Хмельницький національний університет
Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення

ДИПЛОМНИЙ ПРОЕКТ

Веб-орієнтована система для забезпечення роботи сервісу прокату автомобілів
Назва теми

Рівень вищої освіти Перший (бакалаврський)

Галузь знань 12 «Інформаційні технології»

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

Шифр ДПІПЗ.190149.01.04.ПЗ

Виконав студент III курсу група ІПЗс-19-1 _____ В.В. Карпович
Підпис Ініціали, прізвище

Керівник канд. пед. наук, доцент _____ О.Г. Онишко
Науковий ступінь, звання Підпис Ініціали, прізвище

Нормоконтролер канд. пед. наук, доцент _____ Н. І. Праворська
Підпис Ініціали, прізвище

До захисту допускаю:
Завідувач кафедри інженерії програмного забезпечення _____ Л. П. Бедратюк
Підпис Ініціали, прізвище

6 червня 2022 р.

Хмельницький 2022

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій
Кафедра Інженерії програмного забезпечення
Рівень вищої освіти Перший (бакалаврський)
Галузь знань 12 «Інформаційні технології»
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри ІПЗ
Л. П. Бедратюк
05 02 2022 р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ)

Карповичу Вячеславу Васильовичу
Прізвище, ім'я, по батькові студента

1. Тема проєкту (роботи) Веб-орієнтована система для забезпечення роботи сервісу прокату автомобілів
Керівник проєкту (роботи) Онишко Оксана Григорівна
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання
кандидат педагогічних наук, доцент
Затверджена наказом ректора університету від 01.03.2022 р. № 18
2. Строк подання студентом проєкту (роботи) на кафедру 01.06.2022 р.
3. Вихідні дані до проєкту (роботи) Матеріали переддипломної практики
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити) Дослідження предметної області та постановка задачі, проектування програмного забезпечення, програмна реалізація, тестування програмної системи
5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень) Презентаційні матеріали (слайди, 16 шт)

6. Консультанти розділів дипломного проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Праворська Н.І., доцент кафедри ІПЗ		
Антиплагіат	Гурман І.В., доцент кафедри ІПЗ		

7. Дата видачі завдання «_05_» __лютого_ 2022 р.

КАЛЕНДАРНИЙ ПЛАН

Назва етапів (розділів) дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітк
1 Ознайомлення з тематикою дипломного проектування (ДП), визначення та узгодження індивідуальних тем ДП	01.12 – 30.12.2021	
2 Дослідження предметної області, в якій планується використання програмного засобу (ПЗ), визначення задач та вимог, розробка технічного завдання	02.01 – 31.01.2022	
3 Проектування програмного забезпечення	01.02 – 28.02 2022	
4 Програмна реалізація	01.03 – 10.04.2022	
5 Тестування програмного забезпечення	11.04 – 30.04.2022	
6 Написання вступу, загальних висновків, оформлення джерел посилання та додатків. Оформлення пояснювальної записки ДП згідно вимог стандартів	01.05 – 25.05.2022	
7 Попередній захист ДП	травень 2022 (згідно графіка)	
8 Перевірка ДП на плагіат, нормоконтроль, отримання відгуків та рецензій. Брошурування (зшиття) пояснювальної записки	26.05 – 30.05.2022	
9 Підготовка до захисту та захист ДП	з 01.06.2022	

Студент _____ В.В. Карпович
Підпис Ініціали, прізвище

Керівник проекту (роботи) _____ О.Г. Онишко
Підпис Ініціали, прізвище

АНОТАЦІЯ

Тема дипломного проєкту: Веб-орієнтована система для забезпечення роботи сервісу прокату автомобілів.

Автор проєкту: Карпович В.В.

Керівник проєкту: Онишко О.Г.

Пояснювальна записка: 107 с., 43 рис., 1 табл., 7 дод., 19 джерел.

Графічна частина: 16 слайдів.

ВЕБ-ОРІЄНТОВАНА СИСТЕМА, СЕРВІС ПРОКАТУ АВТОМОБІЛІВ, САЙТ, МОБІЛЬНИЙ ДОДАТОК, PHP, MY SQL, RENT CAR.

Метою проєкту є розробка програмного забезпечення у вигляді веб-орієнтованої системи для забезпечення роботи сервісу прокату автомобілів, яка буде складатися з сайту та мобільного додатку, що буде сприяти підвищенню конкурентоспроможності сервісу на ринку та збільшенню привабливості для клієнтів.

У дипломному проєкті вивчена робота сервісу прокату автомобілів, проаналізоване наявне програмне забезпечення, що в даний час використовується сервісами з прокату автомобілів для взаємодії з клієнтами, сформовані вимоги до розробки програмного забезпечення. За допомогою UML моделювання здійснена формалізація описаної задачі та спроєктована веб-орієнтована система. Розроблена база даних. Вибрані технології для програмної реалізації спроєктованої системи та здійснена програмна реалізація. Розроблене програмне забезпечення було протестоване за допомогою автоматичних тестів та вручну.

Для програмної реалізації спроєктованої системи були вибрані наступні технології: мова структурованих запитів SQL, середовища розробки PhpMy Admin та Android Studio, фреймворк Yii2, мови програмування Php та Java, бібліотека Retrofit.

Практична значимість отриманих результатів це розроблена веб-орієнтована система, що складається з сайту та мобільного додатку. Система готова до впровадження і може використовуватися в сервісі прокату автомобілів.

01.06.2022
Дата

Підпис

ВІДОМІСТЬ ДОКУМЕНТІВ

№ рядка	Формат	Позначення документа	Найменування документа	К-сть аркушів	№ екз.	Примітка
			Текстові документи			
1	А4	ДПІПЗ.190149.01.04.ПЗ	Пояснювальна записка	107		
2	А4		Завдання на дипломний	1		
			проект			
3	А4		Анотація	1		
			Графічні документи			
4	А4		Презентаційні матеріали	16		

Змн.	Арк.	№ докум.	Підпис	Дата	ДПІПЗ.190149.01.04.ВД		
Виконав		Карпович В.В.		02.06	Веб-орієнтована система для забезпечення роботи сервісу прокату автомобілів	Літ.	Арк. 1 / Аркушів 1
Керівник		Онишко О.Г.		02.06			
Н. контр.		Праворська Н.І.		03.06	Відомість документів	ХНУ, ІПЗс-19-1	
Зав. каф.		Бедратюк Л.П.		6.06			

ЗМІСТ

ВСТУП

Прокат автомобілів це відомий та доволі розповсюджений вид бізнесу в світі. В той же час, розвиток нових технологій, розповсюдження гаджетів, перехід до цифрової економіки відкривають нові можливості як для власників такого бізнесу, так і для клієнтів.

Український ринок послуг з прокату автомобілів до війни розвивався дуже динамічно. На ринку були представлені послуги як для клієнтів, що хочуть орендувати автомобіль, так і для тих, хто хоче віддати свій автомобіль в оренду і отримувати за це гроші. Насправді, доволі багато власників автомобілів достатньо рідко ним користуються. При цьому, автомобіль, що стоїть в умовному «гаражі» також потребує вкладень і не маленьких. Це комунальні платежі за гараж або оплата паркування, страхові внески, технічне обслуговування тощо. Для відшкодування цих витрат є чудове рішення – здача автомобіля в оренду через управляючу компанію. Середній термін окупності оренди нових автомобілів в Києві до війни складав приблизно 2 роки. Це сприяло розвитку бізнесу з прокату автомобілів. Як і розвиток зовнішнього та внутрішнього туризму, підвищення життєвого рівня в країні та інші фактори.

За даними Асоціації Прокатних компаній в Україні налічувалося більше 200 прокатних компаній. А представити на сьогодні компанію, що працює з клієнтами без використання інформаційних технологій не можливо. Тому тема бакалаврської роботи є актуальною.

Метою дипломного проєктування є розробка програмного забезпечення у вигляді веб-орієнтованої системи для забезпечення роботи сервісу прокату автомобілів, яка буде складатися з сайту та мобільного додатку, що буде сприяти підвищенню конкурентоспроможності сервісу на ринку та збільшенню привабливості для клієнтів .

Об’єктом дослідження є веб-орієнтовані системи для забезпечення роботи сервісу прокату автомобілів.

Предметом дослідження є моделі та методи розробки сайтів та мобільних додатків.

Для досягнення поставленої мети були сформовані такі завдання:

– здійснити дослідження предметної області;

– проаналізувати наявне програмне забезпечення на предмет виявлення його переваг та недоліків;

– здійснити розробку структури веб-орієнтованої системи;

– розробити базу даних для зберігання інформації про прокатні автомобілі та клієнтів;

– розробити frontend та backend сайту веб-орієнтованої системи;

– створити мобільний додаток для клієнтів системи;

– провести тестування розробленої веб-орієнтованої системи забезпечення роботи сервісу з прокату автомобілів.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз та опис предметної області

Темою дипломного проєкту є розробка веб-орієнтованої системи для забезпечення роботи сервісу прокату автомобілів.

Прокат автомобілів це популярна послуга, що прийшла до нас з заходу і яка набирає все більшої популярності. Це пов’язане як з розвитком туризму в країні та збільшенням популярності України в світі, так і з тим, що темп життя в містах значно пришвидшився і для пересування по місту людям конче потрібен автомобіль. Громадський транспорт не завжди може допомогти, а виклик таксі це не завжди зручно та прийнятно по коштам. Тому, найкращим вибором для цілого ряду ситуацій, таких як: короткострокові подорожі, необхідність швидко вирішити справи в різних районах міста, зустріч людей в аеропортах тощо.

Прокат автомобілів без водія має цілий ряд переваг, серед яких:

– можливість вибору автомобіля, що буде відповідати ситуації: тобто автомобіль відповідної марки, класу;

– відсутність необхідності виконувати технічне обслуговування автомобіля та витрачати кошти на його утримання і при цьому мати впевненість в надійності та безпеці автомобіля ;

– висока мобільність в будь-яких ситуаціях.

Прокат автомобілів як послуга –це здача автомобіля в оренду без водія на певний період часу.

Історично зафіксована перша ситуація прокату автомобіля за винагороду відбулася в 1916 році. Власник невеликої автомайстерні в штаті Небраска (США) Джо Саундерс здав в прокат один з своїх автомобілів, обладнавши його лічильником пробігу у переднього лівого колеса. Ця послуга була затребуваною і в 1925 році Сандерсу належало 25 станцій оренди в різних штатах США.

Вартість прокату автомобіля складала 10 центів за мілю. Прибутковий бізнес став нерентабельним під час Великої депресії (1929-1933 рр.), що привело до банкротству компанії.

Інша, компанія яка розвивалася паралельно з компанією Сандерса була заснована Волтером Якобсом. Вона була більш успішною і перед початком Великої депресії вона була продана таксомоторній компанії Yellow Cab, власником якої був Джон Херц. В подальшому ця діяльність була виділена в окремий напрямок компанії Hertz. І сьогодні ця компанія є однією з найбільш крупних автопрокатних компаній світу і володіє більш ніж 5100 пунктами оренди автомобіля. Фактором, який сприяв успіху компанії був розрахунок на здачу автомобілів в оренду в аеропортах для ділових людей.

Закінчення Другої світової війни та зростання економіки позитивно вплинуло на розвиток бізнесу з прокату автомобілів. В той час були створені найбільші та найвідоміші компанії світу такі як Avis, National Car Rental, Dollar Trhifty та інші, які на сьогоднішній день мають представництва в сотнях країн.

Тобто прокат автомобілів, як бізнес, успішно розвивається в світі вже протягом більш ніж 100 років. Хоча для України це доволі новий вид бізнесу. Нове дихання цьому бізнесу надали нові інформаційні технології та зокрема Інтернет. З розвитком Інтернету орендувати автомобіль стало значно легше. На сьогоднішній день компанії з оренди автомобілів надають
послуги онлайн бронювання через сайти та вед-додатки, що допомагають знайти потрібний автомобіль за ціною та класом в визначеному місці та його забронювати.

Зараз в наше життя входить поняття кашерінгу. На відміну від звичайного прокату, це – вид короткострокової оренди, який оплачується не подобово або погодинно, а похвилинно. Це перспективна послуга з точки зору розвитку бізнесу.

Технологія роботи сервісу прокату автомобілів наступна. Коли до сервісу приходять нові автомобілі, то необхідно занести до бази даних наступну інформацію про кожен автомобіль (марка, модель, кольор та рік випуску). Після

цього розраховується ціна одного дня прокату автомобіля. При зверненні клієнта щодо прокату вибраного автомобіля готується договір оренди, до якого заносяться дані про клієнта. Після цього дані клієнта заносяться в базу даних, вибраний автомобіль вилучається зі списку доступних для прокату автомобілів, клієнт отримує чек оплати.

Організаційна структура сервісу з прокату автомобілів виглядає наступним чином (рис.1.1).

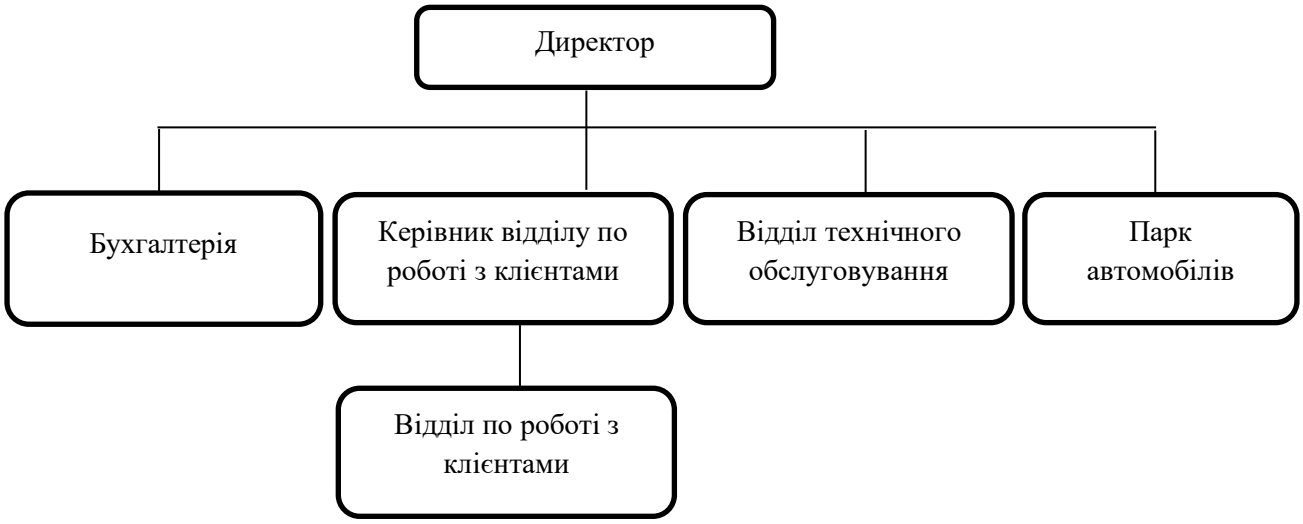


Рисунок 1.1 – Організаційна структура сервісу з прокату автомобілів

Наведена організаційна структура сервісу з прокату автомобілів обґрунтована розвитком компаній з прокату автомобілів, що здійснюють комп'ютеризований облік наявних автомобілів. Щоб здійснювати комп'ютеризований облік наявних автомобілів необхідно структурувати дані потрібні для обліку. Якщо цього не буде зроблено, то така ситуація призводить до проблеми втрати даних та великим часовим витратам на вибірку даних.

Процедура оренди автомобіля наступна. Клієнт має показати менеджеру сервісу прокату автомобілів паспорт та міжнародне водійське посвідчення. В деяких сервісах існує вимога щодо водійського стажу, який має бути не меншим за 2 роки. В деяких сервісах є вимога щодо наявності кредитної карти, але зазвичай оплата послуги може проводитися і готівкою, але має бути внесений

залог. У вартості орендної плати за прокат автомобілів враховуються такі моменти:

– величина пробігу автомобіля;

– необхідність доставки автомобіля;

– ремонт або заміна автомобіля в разі технічної несправності;

– повне страхове відшкодування в разі дорожньо-транспортної пригоди, що відбулася не з вини клієнта;

– страхове відшкодування, що буде покривати збитки нанесені автомобілю в ДТП, яке відбулося з вини клієнта;

– страхове відшкодування пасажирам (крім водія) в разі нещасних випадків;

– податки тощо.

Зазвичай автомобіль надається з повним баком. Повернення автомобіля в сервіс прокату також має відбуватися з повним баком. Автомобілі застраховані від всіх ризиків на умові повне КАСКО. У випадку ДТП, відповідальність клієнта складає розмір залогу, все інше має покривати страхова компанія.

В нашій країні прокат автомобілів – це достатньо новий вид бізнесу, який почав активно розвиватися в останнє десятиліття і був представлений переважно у великих містах. Тенденції бізнесу вказують на затребуваність послуги прокату автомобілів і подальший розвиток цієї послуги, в тому числі з активним використанням нових інформаційних технологій. А це, в свою чергу, буде потребувати розробки веб-додатків та веб-орієнтованих систем з надання таких послуг. Такі системи існують на ринку, але мають складну будову, часами обмежений функціонал, який може не реалізовувати всіх потрібних функцій. Тому, з врахуванням перспективності такого виду бізнесу як прокат автомобілів, є нагальна потреба в розробці веб-орієнтованої системи забезпечення роботи сервісу з прокату автомобілів.

1.2 Аналіз існуючих рішень для забезпечення роботи сервісів з прокату автомобілів

Зростання популярності сервісів з прокату автомобілів спричинило розробку програмних продуктів для забезпечення роботи таких сервісів. Такі програмні продукти можуть суттєво відрізнятися один від одного за існуючим функціоналом. Одні виконують виключно вузькоспеціалізовані задачі, інші представлені сайтами візитками де можна оглянути існуючий парк автомобілів, але не можна здійснити бронювання та виконувати інші потрібні функції.

Здійснимо огляд існуючих програмних продуктів. Визначимо їх переваги та недоліки для врахування при проєктуванні веб-орієнтованої системи забезпечення роботи сервісу прокату автомобілів.

Сервіс EuropCar – https://www.europcar.com/ [1]. Це міжнародна компанія з прокату автомобілів. Вказаний сервіс безкоштовний для користувачів. Ним можна користуватися як у вигляді веб-сайту, так і у вигляді мобільного додатку. Головна сторінка веб-додатку представлена на рисунку 1.2. Тут можна виконати наступні функції:

- пошук автомобілів;
- реєстрація та авторизація;
- бронювання автомобілів;
- перегляд вже створених бронювань.

Рисунок 1.3 – це скрін екрану мобільного додатку. Там наведений список автомобілів, що були відсортовані за певним фільтром та порядок виконання бронювання у мобільному додатку.

Програмне забезпечення сервісу EuropCar є безкоштовним для користувачів, натомість містить рекламу. Основна функція програмного забезпечення – це пошук автомобілів у потрібній локації. Програмне забезпечення дозволяє шукати та бронювати автомобілі у багатьох країнах Європи та Північної Америки.

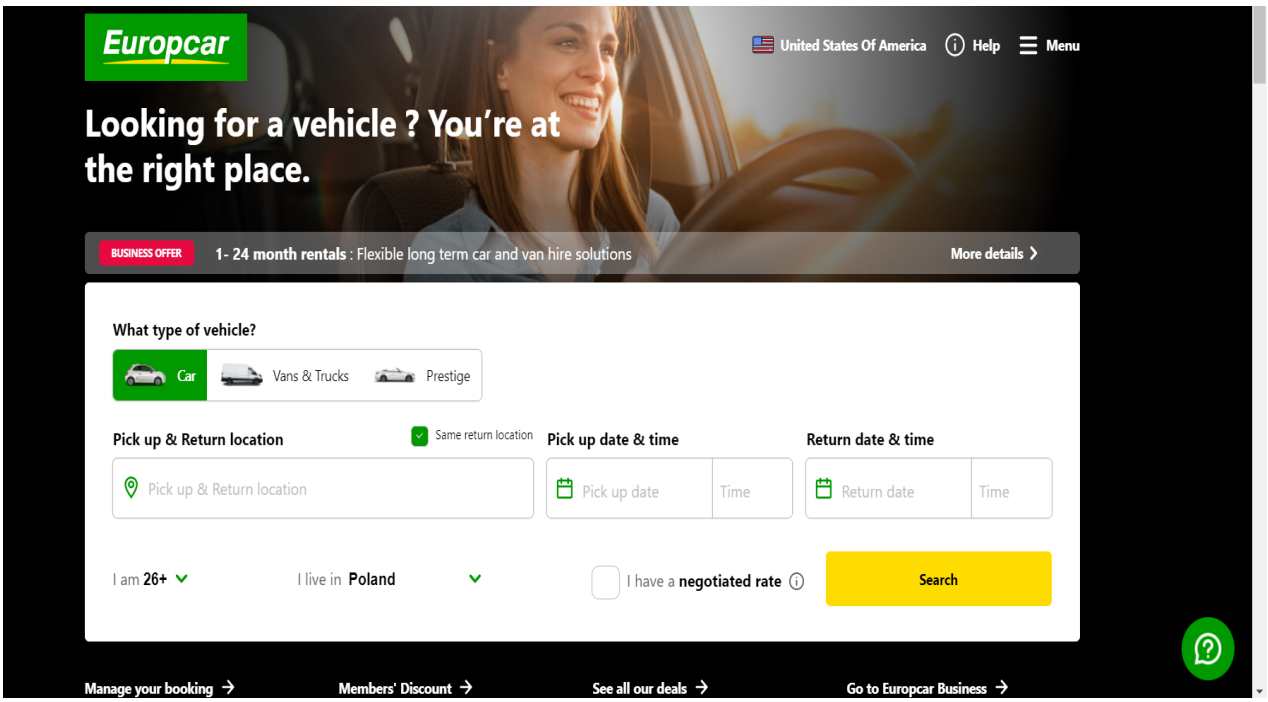


Рисунок 1.2 – Головна сторінка веб-сайту «EuropCar»

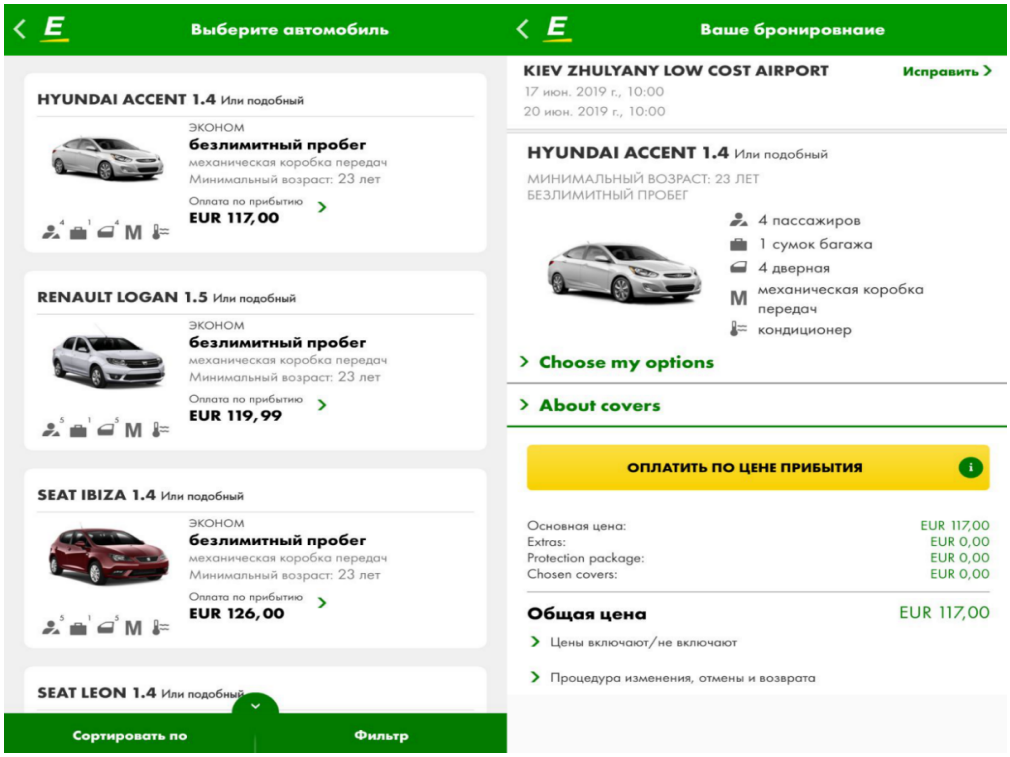


Рисунок 1.3 – Мобільний додаток сервісу «EuropCar»

До переваг веб-орієнтованої системи сервісу EuropCar можна віднести:

– автоматичне визначення місцезнаходження клієнта;

– наявність фільтрів для вибору автомобіля;

– оплата з можливістю використання широкого спектру платіжних систем;

– наявність додаткових опцій при здійсненні бронювання;

– наявність зручного пошуку автомобілів;

– технічна підтримка он-лайн.

До недоліків можна віднести наступне:

– процедура реєстрації складна та довготривала;

– повільна робота програмного забезпечення;

– мовна локалізація не включає українську мову;

– перевантаження інтерфейсу через додаткові вікна та рекламу;

– реєстрація можлива тільки на веб-сайті сервісу і не можлива через веб-додаток.

Веб-додаток «Navaran» – https://en.navaran.com/ [2]. Розроблений для використання на пристроях з операційною системою Android і призначений для бронювання автомобілів в країнах близького сходу, зокрема в Ірані.

Гарною функцією додатку є контроль статусу замовлення через використання електронної пошти. Головний екран веб-додатку «Navaran» наведений на рисунку 1.4. Скріншот виконання бронювання автомобіля представлений на рисунку 1.5.

До переваг веб-додатку належать:

– зручний інтерфейс додатку;

– можливість вибору точки видачі автомобіля;

– розмір додатку, що дозволяє його використовувати на різноманітних пристроях.

До недоліків додатку належать:

– відсутність кабінету користувача;

– відсутність інструкції користувача.

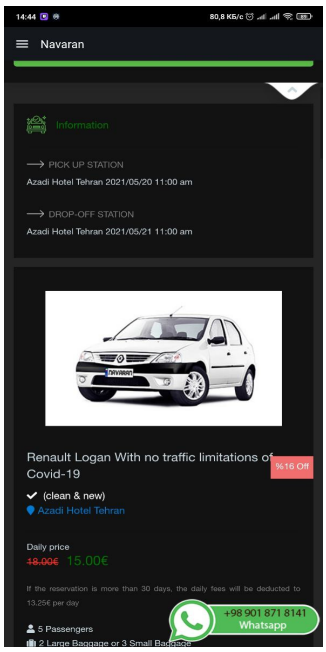


Рисунок 1.4 – Головне вікно мобільного додатку «Navaran»

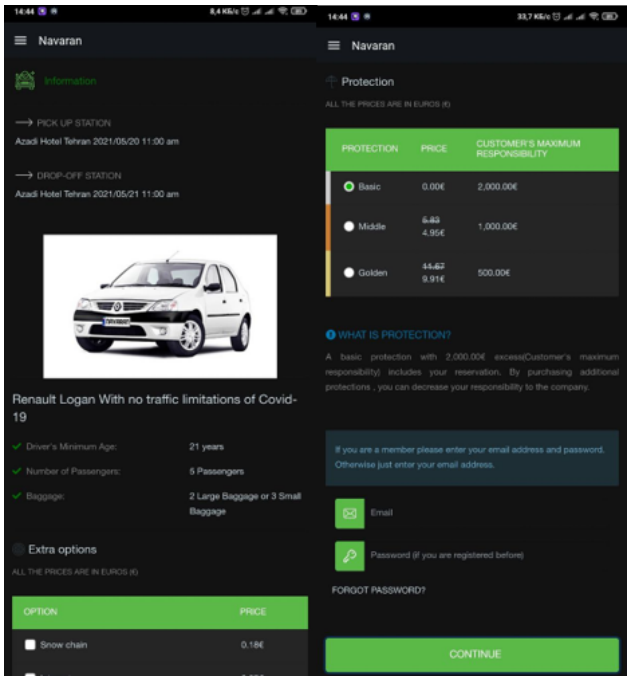


Рисунок 1.5 – Бронювання автомобіля за допомогою мобільного додатку «Navaran»

Веб-додаток «Seven Cars» – https://7cars.com.ua/ [3]. Веб-додаток для замовлення автомобілів в Україні. Додаток достатньо добре зроблений і може бути прикладом для розроблення додатку. До переваг можна віднести:

– сучасний інтерфейс веб-додатку;

– наявність особистого кабінету;

– наявність різноманітних фільтрів.

