

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавра

(ступінь вищої освіти)

на тему ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ПРОКАТУ

(назва теми)

ТУРИСТИЧНОГО СПОРЯДЖЕННЯ

SOFTWARE FOR OUTDOOR EQUIPMENT RENTAL

Виконав(ла): студент(ка) 4 курсу, групи КНТ-210

Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Комп'ютерні науки

ЩЕРБАНЬ В.Д.

(ПРИЗВИЩЕ та ініціали)

Керівник СУББОТІН С.О.

(ПРИЗВИЩЕ та ініціали)

Рецензент ГОЛУБ Т.В.

(ПРИЗВИЩЕ та ініціали)

2024__

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет _____ КНТ _____
Кафедра _____ програмних засобів _____
Ступінь вищої освіти _____ бакалавр _____
Спеціальність _____ 122 Комп'ютерні науки _____
(код і найменування)
Освітня програма (спеціалізація) _____ Комп'ютерні науки _____
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
« _____ » _____ 2024 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ЩЕРБАНЯ Володимира Дмитровича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення прокату туристичного спорядження. Software for Outdoor Equipment Rental

керівник проєкту (роботи) д.т.н., професор СУББОТІН Сергій Олександрович,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від « 24 » квітня 2024 року № 168

2. Строк подання студентом проєкту (роботи) 10 червня 2024 року

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

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Проєктування програми. 3. Розробка програми. 4. Експлуатація та тестування програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	СУББОТІН С.О., зав. кафедри		
Нормоконтроль	ДЕЙНЕГА Л.Ю., ст. викладач		

7. Дата видачі завдання « 15 » квітня 2024 року.

КАЛЕНДАРНИЙ ПЛАН

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

Студент(ка)

_____ Володимир ЩЕРБАНЬ
(підпис) (Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

_____ Сергій СУББОТІН
(підпис) (Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 87 с., 27 рис., 2 табл., 3 дод., 10 джерел.

ТУРИСТИЧНЕ СПОРЯДЖЕННЯ, ПРОКАТ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, БАЗА ДАНИХ, ОБ'ЄКТНО-РЕЛЯЦІЙНЕ ВІДОБРАЖЕННЯ.

Об'єкт роботи – процес прокату туристичного спорядження. Предмет роботи – програмне забезпечення прокату туристичного спорядження. Мета роботи – розробити програмне забезпечення прокату туристичного спорядження для підтримки індивідуальних пунктів прокатів та індивідуальних власників спорядження.

Матеріали, методи та технічні засоби: мова Python, мультипарадигмальне програмування, фреймворк Django, система PostgreSQL.

Результати. Програма розроблена як результат прийняття рішень стосовно визначення основних функцій аналогів, що забезпечують прокат туристичного спорядження, формування відмінностей розробки, вибору засобів, виокремлення сценаріїв використання власниками прокатів, менеджерами системи прокату, винаймачами туристичного спорядження, їхньої програмної реалізації.

Висновки. Розроблено вебдодаток прокату туристичного спорядження, який порівняно з аналогами має особливості, що включають об'єднання можливостей прокату від конкретних пунктів і від власників туристичного спорядження та можливість створення типового замовлення прокату туристичного спорядження на основі визначено шаблону.

Галузь використання. Прокат туристичного спорядження на основі пропозицій індивідуальних пунктів прокату та власників.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 87 pages, 27 figures, 2 tables, 3 appendixes, 10 sources.

OUTDOOR EQUIPMENT, RENTAL, SOFTWARE, DATABASE, OBJECT-RELATIONAL MAPPING.

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

Materials, methods and technical means: Python language, multiparadigm programming, Django framework, PostgreSQL system.

The results. The program was developed as a result of decision-making regarding the definition of the main functions of analogs which provide rental of outdoor equipment, the formation of design differences, the selection of tools, the identification of scenarios of use by rental owners, managers of the rental system, renters of outdoor equipment, and their software implementation.

Conclusions. A web application for the rental of tourist equipment has been developed, which has features that include combining rental options from specific rental points and from owners of outdoor equipment and the possibility of creating a standard order for the rental of outdoor equipment based on a defined template. These functions are its specific features compared to analogues.

Field of use. Rental of tourist equipment proposed by individual rental points and owners.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	7
Вступ.....	8
1 Аналіз предметної області.....	9
1.1 Огляд програмних засобів прокату туристичного спорядження	9
1.2 Особливості розробки.....	13
1.3 Висновки за розділом 1	14
2 Проєктування програми.....	16
2.1 Вибір засобів проєктування і розробки	16
2.2 Визначення функціональних вимог до програмного забезпечення	17
2.3 Проєктування структури бази даних.....	21
2.4 Висновки за розділом 2	31
3 Розробка програми	32
3.1 Порядок реалізації функцій програми	32
3.2 Реалізація функцій програми	34
3.3 Висновки за розділом 3	39
4 Експлуатація та тестування програми	40
4.1 Призначення програми	40
4.2 Умови виконання програми	40
4.3 Виконання і тестування програми.....	40
4.4 Висновки за розділом 4	46
Висновки	47
Перелік джерел посилання	49
Додаток А Технічне завдання	50
Додаток Б Текст програми	55
Додаток В Слайди презентації.....	81

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

ОРВ – об’єктно-реляційне відображення;

ПЗ – програмне забезпечення.

ВСТУП

Туристичне спорядження є достатньо вартісним, а використовується достатньо рідко, що знижує ефективність його придбання для тільки власного використання. У підсумку більшу частину часу туристичне спорядження не використовується.

Прокат туристичного спорядження характеризується збільшенням необхідності в прокаті від винаймачів, але і з іншого боку – потребою здати в прокат власне туристичне спорядження, щоб підвищити ефективність від його придбання. Об'єднання цих двох проблем вимагає створення програмного забезпечення (ПЗ), здатного забезпечувати потрібні для цього функції. Користувачами ж такої програми мають бути винаймачі туристичного спорядження та власники туристичного спорядження.

Винаймачі туристичного спорядження можуть отримати зиск від програми в тому, що вона дозволить їм звертатися і до окремих пунктів прокату, і до індивідуальних власників туристичного спорядження, що має збалансувати ціни, розширити кількість варіантів доступного для прокату туристичного спорядження, знайти більш унікальні варіанти.

Власники туристичного спорядження можуть через ПЗ визначати наявне у них туристичне спорядження для прокату, знижувати власні витрати на його володіння за рахунок додаткових доходів.

Очевидною є потреба в даній сфері стосовно об'єднання інтересів власників туристичного спорядження та винаймачів. Для цього необхідно розробити відповідне ПЗ, що підкреслює актуальність проблеми, яка розглядається.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд програмних засобів прокату туристичного спорядження

Звичайно, що прокат туристичного спорядження вже є на даний момент популярною сферою, а тому існують різні програмні засоби, які підтримують розв'язання цієї проблеми. Проте важливо визначити, на якому рівні забезпечена ця підтримка. До результатів огляду, представленого в розділі 1, подано декілька програмних засобів, що було обрано серед всіх розглянутих.

RentOutdoorGear [1] (рис. 1.1) є американським сервісом прокату туристичного спорядження, який надає послуги в штаті Айдаго, окрім того надаючи можливість придбати нове або використовуване туристичне спорядження. Для кожного туристичного спорядження потрібно задати дату отримання та повернення. Прокат відбувається подовово. У підсумку формується кошик, який в кінці визначає замовлення. Зі збільшенням тривалості прокату середня вартість прокату за день зменшується.

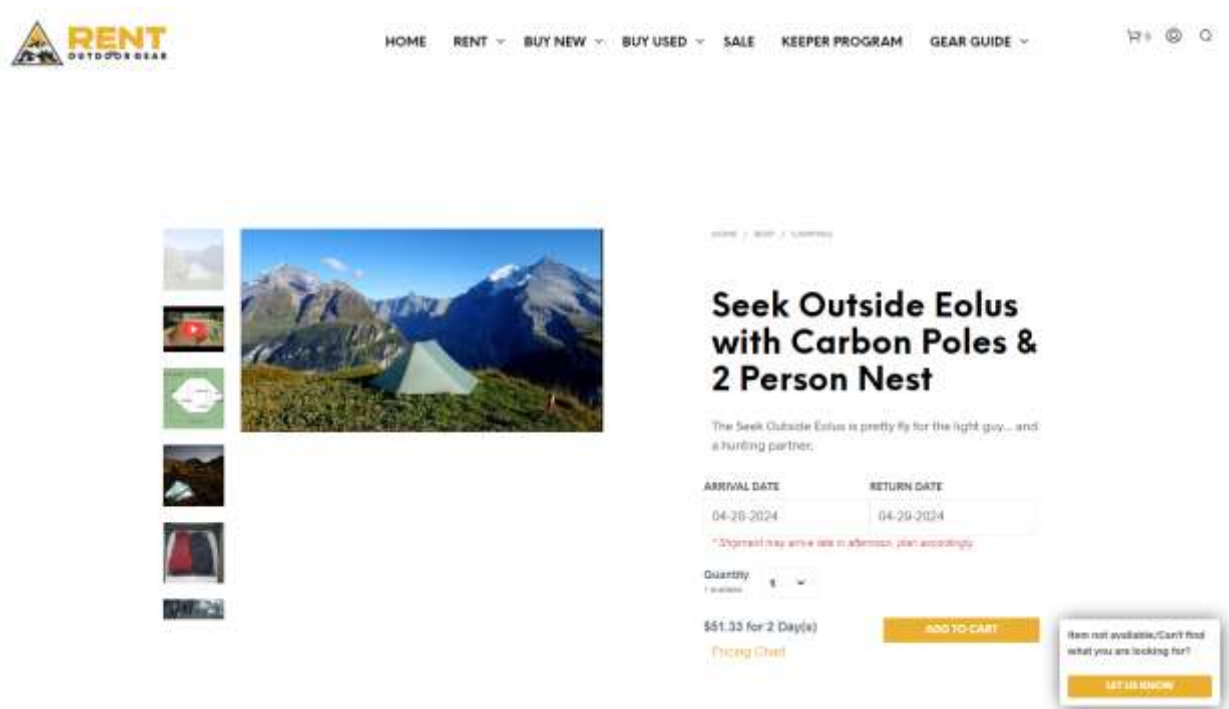
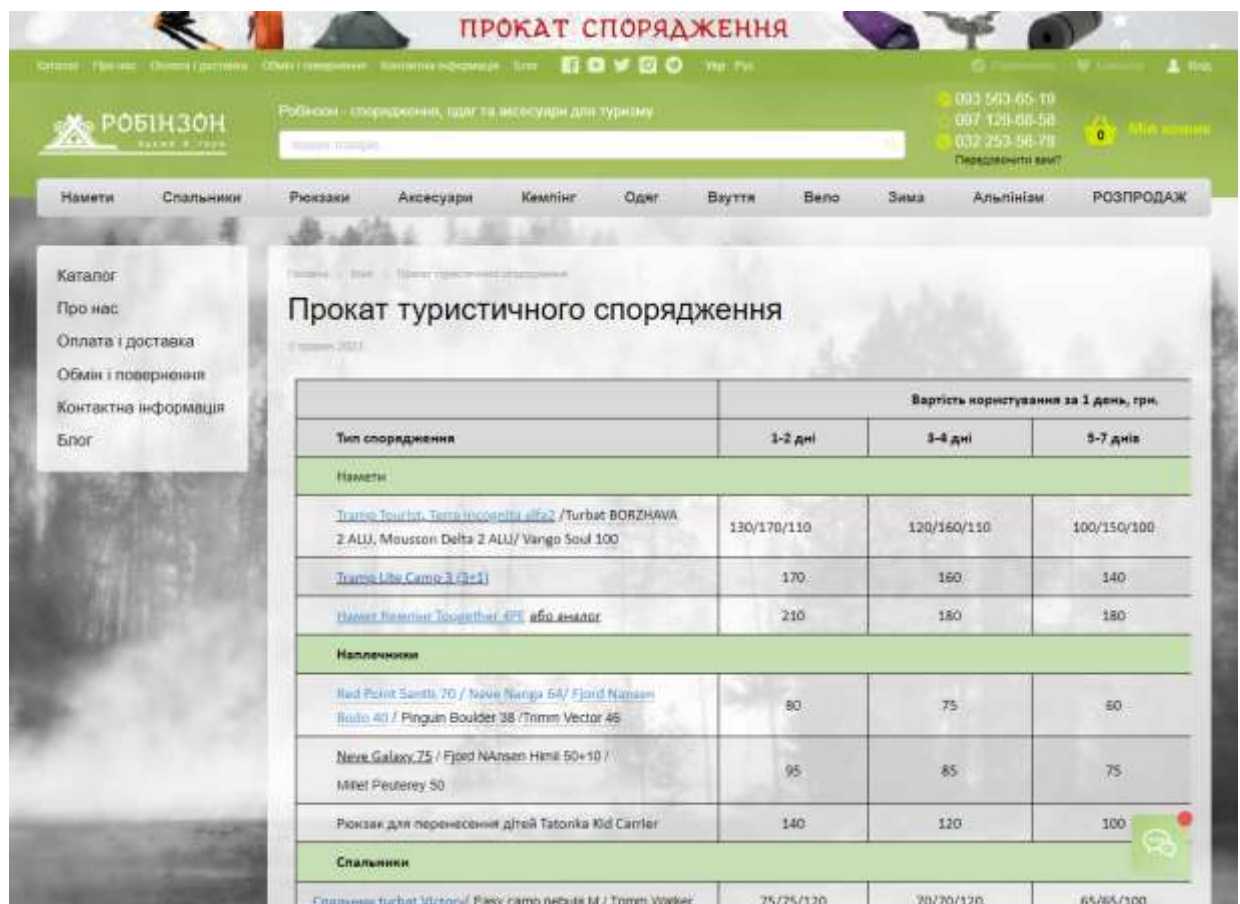


Рисунок 1.1 – Вебсайт RentOutdoorGear

Вебсайт магазину Робінзон [2] (рис. 1.2) є українським прокатом туристичного спорядження, а тому важливий у даному огляді стосовно того, як на рівні ПЗ такі послуги забезпечуються в Україні. Аналізуючи можливості даного вебсайту, потрібно зазначити, що в першу чергу він діє як магазин, тобто забезпечує можливості придбання туристичного спорядження. Прокат же фактично представлений таблицею з позиціями конкретного туристичного спорядження та вартістю такого прокату в залежності від тривалості. Оформити замовлення можливо на придбання туристичного спорядження, прокат же реалізується не через вебсайт. Тобто фактично можна отримати довідку про вартість послуг. Перевірити доступність спорядження на потрібну дату неможливою Замовлення в користувацькому кабінеті стосуються тільки придбання. У даного програмного засобу, як і у попереднього, є можливість авторизації, реєстрації.



Тип спорядження	Вартість користування за 1 день, грн.		
	1-2 дні	3-4 дні	5-7 днів
Намети			
Turbat Tourist, Tatra Inceptus 4F2 / Turbat BORZHAVA 2 ALU, Mousson Delta 2 ALU / Vango Soul 100	130/170/110	120/160/110	100/150/100
Turbat Lite Camp 3 (3+1)	170	160	140
Turbat Premium Together 4F2 або аналог	210	180	180
Наплечники			
Red Point Santa 70 / Neve Nanga 6A / Fjord Nansen Budo 40 / Penguin Boulder 38 / Trimm Vector 46	80	75	60
Neve Galaxy 75 / Fjord Nansen Hime 50+10 / Milet Peulerey 50	95	85	75
Рюкзаки для перенесення дітей Tatonka Kid Carrier	140	120	100
Спальники			
Спальник turbat Victory / Easy camp nebula M / Trimm Walker	75/75/120	70/70/120	65/65/100

Рисунок 1.2 – Вебсайт магазину Робінзон

Вебсайт Outdoor Equipment Rentals [3] (рис. 1.3) спеціалізується безпосередньо на прокаті туристичного спорядження. Доступні різні типи туристичного спорядження, хоча і невелика кількість конкретного спорядження. Надається можливість забронювати потрібне туристичне спорядження на необхідні дати за подовим принципом (рис. 1.4). Вартість розраховується за відповідною ставкою та заданою кількістю. Можна зберегти до переліку туристичне спорядження. Є американським сервісом, передбачається доставка.

