, який реалізується як розширення сценарію перегляду власних прокатів кемперів у випадку, якщо запит не було підтверджено власником і не стало прокатом;
- реєстрація, який є альтернативним сценарієм під час реалізації авторизації і реалізується у тому випадку, якщо орендар не має облікового запису.

2.4 Проєктування бази даних

Під час проєктування бази даних (БД) було на основі результатів вивчення предметної області та виділених функціональних вимог визначено основні сутності предметної області.

Звичайно, що основною сутністю предметної області є оголошення про прокат кемперу. Саме з ним в основному взаємодіє користувач. Проте варто відзначити, що дана сутність потребує певної структуризації в межах БД. Це пов'язано з тим, що кожне оголошення пов'язано з певним конкретним кемпером, проте не всі кемпери повністю між собою відрізняються. Є певні моделі кемпів, певні виробники кемперів. Виходячи з даної логіки, було вирішено виділити сутність виробника кемпера (CamperManufacturer), сутність моделі кемпера (CamperModel). До моделі

кемпера належить основна інформація про модель кемпера, адже моделлю кемпер повністю не характеризується, далі може відрізнятися комплектація, окрім того комплектація в подальшому може змінюватися. Тому до моделі віднесено тільки дані про трансмісію, кількість сидячих і спальних місць, які визначаються технічними даними кемперами і не можуть в подальшому змінюватися, обсяг свіжої води, технічної та брудної води, тип палива, обсяг палива, а також клас кемпера, його довжина та ширина. Усі ці параметри є базовими і їх значення в подальшому не можуть змінюватися.

Звичайно, що модель повністю не описує конкретний кемпер, а тому окремо винесено сутність Camper. Дана сутність включає зручності кемпера, що визначають наявність душа, сонячних батарей, системи мультимедіа тощо. Вся ця комплектація по суті в подальшому в конкретному кемпері може змінюватися, а може залишатися такою ж як початково в моделі, тому без виділення цієї сутності обійтися не можна.

Окремо було виділено сутність, що відповідає самому оголошенню. Вона має назву Advertisement. Оголошення виділено окремо від кемпера з наступною логікою. Звичайно, що в сутності кемпера може зберігатися інформація про конкретний кемпер і тоді б відношення між даною сутністю і кемпером було один до одного. Це було б вірно зокрема для ситуацій, коли сам власник кемпера здає його в прокат. Проте не завжди це саме так. І частіше кемпери надаються в прокат компаніями. Навіть якщо співвідношення і протилежне, тобто на даному вебсайті в результаті частіше кемпери будуть здаватися власниками, то наявність або навіть потенційна наявність компаній призводить до того, що така компанія може мати цілий набір однакових кемперів. Кожен з них буде мати окремий реєстраційний номер, можливо вони будуть відрізнятися кольором салону, але при цьому всі зручності будуть однаковими. Для того щоб в такій ситуації не вносити дані таких кемперів повторно та через деякі вище описані додаткові причини, було вирішено виділити окрему сутність Advertisement.

У той же час оголошення є базовою сутністю для іншої сутності. Це сутність, яка визначає цінові пропозиції щодо прокату кемпера. Для кожного кемпера такі пропозиції можуть встановлюватися окремо. При чому реалізується цей зв'язок саме з оголошенням, а не з кемпером, оскільки сема фізичний кемпер може мати різні пропозиції, не дивлячись на те, що його характеристики повністю відповідають іншому кемперу, який тим не менше представлений іншим оголошенням. Різні цінові пропозиції можуть бути такими через те, що кемпер в певний період часу використовується в іншій країні, де умови можуть відрізнятися, або через інші причини. Дана сутність має назву RentalRate.

Звичайно, що ще однією важливою сутністю предметної області є сутність прокату в БД. Вона має назву CamperRental і визначає відповідно характеристики конкретного запиту на прокат, а в подальшому – прокату кемпера.

Але не завжди прокат здійснюється саме за звичайною схемою, іноді прокат одночасно є виконанням перевезення кемпера з однієї станції на іншу. Це потрібно тоді, коли орендарі залишають кемпер у непопулярному місці, тоді виникає необхідність перемістити кемпер у інше місце. У такому випадку прокат може здійснюватися безкоштовно або за зниженою вартістю. Дану сутність названо Proposition. По суті така пропозиція є особливою ціновою пропозицією, адже в такому випадку також висувається пропозиція щодо прокату кемперу, але прокат тоді здійснюється не між бажаними орендарем станціями, а за фіксованими власником станціями. Саме тому виникла необхідність виділити таку пропозицію в окрему сутність.

Окрім того ще однією сутністю є сутність станції Station. Це станція, на якій розташовується кемпер і де його можна забрати на прокат або повернути з прокату. Окрім того кожна станція знаходиться в певному місці або поруч з ним, тому для забезпечення ідентифікації виділено окрему сутність City.

Також було виділено сутність `GuestProfile`, що визначає гостьовий профіль орендаря. Дана сутність не містить характеристики, які належать до базових характеристик користувача в технології ASP.NET, адже реалізовано дану таблицю буде наслідуванням базової сутності користувача. У свою чергу на основі такої ж логіки окремо виділено сутність профілю власника `OwnerProfile`. Це може бути або прокатне агентство, або безпосередньо власник, якщо він самостійно надає власний кемпер у прокат.

Окремо необхідно розглянути зв'язки між сутностями БД.

Зв'язок між сутністю `CamperManufacturer` і сутністю `CamperModel` один до багатьох: один виробник кемперів може мати від нуля (тимчасово) до великої кількості моделей, а кожна модель кемпера може випускатися тільки одним виробником.

Зв'язок між сутністю `CamperModel` і сутністю `Camper` один до багатьох: одна модель кемпера може бути представлена різними кемперами (з різною комплектацією, докомплектацією тощо), а кожен кемпер може бути розроблений початково тільки як одна конкретна модель.

Зв'язок між сутністю `Camper` і сутністю `Advertisement` один до багатьох: один кемпер може бути представлений різними оголошеннями, тобто фактично різними фізичними кемперами з окремими реєстраційними номерними знаками і відповідним оголошенням (наприклад, агентство може мати набір однакових кемперів), але кожне оголошення може співставлятися тільки з одним кемпером.

Зв'язок між сутністю `Advertisement` і сутністю `RentalRate` один до багатьох: одне оголошення може мати різні цінові пропозиції, у той же час кожна конкретна пропозиція визначається для одного конкретного оголошення.

Зв'язок між сутністю `Advertisement` і сутністю `CamperRental` один до багатьох: одне оголошення може мати багато прокатів, проте кожен прокат стосується тільки одного оголошення.

Зв'язок між сутністю Advertisement і сутністю Proposition один до багатьох: одне оголошення може мати різні пропозиції щодо перевезення кемпера, але кожне перевезення стосується тільки одного конкретного кемпера.

Зв'язок між сутністю OwnerProfile і сутністю Advertisement один до багатьох: один власник може мати багато оголошень про прокат кемперів, але у кожного оголошення тільки один власник.

Зв'язок між сутністю GuestProfile і сутністю CamperRental один до багатьох: один орендар може мати багато прокатів кемперів, а кожен прокат може виконуватись тільки одним орендарем.

Зв'язок між сутністю Station і сутністю Proposition двічі один до багатьох: за однією станцією може бути багато пропозицій, в яких вона вказана як початкова точка перевезення кемпера, і також багато пропозицій, в яких станція вказана як кінцева точка перевезення кемпера, у той же час кожна пропозиція може мати тільки одну початкову і одну кінцеву точки.

Зв'язок між сутністю Station і сутністю CamperRental двічі один до багатьох: за однією станцією може бути багато прокатів, з яких прокат починався, і також багато прокатів, які за даною станцією завершувались, у той же час кожен прокат може мати тільки одну початкову і одну кінцеву станції.

Зв'язок між сутністю City і сутністю Station один до багатьох: одне місто може мати багато станцій, але одна станція розташовується в конкретному місті.

Для реалізації БД було використано систему керування БД Microsoft SQL Server. Це рішення від компанії Microsoft, що є стандартним і логічним рішенням, зважаючи на використання мови програмування C#. Окрім того Microsoft SQL Server є надійною і потужною системою керування БД, що є адекватним рішенням, зважаючи на масштаб системи.

Схема створеної БД представлена на рис. 2.3.

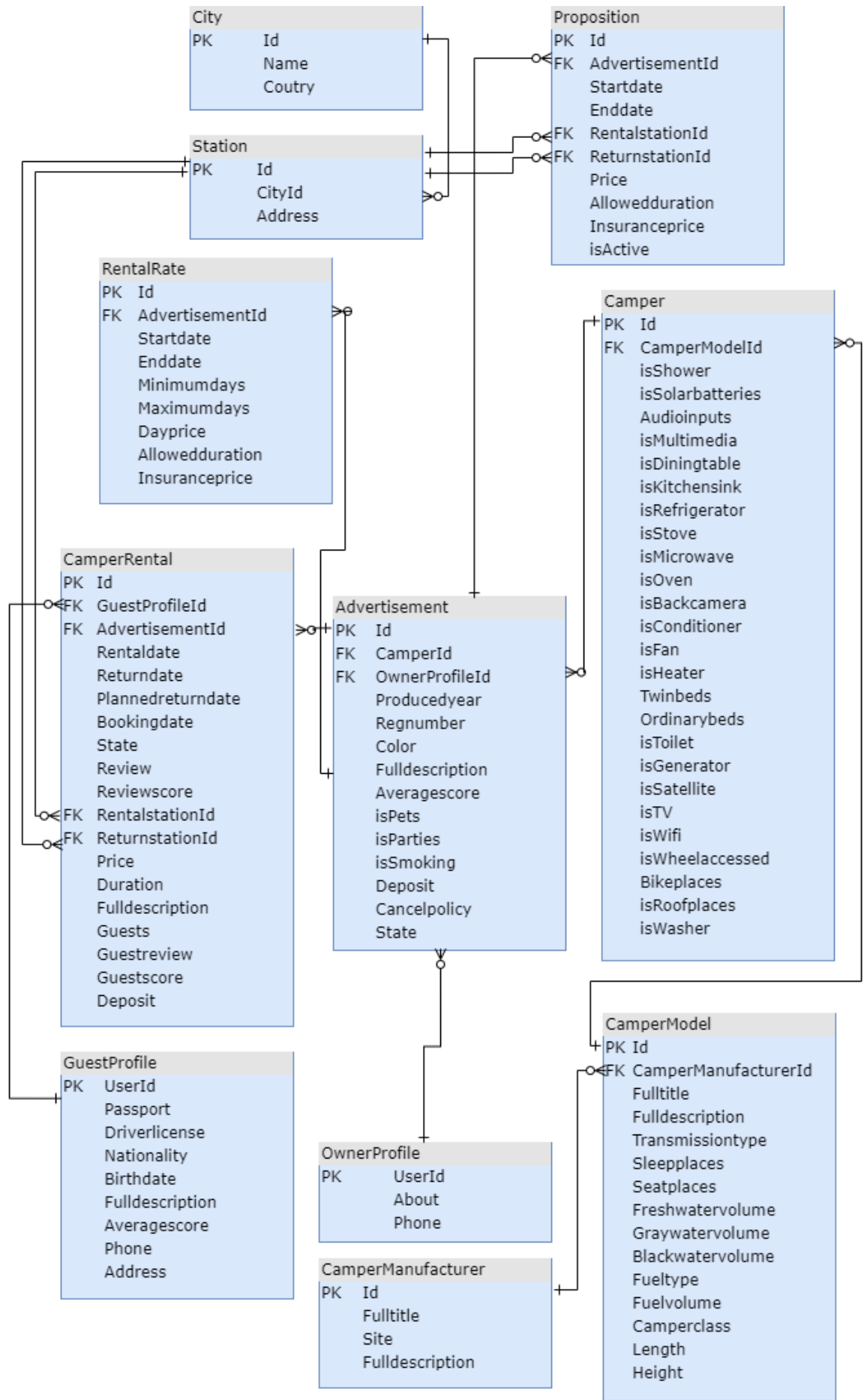


Рисунок 2.3 – Схема БД

2.5 Визначення структури програми

Використовуючи фреймворк ASP.NET Core, була розроблена структура програми (рис. 2.4).