До недоліків даного додатку можна віднести перенасиченість інформацією і складність бронювання автомобілів. На рисунку 1.6 представлене головне вікно веб-додатку, на рисунку 1.7 – форма бронювання, на рисунку 1.8 – встановлення фільтрів для пошуку автомобіля.

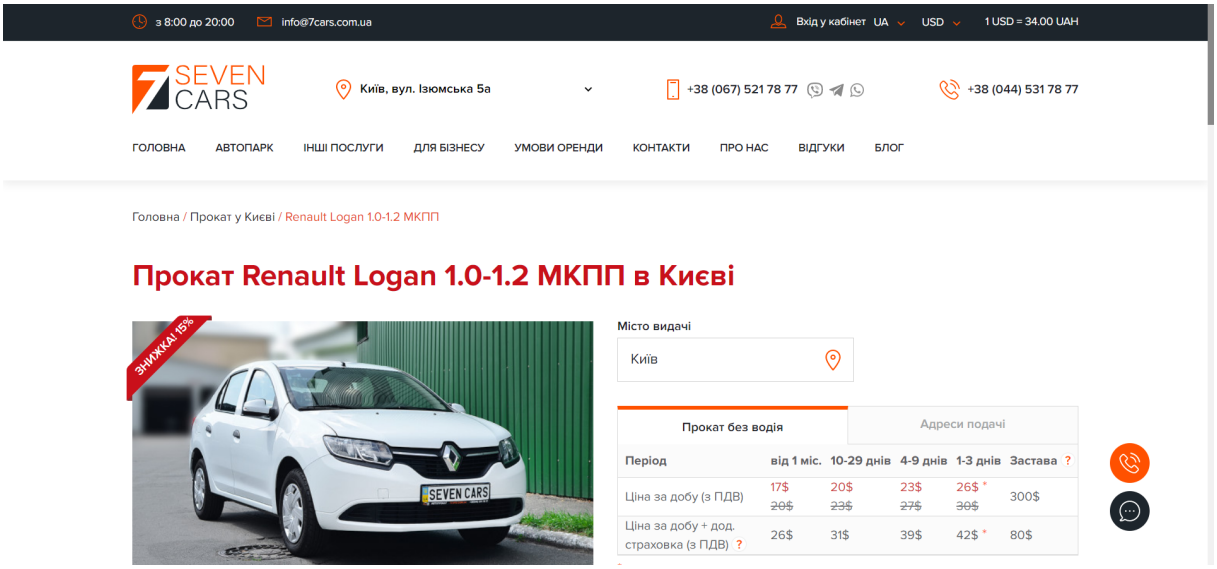


Рисунок 1.6 – Вікно веб-додатку «Seven Cars»

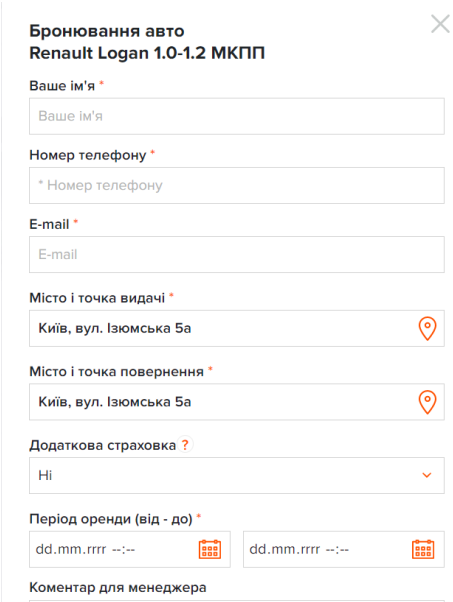


Рисунок 1.7 – Форма для бронювання автомобілів за допомогою веб-додатку «Seven Cars»

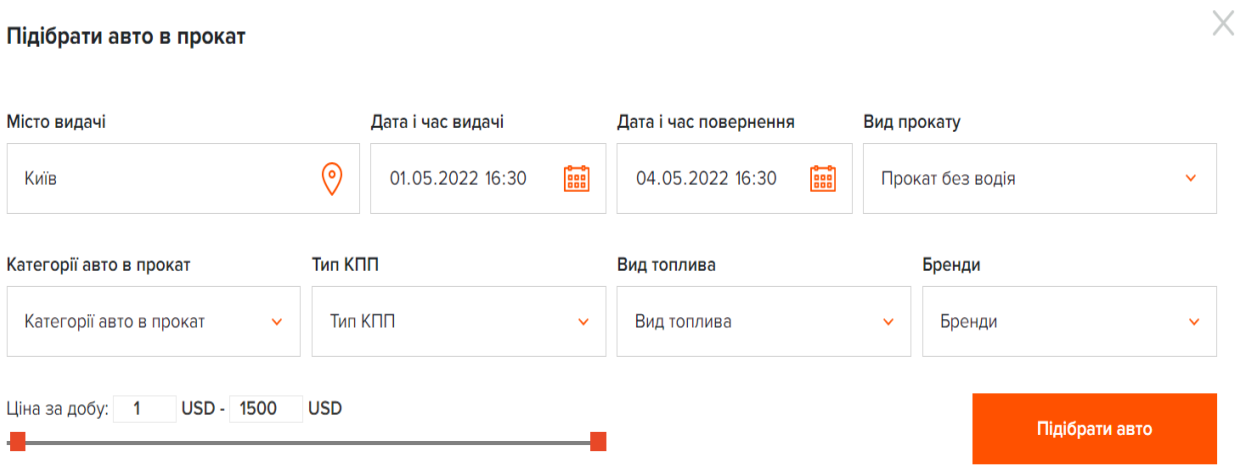


Рисунок 1.8 – Форма для вибору та фільтрації автомобілів за допомогою веб-додатку «Seven Cars»

Веб-додаток «BLS» – https://bls.ua/ua [4]. Компанія BLS одна з найстаріших сервісів з прокату автомобілів. Була заснована у 2007 році. Починала працювати з прокату автомобілів в кількості 7 штук. До 2008 році автопарк зріс втричі. Компанія виступила з ініціативою заснування Асоціації Прокатних компаній, яка працює до сьогодні і об'єднує близько 200 компаній з автопрокату.

Сьогодні компанія BLS – це компанія яка надає повний спектр послуг з оренди автомобілів з європейським рівнем якості обслуговування і європейськими стандартами, що означає автомобілі до трьох років для прокату, повне страхування, цілодобовий Call Center 24/7 для клієнтів, а також Road Assistance 24/7. Компанія співпрацює з міжнародними системами бронювання для комфорту наших клієнтів.

В 2015 році компанія стала піонером у прокаті електромобілів, таких як: Nissan Leaf, Smart, Tesla S, BMW I3. В 2017 році компанія відкрила нове направлення в сфері прокату – мотопрокат. На рисунку 1.9 представлено вікно веб-додатку сервісу «BLS».

Переваги веб-додатку сервісу «BLS»:

– можливість оплати прокату в різних валютах;

– форма бронювання автомобіля на головній сторінці додатку;
– он-лайн підтримка бронювання;
– зручний інтерфейс.

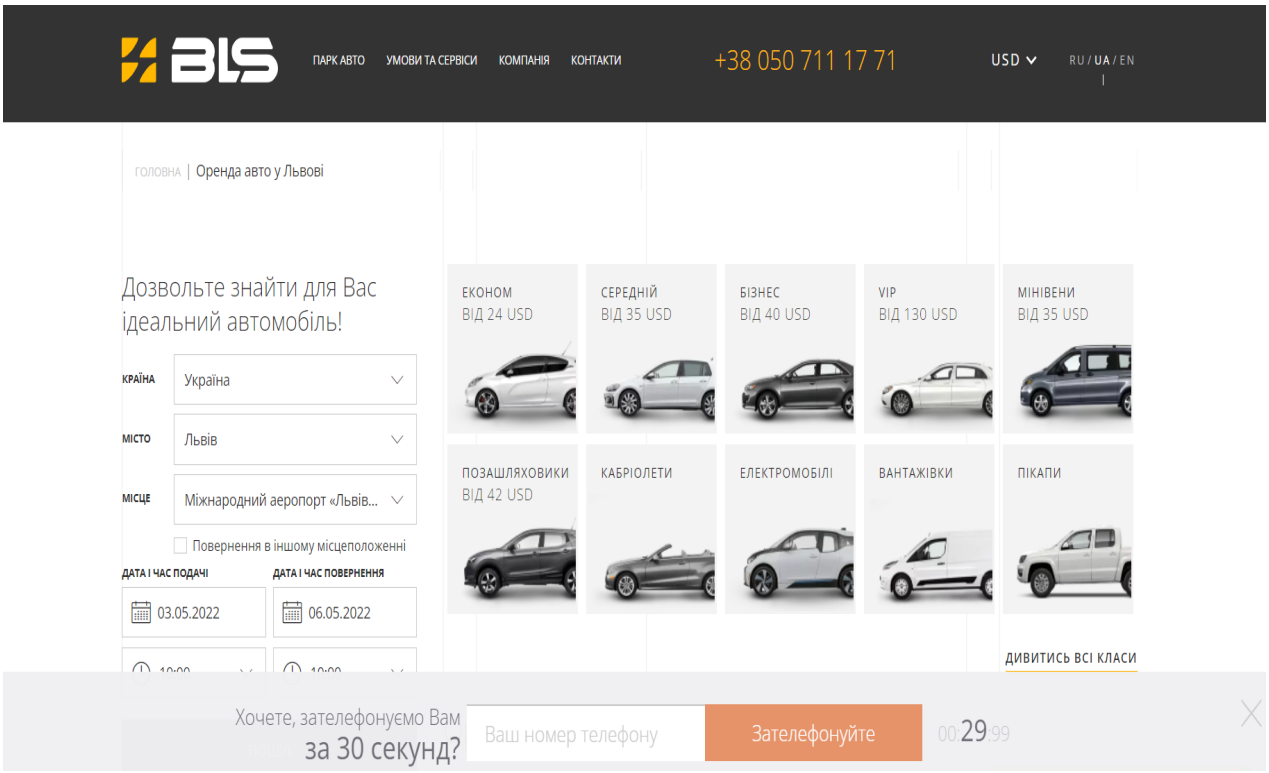


Рисунок 1.9 – Вікно веб-додатку сервісу «BLS»

Ще один приклад веб-додатку це веб-додаток компанії «Race» – https://www.race.net.ua/ [5]. Компанія Race почала працювати на ринку прокату автомобілів у 2016 році. В обласних центрах та великих містах України розташовані офіси компанії. Організаційна структура компанії складається з підрозділів бронювання, виконання замовлень, технічного обслуговування і багато іншого. Компанія подбала про кращі види страхового відшкодування для клієнтів і готова забезпечити їх автомобілем 24 години на добу, 365 днів у тиждень. На рисунку 1.10 представлена форма вікна веб-додатку сервісу «Race». На рисунку 1.11 – форма бронювання автомобіля через веб-додаток.

.

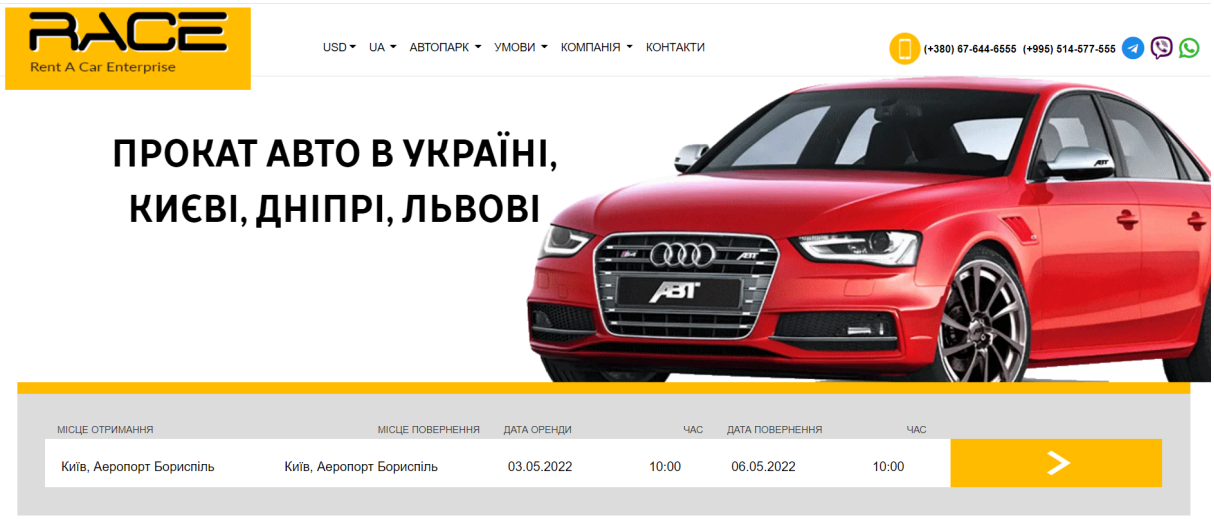


Рисунок 1.10 – Вікно веб-додатку сервісу «Race»

Процес бронювання відбувається в три кроки: спочатку потрібно вибрати потрібний за ціною та параметрами автомобіль, далі є можливість вибору додаткових опцій і тільки після цього здійснюється оформлення замовлення. Недоліком сервісу є відсутність особистого кабінету та онлайн підтримки при бронюванні автомобіля.

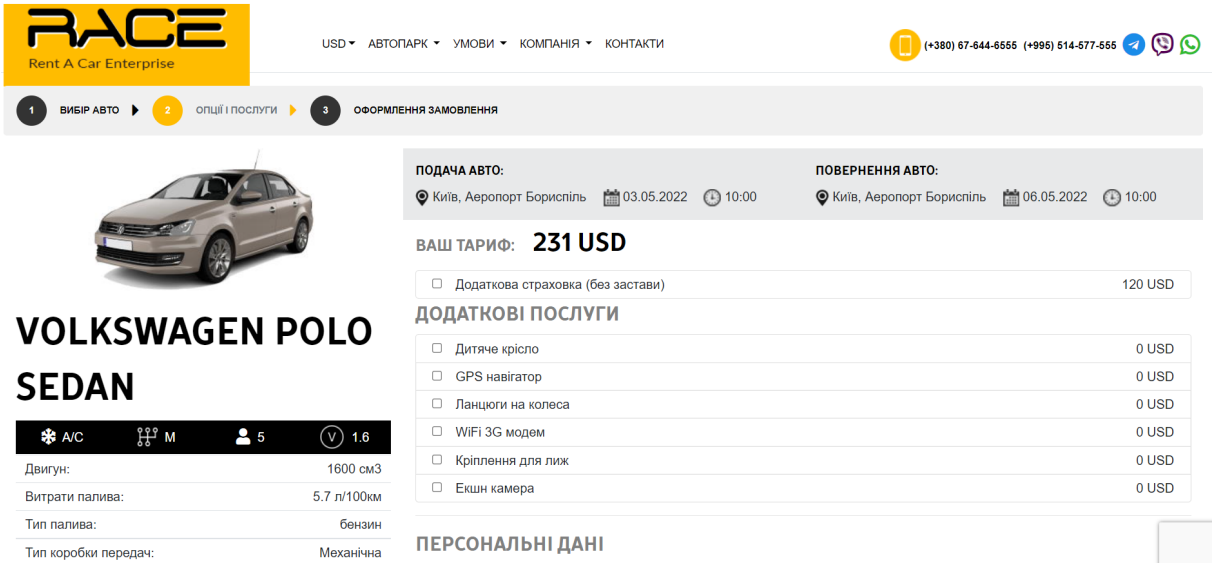


Рисунок 1.11 – Процес бронювання автомобіля за допомогою веб-додатку сервісу «Race»

Проведений аналіз аналогів веб-додатків сервісів з прокату автомобілів, визначенні їх можливості, функціонал, переваги та недоліків. Згрупуємо викладену раніше інформацію у порівняльну таблицю (табл. 1.1).

Вся інформація отримана при проведенні аналізу аналогів веб-додатків сервісів з прокату автомобілів буде основою для визначення функціоналу в розроблюваній веб-орієнтованій системі для забезпечення роботу сервісу з прокату автомобілів. Недоліки, проаналізованих веб-додатків, також будуть взяті до уваги з метою розробки зручної для клієнта веб-орієнтованої системи.

Таблиця 1.1 – Порівняльні характеристики веб-додатків сервісів прокату автомобілів

Характеристика	EuropCar	Navaran	Seven Cars	BLS	Race
Зручність інтерфейсу	+	+	+	+	+
Наявність мобільного додатку	+	+	-	+	-
Наявність веб-додатку	+	-	+	+	+
Вибір автомобіля за допомогою фільтрації	+	+	+	+	+
Бронювання он-лайн	+	+	+	+	+
Можливість написання відгуку для користувачів	-	+	+	+	+
Наявність онлайн підтримки бронювання	+	-	+	+	-

Після проведеного аналізу перейдемо до формування вимог для веб-орієнтованої системи для забезпечення сервісу прокату автомобілів.

1.3 Визначення вимог до веб-орієнтованої системи для забезпечення роботи сервісу прокату автомобілів

Веб-орієнтована система для забезпечення роботи сервісу прокату автомобілів призначена для виконання процедур пошуку, фільтрації та бронювання автомобілів, що виконуються через мережу Інтернет. Робота через мережу Інтернет дозволяє користуватися системою з будь-якого місця знаходження клієнта. Мінімізація людського фактору при проведенні вибору та бронювання автомобіля дозволяє виконувати бронювання незалежно від часу доби та створювати бронювання в дорозі або з дому.

Система призначена для власників сервісів з прокату автомобілів та має надавати користувачам наступні можливості:

– користувач має отримувати автоматичне підтвердження в разі виконання бронювання;

– має виконуватися автоматична перевірка здійснення оплати;

– наявність каталогу автомобілів;

– можливість налаштування та оновлення даних;

– мінімізація впливу менеджера по роботі з клієнтами під час виконання бронювання;

– гнучке налаштування системи під клієнта;

– полегшена процедура реєстрації та бронювання автомобілів;

– блокування ненадійних клієнтів;

– виконання необхідних розрахункових операцій під час бронювання;

– простота та зручність у використанні як для адміністратора система, так і для користувачів.

Опишемо алгоритм, що буде забезпечувати роботу сервісу прокату автомобілів через веб-орієнтовану системи.

Першим кроком до використання системи клієнтом є створення облікового запису нового користувача. Клієнт має зареєструватися в системі та

ввести дані в кабінеті свого облікового запису. Обліковий запис використовується багато разів.

Адміністратор системи має перевірити коректність усіх введених користувачем даних та надає дозвіл або заборону для проведення бронювання даному клієнту.

Після успішної реєстрації в системі користувач отримує змогу здійснювати процедури пошуку та підбору автомобілів.

Після підбору автомобіля користувач через форму бронювання виконує бронювання вибраного автомобілю. Тоді потрібно виконати оплату або он-лайн, або безпосередньо в офісі сервісу з прокату автомобіля.

Після виконання вище зазначених кроків, клієнт приїжджає на місце знаходження автомобілів і підписує умову на бронювання автомобіля. По закінченню терміну бронювання клієнт повертає автомобіль на його місце знаходження.

Веб-додаток, розроблений у відповідності до наведеного вище алгоритму, можна налаштувати під вимоги конкретного сервісу з прокату автомобілів. Процедури пошуку, фільтрації, бронювання та оплати за бронювання можуть проводитися без участі менеджера по роботі з клієнтами.

Отже, проаналізувавши предметну область, здійснивши аналіз аналогів розроблюваної веб-орієнтованої системи, вибудувавши алгоритм роботи системи були поставлені наступні завдання:

– здійснити розробку структури веб-орієнтованої системи;

– розробити базу даних для зберігання інформації про прокатні автомобілі та клієнтів;

– розробити frontend та backend сайту веб-орієнтованої частини;

– створити мобільний додаток для клієнтів системи;

– провести тестування розробленої веб-орієнтованої системи забезпечення роботи сервісу з прокату автомобілів.

Технічне завдання на розробку веб-орієнтованої системи для забезпечення роботи сервісу прокату автомобілів наведене в додатку А.

2. ПРОЄКТУВАННЯ ВЕБ-ОРІЄНТОВАНОЇ СИСТЕМИ ДЛЯ ЗАБЕЗПЕЧЕННЯ РОБОТИ СЕРВІСУ З ПРОКАТУ АВТОМОБІЛІВ

2.1 Формалізація задачі забезпечення роботи сервісу з прокату автомобілів

Метою формалізації – задокументувати процеси та явища предметної області, пов’язати між собою різноманітні процеси, що відбуваються в процесі роботи сервісу з прокату автомобілів для забезпечення можливості автоматизації цих процесів. У якості інструмента для формалізації задачі використаємо UML моделювання [6-8].

В процесі аналізу предметної області було визначено два типи користувачів системи: прості користувачі (клієнти) та адміністратори. Кожен з визначених типів користувачів має різні задачі, а відповідно різний функціонал системи та права.

Прості користувачі (клієнти) можуть:

- реєструватися в системі;
- авторизуватися в системі;
- здійснювати перегляд наявних в сервісі автомобілів;
- бронювати автомобілі;
- відслідковувати історію бронювань.

Адміністратори мають інші функції:

- введення та редагування даних про автомобілі;
- перегляд здійснених клієнтами бронювань;
- редагування статусу бронювань клієнтів;
- блокування користувачів.

Для опису взаємодії клієнтів та адміністраторів з системою та можливі варіанти їх діяльності використано діаграму варіантів використання Use Case (рисунок 2.1).

Діаграма класів Class Diagram, використовується для статичного представлення структури моделі та відображає наступні елементи: класи, типи даних, їх зміст та відношення. Діаграма класів наведена на рисунку 2.2.

Рисунок 2.1 – Діаграма варіантів використання

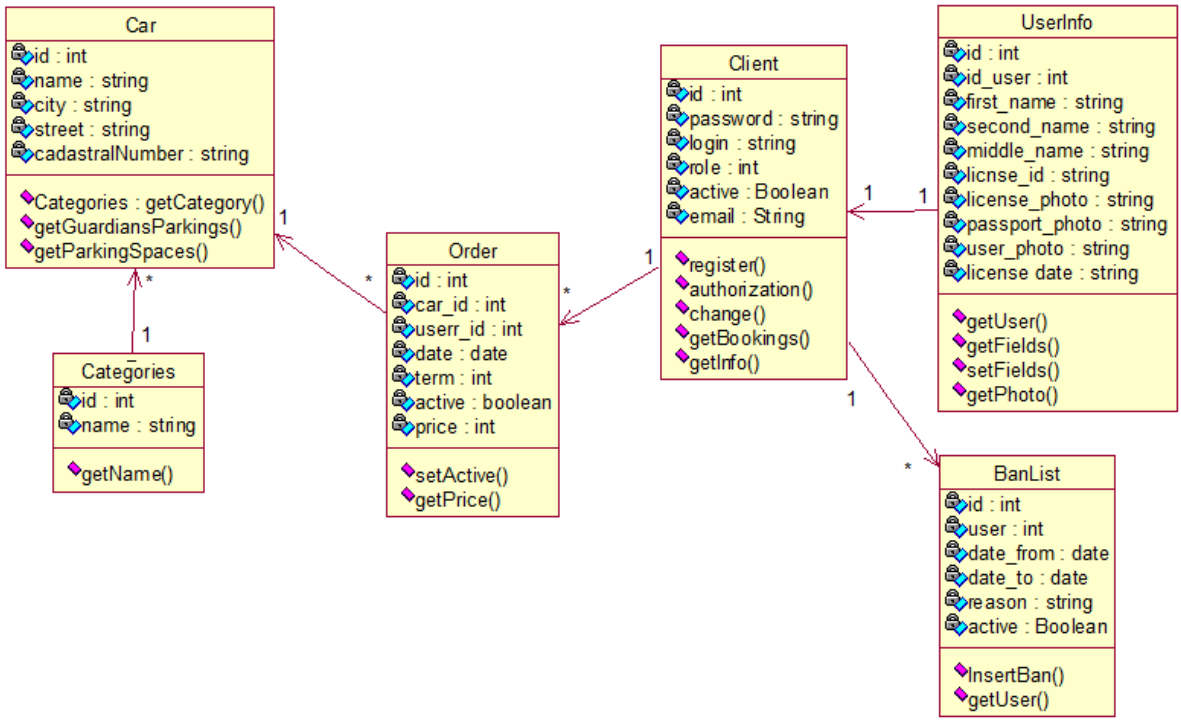


Рисунок 2.2 – Діаграма класів

З метою відображення послідовності роботи системи та взаємодії об’єктів використовується так тип діаграм, як діаграма послідовності Sequence Diagram. Діаграма послідовності відображає взаємодію об’єктів системи в динаміці, в часі. При цьому інформація приймає вигляд повідомлень, а під взаємодією об’єктів розуміємо обмін цими повідомленнями в рамках сценарію. Для даної задачі створено дві діаграми послідовностей – для роботи клієнта та для адміністратора (додаток Б).

Діаграма співпраці collaboration diagram відображає потік подій під час певного сценарію (варіанту використання). На діаграмі співпраці представлена та сама інформація, що є на діаграмі послідовності, але діаграма співпраці по іншому описує потік подій і надає представлення про зв’язки, що існують між об’єктами. Було створено два варіанті діаграми співпраці (додаток В). Перша показує роботу клієнта з системою, а друга – роботу адміністратора.

Діаграма станів State Machine diagram відображає різні стани об’єкта під час його існування та події, які призводять до переходу об’єкта з одного стану в інший. Для клієнтської частина діаграма показує всі можливі активності, і

описує дії, що виконуються при переході на визначену активність або під час перебування на ній. Так, для перегляду історії бронювань клієнт має перейти у нове вікно, куди програма має завантажити всі записи про попередні бронювання клієнта. Діаграми станів для мобільного додатку та адміністративної системи зображені в додатку Г.

Діаграма активності Activity Diagram призначена для опису динамічних аспектів поведінки системи у вигляді схеми, що відображає бізнес-процеси, логіку процедур та потоки робіт, тобто логіку поведінки системи. Діаграма активності для веб-орієнтованої системи для забезпечення роботи сервісу прокату автомобілів наведена в додатку Д.

Діаграма компонентів Component Diagram, яка наведена в додатку Е, відображає компоненти програмного забезпечення, залежності та зв’язки між ними.

Отже, формалізація задачі завершена і можна переходити до проєктування архітектури веб-орієнтованої системи.

2.2 Проєктування архітектури веб-орієнтованої системи

Веб-орієнтована система для забезпечення роботи сервісу прокату автомобілів передбачає розробку мобільного додатку. Це обумовлює вибір трирівневої клієнт-серверної архітектури, що містить наступні складові: клієнтський застосунок, сервер застосунків та сервер баз даних. Клієнтський застосунок підключений до сервера застосунків, який в свою чергу підключено до серверу баз даних.

Для створення бази даних буде використана мова SQL та додаток phpMyAdmin.

Мобільний застосунок буде створений під операційну систему Android, тому робота буде проводитися в середовищі Android Studio на мові програмування Java.

Взаємодія мобільного додатку та серверу буде організована через архітектурний підхід REST.

REST (скор. англ. Representational State Transfer, «передача репрезентативного стану») – підхід до архітектури мережевих протоколів, що забезпечують доступ до інформаційних ресурсів [9,10]. Це стиль проєктування розподілених систем за допомогою обмежень. Дані повинні передаватися у вигляді невеликої кількості стандартних форматів (наприклад, HTML, XML, JSON).

Однією з ключових переваг підходу API REST є те, що він забезпечує велику гнучкість. Дані не прив'язані до ресурсів або методів, тому REST може обробляти декілька типів викликів, повертати різні формати даних і навіть змінювати структурно з правильною реалізацією гіпермедіа. Така гнучкість дозволяє розробникам створювати API, що відповідає їх потребам, а також задовольняти потреби дуже різноманітних клієнтів.