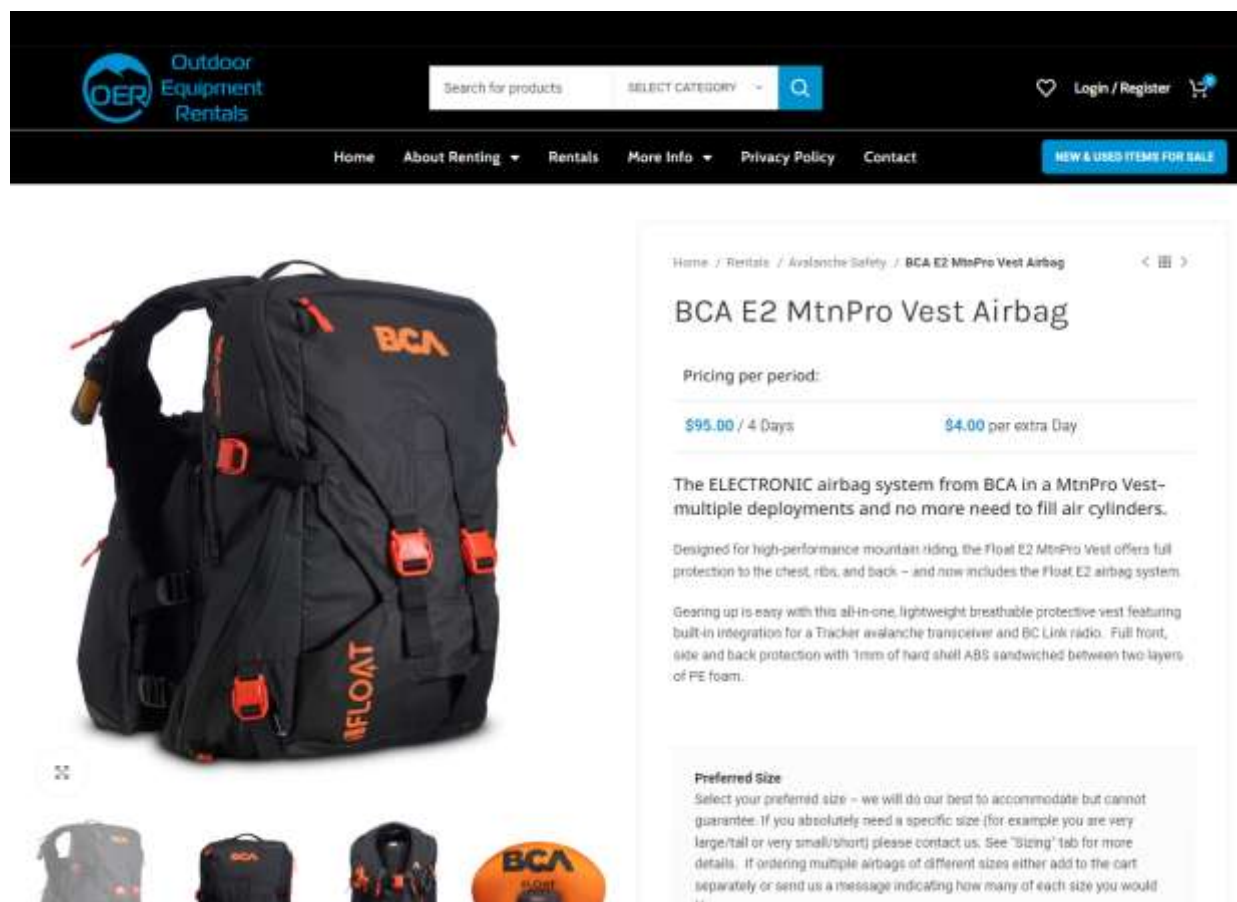


Рисунок 1.3 – Вебсайт Outdoor Equipment Rentals (інформація по спортивне спорядження)

OutdoorHire [4] (рис. 1.5) є сервісом прокату туристичного спорядження, який діє у Великобританії. Авторизація користувачів не передбачається.

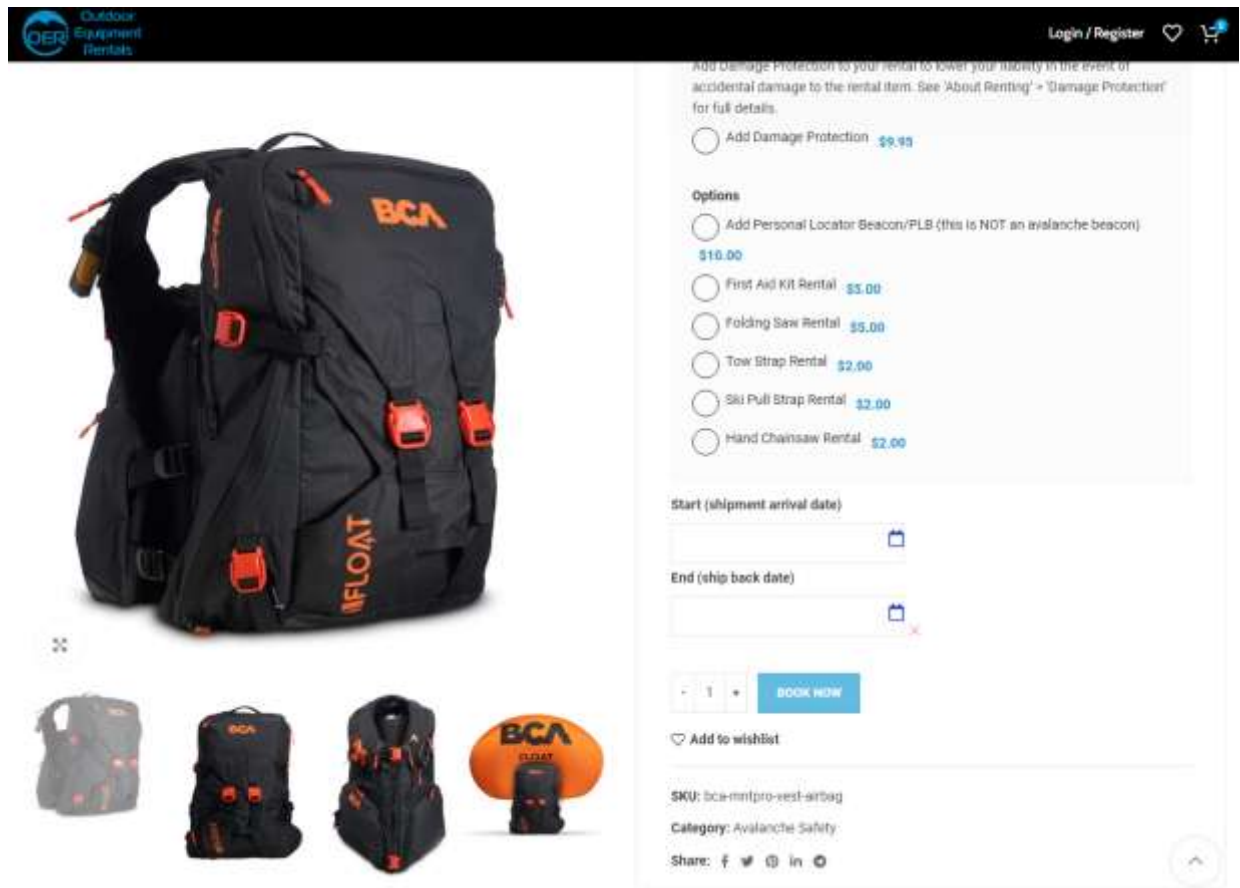


Рисунок 1.4 – Вебсайт Outdoor Equipment Rentals (параметры прокату)

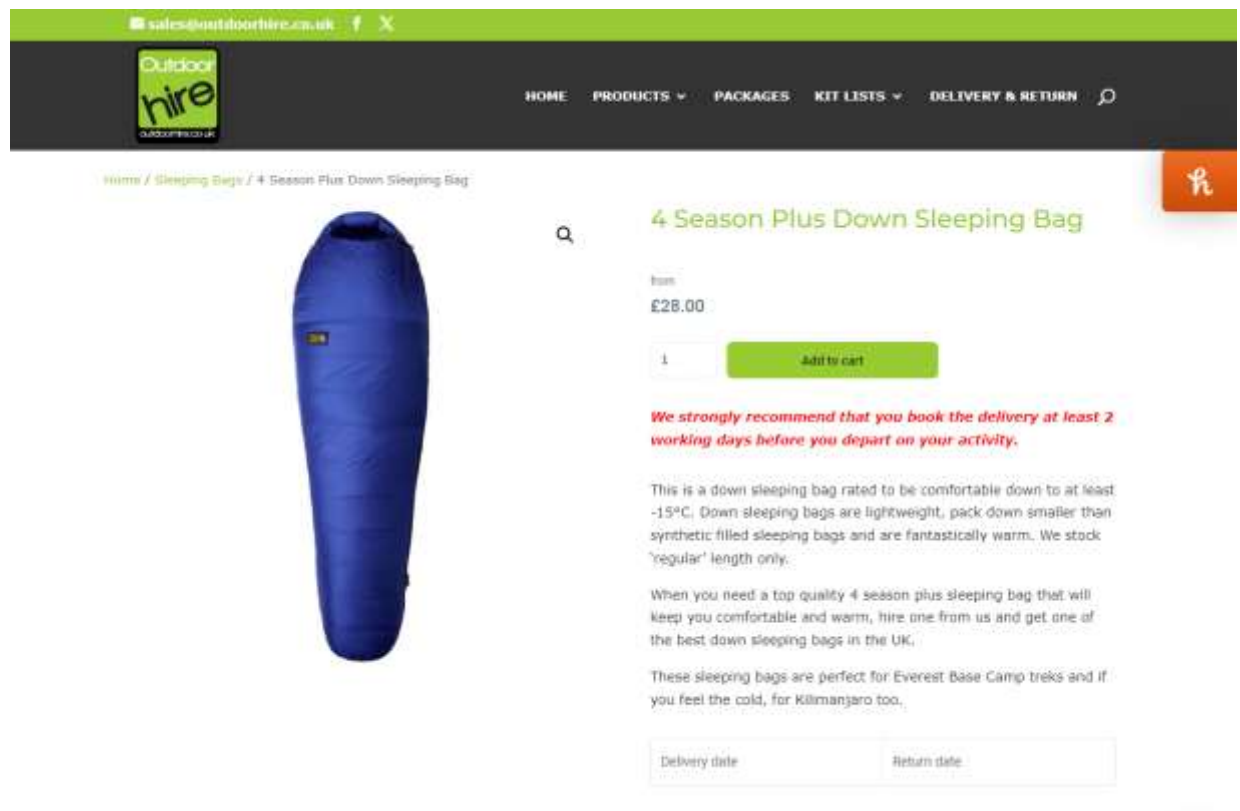
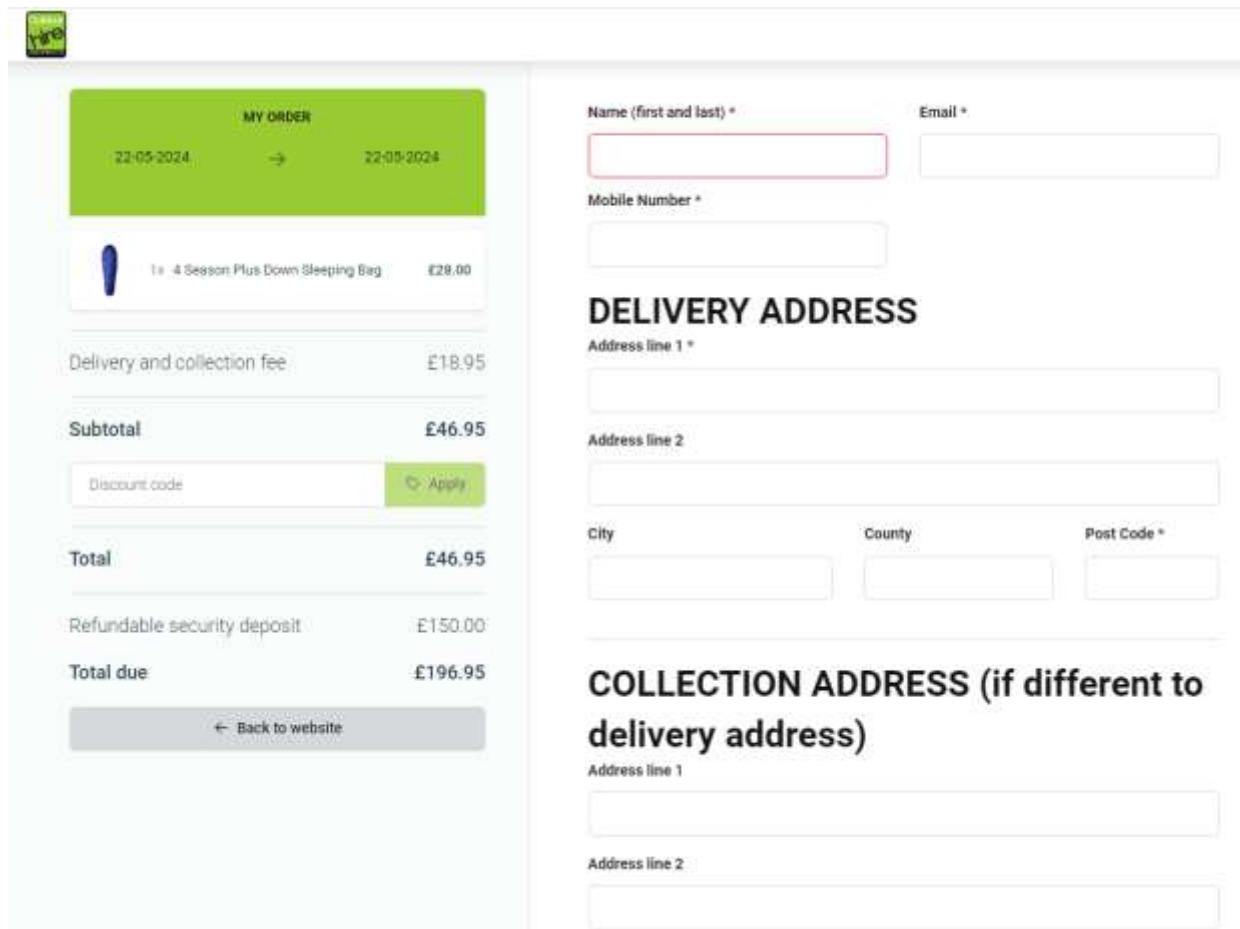


Рисунок 1.5 – Вебсайт OutdoorHire

OutdoorHire передбачає формування кошику, після чого в кінці оформлюється бронювання з визначенням імені, електронної поштової адреси, номеру телефону, адреси доставки (рис. 1.6). Після цього має бути забезпечено сплату.



MY ORDER

22-05-2024 → 22-05-2024

1 x 4 Season Plus Down Sleeping Bag £28.00

Delivery and collection fee £18.95

Subtotal £46.95

Discount code

Total £46.95

Refundable security deposit £150.00

Total due £196.95

[← Back to website](#)

DELIVERY ADDRESS

Name (first and last) * Email *

Mobile Number *

Address line 1 *

Address line 2

City County Post Code *

COLLECTION ADDRESS (if different to delivery address)

Address line 1

Address line 2

Рисунок 1.6 – Оформлення замовлення прокату туристичного спорядження на вебсайті OutdoorHire

1.2 Особливості розробки

Зважаючи на встановлені характеристики аналогів, розробка повинна враховувати цей досвід, але і мати свої особливості (табл. 1.1). Особливостями є об'єднання можливостей прокату від конкретних пунктів і від власників туристичного спорядження та можливість створення типового замовлення прокату туристичного спорядження на основі визначено

шаблону, куди входить збережене туристичне спорядження від власників або пунктів прокату в заданій кількості. Аналоги направлені на підтримку роботи тільки одного конкретного пункту або фактично магазину продажу туристичного спорядження, який додатково надає послуги прокату.

Таблиця 1.1 – Порівняння розробки з проаналізованими аналогами

Характеристика	RentOutdoor-Gear	Робінзон	Outdoor Equipment Rentals	Розробка
Реєстрація	+	+	+	+
Замовлення (бронювання) прокату туристичного спорядження	+	–	+	+
Джерело послуг	Пункт прокату	Магазин	Пункт прокату	Пункти прокату, власники
Створення типового замовлення	–	–	–	+

1.3 Висновки за розділом 1

Завдання, які необхідно виконати у дипломній кваліфікаційній роботі бакалавра:

- аналіз проблеми розробки ПЗ прокату туристичного спорядження;
- визначення вимог до ПЗ прокату туристичного спорядження;
- проєктування ПЗ прокату туристичного спорядження;
- реалізація ПЗ прокату туристичного спорядження.

Перший розділ реалізує виконання завдання аналізу проблеми розробки ПЗ прокату туристичного спорядження, під час якого проаналізовано зокрема можливості вебсайтів RentOutdoorGear, Outdoor Equipment Rentals, OutdoorHire та українського вебсайту Робінзон, визначено особливості розробки порівняно з цими аналогами, що включають об'єднання можливостей прокату від конкретних пунктів і від власників туристичного спорядження та можливість створення типового замовлення прокату туристичного спорядження на основі визначено шаблону.

2 ПРОЄКТУВАННЯ ПРОГРАМИ

2.1 Вибір засобів проєктування і розробки

ПЗ прокату туристичного спорядження доцільно створювати як вебдодаток, таким чином реалізуючи можливість доступу користувачів, які головне повинні мати браузер на власному пристрої. Оскільки мова Python [5] сприяє швидшій розробці, то на її основі вирішено розробляти вебдодаток, проте сама по собі мова Python не забезпечує реалізацію сучасних вебдодатків, а для цього потрібний фреймворк, серед яких мова, наприклад, надає Django [6] і Flask [7]-[8] (табл. 2.1). Порівняння проводилось на основі показників роботи з базою даних, зокрема доступності об'єктно-реляційного відображення (ОРВ), вбудованих засобів керування через спеціальний інтерфейс, наявності вбудованих засобів підтримки захищеності ПЗ, сприяння швидкості розробки.