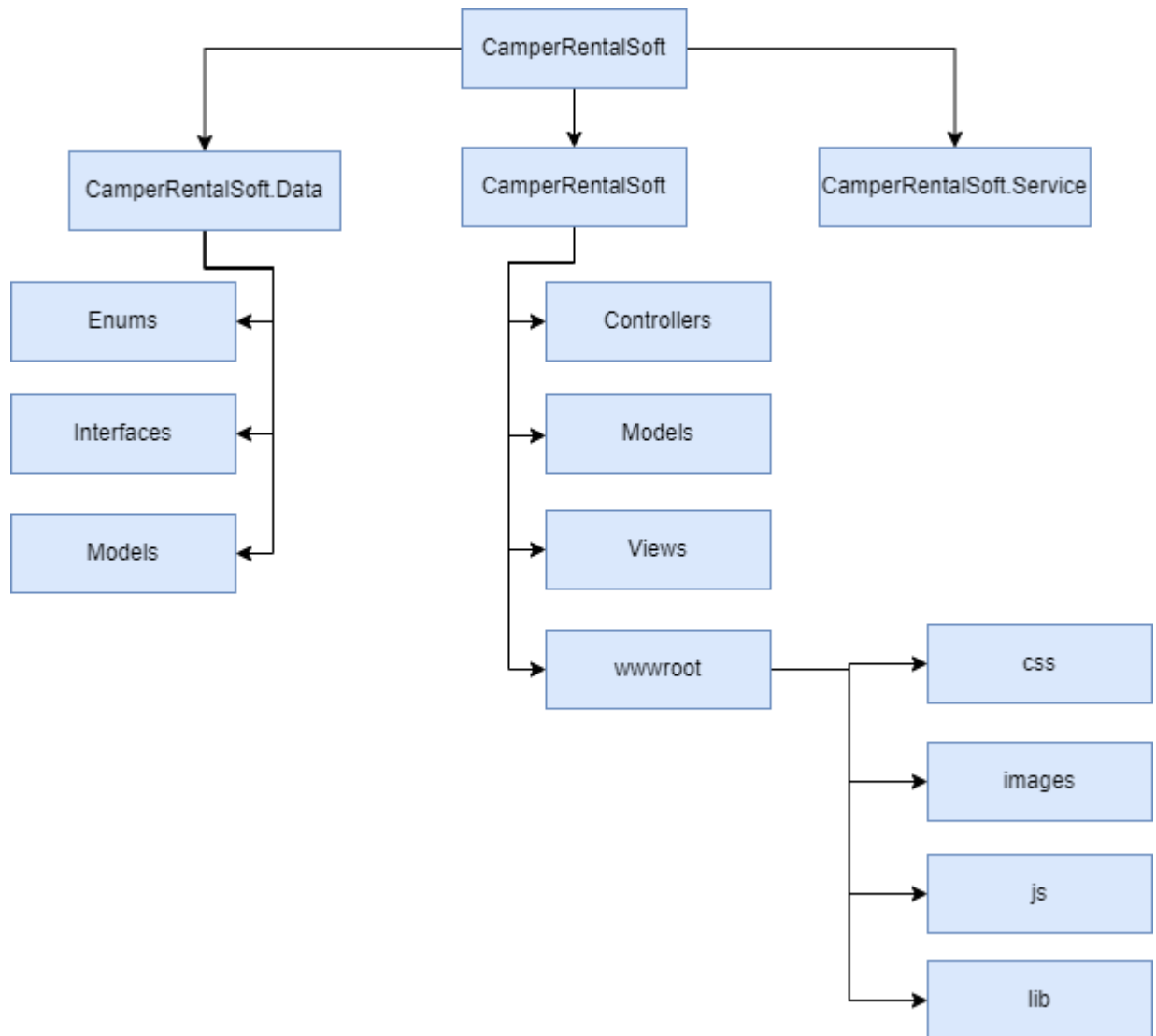


Рисунок 2.4 – Структурна схема програми

Структура програми зокрема складається з набору базових моделей для роботи з даними, що мали визначати в подальшому структуру відповідних таблиць БД. Дані базові моделі було об'єднано в теці CamperRentalSoft.Data, де окрім безпосередньо даних моделей в теці Models було окремо виділено теку Enums з визначенням перелічень, необхідних для визначення варіантів політики скасування замовлень, станів прокатів, класів

кемперів, типів палива, типів трансмісії кемпера. Також було визначено теку Interfaces для визначення інтерфейсів взаємодії з БД, які в подальшому реалізовувались в сервісах системи.

Сервіси системи у свою чергу були виділені в теці CamperRentalSoft.Service. Сервіси визначаються для надання засобів прямої взаємодії з БД для виконання основних функцій, пов'язаних з забезпеченням роботи загальних функцій системи. Сервіси реалізують інтерфейси системи.

Засоби забезпечення роботи ПЗ у відповідності зі сценаріями користувачів виділені в теці CamperRentalSoft в структурі системи. Тут виділено моделі, що забезпечують додаткові можливості, не пов'язані з безпосереднім відтворенням даних у таблицях БД, контролери для забезпечення сторінок системи даними і представлення системи для відтворення цих даних у вигляді сторінок у браузері, а також файли, що відносяться до файлів зображень, скриптів javascript, бібліотек jquery, bootstrap.

2.6 Висновки за розділом 2

Для реалізації програми було обрано мову програмування C#, вебфреймворк ASP.NET Core, що загалом забезпечує можливість створення кросплатформного рішення з підтримкою системи керування базами даних Microsoft SQL Server для роботи з даними. Визначено на основі використання даного вебфреймворку структуру системи та визначено структуру бази даних. Виконано моделювання сценаріїв роботи користувачів з системою.

3 РОЗРОБКА ПРОГРАМИ

3.1 Алгоритм функціонування програми

Послідовність функціонування програми була визначена схемою функціонування, яка передбачає розподіл основного потоку на взаємодію з двома ролями користувачів (рис. 3.1).

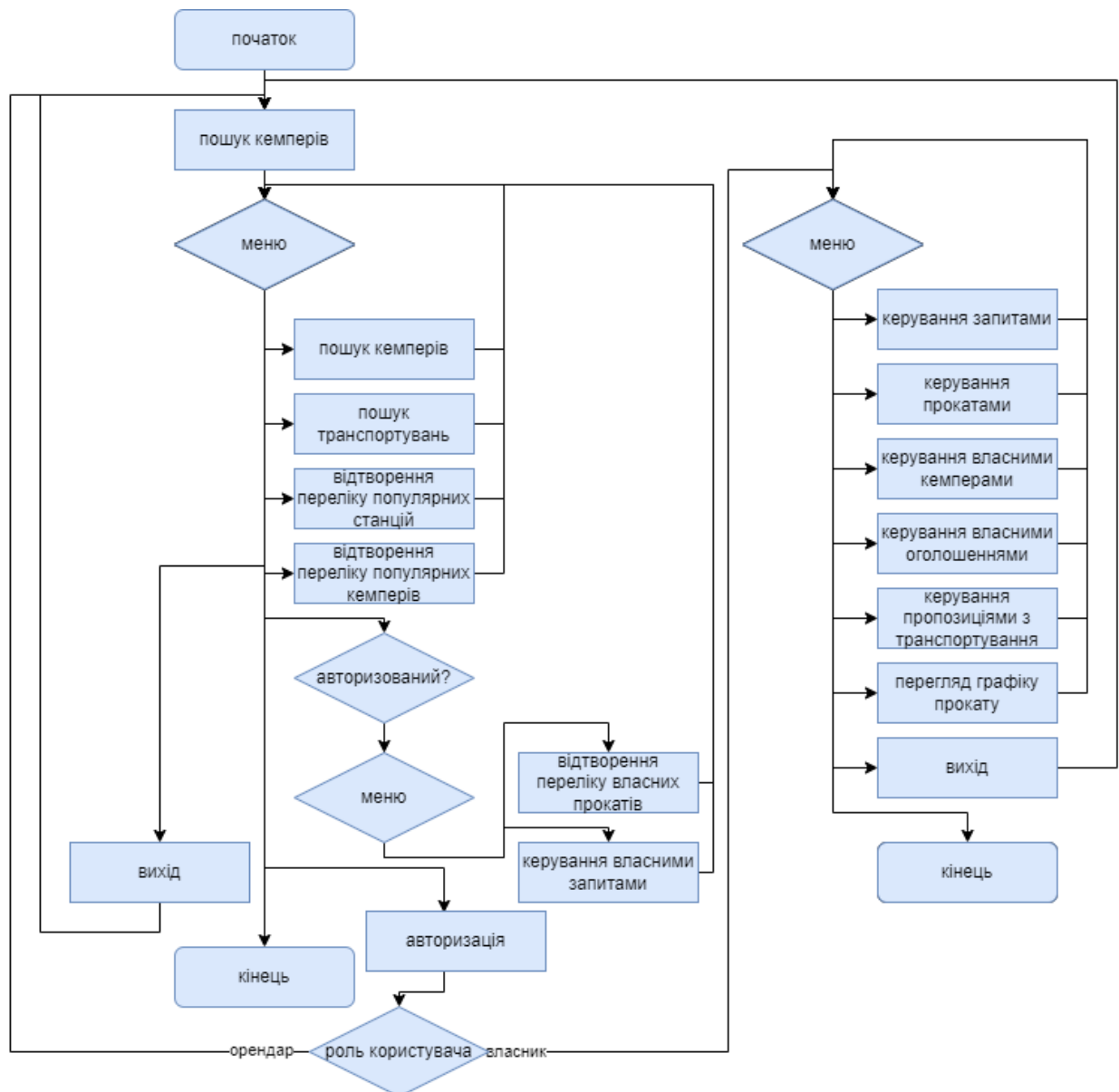


Рисунок 3.1 – Схема функціонування програми

3.2 Опис реалізованих класів

Основні розроблені класи програми охоплюють класи моделей, класи контролерів та класи сервісів. Класи базових моделей використані для роботи модуля об'єктно-реляційного перетворення Entity Framework Core, що дозволило побудувати відповідні таблиці БД на основі написаного коду наступних класів.

Клас CamperManufacturer реалізує модель виробника кемперів на основі полів:

- int Id – ідентифікаційний номер виробника кемперів;
- string Fulltitle – повна назва виробника;
- string Site – вебсайт виробника кемперів;
- string Fulldescription – повний опис виробника кемперів;
- virtual ICollection<CamperModel> CamperModels – колекція моделей кемперів даного виробника.

Клас CamperModel реалізує модель моделі кемпера на основі полів:

- int Id – ідентифікаційний номер моделі кемпера;
- int CamperManufacturerId – ідентифікаційний номер виробника даної моделі кемпера, що є зовнішнім ключем;
- string Fulltitle – повна назва моделі кемпера;
- string Fulldescription – повний опис моделі кемпера;
- TransmT Transmissiontype – тип трансмісії кемпера;
- byte Sleepplaces – кількість спальних місць у кемпері;
- byte Seatplaces – кількість місць для сидіння у кемпері;
- float Freshwatervolume – обсяг баку для питної води;
- float Graywatervolume – обсяг баку для брудної води;
- float Blackwatervolume – обсяг баку для відходів;
- FuelT Fueltype – тип пального;
- float Fuelvolume – обсяг паку з паливом;
- ClassT Camperclass – клас кемпера даної моделі;

- float Length – довжина кемпера даної моделі;
- float Height – висота кемпера даної моделі;
- virtual CamperManufacturer CamperManufacturer – об’єкт виробника даної моделі кемпера;
- virtual ICollection<Camper> Campers – колекція кемперів даної моделі.

Клас Camper реалізує модель кемпера на основі полів:

- int Id – ідентифікаційний номер кемпера;
- int CamperModelId – ідентифікаційний номер моделі кемпера, що є зовнішнім ключем;
- bool isShower – наявність душу;
- bool isSolarbatteries – наявність сонячних батарей;
- byte Audioinputs – кількість аудіо виходів;
- bool isMultimedia – наявність системи мультимедіа;
- bool isDiningtable – наявність обіднього столу;
- bool isKitchensink – наявність кухонної раковини;
- bool isRefrigerator – наявність холодильника;
- bool isStove – наявність кухонної плити;
- bool isMicrowave – наявність мікрохвильової пічі;
- bool isOven – наявність духової шафи;
- bool isBackcamera – наявність камери заднього виду;
- bool isConditioner – наявність кондиціонера;
- bool isFan – наявність вентилятора;
- bool isHeater – наявність обігріву;
- byte Twinbeds – кількість двоспальних ліжок;
- byte Ordinarybeds – кількість одномісних ліжок;
- bool isToilet – наявність туалету;
- bool isGenerator – наявність генератора;
- bool isSatellite – наявність супутникової антени;
- bool isTV – наявність телевізора;

- bool isWifi – наявність wi-fi;
- bool isWheelaccessed – наявність доступу осіб з інвалідністю;
- byte Bikeplaces – кількість місць для велосипедів;
- bool isRoofplaces – наявність місць на даху;
- bool isWasher – наявність пральної машини;
- virtual CamperModel CamperModel – модель кемпера;
- virtual ICollection<Advertisement> Advertisements – колекція оголошень, що відповідають даному кемперу.

Клас Advertisement реалізує модель оголошення, тобто фактично екземпляра кемпера, на основі полів:

- int Id – ідентифікаційний номер оголошення;
- int CamperId – ідентифікаційний номер кемпера, що є зовнішнім ключем;
- int OwnerProfileId – ідентифікаційний номер профіля власника кемпера, що є зовнішнім ключем;
- DateTime Producedyear – рік (дата) виробництва кемпера;
- string Regnumber – реєстраційний номер кемпера;
- string Color – колір кемпера;
- string Fulldescription – повний опис кемпера;
- float Averagescore – середня оцінка кемпера за даним оголошенням;
- bool State – стан оголошення;
- bool isPets – можливість прокату з домашніми тваринами;
- bool isParties – можливість проведення вечірок в кемпері;
- bool isSmoking – можливість паління в кемпері;
- float Deposit – депозит, який необхідно внести для прокату даного кемпера;
- CancelT Cancelpolicy – політика скасування прокату кемпера;
- virtual Camper Camper – кемпер, якому відповідає екземпляр кемпера з оголошення;
- virtual GuestProfile OwnerProfile – власник кемпера з оголошення;

– virtual ICollection<CamperRental> CamperRentals – колекція прокатів даного кемпера;

– virtual ICollection<RentalRate> RentalRates – колекція цінових пропозицій на прокат даного кемпера;

– virtual ICollection<Proposition> Propositions – колекція пропозицій про транспортування кемпера з оголошення.