У загальному представляє собою REST простий інтерфейс для управління інформацією без використання якихось додаткових внутрішніх прошарків. REST використовує найбільш поширений протокол – HTTP та передає дані у форматі json. REST дозволяє діяти клієнту та серверу одночасно незалежно, але при тому спілкуватись шляхом запит — відповідь. Для такої взаємодії створюється API, що приймає запити від клієнтів та надає їм відповідь від серверу.

Схему взаємодії клієнта та серверу у архітектурі REST показано на рисунку 2.3.

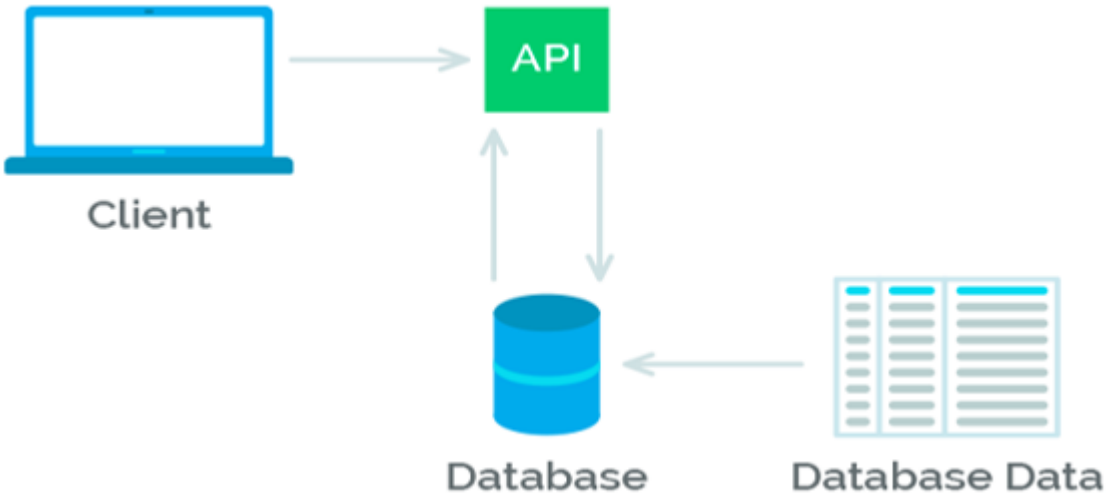


Рисунок 2.3 – Схема взаємодії клієнта та серверу у архітектурі REST

Для зображення моделі розроблювальної системи на фізичному рівні, побудовано діаграму розгортання Deployment Diagram, що показує зв'язки між усіма частинами системи. Основними елементами діаграми є фізичні та програмні компоненти — вузли. Апаратними компонентами даної системи виступають сервер та клієнти, веб-сайт. За допомогою даної діаграми представлено архітектуру системи, встановивши зв'язки між усіма вузлами (рисунок 2.4).

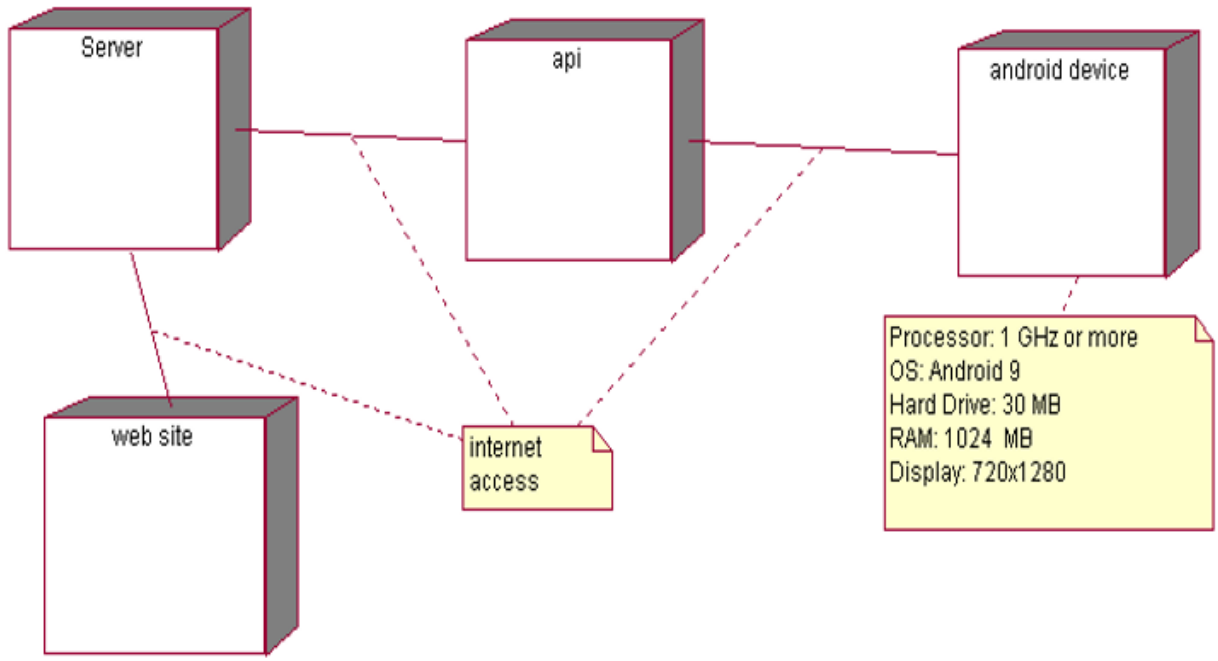


Рисунок 2.4 — Діаграма розгортання

2.3 Опис структури даних та моделі бази даних

Провівши аналіз предметної області можна приступати до проєктування баз даних. Вхідними даними є інформація, що стосується автомобілів, наявних в сервісі для прокату та інформація, що стосується клієнтів сервісу прокату автомобілів.

База даних – це засіб збирання та впорядкування інформації. Бази даних можуть зберігати відомості про людей, продукти, замовлення або будь-що інше. Комп'ютеризована база даних – це контейнер об'єктів. Приступимо до проєктування бази даних веб-орієнтованої системи для забезпечення роботи сервісу прокату автомобілів, використовуючі теорію нормалізації та правила приведення відношень БД до третьої нормальної форми [11].

В зазначеній предметній області були виділені такі інформаційні сутності:

- автомобілі;
- користувачі;
- бронювання;
- категорії;
- записи про заблокованих користувачів.

Реєструючись в системі клієнт з правами звичайного користувача має ввести такі дані: адресу електронної пошти, логін та пароль. Успішно пройшовши реєстрацію та створивши обліковий запис, користувач повинен заповнити форму і ввести паспортні дані та данні з військового посвідчення.

Данні, що стосуються парку автомобілів в сервісі прокату вносяться в систему користувачем з правами адміністратора. Адміністратором в систему мають вноситися дані, що стосуються категорії та моделей автомобілів. При потребі адміністратор адміністратор може редагувати данні клієнта. Крім того, він перевіряє данні про бронювання та змінює їх статус.

Даними, що відносяться до даних про бронювання є данні, що стосуються вибраного автомобіля, дати початку прокату та терміну прокату. У відповідності до сутностей, що були виділені після аналізу предметної області, будуть створені таблиці (відношення) бази даних.

Таблиця «Автомобілі» містить такі атрибути:

- ідентифікатор автомобіля;
- назва;
- кількість;
- ціна;
- зображення;
- об’єм двигуна;
- витрати палива;
- кількість сидінь;
- наявність кондиціонеру;
- типу кузову;
- тип трансмісії;
- кількість дверей;
- категорія.

Таблиця «Категорії» містить такі атрибути:

- ідентифікатор категорії;
- назва;
- статус.

Таблиця «Користувачі» містить такі атрибути:

- ідентифікатор користувача;
- логін;
- електронна пошта;
- пароль;
- роль;
- активність.

Таблиця «Дані про користувача» потрібна для перегляду даних клієнтів адміністратором. Також вона може використовуватися для відокремлення приватних даних користувача. Зазначена таблиця містить атрибути:

- прізвище;
- ім’я;
- по батькові;
- ідентифікатор користувача;
- номер посвідчення водія;
- дата видачі посвідчення водія;
- фотографія користувача;
- копія паспорту;
- копія водійського посвідчення;
- номер телефону.

В системі має бути передбачена можливість щодо блокування недобросовісних клієнтів, тому була створена таблиця «Інформація про блокування» з атрибутами:

- ідентифікатор блокування;
- ідентифікатор користувача;
- дата початку блокування;
- дата закінчення блокування;
- причина блокування та його статус.

Дані про блокування будуть заноситися до бази даних під час здійснення бронювання клієнтом через сайт. Редагування цих даних відбувається в момент надання автомобіля клієнту та після його повернення назад. Таблиця «Дані про бронювання» містить наступні атрибути:

- ідентифікатор бронювання;
- дата початку оренди;
- термін оренди;
- остаточна ціна оренди;

– статус бронювання;
– ідентифікатор автомобіля.

Отже, всі необхідні вхідні данні визначено: описано атрибути, спосіб їх збереження та керування. Описаних даних достатньо для початкової автоматизації сервісу прокату і вирішення поставлених задач.

Як вихідну інформацію можна позначити ті дані, що отримуються при оформленні бронювання автомобіля клієнтами. Після оформлення бронювання об’єкти заносяться до відповідної таблиці у базі даних, що містить необхідні атрибути.

Запис з даними про бронювання буде складатися з наступних атрибутів:

– ідентифікатор бронювання;
– ідентифікатор користувача;
– дата початку оренди;
– термін оренди;
– статус;
– вартість оренди.

Після створеня запису про бронювання користувач може відслідковувати історію бронювань в особистому кабінеті. Після погодження адміністратором бронювання, система має видавати результуючу інформацію, що стосується прокату автомобіля. Запис з даними буде містити наступні атрибути:

– id клієнта;
– id автомобіля;
– дата початку бронювання;
– термін бронювання;
– вартість;
– статус бронювання.

Клієнт, з правами користувача, має можливість перегляду всіх записів про виконані бронювання. Адміністратор, в свою чергу, може переглядати всі записи про бронювання, здійсненні в системі та виконувати операції сортування

та фільтрації даних. Адміністратор бачить також не підтверджені або не завершені бронювання.

Отже, визначивши вхідні ти вихідні дані, можна стверджувати, що описаних даних вистачить для коректної роботи себ-орієнтованої системи.

Приведення бази даних до третьої нормальної форми та встановлення зв’язків між відношеннями дозволили отримати схему бази даних. Схема бази даних наведена на рисунку 2.5.

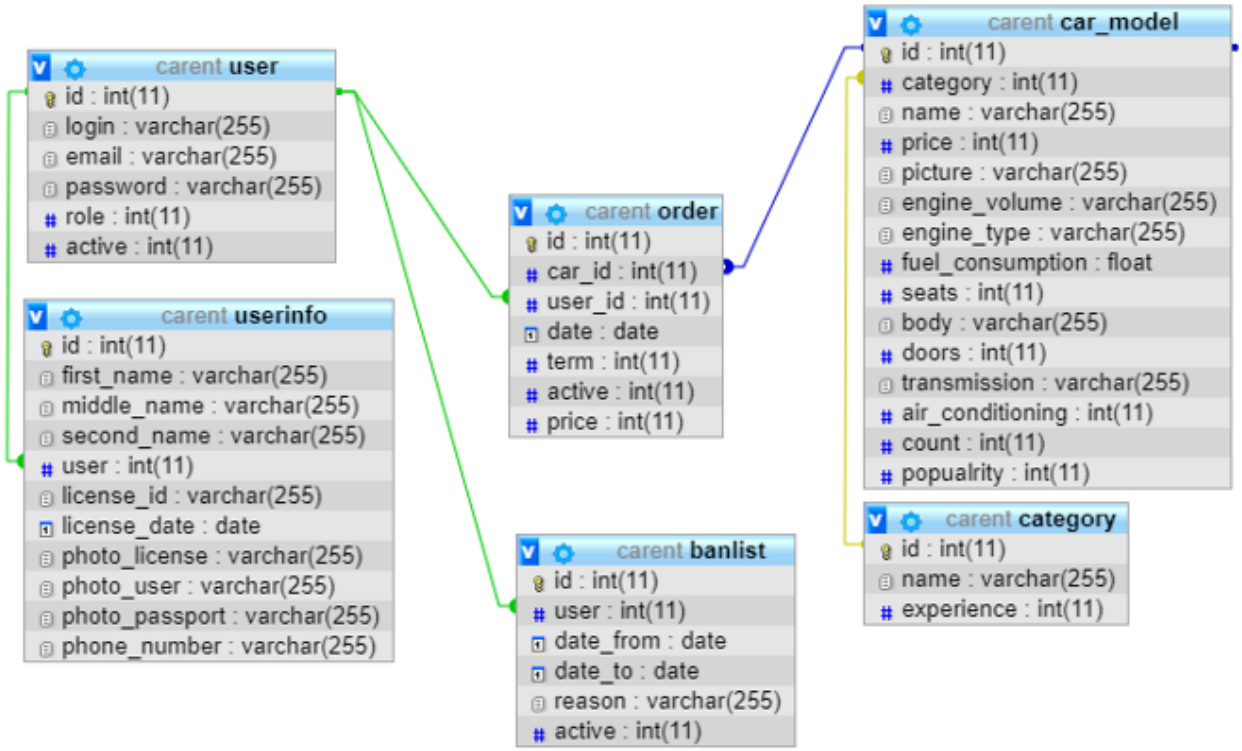


Рисунок 2.5 – Схема бази даних

Наступними кроками після проєктування бази даних є вибір технологій та методів для реалізації веб-орієнтованої системи та проєктування інтерфейсу веб-орієнтованої системи.

2.4 Аналіз та вибір технологій та методів реалізації веб-орієнтованої системи

Спираючись на результати проведеної формалізаці поставленної задачі, спроєктовану архітектуру та базу даних оберемо технології та методи реалізації веб-орієнтованої системи для забезпечення роботи сервісу прокату автомобілів.

База даних буде створена на мові SQL (Structured Query Language) за допомогою веб-додатку phpMyAdmin.

Мова запитів SQL – це універсальна мова для роботи з даними. Мова SQL використовується для управління масивами даних, множинами в базі даних [13].

Мова SQL надає можливість для виводу структурованої потрібної інформації з бази. SQL також використовується для зміни даних, додавання даних з бази.

Мова SQL відноситься до функціональних мов програмування. Вона відрізняється від алгоритмічних мов. В основі мови лежить не алгоритм як поняття, а сукупність команд, що визначають взаємовідносини інформаційних множин і підмножин.

Потрібно зазначити, що системи керування базами даних (СКБД) мають різні реалізації, такі як ORACLE, MS SQL, MySQL. Мова SQL в різних СКБД має невеликі відмінності, наприклад в детальному синтаксі опису операторів. Такі відмінності присутні і в спеціальних функціях, що відносяться до тієї чи іншої СКБД, але не дивлячись на це – мова SQL має загальний синтаксис, що є практично ідентичним для будь-якої СКБД.

Мова SQL – структурована мова запитів являє собою стандартну високорівневу мову опису даних і маніпулювання ними в системах управління базами даних, побудованих на основі реляційної моделі даних.

За допомогою мови SQL можна виконувати будь-які дії, що можна проводити з даними: від запису і читання даних, до адміністрування серверу СКБД. Для повсякденної роботи зовсім не обов'язково досконально знати мову

структурованих запитів, достатньо ознайомитися лише з основними поняттями синтаксису і ключовими словами. Крім того, SQL – це досить проста мова за своєю структурою, тому її вивчення не складає великих труднощів. Мова SQL – це в першу чергу мова запитів, яка є подібною до природньої мови людини. Кожен раз, коли потрібно прочитати або записати будь-яку інформацію в БД, потрібно скласти коректний запит [13]. Такий запит повинен бути виражений в термінах мовм SQL.

PhpMyAdmin – це набір скриптів у вигляді веб-додатку з відкритим кодом, що був написаний на мові PHP і представляє собою веб-інтерфейс для адміністрування баз даних MySQL. За допомогою PhpMyAdmin можна створювати, видаляти та редагувати таблиці бази даних, виконувати окремі SQL запити, створювати та видаляти користувачів, змінювати їх привілеї.

Програма PhpMyAdmin також дозволяє з браузера здійснювати адміністрування серверу MySQL, запускати команди SQL для роботи з вмістом таблиць баз даних, керувати СКБД MySQL без безпосереднього використання SQL команд через дружній користувацький інтефейс.

PhpMyAdmin надає додаткові можливості по роботі з MySQL, що дозволяє більш ефективно та легко працювати з даними. Причому всі функції доступні прямо з браузера, навіть перезавантаження віддаленого сервера (якщо ця можливість дозволена обліковим записом користувача) [13].

Сайт веб-орієнтованої системи для забезпечення роботи сервісу прокату автомобілів буде розроблятися на об'єктно-орієнтованій мові програмування PHP з використанням фреймворку Yii2 [14].

Yii – це компонентний обєктно-орієнтований фреймворк, назва якого розшифровується як «Yes it is!». Він призначений для PHP-розробки, що базується на парадигмі MVC.

Головна перевага фреймворка Yii та його наступної версії Yii2 це швидкодія та продуктивність. Крім цього, ці фреймворкі досить прості для використання. Основна сфера їх застосування це розробка технічно складних проєктів, та таких, що не можуть бути реалізовані за допомогою простих CMS:

– веб-сервісів;
– додатків для бізнесу;
– сайтів, що мають вимоги до швидкодії.

Yii2 можна використовувати для розробки будь-якого виду веб-додатків. Завдяки своїй основі компонентів, архітектурі і складної підтримки кешування, фреймворк підходить для розробки великомасштабних проєктів, таких як портали, форуми, системи управління контентом (CMS), систем електронної комерції, RESTful веб-сервісів тощо.

Говорячи про фреймворкі Yii, Yii2 веб-розробники відзначають їх високу продуктивність в порівнянні з іншими фреймворками для PHP. Також, використовується DAO (data access object) інтерфейс та інтерфейс ActiveRecord, що спрощує роботу з базами даних.

Фреймворк Yii2 використовують бекенд- та фулстек- розробники. Завдяки компонентній структурі, та відмінній підтримці кешування Yii2 можна використовувати для створення порталів, форумів, CMS, магазинів або RESTful-додатків. Yii2 реалізує звичні для усіх фреймворків функції:

– прискорення написання коду;
– створення архітектури додатків;
– прискорення розв’язку рутиних та шаблоних завдань;
– захист написаних сайтів від атак;
– оброблення помилок;
– тестування та відлагодження написаного коду;
– підключення сайту до бази даних, додаткових модулів тощо.

Yii2 має наступні можливості:

– повна підтримка jQuer-фронтенд-бібліотеки для взаємодії між HTML-розміткою сторінок та її JavaScript-кодом;
– підтримка Bootstrap, фреймворка для швидкого створення шаблоних елементів веб-інтерфейсу;
– захист від SQL-ін’єкцій та XSS-атак на сайти;

– можливість роботи з базою даних та здійснення її міграції – контроль версій вмісту;

– робота з шаблонізаторами Smarty та Twig;

– підтримка технології AJAX для динамічної загрузки вмісту та стилю програмування REST, що описує особливості взаємодії компонентів;

– генерація шаблоного коду за допомогоювбудованого інструмента Gii;

– відлагодження за допомогою власного відлагоджувача Yii2-debug;

– автоматичне тестування, обробка помилок та валідація форм;

– підтримка legacy-коду, що був «успадкований» з попередніх циклів.

Перевагами Yii2 є:

– легкість в опануванні. Підходить для програмістів-початківців. Достатньо мати загальне представлення про класи, наслідування та методи;

– універсальність. Призначений для створення будь-яких за розміром додатків завдяки базовому та розширеному шаблонам;

– функціональність. Має багато вбудованих можливостей: від власних інструментів до підтримки популярних шаблонів;

– доступність. Yii2 розповсюджується за модифікованою ліцензією BSD. Його можна безкоштовно скачувати та використовувати для розробки платних та безплатних додатків. Документація доступна для всіх бажаючих.

– підтримка при використанні. Велика спільнота програмістів, що використовує фреймворк може надати відповіді на питання, що виникають при використанні.

Мобільний додаток розробляється під операційну систему Android. Тому, доцільним буде проводити розробку зазначеного застосунку в середовищі Android Studio з використанням мови програмування Java.

Java – є однією з найбільш розповсюджених мов програмування, що стала популярною в 1995 році. Спочатку вона використовувалася розробниками для написання програмного забезпечення для телевізорів та телеприставок. Згодом

ця мова стало універсальною мовою для розробки найрізноманітніших додатків. Згідно деяких підрахунків, мова програмування Java використовується у 3 мільярдах пристроїв. Саме завдяки цьому компанія Google обрала мову Java для створення додатків [15].

Java використовується в наступних сферах:

– розробка бекенду (бекенд таких відомих сайтів як ebay.com, PayPal, amazon.com, LinkedIn, Facebook, Twitter, YouTube створений за допомогою Java);

– розробка мобільних застосунків та рішень для десктопних пристроїв;

– веб-розробка;

– автоматизація QA;

– робота з Big Data;

– програмування контролерів для розумних будинків;

– написання програм для обробки даних в наукових дослідженнях тощо.

Перевагами мови програмування Java є:

– зручність та легкість, навіть для початківців;

– велика кількість літератури;

– розроблені великі бібліотеки та фреймворки;

– кросплатформеність;

– велика кількість напрацьованих практик, концепцій та підходів;

– велика кількість вбудованих функцій для підвищення безпеки;

– розповсюдженість пристроїв на Android, що застосовують Java.

Недоліками мови програмування Java є:

– довгі та складні конструкції;

– складний код, що не сприяє «красі» програмного коду ;

– «багатослівність».

Поняття «об'єктно-орієнтована» відноситься до способу написання структурного коду Java, а саме: поділ коду на так звані «класи», які запускаються разом, щоб забезпечити узгоджене породження об'єктів.

Java знаходиться під впливом C і C ++, тому вона має багато спільного з цими мовами (і C #). Одним з великих переваг Java є те, що вона «переносних незалежний». Це означає, що код, який ви пишете на одній платформі, можна легко запустити на інший. В цьому і є причина використання Java для розробки додатків для платформ Android.

Для роботи мобільного застосунку необхідно створити зв’язок між застосунком та сервером сервісу. Для цього розробляється API (application programming interface). API (програмний інтерфейс програми, інтерфейс прикладного програмування) – опис способів (набір класів, процедур, функцій, структур або констант), за допомогою яких одна комп'ютерна програма може взаємодіяти з іншою програмою. Розробка API проводиться на мові програмування PHP з використанням фреймворку Yii2.

Для роботи мобільного додатку та зокрема виконанню запитів вирішено підключити бібліотеку Retrofit. Retrofit – безпечний клієнт HTTP для Android та Java. За допомогою даної бібліотеки можливо здійснювати запити до API сервера та отримувати відповіді в форматі JSON — текстовий формат обміну даними, заснований на JavaScript. Але при цьому формат незалежний від JS і може використовуватися в будь-якій мові програмування.

Розроблені сайт, API та база даних зберігаються на хостингу, завдяки чому будь-який клієнт може користуватися системою, та отримувати відповіді від серверу, основною умовою є лише наявність мережі.

2.5 Проєктування інтерфейсу веб-орієнтованої системи

Інтерфейс користувача – це сукупність засобів для спілкування користувача з різними пристроями, такими як комп’ютер, побутова техніка або з іншим складним інструментарієм.

Згідно [16] « інтерфейс користувача додатку включає:

– засоби відображення інформації, відображувальну інформацію, формати та коди;

– командні режими, мову «користувач-інтерфейс»;

– пристрої та технології введення-виведення;

– діалоги, взаємодію та транзакції між користувачем та комп’ютером, зворотній зв'язок з користувачем;

– підтримку прийняття рішень в конкретній предметній області;

– порядок використання програми та документації на неї».

Інтерфейс користувача не обмежується зовнішнім виглядом програми. Він об’єднує всі елементи та компоненти веб-орієнтованої системи, що впливають на взаємодію користувача з програмним забезпеченням.

Критеріями при розробці інтерфейсу веб-орієнтованої системи є зрозумілість для користувача та простота в опануванні роботи з програмним забезпеченням. Тобто, користувач, що не має спеціальної комп’ютерної підготовки, повинен розуміти та виконувати без втрату часу всі необхідні дії при роботі з веб-орієнтованою системою.

Інтерфейс користувача веб-орієнтованої системи, що проєктується можна умовно поділити на дві частини: інтерфейс веб частини системи та інтерфейс мобільного додатку.

Для проєктування сайту веб-орієнтованої системи доцільно використати відомий шаблон проєктування Модель Представлення Контролер (MVC, Model View Controller). Модель MVC – це особливий спосіб організації програмного коду, що передбачає виділення блоків, кожен з яких буде відповідати за

рішення своїх задач. Один відповідає за данні додатку, інший – за зовнішній вигляд, третій за роботу додатку. В цьому контексті компоненти шаблону можна описати наступним чином:

– модель (model) – отримує дані від контролера, виконує необхідні операції та передає їх до представлення;

– вид або представлення (view) – отримує дані від моделі та виводить їх для користувача;

– контролер (controller) – обробляє дії користувача, перевіряє отриманні дані та передає їх моделі [17].

Шаблон потрібно застосовувати для того, щоб відокремити дані (моделі) від інтерфейсу користувача, таким чином, щоб зміни інтерфейсу мінімально впливали на роботу з даними. А зміни в моделі даних здійснювалися без змін інтерфейсу користувача.

Мета використання шаблону це можливість гнучкого дизайну програмного забезпечення, що повинен полегшувати подальші зміни чи розширення програм, а також надавати можливість повторного використання окремих компонентів програми [17]. Практика використання цього шаблону у великих складних системах є виправданою і сприяє впорядкованості їх структури, що робить їх більш зрозумілими за рахунок зменшення складності. Схема архітектури шаблону наведена на рисунку 2.6.

На схемі чітко видно розподіл між складовими елементами шаблону. Це надає можливість налаштування інтерфейсу веб-додатку без зміни його логіки.

Обраний для розробки програмного забезпечення фреймворк Yii2 забезпечує створення проєктів будь-якої складності, використовуючі вбудовані елементи і генератори, та налаштовуючі їх під власні За допомогою Gii генератора у проєктах Yii2 створюються компоненти, що забезпечують CRUD інформації, а також, в загальному, полегшують роботу при написанні коду.

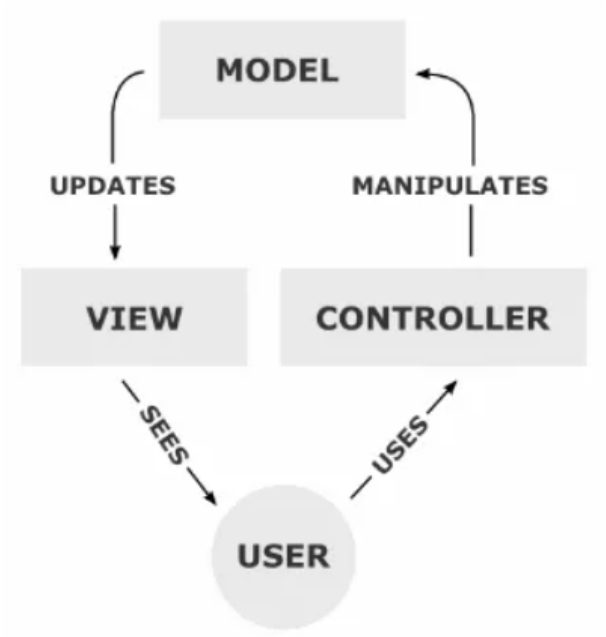


Рисунок 2.6 – Схема архітектури шаблону MVC

CRUD – акронім, що позначає чотири базові функції, які використовуються при роботі з базами даних: створення (англ. Create), читання (read), модифікація (update), видалення (delete). Введено Джеймсом Мартіном в 1983 році як стандартну класифікацію функцій по маніпуляції даними [10].

У SQL цих функцій, операцій відповідають оператори Insert (створення записів), Select (читання записів), Update (редагування записів), Delete (видалення записів). У деяких CASE-засобах використовувалися спеціалізовані CRUD-матриці або CRUD-діаграми, в яких для кожної сутності вказувалося, які базові функції з цією сутністю виконує той чи інший процес або та чи інша роль. У системах, що реалізують доступ до бази даних через API в стилі REST, ці функції реалізуються найчастіше (але не обов'язково) через HTTP-методи POST, GET, PUT і DELETE відповідно.