Таблиця 2.1 – Порівняння фреймворків для створення ПЗ

Показник	Django	Flask
Вбудовані засоби керування даними бази даних через спеціальний інтерфейс	Адміністративна панель	Відсутні (необхідно окремо встановлювати)
Вбудовані засоби підтримки захищеності ПЗ	Широке коло	Відсутні (необхідно окремо встановлювати)
Сприяння швидкості розробки	Більше	Менше
Робота з базами даних	Вбудоване ОРВ підтримує системи SQLite, PostgreSQL, MariaDB, Oracle	Не обмежена, потребує встановлення бібліотеки для роботи з базою даних та SQLAlchemy для ОРВ

Для розробки ПЗ прокату туристичного спорядження обрано мову Python та фреймворк Django.

2.2 Визначення функціональних вимог до програмного забезпечення

Винаймач туристичного спорядження повинен мати доступними сценарії:

- авторизація винаймача, що за передумов відсутності облікового запису винаймача представляється реєстрацією винаймача;
- відображення діючих пунктів прокату туристичного спорядження за містом;
- пошук туристичного спорядження, необхідного для прокату;
- замовлення прокату туристичного спорядження, що за передумови наявності замовлення прокату туристичного спорядження, але відсутності потреби представляє скасування власного замовлення прокату туристичного спорядження;
- замовлення прокату туристичного спорядження на основі власного типового переліку замовлення;
- створення власного типового переліку замовлення туристичного спорядження, що за передумов наявності такого переліку представляє вилучення власного типового переліку замовлення туристичного спорядження;
- створення власного типового переліку замовлення туристичного спорядження на основі створеного раніше власного замовлення прокату;
- перегляд всіх власних типових переліків замовлення туристичного спорядження;
- перегляд складу типового переліку замовлення туристичного спорядження (рис. 2.1).



Рисунок 2.1 – Діаграма прецедентів винаймача туристичного спорядження

Також до сценаріїв винаймача туристичного спорядження належать:

- додавання туристичного спорядження до типового переліку замовлення туристичного спорядження, що за передумов необхідності уточнення вже наявного складу туристичного спорядження може представляти сценарії вилучення туристичного спорядження з типового переліку замовлення туристичного спорядження, зміни кількості туристичного спорядження в типовому переліку замовлення туристичного спорядження;

- перегляд власних замовлень прокатів туристичного спорядження;

- перегляд туристичного спорядження;
- перегляд складу власного замовлення прокату туристичного спорядження;
- відображення туристичного спорядження, доступного для прокату за пунктом прокату, що включає перегляд даних пункту прокату туристичного спорядження.

Власник прокату туристичного спорядження повинен мати доступними сценарії:

- авторизація власника прокату туристичного спорядження, що за передумов відсутності облікового запису для контролю прокату представляється реєстрацією власного пункту прокату туристичного спорядження;
- перегляд даних пункту прокату туристичного спорядження, що за передумов застарілості даних представляє зміну даних;
- відображення туристичного спорядження, доступного для прокату за власним пунктом прокату;
- відображення всього туристичного спорядження за власним пунктом прокату;
- визначення власного туристичного спорядження прокату;
- визначення екземплярів власного туристичного спорядження прокату;
- застосування для власного туристичного спорядження пакетів вартості прокату;
- визначення часу, коли екземпляр туристичного спорядження доступний для прокату;
- керування замовленнями прокату туристичного спорядження за власним пунктом прокату;
- перегляд складу замовлення прокату туристичного спорядження;
- визначення пакетів вартості прокату туристичного спорядження власного пункту прокату (рис. 2.2).

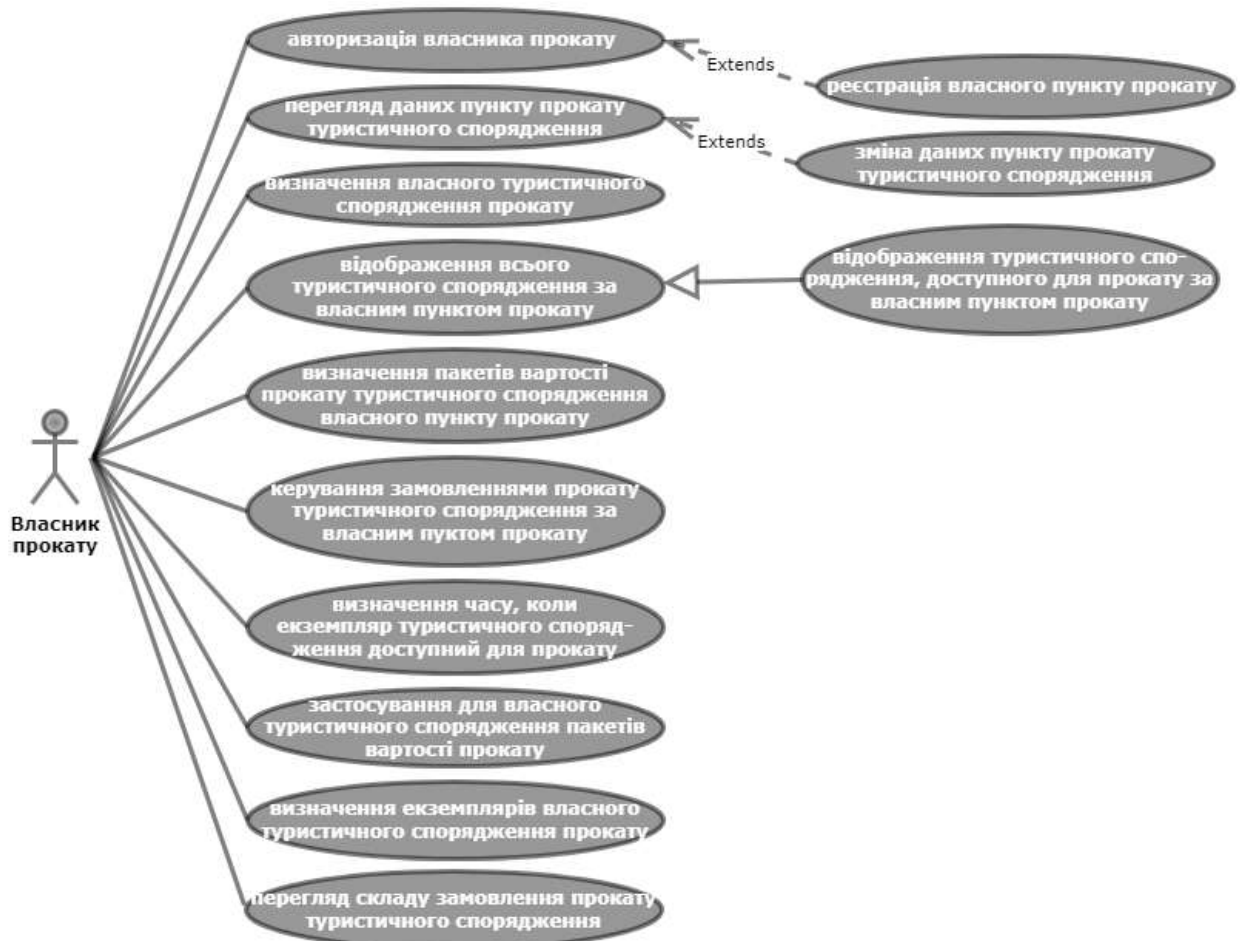


Рисунок 2.2 – Діаграма прецедентів власника прокату туристичного спорядження

Менеджер системи повинен мати доступними сценарії для забезпечення підготовки до роботи інших користувачів:

- авторизація менеджера системи;
- пошук моделей туристичного спорядження;
- створення моделі туристичного спорядження, що за передумов наявності моделі туристичного спорядження в системі та застарілості інформації про неї представляє оновлення моделі туристичного спорядження (рис. 2.3).



Рисунок 2.3 – Діаграма прецедентів менеджера системи прокату

2.3 Проектування структури бази даних

Прокат туристичного спорядження може виконуватися або в окремих пунктах прокату, або у власників такого спорядження. Оскільки, як було прийнято рішення в розділі 1, мережі пунктів прокату не розглядаються, то кожен пункт є окремою одиницею, а на рівні з ними є окремі власники туристичного спорядження, які пропонують власне спорядження для прокату. Це зробило доцільним виділення їхніх даних у таблицю Gearowner з полями:

- id – ідентифікатор, первинний ключ таблиці прокату туристичного спорядження;
- shop_title – назва прокату туристичного спорядження (для прокату від власників назва має бути пропущена, щоб використовувати ім'я самого власника);
- townpoint – місто, де знаходиться прокат або власник туристичного спорядження та саме спорядження;
- detailed_address – адреса, за місцем якої знаходиться туристичне спорядження цього пункту прокату;
- shop_about – опис пункту прокату туристичного спорядження;
- owner_phone – номер телефону, за яким можна контактувати з цим пунктом прокату туристичного спорядження;

Поле townpoint – зовнішній ключ з таблиці Townpoint, тобто представляє зв'язок між таблицями. Це поля представляє місто, де

знаходиться прокат або власник туристичного спорядження та саме спорядження, а самі ці міста знаходяться в таблиці Townpoint з полями:

- id – ідентифікатор, первинний ключ таблиці міст;
- town_name – назва міста.

Оскільки для роботи в програмі потрібно туристичне спорядження певним чином класифікувати, а саме туристичне спорядження представлене різними видами (намети, наплічники, спальні мішки), а кожен вид ще розділяється на типи (наприклад, спальні мішки на зимові, літні, пухові), за якими вже представляються конкретні моделі, то відповідно введено таблиці, які мають представляти такі дані.

Види туристичного спорядження знаходяться в таблиці Gearmenu з полями:

- id – ідентифікатор, первинний ключ таблиці видів туристичного спорядження;
- menu_name – назва виду туристичного спорядження (намети, наплічники, спальні мішки тощо).

Типи туристичного спорядження знаходяться в таблиці Geartype з полями:

- id – ідентифікатор, первинний ключ таблиці типів туристичного спорядження;
- gearmenu – вид туристичного спорядження, до якого належить цей тип, тобто наприклад, спальний мішок для зимового виду;
- type_name – назва виду туристичного спорядження (наприклад, зимовий, літній).

Поле gearmenu – зовнішній ключ з таблиці Gearmenu, тобто представляє зв'язок між таблицями.

Самі ж моделі туристичного спорядження знаходяться в таблиці Gearmodel з полями:

- id – ідентифікатор, первинний ключ таблиці моделей туристичного спорядження;

– `geartype` – тип туристичного спорядження, до якого належить ця модель;

– `model_name` – назва моделі туристичного спорядження;

– `model_about` – опис моделі туристичного спорядження.

Поле `geartype` – зовнішній ключ з таблиці `Geartype`, тобто представляє зв'язок між таблицями.

Однак моделі туристичного спорядження мають дуже різні атрибути. Або точніше – значення цих атрибутів, адже самі атрибути стосуються фактично типів туристичного спорядження. Наприклад, для пухового спальника важливим атрибутом є співвідношення між пухом та пір'ям. Відповідно сам атрибут «співвідношення між пухом та пір'ям» характеризує вид туристичного спорядження. Значення, наприклад, «80/20» характеризує вже саму модель. У той же час такий атрибут не важливий, не потрібний для намета. Це підкреслює, що атрибути стосуються саме типів туристичного спорядження. Тому визначення атрибутів у типах туристичного спорядження визначається в окремій таблиці `Geartypeattribute` з полями:

– `id` – ідентифікатор, первинний ключ таблиці атрибутів типів туристичного спорядження;

– `geartype` – тип туристичного спорядження, до якого належить цей атрибут;

– `attribute_name` – назва атрибуту (співвідношення між пухом та пір'ям, наприклад).

Поле `geartype` – зовнішній ключ з таблиці `Geartype`, тобто представляє зв'язок між таблицями.

Атрибути важливі не принципом свого існування, а конкретними значеннями, що характеризують конкретну модель туристичного спорядження. Саме значення атрибутів моделей туристичного спорядження визначаються в окремій таблиці `Gearmodelattribute` з полями:

– `id` – ідентифікатор, первинний ключ таблиці значення атрибутів моделей туристичного спорядження;

- gearmodel – модель туристичного спорядження, для якої визначено значення атрибуту;
- geartypeattribute – атрибут типу туристичного спорядження, для якого визначено значення за моделлю;
- attribute_value – значення атрибуту моделі туристичного спорядження.

Поле gearmodel – зовнішній ключ з таблиці Gearmodel, тобто представляє зв'язок між таблицями.

Поле geartypeattribute – зовнішній ключ з таблиці Geartypeattribute, тобто представляє зв'язок між таблицями.

Моделі туристичного спорядження представлені конкретним туристичним спорядженням, що знаходиться у власності прокату (конкретного пункту або індивідуального власника, який хоче здати туристичне спорядження на прокат). Дані такого туристичного спорядження знаходяться в таблиці Campgear з полями:

- id – ідентифікатор, первинний ключ таблиці власного туристичного спорядження;
- gearowner – прокат туристичного спорядження, якому належить та який надає в прокат це туристичне спорядження;
- gearmodel – модель, якій належить це туристичне спорядження;
- gear_size – розмір туристичного спорядження (розмір зазвичай є типовим атрибутом, адже його мають і наплічники, і спільні мішки, і наприклад, ті ж самі шоломи, взуття, якщо туристичне спорядження розширювати і до них);
- gear_color – колір туристичного спорядження;
- additional_about – опис туристичного спорядження додатково до загального опису моделі;
- photo – фотографія власного туристичного спорядження.

Поле gearmodel – зовнішній ключ з таблиці Gearmodel, тобто представляє зв'язок між таблицями.

Поле `gearowner` – зовнішній ключ з таблиці `Gearowner`, тобто представляє зв'язок між таблицями.

Тож під власним туристичним спорядженням мається на увазі певне узагальнення туристичного спорядження, але вже з конкретним розміром, конкретним кольором та конкретним власником. У той же час окрім власного туристичного спорядження визначається конкретний екземпляр. Тобто для кожної позиції власного туристичного спорядження може бути декілька конкретних екземплярів, якими володіє власник та які він виставляє на прокат.

Екземпляри власного туристичного спорядження знаходяться в таблиці `Campgearinstance` з полями:

- `id` – ідентифікатор, первинний ключ таблиці екземплярів власного туристичного спорядження;
- `campgear` – власне туристичне спорядження, яке представляє цей екземпляр;
- `closed` – мітка, яка визначає те, чи взагалі надається все ще цей екземпляр туристичного спорядження в прокат;
- `marker` – мітка, яка може визначати характеристики конкретно цього екземпляра туристичного спорядження (наприклад, пошкодження, якісь додаткові ознаки).

Поле `campgear` – зовнішній ключ з таблиці `Campgear`, тобто представляє зв'язок між таблицями.

Якщо мітка `closed` визначає, що цей екземпляр туристичного спорядження не надається в прокат, то для пошуку він вже не має бути доступним, але якщо екземпляр все ще може надаватися в прокат, то далі для нього має бути власником встановлено, на який саме період він надається. Таким чином, якщо власник сам використовує цей екземпляр для власних цілей або якщо він просто протягом певного періоду не може надати його в прокат, то екземпляр не буде доступним і в системі для пошуку. Звичайно, що можна було б для екземпляра визначити період, коли він недоступний,

але в такому випадку весь інший період би вважався доступним. Тобто якщо власник забув зняти певні дати з прокату, то потенційні орендарі будуть думати, що це туристичне спорядження можна орендувати. Тому саме час, коли екземпляр туристичного спорядження доступний для прокату, окремо визначається в зберігається в таблиці `Campgearinstanceavailable` з полями:

- `id` – ідентифікатор, первинний ключ таблиці доступності екземплярів власного туристичного спорядження;
- `campgearinstance` – екземпляр власного туристичного спорядження, для якого встановлено час доступності;
- `available_start` – час, починаючи з якого доступний цей екземпляр туристичного спорядження для прокату;
- `available_end` – час, закінчуючи яким доступний цей екземпляр туристичного спорядження для прокату.

Поле `campgearinstance` – зовнішній ключ з таблиці `Campgearinstance`, тобто представляє зв'язок між таблицями.

Замовлення прокату туристичного спорядження знаходяться в таблиці `Gearapplication` з полями:

- `id` – ідентифікатор, первинний ключ таблиці замовлення прокату туристичного спорядження;
- `applicationuser` – користувач, який замовив прокат туристичного спорядження;
- `gearowner` – прокат туристичного спорядження, за яким створюється замовлення;
- `apply_time` – час замовлення прокату туристичного спорядження;
- `total_price` – сума, яку має бути сплачено за прокат туристичного спорядження;
- `deposit_price` – сума, яку має бути внесено за прокат туристичного спорядження в якості депозиту;
- `rental_date` – дата, коли замовлено початок прокату туристичного спорядження;

- rental_days – дата, коли замовлено завершення прокату туристичного спорядження;
- application_state – стан замовлення прокату туристичного спорядження.

Поле applicationuser – зовнішній ключ з таблиці Applicationuser, тобто представляє зв'язок між таблицями.