Клас CamperRental реалізує модель прокату кемпера на основі полів:

– int Id – ідентифікаційний номер прокату кемпера;
 – int AdvertisementId – ідентифікаційний номер оголошення про прокат кемпера, за яким здійснюється прокат, є зовнішнім ключем;

– int GuestProfileId – ідентифікаційний номер орендаря, який здійснює прокат кемпера, є зовнішнім ключем;

– DateTime Rentaldate – дата початку прокату кемпера;

– DateTime Returndate – дата завершення прокату кемпера;

– DateTime Plannedreturndate – запланована дата повернення кемпера;

– DateTime Bookingdate – дата бронювання прокату кемпера;

– StateT State – стан прокату кемпера;

– string Review – відгук про прокат кемпера від орендаря;

– byte Reviewscore – оцінка прокату кемпера, виставлена орендарем;

– int RentalstationId – ідентифікаційний номер станції, де було взято кемпер у прокат;

– int ReturnstationId – ідентифікаційний номер станції, де було повернуто кемпер з прокату;

– float Price – вартість прокату;

– byte Duration – відстань, яку було подолано кемпером за час прокату;

– string Fulldescription – повний опис прокату;

– byte Guests – кількість гостей;

– string Guestreview – відгук про орендаря від власника кемпера;

– byte Guestscore – оцінка орендарю від власника кемпера;

- float Deposit – внесений депозит або загальна сплачена на поточний момент сума за прокат кемпера;
- virtual Advertisement Advertisement – оголошення, за яким здійснюється прокат;
- virtual Station Rentalstation – станція, з якої взято в прокат кемпер;
- virtual Station Returnstation – станція, куди повернуто кемпер з прокату;
- virtual GuestProfile GuestProfile – орендар.

Клас RentalRate реалізує модель цінової пропозиції прокату кемпера на основі полів:

- int Id – ідентифікаційний номер цінової пропозиції прокату кемпера;
- int AdvertisementId – ідентифікаційний номер оголошення про прокат кемпера, якому відповідає цінова пропозиція, є зовнішнім ключем;
- DateTime Startdate – дата, з якої можна взяти кемпер у прокат, за цією пропозицією;
- DateTime Enddate – дата, до якої можна взяти кемпер у прокат, за цією пропозицією;
- byte Minimumdays – мінімальна кількість днів, на яку діє дана цінова пропозиція;
- byte Maximumdays – максимальна кількість днів, на яку діє дана цінова пропозиція;
- float Dayprice – вартість прокату за день;
- float Allowedduration – дозволена максимальна відстань, яку можна подолати під час прокату кемпера за цією ціною пропозицією;
- float Insuranceprice – вартість страхування;
- virtual Advertisement Advertisement – оголошення, до якого належить дана цінова пропозиція.

Клас Proposition реалізує модель пропозиції про транспортування кемпера на основі полів:

- int Id – ідентифікаційний номер пропозиції;

- int AdvertisementId – ідентифікаційний номер оголошення, є зовнішнім ключем;
- DateTime Startdate – дата, починаючи з якої можна взяти кемпер у прокат;
- DateTime Enddate – дата, до якої потрібно повернути кемпер з прокату в потрібну точку;
- int RentalstationId – ідентифікаційний номер станції, в якій знаходиться кемпер;
- int ReturnstationId – ідентифікаційний номер станції, в яку потрібно повернути кемпер;
- float Price – вартість, яку потрібно сплатити;
- float Allowedduration – відстань, яку можна подолати під час транспортування за цією вартістю;
- float Insuranceprice – вартість страхування;
- bool isActive – активність пропозиції;
- virtual Advertisement Advertisement – оголошення, до якого належить дана пропозиція;
- virtual Station Rentalstation – станція, в якій знаходиться кемпер;
- virtual Station Returnstation – станція, в яку потрібно повернути кемпер.

Окрім того було розроблено класи моделей, які не є базовими і які використовуються для підвищення зручності роботи з даними під час реалізації різних функцій системи.

Клас PropositionSearch визначає модель пошуку пропозиції на основі полів:

- Station station2 – станція, куди орендар хотів би повернути кемпер;
- City city2 – варіант визначення міста повернення кемпера (замість конкретної станції).

Також даний клас має вкладений клас AdvertisementSearch, який визначає модель пошуку оголошення про прокат кемпера на основі полів:

- Station station1 – станція, звідки орендар хотів би забрати кемпер;
- City city1 – варіант визначення міста прокату кемпера (замість конкретної станції);
- int maxprice – максимальна сума, яку готовий сплатити орендар;
- int minprice – мінімальна сума, яку готовий сплатити орендар;
- string date1 – дата, починаючи з якої орендар готовий подорожувати;
- string date2 – дата, завершуючи якою орендар готовий подорожувати;
- string manufacturer – бажаний виробник кемпера;
- string model – бажана модель кемпера;
- TransmT transmission – бажаний тип трансмісії кемпера;
- byte sleep – бажана кількість спальних місць у кемпері;
- byte seat – бажана кількість місць для сидіння у кемпері;
- float minfreshwater – мінімальний обсяг баку для питної води;
- float mingraywater – мінімальний обсяг баку для брудної води;
- float minblackwater – мінімальний обсяг баку для відходів;
- FuelT fuel – бажаний тип пального;
- ClassT campclass – бажаний клас кемпера;
- float maxlength – максимальна довжина кемпера;
- float maxheight – максимальна висота кемпера;
- bool isShower – необхідність наявності душі;
- bool isSolarbatteries – необхідність наявності сонячних батарей;
- byte Audioinputs – кількість аудіо виходів;
- bool isMultimedia – наявність системи мультимедіа;
- bool isDiningtable – наявність обіднього столу;
- bool isKitchensink – наявність кухонної раковини;
- bool isRefrigerator – наявність холодильника;
- bool isStove – наявність кухонної плити;
- bool isMicrowave – наявність мікрохвильової пічі;
- bool isOven – наявність духової шафи;

- bool isBackcamera – наявність камери заднього виду;
- bool isConditioner – наявність кондиціонера;
- bool isFan – наявність вентилятора;
- bool isHeater – наявність обігріву;
- byte Twinbeds – необхідна кількість двоспальних ліжок;
- byte Ordinarybeds – необхідна кількість одномісних ліжок;
- bool isToilet – необхідність наявності туалету;
- bool isGenerator – необхідність наявності генератора;
- bool isSatellite – необхідність наявності супутникової антени;
- bool isTV – необхідність наявності телевізора;
- bool isWifi – необхідність наявності wi-fi;
- bool isWheelaccessed – необхідність наявності доступу осіб з інвалідністю;
- byte Bikeplaces – кількість місць для велосипедів;
- bool isRoofplaces – необхідність наявності місць на даху;
- bool isWasher – необхідність наявності пральної машини;
- byte minscore – мінімальна оцінка даного кемпера;
- byte maxscore – максимальна оцінка даного кемпера;
- bool isPets – необхідність прокату з домашніми тваринами;
- bool isParties – необхідність проведення вечірок в кемпері;
- bool isSmoking – необхідність паління в кемпері;
- CancelT canpolice – бажана політика скасування прокату кемпера.

Клас сервісу оголошень AdvertisementService містить наступні методи:

- AdvertisementService (CRSDatabaseContext crsdbc) – конструктор, який задає в якості контексту БД crsdbc;
- Advertisement GetById(int advid) – метод, який повертає повний об'єкт класу Advertisement за ідентифікаційним номером оголошення advid;
- IEnumerable<Advertisement> GetAllAdvertisements() – метод, який повертає перелік всіх оголошень;

- `IEnumerable<Advertisement> GetAdvertisementsByOwner(int guest)` – метод, який повертає перелік оголошень, які були належать користувачу з ідентифікаційним номером `guest`;
- `IEnumerable<CamperRental> GetCamperRentalsForAdvertisement(int adv)` – повертає перелік всіх прокатів, що оформлені були для оголошення з номером `adv`;
- `IEnumerable<Proposition> GetPropositionsForAdvertisement(int adv)` – повертає перелік всіх пропозицій щодо транспортування кемпера за оголошенням з номером `adv`;
- `IEnumerable<RentalRate> GetRentalRatesForAdvertisement(int adv)` – повертає перелік всіх цінових пропозицій для оголошення з ідентифікаційним номером `adv`;
- `Task Add(Advertisement adv)` – метод, який реалізує створення оголошення з номером `adv` з внесенням відповідних змін в БД;
- `Task CalculateAverageAdvertisementScore(int adv)` – метод, який реалізує обчислення середнього балу за оголошенням з ідентифікаційним номером `adv`;
- `IEnumerable<Advertisement> GetBestAdvertisements(int n)` – метод, який повертає `n` активних оголошень, які є найкращими за середнім балом;
- `Advertisement GetLastAdvertisement()` – метод, який повертає останнє створене оголошення;
- `IEnumerable<Advertisement> GetSearchedAdvertisements(AdvertisementSearch as)` – метод, який реалізує пошук оголошень за параметрами пошуку, визначеними об'єктом `as` класу `AdvertisementSearch`;
- `Advertisement GetAdvertisementByRegnumber(string regnumber, int guest)` – метод, який виконує пошук оголошення про прокат кемпера з реєстраційним номером `regnumber` за власником з ідентифікаційним номером `guest`;
- `IEnumerable<Advertisement> GetPopularOwnAdvertisements(int guest)` – повертає для власника з ідентифікаційним номером `guest` перелік його

активних оголошень, відсортованих за зменшенням кількості замовлень на прокат кемпера за даними оголошеннями.

3.3 Висновки за розділом 3

Для реалізації роботи з даними використано модуль об'єктно-реляційного перетворення Entity Framework Core, що забезпечив реалізацію БД на основі написання коду. Усі складові системи, визначені структурною схемою, були успішно реалізовані на основі відповідних створених класів, а реалізовані рішення відповідно були описані.

4 ЕКСПЛУАТАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМИ

4.1 Призначення програми

Програма призначена для використання агентствами з прокату, індивідуальними власниками кемперів і потенційними орендарями. Програма повинна надавати можливість керування власниками прокатом власних кемперів, а орендарям – можливість замовлення прокату кемперів.

4.2 Умови виконання програми

Для забезпечення підтримки роботи вебсервера та сервера БД необхідно надати апаратне забезпечення, що включає мінімально процесор, що має не менше ніж 4 ядра з тактовою частотою не менше 2,1 ГГц, підтриманий оперативною пам'яттю обсягом не менше 8 Гб та жорстким диском з доступними 10 Гб простору на ньому.

4.3 Експлуатація програми

Експлуатація програми здійснюється через звернення до вебсервера у випадку роботи орендаря та власника. У випадку роботи адміністратора потрібно додати до адреси /admin.

Працювати з програмою починає користувач у будь-якому випадку як неавторизований орендар. Меню неавторизованого орендаря повністю співпадає з меню авторизованого орендаря, проте не всіма пунктами такий користувач зможе скористатися. У випадку вибору недоступних функцій відбувається переадресація на сторінку авторизації користувача.

Відповідне меню включає пункти «Пошук» для вибору пошуку з оголошень (рис. 4.1) або варіантів транспортування випадального підменю, «Прокати» для перегляду власних запитів або прокатів, «Станції» і «Кемпери» для перегляду популярних станцій і кемперів, «Авторизація» (для

Рисунок 4.1 – Визначення параметрів пошуку

У випадку пошуку дозволяється визначити великий набір параметрів, а після натискання кнопки «Шукати» відображаються результати пошуку (рис. 4.2). Для кожного результату відображається одне зображення кемпера (зберігається у відповідній папці проєкту за номером без збереження в самій БД), назва виробника і моделі, логін власника, середня оцінка та кількість відгуків орендарів.

