

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 КОМПЛЕКСНОЇ
(назва теми)
ПІДТРИМКИ ПРОДАЖУ СТАРИХ МЕБЛІВ. SOFTWARE FOR
COMPREHENSIVE SUPPORT OF OLD FURNITURE SALE

Виконав(ла): студент(ка) 4 курсу, групи КНТ-229

Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Комп'ютерні науки

УШАКОВ Д.В.

(ПРІЗВИЩЕ та ініціали)

Керівник ЛЬОВКІН В.М.

(ПРІЗВИЩЕ та ініціали)

Рецензент ТІМЕНКО А.В.

(ПРІЗВИЩЕ та ініціали)

2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет ФКНТ
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
« » 2023 року

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

УШАКОВА Давида Вадимовича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення комплексної підтримки
продажу старих меблів. Software for Comprehensive Support of Old Furniture Sale
керівник проєкту (роботи) к.т.н., доцент ЛЬОВКІН Валерій Миколайович,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від «04» квітня 2023 року № 87

2. Строк подання студентом проєкту (роботи) 01 червня 2023 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	ЛЬОВКІН В.М., доцент		
Нормоконтроль	БЄЛОВА А.В., асистент		

7. Дата видачі завдання « 4 » квітня 2023 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Срок 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)

_____ Давид УШАКОВ
(підпис) (Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

_____ Валерій ЛЬОВКІН
(підпис) (Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 98 с., 32 рис., 3 табл., 3 дод., 12 джерел.

СТАРІ МЕБЛІ, ПРОДАЖ, РЕСТАВРАЦІЯ, МАЙСТЕР, ДОСТАВКА, ВЕБДОДАТОК.

Об'єкт роботи – процес продажу старих меблів. Предмет роботи – програмне забезпечення продажу старих меблів. Мета роботи – розробити програмне забезпечення комплексної підтримки продажу старих меблів для підвищення зручності придбання меблів, які були у використанні, через інтеграцію з реставрацією.

Матеріали, методи та технічні засоби: мова Python, фреймворки Django і Bootstrap, мови HTML, CSS, JavaScript, технологія AJAX, система керування базами даних PostgreSQL.

Результати. У результаті розроблено вебдодаток, готовий для використання під час продажу і придбання старих меблів.

Висновки. Розроблена програма дозволяє інтегрувати необхідні допоміжні складові для виконання реставрації меблів до процесу придбання старих меблів, що в підсумку сприяє збільшенню ринку реставраційних послуг, допомагаючи майстрам, та збільшенню зручності покупців, які можуть зменшити вартість придбання меблів за рахунок меблів, які були у вжитку, але при цьому мати можливість оцінити і замовити всі необхідні роботи, які дозволять після доставки одразу використовувати меблі.

Галузь використання. Програма може використовуватися як майданчик для комунікації людей, які мають у власності старі меблі і готові їх продати, людей, які шукають такі меблі і готові їх придбати, майстрів, які готові виконувати реставраційні роботи, та перевізників, які готові їх з'єднувати між собою.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 98 pages, 32 figures, 3 tables, 3 appendixes, 12 sources.

OLD FURNITURE, SALE, RESTORATION, CRAFTSMAN, DELIVERY, WEB APPLICATION.

The object of the work is the process of selling old furniture. The subject of the work is software for the sale of old furniture. The goal of the work is to develop software for comprehensive support of the old furniture sale to increase the convenience of purchasing used furniture through integration with restoration.

Materials, methods and technical means: Python language, Django and Bootstrap frameworks, HTML, CSS, JavaScript languages, AJAX technology, PostgreSQL database management system.

Results. As a result, the web application, ready for use during the sale and purchase of old furniture, was developed.

Conclusions. The developed software allows to integrate the necessary components for the restoration of furniture into the process of purchasing old furniture, which ultimately contributes to the increase in the market for restoration services, helping craftsmen. It increases convenience of buyers, who can reduce costs of purchasing furniture buying used furniture, but at the same time they are able to estimate and order all the necessary jobs, making possible to use the furniture immediately after delivery.

Field of use. The software can be used as a platform for communication between people who own old furniture and are ready to sell it, people who are looking for such furniture and are ready to buy it, craftsmen who are ready to perform restoration jobs, and transporters who are ready to connect them together.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	7
Вступ.....	8
1 Аналіз предметної області.....	9
1.1 Огляд програмного забезпечення продажу старих меблів	9
1.2 Порівняння аналогів	11
1.3 Висновки за розділом 1	13
2 Проєктування програми.....	14
2.1 Вибір засобів проєктування і розробки	14
2.2 Визначення функціональних вимог до програмного забезпечення	15
2.3 Проєктування структури бази даних.....	22
2.4 Висновки за розділом 2	38
3 Розробка програми	39
3.1 Порядок функціонування програми.....	39
3.2 Рішення з реалізації одиниць додатку	46
3.3 Висновки за розділом 3	50
4 Експлуатація та тестування програми	51
4.1 Опис програми.....	51
4.2 Експлуатація програми.....	51
4.3 Тестування програми.....	58
4.4 Висновки за розділом 4	61
Висновки	62
Перелік джерел посилення	63
Додаток А Технічне завдання	65
Додаток Б Текст програми	71
Додаток В Слайди презентації.....	91

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

БД – база даних;

ПЗ – програмне забезпечення.

ВСТУП

В умовах негативних економічних процесів одним з вирішень побутових проблем є придбання і використання меблів, які вже були у використанні раніше. Це є взаємовигідним процесом, адже є люди, які готові позбутися своїх меблів, а є люди, які готові придбати такі меблі, щоб зменшити власні витрати. Варто відзначити, що за оцінками в роботах [1]-[3] старі меблі коштують зазвичай більше ніж на 50 % дешевше ніж нові, що є суттєвою різницею. Таким чином люди, які не мають достатньо коштів, можуть отримати меблі, якими зможуть користуватися, не очікуючи той момент, коли накопичиться потрібна сума.

У 2021 році ринок старих меблів оцінювався у 29,8 мільярдів американських доларів, але прогнозується, що до 2030 року він досягне 58,58 мільярдів [1]-[3]. Тобто ринок має розширитися в 2 рази. Це є значним приростом, а тому створення нових рішень для такого ринку є перспективним напрямком. Існують багато причин, з яких люди продають меблі, які знаходяться у їх власності, а відповідно це створює можливості для тих, хто готовий ці меблі придбати.

Звичайно, що дошки оголошень є популярним засобом вирішення і цієї проблеми, однак дана проблема є складнішою за суттю: часто недостатньо знайти потрібний варіант і встановити його у себе вдома, а потрібні певні ремонтні або загалом реставраційні процедури. Потрібно знайти майстра, відремонтувати меблі, перевезти їх. Тож у роботі потрібно розглянути комплексний підхід до продажу і придбання старих меблів на основі розробки відповідного програмного забезпечення (ПЗ).

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд програмного забезпечення продажу старих меблів

За результатами попереднього аналізу було виділено ряд додатків, які дозволяють придбати старі меблі. Ті, які мають найбільш придатні інструменти, в підсумку були винесені до представленого огляду.

Kaiyo [4] (рис. 1.1) є одним з найбільш відомих вебдодатків для продажу старих меблів. Окрім того, що він забезпечує можливість продати і придбати старі меблі, він характеризується тим, що в основі знаходиться компанія, яка підтримує продаж. Вона сама виконує підготовку меблів перед доставкою до покупця. Тобто вона виконує чистку меблів. Але це не означає, що вона виконає повноцінний ремонт чи реставрацію меблів.

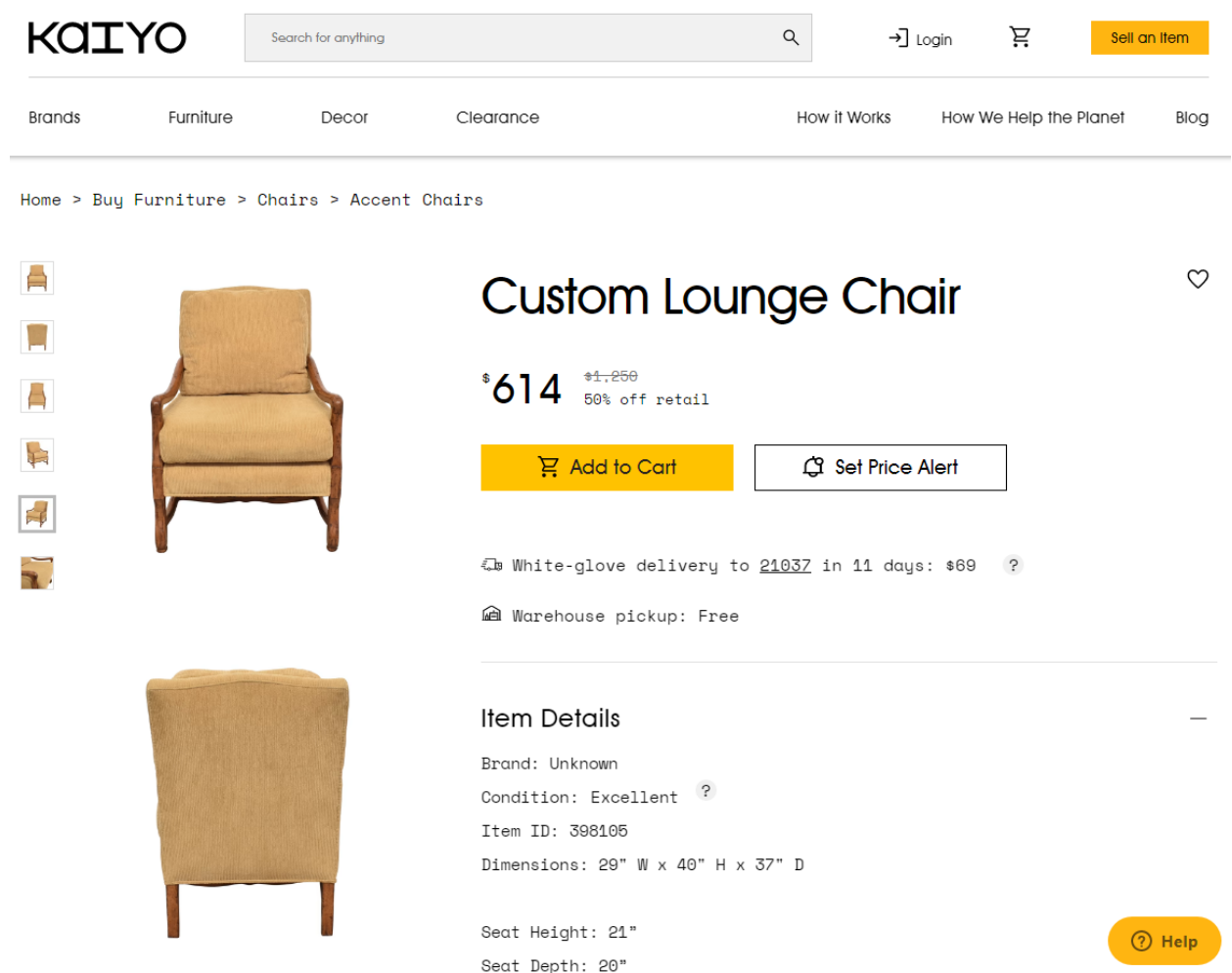


Рисунок 1.1 – Вебдодаток Kaiyo

Для усіх меблів визначається їх загальний опис, фотографії, а також приводиться загальний стан меблів (фактично категорія, що характеризує стан). Придбати меблі можна тільки за встановленою ціною.

Etsy [5] (рис. 1.2) дозволяє не тільки придбати старі меблі, але і підтримує продаж речей, які були зроблені власноруч. Стосовно продажу старих меблів визначається стиль, матеріали, період виробництва. На кожній сторінці також приводяться відгуки про продавця. Користувачі можуть зберегти для повторного перегляду меблі.

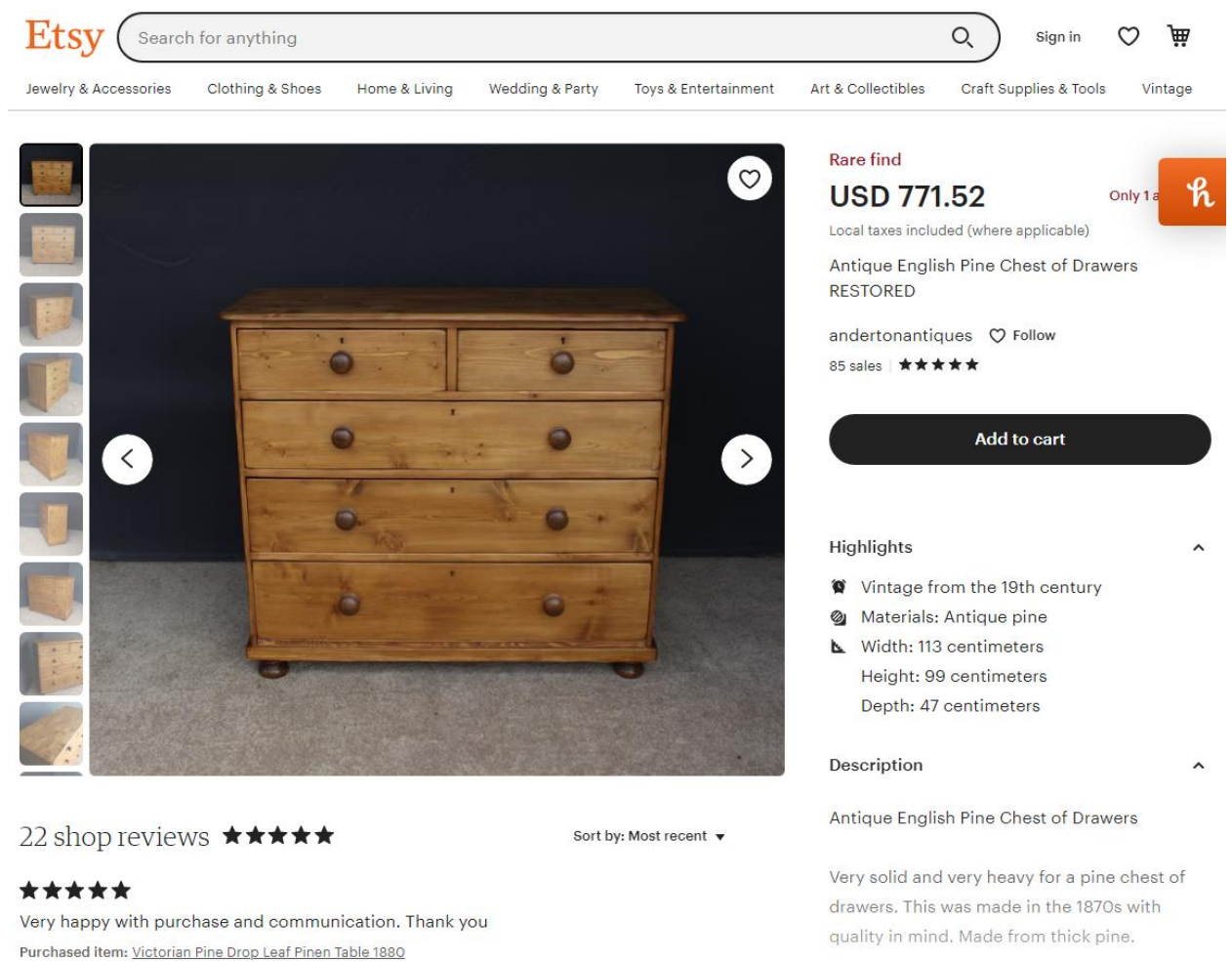


Рисунок 1.2 – Вебдодаток Etsy

Clairish [6] (рис. 1.3) також дозволяє придбати старі меблі. При цьому деталізація є значно більшою, враховуючи особливості антикваріату. Тому за кожним екземпляром приводиться період виробництва, тип екземпляру,

категорія стану і опис стану, стилі, країна виробництва, матеріали, з яких вироблено екземпляр, колір. Покупець може не тільки придбати меблі за вказаними умовами, але має можливість сформулювати і власну пропозицію.

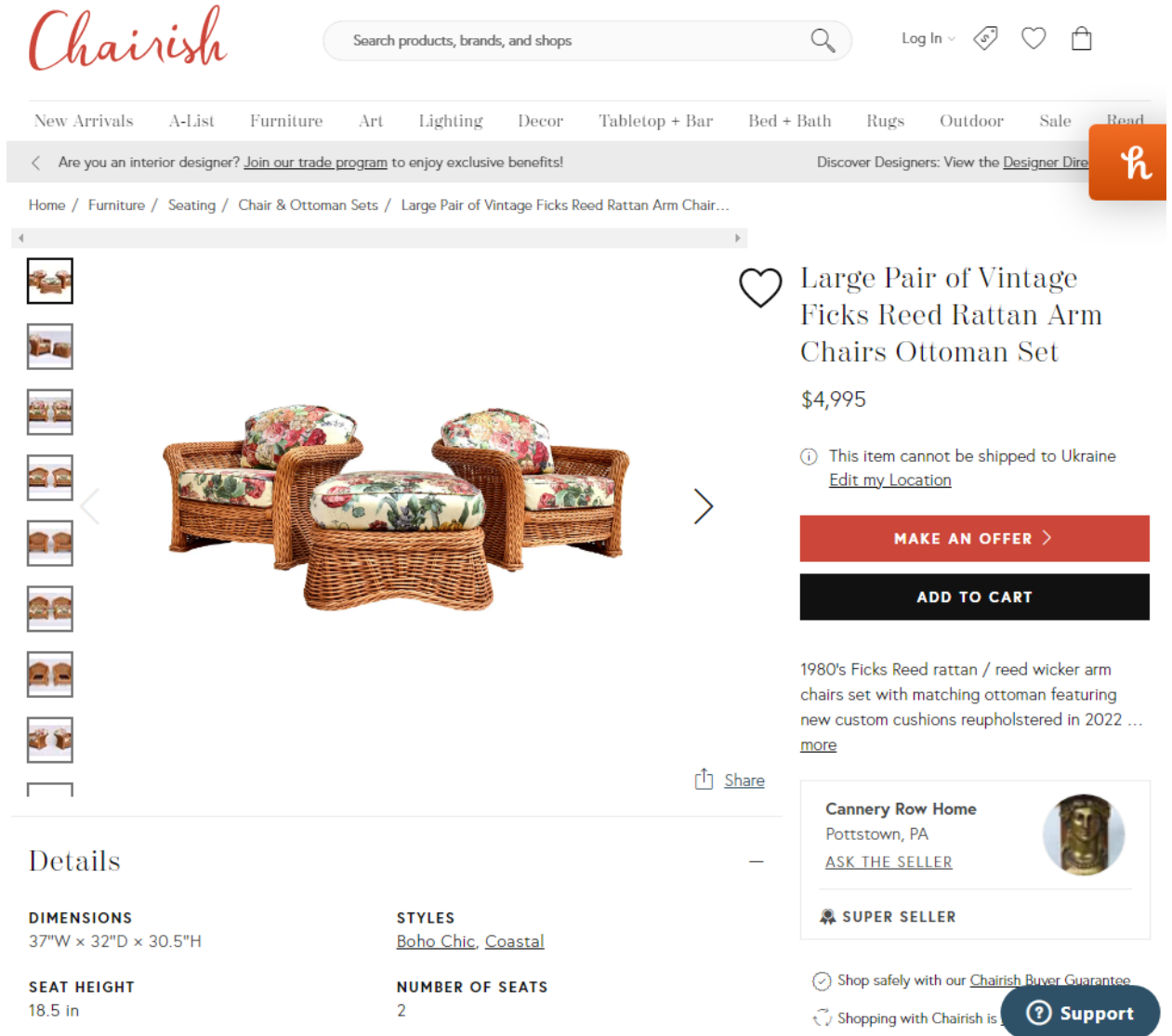


Рисунок 1.3 – Вебдодаток Clairish

1.2 Порівняння аналогів

У результаті огляду було вироблено концепцію власного ПЗ і виконано порівняння його з додатками [7], які було проаналізовано в огляді (табл. 1.1).