Поле gearowner – зовнішній ключ з таблиці Gearowner, тобто представляє зв'язок між таблицями.

Конкретне туристичне спорядження, яке має бути взяте на прокат за замовленням, знаходиться в таблиці Gearapplicationinstance з полями:

- id – ідентифікатор, первинний ключ туристичного спорядження в прокаті;
- gearapplication – замовлення прокату, до якого входить це туристичне спорядження;
- campgear – власне туристичне спорядження, яке обрано для прокату в замовленні;
- quantity – необхідна кількість екземплярів цього туристичного спорядження.

Поле gearapplication – зовнішній ключ з таблиці Gearapplication, тобто представляє зв'язок між таблицями.

Поле campgear – зовнішній ключ з таблиці Campgear, тобто представляє зв'язок між таблицями.

Власні переліки туристичного спорядження, що збережено користувачем, знаходяться в таблиці Campgearlist з полями:

- id – ідентифікатор, первинний ключ переліку туристичного спорядження, що збережено користувачем;
- applicationuser – користувач, який створив цей перелік туристичного спорядження;
- gear_list – назва переліку туристичного спорядження.

Поле `applicationuser` – зовнішній ключ з таблиці `Applicationuser`, тобто представляє зв'язок між таблицями.

Туристичне спорядження, внесене до переліку, збереженого користувачем, знаходиться в таблиці `Campgearlistinstance` з полями:

- `id` – ідентифікатор, первинний ключ туристичного спорядження з переліку туристичного спорядження, що збережено користувачем;
- `campgearlist` – перелік туристичного спорядження, що збережено користувачем, до якого належить це туристичне спорядження;
- `campgear` – власне туристичне спорядження, яке збережено користувачем до цього переліку;
- `quantity` – необхідна кількість цього власного туристичного спорядження, яку зазвичай замовляє на прокат користувач, що необхідно для ситуацій, коли цей перелік використовується для створення типового замовлення.

Поле `campgearlist` – зовнішній ключ з таблиці `Campgearlist`, тобто представляє зв'язок між таблицями.

Поле `campgear` – зовнішній ключ з таблиці `Campgear`, тобто представляє зв'язок між таблицями.

Для простоти визначення типових варіантів вартості прокату визначені пакети вартості. Ці пакети вартості прокату туристичного спорядження кожен власник прокату формує самостійно для себе. Пакети вартості прокату туристичного спорядження знаходяться в таблиці `Typicalpricerack` з полями:

- `id` – ідентифікатор, первинний ключ пакету вартості прокату туристичного спорядження;
- `gearowner` – прокат туристичного спорядження, яким застосовується і відповідно було створено цей пакет вартості прокату;
- `rack_name` – назва пакету вартості прокату туристичного спорядження.

Поле `gearowner` – зовнішній ключ з таблиці `Gearowner`, тобто представляє зв'язок між таблицями.

Самі позиції вартості, що типово застосовуються у пакеті, знаходяться в таблиці `Typicalprice` з полями:

- `id` – ідентифікатор, первинний ключ пакету вартості прокату туристичного спорядження;
- `typicalpricerack` – пакет вартості туристичного спорядження, де визначено цю позицію;
- `min_days` – кількість днів прокату, починаючи з якої починає застосовуватися ця позиція пакету вартості прокату;
- `max_days` – кількість днів прокату, до з якої починає застосовується ця позиція пакету вартості прокату;
- `day_price` – сума, яку потрібно сплатити за день прокату туристичного спорядження;
- `deposit` – сума депозиту, який потрібно внести на час прокату туристичного спорядження.

Поле `typicalpricerack` – зовнішній ключ з таблиці `Typicalpricerack`, тобто представляє зв'язок між таблицями.

Застосування ж позицій пакетів вартості прокату до конкретного власного туристичного спорядження визначається таблицею `Gearprice` з полями:

- `id` – ідентифікатор, первинний ключ застосування пакету вартості прокату до конкретного власного туристичного спорядження;
- `camrgear` – власне туристичне спорядження, прокат якого визначається цією позицією;
- `typicalpricerack` – позиція пакету вартості, яка застосовується для даного туристичного спорядження.

Поле `typicalpricerack` – зовнішній ключ з таблиці `Typicalpricerack`, тобто представляє зв'язок між таблицями.

Поле `campgear` – зовнішній ключ з таблиці `Campgear`, тобто представляє зв'язок між таблицями (рис. 2.4).

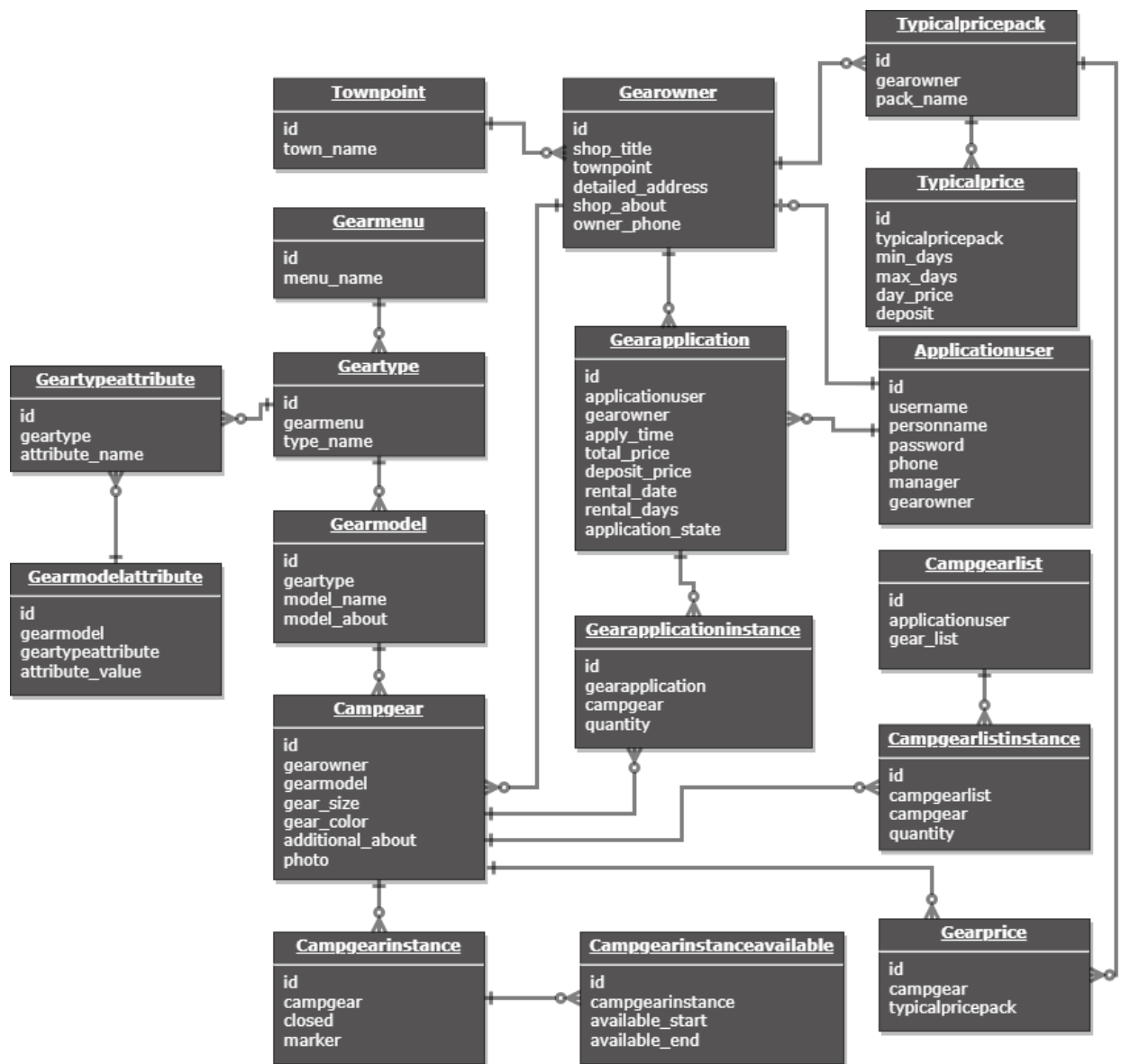


Рисунок 2.4 – Схема бази даних

Дані користувачів системи зберігаються в таблиці `Applicationuser` з полями:

- `id` – ідентифікатор, первинний ключ користувача;
- `username` – логін, який застосовується користувачем за його обліковим записом;

- `personname` – особисте ім'я користувача, який використовує цей обліковий запис;
- `password` – пароль, який застосовується користувачем за його обліковим записом;
- `phone` – номер телефону, який застосовується користувачем за його обліковим записом;
- `manager` – менеджерський запис чи звичайний;
- `gearowner` – прокат туристичного спорядження, який має і представляє цей користувач.

Поле `gearowner` – зовнішній ключ з таблиці `Gearowner`, тобто представляє зв'язок між таблицями.

Реалізацію роботи з базою даних виконано за допомогою безкоштовної системи PostgreSQL [9].

2.4 Висновки за розділом 2

Для розробки ПЗ прокату туристичного спорядження обрано мову Python та фреймворк Django, який було порівняно з фреймворком Flask. Реалізацію роботи з базою даних виконано за допомогою безкоштовної системи PostgreSQL. Сценарії роботи визначено окремо для винаймача туристичного спорядження, власника прокату туристичного спорядження, менеджера системи прокату.

3 РОЗРОБКА ПРОГРАМИ

3.1 Порядок реалізації функцій програми

Для підвищення зручності роботи з програмою при реалізації основних функцій прокату туристичного спорядження було вирішено надати можливість перевірки доступності такого спорядження та визначення вартості прокату. Це є важливим, оскільки фактично майже кожна функція такого ПЗ пов'язана з прокатом як результатом, тому потрібно якомога раніше надати необхідну інформацію про можливість прокату. Тому якщо є функція створення власного типового переліку прокату туристичного спорядження, то необхідно одразу і перевіряти, чи доступне це туристичне спорядження для прокату на даний момент. Тож перевірка доступності та обчислення вартості є типовим інструментом, який має бути вбудованим у інші функціональні можливості. Як наприклад, пошук туристичного спорядження, необхідного для прокату (рис. 3.1), що є прикладом такої функції.

Безпосередньо туристичного спорядження, необхідного для прокату розпочинається з визначення цілого ряду параметрів, що включають:

- місто прокату туристичного спорядження;
- тип туристичного спорядження;
- перелік можливих розмірів туристичного спорядження;
- перелік можливих кольорів туристичного спорядження;
- назву моделі туристичного спорядження;
- період прокату туристичного спорядження.

Спочатку реалізується відфільтровування туристичного спорядження, яке не відповідає цим параметрам, а в підсумку для кожного відібраного спорядження перевіряється, чи доступне воно взагалі для прокату на майбутнє, чи доступне воно на заданий період прокату (хоча би один екземпляр у потрібного прокату). І в кінці обчислюється вартість прокату на задані дати прокату. Отримані результати повертаються.



Рисунок 3.1 – Алгоритм пошуку туристичного спорядження, необхідного для прокату

3.2 Реалізація функцій програми

Програма, як і було визначено в розділі 2, застосовує для реалізації фреймворк Django. Ряд розроблених моделей створено за структурою таблиць бази даних, що були описані в розділі 2.

Представлення та шаблони вебсторінок розроблені за функціями, які надаються програмою.

Відображення діючих пунктів прокату туристичного спорядження за містом реалізовано через функцію `town_gearowners_view`, що підтримує представлення програми, має параметр `idTownpoint`, який визначає ідентифікатор міста прокату, повертає шаблон вебсторінки `towngearowners.html`, заповнений даними, що містять ключ `towns` з інформацією про всі міста програми, ключ `gearowners` з переліком діючих пунктів прокату туристичного спорядження. Пункти прокату туристичного спорядження є діючими, якщо на майбутній час є хоча би один запис про доступність туристичного спорядження за цим пунктом для прокату (не перевіряючи замовлення). Шаблони розроблені за допомогою безкоштовного засобу [10].

Пошук туристичного спорядження, необхідного для прокату, реалізовано через функцію `find_campgears_view`, що підтримує представлення програми, а робота з ним виконується не через параметри функції, а через параметри GET-запиту. Параметри відповідають тим вхідним даним, які були представлені на рис. 3.1.

Перегляд власних замовлень прокатів туристичного спорядження реалізовано через функцію `personal_gearapplications_view`, що підтримує представлення програми, має параметр `sort_variant`, який визначає порядок впорядкування, повертає шаблон вебсторінки `gearapplications.html`, заповнений даними, що містять ключ `gearapplications` з переліком замовлень.

Перегляд всіх власних типових переліків замовлення туристичного спорядження реалізовано через функцію `campgear_lists_view`, що підтримує

представлення програми, повертає шаблон вебсторінки `gearlists.html`, заповнений даними, що містять ключ `gearlists` з всіма переліками замовлення туристичного спорядження, створеними користувачем, якого авторизовано.

Перегляд складу типового переліку замовлення туристичного спорядження реалізовано через функцію `campgear_list_view`, що підтримує представлення програми, повертає шаблон вебсторінки `gearlist.html`, передаючи до нього дані за ключем `gearlist` з даними самого переліку, `gearinstances` з даними туристичного спорядження з типового переліку, де додатково за атрибутами `price` та `deposit` знаходяться відповідно вартість прокату такого туристичного спорядження на потрібну кількість днів та депозит, який необхідно внести, відповідно. Тобто представлення влаштоване таким чином, що одразу розраховується вартість замовлення за типовим переліком на заданий період дат або на наступний день, якщо вони не вказані.

Перегляд складу власного замовлення прокату туристичного спорядження реалізовано через функцію `personal_gearapplication_view`, що підтримує представлення програми, повертає шаблон вебсторінки `gearapplication.html`, передаючи до нього дані за ключем `gearapplication` з даними самого замовлення, а в атрибуті `instances` містить туристичне спорядження, яке входить до цього замовлення.

Створення власного типового переліку замовлення туристичного спорядження на основі створеного раніше власного замовлення прокату реалізовано через функцію `create_list_gearapplication_view`, що підтримує представлення програми, має параметр `gearapplicationId` з номером створеного раніше замовлення прокату туристичного спорядження, створює відповідний перелік, додає позиції туристичного спорядження у кількості з замовлення, повертає шаблон вебсторінки `gearlist.html`, передаючи до нього дані за ключем `gearlist` з даними самого створеного переліку, `gearinstances` з даними туристичного спорядження зі створеного типового переліку.

Замовлення прокату туристичного спорядження реалізовано через функцію `create_gearapplication_view`, що підтримує представлення програми, в якому дата початку прокату передається через параметр `rental_begin` з GET-запиту, відповідно дата завершення прокату – `rental_end` (якщо ці дати не визначені, то визначається найближча доба), перелік ідентифікаторів туристичного спорядження – `gears`, кількість за кожною позицією – `quantities`. Саме замовлення разом з його складом створюється тільки для авторизованого користувача, а його сумарна вартість обчислюється у представленні за заданою датою та поточною вартістю тривалості прокату туристичного спорядження у даного власника. Якщо туристичне спорядження належить не одному власнику, то замовлення не створюється. Так само воно не створюється, якщо екземплярів такого туристичного спорядження у власника, вільних для прокату на потрібні дати, недостатньо.

Скасування власного замовлення прокату туристичного спорядження реалізовано через функцію `cancel_personal_gearapplication_view`, що підтримує представлення програми з параметром `gearapplicationId`, який задає ідентифікатор замовлення прокату туристичного спорядження. Скасування реалізується тільки, якщо власник замовлення прокату туристичного спорядження є авторизованим користувачем, а замовлення або тільки створене, або вже підтверджено власником прокату, але час до початку прокату не менший за 10 діб.

Вилучення власного типового переліку замовлення туристичного спорядження реалізовано через функцію `delete_gearlist_view`, що підтримує представлення програми з параметром `gearlistId`, який задає ідентифікатор типового переліку замовлення туристичного спорядження. Вилучення здійснюється тільки тоді, коли заданий типовий перелік замовлення туристичного спорядження належить авторизованому користувачу.

Вилучення туристичного спорядження з типового переліку замовлення туристичного спорядження реалізовано через функцію `delete_gear_from_gearlist_view`, що підтримує представлення програми з

параметром `campgearlistId`, який задає ідентифікатор туристичного спорядження в типовому переліку замовлення туристичного спорядження. Вилучення здійснюється тільки тоді, коли позиція туристичного спорядження належить заданому типовому переліку замовлення туристичного спорядження, що в свою чергу належить авторизованому користувачу. Після цього повертається вебсторінка з оновленим складом типового переліку замовлення туристичного спорядження та розрахунком вартості на наступну добу.

Зміну кількості туристичного спорядження в типовому переліку замовлення туристичного спорядження реалізовано через функцію `change_gear_from_gearlist_view`, що підтримує представлення програми з параметром `campgearlistId`, що задає ідентифікатор туристичного спорядження в типовому переліку замовлення туристичного спорядження, параметром `gearquantity`, що задає оновлену типову кількість за цією позицією у типовому переліку.