Хоча традиційно оперування в стилі CRUD застосовується до баз даних, такий підхід може бути поширений на будь-які збережені обчислювальні суті (файли, структури в пам'яті, об'єкти). Шаблон проєктування ActiveRecord забезпечує відповідність функцій CRUD об'єктно-орієнтованого підходу, і

широко використовується в різних фреймворків для доступу до баз даних з об'єктно-орієнтованих мов програмування.

Стилізація елементів створених за допомогою Gii-генератору за замовчуванням виконана із використанням Bootstrap – найпопулярнішої HTML, CSS та JS бібліотеки. Bootstrap включає в себе HTML і CSS-шаблони оформлення для типографії, веб-форм, кнопок, міток, блоків навігації та інших компонентів веб-інтерфейсу. Ці шаблони можна сміливо використовувати для створення власного інтерфейсу ПЗ, оскільки вони є сучасними та створені із врахуванням потреб та побажань користувачів.

Стабільна базова версія Yii2 надає готові шаблони для сторінок із контентом, контактами, формою авторизації, а також панель навігації. Результати показані на рисунках 2.7 та 2.8.

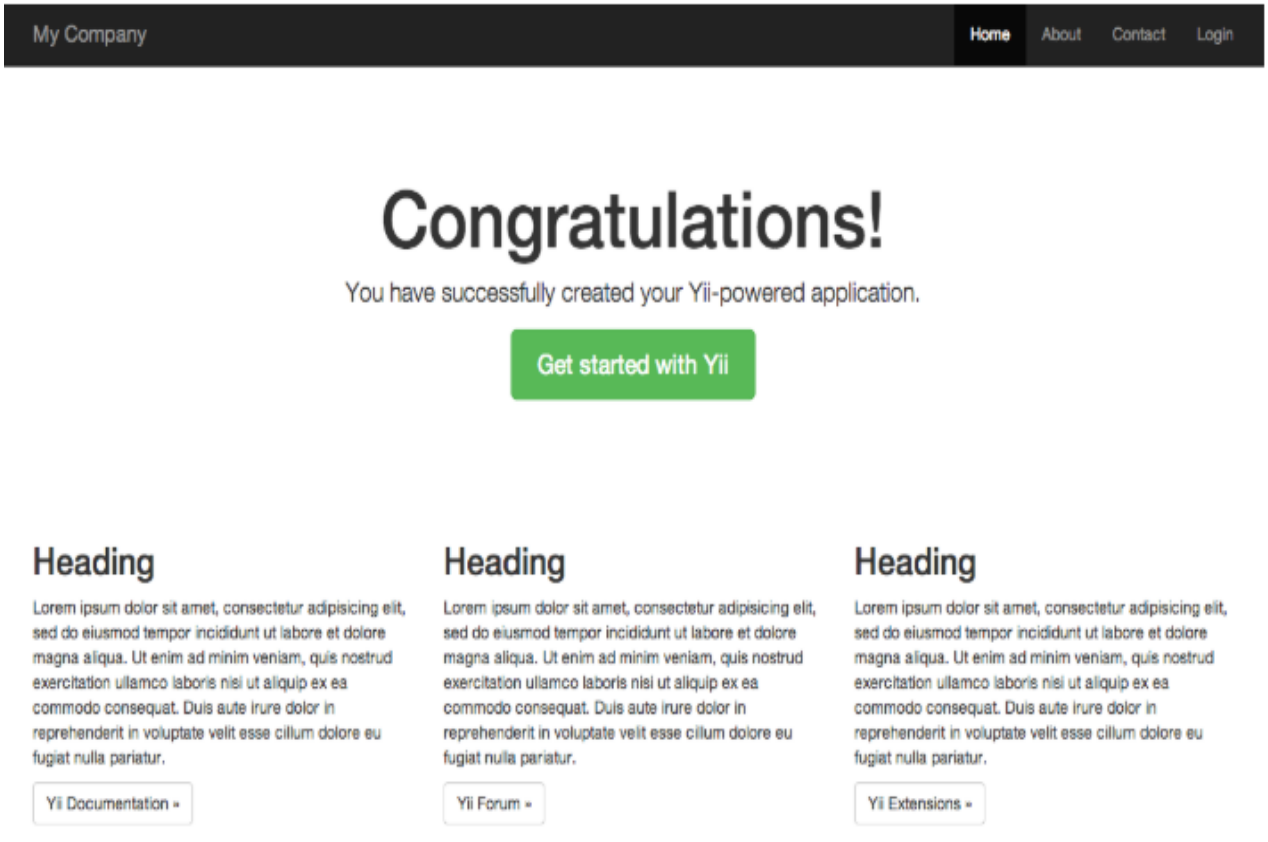


Рисунок 2.7 – Головна сторінка базової версії Yii2 та панель навігації

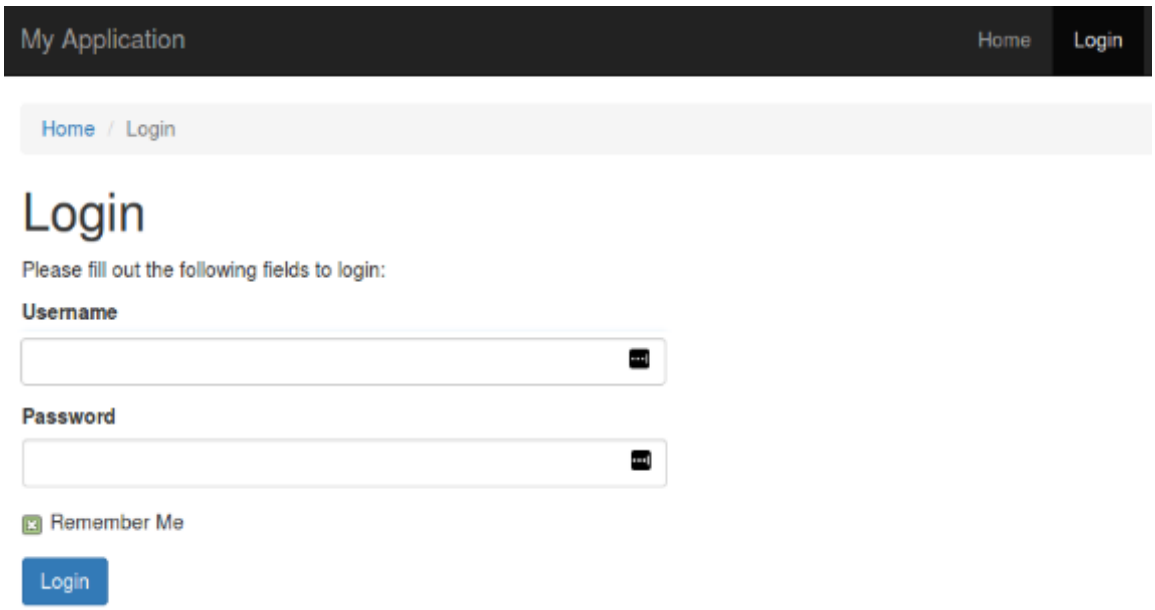


Рисунок 2.8 – Шаблон сторінки авторизації базової версії Yii2

На даному етапі було обрано стандартні шаблони для створення веб частини сервісу зокрема адміністративної частини. Далі необхідно створити інтерфейс для мобільного додатку.

Для створення даного мобільного додатку було обрано стандартні принципи інтерфейсу Material Design . Material Design – це принципи дизайну для сайтів, програмного забезпечення та застосунків, крім того, це правила дизайну інтерфейсів для операційної системи Android від компанії Google.

Однією з особливостей розробки програмного забезпечення для операційної системи Android є наявність xml розмітки. XML (розширювана мова розмітки) – це мова програмування, яка складається з оголошень у вигляді інформації і визначення тегів. З її допомогою зручно зберігати і передавати будь-які дані. Мова не залежить від операційної системи і середовища обробки. XML служить для представлення деяких даних у вигляді структури, яку ви можете самі розробити або підлаштувати під програму або сервіс [15]. Саме тому дану мову називають розширюваною, і в цьому її головна перевага, за яке її так цінують. Xml розмітка відділяє інтерфейс програми від програмного коду. Це допомагає працювати з інтерфейсом програми не заторгаючи алгоритми її роботи.

Під час створення даного проєкту було надано увагу адаптивності рішення, оскільки даний застосунок може працювати на різних пристроях з різними розширеннями екрану. Для цього розміри елементів інтерфейсу вимірюються в dpi, що означає кількість пікселів на сантиметр екрану.

При створення нового проєкту в середовищі Android Studio є можливість вибору базового інтерфейсу вікна. Це полегшує роботу над інтерфейсом та допомагає зекономити час но розробці базових рішень.

Вибір для шаблону створення нової активності показано на рисунку 2.9.

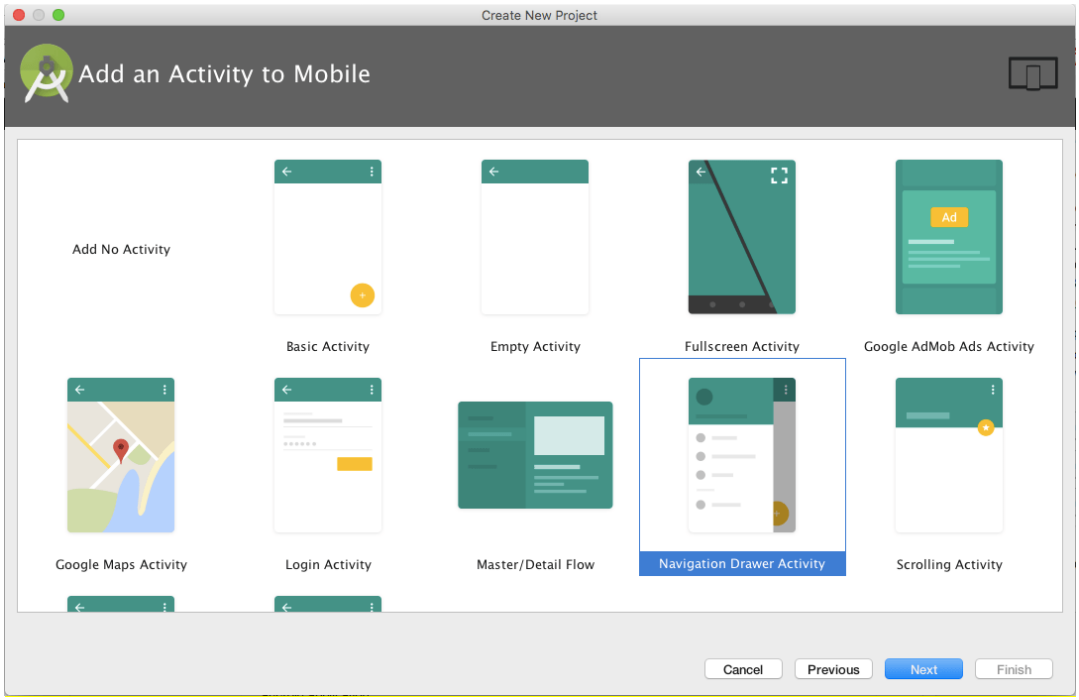


Рисунок 2.9 – Вибір базового шаблону для нової активності

Елементи на розмітці вікна повинні міститись у контейнерах. В Android використовуються так контейнери як RelativeLayout, LinearLayout, GridLayout, TableLayout, ConstraintLayout, FrameLayout. Всі вони мають різну структуру та використовуються для різного розміщення елементів на активності.

Для виводу інформації на екран та виконання дій у Android додатку використовуються такі елементи управління, як TextView, EditText, Button, Spinner, DatePicker, DialogWindow, що показаний на рисунку 2.10 тощо.

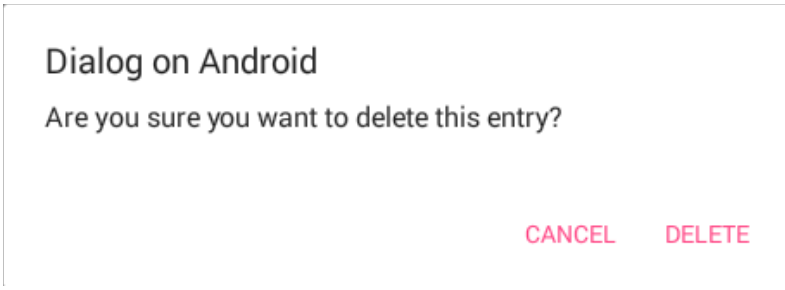


Рисунок 2.10 – Елемент діалогу

Таким чином, при розробці додатку у Android Studio, можна користуватися численними можливостями та елементами для створення зручного інтерфейсу.

Для зручного доступу до всіх головних можливостей додатку було обрано «bottom navigation bar», показаний на рисунку 2.11, який дозволяє переходити між активностями програми.

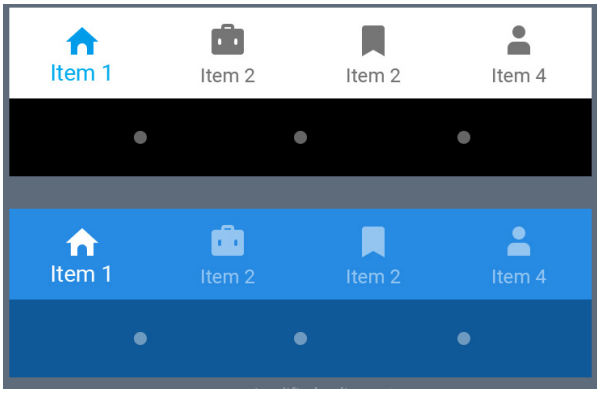


Рисунок 2.11 – Шаблон головного меню додатку

На даному етапі було розроблено інтерфейс програми, а також визначено використання додаткових шаблонів для реалізації додатку.

Таким чином в даному розділі проведено проєктування системи. Визначено вхідні та вихідні дані, спроєктовано базу даних. Для планування роботи системи було проведено формалізацію та розроблено UML діаграми для кращого представлення роботи системи. В даному розділі також було проведено роботу над плануванням інтерфейсу програми та визначено алгоритми зв’язку клієнта та сервера.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Розробка сайту веб-орієнтованої системи

В попередньому розділі був здійснений вибір технологій для розробки програмного забезпечення веб-орієнтованої системи. Розробка ПЗ буде здійснюватися мовою програмування PHP з використанням фреймворку Yii2. Середовищем розробки буде програма PhpStrom. Вбудовані можливості цієї програми дозволяють реалізовувати проєкти на PHP, мають можливість підключення систем контролю версій тощо.

Для відображення моделі роботи сервісу прокату автомобілей на фізичному рівні розроблено діаграму компонентів. На розробленій діаграмі видно зв'язки між усіма складовими веб-орієнтованої системи: апаратними компонентами, бібліотеками, файлами, виконуваними частинами коду. Данна діаграма відображає архітектуру системи, встановлює залежності між програмними компонентами. Діаграма компонентів (Component Diagram) наведена в додатку Е.Є

Розробку програмного забезпечення було розпочато з конструювання бази даних. Для створення бази даних, використовувались СУБД MySQL та веб-додаток PhpMyAdmin.

Наступний етап це створення сайту веб-орієнтованої системи. Для цього використаємо фреймворк Yii2. За допомогою компонента Composer та консольної команди створюємо базовий проєкт Yii2:

```
composer global require "fxp/composer-asset-plugin:^1.4.1"
composer create-project --prefer-dist yiisoft/yii2-app-basic basic .
```

Як було визначено раніше, сайт має бути розроблений за архітектурою MVC. Використовуючі вбудований генератор Gii, що створює необхідні елементи програми автоматично, були створені моделі у відповідності до

спроєктованих таблиць, контролерів та представлень. Наступним кроком було налаштування цих елементів під необхідний результат. Оскільки веб-орієнтована система має адміністративну та клієнську частини, то створювалися окремі модулі для адміністраторів та клієнтів. Клієнтський модуль є публічним, тому його модель є стандартною і не потребує додаткового створення.

Кожен створений модуль підключається у файлі конфігурації web.php.

```
'modules' => [
    'admin' => [
        'class' => 'app\modules\admin\Module',
    ],],
```

З метою розмежування адміністративної панелі та доступу звичайних користувачів було створено права доступу. Наведемо наведено приклад обмеження доступу у модулі для адміністраторів.

```
public function behaviors(){
    return[
      'access' => [
        'class' => AccessControl::className(),
        'denyCallback' => function($rule,$action){
            throw new \yii\web\NotFoundHttpException();
        },
        'rules' => [[
              'allow' => true,
              'matchCallback' => function($rule, $action){
                  return \Yii::$app->user->identity->role;
              }
                                    ]
                          ]
              ]];}
```

Для роботи проєкту створюємо моделі, що є класами в PHP та напряму пов'язані з відповідними таблицями у базі даних. Центральним компонентом шаблону MVC є модель, що відображає поведінку застосунку і є незалежною від інтерфейсу користувача.

При створенні моделей використовуємо генератор Gii. Створені моделі проєкту відображено на рисунку 3.1.

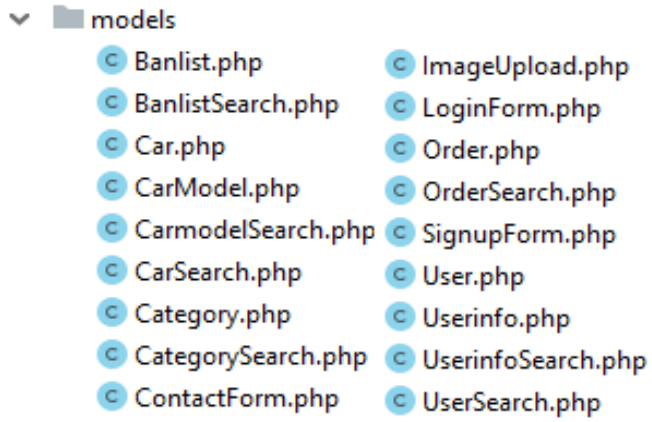


Рисунок 3.1 – Моделі проєкту

Наступним кроком є створення контролерів. Контролер призначений для того, щоб одержувати вхідні дані й перетворювати їх на команди для моделі чи представлення. Контролери створюються за допомогою генератора Gii. Усі створені контролери моделей представлені на рисунку 3.2.

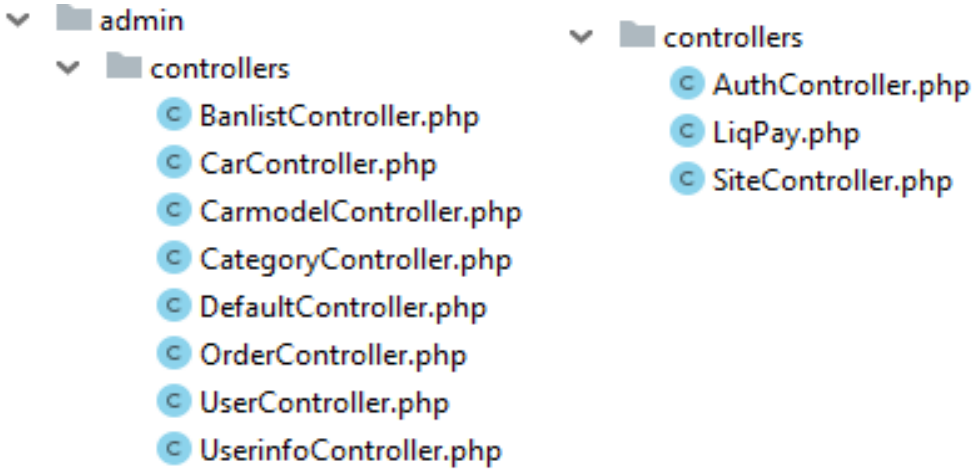


Рисунок 3.2 – Контролери моделей та контролери системи

Наступним кроком, що йде після створення моделей та контролерів є створення графічної частини проєкту, тобто представлень. Створення представлень може відбуватися одночасно зі створенням контролерів в генераторі Gii або безпосередньо в каталозі проєкту. За представлення відповідає окремий каталог з назвою View

Отже, створивши контролери, моделі та представлення ми отримали майже повний функціонал, необхідний для роботи з даними веб-орієнтованої системи для забезпечення роботи сервісу з прокату автомобілів.

3.2 Розробка функціоналу сайту веб-орієнтованої системи

Адміністративна частина сайту веб-орієнтованої системи повинна забезпечувати виконання наступних функцій:

- перегляд та редагування даних про автомобілі, які є в сервісі прокату;
- перегляд інформації про клієнтів сервісу;
- веріфікація клієнтів сервісу прокату автомобілів;
- перегляд виконаних бронювань автомобілів;
- редагування статусу бронювань автомобілів;
- блокування проблемних клієнтів на визначений період;
- додавання та редагування категорій автомобілів.

Розпочинаючи роботу в системі адміністратор має авторизуватися як звичайний користувач, а далі перейти за адресою carent/admin/. Тоді, здійснюється перехід на сторінку з усіма можливостями, що має адміністратор (рисунок 3.3).

Через меню, розташоване на адміністративній панелі, адміністратор взаємодіє з всіма таблицями бази даних: переглядає, редагує, створює та видаляє записи.

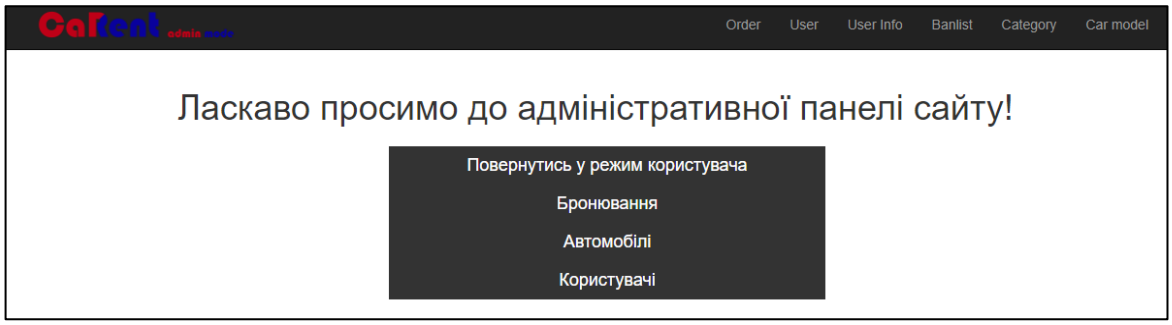


Рисунок 3.3 – Головна сторінка адміністративної панелі

Результати роботи з таблицями бази даних через адмінку представлено на рисунках 3.4 та 3.5.

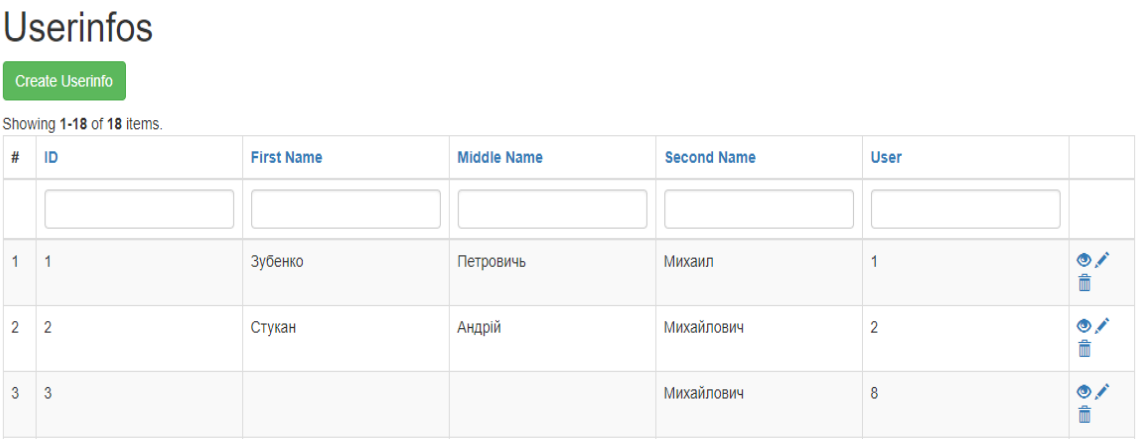


Рисунок 3.4 –Перегляд інформації про користувачів

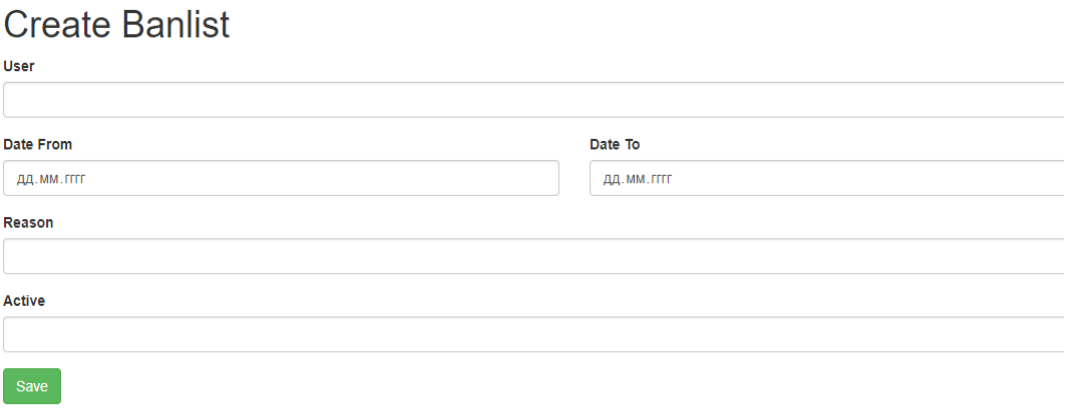


Рисунок 3.5 – Створення запису про блокування користувача

Після розробки функціоналу адміністративної частини сайту перейдемо до розробки клієнтського функціоналу.

Клієнтська частина сайту веб-орієнтованої системи повинна забезпечувати виконання наступних функцій:

– надання інформації про сервіс прокату автомобілів та його умови та можливості;
– реєстрація та авторизація клієнтів;
– перегляд інформації про автомобілі, що в є сервісі прокату;
– бронювання автомобіля;
– перегляд історії та стану бронювання.

Незареєстрований користувач повинен мати можливість перегляду інформації про можливості сервісу прокату автомобілів. Проте незареєстрований користувач не може здійснювати потрібні йому дії з вибору та бронювання автомобіля. Тому, для отримання доступу до основного функціоналу сайту, новий користувач повинен зареєструватися, заповнивши коротку форму реєстрації (рисунок 3.6).

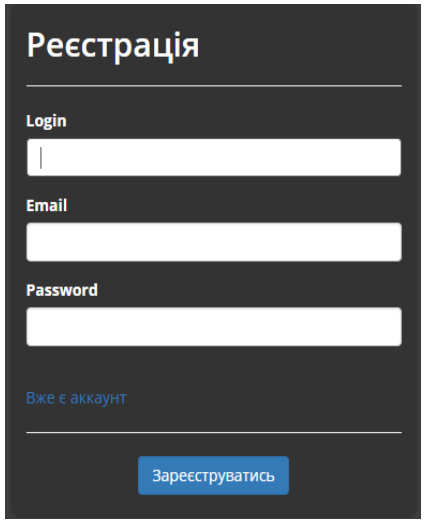


Рисунок 3.6 — Реєстрація нового користувача

Після реєстрації користувач потрапляє до особистого кабінету, де може ввести паспортні дані та данні про посвідчення водія. Ці операції користувач

може здійснити в кабінеті, натиснувши кнопку «Редагувати інформацію про користувача». Результат показано на рисунку 3.7.