Таблиця 1.1 – Порівняння програмних додатків, придатних для продажу старих меблів

Характеристика	Kaiyo	Etsy	Clairish	Розробка
Продаж і покупка старих меблів	Так	Так	Так	Так
Придбання старих меблів від продавців	Ні	Так	Так	Так
Оцінювання продавців	Ні	Так	Так	Так
Пропонування ціни покупцем	Ні	Ні	Так	Так
Реновація старих меблів майстром	Ні	Ні	Ні	Так

У виконуваний розробці вирішено об'єднати основні характеристики існуючих програм та розширити і посилити їх за рахунок концепції комплексної підтримки.

Kaiyo дозволяє продавати і купляти старі меблі, але цей продаж відбувається через цю компанію, відповідно прямого контакту між продавцем і покупцем немає, а компанія виступає в ролі посередника. Меблі компанія чистить, але не виконує їх ремонту, тому купити меблі для деталей, купити меблі для реставрації, виконати реставрацію за допомогою даної програми не вдасться.

Etsy дозволяє контактувати з продавцями, оцінювати їх, але не можна запропонувати власну ціну, не можна знову ж таки виконати реставрацію. У підсумку продавець виставляє власну ціну, чекає придбання, а якщо розуміє, що ціна неадекватна, то вимушений змінювати ціну і чекати далі реакції, що може затримувати процес продажу. Clairish усуває цю проблему. Покупець може запропонувати власну вартість. Відповідно отримавши всі пропозиції, продавець може вирішити, що робити далі: чи варто продати меблі за зниженою ціною, чи почекати ще. Це сприяє більшій гнучкості продавця. Цю

ж ідею вирішено використати у розробці в даній роботі. Відповідно звичайно, що продаж відбувається самими продавцями без посередників. Однак найголовнішою особливістю розробки є концепція комплексної підтримки. На платформі з'являються майстри. Майстри не тільки виконують ремонт чи реставрацію, але і допомагають оцінити необхідні роботи, тобто в такому випадку виконують підтримку покупців, дозволяючи їм вирішити, чи варто взагалі купляти такі старі меблі або як краще відремонтувати меблі, які збирається купити покупець.

1.3 Висновки за розділом 1

Огляд ПЗ продажу старих меблів включав ряд існуючих популярних вебдодатків. У результаті виділено їх основні характеристики, запропоновано власну концепцію комплексної підтримки продажу старих меблів, яку не було виявлено в жодній з існуючих програм.

Досягнення мети має бути забезпечено виконанням завдань:

- аналіз програмних додатків, придатних для продажу старих меблів;
- проєктування ПЗ комплексної підтримки продажу старих меблів;
- реалізація і тестування програми за вимогами для продажу старих меблів.

2 ПРОЄКТУВАННЯ ПРОГРАМИ

2.1 Вибір засобів проєктування і розробки

У табл. 2.1 виконано вибір мови програмування для розробки ПЗ комплексної підтримки продажу старих меблів, що зроблений у підсумку на користь мови Python [8]-[9]. Вона дозволяє простіше створювати класи, дозволяє швидше створювати базові рішення, особливо на основі вебфреймворку, що сприяє швидкості розробки. Оскільки це веброзробка, то логічним є використання мов HTML, CSS, JavaScript, технології AJAX. Вебфреймворк Django [10] спрощує роботу з даними, дозволяючи визначати структуру бази даних (БД) на основі коду, а також зменшуючи кількість необхідних для реалізації запитів, тому обраний він.

Таблиця 2.1 – Вибір мови програмування для розробки ПЗ комплексної підтримки продажу старих меблів

Показник	Python	C#
Підтримувана ліцензія	Відкрита	Відкрита
Складність реалізації об'єктно-орієнтованої парадигми програмування	Нижча	Вища
Вебфреймворк з інтегрованим фреймворком для роботи з даними	Наявний	Наявний
Швидкість створення базових рішень	Вища	Нижча

2.2 Визначення функціональних вимог до програмного забезпечення

Спершу були визначені функції, які мають забезпечуватися для неавторизованого користувача (рис. 2.1).



Рисунок 2.1 – Діаграма прецедентів неавторизованого користувача

Сценаріями, які мають бути забезпеченими для неавторизованого користувача, є:

- пошук старих меблів;

- перегляд найсвіжіших оголошень про продаж старих меблів;
- перегляд найкращих майстрів;
- перегляд найкращих перевізників;
- авторизація з альтернативним сценарієм реєстрація у випадку відсутності акаунта у користувача;

- перегляд повної інформації про користувача, що узагальнює сценарії перегляду повної інформації про перевізника, перегляду повної інформації про майстра, перегляду повної інформації про продавця, перегляду повної інформації про покупця;

- перегляд старих меблів, виставлених на продаж за продавцем;
- перегляд оголошення про продаж старих меблів.

Сценаріями, які мають бути забезпеченими для покупця окрім сценаріїв неавторизованого користувача, є (рис. 2.2):

- збереження старих меблів для повторного перегляду;
- перегляд переліку збережених старих меблів;
- оформлення заявки на придбання старих меблів;
- перегляд старих меблів, виставлених на продаж за продавцем;
- створення заявки на обслуговування доставки;
- оплата доставки меблів;
- відправлення повідомлення;
- читання повідомлень;
- оцінювання результатів взаємодії;
- перегляд повного комплексу дій над придбаними меблями;
- оплата заявки на покупку старих меблів;
- перегляд власних заявок на придбання старих меблів;
- створення плану реставраційних робіт старих меблів;
- оформлення заявки на оцінювання плану реставраційних робіт;
- завершення заявки на реставрацію;
- оплата заявки на реставрацію меблів, що узагальнює сценарій передоплати заявки на реставрацію меблів.

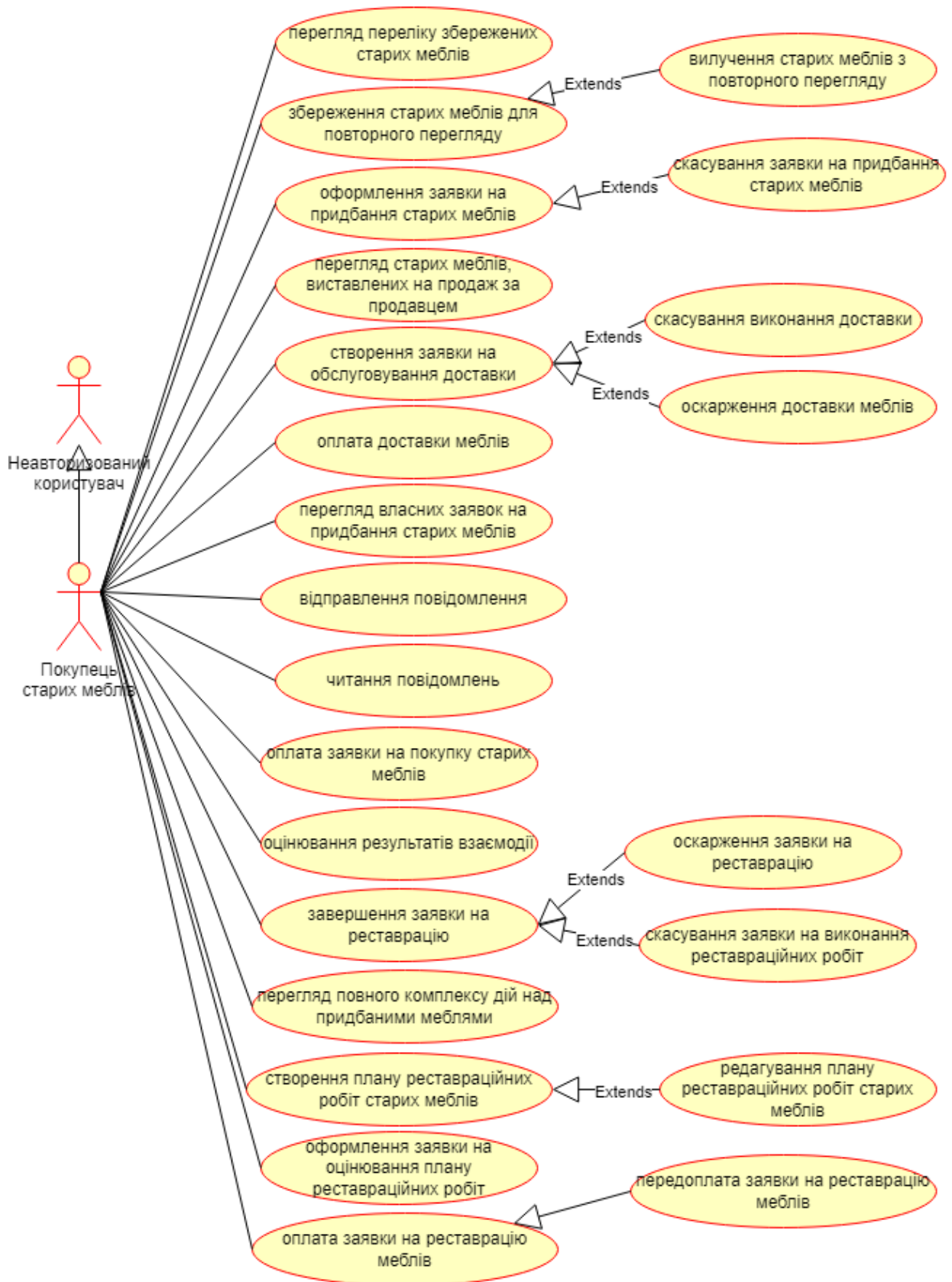


Рисунок 2.2 – Діаграма прецедентів покупця старих меблів

Сценарій збереження старих меблів для повторного перегляду в якості альтернативного сценарію має вилучення старих меблів з повторного

перегляду, коли покупець не бажає більше повертатися до цього набору меблів.

Сценарій завершення заявки на реставрацію в якості альтернативного сценарію має оскарження заявки на реставрацію, коли покупець не отримав відреставровані меблі вчасно або коли він отримав їх не в тому вигляді, як було обумовлено, та скасування заявки на виконання реставраційних робіт, коли покупець передумав виконувати реставрацію за заявкою, яку раніше оформив.

Сценарій створення плану реставраційних робіт старих меблів в якості альтернативного сценарію має редагування плану реставраційних робіт старих меблів, коли покупець бажає внести зміни до плану, який раніше створив.

Сценарій створення заявки на обслуговування доставки в якості альтернативного сценарію має скасування виконання доставки, коли покупець передумав виконувати доставку, на яку раніше оформив обслуговування, та оскарження доставки меблів, коли покупець не отримав меблів вчасно після доставки або отримав їх з пошкодженнями.

Сценарій оформлення заявки на придбання старих меблів в якості альтернативного сценарію має скасування заявки на придбання старих меблів, коли покупець вирішив відмовитись від придбання старих меблів, на яке він раніше подав заявку.

Сценаріями, які мають бути забезпеченими для продавця окрім сценаріїв неавторизованого користувача, є (рис. 2.3):

- перегляд заявок на покупку за власним набором старих меблів;
- виставлення власних старих меблів на продаж, що в якості альтернативного сценарію має скасування продажу власних старих меблів, коли продавець не бажає більше продавати відповідні меблі, та редагування оголошення про продаж власних старих меблів, коли продавець бажає змінити параметри продажу;
- перегляд переліку наборів власних старих меблів;

- відправлення повідомлення;
- читання повідомлень;
- оцінювання результатів взаємодії з покупцем;
- перегляд історії зміни вартості за власним набором меблів;
- підтвердження заявки на покупку старих меблів, що в якості альтернативного сценарію має скасування заявки на придбання старих меблів, коли продавець не бажає продавати власні меблі цьому покупцю;
- підтвердження відправлення старих меблів за заявкою на покупку.

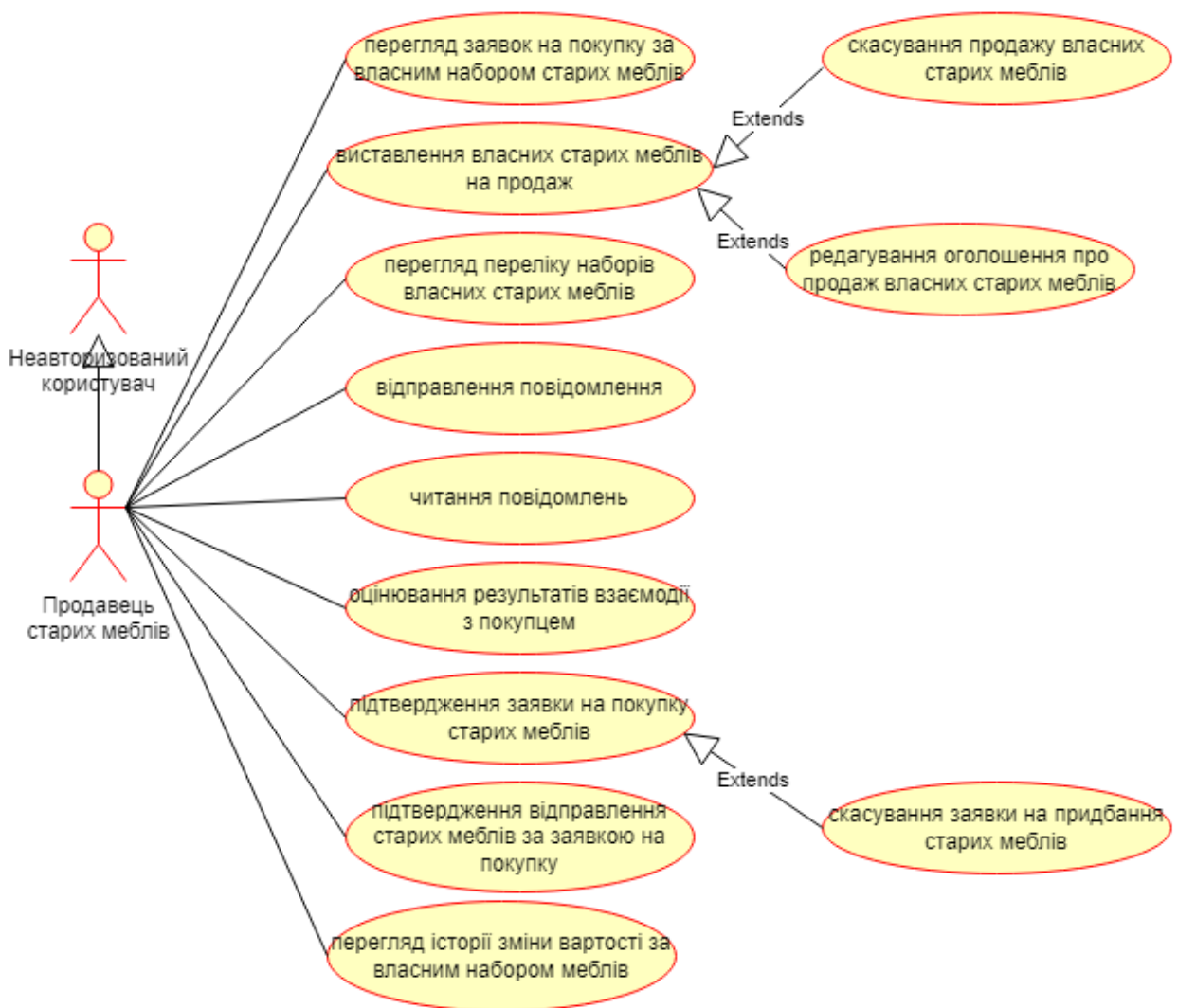


Рисунок 2.3 – Діаграма прецедентів продавця старих меблів

Сценаріями, які мають бути забезпеченими для майстра окрім сценаріїв неавторизованого користувача, є (рис. 2.4):

- відправлення повідомлення;
- читання повідомлень;
- оцінювання результатів взаємодії з покупцем;
- оцінювання реставраційних робіт за заявкою;
- перегляд власних заявок на реставрацію;
- підтвердження початку виконання реставраційних робіт за заявкою;
- підтвердження заявки на реставраційні роботи;
- завершення виконання реставраційних робіт за заявкою.

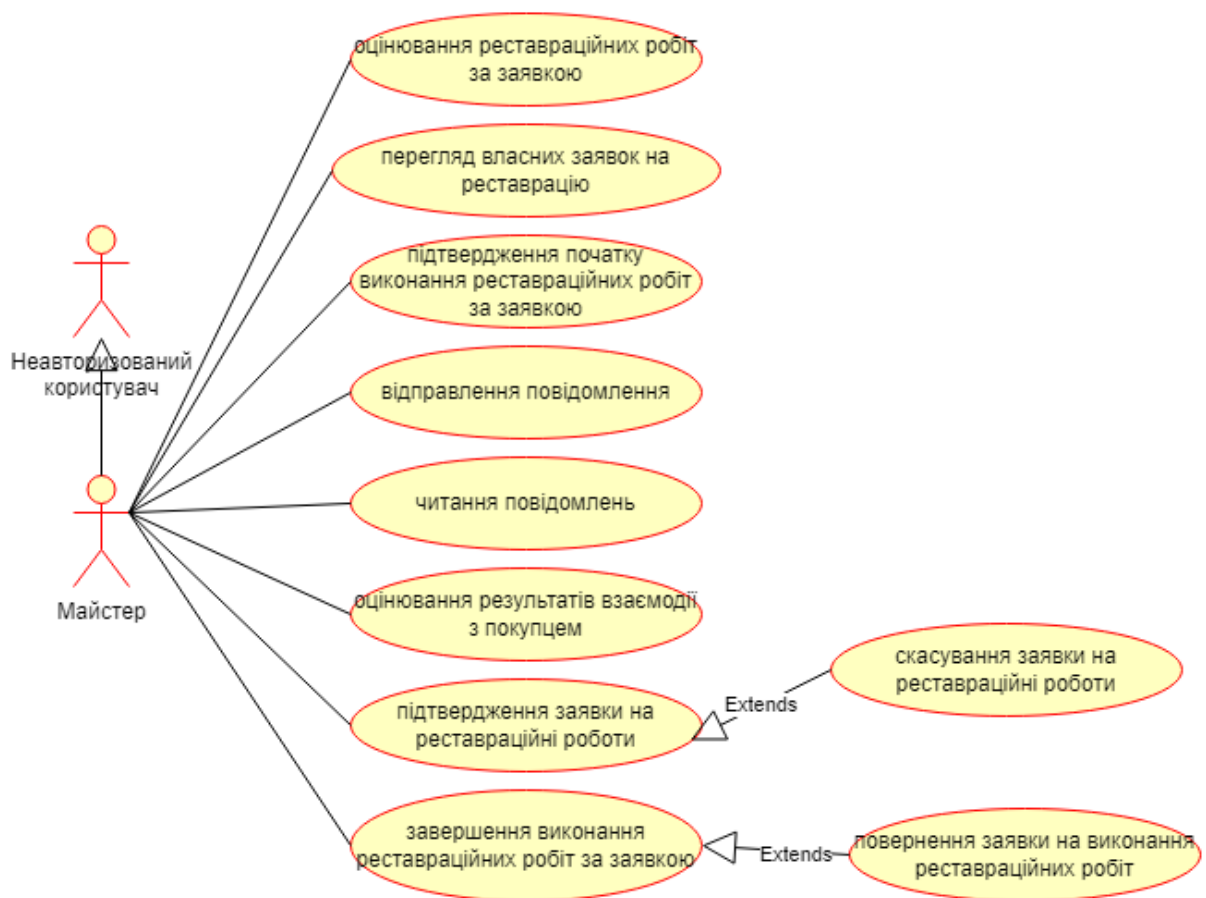


Рисунок 2.4 – Діаграма прецедентів майстра

Сценарій підтвердження заявки на реставраційні роботи має в якості альтернативного сценарію скасування заявки на реставраційні роботи, коли майстер відмовляється виконувати оцінювання або реалізовувати реставрацію. Сценарій завершення виконання реставраційних робіт за заявкою має в якості альтернативного сценарію повернення заявки на

виконання реставраційних робіт, коли доставку не вдалося виконати перевізнику.