Відображення туристичного спорядження, доступного для прокату за пунктом прокату, реалізовано через функцію `owner_gears_view`, що підтримує представлення програми з передачею ідентифікатора прокату через параметр `ownerid` з GET-запиту, дати початку прокату через параметр `rental_begin`, дати завершення прокату через параметр `rental_end`. Відображається туристичне спорядження, що доступне хоча би в одному екземплярі за таким пунктом на задані дати.

Замовлення прокату туристичного спорядження на основі власного типового переліку замовлення реалізовано через функцію `create_gearapplication_from_list_view`, що підтримує представлення програми з передачею ідентифікатора власного типового переліку замовлення прокату через параметр `list_id` з GET-запиту, дати початку прокату через параметр `rental_begin`, дати завершення прокату через параметр `rental_end`. Саме замовлення разом з його складом створюється тільки для авторизованого користувача, який є власником цього переліку, а сумарна вартість

замовлення обчислюється у представленні за заданою датою та поточною вартістю тривалості прокату туристичного спорядження у даного власника. Замовлення не створюється, якщо екземплярів такого туристичного спорядження у власника, вільних для прокату на потрібні дати, недостатньо.

Створення власного типового переліку замовлення туристичного спорядження на основі створеного раніше власного замовлення прокату реалізовано через функцію `create_list_gearapplication_view`, що підтримує представлення програми.

Перегляд туристичного спорядження реалізовано через функцію `campgear_demo_view`, що підтримує представлення програми з передачею дати початку прокату через параметр `rental_begin` GET-запиту, дати завершення прокату через параметр `rental_end`. Виконується перевірка наявності цього туристичного спорядження на потрібні дати, обчислюється вартість прокату.

Пошук моделей туристичного спорядження реалізовано через функцію `find_gear_models_view`, що підтримує представлення програми, а робота з ним виконується не через параметри функції, а через параметри GET-запиту. Параметри включають `gear_type`, що визначає ідентифікатор потрібного типу туристичного спорядження, `gear_menu`, що визначає ідентифікатор потрібного виду туристичного спорядження, `name`, що визначає назву моделі туристичного спорядження.

Відображення туристичного спорядження, доступного для прокату за власним пунктом прокату, реалізовано через функцію `own_owner_gears_view`, що підтримує представлення програми з передачею дати початку прокату через параметр `rental_begin` з GET-запиту, дати завершення прокату через параметр `rental_end`. Відображається туристичне спорядження, що доступне хоча би в одному екземплярі пунктом, яким володіє авторизований користувач, на задані дати.

Відображення всього туристичного спорядження за власним пунктом прокату реалізовано через функцію `all_own_owner_gears_view`. У даному

представленні вже відбір за датами доступного туристичного спорядження не відбувається.

3.3 Висновки за розділом 3

ПЗ прокату туристичного спорядження реалізовано в повній мірі. Реалізовано, представлено і описано основні представлення, які забезпечують підтримку роботи винаймача туристичного спорядження, власника прокату туристичного спорядження, менеджера системи прокату.

4 ЕКСПЛУАТАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМИ

4.1 Призначення програми

Розроблений вебдодаток призначений для підтримки замовлення прокату туристичного спорядження через надання можливостей винаймачам, які таке туристичне спорядження потребують, та власникам туристичного спорядження, які можуть або представляти певні пункти прокату, або представляти себе як індивідуальних власників.

4.2 Умови виконання програми

Технічні засоби визначені на мінімальному рівні:

- процесор не менше чотирьохядерного з частотою 2,0 ГГц;
- оперативна пам'ять – 4 Гб;
- жорсткий диск – 20 Гб простору.

Експлуатація програми операторами в різних ролях не потребує особливих навичок окрім навичок роботи з вебдодатками, але налаштування серверу має враховувати навички у адміністратора для роботи з системою PostgreSQL, керування фреймворком Django (зокрема і роботи з адміністративною панеллю), інтерпретатором Python.

4.3 Виконання і тестування програми

Для підтримки тестування було створено сервер, на якому запущено вебдодаток. Доступ до нього було перевірено в різних режимах з різних пристроїв, у тому числі типів пристроїв: персонального комп'ютера, мобільного телефону.

ПЗ прокату туристичного спорядження за звернення за базовою адресою відображається для винаймача. Головне меню включає кнопку “Увійти”, що перенаправляє на сторінку авторизації. В залежності від типу

облікового запису (звичайний – винаймач, наявний вписаний прокат – власник прокату, наявна відмітка менеджера – менеджер системи) далі використовується один з трьох варіантів подальшої роботи. Якщо працює винаймач (рис. 4.1), то його головне меню включає “Пункти”, “Пошук”, “Переліки”, “Замовлення”, “Немає замовлення” або “Замовити”, “Вийти”.

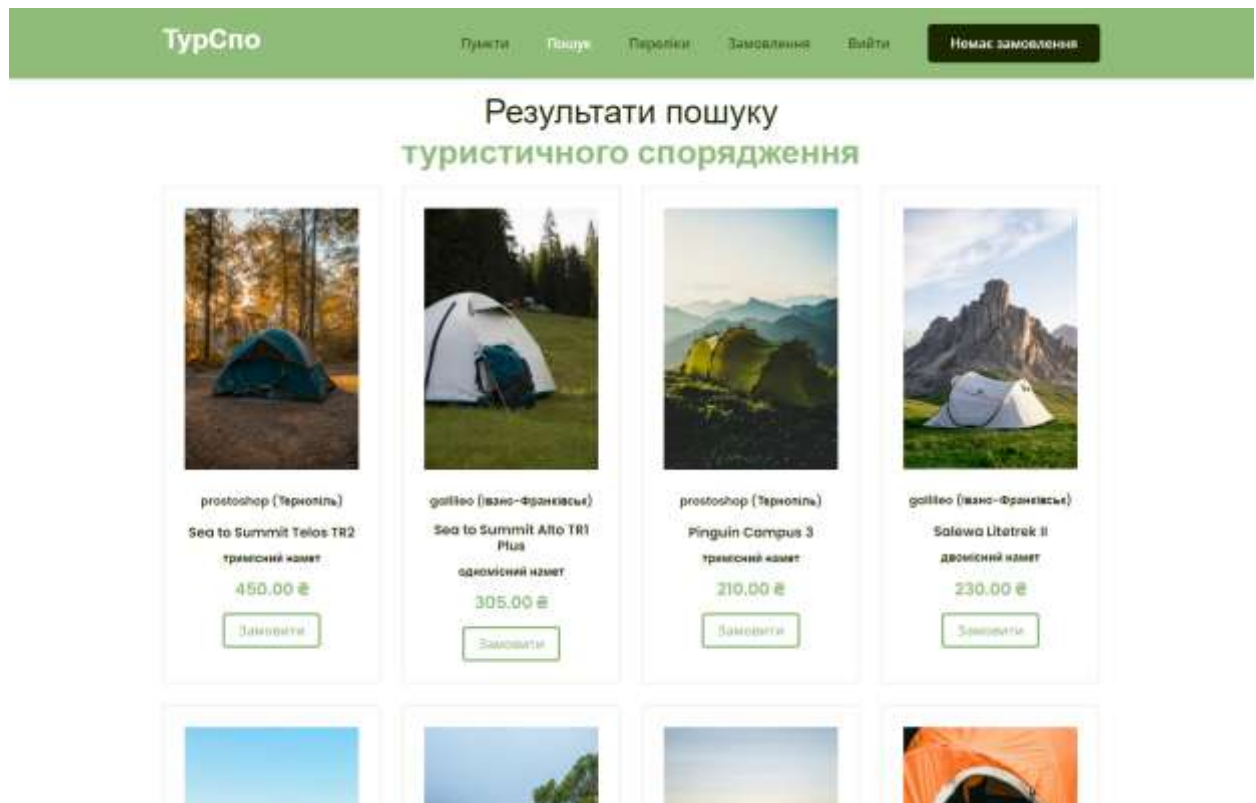


Рисунок 4.1 – Пошук туристичного спорядження

За пунктом “Пункти” показується поле для визначення міста, тоді за результатом показується перелік пунктів прокату туристичного спорядження, які є активними у обраному місті.

Пункт “Пошук” відображає сторінку, де задаються параметри пошуку, включаючи місто прокату, тип спорядження, розмір, колір, назву моделі, період прокату. Будь-які з цих пунктів можуть бути пропущені. Результати пошуку (рис. 4.1) містять для кожного туристичного спорядження вартість прокату на задані дати. Кожну позицію можна додати до замовлення. Кнопка “Замовити” дозволяє це зробити. Якщо до замовлення не додано елементів,

то в головному меню показується кнопка “Немає замовлення”. Якщо до замовлення додано туристичне спорядження, то показується кнопка “Замовити”. Замовлення реалізується тільки тоді, коли воно стосується прокату тільки в одному й тому самому пункті прокату.

За кожним туристичним спорядженням з пошуку показується пункт прокату під картинкою, назва моделі та тип туристичного спорядження. Пункт прокату так само як і зі сторінки “Пункти” можна відкрити та переглянути актуальну інформацію про пункт прокату, включаючи туристичне спорядження, доступне для прокату за цим пунктом. Якщо натиснути на картинку спортивного спорядження, то відкриється окрема сторінка (рис. 4.2), де відображається інформація про конкретне спорядження, пункт прокату, картинку спортивного спорядження, його основні атрибути зі значеннями (з лівого та правого боків від картинки), опис.

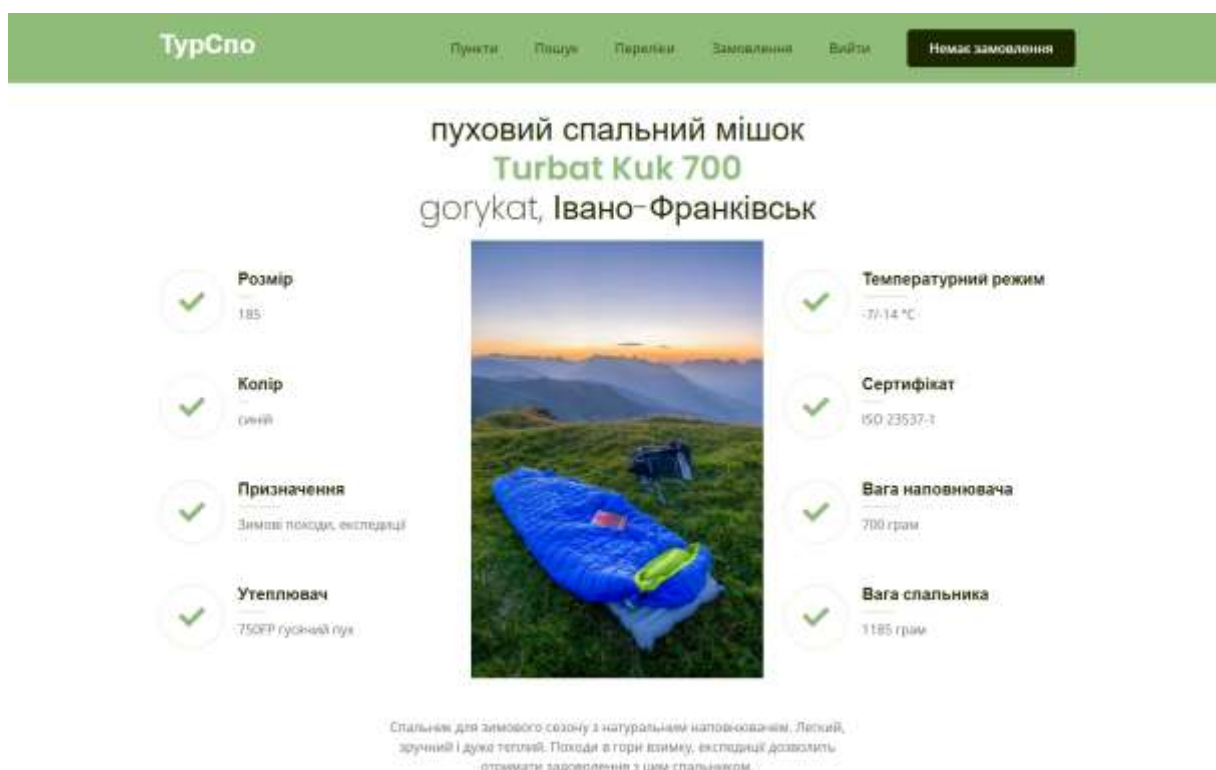


Рисунок 4.2 – Перегляд туристичного спорядження

Нижче знаходиться форма для визначення дат прокату та обчислюється вартість прокату на ці дати, необхідний депозит. Також доступний перелік для додавання цього туристичного спорядження до типового переліку замовлення прокату туристичного спорядження.

Пункт “Переліки” містить всі типові переліки замовлення прокату туристичного спорядження, які були створені авторизованим споживачем. Кожен з переліків з тієї сторінки можна вилучити, також створити новий, а також відкрити один зі створених (рис. 4.3).

ТурСпо Пункти Пошук Переліки Замовлення Вийти **Замовити**

Перелік: lightset

Початок: 18.06.2024 Кінець: 20.06.2024

Замовити

Вартість: 1260.00 ₴

Депозит: 9400.00 ₴

gorykat, Івано-Франківськ Turbat Kuk 700 пуховий спальний мішок ❖ класифікація: 2 720.00 ₴ Забрати	gorykat, Івано-Франківськ Salewa Litetrek II двомісний намет ❖ класифікація: 1 420.00 ₴ Забрати	gorykat, Івано-Франківськ Kelty Upslope 2 трекінгові палики ❖ класифікація: 2 120.00 ₴ Забрати
---	--	---

Вилучити весь перелік

Рисунок 4.3 – Перегляд складу типового переліку замовлення

При перегляді складу типового переліку замовлення прокату туристичного спорядження можна задати дати прокату, тоді буде

розраховано вартість прокату, необхідний депозит. Кнопка “Змінити” біля кожної позиції дозволяє відкрити форму редагування кількості одиниць туристичного спорядження. Біля кожної позиції показується поточна кількість. За цією кнопкою таку кількість можна змінити на потрібну. Переобчислення вартості за зміни дат відбувається автоматично. Кнопка “Замовити” дозволяє сформуванню замовлення прокату туристичного спорядження не на основі поточного замовлення, куди додавалось туристичне спорядження як в кошик, а на основі даного типового переліку, якщо це спорядження доступне для прокату. Кнопка “Вилучити весь перелік” дозволяє вилучити типовий перелік замовлення прокату туристичного спорядження.

Пункт “Замовлення” дозволяє переглядати та керувати власними замовленнями прокату туристичного спорядження. Якщо замовлення тільки створене або було підтверджено пунктом прокату, але до прокату залишається не менше 10 діб, то воно може бути скасоване. Окрім того з переліку можна відкрити замовлення та переглянути його склад повністю, створити типовий перелік замовлення прокату туристичного спорядження на основі відповідної кнопки.

Якщо авторизований користувач є власником прокату туристичного спорядження, то його головне меню складається вже з пунктів: “Опис”, звідки можна отримати доступ до опису власного пункту, “Спорядження”, звідки можна переглянути набір наявного туристичного спорядження на пункті прокату, додати нове, вилучити існуюче, отримати інформацію про екземпляри цього спорядження, додати нові екземпляри, визначити тарифи вартості прокату, що застосовуються до туристичного спорядження, “Вартість”, звідки можна переглянути загальний перелік тарифів вартості прокату туристичного спорядження, змінити створені тарифи, “Замовлення”, звідки можна отримати доступ до переліку замовлень прокату туристичного спорядження (рис. 4.4).

Номер, дата	Замовник, прокат	Стан, вартість	
#32 22.05.2024	Пітер Полєнко 24.06.2024 - 25.06.2024	створене #10.00 #	➤ Підтвердити ➤ Скасувати
#31 22.05.2024	Андрій Щирій 20.06.2024 - 27.06.2024	створене 2800.00 #	➤ Підтвердити ➤ Скасувати

Рисунок 4.4 – Перегляд замовлень прокату туристичного спорядження за власним пунктом прокату

Кожне замовлення можна окремо відкрити, переглянути склад і деталізовану інформацію. З загального переліку доступна інформація про номер замовлення, дату створення, замовника, дати, протягом яких відбувається прокат, стан замовлення, вартість замовлення. Для кожного замовлення надаються засоби управління, що включають підтвердження або скасування для створеного замовлення, видачу та скасування для підтвердженого тощо.

Якщо авторизований користувач є менеджером системи прокату, то його основна сторінка містить засоби пошуку моделей туристичного спорядження, звідки одночасно їх можна змінювати, додавати, що забезпечує загальне керування.

4.4 Висновки за розділом 4

Результати тестування ПЗ прокату туристичного спорядження, що було проведено в даному розділі, включають успішне виконання сценаріїв, які повинні були бути забезпечені для винаймача туристичного спорядження, власника прокату туристичного спорядження, менеджера системи прокату, з описом логіки виконання відповідних дій.