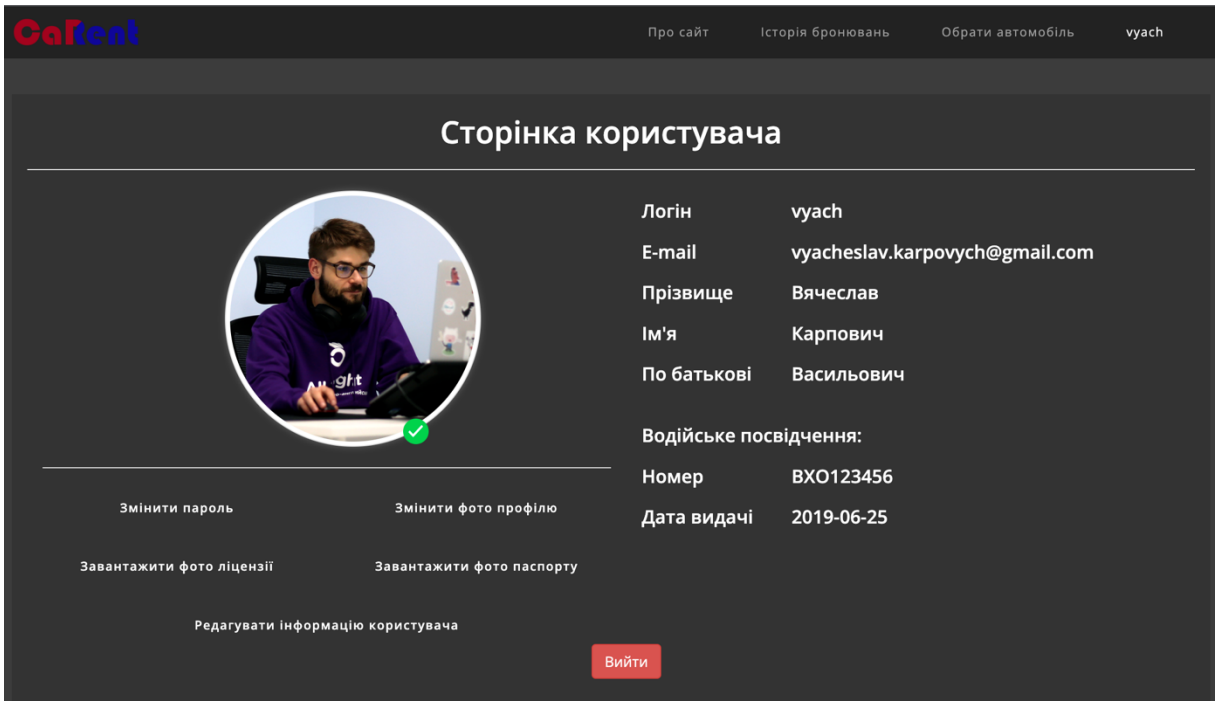


Рисунок 3.7 – Кабінет користувача

Коли користувач вводить дані про себе настає черга адміністратора, який здійснює верифікацію даних та підтверджує або відхиляє реєстрацію користувача. Верифікований користувач отримує відповідну позначку у своєму кабінеті. Результат показано на рисунку 3.8.

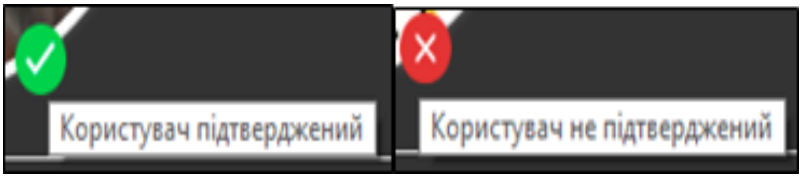


Рисунок 3.8 – Підтверджений та не підтверджений користувач

Після успішної верифікації для користувача відкривається можливість бронювати потрібні автомобілі. Результат показано на рисунку 3.9.

Рисунок 3.9 – Бронювання авто

Після внесення даних в форму бронювання та замовлення бронювання користувач отримує повідомлення про виконання бронювання, яке показане на рисунку 3.10, та може отримати довідку про історію та стан бронювань.

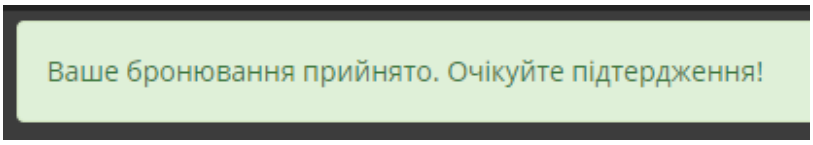


Рисунок 3.10 — Повідомлення про проведення успішного бронювання

Отже, було розроблено функціонал адміністративної та клієнтської частини сайту веб-орієнтованої системи для забезпечення роботи з сервісу прокату автомобілів. Фрагмент програмного коду наведено в додатку Ж. Оскільки веб-орієнтована система складається з сайту та мобільного додатку, то наступним кроком буде розробка API для роботи мобільного додатку.

3.3 Розробка API для мобільного додатку

Вибраний фреймворк Yii2 має в своєму складі набір RESTful API, який може використовуватись для реалізації API. Yii2 включає повноцінний набір засобів для реалізації RESTful API, тому даний фрейворк доцільно використовувати для розробки API проєкту.

Для реалізації API потрібно створити новий модуль. Цю дію можна реалізувати через генератор Gii. В новому модулі для API створимо контролери необхідних сутностей. Модуль API та контролери створені для роботи API системи показані на рисунку 3.11.

Для відправки даних за допомогою RESTapi використовується протокол HTTP. Також, для передачі даних необхідно вказати формат. В проєкті будемо використовувати формат передачі даних json.

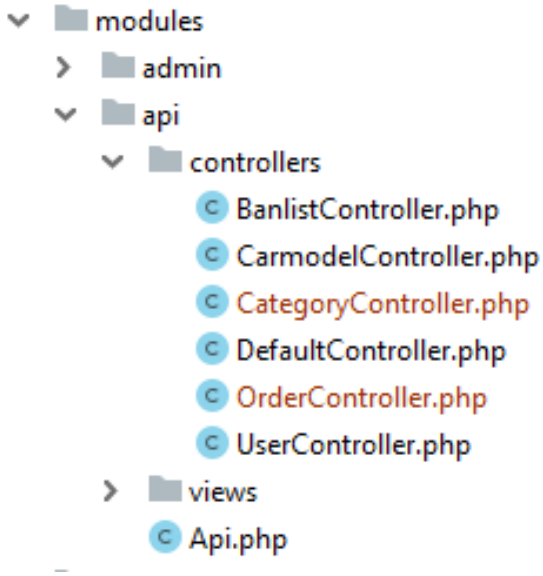


Рисунок 3.11 – Створені контролери для роботи API

Для роботи API також було налаштовано конфігурацію програми у файлі «web.php»:

```
'components' => [
            'request' => [
               'cookieValidationKey' => 'WsOtLgmf4VZx7TIP0KuOSCFD7kmBRg03',
               'baseUrl'=> '',
            ],
'modules' => [
   'admin' => [
      'class' => 'app\modules\admin\Module',],
```

```
'api' => [
    'class' => 'app\modules\api\Api',],],
```

Отже, в процесі програмної реалізації функціоналу веб-орієнтованої системи для забезпечення роботи сервісу прокату автомобілів було створено та налаштовано API на сервері системи для прийому запитів від клієнтського мобільного додатку. Для перевірки вірності написання API функцій використовувалась програма Postman.

3.4 Розробка мобільного додатку для клієнтів

Реалізацій мобільного додатку буде здійснена під операційну систему Android, тому для розробки додатку доцільно сикористовувати середовище Android Studio.

Створюємо в середовищі Android Studio новий проєкт. Наступним кроком є підключення бібліотеки retrofit для виконання http запитів до серверу через мобільний додаток.

Пропишемо залежності у файлі build.gradle для підключення бібліотеки:

```
implementation 'com.squareup.retrofit2:retrofit:2.3.0'
implementation 'com.squareup.retrofit2:converter-gson:2.3.0'.
```

Для з’єднання додтку з сервером через мережу інтернет необхідно надати доступ на використання даної мережі в файлі Manifest:

```
<uses-permission android:name="android.permission.INTERNET" />.
```

Наступним кроком є створення класів для всіх сутностей. Створені класи будуть аналогічними моделям, що були створенні при робробці сайту веб-орієнтованої системи.

Для забезпечення роботи API необхідно створити інтерфейс «API» через який буде відбуватися з'єднання з сервером, прийом та передача даних. Приклад оголошення методу в інтерфейсі:

```
@Multipart
@POST("api/user/uploadphoto")
Call<ResponseBody> uploadImage(@Part MultipartBody.Part file, @Part("upfile")
RequestBody requestBody, @Query("userid") int userid).
```

Щоб викликати створені методи необхідно створити екземпляри бібліотеки та інтерфейсу:

```
retrofit = new Retrofit.Builder()
        .baseUrl(data.url)
        .addConverterFactory(GsonConverterFactory.create())
        .build();
api = retrofit.create(Api.class).
```

Описані вище дії забезпечили створення проєкту та реалізацію його підключення до серверу через власний інтерфейс API за допомогою бібліотеки retrofit.

Веб-орієнтована система для забезпечення роботи сервісу прокату автомобілів крім сайту включає мобільний додаток. Мобільний додаток створюється з метою полегшення доступу клієнтам до послуг прокату автомобілів. Завдяки такій побудові веб-орієнтованої системи, клієнти можуть здійснювати бронювання автомобілів та відслідковувати статуси як через сайт, так і через мобільний додаток. Тому доцільно, щоб мобільний додаток мав функціонал аналогічний функціоналу сайту.

В мобільному додатку аналогічно до сайту була реалізована можливість створення облікового запису і входу через створений обліковий запис. Для цього необхідно натиснути на відповідний пункт меню при старті програми. Результат відображено на рисунку 3.12.

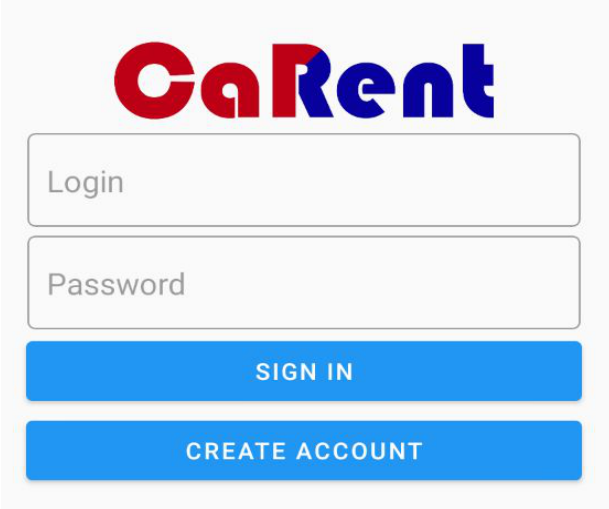


Рисунок 3.12 – Вікно входу до облікового запису

Наступним кроком є створення налаштувань облікового запису, можливостей перегляду списку автомобілів, бронювання та перегляду історії бронювань. Вікна мобільного додатку, що реалізують ці активності наведені на рисунках 3.13 та 3.14.

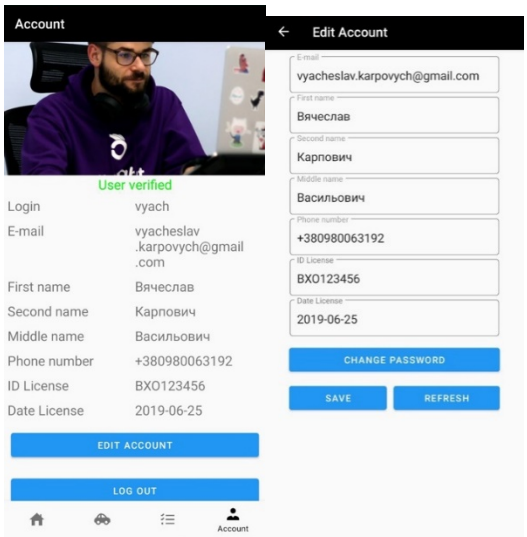


Рисунок 3.13 – Налаштування облікового запису

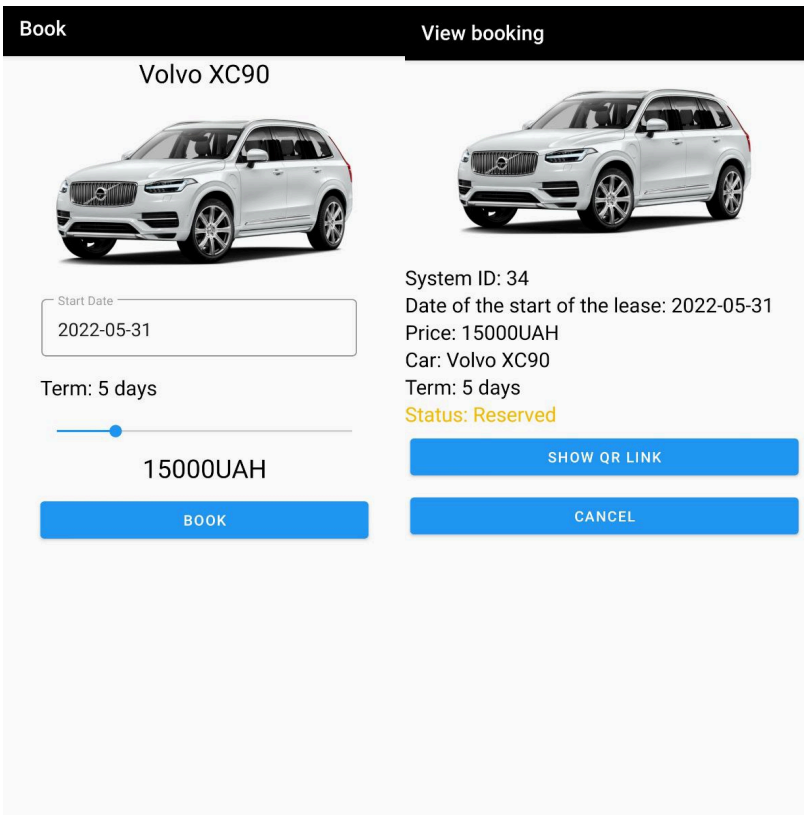


Рисунок 3.14 – Налаштування облікового запису

Отже, в результаті програмної реалізації було створено сайт та мобільний додаток веб-орієнтованої системи для забезпечення роботи сервісу прокату автомобілів. Функціонал сайту та мобільного додатку є аналогічними, що забезпечує можливість для клієнта вибирати зручний для нього спосіб бронювання автомобіля та отримання послуг сервісу з прокату автомобілів.

4. ТЕСТУВАННЯ ВЕБ-ОРІЄНТОВАНОЇ СИСТЕМИ ДЛЯ ЗАБЕЗПЕЧЕННЯ РОБОТИ СЕРВІСУ ПРОКАТУ АВТОМОБІЛІВ

4.1 Тестування сайту веб-орієнтованої системи

Для контролю якості програмного забезпечення, що розробляється використовують тестування. Тестування визначається як процес виконання програми для перевірки відповідності між її реальною поведінкою та тією, що була визначена при написанні технічного завдання. Тестування виконують на кінцевому наборі тестових даних, що були визначені певним способом.

Під тестуванням програмного забезпечення розуміють процес технічного дослідження, що призначений для виявлення інформації щодо якості програмного забезпечення, відносно контексту, в якому він буде використовуватися. Крім того, техніка тестування включає і процес пошуку помилок та інших дефектів, і здійснення випробування всіх програмних складових з метою оцінки [18].

Під час тестування, зазвичай, оцінюють:

– відповідність вимогам, викладеним в специфікації або в технічному завданні;

– здійснення потрібних функцій за заданий проміжок часу;

– зручність виконання запрограмованих функцій;

– сумісність з іншим програмним забезпеченням, що використовується, та з операційними системами;

– відповідність поставленим задачам.

Тестування програмного забезпечення це один з найвитратніших етапів розробки програмного забезпечення.

В якості інструменту для тестування роботи сайту веб-орієнтованої системи доцільно використовувати Codeception. Codeception – це фреймворк для тестування веб-додатків, що включений в базовий набір фреймворку Yii2/basic і відповідно не потребує додаткового встановлення.

Використовуючи Codeception створимо сценарій для виконання тесту. Тест має імітувати дії користувача, тобто має відбуватися автоматичне натискання на посилання, повиненн вводитися текст, здійснюватися перехід по сторінках. Такі автоматичні тести, що заміняють ручне тестування, прискорюють розробку програмного забезпечення. Для запуску тесту використаємо команду «codecept run».

Протестуємо функцію авторизації користувача на сайті:

```
 public function testLoginCorrect(){
$this->model = new LoginForm(
                        [
    'username' => 'demo',
    'password' => 'demo',
]
 );
expect_that(
 $this->model->login()
 );
expect_not(\Yii::$app->user->isGuest);
expect($this->model->errors)->hasntKey('password');
}.
```

Наступний крок – тестування методів модуля користувача:

```
public function testFindUserById(){
   expect_that($user = User::findIdentity(100));
   expect($user->username)->equals('admin');
   expect_not(User::findIdentity(999));
}
public function testFindUserByUsername(){
   expect_that($user = User::findByUsername('admin'));
   expect_not(User::findByUsername('not-admin'));
}
```

```
public function testValidateUser($user){
    $user = User::findByUsername('userfirst');
    expect_that($user->validateAuthKey('test100key'));
    expect_not($user->validateAuthKey('test102key'));
    expect_that($user->validatePassword('admin'));
    expect_not($user->validatePassword('123456'));}.
```

Створенні тести запускаємо на виконання командою codecept run tests/unit. Результати проведеного автоматичного тестування наведені на рисунку 4.1.

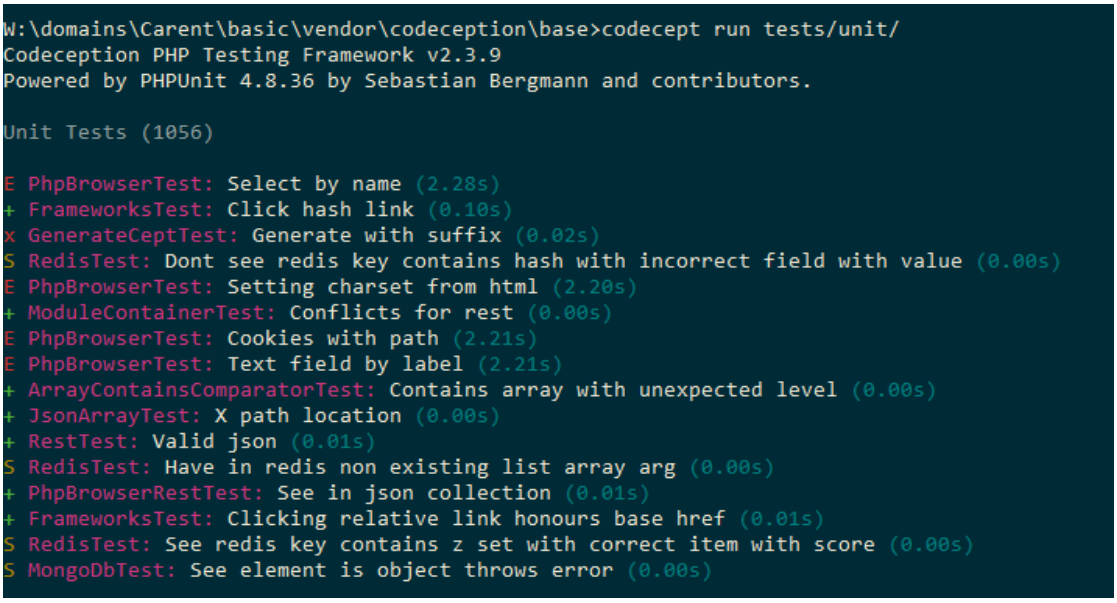


Рисунок 4.1 – Виконання тестування

За результатами проходження UNIT тестування сайту веб-орієнтованої системи отримані наступні дані: помилок не виявлено, система впроралася з поставленої задачею. Після виконання автоматичного тестування перейдемо до ручного тестування системи.

4.2 Перевірка коректності даних

Перевірка коректності даних передбачає роботу із забезпечення запобіганню помилок під час безпосередньої маніпуляції з даними, що проводяться користувачами системи.

Перевіримо на унікальність дані в таблицях бази даних, а саме:

– відповідність зазначених даних визначеним типам даних,

– співпадіння паролів при реєстрації клієнта тощо.

Таким чином буде здійснена перевірка коректності роботи функції авторизації в системі як за допомогою мобільного додатку, так і при авторизації на сайті – звичайних користувачів і користувачів з правами адміністратора. Це дозволить запобігти можливим помилкам при некоректному введені користувацьких даних та несанкціонованому доступу до адміністративної частини системи. Що, в свою чергу, попередить витік персональних даних користувачів розроблюваної системи.

На рисунку 4.2 наведений процес тестування коректності роботи валідаторів при авторизації на сайті та в мобільному додатку. Перевірка коректності даних підтвердила коректну роботу веб-орієнтованої системи.

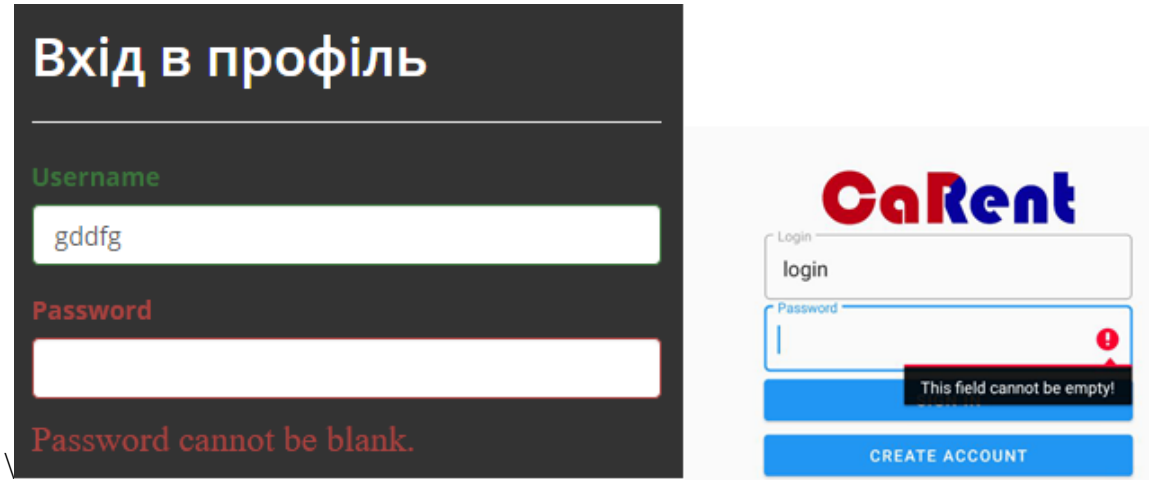


Рисунок 4.2 – Перевірка роботи функції авторизації на сайті та в мобільному додатку

Виконаємо тестування роботи мобільного додатку при відсутності зв’язку з сервером. Сервер був попередньо вимкнений. При відсутності з’єднання відкривається вікно з повідомленням щодо відсутності з’єднання (рисунок 4.3).

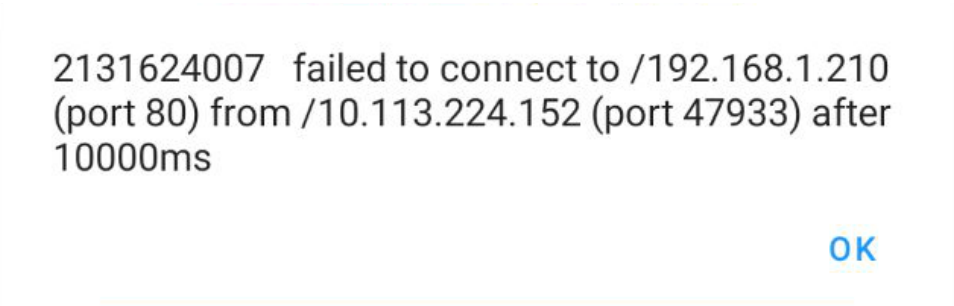


Рисунок 4.3 – Перевірка з’єднання з сервером при роботі з мобільним додатком

Протестуємо функцію бронювання автомобіля на рахунок можливості бронювання на певну дату. Згідно, правил роботи сервісу прокату автомобілів, автомобіль можна забронювати на термін до 3 тижнів. При необхідності більш тривалого прокату здійснюється повторне бронювання. При виконанні бронювання користувач не може мати можливості вибору дати, що передує сьогодняшній даті. Крім того, не має бути можливості бронювання автомобіля довше ніж на 3 тижня.

Проведемо перевірку введення коректної дати, щоб користувач не міг обрати дату старту бронювання у минулому часі (рисунок 4.4).

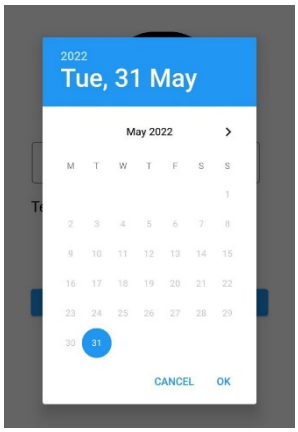


Рисунок 4.4 – Валідація введення коректної дати

Отже, проведене тестування веб-орієнтованої системи на працездітність та реакцію на помилки. Отриманні результати тестування є задовільними. Веб-орієнтована системи працює без збоїв. Функціонал сайту та мобільного додатку працює коректно. Загалом тестування програмного забезпечення показало повну відповідність поставленому технічному завданню.

4.3 Інструкція користувача з інсталяції веб-орієнтованої системи для забезпечення роботи сервісу прокату автомобілів

4.3.1 Інсталяція мобільного додатку

Мобільний додаток призначений для використання на смартфонах та планшетах, що працюють під операційною системою Android та мають наступні технічні характеристики:

- операційна система Android 9.0;
- тактова частота процесора 1.0 ГГц;
- оперативна пам’ять 1 ГБ;
- вільне місце на диску 30 МБ.

Для інсталяції мобільного додатку необхідно скачати на пристрій установочний пакет та запустити його до виконання через провідник. Далі користувач надає згоду на встановлення додатку на пристрої. Час встановлення програми – від 5 секунд залежно від потужності пристрою. На рисунку 4.5 наведений процес інсталяції додатку.

Якщо процес інсталяції успішно завершений, то запуск мобільного додатку на пристрої здійснюється через відповідний ярлик на робочому столі пристрою або за допомогою кнопки Open в інсталяторі програми.

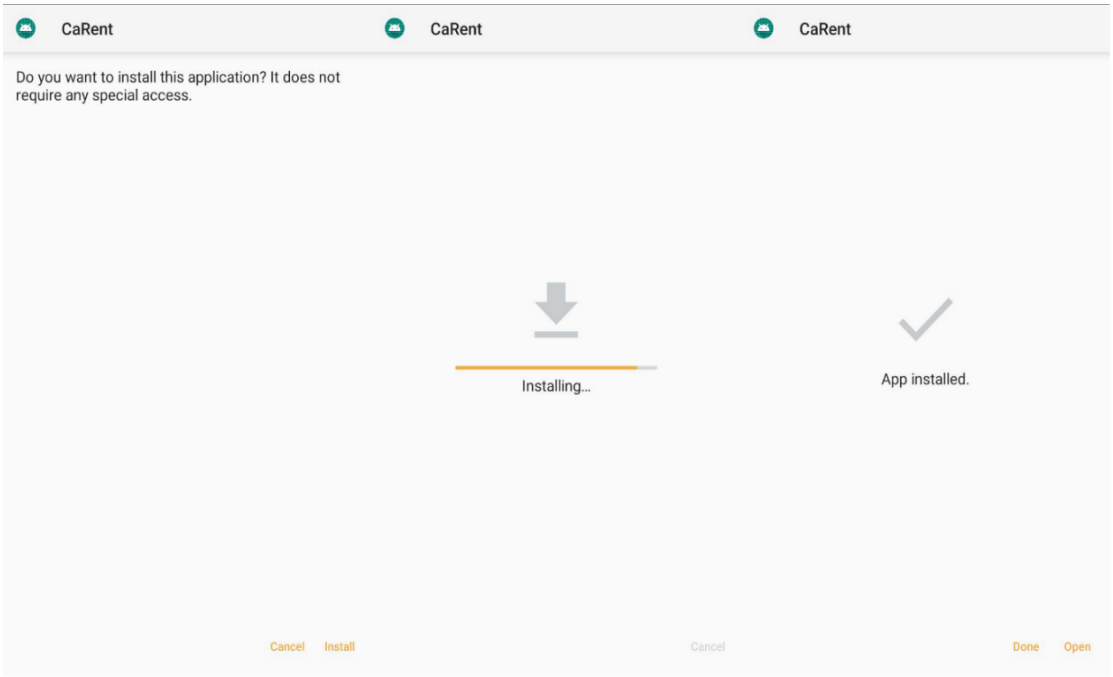


Рисунок 4.5 – Встановлення мобільного додатку

Більш складною задачею є встановлення сайту веб-орієнтованої системи на сервер.