Також було визначено сценарії, які мають бути забезпеченими для перевізника (рис. 2.5).

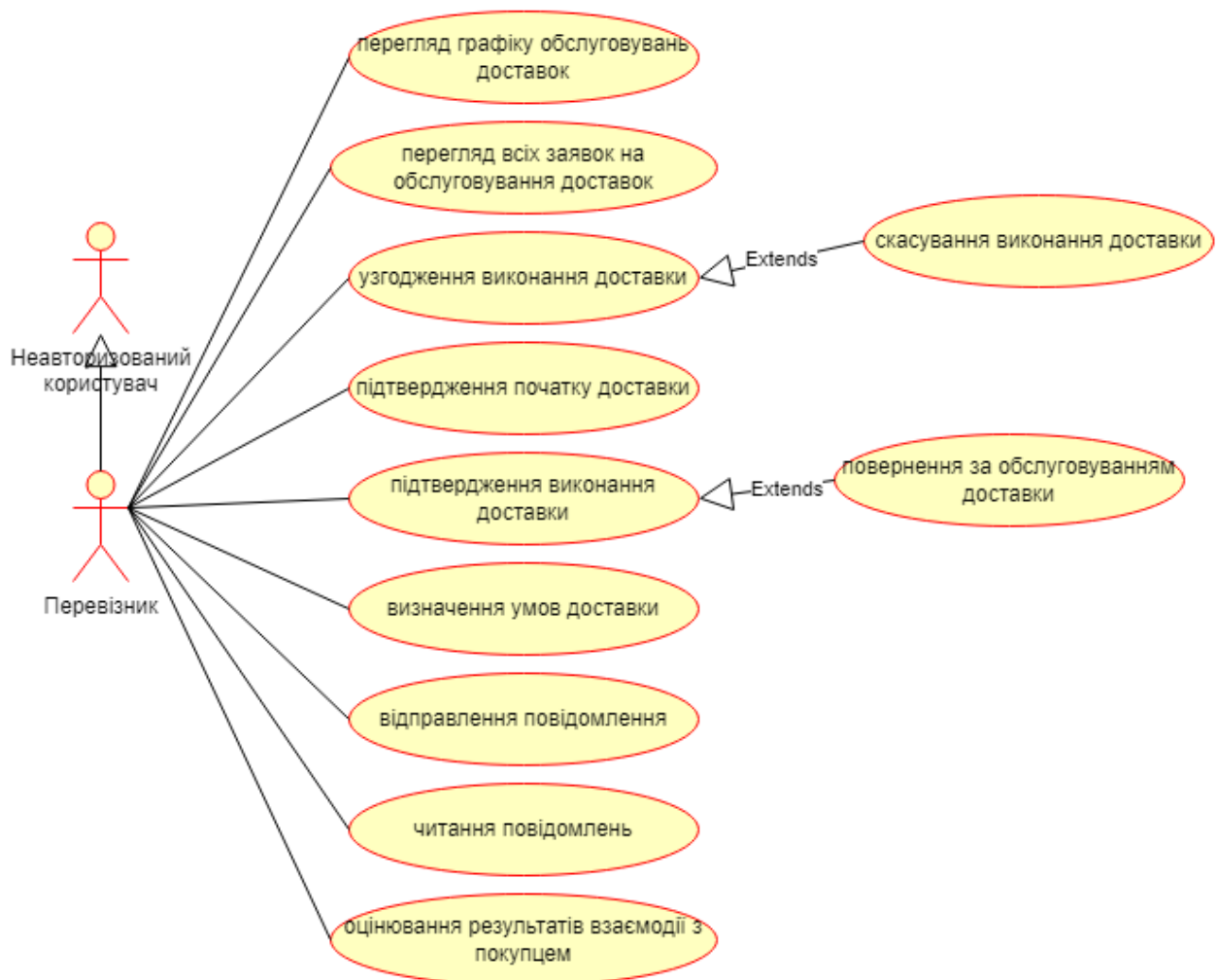


Рисунок 2.5 – Діаграма прецедентів перевізника

Сценаріями, які мають бути забезпеченими для перевізника окрім сценаріїв неавторизованого користувача, є:

- перегляд графіку обслуговувань доставок;
- перегляд всіх заявок на обслуговування доставок;
- узгодження виконання доставки;
- скасування виконання доставки;
- підтвердження початку доставки;

- підтвердження виконання доставки;
- повернення за обслуговуванням доставки;
- визначення умов доставки;
- відправлення повідомлення;
- читання повідомлень;
- оцінювання результатів взаємодії.

2.3 Проєктування структури бази даних

Для роботи з БД використовується система PostgreSQL [11], враховуючи, що вона не вимагає сплати за користування, при тому вона дозволяє працювати з великою кількістю звернень, а така кількість можлива для даного ПЗ у майбутньому.

Під час роботи з самими меблями було вирішено використати в основі цієї роботи конкретний предмет, але при тому вносити їх як елементи до складу меблевих наборів. Це дозволяє закріплювати в подальшому дії і зберігати інформацію як стосовно окремих предметів, так і стосовно цілих меблевих наборів. Як результат цього рішення, покупець може вирішити придбати окремий предмет, а може – цілий набір (комплект). Відповідно продавець може вирішувати, чи готовий він продати якомусь покупцю один або декілька предметів з набору, чи продати весь набір іншому покупцю, якщо така пропозиція є.

Першою базовою сутністю є відповідно предмет меблів, інформація про який вноситься через таблицю `furnitureitem`. Для представлення інформації про предмет меблів визначено поля.

Поле `id` використовується для однозначної ідентифікації предмета меблів, тобто наприклад, стільця чи столу, який входить в набір. Проте якщо стільці рівноцінні між собою, то дані об'єднуються в один запис (один предмет).

Поле `representedtitle` визначає назву предмета меблів.

Поле `totalheight` визначає максимальну загальну висоту предмета меблів.

Поле `totalwidth` визначає максимальну загальну ширину предмета меблів.

Поле `totallength` визначає максимальну загальну довжину предмета меблів. Тобто в підсумку всі три поля визначають габарити предмета меблів.

Поле `additionalmeasurement` визначає будь-який додатковий опис щодо розмірів предмета, який може включати будь-які інші вимірювання або вимірювання певних частин (наприклад, чистий розмір спального місця для ліжка).

Поле `conditiontype` визначає тип стану даного предмета, як його оцінює продавець. Він має задаватися з обмеженого набору, щоб характеризувати значення на шкалі між новим станом і станом, що потребує обов'язкового ремонту або використання його як складових.

Поле `conditiondescription` визначає детальний текстовий опис стану даного предмета меблів. Сюди можуть вписуватися будь-які деталі, які визначають відмінність стану предмета від нового (плями, потертості, подряпини тощо).

Поле `maincolor` визначає основний колір предмета меблів.

Поле `deliveryweight` визначає вагу предмета, якою вона буде під час доставки, тобто у випадку наявності упакування – разом з ним.

Поле `deliverydetail` визначає наявні деталі упакування (наявність коробки, наприклад) для доставки або особливості доставки, які необхідні для доставки цього предмета.

Поле `description` визначає детальний текстовий опис самого предмету меблів, якщо це потрібно.

Поле `ownerprice` визначає вартість предмета, яку встановив продавець, якщо він вважає можливим продати даний предмет окремо від всього набору. У випадку від'ємного значення передбачається, що продавець не бажає

продавати даний предмет окремо, тобто дані про предмет представлені тільки для інформації.

Поле `quantity` визначає кількість однакових предметів у даному наборі. Тобто це кількість, наприклад, стільців з набору, які мають однакові характеристики за значеннями описаних полів.

Поле `assembled` визначає, чи даний предмет меблів знаходиться в зібраному стані, чи потребує зборки для використання.

Поле `picture` визначає фотографію з предметом меблів.

Поле `furnituregroup_id` визначає набір меблів, до якого даний предмет належить через зовнішній ключ до таблиці `furnituregroup`. Кожен предмет меблів може входити тільки в один набір, при цьому він не може не входити в жоден набір. Кожен набір може містити від одного і більше конкретних предметів.

Поле `state` визначає стан продажу даного предмету меблів. Тобто оголошення предмет меблів знаходяться в продажі, предмет проданий, але ще не відвантажений, або предмет меблів відвантажений.

Відповідно другою складовою інформування про старі меблі є набір меблів, що відповідно складається з окремих предметів. Інформація про набір меблів вноситься через таблицю `furnituregroup`. Для представлення інформації про предмет меблів визначено поля.

Поле `id` використовується для однозначної ідентифікації набору меблів.

Поле `owner` визначає продавця всього даного набору через зовнішній ключ до таблиці `user`. Кожен продавець може виставляти на продаж від одного і більше наборів меблів у різний час, але оскільки спочатку він створює свій акаунт, то допускається також відсутність у нього таких наборів взагалі. У той же час для набору меблів при збереженні має бути зазначено продавця, при чому одного.

Поле `original` визначає країну виробництва даного набору меблів.

Поле `ownerprice` визначає вартість набору, яку встановив продавець.

Поле `state` визначає стан продажу даного набору меблів. Тобто меблі знаходяться в продажі, меблі продані, але ще не відвантажені, або меблі відвантажені.

Поле `city_id` визначає місто, в якому знаходиться набір, тобто звідки його може забрати покупець, через зовнішній ключ до таблиці `city`. Кожен набір повинен бути зафіксований у конкретному місті, а місто може охоплювати від 0 і більше наборів меблів.

Поле `address` визначає адресу знаходження набору меблів у межах міста у текстовому вигляді.

Поле `conditiontype` визначає тип стану даного набору меблів, як його оцінює продавець. Він має задаватися з обмеженого набору, щоб характеризувати значення на шкалі між новим станом і станом, що потребує обов'язкового ремонту або використання його як складових.

Поле `conditiondescription` визначає детальний текстовий опис стану даного набору меблів. Сюди можуть вписуватися будь-які деталі, які визначають відмінність стану набору меблів від нового (плями, потертості, подряпини тощо).

Поле `creation` визначає дату створення оголошення про продаж набору меблів.

Поле `lastupdate` визначає дату будь-якого останнього оновлення оголошення про продаж набору меблів.

Поле `manufacturedate` визначає рік виробництва набору меблів. У випадку, якщо конкретний рік не відомий і встановлюється інтервал (як наприклад, для антикваріату), то дане поле містить рік початку інтервалу.

Поле `manufactureperiod` визначає для інтервалу рік його завершення.

Поле `origin` визначає країну виробництва даного набору меблів. Ця країна визначається через зовнішній ключ до таблиці `country`. Кожен набір повинен бути зафіксований за конкретною країною або за жодною, а сама країна може мати від 0 і більше наборів.

Поле `picture` визначає фотографію з набором меблів.

Поле `description` визначає детальний текстовий опис самого набору меблів, якщо це потрібно.

Поле `brand_id` визначає виробника даного набору меблів через зовнішній ключ до таблиці `brand`. Кожен виробник може мати від нуля (тимчасово) і більше конкретних наборів меблів. Кожен набір меблів може мати тільки одного виробника.

Поле `departuredate` визначає дату, коли продавець готовий віддати або відправити даний набір покупцю.

Поле `maxdate` визначає останню дату, коли покупець повинен забрати даний набір меблів.

Меблі знаходяться за конкретними адресами, звідки їх потрібно забрати, адже вони надаються звичайними продавцями, які найчастіше цими меблями володіють, відповідно фактична адреса може бути достатньо різною, а в пошуку потрібно розуміти місто як основну точку структуризації пошуку. Інформація про міста вноситься через таблицю `city`, в якій ідентифікується місто через поле `id`, а сама назва – через поле `name`. Міста знаходяться в межах країн, тому поле `country_id` реалізується через зовнішній ключ до таблиці `country`, де відповідно зберігається інформація про країни, включаючи ідентифікацію через поле `id` та назву країни – через поле `name`.

Необхідна для зручної роботи з меблями інформація про виробників вноситься через таблицю `brand`, в якій ідентифікується виробник меблів через поле `id`, повна назва – через поле `fullname`, а скорочений варіант для відображення в програмі – через поле `presentationname`.

Тип простору, для якого призначені меблі, міститься в таблиці `spacetype`, де ідентифікується через поле `id`, а сама назва – через поле `name`.

При тому кожен набір меблів може бути потенційно розташований в різних типах просторів. Тобто продавець може надавати розширений набір варіантів. Але продавець може і не встановити таку інформацію взагалі, якщо не вважає це за потрібне. Звичайно, що з іншого боку багато різних оголошень можуть стосуватися конкретного типу простору. А в початковий

момент роботи програми таких випадків може бути і 0. Враховуючи це, при проєктуванні реляційної БД була створена таблиця `furnituregroup_spacetype` для реалізації цього зв'язка. Оскільки вона тільки реалізує зв'язок, то до її складу входять поля, що визначають зовнішній ключ до відповідної таблиці: `furnituregroup_id` – набір меблів, `spacetype_id` – тип простору, де може розташовуватися цей набір меблів.

Усіх користувачів було уніфіковано через загальну інформацію про них, ідентифікуючи, чи є користувач, який володіє акаунтом у програмі, продавцем, покупцем, власником чи перевізником, на рівні ролі акаунта, яка визначається відповідним полем. Інформація про користувачів вноситься через таблицю `ouser`. Для представлення інформації про користувачів програми визначено поля.

Поле `id` використовується для однозначної ідентифікації користувача.

Поле `user` використовується для ідентифікації акаунта користувача, що визначається як зовнішній ключ зі стандартної таблиці, яка формується Django.

Поле `phone` визначає номер телефону користувача, у тому числі продавця, покупця, власника чи перевізника.

Поле `rating` визначає середню оцінку, виставлену користувачу. Значення необхідно зберігати для того, щоб кожного разу при зверненні до цієї інформації не перераховувати середній бал для продавця, покупця, власника чи перевізника. Тобто усереднення відбувається під час виставлення оцінки і зберігається для відповідного користувача в даному полі.

Поле `role` визначає конкретну роль, яку виконує користувач у програмі: продавець, покупець, власник, перевізник.

Поле `city` визначає місто, де живе і/або працює користувач. Тобто звідки він готовий відправити меблі, де він виконує перевезення (основне місто у такому випадку), де він реставрує меблі, або де він готовий отримати

меблі. Деталізує цю інформацію поле `address`, яке визначає адресу знаходження користувача у межах міста у текстовому вигляді.

Додатковою характеристикою меблів є їх стиль, що дозволяє чіткіше підбирати меблі за вимогами покупця. Інформація про стилі вноситься через таблицю `style`, в якій ідентифікується стиль через поле `id`, а сама назва – через поле `name`.

Кожен набір меблів може належати різним стилям. Тобто продавець може надавати розширений набір варіантів стилів. Але продавець може і не встановити таку інформацію взагалі, якщо не вважає це за потрібне. Звичайно, що з іншого боку багато різних оголошень можуть стосуватися конкретного стилю. А в початковий момент роботи програми таких випадків може бути і 0. Враховуючи це, при проектуванні реляційної БД була створена таблиця `furnituregroup_style` для реалізації цього зв'язка. Оскільки вона тільки реалізує зв'язок, то до її складу входять поля, що визначають зовнішній ключ до відповідної таблиці: `furnituregroup_id` – набір меблів, `style_id` – стиль, до якого належить даний набір.

Меблі виготовляються з матеріалів, тому в таблиці БД `material` міститься інформація про те, які саме матеріали можуть використовуватися для опису меблів. Інформація про матеріали вноситься через таблицю `material`, в якій ідентифікується стиль через поле `id`, а сама назва – через поле `name`.

Кожен предмет меблів може бути виготовлений з різних матеріалів, при чому оскільки ці дані можуть бути в певний момент або загалом не внесені, то мінімальна кількість може дорівнювати 0. Кожен матеріал може використовуватися у багатьох предметах меблі, але спочатку ці дані можуть бути не внесені: тоді матеріалу може не відповідати жоден предмет меблів. Враховуючи це, при проектуванні реляційної БД була створена таблиця `furnitureitem_material` для реалізації цього зв'язка. Оскільки вона тільки реалізує зв'язок, то до її складу входять поля, що визначають зовнішній ключ до відповідної таблиці: `furnitureitem_id` – предмет меблів, `material_id` –

матеріал, з якого створений цей предмет меблів, а поле `maintype` визначає, чи цей матеріал є основним, чи ні.

Усі предмети меблів також мають бути категоризовані, щоб додатково структурувати пошук. Інформація про категорії вноситься через таблицю `category`, в якій ідентифікується категорія через поле `id`, а сама назва – через поле `name`.

Кожен предметів меблів може відноситися до різних категорій, але має бути доданий хоча би до однієї. Кожна категорія відповідно може включати різні предмети меблів, але в початковий момент предмети можуть бути взагалі відсутні. Враховуючи це, при проектуванні реляційної БД була створена таблиця `furnitureitem_category` для реалізації цього зв'язка. Оскільки вона тільки реалізує зв'язок, то до її складу входять поля, що визначають зовнішній ключ до відповідної таблиці: `furnitureitem_id` – предмет меблів, `category_id` – категорія, до якої відноситься цей предмет меблів.

Так само до категорій належать і набори меблів. Використовуючи аналогічну логіку, була створена таблиця `furnituregroup_category`. Оскільки вона тільки реалізує зв'язок, то до її складу входять поля, що визначають зовнішній ключ до відповідної таблиці: `furnituregroup_id` – набір меблів, `category_id` – категорія, до якої відноситься цей набір меблів.

Обчислення рейтингу користувача базується на відгуках іншої сторони. Інформація про відгуки вноситься через таблицю `feedback`. Для представлення інформації про відгук визначено поля.

Поле `id` використовується для однозначної ідентифікації відгуків.

Поле `owner` визначає автора відгуку, яким може виступати продавець для покупця, майстер для покупця, покупець для майстра, перевізник для покупця, покупець для перевізника. Цей автор визначається через зовнішній ключ до таблиці `user`. Кожен відгук повинен бути зафіксований за конкретним автором, а сам користувач може мати від 0 і більше відгуків.

Поле `addressant` визначає користувача, про якого був створений відгук, яким може виступати покупець для продавця, покупець для майстра, майстер

для покупця, покупець для перевізника, перевізник для покупця. Цей автор визначається через зовнішній ключ до таблиці `ouser`. Кожен відгук повинен стосуватися конкретного користувача, а сам користувач може мати від 0 і більше відгуків.

Поле `star` визначає кількість балів, виставлену у відгуку.

Поле `creation` визначає час, у який був створений відгук.

Поле `fullfeedback` визначає весь текст відгуку.

Історія зміни вартості набору меблів може бути корисною для продавця, щоб розуміти, які зміни він вже виконав. Інформація про відгуки вноситься через таблицю `pricehistory`. Для представлення інформації про історію змін визначено поля.

Поле `id` використовується для однозначної ідентифікації запису про вартість набору меблів.

Поле `furnituregroup_id` визначає набір меблів через зовнішній ключ до таблиці `furnituregroup`. Кожен набір може мати від нуля і більше записів про вартість. Запис робиться тільки тоді, коли вартість змінюється. Тобто при першому збереженні вартості предмету меблів запис не виконується. Якщо значення буде змінено, тільки тоді буде внесено запис у цю таблицю.

Поле `price` визначає значення вартості, якому до даного моменту дорівнювала вартість набору.

Поле `creation` визначає час, коли було змінено цю вартість.