ВИСНОВКИ

У роботі розроблено вебдодаток прокату туристичного спорядження за допомогою мови Python та фреймворку Django, який порівняно з аналогами має особливості, що включають об'єднання можливостей прокату від конкретних пунктів і від власників туристичного спорядження та можливість створення типового замовлення прокату туристичного спорядження на основі визначено шаблону, що повністю відповідає меті дипломної кваліфікаційної роботи та дозволило досягти її.

Розроблене ПЗ дозволяє виконувати реєстрацію власного пункту прокату туристичного спорядження, винаймача, перегляд і зміну даних пункту прокату туристичного спорядження, відображення діючих пунктів прокату туристичного спорядження за містом, пошук туристичного спорядження, необхідного для прокату, замовлення прокату туристичного спорядження, в тому числі на основі власного типового переліку замовлення, керування власними замовленнями прокату туристичного спорядження, створення власного типового переліку замовлення туристичного спорядження, у тому числі на основі створеного раніше власного замовлення прокату, перегляд власних типових переліків замовлення туристичного спорядження та складу конкретного переліку, додавання, вилучення туристичного спорядження з типового переліку, зміну кількості туристичного спорядження в типовому переліку, вилучення власного типового переліку замовлення туристичного спорядження, перегляд власних замовлень прокатів туристичного спорядження, перегляд туристичного спорядження, перегляд складу замовлення прокату туристичного спорядження, відображення туристичного спорядження, доступного для прокату за пунктом прокату, відображення всього туристичного спорядження за пунктом прокату, створення, оновлення моделі туристичного спорядження, пошук моделей туристичного спорядження, визначення власного туристичного спорядження прокату і його екземплярів та

застосування для нього конкретних пакетів вартості прокату, визначення часу, коли екземпляр туристичного спорядження доступний для прокату, керування замовленнями прокату туристичного спорядження за власним пунктом прокату, визначення конкретних пакетів вартості прокату туристичного спорядження пункту прокату.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Rent Outdoor Gear [Electronic resource]. – Access mode : <https://rentoutdoorgear.com/>.
2. Прокат туристичного спорядження – Робінзон – магазин товарів для туризму та спорту [Електрон. ресурс]. – Режим доступу : <https://www.robinzon-ua.com.ua/ukr/prokat-turistichnogo-sporjadzhennja/>.
3. Outdoor Equipment Rentals – Adventure. Anywhere [Electronic resource]. – Access mode : <https://www.oerentals.com/>.
4. Outdoorhire – Outdoor Equipment Hire [Electronic resource]. – Access mode : <https://www.outdoorhire.co.uk/>.
5. Mohan, M. The Python Journey [Text] : A Comprehensive Introduction to the Python Language / M. Mohan. – North Salt Lake : Siri Sarah LLC, 2023. – 103 p.
6. Dinder, M. Becoming an Enterprise Django Developer [Text] : Discover best practices, tooling, and solutions for writing and organizing Django applications in production / M. Dinder. – Birmingham : Packt Publishing, 2022. – 526 p.
7. Adedeji, O. Full-Stack Flask and React [Text] : Learn, code, and deploy powerful web applications with Flask 2 and React 18 / O. Adedeji. – Birmingham : Packt Publishing, 2023. – 408 p.
8. Aggarwal, S. Flask Framework Cookbook – Third Edition [Text] : Enhance your Flask skills with advanced techniques and build dynamic, responsive web applications / S. Aggarwal. – Birmingham : Packt Publishing, 2023. – 318 p.
9. Juba, S. Learning PostgreSQL 11 – Third Edition [Text] : A beginner's guide to building high-performance PostgreSQL database solutions, 3rd Edition / S. Juba, A. Volkov. – Birmingham : Packt Publishing, 2019. – 556 p.
10. Hairnic – Single Product Website Template [Electronic resource]. – Access mode : <https://htmlcodex.com/single-product-website-template/>.

ДОДАТОК А
Технічне завдання

Вступ

Туристичне спорядження є достатньо вартісним, а використовується достатньо рідко, що знижує ефективність його придбання для себе. У підсумку більшу частину часу туристичне спорядження не використовується. Це призводить до збільшення необхідності в прокаті туристичного спорядження винаймачами, але і з іншого боку – до потреби здати в прокат власне туристичне спорядження, щоб підвищити ефективність від його придбання. Об'єднання цих двох проблем вимагає створення ПЗ, здатного забезпечувати потрібні для цього функції. Користувачами ж такої програми мають бути винаймачі туристичного спорядження та власники туристичного спорядження.

А.1 Підстави для розробки

Підстави розробки за дипломною кваліфікаційною роботою визначені та затверджені наказом № 168 від 24 квітня 2024 р. за Національним університетом «Запорізька політехніка», у якому зафіксовано тему «Програмне забезпечення прокату туристичного спорядження».

А.2 Призначення розробки

Розробка призначена для підтримки замовлення прокату туристичного спорядження через надання можливостей винаймачам, які таке туристичне спорядження потребують, та власникам туристичного спорядження, які можуть або представляти певні пункти прокату, або представляти себе як індивідуальних власників.

A.3 Основні вимоги до програми

A.3.1 Вимоги до функціональних характеристик

Підтримка роботи користувача з програмою має забезпечуватися наступними функціями:

- реєстрація власного пункту прокату туристичного спорядження;
- реєстрація винаймача;
- авторизація користувача;
- перегляд і зміна даних пункту прокату туристичного спорядження;
- відображення діючих пунктів прокату туристичного спорядження за містом;
- пошук туристичного спорядження, необхідного для прокату;
- замовлення прокату туристичного спорядження, в тому числі на основі власного типового переліку замовлення;
- керування власними замовленнями прокату туристичного спорядження;
- створення власного типового переліку замовлення туристичного спорядження, у тому числі на основі створеного раніше власного замовлення прокату;
- перегляд власних типових переліків замовлення туристичного спорядження та складу конкретного переліку;
- додавання, вилучення туристичного спорядження з типового переліку;
- зміна кількості туристичного спорядження в типовому переліку;
- вилучення власного типового переліку замовлення туристичного спорядження;
- перегляд власних замовлень прокатів туристичного спорядження;
- перегляд туристичного спорядження;
- перегляд складу замовлення прокату туристичного спорядження;

- відображення туристичного спорядження, доступного для прокату за пунктом прокату;
- відображення всього туристичного спорядження за пунктом прокату;
- створення, оновлення моделі туристичного спорядження;
- пошук моделей туристичного спорядження;
- визначення власного туристичного спорядження прокату і його екземплярів та застосування для нього пакетів вартості прокату;
- визначення часу, коли екземпляр туристичного спорядження доступний для прокату;
- керування замовленнями прокату туристичного спорядження за пунктом прокату;
- визначення пакетів вартості прокату туристичного спорядження пункту.

A.3.2 Умови експлуатації

Експлуатація програми операторами в різних ролях не потребує особливих навичок окрім навичок роботи з вебдодатками, але налаштування серверу має враховувати навички у адміністратора для роботи з системою PostgreSQL, керування фреймворком Django (зокрема і роботи з адміністративною панеллю), інтерпретатором Python.

A.3.3 Вимоги до складу та параметрів технічних засобів

Технічні засоби визначені на мінімальному рівні:

- процесор не менше чотирьохядерного з частотою 2,0 ГГц;
- оперативна пам'ять – 4 Гб;
- жорсткий диск – 20 Гб простору.

А.4 Порядок контролю та приймання

Строки виконання та результати виконання за кожним етапом дипломної кваліфікаційної роботи визначаються календарним планом у завданні на роботу та контролюються керівником роботи, який закріплений наказом. Захист роботи реалізується перед екзаменаційною комісією як відкрита процедура.

ДОДАТОК Б
Текст програми

```

from .models import Gearowner
from .models import Townpoint
from .models import Campgearinstanceavailable
from .models import Campgearlist
from .models import Campgearlistinstance
from .models import Applicationuser
from .models import Geartype
from .models import Gearapplicationinstance
from .models import Gearprice
from .models import Typicalprice
from django.shortcuts import render

def town_gearowners_view(request, idTownpoint = 0):
    town_gearowners_result = []
    towns = Townpoint.objects.all()
    town = Townpoint.objects.filter(id = idTownpoint)
    town_gearowners = Gearowner.objects.filter(townpoint =
town)

    town_gearowners =
town_gearowners.order_by("shop_title")
    import datetime
    time_point = datetime.datetime.now()
    for gearowner in town_gearowners:
        availables =
Campgearinstanceavailable.objects.filter(campgearinstance__campg
ear__gearowner = gearowner,
            available_start__lte = time_point,
            available_end__gte = time_point)
        if availables:
            town_gearowners_result.append(gearowner)
    return render(request,
"outequprog/towngearowners.html",
        {"towns": towns,
        "gearowners": town_gearowners_result
        })

def campgear_lists_view(request):
    logged_user = request.user
    application_user =
Applicationuser.objects.filter(username =
logged_user.username).first()
    campgear_lists =
Campgearlist.objects.filter(applicationuser = application_user)
    campgear_lists = campgear_lists.order_by("gear_list")
    return render(request, "outequprog/gearlists.html",
        {"gearlists": campgear_lists })

def campgear_list_view(request):

```

```

        idGearlist = self.request.GET.get('gearlist', None)
        idGearlist = int(idGearlist)
        logged_user = request.user
        application_user =
Applicationuser.objects.filter(username =
logged_user.username).first()
        gearrental_begin_date =
self.request.GET.get('rental_begin', None)
        if gearrental_begin_date:
            gearrental_begin_date_parts =
gearrental_begin_date.split('.')
            gearrental_begin_date =
datetime.datetime(int(gearrental_begin_date_parts[0]),
                    int(gearrental_begin_date_parts[1]),
                    int(gearrental_begin_date_parts[2]))
        else:
            gearrental_begin_date = datetime.datetime.now()
        gearrental_end_date =
self.request.GET.get('rental_end', None)
        if gearrental_end_date:
            gearrental_end_date_parts =
gearrental_end_date.split('.')
            gearrental_end_date =
datetime.datetime(int(gearrental_end_date_parts[0]),
                    int(gearrental_end_date_parts[1]),
                    int(gearrental_end_date_parts[2]))
        else:
            gearrental_end_date = gearrental_begin_date +
datetime.timedelta(days = 1)
        rental_days = gearrental_end_date -
gearrental_begin_date
        rental_days = rental_days.days
        campgear_list = Campgearlist.objects.filter(id =
idGearlist).first()
        if campgear_lists.applicationuser == application_user:
            gearinstances =
Campgearlistinstance.objects.filter(campgearlist =
campgear_list).select_related("campgear")
            gearinstances = gearinstances.order_by("id")
            for gearinstance in gearinstances:
                availables =
Campgearinstanceavailable.objects.filter(campgearinstance__campg
ear = gearinstance.campgear,
                available_start__lte =
gearrental_begin_date,
                available_end__gte = gearrental_end_date)
            if not availables:
                gearinstance.available = False

```

```

        else:
            instances_number =
Campgearinstance.objects.filter(campgear =
gearinstance.campgear, closed = False).count()
            camprent_date = gearrental_begin_date
            noavailable = False
            while camprent_date < gearrental_end_date:
                applications =
Gearapplicationinstance.objects.filter(campgear =
gearinstance.campgear,
                                gearapplication__rental_date__lte =
camprent_date,
                                gearapplication__rental_days__gte =
camprent_date,
                                gearapplication__application_state__in = [1,3])
                rent_number = 0
                for rent_application in applications:
                    rent_number = rent_number +
rent_application.quantity
                if (instances_number - rent_number) <
gearinstance.quantity:
                    gearinstance.available = False
                    noavailable = True
                    break
                    camprent_date = camprent_date +
datetime.timedelta(days = 1)
                    if not noavailable:
                        gearinstance.available = True
                    for gearinstance in gearinstances:
                        if gearinstance.available:
                            camprent_date = gearrental_begin_date
                            pack = Gearprice.objects.filter(campgear =
gearinstance.campgear).first()
                            prices =
Typicalprice.objects.filter(typicalpricepack = pack,
min_days__lte = rental_days, max_days__gte =
rental_days).first()
                            gearinstance.deposit = prices.deposit
                            gearinstance.price = prices.day_price *
rental_days
            return render(request, "outequiprog/gearlist.html",
                {"gearlist": campgear_list,
                 "gearinstances": gearinstances
                })

def find_campgears_view(request):
    campgear_town = self.request.GET.get('townid', None)

```

```

        campgears =
Campgear.objects.all().select_related('gearowner', 'gearmodel',
'gearmodel__geartype', 'gearmodel__geartype__gearmenu')
        if campgear_town:
            campgear_town = int(campgear_town)
            town_point = Townpoint.objects.filter(id =
campgear_town)
            campgears = campgears.filter(gearowner__townpoint =
town_point)
            gear_types = self.request.GET.get('geartypes', None)
            if gear_types:
                gear_types = gear_types.split(',')
                gear_types_int = []
                for gear_type in gear_types:
                    gear_types_int.append(int(gear_type))
                gear_types = Geartype.objects.filter(id__in =
gear_types_int)
                campgears =
campgears.filter(gearmodel__geartype__in = gear_types)
            gear_sizes = self.request.GET.get('gearsizes', None)
            if gear_sizes:
                gear_sizes = gear_sizes.split(',')
                campgears = campgears.filter(gear_size__in =
gear_sizes)
            gear_colors = self.request.GET.get('gearcolors', None)
            if gear_colors:
                gear_colors = gear_colors.split(',')
                campgears = campgears.filter(gear_color__in =
gear_colors)
            gear_name = self.request.GET.get('gearname', None)
            if gear_name:
                campgears = campgears.filter(gearmodel__model_name
= gear_name)
            import datetime
            gearrental_begin_date =
self.request.GET.get('rental_begin', None)
            gearrental_begin_date_parts =
gearrental_begin_date.split('.')
            gearrental_begin_date =
datetime.datetime(int(gearrental_begin_date_parts[0]),
                    int(gearrental_begin_date_parts[1]),
                    int(gearrental_begin_date_parts[2]))
            gearrental_end_date =
self.request.GET.get('rental_end', None)
            gearrental_end_date_parts =
gearrental_end_date.split('.')
            gearrental_end_date =
datetime.datetime(int(gearrental_end_date_parts[0]),

```

```

        int(gearrental_end_date_parts[1]),
        int(gearrental_end_date_parts[2]))
    rental_days = gearrental_end_date -
gearrental_begin_date
    rental_days = rental_days.days
    campgear_results = []
    for campgear in campgears:
        availables =
Campgearinstanceavailable.objects.filter(campgearinstance__campg
ear = campgear,
        available_start__lte = gearrental_begin_date,
        available_end__gte = gearrental_end_date)
    if availables:
        instances_number =
Campgearinstance.objects.filter(campgear = campgear, closed =
False).count()
        camprent_date = gearrental_begin_date
        noavailable = False
        min_gears = instances_number
        while camprent_date < gearrental_end_date:
            applications =
Gearapplicationinstance.objects.filter(campgear = campgear,
            gearapplication__rental_date__lte =
camprent_date,
            gearapplication__rental_days__gte =
camprent_date,
            gearapplication__application_state__in
= [1,3])
            rent_number = 0
            for rent_application in applications:
                rent_number = rent_number +
rent_application.quantity
            if rent_number == instances_number:
                noavailable = True
                break
            elif instances_number - rent_number <=
min_gears:
                min_gears = instances_number -
rent_number
                camprent_date = camprent_date +
datetime.timedelta(days = 1)
            if not noavailable:
                campgear.quantity = min_gears
                campgear_results.append(campgear)
    for campgear in campgear_results:
        pack = Gearprice.objects.filter(campgear =
gearinstance.campgear).first()

```

```

        prices =
Typicalprice.objects.filter(typicalpricepack = pack,
min_days__lte = rental_days, max_days__gte =
rental_days).first()
        campgear.deposit = prices.deposit
        campgear.price = prices.day_price * rental_days
        campgear_results = campgear_results.sort(key = lambda
x, y: x.price >= y.price)
        gear_towns =
Townpoint.objects.all().order_by("town_name")
        gear_menus =
Gearmenu.objects.all().order_by("menu_name")
        for gear_menu in gear_menus:
            gear_menu.types = Geartype.objects.filter(gearmenu
= gear_menu).order_by("type_name")
        return render(request, "outequprog/gearsearch.html",
            {"campgears": campgear_results,
            "towns": gear_towns,
            "menus": gear_menus
            })

def personal_gearapplications_view(request, sort_variant =
0):
    logged_user = request.user
    application_user =
Applicationuser.objects.filter(username =
logged_user.username).first()
    if sort_variant == 0:
        gear_applications =
Gearapplication.objects.filter(applicationuser =
application_user).order_by("-rental_date")
    elif sort_variant == 1:
        gear_applications =
Gearapplication.objects.filter(applicationuser =
application_user).order_by("rental_date")
    elif sort_variant == 2:
        gear_applications =
Gearapplication.objects.filter(applicationuser =
application_user).order_by("apply_time")
    elif sort_variant == 3:
        gear_applications =
Gearapplication.objects.filter(applicationuser =
application_user).order_by("-apply_time")
    elif sort_variant == 4:
        gear_applications =
Gearapplication.objects.filter(applicationuser =
application_user).order_by("total_price")
    elif sort_variant == 5:

```

```

        gear_applications =
Gearapplication.objects.filter(applicationuser =
application_user).order_by("-total_price")
        return render(request,
"outequprog/gearapplications.html",
        {"gearapplications": gear_applications
        })

    def personal_gearapplication_view(request,
gearapplicationId):
        logged_user = request.user
        application_user =
Applicationuser.objects.filter(username =
logged_user.username).first()
        gear_application = Gearapplication.objects.filter(id =
gearapplicationId).select_related("applicationuser").first()
        if gear_application.applicationuser ==
application_user:
            instances =
Gearapplicationinstance.objects.filter(gearapplication =
gear_application)
            gear_application.instances = instances
            return render(request,
"outequprog/gearapplication.html",
            {"gearapplication": gear_application
            })
        else:
            return render(request,
"outequprog/gearapplication.html",
            {"gearapplication": None
            })

    def create_list_gearapplication_view(request,
gearapplicationId):
        logged_user = request.user
        application_user =
Applicationuser.objects.filter(username =
logged_user.username).first()
        gear_application = Gearapplication.objects.filter(id =
gearapplicationId).select_related("applicationuser").first()
        if gear_application.applicationuser ==
application_user:
            instances =
Gearapplicationinstance.objects.filter(gearapplication =
gear_application)
            campgear_list =
Campgearlist.objects.create(applicationuser = application_user,
            gear_list = "List " + str(gear_application.id))

```

```

        campgear_list.save()
        gearinstances = []
        for gearinstance in instances:
            gearlistinstance =
Campgearlistinstance.objects.create(campgearlist =
campgear_list,
            campgear = gearinstance.campgear,
            quantity = gearinstance.quantity)
        gearlistinstance.save()
        gearinstances.append(gearlistinstance)
    return render(request, "outequiprog/gearlist.html",
        {"gearlist": campgear_list,
        "gearinstances": gearinstances
        })

def create_campgear_list_view(request):
    logged_user = request.user
    application_user =
Applicationuser.objects.filter(username =
logged_user.username).first()
    if application_user:
        gear_list = self.request.GET.get('name', None)
        gears = self.request.GET.get('gears', None)
        gear_ids = gears.split('.')
        gear_id_list = []
        for gear_id in gear_ids:
            gear_id_list.append(int(gear_id))
        gear_quantities =
self.request.GET.get('quantities', None)
        gear_quantities = gear_quantities.split('.')
        gear_quantity_list = []
        for gear_quantity in gear_quantities:
            gear_quantity_list.append(int(gear_quantity))
        campgear_list =
Campgearlist.objects.create(applicationuser = application_user,
            gear_list = gear_list)
        campgear_list.save()
        pos = 0
        gearinstances = []
        while pos < len(gear_ids):
            campgear = Campgear.objects.create(id =
gear_id_list[pos])
            gearlistinstance =
Campgearlistinstance.objects.create(campgearlist =
campgear_list,
            campgear = campgear,
            quantity = gear_quantity_list[pos])
            gearlistinstance.save()

```

```

        pos = pos + 1
        gearinstances.append(gearlistinstance)
    return render(request, "outequiprog/gearlist.html",
        {"gearlist": campgear_list,
        "gearinstances": gearinstances
        })
else:
    return render(request, "outequiprog/gearlist.html",
        {"gearlist": None,
        "gearinstances": []
        })

def owner_gears_view(request):
    gear_owner_id = self.request.GET.get('ownerid', None)
    gear_owner_id = int(gear_owner_id)
    gear_owner =
Gearowner.objects.all().select_related('townpoint')
    gear_owner = gear_owner.get(id = gear_owner_id)
    campgears =
Campgear.objects.all().select_related('gearmodel',
'gearmodel__geartype', 'gearmodel__geartype__gearmenu')
    import datetime
    time_point = datetime.datetime.now()
    gearrental_begin_date =
self.request.GET.get('rental_begin', None)
    if gearrental_begin_date:
        gearrental_begin_date_parts =
gearrental_begin_date.split('.')
        gearrental_begin_date =
datetime.datetime(int(gearrental_begin_date_parts[0]),
                    int(gearrental_begin_date_parts[1]),
                    int(gearrental_begin_date_parts[2]))
    else:
        gearrental_begin_date = datetime.datetime.now()
    gearrental_end_date =
self.request.GET.get('rental_end', None)
    if gearrental_end_date:
        gearrental_end_date_parts =
gearrental_end_date.split('.')
        gearrental_end_date =
datetime.datetime(int(gearrental_end_date_parts[0]),
                    int(gearrental_end_date_parts[1]),
                    int(gearrental_end_date_parts[2]))
    else:
        gearrental_end_date = gearrental_begin_date +
datetime.timedelta(days = 1)
    rental_days = gearrental_end_date -
gearrental_begin_date

```

```

        rental_days = rental_days.days
        gear_result = []
        for campgear in campgears:
            availables =
Campgearinstanceavailable.objects.filter(campgearinstance__campg
ear = campgear,
            available_start__lte = gearrental_begin_date,
            available_end__gte = gearrental_end_date)
            if availables:
                instances_number =
Campgearinstance.objects.filter(campgear =
gearinstance.campgear, closed = False).count()
                camprent_date = gearrental_begin_date
                noavailable = False
                while camprent_date < gearrental_end_date:
                    applications =
Gearapplicationinstance.objects.filter(campgear = campgear,
                    gearapplication__rental_date__lte =
camprent_date,
                    gearapplication__rental_days__gte =
camprent_date,

gearapplication__application_state__in = [1,3])
                    rent_number = 0
                    for rent_application in applications:
                        rent_number = rent_number +
rent_application.quantity
                    if (instances_number - rent_number) < 1:
                        noavailable = True
                        break
                    camprent_date = camprent_date +
datetime.timedelta(days = 1)
                    if not noavailable:
                        gear_result.append(campgear)
                for campgear in gear_result:
                    pack = Gearprice.objects.filter(campgear =
gearinstance.campgear).first()
                    prices =
Typicalprice.objects.filter(typicalpricepack = pack,
min_days__lte = rental_days, max_days__gte =
rental_days).first()
                    campgear.deposit = prices.deposit
                    campgear.price = prices.day_price * rental_days
                return render(request, "outequprog/gearowner.html",
                    {"owner": gear_owner,
                     "gears": gear_result
                    })

```

```

def create_gearapplication_from_list_view(request):
    logged_user = request.user
    application_user =
Applicationuser.objects.filter(username =
logged_user.username).first()
    import datetime
    time_point = datetime.datetime.now()
    gearrental_begin_date =
self.request.GET.get('rental_begin', None)
    if gearrental_begin_date:
        gearrental_begin_date_parts =
gearrental_begin_date.split('.')
        gearrental_begin_date =
datetime.datetime(int(gearrental_begin_date_parts[0]),
                    int(gearrental_begin_date_parts[1]),
                    int(gearrental_begin_date_parts[2]))
    else:
        gearrental_begin_date = datetime.datetime.now()
    gearrental_end_date =
self.request.GET.get('rental_end', None)
    if gearrental_end_date:
        gearrental_end_date_parts =
gearrental_end_date.split('.')
        gearrental_end_date =
datetime.datetime(int(gearrental_end_date_parts[0]),
                    int(gearrental_end_date_parts[1]),
                    int(gearrental_end_date_parts[2]))
    else:
        gearrental_end_date = gearrental_begin_date +
datetime.timedelta(days = 1)
    rental_days = gearrental_end_date -
gearrental_begin_date
    rental_days = rental_days.days
    rental_list_id = self.request.GET.get('list_id', None)
    rental_list_id = int(rental_list_id)
    gear_list = Campgearlist.objects.filter(id =
rental_list_id)
    if gear_list.applicationuser == application_user:
        instances =
Campgearlistinstance.objects.filter(campgearlist =
gear_list).select_related('campgear', 'campgear__gearowner')
        owner = None
        notpossible = False
        for gear_instance in instances:
            if not owner:
                owner = gear_instance.campgear.gearowner
            elif owner != gear_instance.campgear.gearowner:
                notpossible = True

```

```

        if not notpossible:
            for gear_instance in instances:
                availables =
Campgearinstanceavailable.objects.filter(campgearinstance__campg
ear = gear_instance.campgear,
                available_start__lte =
gearrental_begin_date,
                available_end__gte =
gearrental_end_date)
                if not availables:
                    gear_instance.available = False
                else:
                    instances_number =
Campgearinstance.objects.filter(campgear =
gear_instance.campgear, closed = False).count()
                    camprent_date = gearrental_begin_date
                    noavailable = False
                    while camprent_date <
gearrental_end_date:
                        applications =
Gearapplicationinstance.objects.filter(campgear =
gear_instance.campgear,
                        gearapplication__rental_date__lte = camprent_date,
                        gearapplication__rental_days__gte = camprent_date,
                        gearapplication__application_state__in = [1,3])
                        rent_number = 0
                        for rent_application in
applications:
                            rent_number = rent_number +
rent_application.quantity
                        if (instances_number - rent_number)
< gear_instance.quantity:
                            gear_instance.available = False
                            noavailable = True
                            break
                            camprent_date = camprent_date +
datetime.timedelta(days = 1)
                        if not noavailable:
                            gear_instance.available = True
                    available_number = 0
                    sum_price = 0
                    sum_deposit = 0
                    for gearinstance in instances:
                        if gearinstance.available:
                            camprent_date = gearrental_begin_date

```

```

        pack =
Gearprice.objects.filter(campgear =
gearinstance.campgear).first()
        prices =
Typicalprice.objects.filter(typicalpricepack = pack,
min_days__lte = rental_days, max_days__gte =
rental_days).first()
        gearinstance.deposit = prices.deposit *
gearinstance.quantity
        gearinstance.price = prices.day_price *
rental_days * gearinstance.quantity
        available_number = available_number + 1
        sum_price = sum_price +
gearinstance.price
        sum_deposit = sum_deposit +
gearinstance.deposit
        if available_number > 0:
            gear_application =
Gearapplication.objects.create(applicationuser =
application_user,
                                gearowner = owner,
                                apply_time = datetime.datetime.now(),
                                rental_date = gearrental_begin_date,
                                rental_days = gearrental_end_date,
                                application_state = 0,
                                total_price = sum_price,
                                deposit_price = sum_deposit))
            gear_application.save()
            gear_appl_instances = []
            for gearinstance in instances:
                if gearinstance.available:
                    gear_application_instance =
Gearapplicationinstance.objects.create(gearapplication =
gear_application,
                                        campgear =
gearinstance.campgear,
                                        quantity =
gearinstance.quantity)
                    gear_application_instance.save()

            gear_appl_instances.append(gear_application_instance)
            gear_application.instances =
gear_appl_instances
        return render(request,
"outequiprog/gearapplication.html",
                        {"gearapplication": gear_application
                        })

```

```

        return render(request,
            "outequprog/gearapplication.html",
            {"gearapplication": None
            })

    def create_gearapplication_view(request):
        logged_user = request.user
        application_user =
Applicationuser.objects.filter(username =
logged_user.username).first()
        import datetime
        time_point = datetime.datetime.now()
        gearrental_begin_date =
self.request.GET.get('rental_begin', None)
        if gearrental_begin_date:
            gearrental_begin_date_parts =
gearrental_begin_date.split('.')
            gearrental_begin_date =
datetime.datetime(int(gearrental_begin_date_parts[0]),
                    int(gearrental_begin_date_parts[1]),
                    int(gearrental_begin_date_parts[2]))
        else:
            gearrental_begin_date = datetime.datetime.now()
        gearrental_end_date =
self.request.GET.get('rental_end', None)
        if gearrental_end_date:
            gearrental_end_date_parts =
gearrental_end_date.split('.')
            gearrental_end_date =
datetime.datetime(int(gearrental_end_date_parts[0]),
                    int(gearrental_end_date_parts[1]),
                    int(gearrental_end_date_parts[2]))
        else:
            gearrental_end_date = gearrental_begin_date +
datetime.timedelta(days = 1)
        rental_days = gearrental_end_date -
gearrental_begin_date
        rental_days = rental_days.days
        if application_user:
            gears = self.request.GET.get('gears', None)
            gear_ids = gears.split('.')
            gear_id_list = []
            for gear_id in gear_ids:
                gear_id_list.append(int(gear_id))
            gear_quantities =
self.request.GET.get('quantities', None)
            gear_quantities = gear_quantities.split('.')
            gear_quantity_list = []

```



```

                                gear_application_instance =
Gearapplicationinstance.objects.create(gearapplication =
gear_application,
                                campgear = gearinstance,
                                quantity =
gear_quantity_list[pos])
                                gear_application_instance.save()

gear_appl_instances.append(gear_application_instance)
                                gear_application.instances =
gear_appl_instances
                                return render(request,
"outequiprog/gearapplication.html",
                                {"gearapplication": gear_application
                                })
                                return render(request,
"outequiprog/gearapplication.html",
                                {"gearapplication": None
                                })

def campgear_demo_view(request):
    campgear_id = self.request.GET.get('campgear', None)
    if not campgear_id:
        render(request, "outequiprog/campgear.html",
            {"campgear": None
            })
    else:
        campgear_id = int(campgear_id)
        campgear = Campgear.objects.get(id =
campgear_id).select_related("gearowner", "gearmodel",
"gearmodel__geartype", "gearmodel__geartype__gearmenu")
        application_user =
Applicationuser.objects.filter(username =
logged_user.username).first()
        if application_user:
            campgear.lists =
Campgearlistinstance.objects.filter(campgear = campgear,
campgearlist__applicationuser =
application_user).select_related("campgearlist")
            campgear.possible_lists =
Campgearlist.objects.filter(applicationuser = application_user)
            import datetime
            time_point = datetime.datetime.now()
            gearrental_begin_date =
self.request.GET.get('rental_begin', None)
            if gearrental_begin_date:
                gearrental_begin_date_parts =
gearrental_begin_date.split('.')

```

```

        gearrental_begin_date =
datetime.datetime(int(gearrental_begin_date_parts[0]),
                    int(gearrental_begin_date_parts[1]),
                    int(gearrental_begin_date_parts[2]))
    else:
        gearrental_begin_date = datetime.datetime.now()
        gearrental_end_date =
self.request.GET.get('rental_end', None)
        if gearrental_end_date:
            gearrental_end_date_parts =
gearrental_end_date.split('.')
            gearrental_end_date =
datetime.datetime(int(gearrental_end_date_parts[0]),
                    int(gearrental_end_date_parts[1]),
                    int(gearrental_end_date_parts[2]))
        else:
            gearrental_end_date = gearrental_begin_date +
datetime.timedelta(days = 1)
            rental_days = gearrental_end_date -
gearrental_begin_date
            rental_days = rental_days.days
            availables =
Campgearinstanceavailable.objects.filter(campgearinstance__campg
ear = campgear,
            available_start__lte = gearrental_begin_date,
            available_end__gte = gearrental_end_date)
            if not availables:
                campgear.available = False
            else:
                instances_number =
Campgearinstance.objects.filter(campgear = campgear, closed =
False).count()
                camprent_date = gearrental_begin_date
                noavailable = False
                while camprent_date < gearrental_end_date:
                    applications =
Gearapplicationinstance.objects.filter(campgear = campgear,
                    gearapplication__rental_date__lte =
camprent_date,
                    gearapplication__rental_days__gte =
camprent_date,
                    gearapplication__application_state__in =
[1,3])
                    rent_number = 0
                    for rent_application in applications:
                        rent_number = rent_number +
rent_application.quantity
                    if (instances_number - rent_number) < 1:

```

```

        campgear.available = False
        noavailable = True
        break
    camprent_date = camprent_date +
datetime.timedelta(days = 1)
    if not noavailable:
        campgear.available = True
    if campgear.available:
        camprent_date = gearrental_begin_date
        pack = Gearprice.objects.filter(campgear =
campgear).first()
        prices =
Typicalprice.objects.filter(typicalpricepack = pack,
min_days__lte = rental_days, max_days__gte =
rental_days).first()
        campgear.deposit = prices.deposit
        campgear.price = prices.day_price * rental_days
    render(request, "outequprog/campgear.html",
        {"campgear": campgear
        })

def cancel_personal_gearapplication_view(request,
gearapplicationId):
    logged_user = request.user
    application_user =
Applicationuser.objects.filter(username =
logged_user.username).first()
    gear_application = Gearapplication.objects.filter(id =
gearapplicationId).select_related("applicationuser").first()
    import datetime
    time_point = datetime.datetime.now()
    rental_days = gear_application.rental_date - time_point
    rental_days = rental_days.days
    if gear_application.applicationuser == application_user
and (gear_application.application_state == 0 or
(gear_application.application_state == 1 and rental_days >=
10)):
        gear_application.application_state = 2
        gear_application.save()
        instances =
Gearapplicationinstance.objects.filter(gearapplication =
gear_application)
        gear_application.instances = instances
    return render(request,
"outequprog/gearapplication.html",
        {"gearapplication": gear_application,
        "result": "успішно"
        })