4.3.2 Встановлення сайту веб-орієнтованої системи на сервер

Для успішного встановлення сайту на сервер потрібно забезпечити такі мінімальні вимоги для серверних систем:

– операційна система XP SP2, Mac OS X 10.6, Ubuntu 10.04;
– процесор Intel Pentium 4/Athlon 64;
– пам'ять на диску 4ГБ;
– оперативна пам’ять 1024МБ;
– підтримка PHP 5.4;
– підтримка Apache 2.4.

Проілюструємо процес встановлення сайту веб-орієнтованої системи на локальний сервер за допомогою програми OpenServer.

Першим кроком для запуску сайту у OpenServer є імпорт бази даних у додатку phpMyAdmin або міграція через консоль. Імпорт бази даних у phpMyAdmin показано на рисунку 4.6.

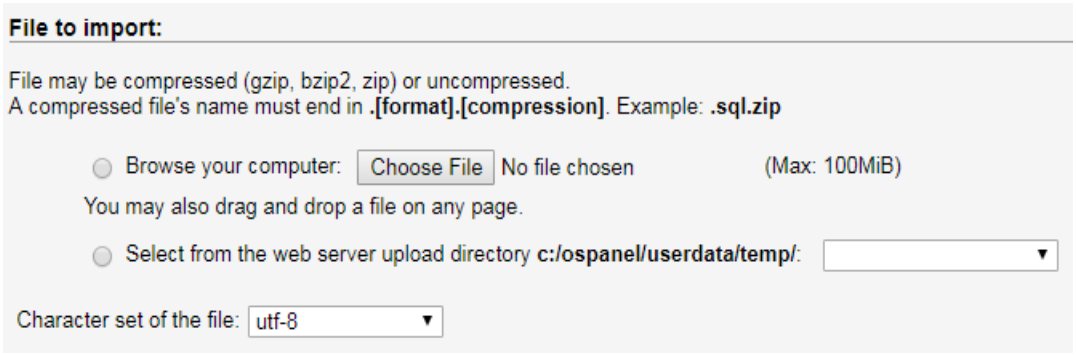


Рисунок 4.6 – Імпорт бази даних у phpMyAdmin

Другий крок – це перенесення папки сайту у папку domains, що знаходиться в кореневому каталозі програми OpenServer. Це забезпечить доступність сайту за адресою, аналогічною назві папки сайту. Проте цю адресу можна налаштувати. Для цього в налаштуванні програми OpenServer на вкладці «Домени» створюємо власний домен (рисунок 4.7).

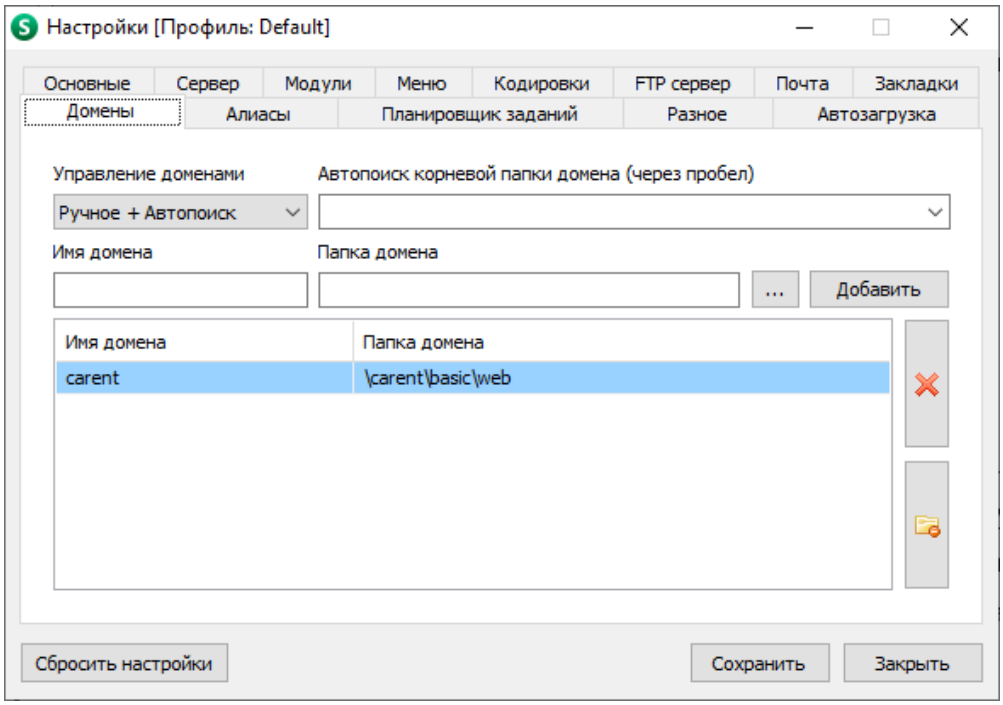


Рисунок 4.7 – Створення власного домену у OpenServer

Наступний крок це зміна хоста, назви бази даних та пароля у файлі конфігурації db.php:

```
return [
    'class' => 'yii\db\Connection',
    'dsn' => 'mysql:host=’’;dbname=’’',
    'username' =>,
    'password' => '',
    'charset' => 'utf8',
];.
```

Далі потрібно перезапустити сервер. Якщо в назвах файлів та бази даних не було змін, то нічого робити не потрібно. Розроблений сайт може бути запущений на будь-якому сервері. Загальна послідовність кроків залишається незмінною. При завантаженні сайту на хостинг додатково налаштовуються права та дозволи, що вимагаються конкретним хостингом.

Після завершення процесу інсталяції всіх складових веб-орієнтованої системи для забезпечення роботу сервісу прокату автомобілів, система готова до експлуатації.

ВИСНОВКИ

В процесі написання кваліфікаційної роботи бакалавра поетапно були пройдені всі стадії розробки програмного забезпечення:

– дослідження предметної області та наявного програмного забезпечення;
– постановка задачі;
– визначення вимог;
– проєктування архітектури;
– вибір технологій та методів програмної реалізації;
– програмування та тестування.

В результаті виконання зазначених стадій було розроблено веб-орієнтовану систему для забезпечення роботи фірми з прокату автомобілів. Ця система, може використовуватися клієнтом сервісу прокату автомобілів, як через сайт, так і через мобільний додаток. Що, в свою чергу розширює мобільність та ефективність системи.

Розроблена веб-орієнтована система має такі переваги:

– зручний графічний інтерфейс;
– повний функціонал для здійснення бронювання та замовлення автомобілів клієнтами;
– можливість блокування ненадійних клієнтів;
– гнучка система ціноутворення залежно від строків прокату автомобіля.

Недоліками розробленої веб-орієнтованої системи є:

– відсутність відслідковування автомобілів за допомогою карт;
– відсутність окремого додатку для iOS пристроїв.

Тестування веб-орієнтованої системи підтвердило працездатність розробленого програмного забезпечення та відповідність поставленому технічному завданню.

Отже, розроблена веб-орієнтована система має достатньо функціоналу та переваг, щоб вважатись повноцінним продуктом, що може забезпечувати роботу сервісу прокату автомобілів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Car Rental – rent a new Car & Van Worldwide [Електронний ресурс] / Веб-сайт EuropCar. – Режим доступу: https://www.europcar.com/. – Назва з екрану.
2. Navaran: Rent a Car – Car Rental Company in Iran [Електронний ресурс] / Веб-сайт Navaran. – Режим доступу: https://en.navaran.com/. – Назва з екрану.
3. Прокат авто Київ, оренда машин від економ до…. [Електронний ресурс] / Веб-сайт Seven Cars. – Режим доступу: https://7cars.com.ua/. – Назва з екрану.
4. Оренда авто Київ [Електроний ресурс] / Веб-сайт BLS. – Режим доступу: https://bls.ua/ua. – Назва з екрану.
5. Прокат автомобілів в Україні та Грузії – компанія Race [Електронний ресурс] / Веб-сайт компанії Race – Режим доступу: https://www.race.net.ua. – Назва з екрану.
6. Шаховська Н. Б. Проектування інформаційних систем : навч. пос. / Н. Б. Шаховська, В. В. Литвин. – Львів : Магнолія-2006, 2011. – 380 с.
7. Новицький О.В. Розширення UML специфікації для моделювання семантичних об’єктів. Проблеми програмування. 2016. N 2–3. С. 211–219.
8. Буч. Г., Рамбо Д., Джекобсон А. Мова UML / Керівництво користувача: пер.с англ. - М .: ДМК, 2000 - 432 с .
9. Что такое API? [Електронний ресурс] / Веб-сайт Amazon. – Режим доступу: https://aws.amazon.com/ru/what-is/api/. – Назва з екрану.
10. REST API – Основи веб-програмування [Електронний ресурс] – Режим доступу: https://lectureswww.readthedocs.io/6.www.sync/3.framework/pyramid/5.1.rest.html/. – Назва з екрану.
11. Підручник з нормалізації баз даних [Електронний ресурс] – Режим доступу: https://uk.myservername.com/java-map-interface-tutorial-with-implementation-examples. – Назва з екрану.

12. Булатецька Л. В. Мова запитів SQL : текст лекцій / Л. В. Булатецька, В. В. Булатецький. – Луцьк : СНУ імені Лесі Українки, 2018. – 92 с.

13. Джеймс Р. Грофф, Пол Н. Вайнберг, Эндрю Дж. Оппель – SQL. Полное руководство. Третье издание. – «Вильямс» – Киев, 2015 – С. 10 –14.

14. Фреймворк Yii2 [Електронний ресурс] – Режим доступу: https://www.yiiframework.com. – Назва з екрану.

15. 13 мов програмування, що будуть домінувати у 2022 році. [Електронний ресурс] – Режим доступу: https://careers.easternpeak.com/blog/popular-programming-languages/. – Назва з екрану.

16. Опорний конспект лекцій з дисципліни «Сучасні інструментальні засоби розробки користувацького інтерфейсу» / Р.П.Шевчук – Тернопіль, 2012. – 103 с.

17. Что такое модель-представление-контроллер (MVC) Framework? Модель MVC с анализом устойчивости UML [Електронний ресурс] – Режим доступу: https://www.cybermedian.com/ru/what-is-model-view-controller-mvc-framework-model-mvc-with-uml-robustness-analysis/ – Назва з екрану.

18. Авраменко А.С. Тестування програмного забезпечення: навчальний посібник / А. С. Авраменко, В. С. Авраменко, Г.В. Косенюк– Черкаси: ЧНУ імені Богдана Хмельницького, 2017. – 284 с.

19. Дипломний проект : методичні вказівки щодо його виконання для студентів-бакалаврів спеціальності «Інженерія програмного забезпечення» / Л. П. Бедратюк, Г. І. Радельчук, Ю. В. Форкун, О. М. Ящина. – Хмельницький : ХНУ, 2020. – 79 с.

ДОДАТОК А

(обов’язковий)

ТЕХНІЧНЕ ЗАВДАННЯ

Введення

Технічне завдання на розробку веб-орієнтованої системи для забезпечення роботи сервісу прокату автомобілів

1 Підстава для розробки

Підставою для розробки веб-орієнтованої системи для забезпечення роботи сервісу прокату автомобілів є завдання на дипломне проектування, затверджене кафедрою інженерії програмного забезпечення Хмельницького національного університету

Найменування розробки: Веб-орієнтована система для забезпечення роботи сервісу прокату автомобілів.

2 Призначення розробки

Розробка призначена для забезпечення роботи сервісів з прокату автомобілів в частині взаємодії з клієнтами при виборі та бронюванні автомобіля. Метою розробки є створення веб-орієнтованої системи для забезпечення роботи сервісу прокату автомобілів, яка буде складатися з сайту та мобільного додатку, що буде сприяти підвищенню конкурентоспроможності сервісу на ринку та збільшенню привабливості для клієнтів.

3 Вимоги до програми

3.1 Вимоги до функціональних характеристик

Веб-орієнтована система для забезпечення роботи сервісу прокату автомобілів має складатися з двох частин: сайту та мобільного додатку.

Програмне забезпечення повинно забезпечувати виконання наступних основних функції:

Для користувача:

– реєструватися в системі;
– авторизуватися в системі;
– здійснювати перегляд наявних в сервісі автомобілів;
– бронювати автомобілі;
– відслідковувати історію бронювань.

Для адміністратора:

– вводити та редагувати данні про автомобілі;

– переглядати здійснені клієнтами бронювання;

– редагувати статуси бронювань клієнтів;

– блокувати користувачів;

– видаляти неактуальні данні з системи.

Розробку виконати для використання на персональних комп’ютерах під управлінням операційної системи сімейства Windows.

3.2 Вимоги до надійності

Передбачити контроль введення інформації.

Передбачити захист від некоректних дій користувача.

3.3 Умови експлуатації:

Не висуваються.

3.4 Вимоги до складу і параметрів технічних засобів

Програмне забезпечення повинно функціонувати на персональних комп‘ютерах під управлінням ОС Windows та на планшетах і смартфонах з операційною системою Android.

Наявність вільної дискової пам’яті для встановлення веб-орієнтованої системи.

3.5 Вимоги до інформаційної та програмної сумісності

Програмне забезпечення повинно працювати під управлінням операційних систем сімейства Windows 7 або вище на персональних комп’ютерах, та операційної системи Android на планшетах і смартфонах.

Вхідні дані до кабінету користувача мають заноситися користувачем та веріфікуватися адміністратором.

3.6 Вимоги до маркування та пакування.

Вимоги до маркування та пакування не пред‘являються.

3.7 Вимоги до транспортування та зберігання.

Вимоги до транспортування та зберігання не пред'являються.

3.8 Спеціальні вимоги.

Згенерувати установчу версію програмного забезпечення.

5 Вимоги до програмної документації

Програмні модулі, які розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

У склад супроводжувальної документації повинні входити наступні документи:

– пояснювальна записка не менше ніж на 60 аркушах формату А4 (без додатків);

– технічне завдання;

– інструкція користувача з інсталяції веб-орієнтованої системи для забезпечення роботу сервісу прокату автомобілів.

Графічна частина повинна бути виконана у вигляді слайдів та містити наступні схеми та діаграми:

– діаграма варіантів використання;

– діаграма класів;

– діаграми послідовностей для клієнта та адміністратора;

– діаграми співпраці для клієнта та адміністратора;

– діаграми станів для клієнтської частини та адміністративної частини;

– діаграма компонентів;

– схема бази даних.

ДОДАТОК Б

(обов’язковий)

ДІАГРАМИ ПОСЛІДОВНОСТЕЙ

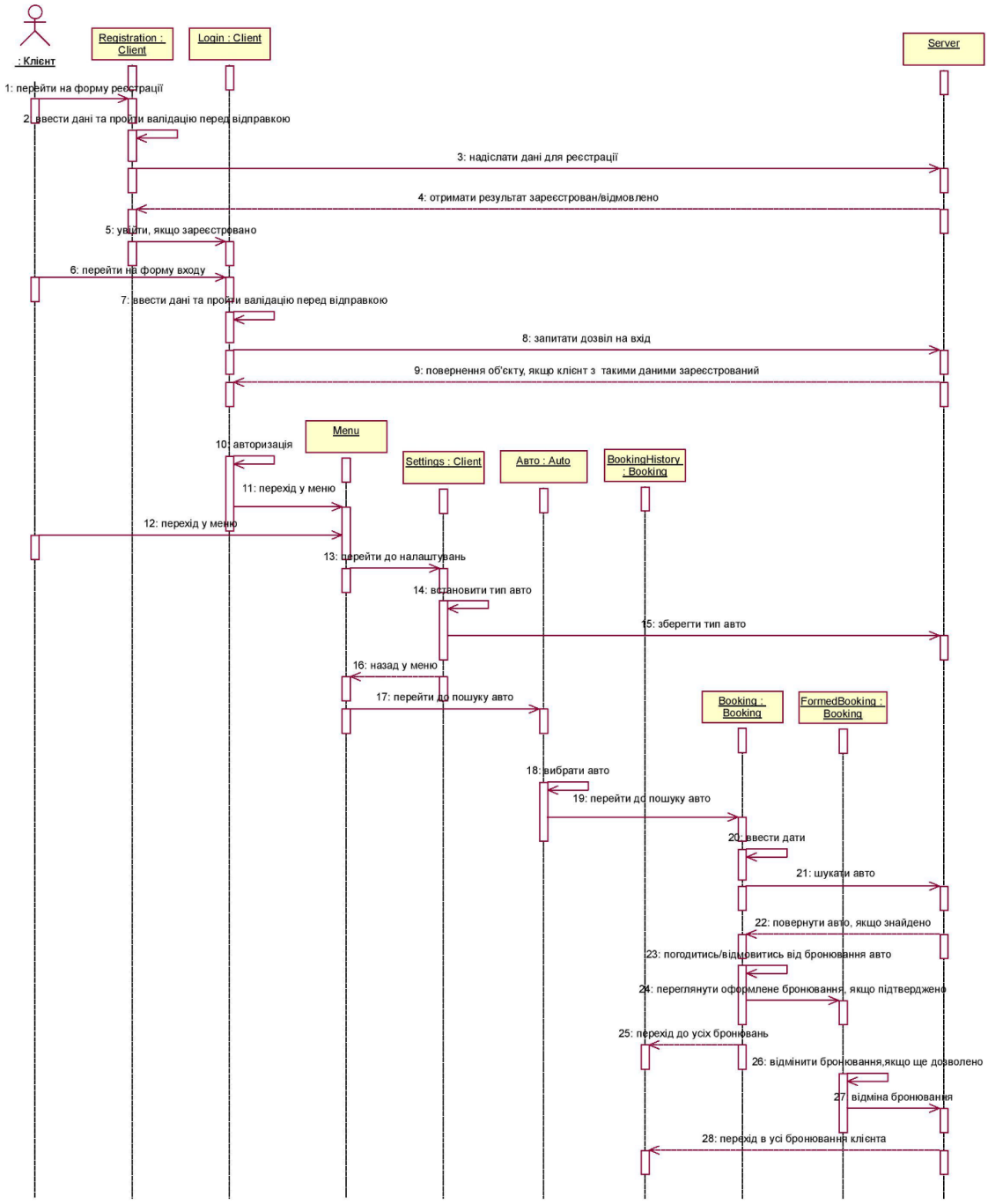


Рисунок Б.1 – Діаграма послідовностей для мобільного додатку

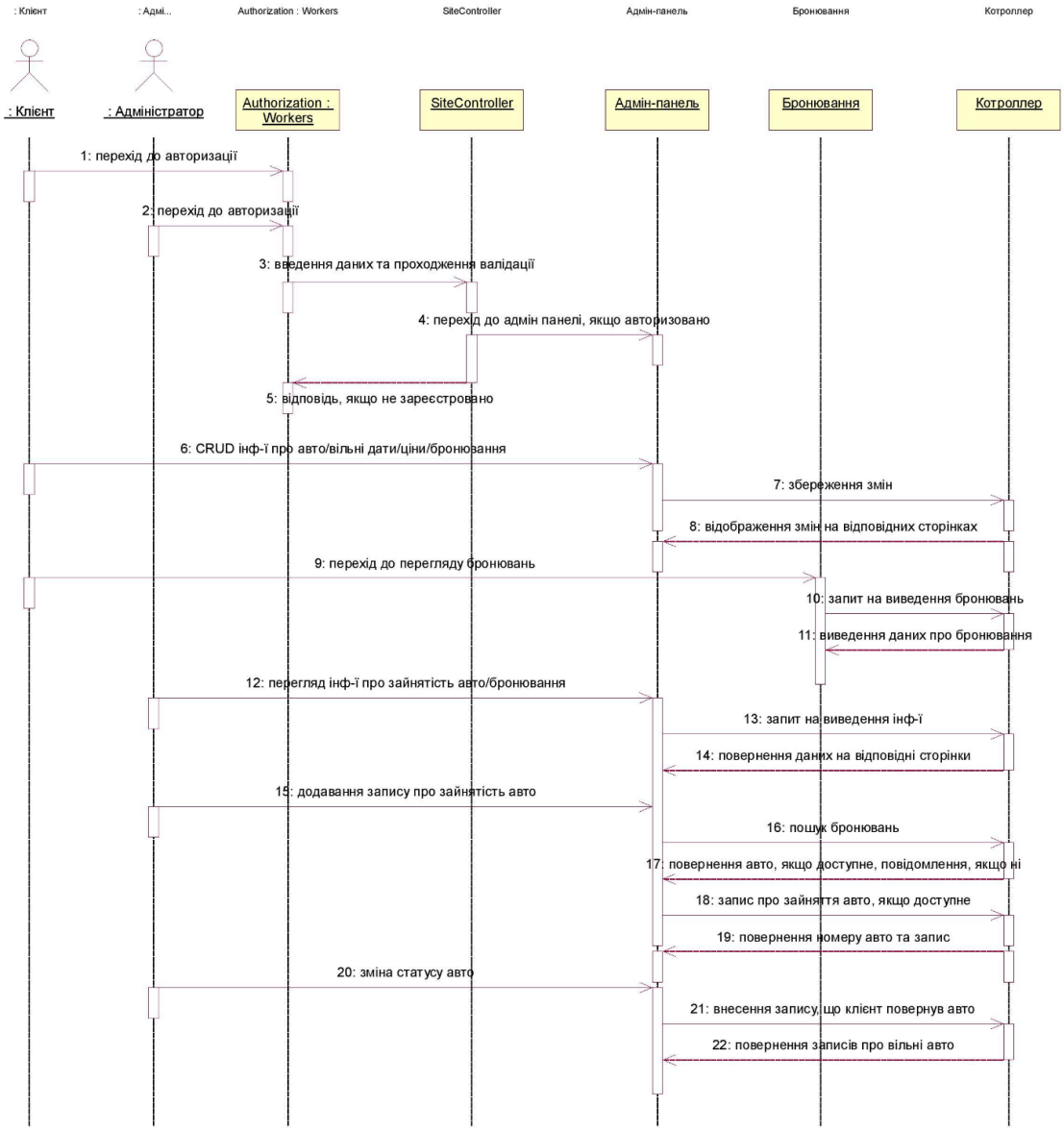


Рисунок Б.2 – Діаграма послідовностей для веб сайту

ДОДАТОК В

(обов’язковий)

ДІАГРАМИ СПІВПРАЦІ

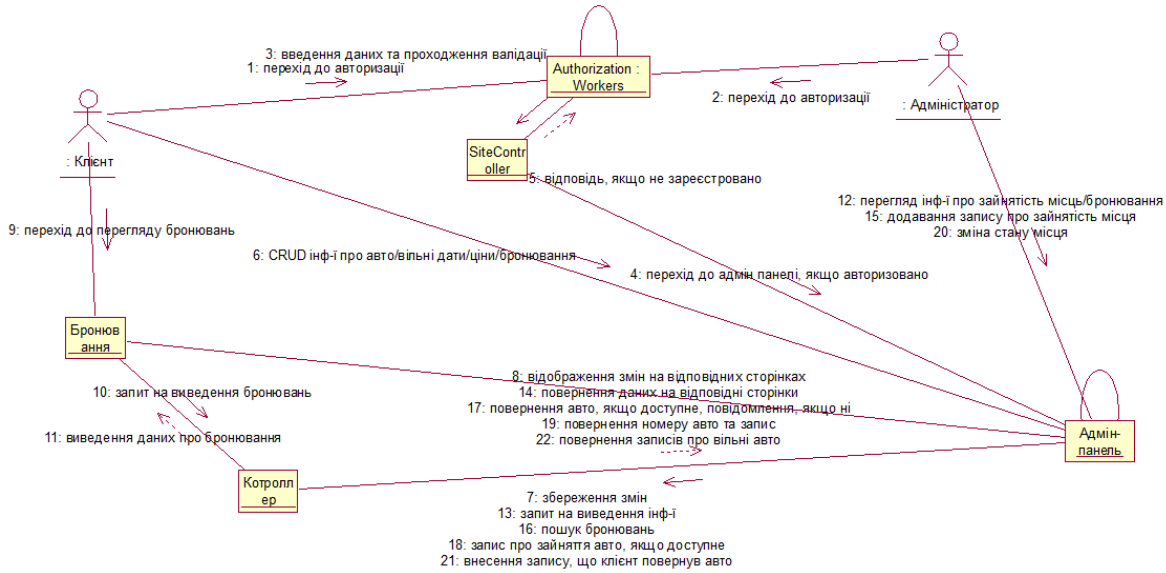


Рисунок В.1 — Діаграма співпраці для мобільного додатку

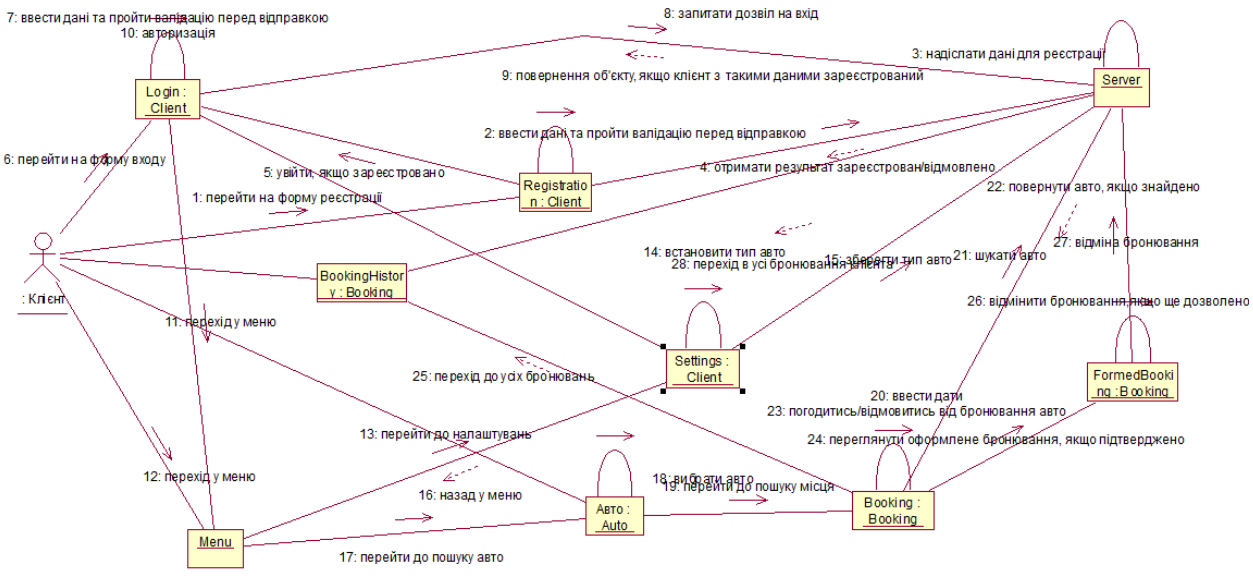


Рисунок В.2 — Діаграма співпраці для веб сайту

ДОДАТОК Г

(обов’язковий)