Реставраційні роботи стосуються конкретних предметів і наборів, проте в програмі має бути визначено каталог реставраційних робіт. Інформація про реставраційні роботи вноситься через таблицю `renovation`, в якій ідентифікується категорія через поле `id`, а сама назва роботи – через поле `jobname`.

Самі ж реставраційні роботи, які надаються майстром, зберігаються в таблиці `masterrenovation`. Для представлення інформації про реставраційні роботи, які надаються майстром, визначено поля.

Поле `id` використовується для однозначної ідентифікації реставраційної роботи майстра.

Поле `renovation` визначає реставраційну роботу через зовнішній ключ до таблиці `renovation`. Кожна робота може від нуля і більше разів стосуватися записів про реставраційні роботи конкретних майстрів. Нуль записів зберігається тоді, коли реставраційна робота вже внесена в БД, але вона ще не використана жодним майстром для деталізації.

Поле `master` визначає майстра через зовнішній ключ до таблиці `master`. Кожен майстер може мати від нуля і більше записів про реставраційні роботи. Нуль записів зберігається тоді, коли майстер вже внесений у БД, а його реставраційна робота ще не зазначена.

Поле `askprice` визначає вартість оцінювання такої реставраційної роботи майстром.

Поле `price` визначає середню вартість виконання даної реставраційної роботи майстром.

Для реставрації меблів створюються плани, які визначають перелік реставраційних робіт. Самі плани зберігаються в таблиці `renovationplan`. Для представлення інформації про плани реставраційних робіт визначено поля.

Поле `id` використовується для однозначної ідентифікації плану реставраційних робіт.

Поле `customer` визначає покупця через зовнішній ключ до таблиці `customer`. Кожен покупець може мати від нуля і більше планів реставраційних робіт.

Поле `furnitureitem` визначає предмет меблів через зовнішній ключ до таблиці `furnitureitem`. Кожен предмет меблів може мати від нуля і більше планів реставраційних робіт.

Поле `fgroup` визначає, чи цей план стосується конкретного предмету меблів, чи цей предмет тільки представляє весь набір, над яким потрібно виконати ці реставраційні роботи.

Поле `planfile` визначає файл з зображенням плану реставраційних робіт, якщо покупець хоче виконати не тільки ремонт, але і модернізацію, зміну меблів, і має конкретний план результату.

Поле `creation` визначає час, коли план реставраційних робіт було створено.

Поле `state` визначає стан, в якому знаходиться план.

Кожен план реставрації може включати різні роботи, але повинен містити хоча би одну. Кожна реставраційна робота може стосуватися різних планів, але в початковий момент може не бути включена до жодного плану. Враховуючи це, при проектуванні реляційної БД була створена таблиця `renovationplanjob` для реалізації цього зв'язка. Оскільки вона тільки реалізує зв'язок, то до її складу входять поля, що визначають зовнішній ключ до відповідної таблиці: `renovationplan_id` – план, `renovation_id` – реставраційна робота, яка входить до цього плану.

План реставраційних робіт створюється не тільки для того, щоб визначити наявність такого плану, роботи, які в нього входять, але і для того, щоб визначити, який майстер і за яку суму готовий виконати цей план. Ця інформація разом з самими результатами реставрації представляється сутністю Заявка. Вона фактично об'єднує запит покупця на виконання оцінки, результати консультацій з майстром і подальші результати реставрації, що представляються через таблицю `renovationplanapplication` з заявкою з наступними полями.

Поле `id` використовується для однозначної ідентифікації заявки.

Поле `renovationplan_id` визначає план реставраційних робіт, до якого було визначено заявку, через зовнішній ключ до таблиці `renovationplan`. Кожен план реставраційних робіт може мати від нуля і більше заявок до майстрів.

Поле `master` визначає майстра, до якого покупець звернувся з необхідністю проаналізувати або відремонтувати старі меблі за відповідною

заявкою. Визначається через зовнішній ключ до таблиці `ouser`. До кожного майстра може створювати від нуля і більше заявок.

Поле `price` визначає сумарну вартість, на яку оцінив всі реставраційні роботи за планом майстер.

Поле `paid` визначає суму, яку було сплачено за даною заявкою.

Поле `state` визначає стан цієї заявки, що важливо, щоб відрізняти, чи вона тільки сформована покупцем, чи майстер вже оцінив її, чи покупець прийняв цю пропозицію, чи вона вже виконана.

Поле `creation` визначає час, коли було створено заявку на реставрацію.

Поле `lastupdate` визначає час, коли було востаннє змінено заявку на реставрацію.

Поле `customercomment` визначає коментар від покупця, якщо він бажає надати додаткову інформацію майстру.

Поле `mastercomment` визначає коментар від майстра, якщо він бажає надати додаткову інформацію покупцю.

Поле `startdate` визначає час, коли майстер зможе розпочати реставрацію меблів.

Поле `enddate` визначає час, коли майстер планує закінчити реставрацію меблів.

Інформація про самих перевізників представляється через інформацію про користувачів. Це дозволяє працювати звичайним приватним перевізникам. Але покупців цікавлять умови, на яких перевізник готовий перевозити меблі. Інформація про набір меблів вноситься через таблицю `delivery`. Для представлення інформації про умови доставки визначено поля.

Поле `id` використовується для однозначної ідентифікації умов доставки.

Поле `carrier` визначає перевізника, який надає ці умови доставки, через зовнішній ключ до таблиці `ouser`. Кожен перевізник може визначити від нуля і більше умов доставки. Мінімальне нульове значення необхідне, коли майстер вже зареєструвався, але ще не має жодної визначеної умови.

Поле `city` визначає місто, в якому надає послуги перевізник, через зовнішній ключ до таблиці `city`. Кожен місто може мати від нуля і більше умов доставки від різних перевізників.

Поле `startradius` визначає відстань від меж міста, за яку готовий виконувати доставку перевізник.

Поле `endradius` визначає відстань від точки початку доставки, на яку готовий виконувати доставку перевізник.

Поле `price` визначає вартість доставки за даних умов стосовно одного кілометра.

Поле `maxweight` визначає максимальну вагу, яку можна перевезти за даним тарифом.

Поле `maxheight` визначає максимальні сумарні габарити меблів за висотою, які готовий доставити перевізник за даних умов.

Поле `maxwidth` визначає максимальні сумарні габарити меблів за шириною, які готовий доставити перевізник за даних умов.

Поле `maxlength` визначає максимальні сумарні габарити меблів за довжиною, які готовий доставити перевізник за даних умов.

Поле `distance` визначає максимальну відстань доставки у кілометрах, на яку діють дані умови.

Безпосередньо інформація про обслуговування доставки зберігається в таблиці `deliveryservice`. Для представлення інформації про обслуговування доставки визначено поля.

Поле `id` використовується для однозначної ідентифікації обслуговування доставки.

Поле `application_id` визначає заявку на покупку (фактично покупку на момент створення обслуговування доставки), за якою відбувається доставка, через зовнішній ключ до таблиці `application`. Кожна заявка може мати від нуля до одного обслуговування доставки. Кожне обслуговування стосується тільки однієї заявки.

Поле `delivery_id` визначає умови доставки, за якими відбувається це обслуговування доставки, через зовнішній ключ до таблиці `delivery`. Кожна умова доставки може мати від нуля і більше обслуговувань. Кожне обслуговування стосується тільки однієї конкретної умови.

Поле `price` визначає сумарну вартість обслуговування цієї доставки, яку має сплатити покупець.

Поле `paid` визначає суму, яку було сплачено за обслуговування на даний момент.

Поле `creation` визначає час, коли було створено запис про обслуговування доставки.

Поле `state` визначає стан обслуговування.

Поле `completedate` визначає час, коли буде розпочато доставку.

Поле `startuser` визначає користувача, від якого виконується доставка, через зовнішній ключ до таблиці `auser`. Таким користувачем може бути і продавець, і майстер. Кожен продавець або майстер може відправляти різну кількість доставок. Мінімальне число – нуль, коли після реєстрації користувача ще жодної доставки не виконувалось від нього.

Поле `enduser` визначає користувача, до якого виконується доставка, через зовнішній ключ до таблиці `auser`. Таким користувачем може бути і покупець, і майстер. Кожен покупець або майстер може отримувати різну кількість доставок. Мінімальне число – нуль, коли після реєстрації користувача ще жодної доставки не виконувалось до нього.

Поле `startaddress` визначає повну адресу, за якою потрібно забрати меблі на початку доставки.

Поле `endaddress` визначає повну адресу, за якою потрібно повернути меблі в кінці доставки.

Взаємодія покупців з продавцями відбувається головним чином через заявки на покупку, які можуть або стати покупками (продажами), або бути відхиленими. Інформація про заявки вноситься через таблицю `application`. Для представлення інформації про заявки на покупку визначено поля.

Поле `id` використовується для однозначної ідентифікації заявки на покупку.

Поле `furnitureitem_id` визначає предмет меблів, якого стосується пропозиція, через зовнішній ключ до таблиці `furnitureitem`. Кожен предмет меблів може мати від 0 і більше пропозицій.

Поле `customer` визначає покупця, який створив цю заявку на покупку предмету меблів, через зовнішній ключ до таблиці `user`. Кожен покупець може мати від 0 і більше пропозицій.

Поле `lastupdate` визначає дату будь-якого останнього оновлення заявки на покупку.

Поле `fgroup` визначає, чи ця пропозиція стосується конкретного предмета меблів, чи вона стосується всього набору загалом.

Поле `state` визначає стан заявки про придбання предмету меблів, що потрібно головним чином через те, що заявки можуть ставати купівлями.

Поле `creation` визначає час, коли було створено цю заявку на покупку предмету меблів.

Поле `price` визначає вартість, яку було запропоновано або за якою придбано предмет або набір меблів.

Поле `quantity` визначає кількість одиниць, яку готовий купити або купив покупець. Стосується саме предметів меблів.

Поле `comment` визначає коментар покупця щодо цієї заявки.

Для зручного повернення покупцем до відібраних наборів меблів використовуються переліки, до яких кожен користувач може за потреби зберегти той набір, який його зацікавив. Інформація про переліки вноситься через таблицю `wishlist`. Для представлення інформації про переліки визначено поля.

Поле `id` використовується для однозначної ідентифікації позиції у переліку.

Поле customer визначає користувача, якого стосується цей перелік, через зовнішній ключ до таблиці auser. Кожен користувач може мати від 0 і більше збережених наборів (рис. 2.6).

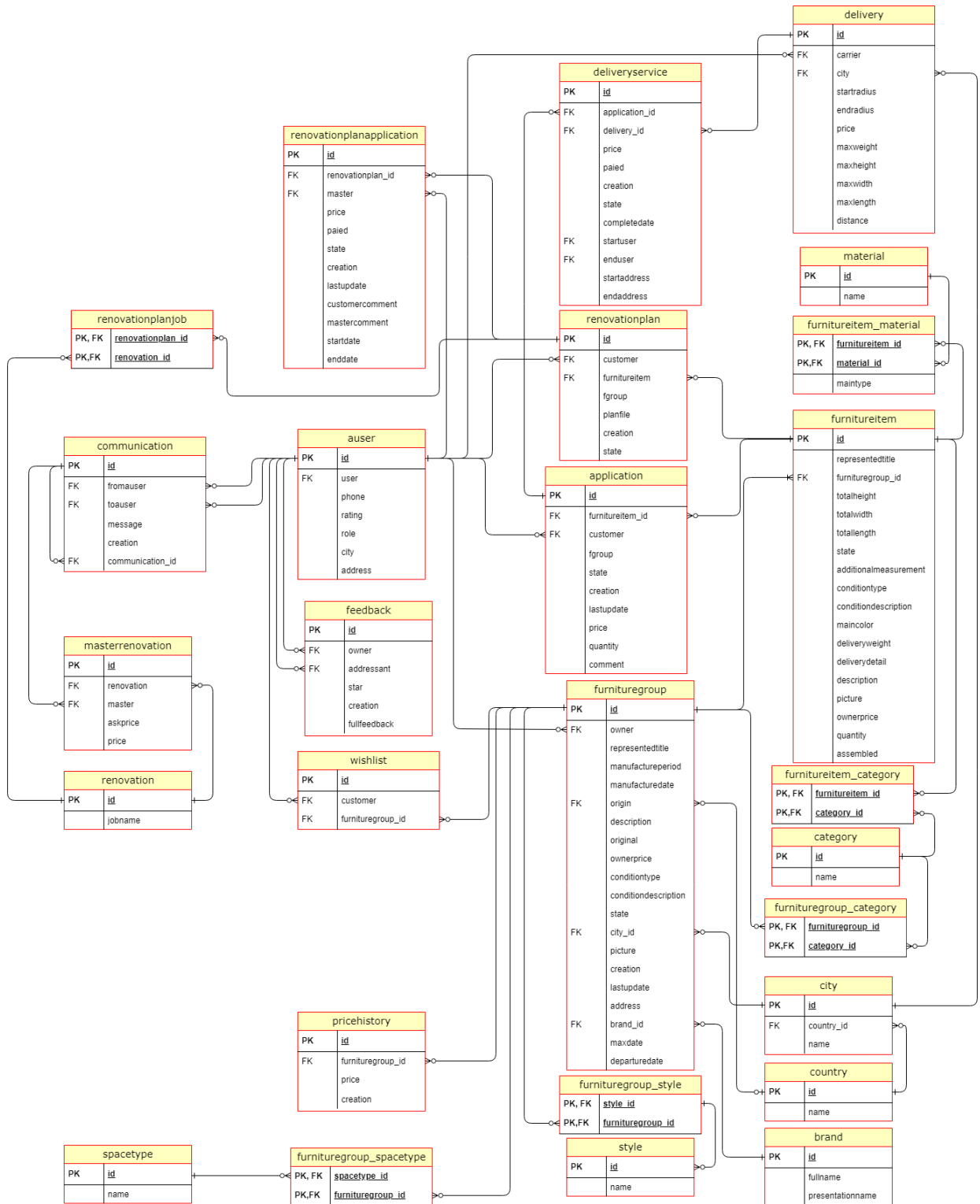


Рисунок 2.6 – Схема БД

Поле `furnituregroup_id` визначає набір меблів, який було збережено в переліку, через зовнішній ключ до таблиці `furnituregroup`. Кожен набір меблів може мати від 0 і більше збережень у переліках різних користувачів.

Для спілкування між користувачами з різними ролями, тобто покупців з майстрами, продавцями і перевізниками, використовується збереження інформації через таблицю `communication`. Для представлення інформації визначено поля.

Поле `id` використовується для однозначної ідентифікації спілкування.

Поле `fromauser` визначає відправника через зовнішній ключ до таблиці `auser`. Кожен користувач може мати від 0 і більше комунікаційних повідомлень.

Поле `toauser` визначає отримувача через зовнішній ключ до таблиці `auser`. Кожен користувач може мати від 0 і більше комунікаційних повідомлень.

Поле `communication_id` визначає ситуації, коли повідомлення є відповіддю на минуле повідомлення, тобто дозволяє зберегти діалог через зовнішній ключ до таблиці `communication`.

Поле `message` визначає сам повний текст повідомлення.

Поле `creation` визначає час, коли безпосередньо було створено повідомлення.

2.4 Висновки за розділом 2

У другому розділі описано результати, пов'язані з вибором мов і технологій розробки, включаючи зокрема мову Python, супутні мови і технології веброботи, фреймворк Django, результати, пов'язані з визначенням функціональних вимог і моделюванням їх на основі діаграм прецедентів, проєктуванням БД.

З РОЗРОБКА ПРОГРАМИ

3.1 Порядок функціонування програми

Порядок функціонування програми для даної предметної області є дуже важливим, адже в загальній системі відбувається взаємодія не тільки різних користувачів, але і різних ролей користувачів. Тому покупка старих меблів в повному вигляді перетворюється на процес. Цей процес може бути тривалим. Окрім того він потребує комунікації між різними учасниками.

Під час комунікації в процесі підтримки покупки старих меблів беруть участь покупець, продавець, майстер і перевізник. Кожен з них має свою долю відповідальності, впливаючи на те, чи буде підсумковий результат успішним. Кожен з цих етапів, які стосуються кожного з перелічених учасників, проходить через набір станів. Але це є не єдиною проблемою. Між собою підпроцеси, які стосуються користувачів з різними ролями, перетинаються. Тому для визначення результатів роботи за кожним з підпроцесів було виділено окремі сутності, описані в попередньому розділі. Порядок роботи з цими сутностями визначає фактично порядок роботи користувача з програмою.

Основними сутностями і відповідними об'єктами, які визначають різні підпроцеси стосовно роботи над покупкою старих меблів, є заявка на покупку старих меблів, заявка на реставраційні роботи та обслуговування доставки старих меблів.

Заявка на реставрацію потребує постійної взаємодії покупця і майстра. Однак вона розпочинається при першому зверненні до майстра, яке найчастіше відбувається ще до придбання старих меблів. Хоча може відбутися і пізніше. Це стосується випадків, коли користувач уже придбав старі меблі, наприклад, потім доставив їх собі додому, а тільки після того вирішив виконати ремонт. Ця робота продовжується, коли покупець отримав меблі і, маючи виконану оцінку майстром, передав меблі на реставрацію. У підсумку було визначено цілий набір станів заявки на реставрацію.

Спочатку покупець створює запит на оцінювання реставраційних робіт. Саме в цьому стані заявка на реставрацію з'являється в системі.

Далі майстер може або погодитися виконати оцінювання (заявка переходить до стану підтвердженої). Тоді покупець має виконати передоплату. Тобто це фактично оплата за послуги оцінювання. За домовленістю між покупцем, продавцем і майстром ця процедура може включати і виїзд до покупця майстра. Однак це вже відбувається поза межами програми у такому випадку.

Якщо майстер не має часу, експертності або з якихось інших причин відмовляється від оцінювання, то заявка переходить до стану відхиленої.

Якщо заявка є підтвердженою, а покупець сплатив передоплату за оцінювання майстром робіт, то заявка переходить до стану передоплаченої. У цьому стані роботи за реставрацію ще не сплачені покупцем, а майстер ще не оцінив їх. Але маючи підтверджену заявку, покупець може змінити плани і вирішити відмовитися від необхідності оцінювання робіт майстром. Тоді заявка переходить також до стану відхиленої.

Якщо заявка є передоплаченою, а майстер почав оцінювати реставраційні роботи за планом, визначивши це в програмі, то заявка переходить до стану «в оцінюванні». Цей стан призводить до того, що покупець вже не може скасувати заявку. Її може скасувати тільки майстер. Якщо він вирішив, що не зможе виконати таку реставрацію або якщо він не може виконати оцінювання, то з цього стану заявка переходить до стану скасованого. Цей стан відрізняється від відхиленого тим, що він не тільки призводить до завершення роботи над заявкою, але і тим, що в такому випадку сплачені гроші мають бути повернені покупцю.

Якщо майстер завершив оцінювання, то заявка переходить до стану оціненої. Далі покупець має вирішити, чи бажає він виконувати реставрацію цих меблів і у цього майстра. Якщо покупець не бажає виконувати реставрацію, то заявка переходить до стану завершеного. У цьому стані

повернення коштів не відбувається, а робота над заявкою успішно завершується.

Якщо покупець обрав цього майстра і ці старі меблі для придбання, то він може сплатити за заявкою за реставрацію. Тоді заявка переходить до стану сплаченої. Після цього майстер може через різні причини відмовитися від реставрації. Тоді заявка переходить до стану скасованої. У цьому випадку сплачені кошти мають бути повернені покупцю.

Якщо майстер, як це передбачено стандартно, виконує свої зобов'язані, то після того, як він отримав старі меблі для реставрації, заявка переходить до стану прийнятої. За стандартним ланцюжком після того, як реставрацію виконано, заявка переходить до стану виконаної. Якщо покупець погоджується з результатами, то на цьому робота з нею закінчується.