```

```

        else:
            return render(request,
                "outequprog/gearapplication.html",
                {"gearapplication": None,
                 "result": "проблема"
                })

    def delete_gearlist_view(request, gearlistId):
        logged_user = request.user
        application_user =
        Applicationuser.objects.filter(username =
        logged_user.username).first()
        gear_application = Campgearlist.objects.filter(id =
        gearlistId).select_related("applicationuser").first()
        if gear_application.applicationuser ==
        application_user:
            gear_application.delete()
            campgear_lists =
            Campgearlist.objects.filter(applicationuser = application_user)
            campgear_lists =
            campgear_lists.order_by("gear_list")
            return render(request, "outequprog/gearlists.html",
                {"gearlists": campgear_lists })
        else:
            return render(request, "outequprog/gearlists.html",
                {"gearlists": [] })

    def delete_gear_from_gearlist_view(request,
        campgearlistId):
        logged_user = request.user
        application_user =
        Applicationuser.objects.filter(username =
        logged_user.username).first()
        gear_instance = Campgearlistinstance.objects.filter(id
        = campgearlistId).select_related("campgearlist",
        "campgearlist__applicationuser").first()
        if gear_instance.campgearlist.applicationuser ==
        application_user:
            gear_instance.delete()
            gearrental_begin_date = datetime.datetime.now()
            gearrental_end_date = gearrental_begin_date +
            datetime.timedelta(days = 1)
            rental_days = 1
            gearinstances = []
            gearinstances =
            Campgearlistinstance.objects.filter(campgearlist =
            gear_instance.campgearlist).select_related("campgear")
            gearinstances = gearinstances.order_by("id")

```

```

        for gearinstance in gearinstances:
            availables =
Campgearinstanceavailable.objects.filter(campgearinstance__campg
ear = gearinstance.campgear,
            available_start__lte =
gearrental_begin_date,
            available_end__gte = gearrental_end_date)
            if not availables:
                gearinstance.available = False
            else:
                instances_number =
Campgearinstance.objects.filter(campgear =
gearinstance.campgear, closed = False).count()
                camprent_date = gearrental_begin_date
                noavailable = False
                while camprent_date < gearrental_end_date:
                    applications =
Gearapplicationinstance.objects.filter(campgear =
gearinstance.campgear,
                    gearapplication__rental_date__lte =
camprent_date,
                    gearapplication__rental_days__gte =
camprent_date,
                    gearapplication__application_state__in = [1,3])
                    rent_number = 0
                    for rent_application in applications:
                        rent_number = rent_number +
rent_application.quantity
                    if (instances_number - rent_number) <
gearinstance.quantity:
                        gearinstance.available = False
                        noavailable = True
                        break
                    camprent_date = camprent_date +
datetime.timedelta(days = 1)
                    if not noavailable:
                        gearinstance.available = True
                for gearinstance in gearinstances:
                    if gearinstance.available:
                        camprent_date = gearrental_begin_date
                        pack = Gearprice.objects.filter(campgear =
gearinstance.campgear).first()
                        prices =
Typicalprice.objects.filter(typicalpricepack = pack,
min_days__lte = rental_days, max_days__gte =
rental_days).first()
                        gearinstance.deposit = prices.deposit

```

```

        gearinstance.price = prices.day_price *
rental_days
        return render(request, "outequiprog/gearlist.html",
            {"gearlist": gear_instance.campgearlist,
             "gearinstances": gearinstances
            })
        return render(request, "outequiprog/gearlist.html",
            {"gearlist": None,
             "gearinstances": []
            })

    def change_gear_from_gearlist_view(request, campgearlistId,
gearquantity):
        logged_user = request.user
        application_user =
Applicationuser.objects.filter(username =
logged_user.username).first()
        gear_instance = Campgearlistinstance.objects.filter(id
= campgearlistId).select_related("campgearlist",
"campgearlist__applicationuser").first()
        if gear_instance.campgearlist.applicationuser ==
application_user:
            gear_instance.quantity = gearquantity
            gear_instance.save()
            gearrental_begin_date = datetime.datetime.now()
            gearrental_end_date = gearrental_begin_date +
datetime.timedelta(days = 1)
            rental_days = 1
            gearinstances = []
            gearinstances =
Campgearlistinstance.objects.filter(campgearlist =
gear_instance.campgearlist).select_related("campgear")
            gearinstances = gearinstances.order_by("id")
            for gearinstance in gearinstances:
                availables =
Campgearinstanceavailable.objects.filter(campgearinstance__campg
ear = gearinstance.campgear,
                available_start__lte =
gearrental_begin_date,
                available_end__gte = gearrental_end_date)
            if not availables:
                gearinstance.available = False
            else:
                instances_number =
Campgearinstance.objects.filter(campgear =
gearinstance.campgear, closed = False).count()
                camprent_date = gearrental_begin_date
                noavailable = False

```

```

        while camprent_date < gearrental_end_date:
            applications =
Gearapplicationinstance.objects.filter(campgear =
gearinstance.campgear,
                                     gearapplication__rental_date__lte =
camprent_date,
                                     gearapplication__rental_days__gte =
camprent_date,

gearapplication__application_state__in = [1,3])
            rent_number = 0
            for rent_application in applications:
                rent_number = rent_number +
rent_application.quantity
            if (instances_number - rent_number) <
gearinstance.quantity:
                gearinstance.available = False
                noavailable = True
                break
            camprent_date = camprent_date +
datetime.timedelta(days = 1)
            if not noavailable:
                gearinstance.available = True
            for gearinstance in gearinstances:
                if gearinstance.available:
                    camprent_date = gearrental_begin_date
                    pack = Gearprice.objects.filter(campgear =
gearinstance.campgear).first()
                    prices =
Typicalprice.objects.filter(typicalpricepack = pack,
min_days__lte = rental_days, max_days__gte =
rental_days).first()
                    gearinstance.deposit = prices.deposit
                    gearinstance.price = prices.day_price *
rental_days
            return render(request, "outequiprog/gearlist.html",
                {"gearlist": gear_instance.campgearlist,
                 "gearinstances": gearinstances
                })
            return render(request, "outequiprog/gearlist.html",
                {"gearlist": None,
                 "gearinstances": []
                })

def add_gear_to_gearlist_view(request, campgearlistId,
gearId, gearquantity = 1):
    logged_user = request.user

```

```

        application_user =
Applicationuser.objects.filter(username =
logged_user.username).first()
        campgear = Campgear.objects.get(id = gearId)
        gear_list = Campgearlist.objects.filter(id =
campgearlistId).select_related("applicationuser")
        if gear_list and username and gear_list.applicationuser
== application_user:
            gear_list_instance =
Campgearlistinstance.objects.create(campgearlist = gear_list,
                                     campgear = campgear,
                                     quantity = gearquantity)
            gear_list_instance.save()
            gearrental_begin_date = datetime.datetime.now()
            gearrental_end_date = gearrental_begin_date +
datetime.timedelta(days = 1)
            rental_days = 1
            gearinstances = []
            gearinstances =
Campgearlistinstance.objects.filter(campgearlist =
gear_instance.campgearlist).select_related("campgear")
            gearinstances = gearinstances.order_by("id")
            for gearinstance in gearinstances:
                availables =
Campgearinstanceavailable.objects.filter(campgearinstance__campg
ear = gearinstance.campgear,
                                           available_start__lte =
gearrental_begin_date,
                                           available_end__gte = gearrental_end_date)
                if not availables:
                    gearinstance.available = False
                else:
                    instances_number =
Campgearinstance.objects.filter(campgear =
gearinstance.campgear, closed = False).count()
                    camprent_date = gearrental_begin_date
                    noavailable = False
                    while camprent_date < gearrental_end_date:
                        applications =
Gearapplicationinstance.objects.filter(campgear =
gearinstance.campgear,
                                         gearapplication__rental_date__lte =
camprent_date,
                                         gearapplication__rental_days__gte =
camprent_date,
                                         gearapplication__application_state__in = [1,3])
                        rent_number = 0

```

```

        for rent_application in applications:
            rent_number = rent_number +
rent_application.quantity
            if (instances_number - rent_number) <
gearinstance.quantity:
                gearinstance.available = False
                noavailable = True
                break
                camprent_date = camprent_date +
datetime.timedelta(days = 1)
                if not noavailable:
                    gearinstance.available = True
            for gearinstance in gearinstances:
                if gearinstance.available:
                    camprent_date = gearrental_begin_date
                    pack = Gearprice.objects.filter(campgear =
gearinstance.campgear).first()
                    prices =
Typicalprice.objects.filter(typicalpricepack = pack,
min_days__lte = rental_days, max_days__gte =
rental_days).first()
                    gearinstance.deposit = prices.deposit
                    gearinstance.price = prices.day_price *
rental_days
            return render(request, "outequiprog/gearlist.html",
                {"gearlist": gear_instance.campgearlist,
                 "gearinstances": gearinstances
                })
    return render(request, "outequiprog/gearlist.html",
        {"gearlist": None,
         "gearinstances": []
        })

```

ДОДАТОК В
Слайди презентації

Міністерство освіти і науки України
Національний університет «Запорізька політехніка»
кафедра програмних засобів

Тема дипломної кваліфікаційної роботи бакалавра
Програмне забезпечення прокату
туристичного спорядження
Software for Outdoor Equipment Rental

Виконавець:

Володимир ЩЕРБАНЬ

Керівник: д.т.н., професор

Сергій СУББОТІН

2024

Рисунок В.1 – Початковий слайд

Об'єкт, предмет, мета роботи

Об'єкт роботи – процес прокату туристичного спорядження.

Предмет роботи – програмне забезпечення прокату туристичного спорядження.

Мета роботи – розробити програмне забезпечення прокату туристичного спорядження для підтримки індивідуальних пунктів прокатів та індивідуальних власників спорядження.

Рисунок В.2 – Об'єкт, предмет, мета роботи

Завдання роботи

- аналіз проблеми розробки програмного забезпечення прокату туристичного спорядження;
- визначення вимог до програмного забезпечення прокату туристичного спорядження;
- проектування програмного забезпечення прокату туристичного спорядження;
- реалізація програмного забезпечення прокату туристичного спорядження.

3

Рисунок В.3 – Завдання роботи

Порівняння програмного забезпечення прокату туристичного спорядження з проаналізованими аналогами

Характеристика	RentOutdoorGear	Робінзон	Outdoor Equipment Rentals	Розробка
Реєстрація	+	+	+	+
Замовлення (бро-нування) прокату туристичного спорядження	+	–	+	+
Джерело послуг	Пункт прокату	Магазин	Пункт прокату	Пункти прокату, власники
Створення типового замовлення	–	–	–	+

4

Рисунок В.4 – Порівняння програмного забезпечення прокату туристичного спорядження з проаналізованими аналогами

Порівняння фреймворків для створення програмного забезпечення прокату туристичного спорядження

Показник	Django	Flask
Вбудовані засоби керування даними бази даних через спеціальний інтерфейс	Адміністративна панель	Відсутні (необхідно окремо встановлювати)
Вбудовані засоби підтримки захищеності ПЗ	Широке коло	Відсутні (необхідно окремо встановлювати)
Сприяння швидкості розробки	Більше	Менше
Робота з базами даних	Вбудоване OPB підтримує системи SQLite, PostgreSQL, MariaDB, Oracle	Не обмежена, потребує встановлення бібліотеки для роботи з базою даних та SQLAlchemy для об'єктно-реляційного відображення

Рисунок В.5 – Порівняння фреймворків для створення програмного забезпечення прокату туристичного спорядження

Діаграма прецедентів винаймача туристичного спорядження



Рисунок В.6 – Діаграма прецедентів винаймача туристичного спорядження

Алгоритм пошуку туристичного спорядження, необхідного для прокату



Рисунок В.9 – Алгоритм пошуку туристичного спорядження, необхідного для прокату

Інтерфейси програмного забезпечення прокату туристичного спорядження

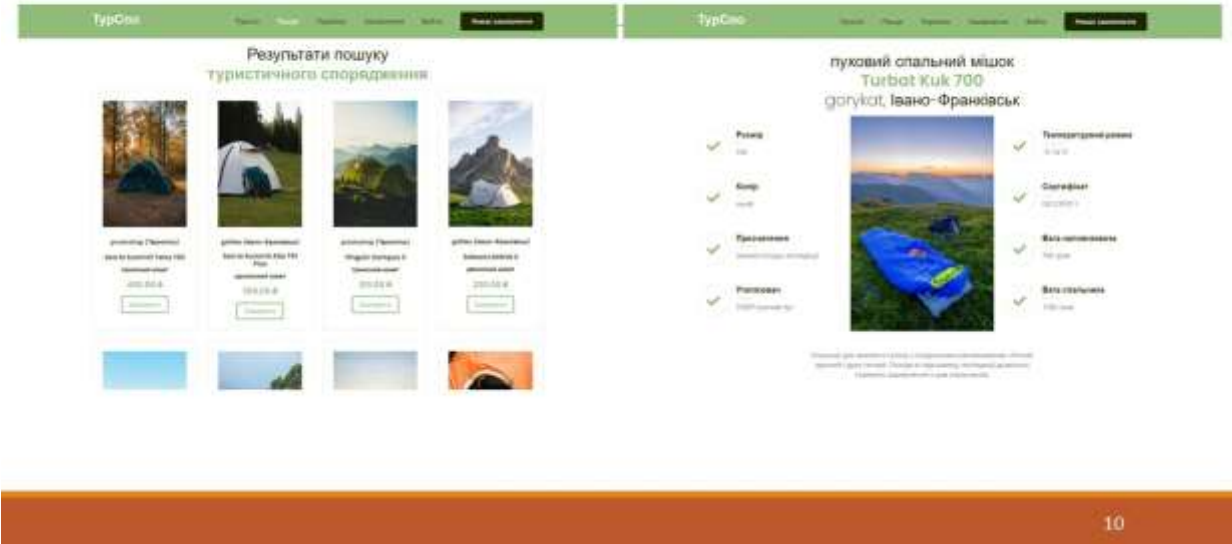


Рисунок В.10 – Інтерфейси програмного забезпечення прокату туристичного спорядження

Інтерфейси програмного забезпечення прокату туристичного спорядження

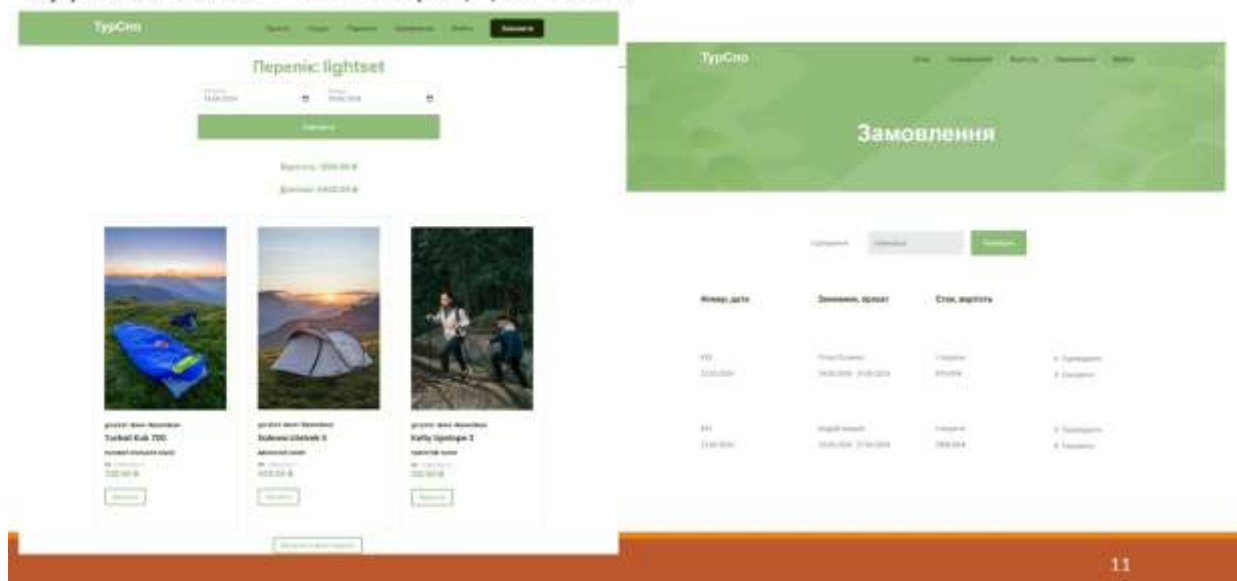


Рисунок В.11 – Інтерфейси програмного забезпечення прокату туристичного спорядження

Висновки

У роботі розроблено вебдодаток прокату туристичного спорядження за допомогою мови Python та фреймворку Django, який порівняно з аналогами має особливості, що включають об'єднання можливостей прокату від конкретних пунктів і від власників туристичного спорядження та можливість створення типового замовлення прокату туристичного спорядження на основі визначено шаблону, що повністю відповідає меті дипломної кваліфікаційної роботи та дозволило досягти її.

Програма розроблена як результат прийняття рішень стосовно визначення основних функцій аналогів, що забезпечують прокат туристичного спорядження, формування відмінностей розробки, вибору засобів, виокремлення сценаріїв використання власниками прокатів, менеджерами системи прокату, винаймачами туристичного спорядження, їхньої програмної реалізації.

Рисунок В.12 – Висновки