ДІАГРАМИ СТАНІВ

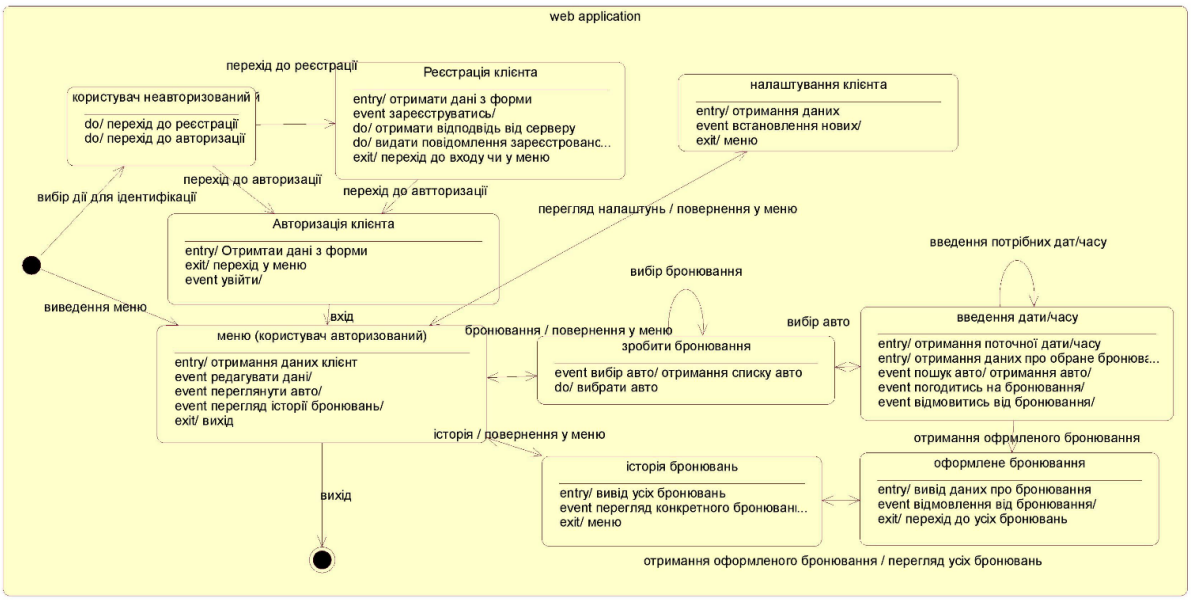


Рисунок Г.1 — Діаграма станів для мобільного додатку

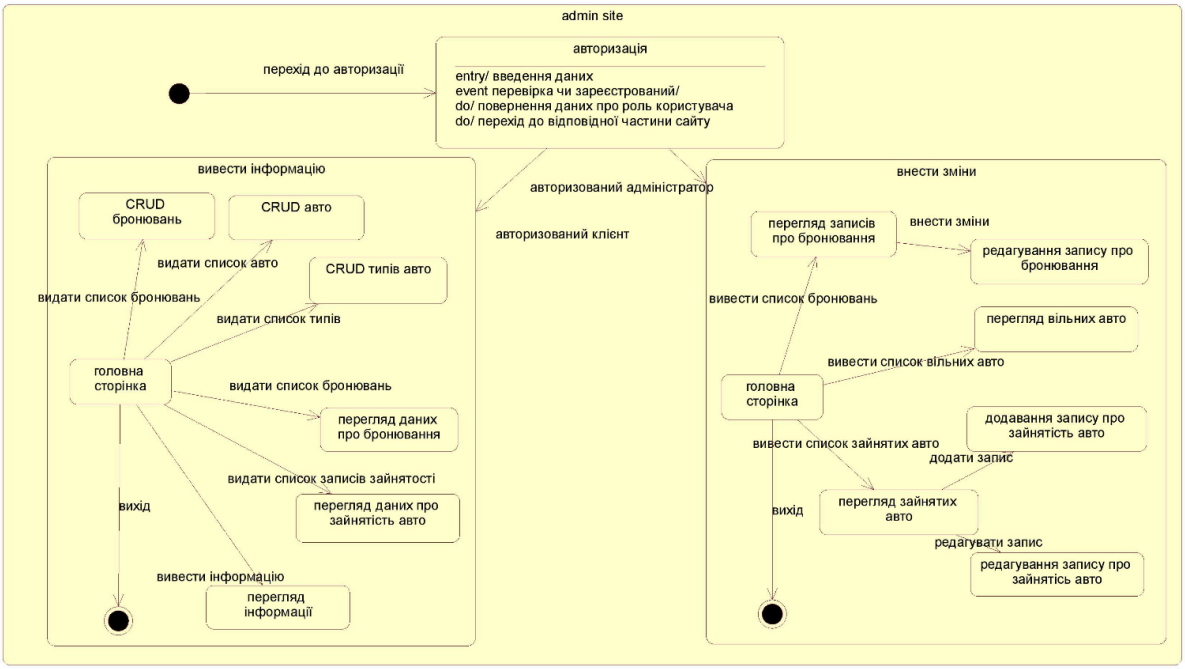


Рисунок Г.2 — Діаграма станів для веб-сайту

ДОДАТОК Д

(обов’язковий)

ДІАГРАМА АКТИВНОСТІ

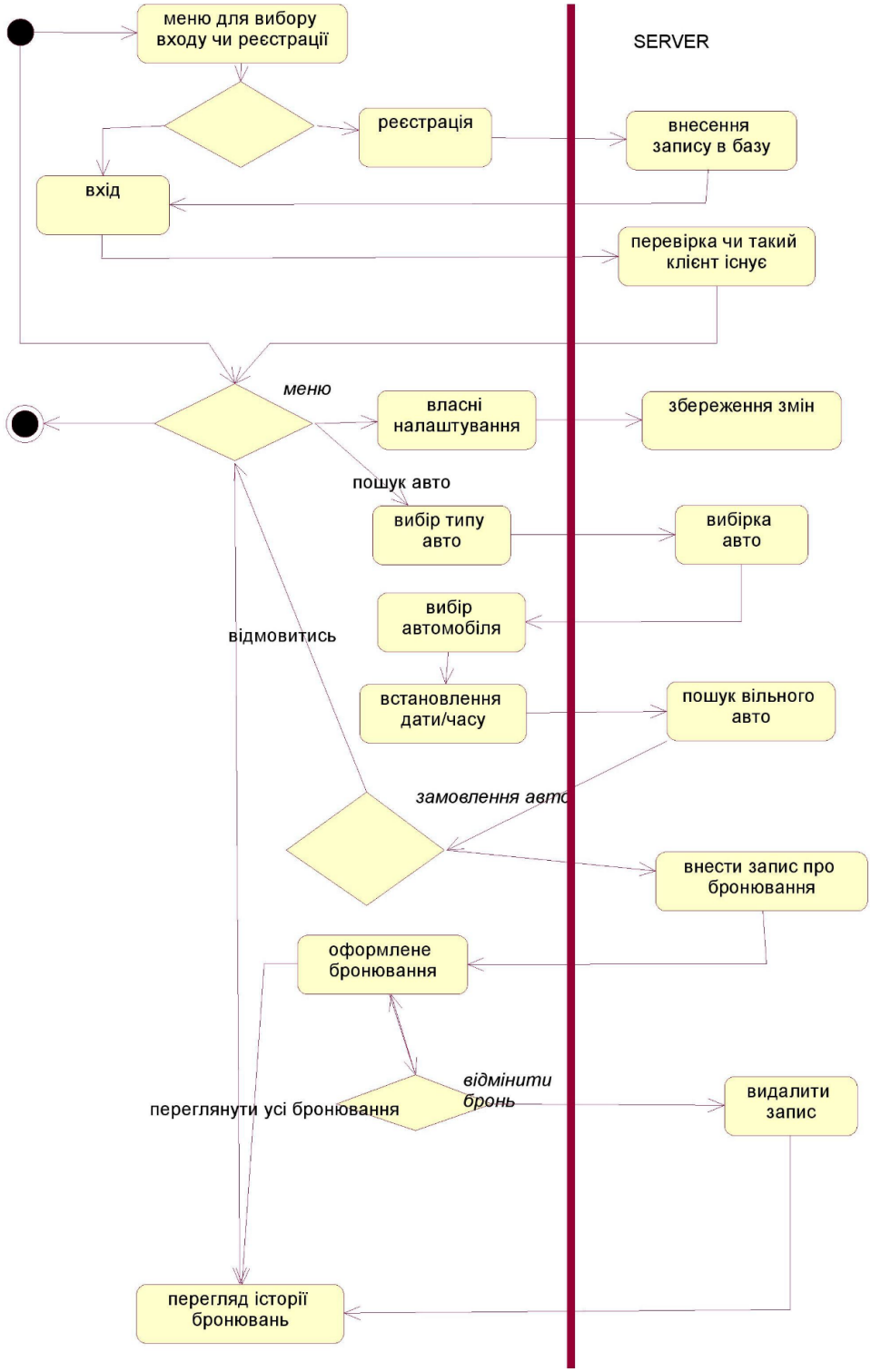


Рисунок Д.1 — Діаграма активності

ДОДАТОК Е

(обов’язковий)

ДІАГРАМА КОМПОНЕНТІВ

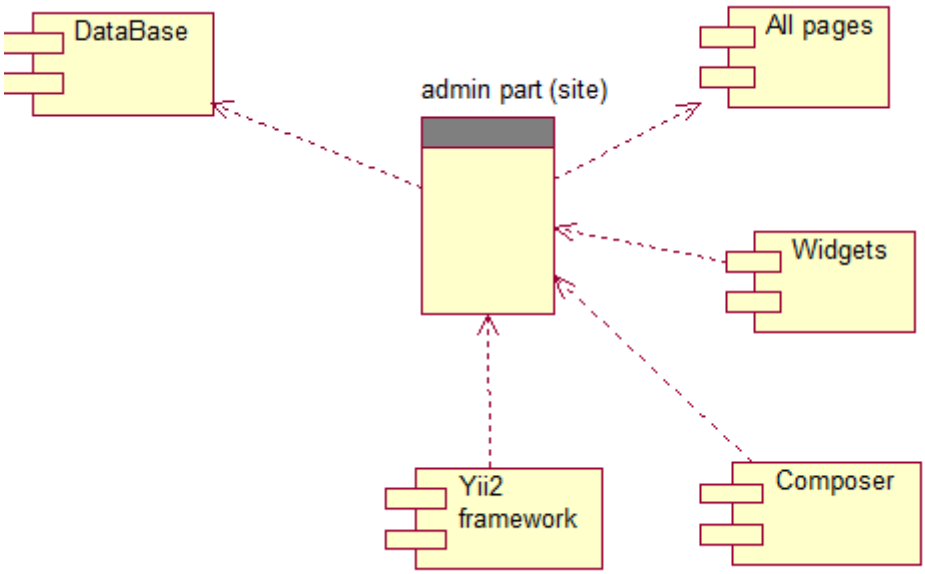


Рисунок Е.1 — Діаграма компонентів

ДОДАТОК Є

(обов’язковий)

ФРАГМЕНТ ПРОГРАМНОГО КОДУ

Є.1 Моделі системи для забезпечення роботи сервісу з прокату автомобілів

```
'model' => $model,
                'parkings' => $parkings,
                'parkingsCount' => $parkingsCount,
                'dateBegin' => $model->dateBegin,
                'dateEnd' => $model->dateEnd,
                'city' => Cities::findOne($cityId)
            ]);
        }
        $model->dateEnd = date('Y-m-d');
        $model->dateBegin = date('Y-m-01');

        return $this->render('graph', [
            'model' => $model,
            'parkings' => $parkings,
            'parkingsCount' => $parkingsCount
<?php
namespace app\models;
use Yii;
class Userinfo extends \yii\db\ActiveRecord{
    public static function tableName(){
        return 'userinfo';
    }
    public function rules(){
        return [
            [['user'], 'integer'],
            [['license_date'], 'safe'],
            [['first_name', 'middle_name', 'second_name', 'license_id', 'photo_license', 'photo_user',
'photo_passport', 'phone_number'], 'string', 'max' => 255],
            [['user'], 'exist', 'skipOnError' => true, 'targetClass' => User::className(), 'targetAttribute'
=> ['user' => 'id']],
        ];
    }
    public function attributeLabels(){
        return [
            'id' => 'ID',
            'first_name' => 'First Name',
            'middle_name' => 'Middle Name',
            'second_name' => 'Second Name',
            'user' => 'User',
```

```
            'license_id' => 'License ID',

            'license_date' => 'License Date',
            'photo_license' => 'Photo License',
            'photo_user' => 'Photo User',
            'photo_passport' => 'Photo Passport',
            'phone_number' => 'Phone Number',
        ];
    }
    public function getUser0(){
        return $this->hasOne(User::className(), ['id' => 'user']);
    }
    public function getPicture(){
        return ($this->photo_user) ? '/uploads/User/' . $this->photo_user : '../../no_image.png';
    }
    public function saveImage($filename){
        $this->photo_user = $filename;
        return $this->save(false);
    }
}
```

```
<?php
namespace app\models;
class User extends \yii\db\ActiveRecord implements IdentityInterface{
    public $currentPassword;
    public $newPasswordConfirm;
    public $newPassword;
    public function rules(){
        return [
            [['role', 'active'], 'integer'],
            [['login', 'email'], 'string', 'max' => 255],
            [['newPassword','currentPassword','newPasswordConfirm'], 'required'],
            [['currentPassword'],'validateCurrentPassword'],
            [['newPassword','newPasswordConfirm'], 'string', 'min'=>3],
            [['newPassword','newPasswordConfirm'], 'filter', 'filter'=>'trim'],
            [['newPasswordConfirm'], 'compare', 'compareAttribute'=>'newPassword', 'message' =>
'Паролі не співпадають'],
        ];
    }
    public function validateCurrentPassword(){
        if(!$this->verifyPassword(!$this->currentPassword)){
            $this->addError('currentPassword','Не вірний пароль!');
        }
    }
    public function verifyPassword($password){
        $dbpassword = static::findOne(['id' => Yii::$app->user->id])->password;
        if($password==$dbpassword)
            return true;
        else return false;
    }
    public function attributeLabels(){
        return [
```

```
            'id' => 'ID',
            'login' => 'Login',
            'email' => 'Email',
            'password' => 'Password',
            'role' => 'Role',
            'active' => 'Active',
        ];
    }
}
```

```
<?php
namespace app\models;
use Yii;
use yii\base\Model;
use yii\web\UploadedFile;
class ImageUpload extends Model{
    public $image;
    public $folder;
    public function rules(){
        return [
            [['image'],'required'],
            [['image'], 'file', 'extensions' => 'jpg,png']
        ];
    }
    public function uploadImage(UploadedFile $file, $currentImage){
        $this->image = $file;
        if($this->validate()){
            $this->deleteCurrentImage($currentImage);
            return $this->saveImage();
        }
    }
    private function getFolder(){
        if($this->folder=='car'){
            return Yii::getAlias('@web').'uploads/CarModels/';
        }
        if($this->folder=='user'){
            return Yii::getAlias('@web').'uploads/User/';
        }
    }
    private function generateFileName(){
        return strtolower(md5(uniqid($this->image->baseName)) . '.' . $this->image->extension);
    }
    public function deleteCurrentImage($currentImage){
        if($currentImage!=null)
        if(file_exists($this->getFolder().$currentImage)) {
            unlink($this->getFolder() . $currentImage);
        }
    }
    private function saveImage(){
        $filename = $this->generateFileName();
        $this->image->saveAs($this->getFolder().$filename);
        return $filename;
```

```
    }
}
```

```
<?php
namespace app\models;
use Yii;
class CarModel extends \yii\db\ActiveRecord{
    public static function tableName(){
        return 'car_model';
    }
    public function rules(){
        return [
            [['category', 'price', 'seats', 'doors', 'air_conditioning', 'count'], 'integer'],
            [['fuel_consumption'], 'number'],
            [['name', 'picture', 'engine_volume', 'engine_type', 'body', 'transmission'], 'string', 'max' =>
255],
            [['category'], 'exist', 'skipOnError' => true, 'targetClass' => Category::className(),
'targetAttribute' => ['category' => 'id']],
        ];
    }
    public function attributeLabels(){
        return [
            'id' => 'ID',
            'category' => 'Category',
            'name' => 'Name',
            'price' => 'Price',
            'picture' => 'Picture',
            'engine_volume' => 'Engine Volume',
            'engine_type' => 'Engine Type',
            'fuel_consumption' => 'Fuel Consumption',
            'seats' => 'Seats',
            'body' => 'Body',
            'doors' => 'Doors',
            'transmission' => 'Transmission',
            'air_conditioning' => 'Air Conditioning',
            'count' => 'Count',
        ];
    }
    public function saveImage($filename){
        $this->picture = $filename;
        return $this->save(false);
    }
    public function getPicture(){
        return ($this->picture) ? '/uploads/CarModels/' . $this->picture : '../../no_vehicle.png';
    }
    public function deleteImage(){
        $imageUploadModel = new ImageUpload();
        $imageUploadModel->deleteCurrentImage($this->picture);
    }
    public function beforeDelete(){
        $this->deleteImage();
        return parent::beforeDelete();
```

```
    }
    public function getCategory(){
        return $this->hasOne(Category::className(), ['id' => 'category']);
    }
    public function saveCategory($category_id){
        $category = Category::findOne($category_id);
        if($category!=null)
        {
            $this->link('category', $category);
            return true;
        }
    }
}
```

```
<?php
namespace app\models;
use Yii;
class Category extends \yii\db\ActiveRecord{
    public static function tableName(){
        return 'category';
    }
    public function rules(){
        return [
            [['experience'], 'integer'],
            [['name'], 'string', 'max' => 255],
        ];
    }
    public function attributeLabels(){
        return [
            'id' => 'ID',
            'name' => 'Name',
            'experience' => 'Experience',];}}
```

```
<?php
namespace app\models;
use Yii;
class Order extends \yii\db\ActiveRecord{
    public static function tableName(){
        return 'order';
    }
    public function rules(){
        return [
            [['car_id','user_id','term','date'],'required'],
            [['car_id', 'user_id', 'term', 'active', 'price'], 'integer'],
            [['date'], 'default', 'value' => 1552700880],
            [['term'], 'integer', 'min' => 1, 'max' => 20],
            [['price'], 'required'],
            [['car_id'], 'exist', 'skipOnError' => true, 'targetClass' => Car::className(),
'targetAttribute' => ['car_id' => 'id']],
            [['user_id'], 'exist', 'skipOnError' => true, 'targetClass' => User::className(),
'targetAttribute' => ['user_id' => 'id']],
        ];
    }
```

```
    public function attributeLabels(){
        return [
            'date' => 'Дата початку оренди',
            'term' => 'Термін оренди (днів)',
        ];}
    public function getCar(){
        return $this->hasOne(Car::className(), ['id' => 'car_id']);}
    public function getUser(){
        return $this->hasOne(User::className(), ['id' => 'user_id']);}}
```

Фрагменти контролерів

```
<?php
class AuthController extends Controller{
    public function actionLogin(){
        if (!Yii::$app->user->isGuest) {
            return $this->goHome();
        }
        $model = new LoginForm();
        if ($model->load(Yii::$app->request->post()) && $model->login()) {
            return $this->goBack();
        }
        return $this->render('login', [
            'model' => $model,
        ]);
    }
    public function actionLogout(){
        Yii::$app->user->logout();
        return $this->goHome();
    }
    public function actionSignup(){
        $model = new SignupForm();
        if(Yii::$app->request->isPost){
            $model->load(Yii::$app->request->post());
            if($model->signup()){
                return $this->redirect(['../site/index']);
            }
        }
        return $this->render('signup', ['model'=>$model]);}
    public function actionTest(){
        echo Yii::$app->user->isGuest;
        echo Yii::$app->user->identity->login;
    }
}
```

```
    public function actionSetpimage(){
        if(!Yii::$app->user->isGuest){
            $model = new ImageUpload;
            $model->folder='user';
            if(Yii::$app->request->isPost){
                $userinfo = Userinfo::find()->where(['user'=>Yii::$app->user->id])->one();
                $file = UploadedFile::getInstance($model, 'image');
                if($userinfo->savepImage($model->uploadImage($file, $userinfo->photo_passport)))
```

```
            {
                $this->redirect('userpage');
            }
        }
        return $this->render('userimage', ['model'=>$model]);
    }
    else
        return $this->redirect('../auth/login');
}
public function actionSetdimage(){
    if(!Yii::$app->user->isGuest){
        $model = new ImageUpload;
        $model->folder='user';
        if(Yii::$app->request->isPost){
            $userinfo = Userinfo::find()->where(['user'=>Yii::$app->user->id])->one();
            $file = UploadedFile::getInstance($model, 'image');
            if($userinfo->savedImage($model->uploadImage($file, $userinfo->photo_license)))
                $this->redirect('userpage');
        }
        return $this->render('userimage', ['model'=>$model]);
    }
    else
        return $this->redirect('../auth/login');
}
public function actionEditprofile(){
    if(!Yii::$app->user->isGuest){
        $userinfo = Userinfo::find()->where(['user'=>Yii::$app->user->id])->one();
        if ($userinfo->load(Yii::$app->request->post()) && $userinfo->save()) {
            return $this->redirect(['userpage']);
        }
        return $this->render('editprofile', [
            'userinfo' => $userinfo,
        ]);
    }
    else
        return $this->redirect('../auth/login');
}}

<?php
class CarmodelController extends Controller{
    public function behaviors(){
        return [
            'verbs' => [
                'class' => VerbFilter::className(),
                'actions' => [
                    'delete' => ['POST'],
                ],
            ],
        ];}
    public function actionUpdate($id){
        $model = $this->findModel($id);
        if ($model->load(Yii::$app->request->post()) && $model->save()) {
```

```
            return $this->redirect(['view', 'id' => $model->id]);
        }
        return $this->render('update', [
            'model' => $model,
        ]);
    }
    public function actionSetImage($id){
        $model = new ImageUpload;
        $model->folder='car';
        if(Yii::$app->request->isPost){
            $carmodel = $this->findModel($id);
            $file = UploadedFile::getInstance($model, 'image');
            if($carmodel->saveImage($model->uploadImage($file, $carmodel->picture)))
                $this->redirect(['view','id'=>$carmodel->id]);
        }
        return $this->render('image', ['model'=>$model]);}
    public function actionSetCategory($id){
        $carmodel = $this->findModel($id);
        $currentCategory = $carmodel->category0->id;
        $categories = ArrayHelper::map(Category::find()->all(), 'id', 'name');
        if (Yii::$app->request->isPost){
          $category = Yii::$app->request->post('category');
          if($carmodel->saveCategory($category)){
             return $this->redirect(['view', 'id'=>$carmodel->id]);
          }
          else{
          }}
        return $this->render('category', [
            'carmodel' => $carmodel,
            'selectedCategory' => $currentCategory,
            'categories' => $categories
        ]);
    }}

<?php
class OrderController extends Controller{
    public function behaviors(){
        return [
            'verbs' => [
                'class' => VerbFilter::className(),
                'actions' => [
                    'delete' => ['POST'],
                ],
            ],
        ];
    }
    public function actionCreate(){
        $model = new Order();
        if ($model->load(Yii::$app->request->post()) && $model->save()) {
            return $this->redirect(['view', 'id' => $model->id]);
        }
        return $this->render('create', [
```

```
            'model' => $model,
        ]);
    }
    public function actionUpdate($id){
        $model = $this->findModel($id);
        if ($model->load(Yii::$app->request->post()) && $model->save()) {
            return $this->redirect(['view', 'id' => $model->id]);
        }
        return $this->render('update', [
            'model' => $model,
        ]);
    }
    public function actionDelete($id){
        $this->findModel($id)->delete();
        return $this->redirect(['index']);
    }
    protected function findModel($id){
        if (($model = Order::findOne($id)) !== null) {
            return $model;
        }
        throw new NotFoundHttpException('The requested page does not exist.');
    }
}
```

```
<?php
namespace app\modules\admin\controllers;
class UserController extends Controller{
    public function behaviors(){
        return [
            'verbs' => [
                'class' => VerbFilter::className(),
                'actions' => [
                    'delete' => ['POST'],
                ],
            ],
        ];
    }
    public function actionIndex(){
        $searchModel = new UserSearch();
        $dataProvider = $searchModel->search(Yii::$app->request->queryParams);
        return $this->render('index', [
            'searchModel' => $searchModel,
            'dataProvider' => $dataProvider,
        ]);
    }
    protected function findModel($id){
        if (($model = user::findOne($id)) !== null) {
            return $model;
        }
        throw new NotFoundHttpException('The requested page does not exist.');
    }
    public function actionBan($id){
```

```
        $ban=new Banlist;
        $ban->user=$id;
        $ban->save();
        return $this->redirect('../../admin/banlist/update?id='.$ban->id);
    }
    public function actionVerification($id){
        $user = User::findOne($id);
        if(!$user->active==1)
            $user->active = 1;
        else
            $user->active = 0;
        $user->save(false);
        return $this->redirect(['view', 'id' => $id]);
    }
    public function actionSetadmin($id){
        $user = User::findOne($id);
        if(!$user->role==1)
            $user->role = 1;
        else
            $user->role = 0;
        $user->save(false);
        return $this->redirect(['view', 'id' => $id]);
    }
```

Є.2 Фрагменти API - контролер користувача

```
<?php
class UserController extends ActiveController{
    public $modelClass = 'app\models\User';
    public function actions(){
        $actions = parent::actions();
        unset($actions['uploadphoto']);
        return $actions;
    }
    public function actionSignin($login, $password){
        \Yii::$app->response->format=Response::FORMAT_JSON;
        $user = User::findOne(['login'=>$login, 'password'=>$password]);
        if($user!=null)return $user;
        else return null;
    }
    public function actionCheck($login, $password){
        \Yii::$app->response->format=Response::FORMAT_JSON;
        $user = User::findOne(['login'=>$login, 'password'=>$password]);
        if($user!=null)return $user;
        else return null;
    }
    public function actionSignup($login, $email, $password){
        \Yii::$app->response->format=Response::FORMAT_JSON;
        $user1 = User::findOne(['login'=>$login]);
        $user2 = User::findOne(['email'=>$email]);
        if(($user1==null)&&($user2==null)){
            $user = new User();
```

```
        $user->login = $login;
        $user->email = $email;
        $user->password = $password;
        $user->save(false);
        $newuser = User::findOne(['login'=>$login]);
        $userinfo = new Userinfo();
        $userinfo->user=$newuser->id;
        $userinfo->photo_user="no_image.png";
        $userinfo->save(false);
        return $newuser;
    }
    else return null;
}
public function actionView(){
    \Yii::$app->response->format=Response::FORMAT_JSON;
    $car_model = CarModel::find()->all();
    if(count($car_model)>0){
        return $car_model;
    }
    else {
        return array('status'=>true,'data'=>'No CarModels found.');
    }
}
public function actionSetdate($date, $login)
{
    \Yii::$app->response->format=Response::FORMAT_JSON;
    $user = User::findOne(['login'=>$login]);
    if($user!=null){
        $userinfo = Userinfo::findOne(['user'=>$user->id]);
        if($userinfo !=null)
        {
            $userinfo->license_date=$date;
            $userinfo->save(true);
            return 1;
        }
    }
    else return null;
}
public function actionChangepassword($login, $oldpassword, $newpassword){
    \Yii::$app->response->format=Response::FORMAT_JSON;
    $user = User::findOne(['login'=>$login]);
    if ($user->password==$oldpassword){
        $user->password=$newpassword;
        $user->save(false);
        return true;
    }
    else
        return false;
}
public $documentPath = 'documents/';
public function verbs(){
    $verbs = parent::verbs();
```

```
        $verbs[ "upload" ] = ['POST' ];
        return $verbs;
    }
    public function actionUploadphoto($userid){
        $userinfo = Userinfo::findOne(['user'=>$userid]);
        $userinfo->photo_passport='a';
        if($userinfo!=null){
            $uploads = \yii\web\UploadedFile::getInstancesByName('upfile');
            if (empty($uploads)){
                return false;
                // handle error reporting somewhere else
            }
            $path =  Yii::getAlias('@web').'uploads/User/';
            foreach ($uploads as $upload){
                $filename = time() .'_'. $upload->name ;
                $userinfo->photo_user=$filename;
                $upload->saveAs($path.$filename);
            }
            $userinfo->save(false);
        }
        return false;
    }
}
```

Є.3 Інтерфейс API

```
public interface Api {
    @GET("api/user/signin?")
    Call<User> userLogin(@Query("login") Editable login, @Query("password") Editable pass);
    @GET("api/user/changepassword?")
    Call<Boolean> userChangepassword(@Query("login") String login, @Query("oldpassword")
String oldpass,   @Query("newpassword") String newpass);
    @GET("api/user/check?")
    Call<User> userCheck(@Query("login") String login, @Query("password") String pass);
    @GET("api/user/info?")
    Call<UserInfo> userInfo(@Query("id") int id);
    @GET("api/user/signup?")
    Call<User> userRegistration(@Query("login") CharSequence login, @Query("email")
CharSequence email, @Query("password") CharSequence pass);
    @GET("api/user/setinfo?")
    Call<Boolean> userUpdateinfo(@Query("userid") Integer userid, @Query("email")
CharSequence email,  @Query("firstname") CharSequence firstname, @Query("secondname")
CharSequence secondname, @Query("middlename") CharSequence middlename,
@Query("phonenumber") CharSequence phonenumber, @Query("idlicense") CharSequence
idlicense, @Query("datelicense") CharSequence datelicense);
    @Multipart
    @POST("api/user/uploadphoto")
    Call<ResponseBody> uploadImage(@Part MultipartBody.Part file, @Part("upfile")
RequestBody requestBody, @Query("userid") int userid);
    @GET("api/banlist/getban?")
    Call<Banlist> getBan(@Query("user_id") int user_id);
    @Headers("Accept: application/json")
```

```
    @GET("api/carmodel/getCarmodel")
    Call<CarModel> getCarModel(@Query("id") Integer id);
    @Headers("Accept: application/json")
    @GET("api/carmodel/view")
    Call<ArrayList<Order>> getOrders(@Query("userid") Integer userid);
    @Headers("Accept: application/json")
    @GET("api/order/showone")
    Call<Order> getOrder(@Query("id") Integer id);
    @Headers("Accept: application/json")
    @GET("api/order/setstatus")
    Call<Order> setStatus(@Query("id") Integer id, @Query("status") Integer status);
    @Headers("Accept: application/json")
    @GET("api/order/create")
    Call<Order> createOrder(@Query("user_id") Integer user_id, @Query("car_id") Integer
car_id, @Query("date") CharSequence date,  @Query("term") Integer term);
    @Headers("Accept: application/json")
    @GET("api/category/categorylist")
    Call<ArrayList<Categry>> getCategories();
}
```