Однак у певних ситуаціях цього може бути недостатньо, а тому для захисту інтересів сторін було створено ряд додаткових станів. У випадку, якщо майстер не зміг успішно завершити реставрацію через ті чи інші причини, заявка переходить до стану повернутої. Заявка у стані повернутої передбачає, що не тільки кошти за реставрацію повертаються покупцю, але і самі старі меблі також повертаються йому.

Якщо майстер розпочав роботу, але не завершив її вчасно (можливо не виходить на контакт), то покупець може оскаржити заявку. Тоді вона переходить до стану оскарженої. У такому ж стані заявка може опинитися у тому випадку, якщо заявка була виконана, але покупець оскаржив роботи.

Зі стану оскарженої є 3 варіанти зміни станів. Якщо майстер відмовився взаємодіяти, його не вдалось знайти, компанія, яка підтримує роботу програми, вирішила, що досягти за цією заявкою вже нічого не вдасться, то заявка переходить до стану порушеної.

Якщо за оскарженою заявкою майстер погодився, що не виконав свої зобов'язання і повернув меблі, то заявка переходить до стану повернутої. Якщо майстер виконав додаткові роботи, то оскаржена заявка стає виконаною (рис. 3.1).

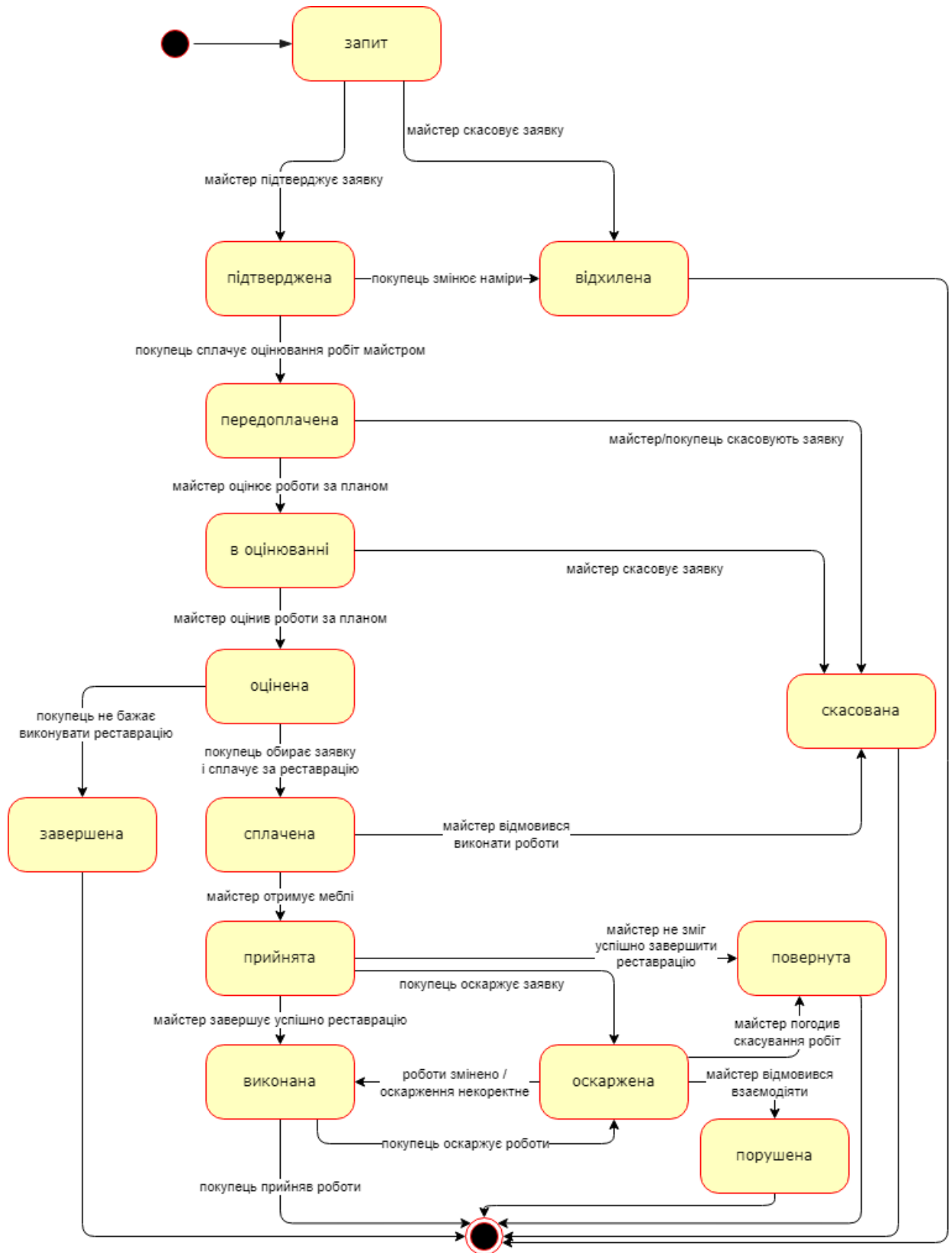


Рисунок 3.1 – Діаграма станів заявки на реставрацію

Майстер отримує сплачену суму тільки після 1 доби після того, як заявка стала виконаною. Передоплату за оцінювання майстер отримує тільки

після того, як заявка стає оціненою.

Заявка на покупку старих меблів розпочинає свій життєвий цикл зі стану запиту. У запиті формується пропозиція покупця на придбання меблів. Пропозиція може включати іншу суму, яка не співпадає з сумою, вказаною продавцем. Продавець вирішує, чи продати свої меблі покупцю за його заявкою. Якщо він погоджується, то заявка переходить до стану підтвердженої. Якщо він відмовляється, то заявка переходить до стану відхиленої (рис. 3.2).

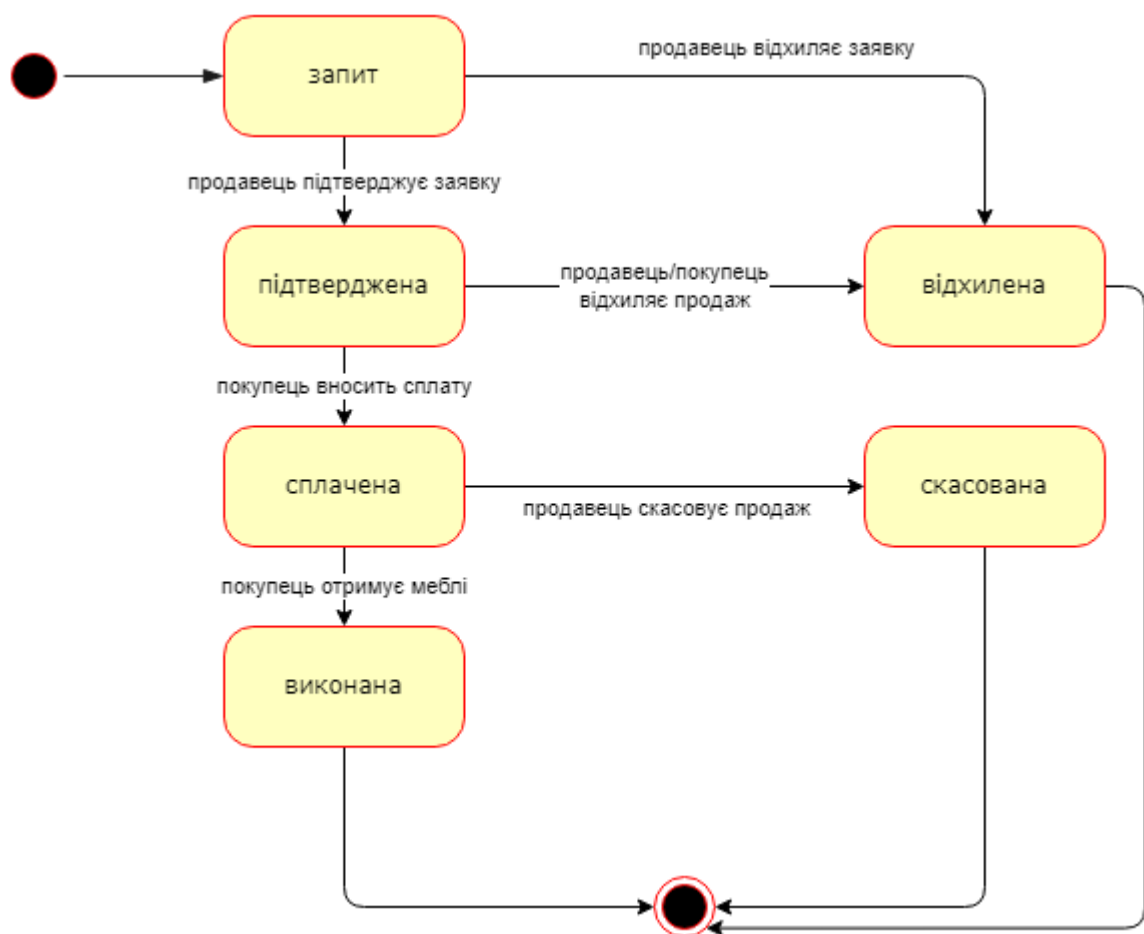


Рисунок 3.2 – Діаграма станів заявки на покупку старих меблів

Якщо заявка на покупку старих меблів є підтвердженою, то покупець може вирішити відмовитися від ідеї придбати саме ці старі меблі. Тоді він переводить заявку до стану відхиленої. Так само може в цей момент вчинити і продавець.

Якщо покупець вирішив скористатися підтвердженням продавця, то заявка зі стану підтвердженої переходить до стану сплаченої після того, як він внесе відповідну суму.

Коли заявка є сплаченою, продавець може скасувати продаж. Тоді вона переходить до стану скасованої, а внесені кошти повертаються. Якщо продавець від ідеї не відмовляється і віддає меблі покупцю, то заявка переходить до стану виконаної.

Продавець отримує сплачену суму тільки після 1 доби після того, як заявка стала виконаною.

Результати взаємодії перевізника з покупцем фіксуються в обслуговуванні доставки (рис. 3.3).

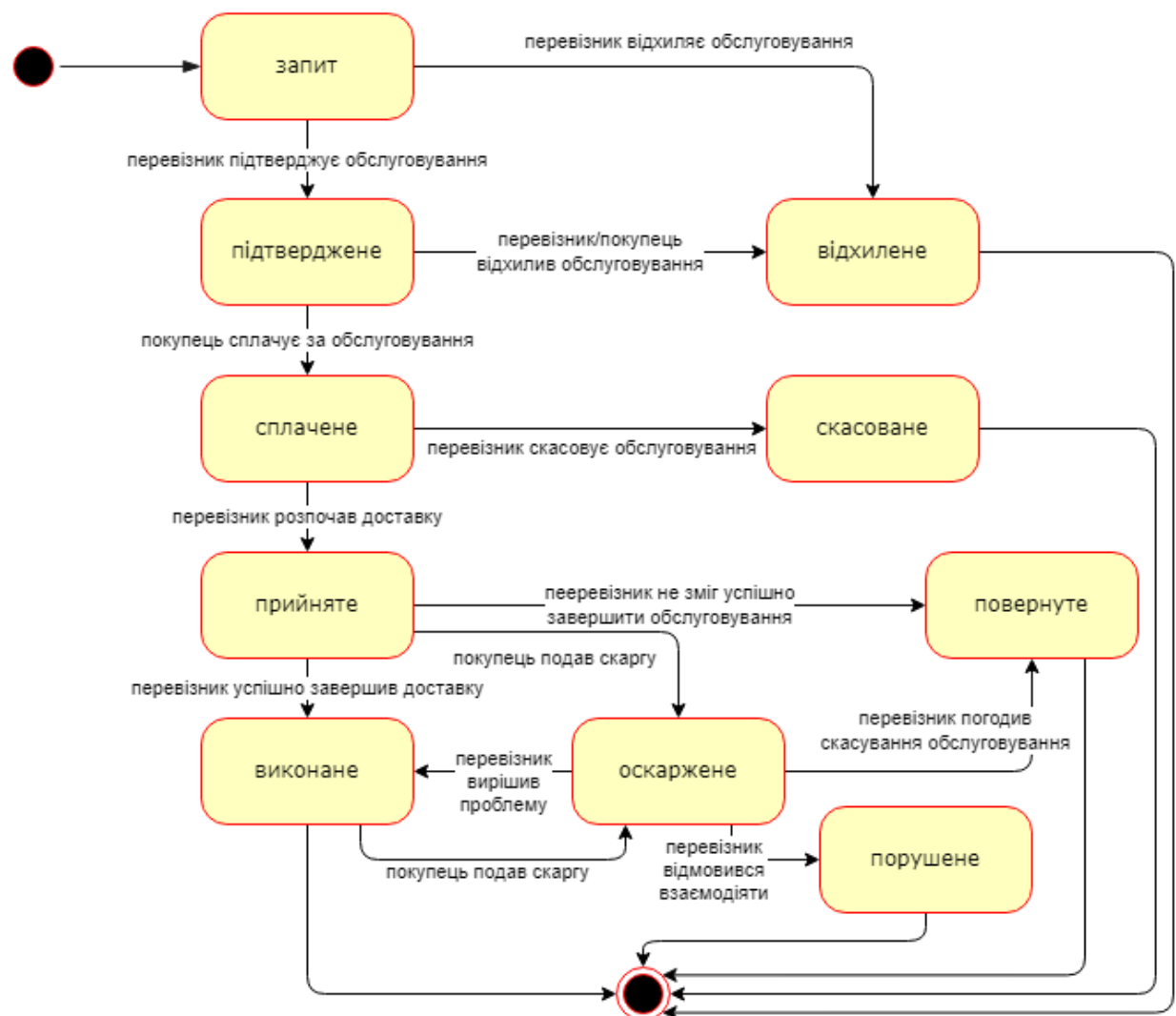


Рисунок 3.3 – Діаграма станів обслуговування доставки

Після створення обслуговування покупцем, воно знаходиться в стані запиту. Після цього з ним взаємодіє перевізник. Якщо перевізник відхиляє обслуговування, то воно переходить до стану відхиленого. Якщо перевізник підтверджує його, то воно переходить до стану підтвердженого. За підтвердженим обслуговуванням перевізник або покупець можуть скасувати обслуговування. Тоді воно переходить до стану відхиленого.

Якщо покупець сплачує за підтверджене обслуговування, то воно переходить до стану сплаченого. Після цього перевізник може скасувати обслуговування у випадку неможливості забезпечити доставку. Обслуговування переходить до стану скасованого. У такому випадку кошти повертаються покупцю.

Якщо обслуговування сплачене, а перевізник розпочав доставку, то воно переходить до стану прийнятого. Якщо перевізник не зміг успішно завершити обслуговування, тобто доставку виконати не вдалось, то обслуговування може перейти до стану повернутого. Якщо доставка не виконана, але перевізник це належним чином не оформив, то покупець може перевести обслуговування до стану оскарженого. Якщо з проблемами погодився перевізник, то обслуговування переводиться до повернутого. Якщо у такій ситуації перевізник зміг вирішити проблему, то обслуговування переходить до стану виконаного. Якщо на випадок оскарженого обслуговування перевізник не відреагував, вирішити проблему не вдалось, то обслуговування переходить до стану порушеного.

Якщо обслуговування у стані прийнятого, а перевізник успішно виконав без проблем доставку, то воно переходить до стану виконаного. Якщо після цього покупець подав скаргу, то обслуговування знову ж таким переходить до стану оскарженого.

Перевізник отримує сплачену суму тільки після 1 доби після того, як обслуговування стало виконаним.

3.2 Рішення з реалізації одиниць додатку

Рішення з реалізації одиниць додатку визначаються, в першу чергу, використанням вебфреймворку Django. Це зокрема пояснює те, що описана в розділі 2 логіка роботи з даними була використана під час створення моделей додатку. Сценарії роботи користувачів були реалізовані в представленнях додатку, формах, підтримувані відповідними шаблонами. Інтерфейс шаблонів побудований на основі вільного шаблону [12].

Усі представлення для зручності підтримки розділені на окремі файли:

- `views_furniture`, що визначає представлення, які стосуються наборів, предметів меблів;
- `views_feedback`, що визначає представлення комунікації, оцінювання здійснених взаємодій;
- `views_application`, що визначає представлення підтримки продажу і покупки старих меблів;
- `views_delivery`, що визначає представлення підтримки обслуговування доставки;
- `views_renovation`, що визначає представлення підтримки реноваційних робіт;
- `views_auth`, що визначає представлення підтримки акаунтів та допоміжних функцій.

Кожному сценарію відповідає певне представлення. Усі вони реалізовані на загальних принципах. Нижче описані основні такі принципи.

Функція `furniture_search` підтримує представлення пошуку меблів. Вона організована універсально і дозволяє виконувати пошук і цілих наборів, і предметів.

Спочатку реалізується пошук всіх наборів меблів, оголошення за якими знаходяться в стані відкритих (значення 0), а потім на цей набір встановлюється ряд фільтрів, що відповідають параметрам пошуку.

Першим перевіряється параметр GET-запиту title, пошук якого виконується в назві набору та в описі. Якщо введене слово або текст містяться в назві або описі, то відповідний набір стає частиною результатів.

Наступними параметрами перевіряються країна виробництва origin, що відповідає ідентифікатору країни, місто, де знаходиться набір меблів, ідентифікатор, якого відповідає параметру city, виробник меблів, ідентифікатор якого відповідає параметру brand, оригінальність меблів (наявне значення параметру original визначає оригінальні меблів, а відсутнє – репліки), вартість меблів, що має знаходитися в інтервалі від min_price до max_price, стан набору меблів, що має знаходитися в інтервалі від min_conditiontype до max_conditiontype, тип простору, де може бути розміщений даний набір, що визначається параметром spacetype, стиль набору, що визначається параметром style, категорія набору, що визначається параметром category.

Перевірка періоду виробництва набору меблів реалізується через співставлення інтервалів, адже виробництво набору задається двома значеннями в БД, а параметри також встановлюються 2 значеннями: manufactur_start для визначення найменшого року виробництва та manufactur_end для визначення найпізнішого року виробництва. Розглядаються всі варіанти перетину цих інтервалів, а всі набори, які таким перетинам відповідають, залишаються в результуючому наборі стосовно даного параметра. За тим самим принципом перевіряється відповідність інтервалів відправлення меблів. Тобто покупець може потребувати забрати меблі в певний період, у той же час продавець також встановлює, коли він найраніше готовий віддати меблі і коли найпізніше покупець має ці меблі забрати. Ці параметри покупцем встановлюються значеннями deliv_start і deliv_end.

Усе це в підсумку дозволяє визначити ті набори меблів, що відповідають даним параметрам, встановленим покупцем.

Після цього окремо виконується пошук окремих предметів меблів. Тобто пошук є універсальним, дозволяючи в підсумку знайти і набори меблів, у тому числі такі, в які входять предмети, що відповідають параметрам пошуку, а також окремі предмети меблів, якщо саме вони цікавлять покупця.

У другій частині реалізується спочатку підготовка результатів, що включають предмети меблів, набори яких все ще знаходяться в продажу (стан зі значенням 0), потім додаються у випадку їх наявності фільтри, що визначають мінімальну висоту предмету `min_height`, його максимальну висоту `max_height`, мінімальну ширину `min_width`, максимальну ширину `max_width`, мінімальну довжину `min_length`, максимальну довжину `max_length`, стан набору меблів, що має знаходитися в інтервалі від `min_conditiontype` до `max_conditiontype`, зібраність меблів (наявне значення параметру `assembled` визначає зібраний предмет меблів, а відсутнє – той, який потребує збирання), кількість одиниць цього предмету меблів, що має знаходитися в інтервалі від `min_quantity` до `max_quantity`, основний колір `maincolor`, вага, що має знаходитися в інтервалі від `min_weight` до `max_weight`, вартість, що має знаходитися в інтервалі від `min_price` до `max_price`. Якщо за якимось з додаткових параметрів окрім тих, які встановлюються також і для наборів меблів, визначено значення, то формується ще один перелік наборів меблів. Цей перелік повинен включати ті набори, які не включені в перелік, сформований на першому етапі пошуку. Потім ці переліки об'єднуються і реалізується сортування. Сортування відбувається так само і в кінці формування першого переліку. Сортування відбувається або за рейтингом продавця, або за вартістю. Це залежить від значення параметра GET-запиту, що називається `sort`. Якщо значення встановлено, то сортування реалізується за рейтингом, а якщо ні, то за вартістю.