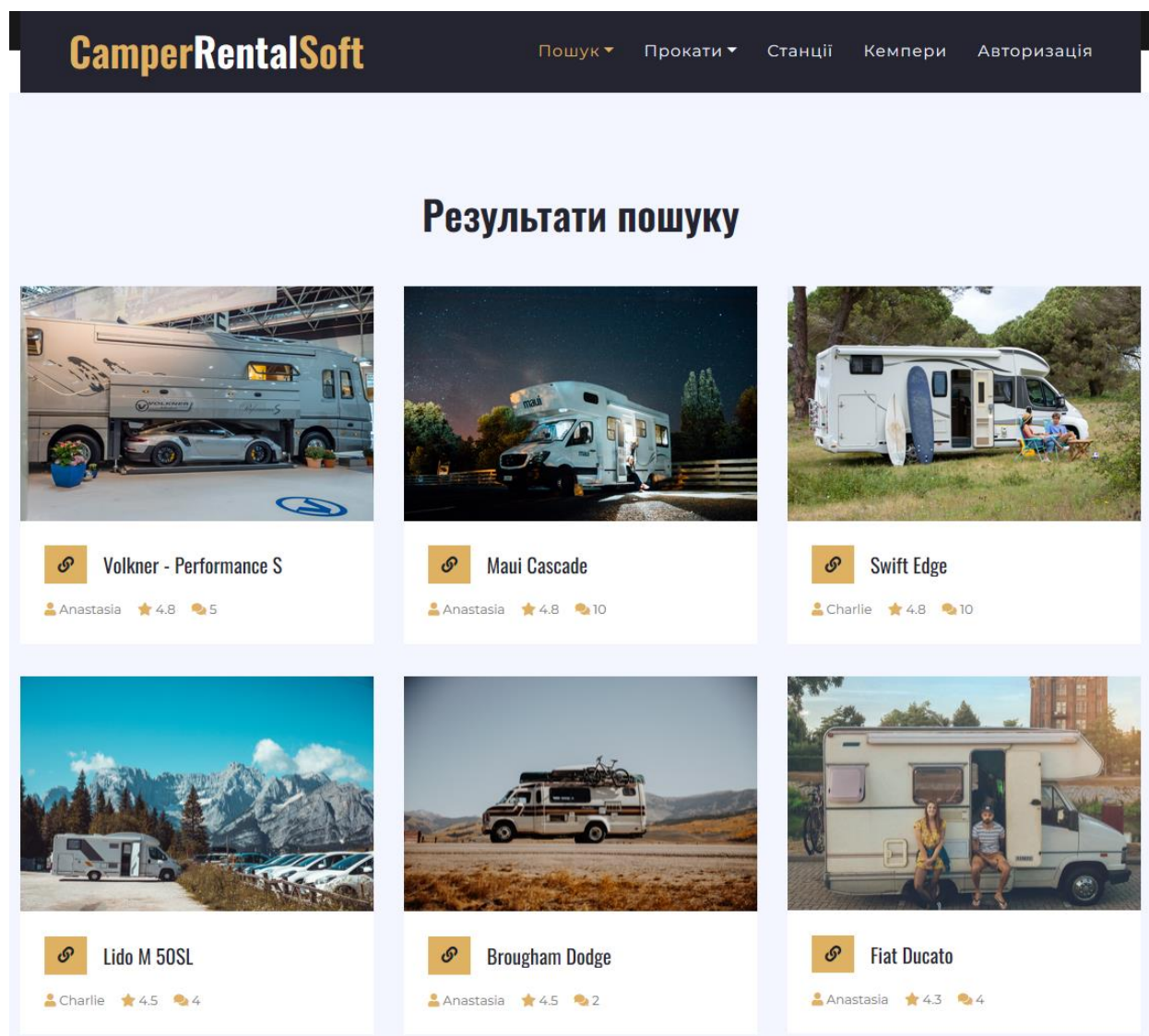


Рисунок 4.2 – Результати пошуку

При відкритті відповідного оголошення відтворюються фотографії та чотири блоки: «Кемпер», «Оголошення», «Транспортування», «Відгуки» (рис. 4.3), де можна переглянути всі відгуки.

CamperRentalSoft

Пошук

Прокати

Станції

Кемпери

Профіль

ОГОЛОШЕННЯ

FIAT Bavaria T6 MFD

Кемпер

Оголошення

Транспортування

Відгуки



Відгуки наших орендарів

ВІДГУК

3 перших вуст



Charlie

Оцінка: 5.

Кемпер не новенький, але в прекрасному стані. Отримали велике задоволення від подорожі Ісландією. Всі зручності у кемпері майже наче вдома.

24.05.2022

<

>

Рисунок 4.3 – Перегляд відгуків за оголошенням про прокат кемпера

У блоці «Кемпер» можна переглянути всю інформацію про даний кемпер, у блоці «Оголошення» все те, що стосується до умов оголошення, а також можливість визначення дат прокату. У блоці «Транспортування» можна переглянути доступні пропозиції з транспортування для кемпера з оголошення.

У власника меню включає пункти «Запити» для керування запитами орендарів (рис. 4.4), «Прокати» для керування підтвердженими прокатами кемперів, що належать власнику, «Кемпери» для роботи над створенням в системі власних кемперів, перегляду їх популярності та популярності станцій, «Оголошення» для керування оголошеннями, керування пропозиціями з транспортування кемперів, перегляду графіку прокату.

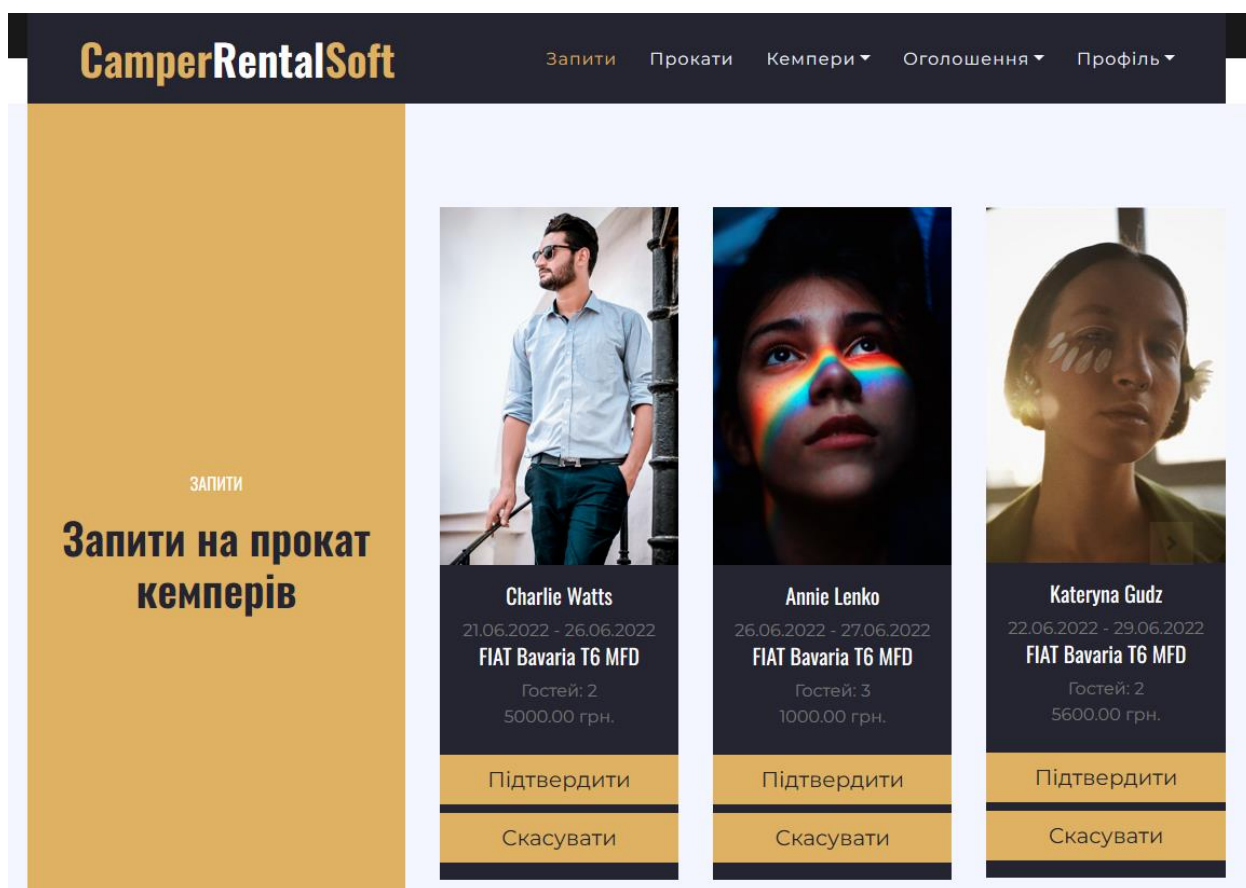


Рисунок 4.4 – Керування запитами на прокат кемперів власником

4.4 Тестування програми

Для перевірки працездатності програми було виконано її тестування, що підтвердило реалізацію всіх запланованих функцій програми (табл. 4.1).

Таблиця 4.1 – Тестування програми

Функція	Власник	Орендар		Адміністратор
		Неавто- ризований	Авто- ризований	
1	2	3	4	5
Пошук варіантів прокату кемперів		+	+	
Перегляд власних запитів прокату кемпера			+	
Перегляд власних прокатів кемперів			+	
Оцінювання прокату			+	
Пошук варіантів транспортування кемпера		+	+	
Перегляд запитів на прокат кемпера	+			
Підтвердження запита на прокат кемпера	+			
Скасування запита на прокат кемпера	+			
Перегляд власних вільних кемперів	+			

Продовження таблиці 4.1

1	2	3	4	5
Визначення пропозицій з транспортування кемпера	+			
Скасування пропозиції з транспортування кемпера	+			
Перегляд графіка прокату заданого власного кемпера	+			
Перегляд популярності власних кемперів	+			
Створення відгуку про орендаря			+	
Перегляд популярних станцій прокату	+	+	+	
Перегляд популярних моделей кемперів	+	+	+	
Перегляд неоцінених власних прокатів			+	
Реєстрація	+	+	+	
Авторизація	+	+	+	+
Перегляд профілю орендаря	+	+	+	
Перегляд профілю власника	+	+	+	
Видача кемпера в прокат	+			
Повернення кемпера з прокату	+			
Перенесення фактичного знаходження кемпера в задану станцію	+			

Продовження таблиці 4.1

1	2	3	4	5
Додавання виробника кемпера				+
Редагування виробника кемпера				+
Додавання моделі кемпера				+
Редагування моделі кемпера				+
Додавання кемпера	+			
Редагування кемпера	+			
Додавання оголошення	+			
Редагування оголошення	+			
Визначення недоступних для прокату кемпера дат	+			
Пошук серед власних кемперів	+			
Створення запиту на транспортування кемпера			+	
Перегляд найкращих кемперів	+			
Архівувати власне оголошення	+			

Кінець таблиці 4.1

4.5 Висновки за розділом 4

Описано принципи роботи з програмою для орендаря та власника кемперів, а також виконано тестування програми, що підтвердило коректну реалізацію функціональних вимог.

ВИСНОВКИ

Під час виконання роботи було досягнуто її мету, що полягала в розробці ПЗ прокату кемперів для підтримки роботи агентств з прокату та індивідуальних власників кемперів. Для цього було проведено аналіз предметної області, включаючи визначення класів кемперів, основних їх сучасних складових, огляд програмних аналогів, за результатами якого визначено особливості програми, яка розроблялась.

Для реалізації програми було обрано мову програмування C#, вебфреймворк ASP.NET Core, що загалом забезпечило можливість створення кросплатформного рішення з підтримкою системи керування базами даних Microsoft SQL Server для роботи з даними. Визначено на основі використання даного вебфреймворку структури системи та визначено структуру бази даних. Для реалізації роботи з даними використано модуль об'єктно-реляційного перетворення Entity Framework Core, що забезпечив реалізацію БД на основі написання коду. Усі складові системи, визначені структурною схемою, були успішно реалізовані на основі відповідних створених класів, а реалізовані рішення відповідно були описані.

Розробка ПЗ прокату кемперів надала можливість організувати платформу для підтримки роботи окремого агентства, що може бути організовано в різних країнах світу, в тому числі в декількох одночасно, з можливістю підтримки роботи одразу декількох агентств, а також створення оголошень від індивідуальних власників кемперів. Програма дозволяє виконувати пошук кемперів, бронювання прокату кемперів, керувати бронюваннями власних кемперів, створювати відгуки орендарю про власника, а власнику про орендаря.

Роботу розробленого ПЗ було підтверджено за результатами проведеного тестування, коли було перевірено функціонування усіх визначених функцій програми.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Latest RV Statistics 2020-2021 [Electronic resource] : Latest Data & Trends. – Access mode : <https://www.condorferries.co.uk/rv-statistics>.
2. Campervan Popularity in Europe vs. The US [Electronic resource]. – Access mode : <https://mobilerving.com/amp/the-buzz-articles/tech-trends/campervan-popularity-in-europe-vs-the-us>.
3. Chang, B. Millennials are flocking to RVs like never before during the pandemic, new data shows [Electronic resource] / B. Chang. – Access mode : <https://www.businessinsider.com/data-millennials-are-now-more-interested-in-buying-rvs-2020-12>.
4. Barber, M. From Tiny Trailers to Huge RVs, a Definitive Guide to Every Type of Camper [Electronic resource] / M. Barber. – Access mode : <https://www.curbed.com/article/rv-class-comparison-travel-trailer-camper-van-guide.html>.
5. Complete List of RV Manufacturers [Electronic resource] : All the Brands & Types. – Access mode : <https://www.thewanderingrv.com/rv-manufacturers/>.
6. Features of RVs [Electronic resource] : What Do You Really Need?. – Access mode : <https://www.peanutorspretzels.com/features-of-rv/>.
7. Wendland, M. The 10 Most Important RV Features We Can't Live Without [Electronic resource] / M. Wendland. – Access mode : <https://rvlifestyle.com/most-important-rv-features/>.
8. Trusted RV Rental Marketplace. Outdoorsy [Electronic resource]. – Access mode : <https://www.outdoorsy.com/>.
9. RV Rentals – Direct from Local Owners [Electronic resource]. – Access mode : <https://rvshare.com/>.
10. America's RV Rental Experts – Guaranteed Bookings [Electronic resource]. – Access mode : <https://www.cruiseamerica.com/>.
11. Price, M. J. C# 10 and .NET 6 – Modern Cross-Platform Development

[Text] : Build apps, websites, and services with ASP.NET Core 6, Blazor, and EF Core 6 using Visual Studio 2022 and Visual Studio Code / M. J. Price. – Birmingham : Packt Publishing, 2021. – 824 p.