Є.4 Функціонал вікна авторазиції

```
protected void onCreate(Bundle savedInstanceState) {
    createaccount.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            savedata(context);
            Intent intent = new Intent(context, RegistrationActivity.class);
            startActivity(intent);
        }
    });
    Button signin = findViewById(R.id.login_signin);
    signin.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            if(checkFields()){
                final AlertDialog.Builder alert = new AlertDialog.Builder(context);
                alert.setPositiveButton(
                        R.string.ok,
                        new DialogInterface.OnClickListener() {
                            public void onClick(DialogInterface dialog, int id) {
                                dialog.cancel();
                            }
                        });
                String url = data.url;
                Retrofit retrofit = new Retrofit.Builder()
                        .baseUrl(url)
                        .addConverterFactory(GsonConverterFactory.create())
                        .build();
                Api api = retrofit.create(Api.class);
                Call<User> call = api.userLogin(login.getText(), password.getText());
                call.enqueue(new Callback<User>() {
```

```
                @Override
                public void onResponse(Call<User> call, Response<User> response) {
                    if (response.isSuccessful()) {
                        if(response.body()!=null){
                            data.user = response.body();
                            data.SaveUser(response.body(),context);
                            alert.setMessage(R.string.successfuly_login);
                            alert.show();
                            gotomain();
                        }
                        else{
                            alert.setMessage(R.string.incorrect);
                            alert.show();
                        }
                    }
                }
                @Override
                public void onFailure(Call<User> call, Throwable t) {
                    alert.setMessage(R.string.errorconnection+"   "+t.getMessage());
                    alert.show();
                }
            });
        }
    }
});
}
private void gotomain(){
    savedata(context);
    Intent intent = new Intent(this, MainActivity.class);
    startActivity(intent);
}
```

Є.5 Фрагмент функціоналу головного вікна

```
private void reloadOrders()
{
    LinearLayout orders = findViewById(R.id.orders);
    orders.removeAllViews();
    Call<ArrayList<Order>> call = api.getOrders(data.user.getId());
    call.enqueue(new Callback<ArrayList<Order>>() {
        @Override
        public void onResponse(Call<ArrayList<Order>> call, Response<ArrayList<Order>>
response) {
            if(!response.isSuccessful()){
                error("Code: "+response.code());
                return;
            }
            else
            {
                main.setVisibility(View.VISIBLE);
                Button button = findViewById(R.id.goorder);
                button.setOnClickListener(new View.OnClickListener() {
```

```
                @Override
                public void onClick(View v) {
                    BottomNavigationView bottomNavigationView;
                    bottomNavigationView = (BottomNavigationView)
findViewById(R.id.nav_view);
                    bottomNavigationView.setSelectedItemId(R.id.navigation_car_list);
                }
            });
        }
    }
    @Override
    public void onFailure(Call<ArrayList<Order>> call, Throwable t) {
        error(t.getMessage());
    }
  });
}
```

Є.6 Зміна паролю обілкового запису користувача

```
private void changepass(){
  String url = data.url;
    Call<Boolean> call = api.userChangepassword(data.user.getLogin(),
data.user.getPassword() , newpassowrdrepeat.getText().toString());
  call.enqueue(new Callback<Boolean>() {
    @Override
    public void onResponse(Call<Boolean> call, Response<Boolean> response) {
      if (response.isSuccessful()) {
        if(response.body()){
MainActivity.data.user.setPassword(newpassword.getText().toString());
data.user.setPassword(newpassowrdrepeat.getText().toString());
            savedata(context);
            alert.setCancelable(false);
            alert.setPositiveButton(
                R.string.ok,
                new DialogInterface.OnClickListener() {
                  public void onClick(DialogInterface dialog, int id) {
                    Intent intent = new Intent(context, MainActivity.class);
                    startActivity(intent);
                    dialog.cancel();
                  }
                }); }}
        @Override
    public void onFailure(Call<Boolean> call, Throwable t) {
      alert.setCancelable(false);
      alert.setPositiveButton(
            R.string.ok,
            new DialogInterface.OnClickListener() {
              public void onClick(DialogInterface dialog, int id){
                Intent intent = new Intent(context, MainActivity.class);
                startActivity(intent);
                dialog.cancel();
              }
```

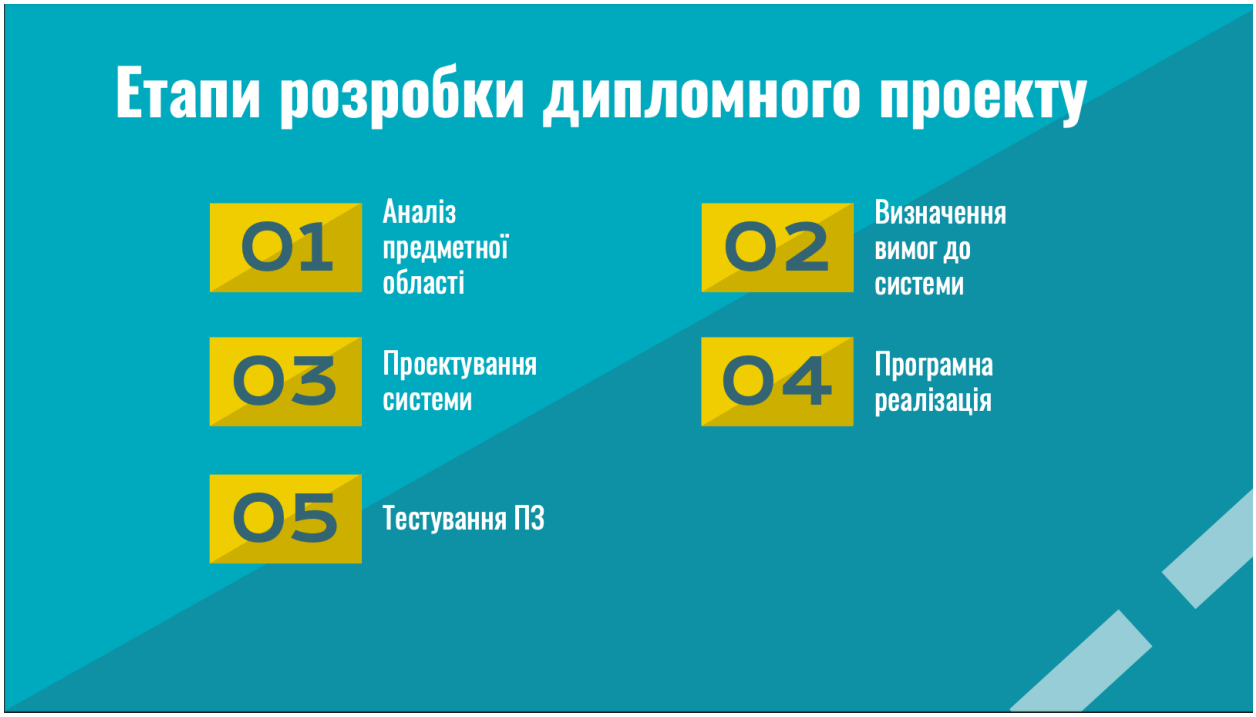
```
                });
            alert.setMessage(R.string.errorrequest);
            alert.show();
        }
    });
}
```

ДОДАТОК Ж

(обов’язковий)

ПРЕЗЕНТАЦІЙНІ МАТЕРІАЛИ


Хмельницький національний університет
Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення
Дипломний проект на тему:
«Веб-орієнтована система для забезпечення роботи сервісу прокату автомобілів»
Виконав:
студент групи ІПЗс-19-1 Карпович В.В.
Керівник:
канд. пед. наук, доцент Онишко О. Г.
Хмельницький 2022


Етапи розробки дипломного проекту
01
Аналіз предметної області
02
Визначення вимог до системи
03
Проектування системи
04
Програмна реалізація
05
Тестування ПЗ

Актуальність теми

Прокат автомобілів це відомий та доволі розповсюджений вид бізнесу в світі. В той же час, розвиток нових технологій, розповсюдження гаджетів, перехід до цифрової економіки відкривають нові можливості як для власників такого бізнесу, так і для клієнтів.

Український ринок послуг з прокату автомобілів до війни розвивався дуже динамічно. На ринку були представлені послуги як для клієнтів, що хочуть орендувати автомобіль, так і для тих, хто хоче віддати свій автомобіль в оренду і отримувати за це гроші.

За даними Асоціації Прокатних компаній в Україні налічувалося більше 200 прокатних компаній. А представити на сьогодні компанію, що працює з клієнтами без використання інформаційних технологій неможливо. Тому тема бакалаврської роботи є актуальною.

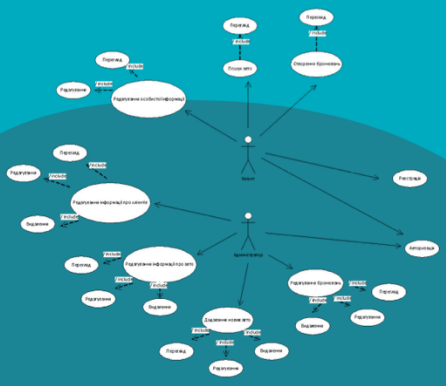
Мета роботи

Розробити програмне забезпечення у вигляді веб-орієнтованої системи для забезпечення роботи сервісу прокату автомобілів, яка буде складатися з веб-сайту та мобільного додатку, що буде сприяти підвищенню конкурентоспроможності сервісу на ринку та збільшенню привабливості для клієнтів

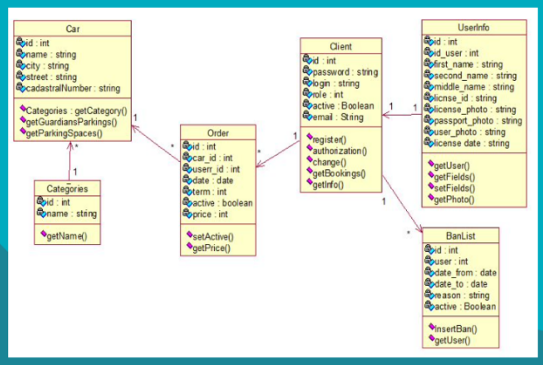
Завдання роботи

- здійснити розробку структури веб-орієнтованої системи;
- розробити базу даних для зберігання інформації про прокатні автомобілі та клієнтів;
- спроектувати серверну та клієнтську частини веб-додатку;
- створити веб-сайт для клієнтської та адміністративної частин системи;
- створити мобільний додаток для клієнтів системи;
- провести тестування розробленої веб-орієнтованої системи забезпечення роботи сервісу з прокату автомобілів.

Формалізація задачі забезпечення роботи сервісу з прокату автомобілів



Діаграма використання



Діаграма класів

Проектування архітектури веб-орієнтованої системи

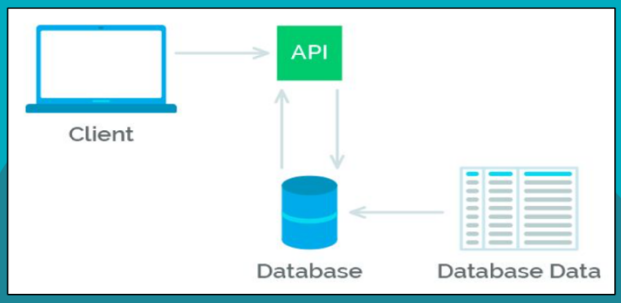
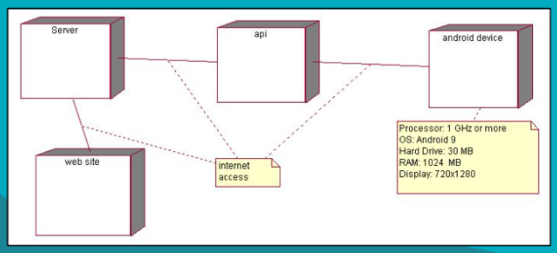
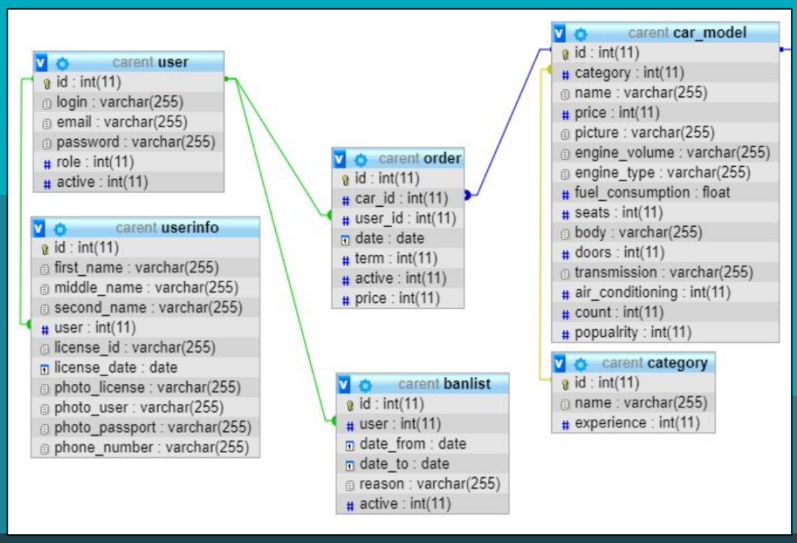


Схема взаємодії клієнта та серверу у архітектурі REST



Діаграма розгортання

База даних веб-орієнтованої системи



Вибір технологій розробки

Веб сайт та серверна частина

Технології:

- PHP
- Yii2
- mySQL
- docker

Середовища розробки:

- phpStorm
- dataGrip

yiiframework

php

phpMyAdmin

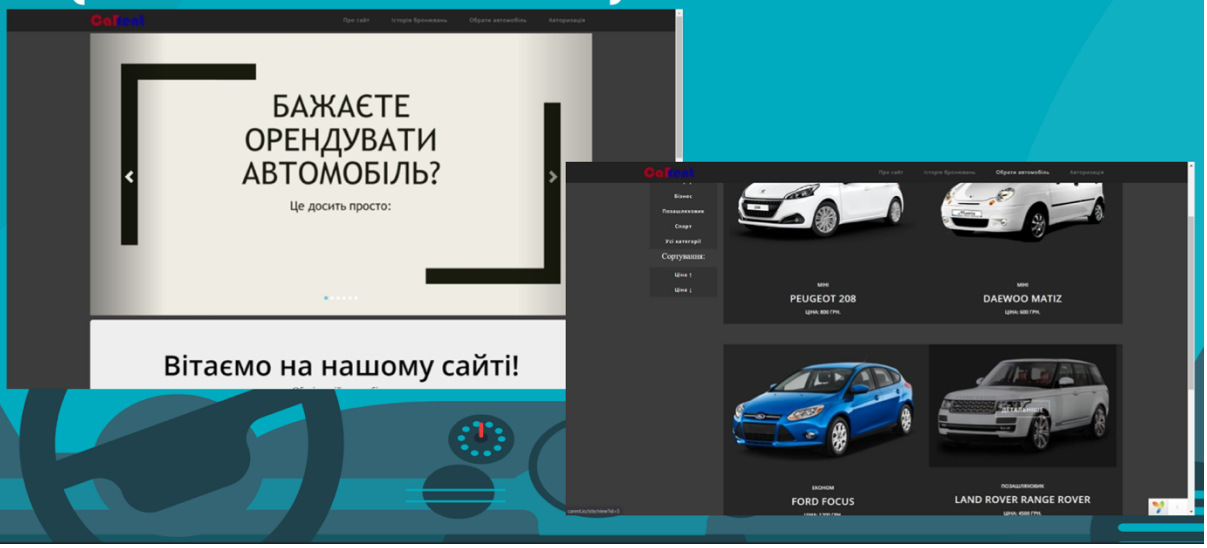
MySQL

Мобільний додаток

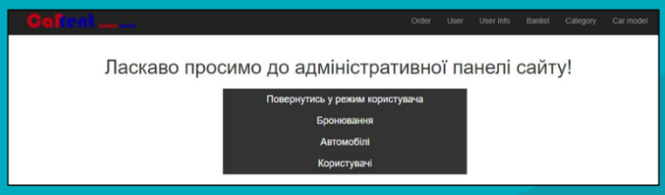
Технології:
- Java Android
- Retrofit

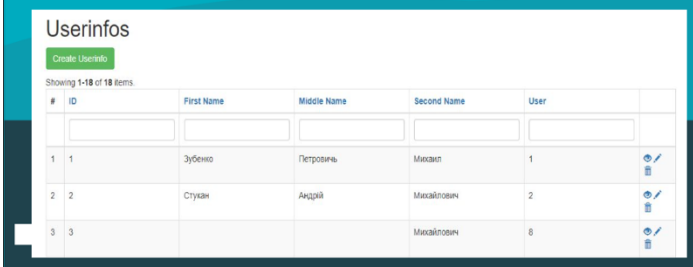
Середовища розробки:
- Android Studio

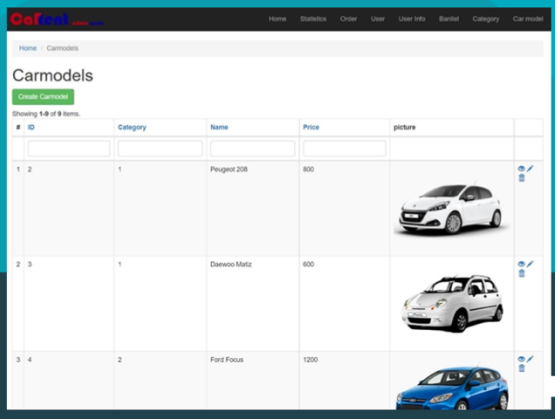
Програмна реалізація (клієнтська частина)



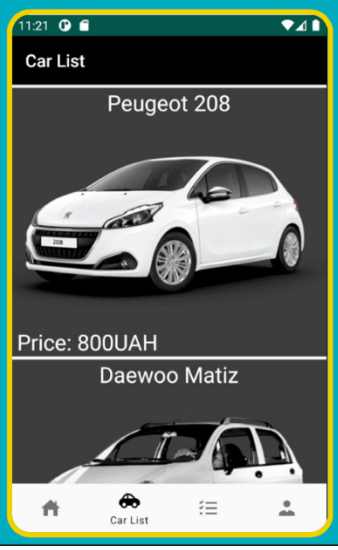
Програмна реалізація (адміністративна частина)

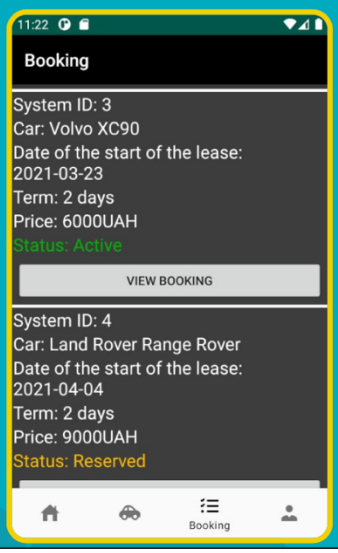




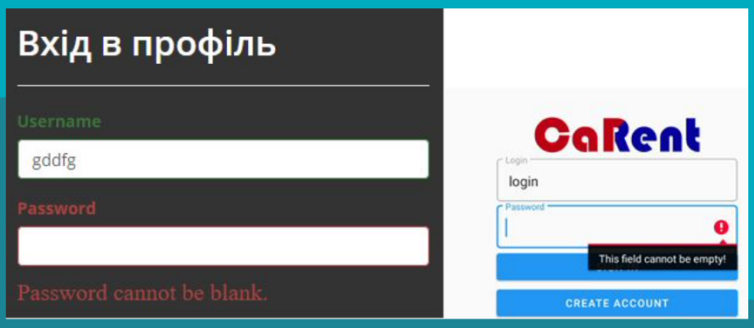


Програмна реалізація (мобільний додаток)





Тестування системи



Висновки

1. Проаналізовано проблему та поставлено мету задачі
2. Проаналізовано існуючі рішення
3. Зпроектовано серверну та клієнтську частини ПЗ
4. Визначено технології розробки
5. Розроблено адміністративну та клієнтські частини веб-сайту
6. Розроблено мобільний додаток
7. Проведено тестування системи

Завідувачу кафедри
інженерії програмного забезпечення
проф. Бедратюку Л. П.
студента групи ІПЗс-19-1
Карповича В.В.
Прізвище, ініціали

ЗАЯВА

Прошу закріпити за мною тему дипломного проекту освітнього ступеня «бакалавр» за спеціальністю 121 «Інженерія програмного забезпечення»:
Веб-орієнтована система для забезпечення роботи сервісу прокату автомобілів

(керівник дипломного проекту – Онишко Оксана Григорівна)
Прізвище, ім'я, по батькові

02.02.2022
Дата

Підпис студента

Завідувачу кафедри інженерії програмного забезпечення проф. Бедратюку Л. П.

здобувача вищої освіти
Карповича В. В.
Прізвище, ініціали

факультет ІТ, 3 курс, група ІПЗс-19-1

ЗАЯВА

З правилами чинного Положення «Про дотримання академічної доброчесності в Хмельницькому національному університеті» від 26.09.2020 (зі змінами від 26.11.2020), згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений. Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщений та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

02.02.2022
дата

підпис

Fri Jun 03 12:21:21 EEST 2022, Хіврич Володимир Русланович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 5.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 9%**

	Документ		Сумарний збіг по Базі Даних	
ID: 104413 Назва: Веб-орієнтована система для забезпечення роботи сервісу прокату автомобілів Додано в БД: 2022-06-03 Автора: В.В. Карпович Керівники: О.Г. Онишко Консультанти: Опоненти:	Символи	Лексеми	Символи	Лексеми
	62227	973	5455 (9%)	98 (10%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

UNICHECK

Ім'я користувача:
Кафедра ІПЗ

ID перевірки:
1011450695

Дата перевірки:
03.06.2022 13:16:23 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
03.06.2022 13:16:56 EEST

ID користувача:
100005589

Назва документа: записка_титулка_Карпович_ВВ

Кількість сторінок: 72 Кількість слів: 11163 Кількість символів: 88825 Розмір файлу: 6.26 MB ID файлу: 1011329691

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

5.29%
Схожість

Найбільша схожість: 1.35% з джерелом з Бібліотеки (ID файлу: 1011305742)

3.69% Джерела з Інтернету 134 ..... Сторінка 74

2.62% Джерела з Бібліотеки 67 ..... Сторінка 75

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Підозріле форматування 18 сторінок

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА ДИПЛОМНИЙ ПРОЕКТ
освітнього ступеня «Бакалавр»

Дипломник Карпович Вячеслав Васильович

Тема Веб-орієнтована система для забезпечення роботи сервісу прокату автомобілів

Спеціальність 121 – Інженерія програмного забезпечення

Обсяг дипломного проекту:

Кількість листів креслень _____; кількість сторінок записки 107

1.Короткий зміст пояснювальної записки та прийнятих рішень У дипломному проекті було досліджено і проаналізовано предметну область, усі функціональні та нефункціональні вимоги. Був проведений аналіз існуючих рішень на ринку, розглянуто їх переваги і недоліки. На основі отриманих даних були сформовані функціональні вимоги до дипломного проекту та доведено актуальність розробки нового програмного забезпечення. Було розглянуто інструменти для реалізації дипломного проекту, в результаті чого створено програмне забезпечення. Було проведено тестування програми, за результатами якого доведено, що розроблене програмне забезпечення працює коректно та готове до експуатації.

2. Висновок про відповідність проекту поставленому завданню Дипломний проект виконаний відповідно до поставленого завдання та з дотриманням всіх вимог.

3. Характеристика виконання кожного розділу проекту, ступінь використання останніх досягнень науки і техніки та передових методів роботи У вступі доведено актуальність теми, визначено мету та завдання дипломноого проектування. У першому розділі проведено аналіз предметної області, розглянуто існуючі рішення та визначені функціональні і нефункціональні вимоги до розроблюваного програмного забезпечення. У другому розділі проведено аналіз сучасних архітектур, розглянуто їх переваги і недоліки та визначено, що система буде відповідати монолітній архітектурі та моделі клієнт-сервер. У третьому розділі підготовлено всі залежності для написання коду та виконано практичну розробку програмних модулів і описано їх особливості, в результаті чого створено програмний продукт. В четвертому розділі було виконано модульне тестування системи та проведено його у відповідності до функціональних вимог, в результаті було підтверджено коректну роботу програми.

4. Позитивні сторони проекту Тематика дипломного проекту є актуальною, оскільки на сьогодні в Україні сервіси даної тематики є особливо актуальними і популярними серед молоді. Також було застосовано новітні технології для побудови програмного продукту та актуальні архітектурні рішення. Програмне забезпечення має сучасний дизайн, задовільняє потреби кінцевого користувача

5. Негативні сторони проекту У проекті відсутній функціонал відслідковування автомобілів в онлайн режимі за допомогою карт. Також до недоліків можна віднести наявність додатку тільки для платформи Android, для готового продукту наявність ще iOS додатку необхідна в умовах сучасного ринку.

6. Оцінка графічного оформлення та пояснювальної записки проекту Графічне оформлення виконано відповідно до теми дипломного проекту та подано у вигляді діаграм і рисунків. Пояснювальна записка оформлена згідно вимог чинних стандартів.

7. Відгук про дипломний проект в цілому Дипломний проект заслуговує позитивної оцінки. Матеріал пояснювальної записки структурований, послідовний, чіткий та простий, що дозволяє чітко зрозуміти викладений матеріал у рамках тематики дипломного проекту. Графічний матеріал дає можливість наочно побачити деталі проектування системи.

8. Інші зауваження

9. Оцінка дипломного проекту Дипломний проект виконаний у повному обсязі, відповідає поставленій задачі та заслуговує на оцінку «добре».

РЕЦЕНЗЕНТ доктор технічних наук, професор Говорущенко Т.О., завідувач кафедри комп'ютерної інженерії та інформаційних систем

"27" травня 2022 р. _____
(підпис)

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ
КАФЕДРИ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: «Веб-орієнтована система для забезпечення роботи сервісу прокату автомобілів»

Автор: Карпович Вячеслав Васильович
Спеціальність: 121 – Інженерія програмного забезпечення
Освітня програма: Освітньо-професійна програма «Інженерія програмного забезпечення»
Науковий керівник: Онишко Оксана Григорівна, кандидат педагогічних наук, доцент
Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	**відповідає**
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
5	Інше:	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) у тексті дипломної роботи системами перевірки на плагіат виявлено схожість з деякими документами в частині загальновживаних обов'язкових словосполучень у стандартних бланках (титулка, бланк завдання), у структурі змісту, назвах розділів/підрозділів тощо та в назвах публікацій у переліку джерел посилання;

2) в якості запозичень системою було зафіксовано деякі послідовності вихідного коду і посилання на бібліотеки, які є стандартними мовними конструкціями програмування та не можуть розглядатися як об'єкт авторських прав і, відповідно, їх порушення;

3) усі запозичення є фрагментарними або мають належним чином оформленні посилання;

4) виявлені модифікації тексту не впливають на відсоток схожості.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів ідентичності/ схожості, складає 5,29% і адресується до 201 джерел, що, з урахуванням наведених обґрунтувань, відповідає характеру теми і свідчить на користь дипломної роботи.

Керівник _____ О. Г. Онишко

Гарант ОП _____ Л. П. Бедратюк

Завідувач кафедри _____ Л. П. Бедратюк