Ще одним параметром є `onlyitems`. Якщо він встановлений, то повертаються предмети меблів, а якщо ні, то повертаються тільки набору меблів.

Клас `FurnituregroupSaveView` реалізований через наслідування класів `LoginRequiredMixin` і `View` для підтримки представлення збереження набору меблів для повторного перегляду користувачем. Цей клас підтримує роботи тільки через POST-запити. У відповідному методі виконується витягання даних авторизованого користувача та набору меблів, номер якого відповідає ідентифікатору `pk`, що є параметром методу `post` цього класу. Після витягання цих даних створюється об'єкт класу `Wishlist` з внесенням відповідних значень параметрів, а далі реалізується його збереження. Якщо збереження відбувається неуспішно, то реєструється відповідний запис у лог-файлі. У випадку успішних дій повертається результат `HttpResponse`. Це дозволяє підтримувати технологію AJAX і таким чином оновлювати інформацію про те, чи внесений у поточний момент цей набір меблів у перелік збережених, чи ні в актуальному стані кожну мить. Аналогічно реалізовано зворотню підтримку на основі класу `FurnituregroupSaveUnView` для вилучення набору меблів з переліку збережених.

Підтримка перегляду наборів меблів за продавцем відрізняється тим, що вона виконується на основі класу. Клас `SellerFurnituregroupsView` реалізований через наслідування класу `ListView`. Ідентифікатор продавця передається через параметр `seller` GET-запиту. Аналогічна функціональність без даного параметра, тобто для власних наборів меблів авторизованого продавця, реалізується через клас представлення `OwnFurnituregroupsView`. При цьому через параметр `state` передається стан наборів. Таким чином, якщо конкретний стан задано, то відбираються тільки набори, які відповідають цьому значенню.

Функція `create_furnituregroup` підтримує представлення виставлення набору меблів на продаж. В основі роботи цієї функції знаходиться форма `FurnituregroupForm` для набору меблів та форма `FurnitureitemForm` для

предмету меблів. Використання саме функції обумовлено необхідністю підтримки роботи з двома формами одночасно, адже передбачається, що набір має складатися хоча би з одного предмету. Тому одночасно зі створенням набору створюється перший предмет меблів. Далі можна створювати нові предмети і вносити їх у існуючий набір меблів. Це представлення підтримується через клас `FurnitureitemCreateView`. За необхідності можна редагувати внесені дані. Представлення редагування предмету меблів підтримується через клас `FurnitureitemUpdateView`, а представлення редагування набору меблів підтримується через клас `FurnituregroupUpdateView`. Однак останній клас є значно ширшим, адже в БД також зберігається інформація про зміну вартості набору меблів. Тому якщо за результатами роботи з формою виявляється, що вартість відрізняється від вартості в БД, то створюється об'єкт класу `Pricehistory` і відповідне значення зберігається в БД, вказуючи, яким вартість набору меблів була раніше. Поточне значення зберігається у відповідному полі самого набору. Витягання даних про історію зміни вартості набору меблів виконується через клас `PricehistoryView`. Окрім витягання самих записів про історію зміни також вноситься додатковий останній запис про те, якою є вартість даного набору в поточний момент. Це дозволяє зручно керувати змінами продавцю, маючи для цього всю необхідну інформацію.

3.3 Висновки за розділом 3

У третьому розділі докладно описано послідовність роботи з основними сутностями предметної області на основі зміни їх станів за допомогою діаграм станів та приведення відповідних логічних описів, приведено базові рішення щодо реалізації вебдодатку за допомогою Django, описано класи і функції, які було створено і які є основою створення інших розроблених класів і функцій у даному вебдодатку.

4 ЕКСПЛУАТАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМИ

4.1 Опис програми

Комплексна підтримка спрямовує програмне забезпечення на використання і продавцями, і покупцями, і майстрами, і перевізниками. Для кожного з них програма представляє своє окреме призначення, адже сценарії використання у них різні. Проте у центрі виступає покупець, адже саме йому потрібна в підсумку участь кожного окремого типу користувачів у одній програмі. Для покупця програма призначена для надання можливостей йому знайти меблі, які вже були у використанні і які готові продати їх власники, прокомунікувати з продавцями і досягти угоди, знайти майстра і виконати реставрацію меблів після придбання, якщо така необхідність є, та виконати перевезення в найширшому вигляді за ланцюжком «продавець – майстер – покупець».

Набір мінімальних технічних засобів, що мають забезпечуватися на сервері для підтримки роботи вебдодатку, включає:

- процесор не менше ніж Хеон сімейства 3000 або Core 2 Duo;
- оперативна пам'ять обсягом у 4 Гб;
- простір на жорсткому диску з доступним вільним обсягом не менше ніж 3 Гб, але для можливості збереження файлів зображень бажано не менше 20 Гб;
- мережна карта і підключення до мереж Інтернет.

4.2 Експлуатація програми

Експлуатація програми визначається зі сторони користувача тим, яку роль він має. Ролі були визначені у розділі 2. Тобто користувач або є неавторизованим, або після авторизації стає продавцем (рис. 4.1), покупцем, майстром або перевізником. В залежності від того, в якій ролі він

зареєструвався. Якщо він не зареєструвався, то зі сторінки авторизації є відповідне посилання.



Рисунок 4.1 – Робота продавця з програмою

Продавець має доступ до всіх власних оголошень через меню «Мої меблі». З цієї сторінки він може перейти до створення нового оголошення кнопкою «Виставити меблі». Також це можна зробити меню «Продати». Оцінити покупців продавець може через меню «Відгуки». Перейти до

відповідей на повідомлення продавець може через меню «Комунікація», де представляється перелік всіх користувачів, з якими він контактував.

На сторінці власних оголошень продавець може обрати стани оголошень, які його цікавлять. Стандартно відображаються всі оголошення. Кнопками «У продаж», «Продані», «Скасовані» можна відобразити тільки ті оголошення, які відповідають тільки вказаному назвою кнопки стану.

За кожним оголошенням продавець може перейти до отриманих на нього заявок за відповідною кнопкою. Кнопкою «Редагувати» продавець може розпочати внесення змін до оголошення щодо набору меблів. Якщо натиснути на зображення набору, то можна перейти до його сторінки, а з неї відображається набір предметів у комплекті, звідки можна перейти до редагування потрібного предмету. Або перейти до додавання нового предмету в набір. Кнопкою «Скасувати» можна скасувати продаж меблів за оголошенням, яке знаходиться в продажі. Ця кнопка не відображається для проданих або вже скасованих меблів.

Для покупця доступне інше меню. Покупцю доступний швидкий пошук шляхом введення тексту, який буде шукатися в назві та описі старих меблів. За результатами пошуку можна відкрити відповідне оголошення (рис. 4.2). З цієї сторінки видно фотографію набору меблів, інформацію про продавця. За посиланням на імені продавця можна перейти до сторінки з інформацією про нього, меблями, які він продає, відгуками. Кнопкою «написати» можна написати повідомлення цьому продавцю. У лівому нижньому куті фотографії відображається назва виробника. Під фотографією кнопка «зберегти» дозволяє зберегти меблі для повторного перегляду. Якщо вони вже збережені, відображається кнопка «вилучити». Кнопка «купити» дозволяє перейти до створення заявки на покупку меблів, звідки можна встановити власну вартість за потреби.

За набором відображається вся інформація, що його характеризує. Кнопка «Створити план реставрації» дозволяє створити план і внести потрібні роботи з реставрації. Зі сторінки плану можна перейти до пошуку

майстра в цьому місті. Також з меню «Майстри» можна переглянути найкращих за середньою оцінкою майстрів у обраному місті.

Пошук

ЗАЯВКИ

РЕСТАВРАЦІЇ

ЗБЕРЕЖЕНІ

ПОВНИЙ ПОШУК

МЕБЛІ

МАЙСТРИ

ПЕРЕВІЗНИКИ

ПРОФІЛЬ

ВИЙТИ

Вишуканий комплект

Продавець: Царичан Петро написати



Зберегти

Вироблено: 2020
Країна: Італія
Створено: 15.05.2023
Оновлено востаннє: 17.05.2023
До відправлення: 15.05.2023
Забрати до: 30.05.2023
Місто: Запоріжжя, Україна

Створити план реставрації

Купити

Стиль: класичний
Простір: вітальня
Категорії: набір м'яких меблів
Категорія стану: прекрасний
Опис стану:
без видимих і відчутних пошкоджень,
меблі використовувались дуже охайно
Опис:
Комплект фірмових італійських меблів
для вітальні. Висока якість та класичний
дизайн.

Рисунок 4.2 – Робота покупця з програмою при перегляді оголошення

Нижче на сторінці оголошення відображається перелік всіх предметів, які входять до набору. Наприклад, крісло, диван. За кожним предметом виводиться його фотографія і інформація. Звідти можна при потребі оформити заявку на попку потрібного предмету меблів замість набору.

Меню «Повний пошук» дозволяє встановити значення для широкого набору характеристик. Меню «Реставрації» містить усі плани реставрацій, які створив покупець. Меню «Збережені» дозволяє переглянути перелік

збережених наборів. Меню «Меблі» дозволяє переглянути останні опубліковані оголошення.

Меню «Заявки» (рис. 4.3) дозволяє переглянути перелік оформлених покупцем заявок на покупку меблів. За кожною заявкою відображається назва набору або предмету меблів, інформація про продавця.

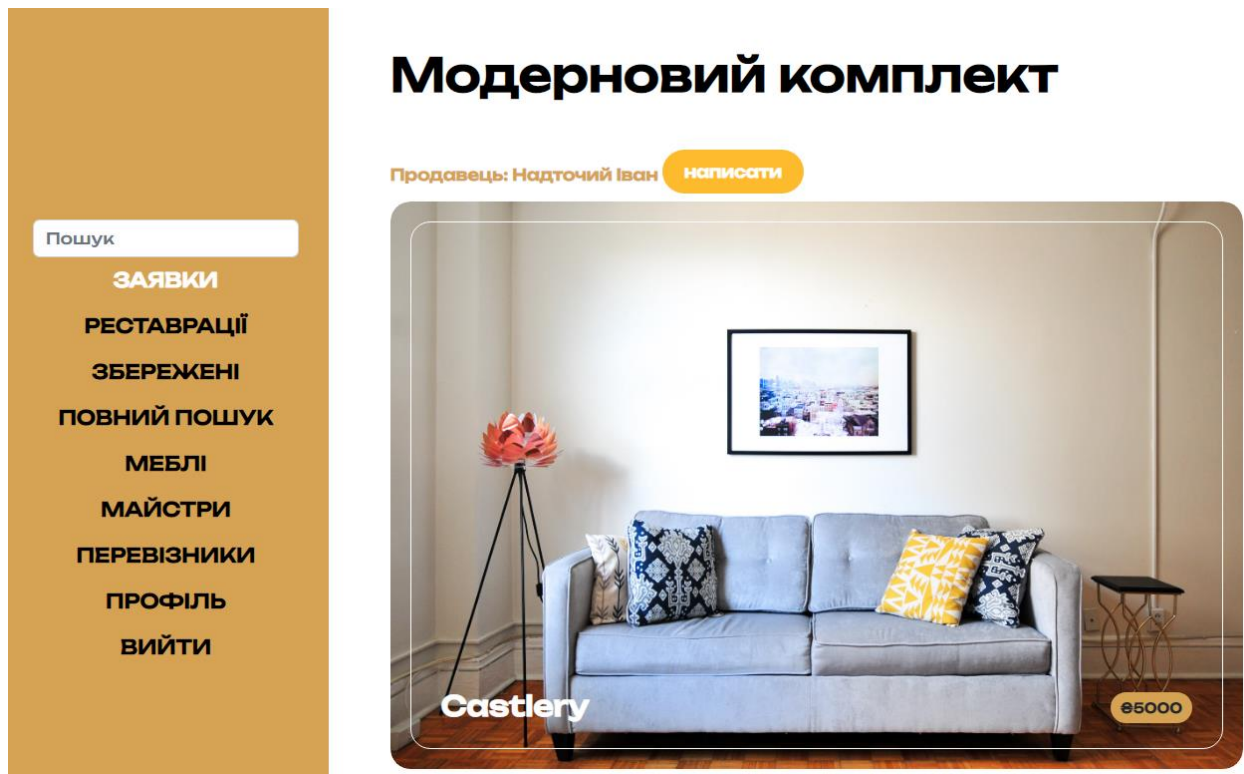


Рисунок 4.3 – Робота покупця з програмою щодо власних придбаних меблів

Нижче на сторінці відображається інформація про всі етапи роботи над заявкою (рис. 4.4). Якщо якийсь етап ще не виконаний, то відображається кнопка з можливістю розпочати його. Наприклад, кнопка «знайти перевізника». Відображаються секції «Придбання» з результатами взаємодії з продавцем, звідки можна, якщо це відповідний стан, виконати скасування заявки або сплату, «Реставрація», звідки можна скасувати, завершити роботу над заявкою, виконати оплату або передоплату, та потрібна кількість секцій «Доставка». Перейти до відправлення повідомлення продавцю можна з позиції вище на сторінці, а для всіх інших – з позицій нижче.

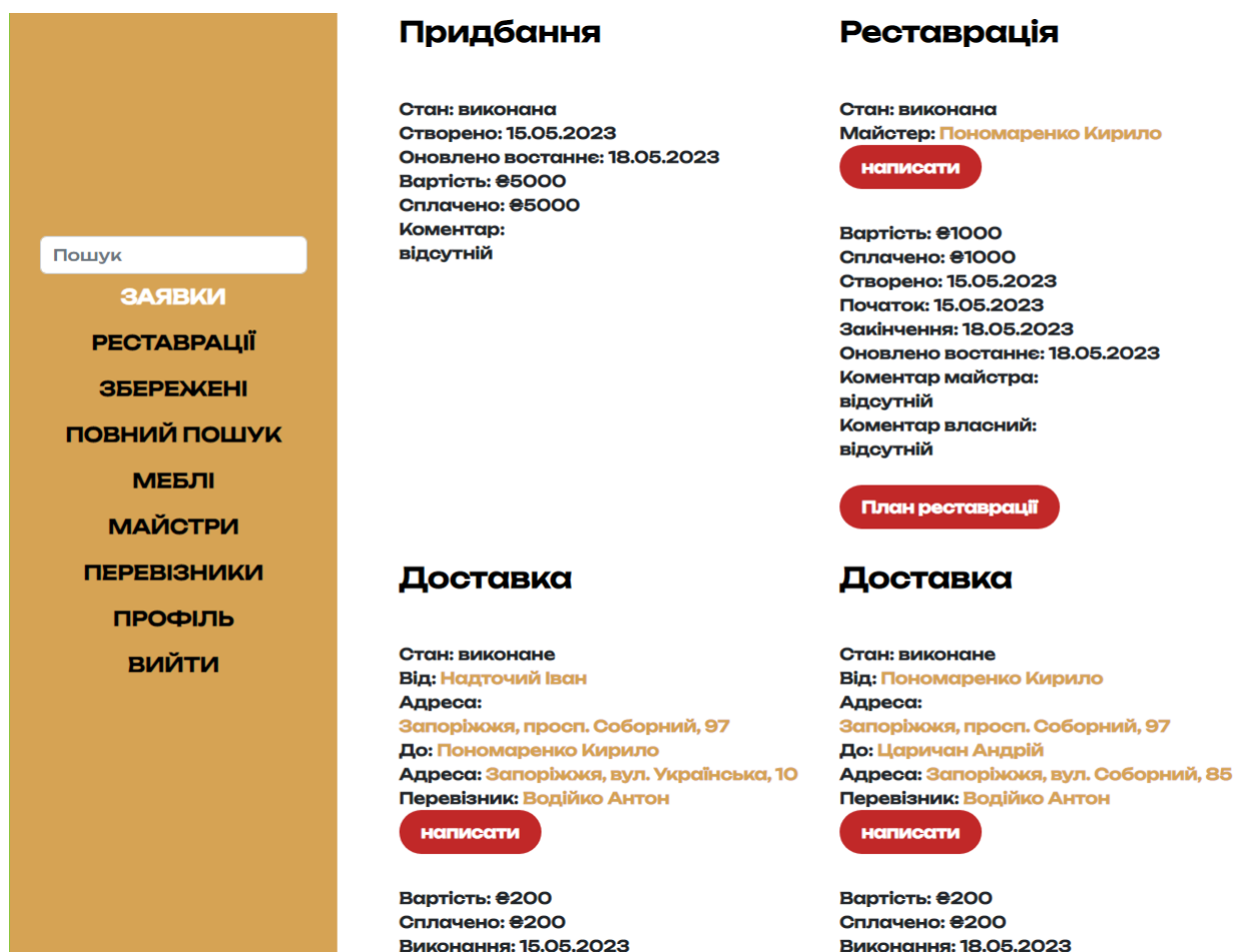


Рисунок 4.4 – Робота покупця з власною заявкою на покупку старих меблів

Для майстра доступне меню «Роботи», звідки він може визначити роботи, які він виконує. Якщо ці роботи співпадають з планом реставрації, а також співпадає місто, то він доступний за результатами пошуку покупцю.

У меню «Заявки» майстру відображаються всі подані заявки, починаючи з останньої. За заявкою відображається назва набору або предмету меблів, звідки можна перейти на сторінку з інформацією про них, інформація про продавця, звідки майстер також може написати продавцю, інформація про час створення заявки на реставрацію, час останнього оновлення та сам план з вартістю робіт. Якщо заявка передоплачена (рис. 4.5), то майстер може вносити вартість за кожною позицією. Сума рахується автоматично. Майстер може внести коментар. Після внесення цих даних потрібно натиснути на кнопку «оцінити». У випадку виконаної заявки відображаються тільки результуючі дані (рис. 4.6).