12. Albahari, J. C# 10 in a Nutshell [Text] : The Definitive Reference / J. Albahari. – Massachusetts : O'Reilly Media, 2022. – 1058 p.

13. Murach, J. Murach's ASP.NET Core MVC [Text] / J. Murach. – California : Mike Murach & Associates, 2020. – 780 p.

ДОДАТОК А
Технічне завдання

Вступ

Прокат кемперів є однією з потужних ніш розвитку туризму в світі, але в даному випадку індивідуального або сімейного туризму, тобто без участі туристичних агентств. У деяких країнах достатньо популярним є придбання у власність кемперів, проте в менш економічно розвинутих країнах більш популярним є прокат кемперів. Саме для даного напрямлення і призначено розробку ПЗ прокату кемперів у роботі.

A.1 Підстави для розробки

Дипломна кваліфікаційна робота виконується за темою «Програмне забезпечення прокату кемперів» у відповідності з наказом № 102 від 27 квітня 2022 р. за Національним університетом «Запорізька політехніка».

A.2 Призначення розробки

Програма призначена для використання агентствами з прокату, індивідуальними власниками кемперів і потенційними орендарями. Програма повинна надавати можливість керування власниками прокатом власних кемперів, а орендарям – можливість замовлення прокату кемперів.

A.3 Основні вимоги до програми

A.3.1 Вимоги до функціональних характеристик

Від набору функціональних характеристик вимагається виконання наступного:

- пошук варіантів прокату кемперів;
- перегляд власних запитів прокату кемпера;
- перегляд власних прокатів кемперів;

- оцінювання прокату;
- пошук варіантів транспортування кемпера;
- визначення витрат на прокат кемпера;
- перегляд запитів на прокат кемпера;
- підтвердження або скасування запита на прокат кемпера;
- перегляд власних вільних кемперів на задану дату або за заданою станцією;
- перегляд прокатів власником за календарем;
- визначення пропозицій з транспортування кемпера;
- скасування пропозиції з транспортування кемпера;
- перегляд графіка прокату заданого власного кемпера;
- перегляд популярності власних кемперів;
- створення відгуку про орендаря;
- перегляд популярних станцій прокату;
- перегляд популярних моделей кемперів;
- перегляд неоцінених власних прокатів;
- архівувати власне оголошення;
- реєстрація орендарів, власників;
- авторизація;
- перегляд профілів орендарів і власників;
- видача кемпера в прокат;
- повернення кемпера з прокату;
- перенесення фактичного знаходження кемпера в задану станцію;
- додавання і редагування виробників кемперів;
- додавання і редагування моделей кемперів;
- додавання і редагування кемперів власником;
- додавання і редагування оголошення про прокат кемпера;
- визначення недоступних для прокату кемпера дат;
- створення запиту на прокат кемпера
- скасування запиту на прокат кемпера;

- створення запиту на транспортування кемпера;
- пошук власником серед власних кемперів;
- перегляд найкращих кемперів.

A.3.2 Умови експлуатації

Для експлуатації ПЗ потрібно забезпечити окремий сервер, на якому буде запущено вебсервер і сервер БД для постійного доступу користувачів до сервісу прокату кемперів.

A.3.3 Вимоги до складу та параметрів технічних засобів

Для забезпечення підтримки роботи вебсервера та сервера БД необхідно надати апаратне забезпечення, що включає мінімально процесор, що має не менше ніж 4 ядра з тактовою частотою не менше 2,1 ГГц, підтриманий оперативною пам'яттю обсягом не менше 8 Гб та жорстким диском з доступними 10 Гб простору на ньому.

A.4 Порядок контролю та приймання

Для контролю виконання дипломної кваліфікаційної роботи повинен використовуватися відповідний календарний план, який затверджено викладачем у завданні на роботу. Приймання роботи відбувається на основі проведення процедури захисту роботи.

ДОДАТОК Б
Текст програми

Б.1 Текст файла CamperManufacturer.cs

```
using System;
using System.Collections.Generic;

namespace CamperRentalSoft.Data.Models
{
    public class CamperManufacturer
    {
        public int Id { get; set; }
        public string Fulltitle { get; set; }
        public string Site { get; set; }
        public string Fulldescription { get; set; }
        public virtual ICollection<CamperModel> CamperModels
        { get; set; }
    }
}
```

Б.2 Текст файла CamperModel.cs

```
using System;
using System.Collections.Generic;
using CamperRentalSoft.Data.Enums;

namespace CamperRentalSoft.Data.Models
{
    public class CamperModel
    {
        public int Id { get; set; }
        public int CamperManufacturerId { get; set; }
        public string Fulltitle { get; set; }
        public string Fulldescription { get; set; }
        public TransmT Transmissiontype { get; set; }
        public byte Sleepplaces { get; set; }
        public byte Seatplaces { get; set; }
        public float Freshwatervolume { get; set; }
        public float Graywatervolume { get; set; }
        public float Blackwatervolume { get; set; }
        public FuelT Fueltype { get; set; }
        public float Fuelvolume { get; set; }
        public ClassT Camperclass { get; set; }
        public float Length { get; set; }
    }
}
```

```

        public float Height { get; set; }
        public virtual CamperManufacturer CamperManufacturer {
get; set; }
        public virtual ICollection<Camper> Campers
        { get; set; }
    }
}

```

Б.3 Текст файла Camper.cs

```

using System;
using System.Collections.Generic;

namespace CamperRentalSoft.Data.Models
{
    public class Camper
    {
        public int Id { get; set; }
        public int CamperModelId { get; set; }
        public bool isShower { get; set; }
        public bool isSolarbatteries { get; set; }
        public byte Audioinputs { get; set; }
        public bool isMultimedia { get; set; }
        public bool isDiningtable { get; set; }
        public bool isKitchensink { get; set; }
        public bool isRefrigerator { get; set; }
        public bool isStove { get; set; }
        public bool isMicrowave { get; set; }
        public bool isOven { get; set; }
        public bool isBackcamera { get; set; }
        public bool isConditioner { get; set; }
        public bool isFan { get; set; }
        public bool isHeater { get; set; }
        public byte Twinbeds { get; set; }
        public byte Ordinarybeds { get; set; }
        public bool isToilet { get; set; }
        public bool isGenerator { get; set; }
        public bool isSatellite { get; set; }
        public bool isTV { get; set; }
        public bool isWifi { get; set; }
        public bool isWheelaccessed { get; set; }
        public byte Bikeplaces { get; set; }
        public bool isRoofplaces { get; set; }
    }
}

```

```

        public bool isWasher { get; set; }
        public virtual CamperModel CamperModel { get; set; }
        public virtual ICollection<Advertisement> Advertisements
        { get; set; }
    }
}

```

Б.4 Текст файла Advertisement.cs

```

using System;
using System.Collections.Generic;
using CamperRentalSoft.Data.Enums;

namespace CamperRentalSoft.Data.Models
{
    public class Advertisement
    {
        public int Id { get; set; }
        public int CamperId { get; set; }
        public int GuestProfileId { get; set; }
        public DateTime Producedyear { get; set; }
        public string Regnumber { get; set; }
        public string Color { get; set; }
        public string Fulldescription { get; set; }
        public float Averagescore { get; set; }
        public bool State { get; set; }
        public bool isPets { get; set; }
        public bool isParties { get; set; }
        public bool isSmoking { get; set; }
        public float Deposit { get; set; }
        public CancelT Cancelpolicy { get; set; }
        public virtual Camper Camper { get; set; }
        public virtual GuestProfile GuestProfile { get; set; }
        public virtual ICollection<CamperRental> CamperRentals
        { get; set; }
        public virtual ICollection<RentalRate> RentalRates
        { get; set; }
        public virtual ICollection<Proposition> Propositions
        { get; set; }
    }
}

```

Б.5 Текст файла CamperRental.cs

```

using System;
using System.Collections.Generic;
using CamperRentalSoft.Data.Enums;

namespace CamperRentalSoft.Data.Models
{
    public class CamperRental
    {
        public int Id { get; set; }
        public int AdvertisementId { get; set; }
        public int GuestProfileId { get; set; }
        public DateTime Rentaldate { get; set; }
        public DateTime Returndate { get; set; }
        public DateTime Plannedreturndate { get; set; }
        public DateTime Bookingdate { get; set; }
        public StateT State { get; set; }
        public string Review { get; set; }
        public byte Reviewscore { get; set; }
        public int RentalstationId { get; set; }
        public int ReturnstationId { get; set; }
        public float Price { get; set; }
        public byte Duration { get; set; }
        public string Fulldescription { get; set; }
        public byte Guests { get; set; }
        public string Guestreview { get; set; }
        public byte Guestscore { get; set; }
        public float Deposit { get; set; }
        public virtual Advertisement Advertisement { get; set; }
        public virtual Station Rentalstation { get; set; }
        public virtual Station Returnstation { get; set; }
        public virtual GuestProfile GuestProfile { get; set; }
    }
}

```

Б.6 Текст файла RentalRate.cs

```

using System;
using System.Collections.Generic;

namespace CamperRentalSoft.Data.Models
{

```

```

public class RentalRate
{
    public int Id { get; set; }
    public int AdvertisementId { get; set; }
    public DateTime Startdate { get; set; }
    public DateTime Enddate { get; set; }
    public byte Minimumdays { get; set; }
    public byte Maximumdays { get; set; }
    public float Dayprice { get; set; }
    public float Allowedduration { get; set; }
    public float Insuranceprice { get; set; }
    public virtual Advertisement Advertisement { get; set; }
}
}

```

Б.7 Текст файла Proposition.cs

```

using System;
using System.Collections.Generic;
using CamperRentalSoft.Data.Enums;

namespace CamperRentalSoft.Data.Models
{
    public class Proposition
    {
        public int Id { get; set; }
        public int AdvertisementId { get; set; }
        public DateTime Startdate { get; set; }
        public DateTime Enddate { get; set; }
        public int RentalstationId { get; set; }
        public int ReturnstationId { get; set; }
        public float Price { get; set; }
        public float Allowedduration { get; set; }
        public float Insuranceprice { get; set; }
        public bool isActive { get; set; }
        public virtual Advertisement Advertisement { get; set; }
        public virtual Station Rentalstation { get; set; }
        public virtual Station Returnstation { get; set; }
    }
}

```

Б.8 Текст файла AdvertisementService.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using CamperRentalSoft.Data;
using CamperRentalSoft.Data.Interfaces;
using CamperRentalSoft.Data.Models;
using Microsoft.EntityFrameworkCore;

namespace CamperRentalSoft.Service
{
    public class AdvertisementService: IAdvertisement
    {
        private readonly CRSDatabaseContext _crsdbc;

        public AdvertisementService (CRSDatabaseContext crsdbc)
        {
            _crsdbc = crsdbc;
        }

        public Advertisement GetById(int advid)
        {
            return _crsdbc.Advertisements.Where(adv => adv.Id ==
advid)
                .Include(adv => adv.Producedyear)
                .Include(adv => adv.Regnumber)
                .Include(adv => adv.Color)
                .Include(adv => adv.Fulldescription)
                .Include(adv => adv.Averagescore)
                .Include(adv => adv.isPets)
                .Include(adv => adv.isParties)
                .Include(adv => adv.isSmoking)
                .Include(adv => adv.Deposit)
                .Include(adv => adv.Cancelpolicy)
                .Include(adv => adv.Returnpolicy)
                .Include(adv => adv.GuestProfile)
                .Include(adv => adv.Camper)
                .Include(adv => adv.RentalRates)
                .Include(adv => adv.Propositions).ThenInclude(p
=> p.Rentalstation)
        }
    }
}

```

```

        .Include(adv => adv.Propositions).ThenInclude(p
=> p.Returnstation)
        .Include(adv =>
adv.CamperRentals).ThenInclude(cr => cr.GuestProfile)
        .Include(adv =>
adv.CamperRentals).ThenInclude(cr => cr.Rentalstation)
        .Include(adv =>
adv.CamperRentals).ThenInclude(cr => cr.Returnstation)
        .First();
    }

    public IEnumerable<Advertisement> GetAllAdvertisements()
    {
        return _crsdbc.Advertisements
            .Include(adv => adv.Producedyear)
            .Include(adv => adv.Regnumber)
            .Include(adv => adv.Color)
            .Include(adv => adv.Fulldescription)
            .Include(adv => adv.Averagescore)
            .Include(adv => adv.isPets)
            .Include(adv => adv.isParties)
            .Include(adv => adv.isSmoking)
            .Include(adv => adv.Deposit)
            .Include(adv => adv.Cancelpolicy)
            .Include(adv => adv.Returnpolicy)
            .Include(adv => adv.GuestProfile)
            .Include(adv => adv.Camper)
            .Include(adv => adv.RentalRates)
            .Include(adv => adv.Propositions).ThenInclude(p
=> p.Rentalstation)
            .Include(adv => adv.Propositions).ThenInclude(p
=> p.Returnstation)
            .Include(adv =>
adv.CamperRentals).ThenInclude(cr => cr.GuestProfile)
            .Include(adv =>
adv.CamperRentals).ThenInclude(cr => cr.Rentalstation)
            .Include(adv =>
adv.CamperRentals).ThenInclude(cr => cr.Returnstation);
    }

    public IEnumerable<Advertisement>
    GetAdvertisementsByOwner(int guest)
    {
        return _crsdbc.Advertisements.Where(adv =>
adv.GuestProfile.Id == guest)
    }

```