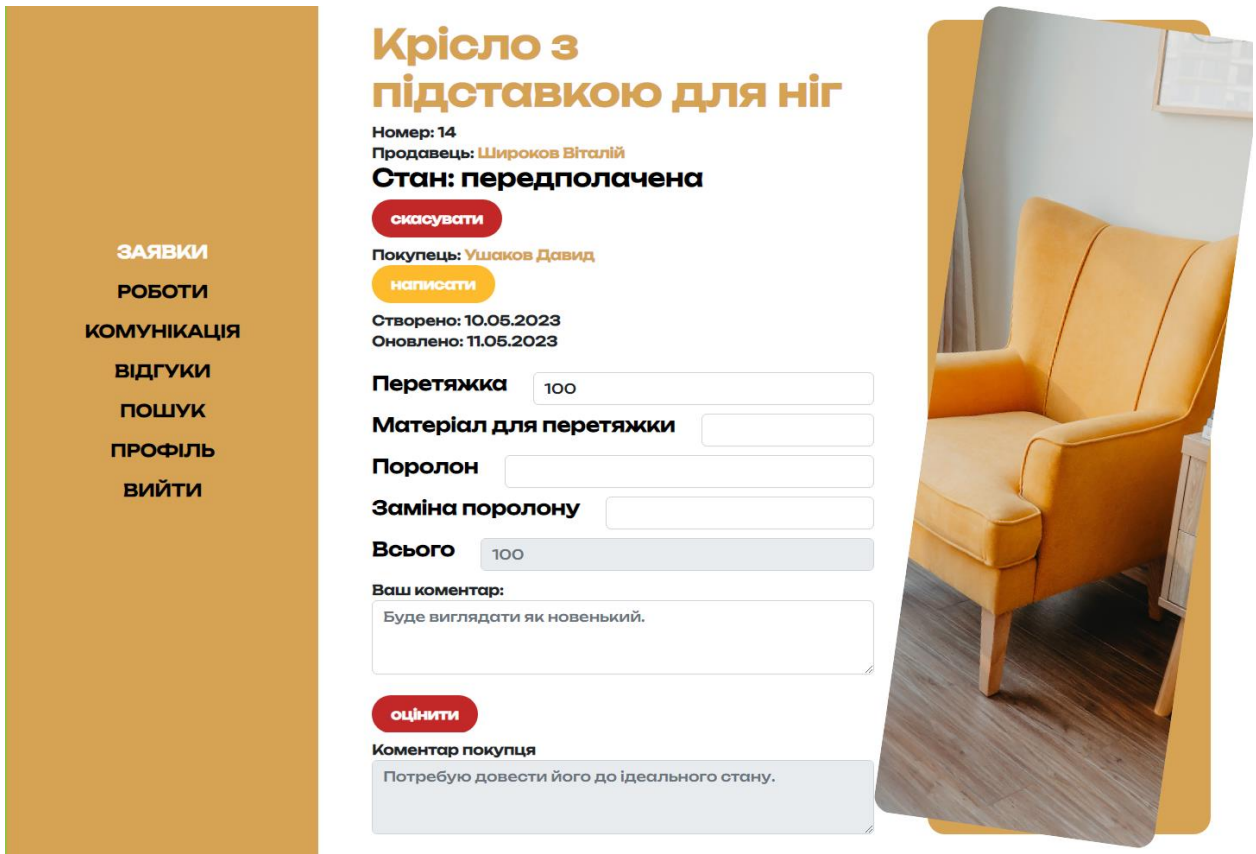


Рисунок 4.5 – Робота майстра з програмою на етапі передоплаченої заявки

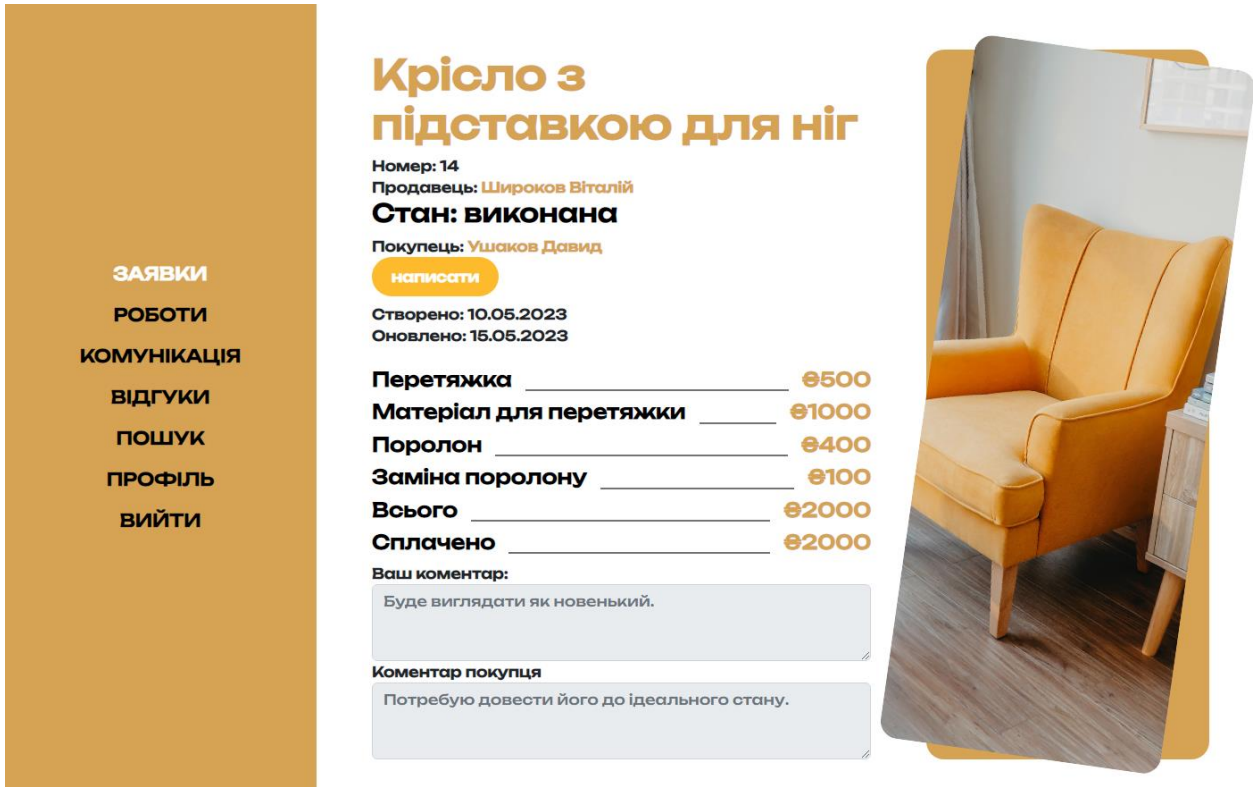


Рисунок 4.6 – Робота майстра з програмою на етапі виконаної заявки

Аналогічно організована робота перевізника з програмою. Через меню «Графік» йому доступний перелік сплачених обслуговувань перевезень за датами. Через меню «Умови» він може вносити тарифи. Через меню «Заявки» він може керувати заявками на перевезення.

Неавторизований користувач має меню «Пошук», «Меблі», «Майстри», «Перевізники», які були описані вище для покупця.

4.3 Тестування програми

Усі функціональні вимоги, які мали бути забезпечені реалізацією програми, були перевірені на основі тестування. Для цього було створено таблицю з листом-перевіркою (табл. 4.1). До неї було занесено усі сценарії, представлені в розділі 2, з деталізацією, де один сценарій включав інші. Кожен сценарій було протестовано: у випадку успішного виконання до таблиці вносився «+», а у випадку виявлення проблеми – «-».

Таблиця 4.1 – Тестування програми

Функція	Неавторизований користувач	Покупець	Продавець	Майстер	Перевізнак
1	2	3	4	5	6
Пошук старих меблів	+	+	+	+	+
Перегляд оголошення про продаж старих меблів	+	+	+	+	+
Перегляд найсвіжіших оголошень про продаж старих меблів	+	+	+	+	+
Перегляд найкращих майстрів	+	+	+	+	+

Продовження таблиці 4.1

1	2	3	4	5	6
Перегляд найкращих перевізників	+	+	+	+	+
Перегляд заявок на покупку за власним набором старих меблів			+		
Виставлення власних старих меблів на продаж			+		
Перегляд переліку наборів власних старих меблів			+		
Скасування продажу власних старих меблів			+		
Редагування оголошення про продаж власних старих меблів			+		
Збереження старих меблів для повторного перегляду		+			
Вилучення старих меблів з повторного перегляду		+			
Перегляд переліку збережених старих меблів		+			
Оформлення заявки на придбання старих меблів		+			
Скасування заявки на придбання старих меблів		+	+		
Авторизація	+	+	+	+	+
Реєстрація	+	+	+	+	+
Перегляд повної інформації про перевізника	+	+	+	+	+
Перегляд повної інформації про майстра	+	+	+	+	+
Перегляд повної інформації про продавця	+	+	+	+	+
Перегляд повної інформації про покупця	+	+	+	+	+
Перегляд старих меблів, виставлених на продаж за продавцем	+	+	+	+	+
Перегляд графіку обслуговувань доставок					+
Перегляд всіх заявок на обслуговування доставок					+
Узгодження виконання доставки					+
Скасування виконання доставки		+			+
Підтвердження початку доставки					+
Підтвердження виконання доставки					+

Продовження таблиці 4.1

1	2	3	4	5	6
Повернення за обслуговуванням доставки					+
Створення заявки на обслуговування доставки		+			
Оплата доставки меблів		+			
Оскарження доставки меблів		+			
Визначення умов доставки					+
Перегляд історії зміни вартості за власним набором меблів			+		
Відправлення повідомлення		+	+	+	+
Читання повідомлень		+	+	+	+
Оцінювання результатів взаємодії		+	+	+	+
Перегляд повного комплексу дій над придбаними меблями		+			
Підтвердження заявки на покупку старих меблів			+		
Підтвердження відправлення старих меблів за заявкою на покупку			+		
Оплата заявки на покупку старих меблів		+			
Перегляд власних заявок на придбання старих меблів		+			
Створення плану реставраційних робіт старих меблів		+			
Редагування плану реставраційних робіт старих меблів		+			
Оформлення заявки на оцінювання плану реставраційних робіт		+			
Оцінювання реставраційних робіт за заявкою				+	
Скасування заявки на реставраційні роботи		+		+	
Завершення заявки на реставрацію		+			
Оскарження заявки на реставрацію		+			
Перегляд власних заявок на реставрацію				+	

Кінець таблиці 4.1

1	2	3	4	5	6
Передоплата заявки на реставрацію меблів		+			
Оплата заявки на реставрацію меблів		+			
Підтвердження початку виконання реставраційних робіт за заявкою				+	
Повернення заявки на виконання реставраційних робіт				+	
Завершення виконання реставраційних робіт за заявкою				+	

4.4 Висновки за розділом 4

Реалізовані сценарії роботи покупця як авторизованого, так і неавторизованого, продавця, майстра та перевізника були описані і представлені, а також протестовані, що було успішно завершено.

ВИСНОВКИ

У дипломній кваліфікаційній роботі розроблено ПЗ комплексної підтримки продажу старих меблів для підвищення зручності придбання меблів, які були у використанні, через інтеграцію з реставрацією. Для цього було виконано огляд подібних рішень, що дозволило підтвердити унікальність саме комплексного підходу з об'єднанням не тільки покупців з продавцями, але і з майстрами, і з перевізниками. Таке рішення є корисним для того, щоб покупець у підсумку отримав меблі, які можна одразу встановити у власному приміщенні і використовувати, при цьому маючи всі можливості, щоб оцінити, на скільки такий варіант може бути придатним для конкретного покупця. Для цього покупець може запропонувати власну вартість придбання старих меблів, отримати оцінку від майстра щодо вартості реставраційних робіт, попередньо оцінити вартість доставки.

Для реалізації програми було використано набір технологій, що включають мову Python, фреймворки Django і Bootstrap, мови HTML, CSS, JavaScript, технологію AJAX, систему керування базами даних PostgreSQL. Усі вони в підсумку сприяли швидшій розробці та підтримці на актуальному технологічному рівні тієї концепції, яка була закладена в роботу. Перед реалізацією для цього було виконано докладний аналіз, проектування на основі діаграм прецедентів, схеми БД, що було в подальшому підтримано процесом реалізації з використанням діаграм станів для вираження послідовності роботи, закладеної у процес придбання меблів з реставрацією.

Розроблена програма дозволяє інтегрувати необхідні допоміжні складові для виконання реставрації меблів до процесу придбання старих меблів, що в підсумку сприяє збільшенню ринку реставраційних послуг, допомагаючи майстрам, та збільшенню зручності покупців. Для підтвердження працездатності програми було виконано тестування, що зафіксувало успішну реалізацію всіх запланованих функцій.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Off-the-Shelf Second-Hand Furniture Market Size, Analysis, Report to 2030 [Electronic resource]. – Access mode : [https://straitsresearch.com/report/off-the-shelf-second-hand-furniture-market#:~:text=The%20global%20off%2Dthe%2Dshelf,period%20\(2022%2D2030\).](https://straitsresearch.com/report/off-the-shelf-second-hand-furniture-market#:~:text=The%20global%20off%2Dthe%2Dshelf,period%20(2022%2D2030).)
2. Off-the-Shelf Second Hand Furniture Market Growth, Trends and Industry Forecast 2023-2028 with Top Countries Data [Electronic resource]. – Access mode : <https://www.marketwatch.com/press-release/off-the-shelf-second-hand-furniture-market-growth-trends-and-industry-forecast-2023-2028-with-top-countries-data-2023-02-27>.
3. Off-the-Shelf Second Hand Furniture Market Size By Product Type, By Application, By Geographic Scope And Forecast [Electronic resource]. – Access mode : <https://www.verifiedmarketresearch.com/product/off-the-shelf-second-hand-furniture-market/>.
4. Kaiyo – Buy and Sell Used Furniture [Electronic resource]. – Access mode : <https://kaiyo.com/>.
5. Etsy Ukraine – Shop for handmade, vintage, custom, and unique gifts for everyone [Electronic resource]. – Access mode : <https://www.etsy.com/>.
6. Chairish. For Chic And Unique Homes [Electronic resource]. – Access mode : <https://www.chairish.com/>.
7. Valeris, M. 12 Best Thrift and Used Furniture Stores Online [Electronic resource] / M. Valeris. – Access mode : <https://www.goodhousekeeping.com/home-products/g42115733/best-used-furniture-stores/>.
8. Ernesti, J. Python 3 [Text] : The Comprehensive Guide to Hands-On Python Programming / Johannes Ernesti, Peter Kaiser. – Bonn : Rheinwerk Computing, 2022. – 1078 p.
9. Downey, A. Think Python [Text] : How to Think Like a Computer Scientist / A. Downey. – Sebastopol, USA : O'Reilly Media, 2016. – 289 p.

10. Mele, A. Django 4 By Example [Text] : Build powerful and reliable Python web applications from scratch / Antonio Mele, Bob Belderbos. – Birmingham : Packt Publishing, 2022. – 766 p.
11. Obe, R. PostgreSQL [Text] : Up and Running : A Practical Guide to the Advanced Open Source Database / Regina Obe, Leo Hsu. – Sebastopol, USA : O'Reilly Media, 2017. – 312 p.
12. Barber Shop Template [Electronic resource]. – Access mode : <https://templatemo.com/tm-585-barber-shop>.

ДОДАТОК А
Технічне завдання

Вступ

В умовах негативних економічних процесів одним з вирішень побутових проблем є придбання і використання меблів, які вже були у використанні раніше. Це є взаємовигідним процесом, адже є люди, які готові позбутися своїх меблів, а є люди, які готові придбати такі меблі, щоб зменшити власні витрати. Звичайно, що дошки оголошень є популярним засобом вирішення і цієї проблеми, однак дана проблема є складнішою за суттю: часто недостатньо знайти потрібний варіант і встановити його у себе вдома, а потрібні певні ремонтні або загалом реставраційні процедури. Потрібно знайти майстра, відремонтувати меблі, перевезти їх. Тож у розробці потрібно розглянути комплексний підхід до продажу і придбання старих меблів.

A.1 Підстави для розробки

Завдання на дипломну кваліфікаційну роботу сформульовано у відповідності з наказом № 87 від 4 квітня 2023 р. за Національним університетом «Запорізька політехніка», за яким було визначено тему «Програмне забезпечення комплексної підтримки продажу старих меблів».

A.2 Призначення розробки

Комплексна підтримка спрямовує програмне забезпечення на використання і продавцями, і покупцями, і майстрами, і перевізниками. Для кожного з них програма представляє своє окреме призначення, адже сценарії використання у них різні. Проте у центрі виступає покупець, адже саме йому потрібна в підсумку участь кожного окремого типу користувачів у одній програмі. Для покупця програма призначена для надання можливостей йому знайти меблі, які вже були у використанні і які готові продати їх власники,

прокомунікувати з продавцями і досягти угоди, знайти майстра і виконати реставрацію меблів після придбання, якщо така необхідність є, та виконати перевезення в найширшому вигляді за ланцюжком «продавець – майстер – покупець».

А.3 Основні вимоги до програми

А.3.1 Вимоги до функціональних характеристик

Забезпечення призначення програми виконується через реалізацію сценаріїв використання програми на основі наступних функцій:

- пошук старих меблів зі структуризацією за категоріями, стилем, типом простору тощо;
- перегляд найсвіжіших оголошень про продаж старих меблів за обраною категорією у заданому місті;
- керування заявками на покупку старих меблів;
- виставлення власних старих меблів на продаж;
- керування переліком власних старих меблів;
- збереження старих меблів для повторного перегляду та вилучення внесених;
- оформлення заявки на покупку старих меблів покупцем та її скасування;
- підтвердження, скасування та підтвердження відправлення старих меблів продавцем за заявкою на покупку;
- перегляд заявок на покупку за власним набором або предметом старих меблів;
- перегляд власних заявок на придбання старих меблів;
- створення і редагування плану реставраційних робіт старих меблів;
- оформлення заявки на оцінювання плану реставраційних робіт майстром;

- оцінювання реставраційних робіт майстром за заявкою або її скасування;
- завершення заявки на реставрацію;
- перегляд власних заявок на реставрацію майстром;
- підтвердження початку і завершення виконання реставраційних робіт за заявкою майстром;
- оплата заявки на реставрацію меблів;
- оплата доставки меблів;
- оплата заявки на покупку меблів;
- оскарження заявки на реставрацію;
- оскарження доставки меблів;
- перегляд найкращих майстрів;
- перегляд найкращих перевізників;
- перегляд покупцем повного комплексу дій над його набором або предметом старих меблів;
- оцінювання результатів взаємодії з покупцем, продавцем, майстром, перевізником;
- комунікація продавця, майстра і перевізника з покупцем;
- перегляд історії зміни вартості за власним набором меблів;
- узгодження, скасування та підтвердження виконання доставки перевізником;
- створення запиту на обслуговування доставки та його скасування;
- визначення умов доставки перевізником;
- перегляд всіх запитів на обслуговування доставок перевізником;
- перегляд графіку обслуговувань доставок перевізником;
- перегляд повної інформації про перевізника, включаючи умови доставки і відгуки;
- перегляд повної інформації про майстра, включаючи його реставраційні роботи і відгуки;
- перегляд повної інформації про продавця, включаючи відгуки;

- перегляд старих меблів, виставлених на продаж продавцем;
- перегляд повної інформації про покупця, включаючи відгуки;
- авторизація, реєстрація акаунтів майстра, перевізника, продавця, покупця;
- перегляд оголошення про продаж старих меблів.

A.3.2 Умови експлуатації

Експлуатація програми на сервері потребує підтримки, що включає навички роботи з БД, зокрема PostgreSQL, виконання міграцій для Django-додатків та внесення даних через адміністративний інтерфейс при необхідності розширення або зміни категорій, стилів або інших каталогів стандартних даних.

A.3.3 Вимоги до складу та параметрів технічних засобів

Набір мінімальних технічних засобів, що мають забезпечуватися на сервері для підтримки роботи вебдодатку, включає:

- процесор не менше ніж Xeon сімейства 3000 або Core 2 Duo;
- оперативна пам'ять обсягом у 4 Гб;
- простір на жорсткому диску з доступним вільним обсягом не менше ніж 3 Гб, але для можливості збереження файлів зображень бажано не менше 20 Гб;
- мережна карта і підключення до мереж Інтернет.

A.4 Порядок контролю та приймання

Для виконання дипломної кваліфікаційної роботи критичним є виконання визначеного завдання, що включає безпосередньо завдання на роботу і технічне завдання, за темою, яка зафіксована наказом, у строки, які

визначені завданням на роботу.

У підсумку виконується захист роботи за оприлюдненим наперед графіком перед екзаменаційною комісією.