```

        .Include(adv => adv.Producedyear)
        .Include(adv => adv.Regnumber)
        .Include(adv => adv.Color)
        .Include(adv => adv.Fulldescription)
        .Include(adv => adv.Averagescore)
        .Include(adv => adv.isPets)
        .Include(adv => adv.isParties)
        .Include(adv => adv.isSmoking)
        .Include(adv => adv.Deposit)
        .Include(adv => adv.Cancelpolicy)
        .Include(adv => adv.Returnpolicy)
        .Include(adv => adv.Camper)
        .Include(adv => adv.RentalRates)
        .Include(adv => adv.Propositions).ThenInclude(p
=> p.Rentalstation)
        .Include(adv => adv.Propositions).ThenInclude(p
=> p.Returnstation)
        .Include(adv =>
adv.CamperRentals).ThenInclude(cr => cr.GuestProfile)
        .Include(adv =>
adv.CamperRentals).ThenInclude(cr => cr.Rentalstation)
        .Include(adv =>
adv.CamperRentals).ThenInclude(cr => cr.Returnstation);
    }

    public IEnumerable<CamperRental>
GetCamperRentalsForAdvertisement(int adv)
    {
        return _crsdbc.CamperRentals.Where(cr =>
cr.Advertisement.Id == adv);
    }

    public IEnumerable<Proposition>
GetPropositionsForAdvertisement(int adv)
    {
        return _crsdbc.Propositions.Where(p =>
p.Advertisement.Id == adv);
    }

    public IEnumerable<RentalRate>
GetRentalRatesForAdvertisement(int adv)
    {
        return _crsdbc.RentalRates.Where(rr =>
rr.Advertisement.Id == adv);
    }

```

```

public async Task Add(Advertisement adv)
{
    _crsdbc.Add(adv);
    await _crsdbc.SaveChangesAsync();
}

public async Task CalculateAverageAdvertisementScore(int
adv)
{
    var advert = GetById (adv);
    var advrentals = advert.CamperRentals.ToList();
    float avscore = 0;
    int num = advrentals.Count;
    foreach (var rent in advrentals)
    {
        avscore = avscore + rent.Reviewscore / num;
    }
    advert.Averagescore = avscore;
    _crsdbc.Update(advert);
    await _crsdbc.SaveChangesAsync();
}

public IEnumerable<Advertisement>
GetBestAdvertisements(int n)
{
    return GetAllAdvertisements().Where(adv => adv.State
== true).OrderByDescending(adv => adv.Averagescore).Take(n);
}

public Advertisement GetLastAdvertisement()
{
    return GetAllAdvertisements().Last();
}

public IEnumerable<Advertisement>
GetSearchedAdvertisements(AdvertisementSearch as)
{
    var advertisements = as.manufacturer == ""
        ? GetAllAdvertisements()
        : GetAllAdvertisements().Where(adv =>
adv.Camper.CamperModel.CamperManufacturer.Fulltitle.Contains(as.manufa
cturer));

    advertisements = as.model == ""
        ? advertisements

```

```

        : advertisements.Where(adv =>
adv.Camper.CamperModel.Fulltitle.Contains(as.model));
        advertisements = as.transmission == TransmT.None
        ? advertisements
        : advertisements.Where(adv =>
adv.Camper.CamperModel.Transmissiontype == as.transmission);
        advertisements = as.sleep == 0
        ? advertisements
        : advertisements.Where(adv =>
adv.Camper.CamperModel.Sleepplaces == as.sleep);
        advertisements = as.seat == 0
        ? advertisements
        : advertisements.Where(adv =>
adv.Camper.CamperModel.Seatplaces == as.seat);
        advertisements = as.minfreshwater <= 0.00001
        ? advertisements
        : advertisements.Where(adv =>
adv.Camper.CamperModel.Freshwatervolume >= as.minfreshwater);
        advertisements = as.mingraywater <= 0.00001
        ? advertisements
        : advertisements.Where(adv =>
adv.Camper.CamperModel.Graywatervolume >= as.mingraywater);
        advertisements = as.minblackwater <= 0.00001
        ? advertisements
        : advertisements.Where(adv =>
adv.Camper.CamperModel.Blackwatervolume >= as.minblackwater);
        advertisements = as.fuel == FuelT.None
        ? advertisements
        : advertisements.Where(adv =>
adv.Camper.CamperModel.Fueltype == as.fuel);
        advertisements = as.campclass == ClassT.None
        ? advertisements
        : advertisements.Where(adv =>
adv.Camper.CamperModel.Camperclass == as.campclass);
        advertisements = as.maxlength <= 0.00001
        ? advertisements
        : advertisements.Where(adv =>
adv.Camper.CamperModel.Length <= as.maxlength);
        advertisements = as.maxheight <= 0.00001
        ? advertisements
        : advertisements.Where(adv =>
adv.Camper.CamperModel.Height <= as.maxheight);
        advertisements = as.isShower == False
        ? advertisements

```

```

        : advertisements.Where(adv =>
adv.Camper.isShower == true);
        advertisements = as.isSolarbatteries == False
        ? advertisements
        : advertisements.Where(adv =>
adv.Camper.isSolarbatteries == true);
        advertisements = as.Audioinputs == 0
        ? advertisements
        : advertisements.Where(adv =>
adv.Camper.Audioinputs == as.Audioinputs);
        advertisements = as.isMultimedia == False
        ? advertisements
        : advertisements.Where(adv =>
adv.Camper.isMultimedia == true);
        advertisements = as.isDiningtable == False
        ? advertisements
        : advertisements.Where(adv =>
adv.Camper.isDiningtable == true);
        advertisements = as.isKitchensink == False
        ? advertisements
        : advertisements.Where(adv =>
adv.Camper.isKitchensink == true);
        advertisements = as.isRefrigerator == False
        ? advertisements
        : advertisements.Where(adv =>
adv.Camper.isRefrigerator == true);
        advertisements = as.isStove == False
        ? advertisements
        : advertisements.Where(adv => adv.Camper.isStove
== true);
        advertisements = as.isMicrowave == False
        ? advertisements
        : advertisements.Where(adv =>
adv.Camper.isMicrowave == true);
        advertisements = as.isOven == False
        ? advertisements
        : advertisements.Where(adv => adv.Camper.isOven
== true);
        advertisements = as.isBackcamera == False
        ? advertisements
        : advertisements.Where(adv =>
adv.Camper.isBackcamera == true);
        advertisements = as.isConditioner == False
        ? advertisements

```

```

        : advertisements.Where(adv =>
adv.Camper.isConditioner == true);
        advertisements = as.isFan == False
        ? advertisements
        : advertisements.Where(adv => adv.Camper.isFan
== true);
        advertisements = as.isHeater == False
        ? advertisements
        : advertisements.Where(adv =>
adv.Camper.isHeater == true);
        advertisements = as.Twinbeds == 0
        ? advertisements
        : advertisements.Where(adv =>
adv.Camper.Twinbeds == as.Twinbeds);
        advertisements = as.Ordinarybeds == 0
        ? advertisements
        : advertisements.Where(adv =>
adv.Camper.Ordinarybeds == as.Ordinarybeds);
        advertisements = as.isToilet == False
        ? advertisements
        : advertisements.Where(adv =>
adv.Camper.isToilet == true);
        advertisements = as.isGenerator == False
        ? advertisements
        : advertisements.Where(adv =>
adv.Camper.isGenerator == true);
        advertisements = as.isSatellite == False
        ? advertisements
        : advertisements.Where(adv =>
adv.Camper.isSatellite == true);
        advertisements = as.isTV == False
        ? advertisements
        : advertisements.Where(adv => adv.Camper.isTV ==
true);
        advertisements = as.isWifi == False
        ? advertisements
        : advertisements.Where(adv => adv.Camper.isWifi
== true);
        advertisements = as.isWheelaccessed == False
        ? advertisements
        : advertisements.Where(adv =>
adv.Camper.isWheelaccessed == true);
        advertisements = as.Bikeplaces == 0
        ? advertisements

```

```

        : advertisements.Where(adv =>
adv.Camper.Bikeplaces == as.Bikeplaces);
        advertisements = as.isRoofplaces == False
        ? advertisements
        : advertisements.Where(adv =>
adv.Camper.isRoofplaces == true);
        advertisements = as.isWasher == False
        ? advertisements
        : advertisements.Where(adv =>
adv.Camper.isWasher == true);
        advertisements = as.minscore == 0
        ? advertisements
        : advertisements.Where(adv => adv.Averagescore
>= as.minscore);
        advertisements = as.maxscore == 0
        ? advertisements
        : advertisements.Where(adv => adv.Averagescore
<= as.maxscore);
        advertisements = as.isPets == False
        ? advertisements
        : advertisements.Where(adv => adv.Camper.isPets
== true);
        advertisements = as.isParties == False
        ? advertisements
        : advertisements.Where(adv =>
adv.Camper.isParties == true);
        advertisements = as.isSmoking == False
        ? advertisements
        : advertisements.Where(adv =>
adv.Camper.isSmoking == true);
        advertisements = as.canpolice == CancelT.None
        ? advertisements
        : advertisements.Where(adv =>
adv.Camper.Cancelpolicy == as.canpolice);
        advertisements = as.retpolice == ReturnT.None
        ? advertisements
        : advertisements.Where(adv =>
adv.Camper.Returnpolicy == as.retpolice);
        advertresults = advertisements.ToList();
        var dtate1 = DateTime.Parse(ps.date1);
        var dtate2 = DateTime.Parse(ps.date2);
        foreach (var advert in advertisements.ToList())
        {

```

```

        crs = advert.CamperRentals.Where(cr =>
cr.Rentaldate >= as.date1).Where(cr => cr.Plannedreturndate <=
as.date2);

        if (crs)
        {
            advertresults.Remove(advert);
        }
        else
        {
            crs = advert.CamperRentals.Where(cr =>
cr.Plannedreturndate <= as.date1).order_by('-
Plannedreturndate').First();
            if (!(as.station1 != Station.None &&
crs.Returnstation == as.station1) || (as.station1 == Station.None &&
crs.Returnstation.City == as.city1)))
            {
                advertresults.Remove(advert);
            }
            else
            {
                dif = (date2 - date1).TotalDays;
                rt = advert.RentalRates.Where(r =>
r.Startdate <= date1 && r.Enddate >= date2 && r.Dayprice >=
as.minprice && r.Dayprice <= as.maxprice);
                rt = rt.Where(rr => rr.Minimumdays <=
dif && rr.Maximumdays >= dif)
                if (!rt)
                {
                    advertresults.Remove(advert);
                }
            }
        }
    }
    return advertisements.order_by('-Averagescore');
}

public Advertisement GetAdvertisementByRegnumber(string
regnumber, int guest)
{
    var advertisements = regnumber == None
    ? GetAdvertisementsByOwner (guest)
    : GetAdvertisementsByOwner(guest)
        .Where(adv => adv.Regnumber == regnumber)
        .First();
    return advertisements;
}

```

```

    }

    public IEnumerable<Advertisement>
    GetPopularOwnAdvertisements(int guest)
    {
        return GetAdvertisementsByOwner (guest)
            .Where(adv => adv.State == true)
            .OrderByDescending(adv =>
adv.CamperRentals.Count());
    }
}
}

```

Б.9 Текст файла PropositionService.cs

```

using System;
using System.Collections.Generic;
using System.Globalization;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using CamperRentalSoft.Data;
using CamperRentalSoft.Data.Interfaces;
using CamperRentalSoft.Data.Models;
using Microsoft.EntityFrameworkCore;

namespace CamperRentalSoft.Service
{
    public class PropositionService: IProposition
    {
        private readonly CRSDatabaseContext _crsdbc;

        public PropositionService (CRSDatabaseContext crsdbc)
        {
            _crsdbc = crsdbc;
        }

        public Proposition GetById(int pid)
        {
            return _crsdbc.Propositions.Where(p => p.Id == pid)
                .Include(p => p.Price)
                .Include(p => p.Startdate)
                .Include(p => p.Enddate)

```

```

        .Include(p => p.Allowedduration)
        .Include(p => p.Insuranceprice)
        .Include(p => p.isActive)
        .Include(p => p.Advertisement).ThenInclude(c =>
c.Camper).ThenInclude(cm => cm.CamperModel).ThenInclude(m =>
m.CamperManufacturer)
        .Include(p => p.Rentalstation).ThenInclude(s =>
s.City)
        .Include(p => p.Returnstation).ThenInclude(s =>
s.City)
        .First();
    }

    public IEnumerable<Advertisement> GetAllPropositions()
    {
        return _crsdbc.Propositions
            .Include(p => p.Price)
            .Include(p => p.Startdate)
            .Include(p => p.Enddate)
            .Include(p => p.Allowedduration)
            .Include(p => p.Insuranceprice)
            .Include(p => p.isActive)
            .Include(p => p.Advertisement).ThenInclude(c =>
c.Camper).ThenInclude(cm => cm.CamperModel).ThenInclude(m =>
m.CamperManufacturer)
            .Include(p => p.Rentalstation).ThenInclude(s =>
s.City)
            .Include(p => p.Returnstation).ThenInclude(s =>
s.City);
    }

    public async Task Add(Proposition prop)
    {
        _crsdbc.Add(prop);
        await _crsdbc.SaveChangesAsync();
    }

    public async Task UnActivateProposition(int prop)
    {
        var pr = GetById (prop);
        pr.isActive = false;
        _crsdbc.Update(pr);
        await _crsdbc.SaveChangesAsync();
    }

```

```

        public Proposition GetLastProposition()
        {
            return GetAllPropositions().Last();
        }

        public IEnumerable<Proposition>
GetSearchedPropositions(PropositionSearch ps)
        {
            var propositions = ps.maxprice == 0
                ? GetAllPropositions ()
                : GetAllPropositions ().Where(p => p.Price <=
ps.maxprice);

            propositions = ps.minprice == 0
                ? propositions
                : propositions.Where(p => p.Price <=
ps.minprice);

            if (ps.station1 == Station.None)
            {
                propositions = ps.city1 == City.None
                    ? propositions
                    : propositions.Where(p => p.Rentalstation.City
== ps.city1);
            }
            else
            {
                propositions = propositions.Where(p =>
p.Rentalstation == ps.station1);
            }
            if (ps.station2 == Station.None)
            {
                propositions = ps.city == City.None
                    ? propositions
                    : propositions.Where(p => p.Returnstation.City
== ps.city2);
            }
            else
            {
                propositions = propositions.Where(p =>
p.Returnstation == ps.station2);
            }
            var dtate1 = DateTime.Parse(ps.date1);
            var dtate2 = DateTime.Parse(ps.date2);
            propositions = ps.date1 == ""
                ? propositions

```

```

        : propositions.Where(p => p.Startdate <= dtate1
&& p.Enddate >= dtate1);
        propositions = ps.date2 == ""
        ? propositions
        : propositions.Where(p => p.Startdate <= dtate2
&& p.Enddate >= dtate2);
        var propositions = ps.manufacturer == ""
        ? propositions
        : propositions.Where(p =>
p.Advertisement.Camper.CamperModel.CamperManufacturer.Fulltitle.Contai
ns(ps.manufacturer));
        propositions = ps.model == ""
        ? propositions
        : propositions.Where(p =>
p.Advertisement.Camper.CamperModel.Contains(Fulltitle) == ps.model);
        propositions = ps.transmission == TransmT.None
        ? propositions
        : propositions.Where(p =>
p.Advertisement.Camper.CamperModel.Transmissiontype ==
ps.transmission);
        propositions = ps.sleep == 0
        ? propositions
        : propositions.Where(p =>
p.Advertisement.Camper.CamperModel.Sleepplaces == ps.sleep);
        propositions = ps.seat == 0
        ? propositions
        : propositions.Where(p =>
p.Advertisement.Camper.CamperModel.Seatplaces == ps.seat);
        propositions = ps.minfreshwater <= 0.00001
        ? propositions
        : propositions.Where(p =>
p.Advertisement.Camper.CamperModel.Freshwatervolume >=
ps.minfreshwater);
        propositions = ps.mingraywater <= 0.00001
        ? propositions
        : propositions.Where(p =>
p.Advertisement.Camper.CamperModel.Graywatervolume >=
ps.mingraywater);
        propositions = ps.minblackwater <= 0.00001
        ? propositions
        : propositions.Where(p =>
p.Advertisement.Camper.CamperModel.Blackwatervolume >=
ps.minblackwater);
        propositions = ps.fuel == FuelT.None
        ? propositions

```

```

        : propositions.Where(p =>
p.Advertisement.Camper.CamperModel.Fueltype == ps.fuel);
        propositions = ps.campclass == ClassT.None
        ? propositions
        : propositions.Where(p =>
p.Advertisement.Camper.CamperModel.Camperclass == ps.campclass);
        propositions = ps.maxlength <= 0.00001
        ? propositions
        : propositions.Where(p =>
p.Advertisement.Camper.CamperModel.Length <= ps.maxlength);
        propositions = ps.maxheight <= 0.00001
        ? propositions
        : propositions.Where(p =>
p.Advertisement.Camper.CamperModel.Height <= ps.maxheight);
        propositions = ps.isShower == False
        ? propositions
        : propositions.Where(p =>
p.Advertisement.Camper.isShower == true);
        propositions = ps.isSolarbatteries == False
        ? propositions
        : propositions.Where(p =>
p.Advertisement.Camper.isSolarbatteries == true);
        propositions = ps.Audioinputs == 0
        ? propositions
        : propositions.Where(p =>
p.Advertisement.Camper.Audioinputs == ps.Audioinputs);
        propositions = ps.isMultimedia == False
        ? propositions
        : propositions.Where(p =>
p.Advertisement.Camper.isMultimedia == true);
        propositions = ps.isDiningtable == False
        ? propositions
        : propositions.Where(p =>
p.Advertisement.Camper.isDiningtable == true);
        propositions = ps.isKitchensink == False
        ? propositions
        : propositions.Where(p =>
p.Advertisement.Camper.isKitchensink == true);
        propositions = ps.isRefrigerator == False
        ? propositions
        : propositions.Where(p =>
p.Advertisement.Camper.isRefrigerator == true);
        propositions = ps.isStove == False
        ? propositions

```

```

        : propositions.Where(p => p.Camper.isStove ==
true);
        propositions = ps.isMicrowave == False
        ? propositions
        : propositions.Where(p =>
p.Advertisement.Camper.isMicrowave == true);
        propositions = ps.isOven == False
        ? propositions
        : propositions.Where(p => p.Camper.isOven ==
true);
        propositions = ps.isBackcamera == False
        ? propositions
        : propositions.Where(p =>
p.Advertisement.Camper.isBackcamera == true);
        propositions = ps.isConditioner == False
        ? propositions
        : propositions.Where(p =>
p.Advertisement.Camper.isConditioner == true);
        propositions = ps.isFan == False
        ? propositions
        : propositions.Where(p => p.Camper.isFan ==
true);
        propositions = ps.isHeater == False
        ? propositions
        : propositions.Where(p =>
p.Advertisement.Camper.isHeater == true);
        propositions = ps.Twinbeds == 0
        ? propositions
        : propositions.Where(p =>
p.Advertisement.Camper.Twinbeds == ps.Twinbeds);
        propositions = ps.Ordinarybeds == 0
        ? propositions
        : propositions.Where(p =>
p.Advertisement.Camper.Ordinarybeds == ps.Ordinarybeds);
        propositions = ps.isToilet == False
        ? propositions
        : propositions.Where(p =>
p.Advertisement.Camper.isToilet == true);
        propositions = ps.isGenerator == False
        ? propositions
        : propositions.Where(p =>
p.Advertisement.Camper.isGenerator == true);
        propositions = ps.isSatellite == False
        ? propositions

```

```

        : propositions.Where(p =>
p.Advertisement.Camper.isSatellite == true);
        propositions = ps.isTV == False
        ? propositions
        : propositions.Where(p =>
p.Advertisement.Camper.isTV == true);
        propositions = ps.isWifi == False
        ? propositions
        : propositions.Where(p =>
p.Advertisement.Camper.isWifi == true);
        propositions = ps.isWheelaccessed == False
        ? propositions
        : propositions.Where(p =>
p.Advertisement.Camper.isWheelaccessed == true);
        propositions = ps.Bikeplaces == 0
        ? propositions
        : propositions.Where(p =>
p.Advertisement.Camper.Bikeplaces == ps.Bikeplaces);
        propositions = ps.isRoofplaces == False
        ? propositions
        : propositions.Where(p =>
p.Advertisement.Camper.isRoofplaces == true);
        propositions = ps.isWasher == False
        ? propositions
        : propositions.Where(p =>
p.Advertisement.Camper.isWasher == true);
        propositions = ps.minscore == 0
        ? propositions
        : propositions.Where(p =>
p.Advertisement.Averagescore >= ps.minscore);
        propositions = ps.maxscore == 0
        ? propositions
        : propositions.Where(p =>
p.Advertisement.Averagescore <= ps.maxscore);
        propositions = ps.isPets == False
        ? propositions
        : propositions.Where(p =>
p.Advertisement.Camper.isPets == true);
        propositions = ps.isParties == False
        ? propositions
        : propositions.Where(p =>
p.Advertisement.Camper.isParties == true);
        propositions = ps.isSmoking == False
        ? propositions

```

```

        : propositions.Where(p =>
p.Advertisement.Camper.isSmoking == true);
        propositions = ps.canpolice == CancelT.None
        ? propositions
        : propositions.Where(p =>
p.Advertisement.Camper.Cancelpolicy == ps.canpolice);
        propositions = ps.retpolice == ReturnT.None
        ? propositions
        : propositions.Where(p =>
p.Advertisement.Camper.Returnpolicy == ps.retpolice);
        return propositions;
    }

}
}

```

ДОДАТОК В
Слайди презентації

Національний університет «Запорізька політехніка»

кафедра програмних засобів

Дипломна кваліфікаційна робота

Програмне забезпечення прокату кемперів
Camper Rental Software

Виконала
ст. гр. КНТ-138

Клінкова А.Д.

Керівник
к.т.н., доц.

Льовкін В.М.

2022

Рисунок В.1 – Слайд № 1

Об'єкт роботи –
процес прокату кемперів.

Предмет роботи –
програмне забезпечення прокату кемперів.

Мета роботи –
розробити програмне забезпечення прокату кемперів
для підтримки роботи агентств з прокату та
індивідуальних власників кемперів.

Рисунок В.2 – Слайд № 2

Завдання дипломної кваліфікаційної роботи

- аналіз предметної області прокату кемперів;
- визначення вимог до програмного забезпечення прокату кемперів;
- проєктування програмного забезпечення прокату кемперів;
- реалізація і налагодження програмного забезпечення прокату кемперів.

3

Рисунок В.3 – Слайд № 3

Порівняння аналогів

Критерій	Outdoorsy	RVshare	Cruiseamerica	Розробка
Країна дії	США, Канада, Мексика	США	США	Без обмеження на країни
Поширення	Індивідуальні власники	Індивідуальні власники	Окреме агентство	Агентства, індивідуальні власники
Встановлення інтервалу цін для класів кемперів	Так	Так	Так	Ні
Ціноутворення	День, тиждень, місяць	День, тиждень, місяць	День, тиждень, місяць	Гнучке
Розширений пошук	Так	Так	Ні	Так
Відгуки за всіма оголошеннями власника	Так	Ні	Ні	Так
Відгуки про орендаря	Так	Так	Ні	Так
Відгуки про оголошення	Так	Так	Ні	Так

4

Рисунок В.4 – Слайд № 4

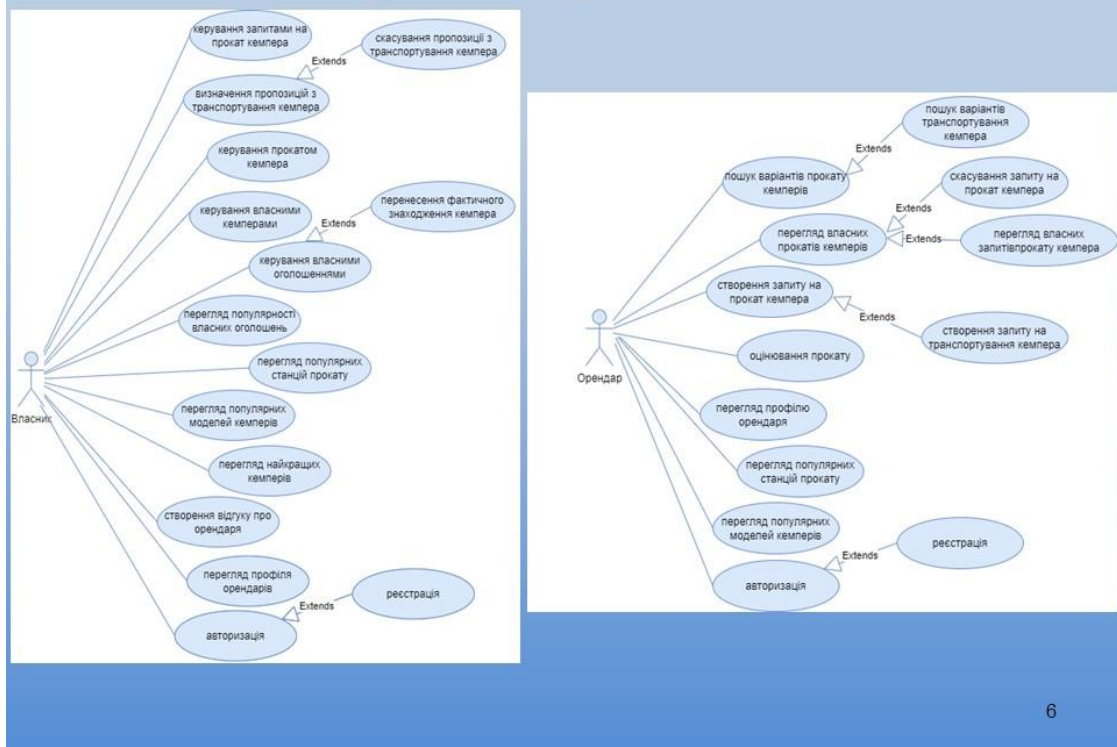
Вибір мови програмування

Критерій	C#	PHP
Придатність до розробки вебзастосунків	Так	Так
Придатність до потенційної розробки десктопних, мобільних застосунків	Так	Ні
Швидкість роботи застосунків	Більше	Менше
Вебфреймворк	ASP.NET	Laravel, Zend, Yii, Zend

5

Рисунок В.5 – Слайд № 5

Діаграми прецедентів



6

Рисунок В.6 – Слайд № 6

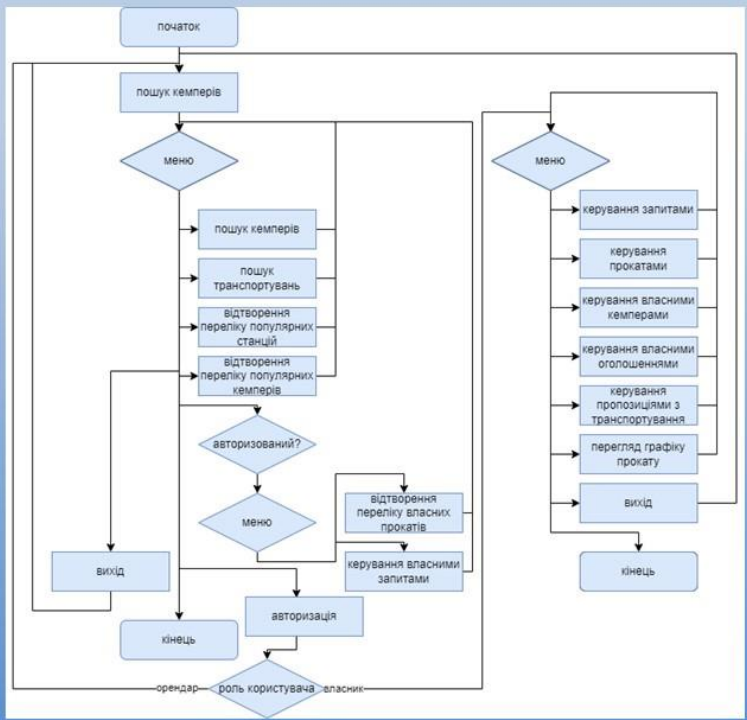


Рисунок В.7 – Слайд № 7



Рисунок В.8 – Слайд № 8

Схема функціонування програми



9

Рисунок В.9 – Слайд № 9

CamperRentalSoft

ПошукПрокатиСтанціїКемпериАвторизація

Місто

Львів

Станція

Аеропорт

Початок прокату

7/25/2022

Завершення прокату

7/30/2022

Мінімальна оцінка

4

Максимальна оцінка

5

Виробник кемпера

HYMER

Модель

Будь-яка

Клас кемпера

Клас А

Тип пального

дизель

Трансмісія

механіка

Спальних місць

4

Місць для сидіння

4

Міп обсяг питної води

10

Міп обсяг брудної води

100

Міп обсяг баку для відходів

100

Мак довжина кемпера, см

Введіть значення

Мак висота кемпера, см

Введіть значення

Кількість аудіо виходів

Введіть значення

Політика скасування

повне повернення

☒ душ

☐ сонячні батареї

☐ мультимедіа

☐ обідній стіл

☐ кухонна раковина

☐ холодильник

☐ кухонна плита

☐ мікрохвильовка

☐ духовка шафа

☐ камера заднього виду

☐ кондиціонер

☐ вентилятор

☐ обігрів

☐ туалет

☐ генератор

☐ супутникова антена

☐ телевізор

☐ wi-fi

☐ доступ осіб з інвалідністю

☐ місця на даху

☐ пральна машина

☐ домашні тварини

☐ паліна

☐ вечірки

Кількість двоспальних ліжок

Введіть значення

Кількість одномісних ліжок

Введіть значення

Кількість місць для велосипедів

Введіть значення

Шукати

Робота з програмою

10

Рисунок В.10 – Слайд № 10

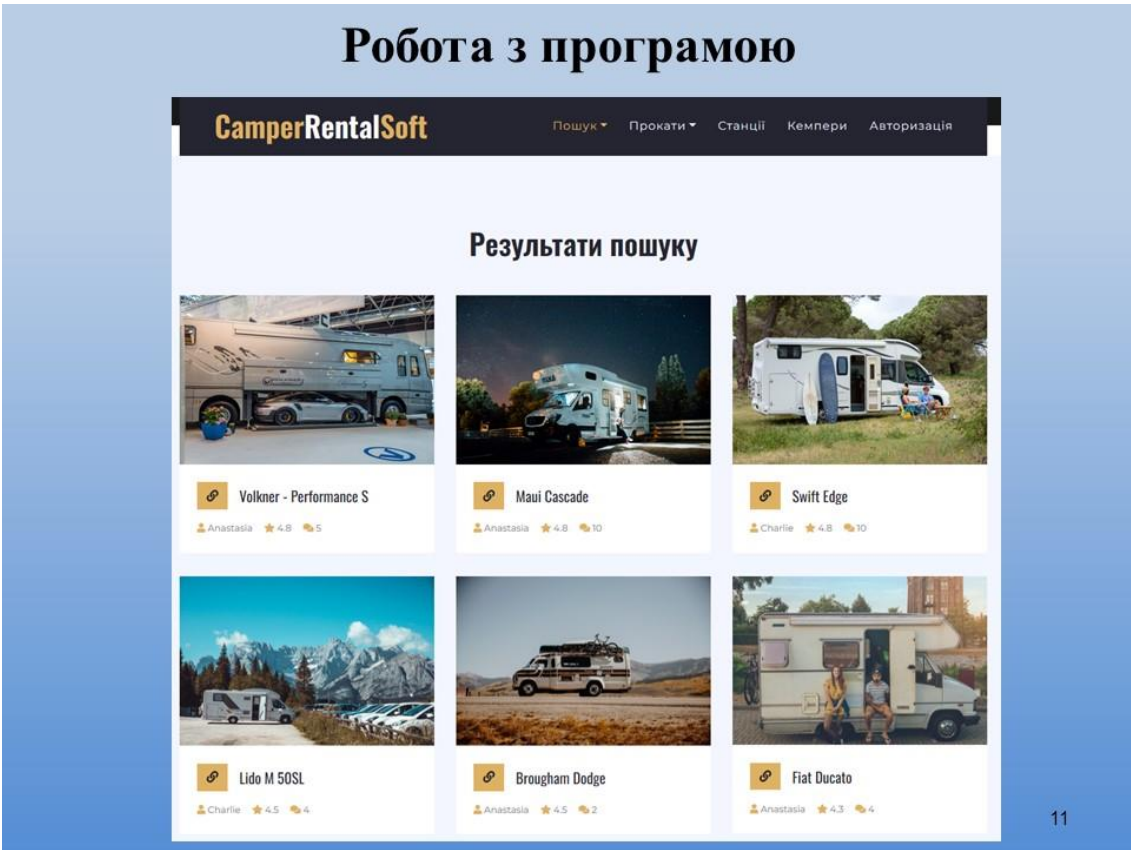


Рисунок В.11 – Слайд № 11

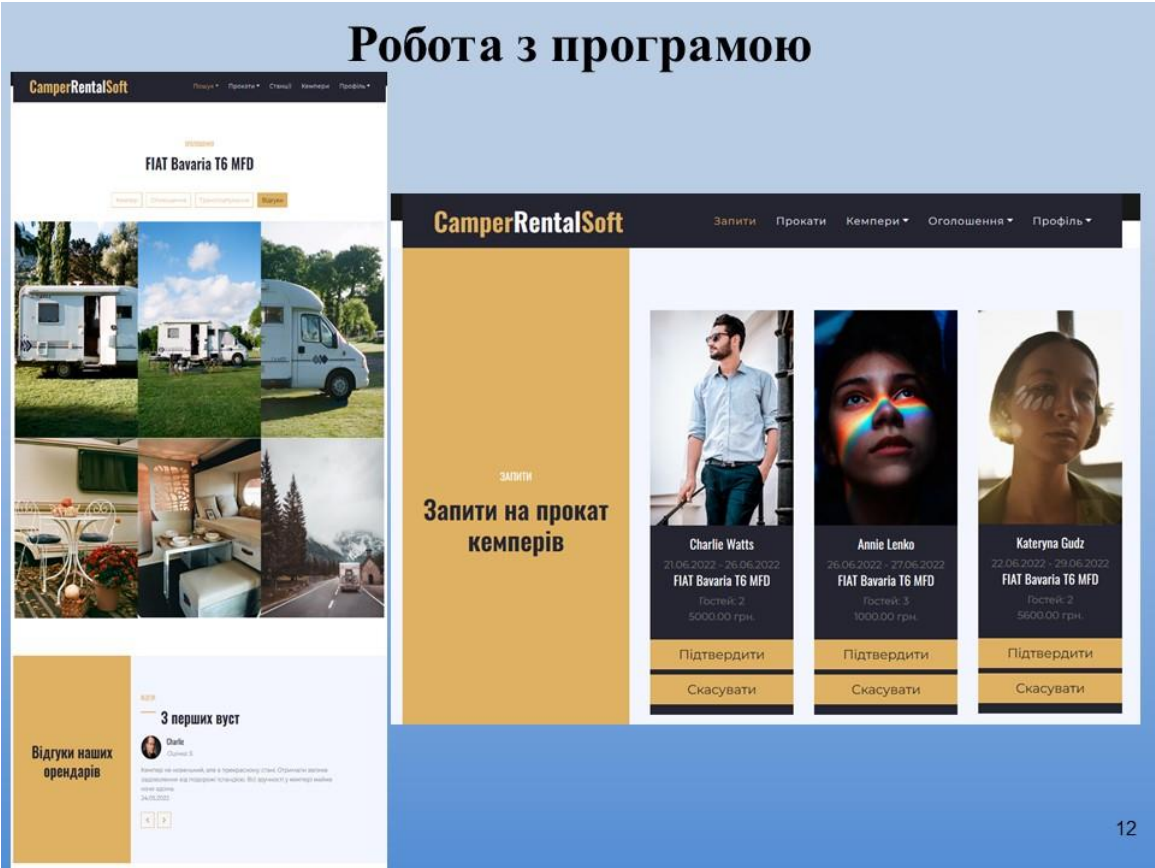


Рисунок В.12 – Слайд № 12

Тестування програми

Функція	Власник	Орендар		Адміністратор
		Неавторизований	Авторизований	
Пошук варіантів прокату кемперів		+	+	
Перегляд власних запитів прокату кемпера			+	
Перегляд власних прокатів кемперів			+	
Оцінювання прокату			+	
Пошук варіантів транспортування кемпера		+	+	
Перегляд запитів на прокат кемпера	+			
Підтвердження запита на прокат кемпера	+			
Скасування запита на прокат кемпера	+			
Перегляд власних вільних кемперів	+			
Визначення пропозицій з транспортування кемпера	+			
Скасування пропозицій з транспортування кемпера	+			
Перегляд графіка прокату заданого власного кемпера	+			
Перегляд популярності власних кемперів	+			
Створення відгуку про орендаря			+	
Перегляд популярних станцій прокату	+	+	+	
Перегляд популярних моделей кемперів	+	+	+	
Перегляд неоцінених власних прокатів			+	
Рєєстрація	+	+	+	
Авторизація	+	+	+	+
Перегляд профілю власника	+	+	+	
Перегляд профілю				
Видача кемпера в прокат	+			
Повернення кемпера з прокату	+			
Перенесення фактичного знаходження кемпера в задану станцію	+			
Додавання виробника кемпера				+
Редагування виробника кемпера				+
Додавання моделі кемпера				+
Редагування моделі кемпера				+
Додавання кемпера	+			
Редагування кемпера	+			
Додавання оголошення	+			
Редагування оголошення	+			
Визначення недоступних для прокату кемпера дат	+			
Пошук серед власних кемперів	+			
Створення запиту на транспортування кемпера			+	
Перегляд найкращих кемперів	+			
Архівувати власне оголошення	+			

13

Рисунок В.13 – Слайд № 13

Висновки

Розробка програмного забезпечення прокату кемперів надала можливість організувати платформу для підтримки роботи окремого агентства, що може бути організовано в різних країнах світу, в тому числі в декількох одночасно, з можливістю підтримки роботи одразу декількох агентств, а також створення оголошень від індивідуальних власників кемперів. Програма дозволяє виконувати пошук кемперів, бронювання прокату кемперів, керувати бронюваннями власних кемперів, створювати відгуки орендарю про власника, а власнику про орендаря.

Роботу розробленого програмного забезпечення було підтверджено за результатами проведеного тестування, коли було перевірено функціонування усіх визначених функцій програми.

14

Рисунок В.14 – Слайд № 14