ДОДАТОК Б
Текст програми

Б.1 Текст файла `views_furniture.py`

```

from django.shortcuts import get_object_or_404
from .models import Furnituregroup
from .models import Wishlist
from .models import Auser
from .models import Brand
from .models import Country
from .models import City
from .models import Spacetype
from .models import Category
from .models import Style
from .models import Furnitureitem
from .models import Pricehistory
from django.shortcuts import render
from django.shortcuts import redirect
from django.urls import reverse_lazy

import logging
logger = logging.getLogger(__name__)
from django.db.models import Q

def furniture_quick_search(request):
    furniture_groups = Furnituregroup.objects.filter(state
=
0).select_related("owner").prefetch_related("styles").prefetch_r
elated("categories").prefetch_related("spacetypes")
    fgrouptitle = self.request.GET.get('title')
    if fgrouptitle:
        furniture_groups =
furniture_groups.filter(Q(representedtitle__icontains =
fgrouptitle) | Q(description__icontains = fgrouptitle))
        return render(request, "ofurnsoft/search_results.html",
{"furniture_groups": furniture_groups })

import datetime
from django.views import View
from django.views.generic.list import ListView
from django.views.generic.edit import CreateView
from django.views.generic.edit import UpdateView
from django.contrib.auth.mixins import LoginRequiredMixin

def furniture_search(request):
    furniture_groups = Furnituregroup.objects.filter(state
=

```

```

0).select_related("owner").prefetch_related("styles").prefetch_r
elated("categories").prefetch_related("spacetypes")
    fgrouptitle = self.request.GET.get('title')
    if fgrouptitle:
        furniture_groups =
furniture_groups.filter(Q(representedtitle__icontains =
fgrouptitle) | Q(description__icontains = fgrouptitle))
    origin = self.request.GET.get('origin')
    if origin:
        origin = int(origin)
        origin_country = Country.objects.get(id = origin)
        furniture_groups = furniture_groups.filter(origin =
origin_country)
    buy_city_id = self.request.GET.get('city')
    if buy_city_id:
        buy_city_id = int(buy_city_id)
        buy_city_object = City.objects.get(id =
buy_city_id)
        furniture_groups = furniture_groups.filter(city_id
= buy_city_object)
    brand_id = self.request.GET.get('brand')
    if brand_id:
        brand_id = int(brand_id)
        brand_id_object = Brand.objects.get(id = brand_id)
        furniture_groups = furniture_groups.filter(brand_id
= brand_id_object)
    original = self.request.GET.get('original')
    if original:
        furniture_groups = furniture_groups.filter(original
= True)
    min_price = self.request.GET.get('min_price')
    if min_price:
        min_price = int(min_price)
        furniture_groups =
furniture_groups.filter(ownerprice__gte = min_price)
    min_conditiontype =
self.request.GET.get('min_conditiontype')
    max_conditiontype =
self.request.GET.get('max_conditiontype')
    if min_conditiontype:
        min_conditiontype = int(min_conditiontype)
        furniture_groups =
furniture_groups.filter(conditiontype__gte = min_conditiontype)
    if max_conditiontype:
        max_conditiontype = int(max_conditiontype)

```

```

        furniture_groups =
furniture_groups.filter(conditiontype__lte = max_conditiontype)
        max_price = self.request.GET.get('max_price')
        if max_price:
            max_price = int(max_price)
            furniture_groups =
furniture_groups.filter(ownerprice__lte = max_price)
            manufactur_start =
self.request.GET.get('manufactur_start')
            manufactur_end = self.request.GET.get('manufactur_end')
            if manufactur_start and not manufactur_end:
                manufactur_start = int(manufactur_start)
                furniture_groups =
furniture_groups.filter(manufacturedate__gte = manufactur_start)
            elif not manufactur_start and manufactur_end:
                manufactur_end = int(manufactur_end)
                furniture_groups =
furniture_groups.filter(manufactureperiod__lte = manufactur_end)
            elif manufactur_start and not manufactur_end:
                manufactur_start = int(manufactur_start)
                manufactur_end = int(manufactur_end)
                q1_object = Q(manufacturedate__gte =
manufactur_start)
                q1_object.add(Q(manufacturedate__lte =
manufactur_end), Q.AND)
                q2_object = Q(manufactureperiod__gte =
manufactur_start)
                q2_object.add(Q(manufactureperiod__lte =
manufactur_end), Q.AND)
                q3_object = Q(manufacturedate__lte =
manufactur_start)
                q3_object.add(Q(manufactureperiod__gte =
manufactur_end), Q.AND)
                q4_object = Q(manufacturedate__gte =
manufactur_start)
                q4_object.add(Q(manufactureperiod__lte =
manufactur_end), Q.AND)
                q1_object.add(q2, Q.OR)
                q1_object.add(q3, Q.OR)
                q1_object.add(q4, Q.OR)
                furniture_groups = furniture_groups.filter(q1)
            deliver_start = self.request.GET.get('deliv_start')
            if deliver_start:
                deliver_start = datetime.strptime(deliver_start,
'%d/%m/%y')

```

```

        furniture_groups =
furniture_groups.filter(departuredate__gte = deliver_start)
        deliver_end = self.request.GET.get('deliv_end')
        if deliver_end:
            deliver_end = datetime.strptime(deliver_end,
'%d/%m/%y')
        furniture_groups =
furniture_groups.filter(departuredate__lte = deliver_end)
        if deliver_start and not deliver_end:
            furniture_groups =
furniture_groups.filter(departuredate__gte = deliver_start)
        elif not deliver_start and deliver_end:
            furniture_groups =
furniture_groups.filter(maxdate__lte = deliver_end)
        elif deliver_start and not deliver_end:
            q1_object = Q(departuredate__gte = deliver_start)
            q1_object.add(Q(departuredate__lte = deliver_end),
Q.AND)

            q2_object = Q(maxdate__gte = deliver_start)
            q2_object.add(Q(maxdate__lte = deliver_end), Q.AND)
            q3_object = Q(departuredate__lte = deliver_start)
            q3_object.add(Q(maxdate__gte = deliver_end), Q.AND)
            q4_object = Q(departuredate__gte = deliver_start)
            q4_object.add(Q(maxdate__lte = deliver_end), Q.AND)
            q1_object.add(q2, Q.OR)
            q1_object.add(q3, Q.OR)
            q1_object.add(q4, Q.OR)
            furniture_groups = furniture_groups.filter(q1)
        spacetype_id = self.request.GET.get('spacetype')
        if spacetype_id:
            spacetype_id_object = Spacetype.objects.get(id =
brand_id)
            furniture_groups =
furniture_groups.filter(spacetypes = spacetype_id_object)
            style_id = self.request.GET.get('style')
            if style_id:
                style_id_object = Style.objects.get(id = style_id)
                furniture_groups = furniture_groups.filter(styles =
style_id_object)
            category_id = self.request.GET.get('category')
            if category_id:
                category_id_object = Category.objects.get(id =
category_id)
            furniture_groups =
furniture_groups.filter(categories = category_id_object)

```

```

        furniture_items =
Furnitureitem.objects.all().select_related("furnituregroup_id")
        furniture_items =
furniture_items.filter(furnituregroup_id__state = 0)
        itemed = False
        if fgrouptitle:
            itemed = True
            furniture_items =
furniture_items.filter(Q(representedtitle__icontains =
fgrouptitle) | Q(description__icontains = fgrouptitle))
        min_height = self.request.GET.get('min_height')
        max_height = self.request.GET.get('max_height')
        min_width = self.request.GET.get('min_width')
        max_width = self.request.GET.get('max_width')
        min_length = self.request.GET.get('min_length')
        max_length = self.request.GET.get('max_length')
        if min_height:
            itemed = True
            min_height = int(min_height)
            furniture_items =
furniture_items.filter(totalheight__gte = min_height)
            if max_height:
                itemed = True
                max_height = int(max_height)
                furniture_items =
furniture_items.filter(totalheight__lte = max_height)
            if min_width:
                itemed = True
                min_width = int(min_width)
                furniture_items =
furniture_items.filter(totalwidth__gte = min_width)
                if max_width:
                    itemed = True
                    max_width = int(max_width)
                    furniture_items =
furniture_items.filter(totalwidth__lte = max_width)
                if min_length:
                    itemed = True
                    min_length = int(min_length)
                    furniture_items =
furniture_items.filter(totallength__gte = min_length)
                    if max_length:
                        itemed = True
                        max_length = int(max_length)
                        furniture_items =
furniture_items.filter(totallength__lte = max_length)

```

```

        if min_conditiontype:
            itemed = True
            min_conditiontype = int(min_conditiontype)
            furniture_items =
furniture_items.filter(conditiontype__gte = min_conditiontype)
        if max_conditiontype:
            itemed = True
            max_conditiontype = int(max_conditiontype)
            furniture_items =
furniture_items.filter(conditiontype__lte = max_conditiontype)
        assembled = self.request.GET.get('assembled')
        if assembled:
            itemed = True
            furniture_items = furniture_items.filter(assembled
= True)

        min_quantity = self.request.GET.get('min_quantity')
        max_quantity = self.request.GET.get('max_quantity')
        if min_quantity:
            itemed = True
            min_quantity = int(min_quantity)
            furniture_items =
furniture_items.filter(quantity__gte = min_quantity)
        if max_quantity:
            itemed = True
            max_quantity = int(max_quantity)
            furniture_items =
furniture_items.filter(quantity__lte = max_quantity)
        maincolor = self.request.GET.get('maincolor')
        if maincolor:
            itemed = True
            maincolor = int(maincolor)
            furniture_items = furniture_items.filter(maincolor
= maincolor)

        min_weight = self.request.GET.get('min_weight')
        max_weight = self.request.GET.get('max_weight')
        if min_weight:
            itemed = True
            min_weight = int(min_weight)
            furniture_items =
furniture_items.filter(deliveryweight__gte = min_weight)
        if max_weight:
            itemed = True
            max_weight = int(max_weight)
            furniture_items =
furniture_items.filter(deliveryweight__lte = max_weight)
        if min_price:

```

```

        min_price = int(min_price)
        furniture_items =
furniture_items.filter(ownerprice__gte = min_price)
        if max_price:
            itemed = True
            max_price = int(max_price)
            furniture_items =
furniture_items.filter(ownerprice__lte = max_price)
            sorted = self.request.GET.get('sort')
            if sorted:
                furniture_groups = furniture_groups.order_by('-
owner__rating')
            else:
                furniture_groups = furniture_groups.order_by('-
ownerprice')
            furniture_groups = furniture_groups.values()
            if itemed:
                furniture_items =
furniture_items.exclude(furnituregroup_id__in =
furniture_groups)
                furniture_groups_itemed =
Furnituregroup.objects.filter(items__in = furniture_items)
                furniture_groups_itemed =
furniture_groups_itemed.values()
                from itertools import chain
                from operator import attrgetter
                if sorted:
                    furniture_groups =
sorted(chain(furniture_groups, furniture_groups_itemed), key =
attrgetter('owner__rating'))
                else:
                    furniture_groups =
sorted(chain(furniture_groups, furniture_groups_itemed), key =
attrgetter('ownerprice'))
                onlyitems = self.request.GET.get('only')
                if onlyitems:
                    return render(request,
"ofurnsoft/search_results.html", {"furniture_items":
furniture_items})
                else:
                    return render(request,
"ofurnsoft/search_results.html", {"furniture_groups":
furniture_groups})

```

```

from django.views.decorators.csrf import csrf_exempt
from django.utils.decorators import method_decorator

```

```

from django.db.utils import IntegrityError

@method_decorator(csrf_exempt, name='dispatch')
class FurnituregroupSaveView(LoginRequiredMixin, View):
    def post(self, request, pk):
        furnit_group = get_object_or_404(Furnituregroup, id
= pk)

        customer_user = get_object_or_404(Auser, user =
request.user)
        wish = Wishlist(customer = customer_user,
furnituregroup_id = furnit_group)
        try:
            wish.save()
        except IntegrityError:
            logger.info("Спроба зберегти набір меблів %d
для повторного перегляду, який вже було збережено користувачем
%d", furnit_group.id, request.user.id)
            return HttpResponse()

@method_decorator(csrf_exempt, name='dispatch')
class FurnituregroupSaveUnView(LoginRequiredMixin, View):
    def post(self, request, pk):
        furnit_group = get_object_or_404(Furnituregroup,
id=pk)

        try:
            customer_user = get_object_or_404(Auser, user =
request.user)

            wish = Wishlist.objects.get(customer =
customer_user, furnituregroup_id = furnit_group).delete()
        except Wishlist.DoesNotExist:
            logger.info("Спроба вилучити набір меблів %d з
повторного перегляду користувачем %d, але він вже відсутній",
furnit_group.id, request.user.id)
            return HttpResponse()

class SellerFurnituregroupsView(ListView):
    template_name = 'seller_furniture.html'
    model = Furnituregroup

    def get_queryset(self):
        seller = self.request.GET.get('seller')
        seller = Auser.objects.get(id = seller)
        furniture_groups =
Furnituregroup.objects.filter(owner = seller)
        furniture_groups = furniture_groups.filter(state =
0)

```

```

        furniture_groups = furniture_groups.order_by('-
creation')
        return furniture_groups

    def get_context_data(self, **kwargs):
        seller_furniture_context =
super(SellerFurnituregroupsView,
self).get_context_data(**kwargs)
        seller = self.request.GET.get('seller')
        seller_furniture_context['seller'] = seller
        return seller_furniture_context

class OwnFurnituregroupsView(ListView):
    template_name = 'own_furniture.html'
    model = Furnituregroup

    def get_queryset(self):
        seller = Auser.objects.get(user =
self.request.user.id)
        furniture_groups =
Furnituregroup.objects.filter(owner = seller)
        lookup_state = self.request.GET.get('state')
        if lookup_state:
            furniture_groups =
furniture_groups.filter(state = lookup_state)
        furniture_groups = furniture_groups.order_by('-
creation')
        return furniture_groups

    def get_context_data(self, **kwargs):
        seller_furniture_context =
super(SellerFurnituregroupsView,
self).get_context_data(**kwargs)
        lookup_state = self.request.GET.get('state')
        seller_furniture_context['state'] = lookup_state
        return seller_furniture_context

from ofurnsoft.forms import FurnituregroupForm
from ofurnsoft.forms import FurnitureitemForm

def create_furnituregroup(request):
    if request.method == "POST":
        furnituregroup_form =
FurnituregroupForm(request.POST)
        furnitureitem_form =
FurnitureitemForm(request.POST)

```

```

        if furnituregroup_form.is_valid() and
furnitureitem_form.is_valid():
            furnituregroup =
furnituregroup_form.save(commit = False)
            auser = Auser.objects.get(user = request.user)
            furnituregroup.owner = auser
            furnituregroup.state = 0
            furnituregroup.save()
            logger.info("Утворено набір меблів %d для
користувача %s", furnituregroup.pk, auser.user)
            furnitureitem = furnitureitem_form.save(commit
= False)
            furnitureitem.furnituregroup_id =
furnituregroup
            furnitureitem.save()
            logger.info("Утворено предмет набору меблів %d
для користувача %s", furnitureitem.pk, auser.user)
            furnituregroup_form.save_m2m()
            logger.info("Завершено збереження набору меблів
%d", furnituregroup.pk)
            return redirect(request.path_info)
        else:
            furnituregroup_form = FurnituregroupForm()
            furnitureitem_form = FurnitureitemForm()
            return render(request,
"ofurnsoft/create_furnituregroup.html", {"group_form":
furnituregroup_form, "item_form": furnitureitem_form })

class FurnitureitemCreateView(LoginRequiredMixin,
CreateView):
    model = Furnitureitem
    template_name = 'create_furnitureitem.html'

class FurnitureitemUpdateView(LoginRequiredMixin,
UpdateView):
    model = Furnitureitem
    template_name = 'update_furnitureitem.html'

class FurnituregroupUpdateView(LoginRequiredMixin,
UpdateView):
    model = Furnituregroup
    fields = ['representedtitle', 'manufactureperiod',
'manufacturedate', 'conditiontype', 'conditiondescription',
'origin', 'description', 'original', 'ownerprice', 'city_id',
'address', 'brand_id', 'maxdate', 'departuredate']
    template_name = 'update_furnituregroup.html'

```

```

    def dispatch(self, request, *args, **kwargs):
        furniture_group = self.get_object()
        seller = Auser.objects.get(user =
self.request.user)
        if furniture_group.owner != seller:
            return redirect(furniture_group)
        return super(FurnituregroupUpdateView,
self).dispatch(request, *args, **kwargs)

    def form_valid(self, form):
        form.instance.lastupdate = datetime.now()
        newprice = form.instance.ownerprice
        furniture_group = Furnituregroup.objects.get(id =
form.instance.pk)
        if newprice != furniture_group.ownerprice:
            history = Pricehistory(furnituregroup_id =
form.instance.pk, price = form.instance.ownerprice, creation =
datetime.now())
            try:
                history.save()
            except IntegrityError:
                logger.info("Спроба зберегти історію зміни
вартості набору меблів %d за вартістю %d", form.instance.pk,
form.instance.ownerprice)
                redirect_url = super(FurnituregroupUpdateView,
self).form_valid(form)
                return redirect_url

    def get_success_url(self):
        return reverse_lazy('furnituregroupview',
kwargs={'pk': self.object.pk})

class PricehistoryView(ListView):
    template_name = 'pricehistory.html'
    model = Pricehistory
    paginate_by = 10

    def get_queryset(self):
        furniture_group =
self.request.GET.get('furniture_group')
        furniture_group = Furnituregroup.objects.get(id =
furniture_group)
        history =
Pricehistory.objects.filter(furnituregroup_id = furniture_group)
        history = history.order_by('-creation')
        prices = history.values

```

```

        price = prices[0]
        price.price = furniture_group.ownerprice
        price.creation = furniture_group.lastupdate
        prices.append(price)
        return prices

    def get_context_data(self, **kwargs):
        furniture_group_context = super(PricehistoryView,
self).get_context_data(**kwargs)
        furniture_group =
self.request.GET.get('furniture_group')
        furniture_group_context['furniture_group'] =
furniture_group
        return furniture_group_context

    def customer_wishlist(request):
        customer = Auser.objects.get(user = self.request.user)
        wishlists = Wishlist.objects.filter(customer =
customer).order_by('-id')
        wishlists = wishlists.values()
        furniture_groups = []
        for wishlist in wishlists:
            furniture_groups.append(wishlist.furnituregroup_id)
        return render(request,
"ofurnsoft/customer_wishlist.html", {"furniture_groups":
furniture_groups })

    from django.views.decorators.http import
require_http_methods

    @require_http_methods(['POST'])
    def cancel_furnituregroup(request, id):
        furniture_group = Furnituregroup.objects.get(pk = id)
        furniture_group.state = 2
        furniture_group.save()
        furniture_items =
Furnitureitem.objects.filter(furnituregroup_id =
furniture_group)
        for furniture_item in furniture_items:
            furniture_item.state = 2
            furniture_item.save()
        return redirect('ownfurnituregroupsvew', pk =
request.user.id)

    @require_http_methods(['POST'])
    def cancel_furnituregroup(request, id):

```

```

        furniture_item = Furnitureitem.objects.get(pk = id)
        furniture_item.state = 2
        furniture_item.save()
        furniture_group = Furnituregroup.objects.get(pk =
furniture_item.furnituregroup_id.id)
        furniture_items =
Furnitureitem.objects.filter(furnituregroup_id =
furniture_group)
        ended = True
        for furniture_item in furniture_items:
            if furniture_item.state == 0:
                ended = False
        if ended:
            furniture_group.state = 2
            furniture_group.save()
        return redirect('ownfurnituregroupsview', pk =
request.user.id)

class LastFurnituregroupsView(ListView):
    template_name = 'last_furnituregroups.html'
    model = Furnituregroup

    def get_queryset(self):
        city = self.request.GET.get('city')
        category = self.request.GET.get('category')
        if city:
            city = int(city)
            city = City.objects.get(id = city)
        if category:
            category = int(category)
            category = Category.objects.get(id = category)
        furniture_groups =
Furnituregroup.objects.filter(categories = category, city_id =
city)

        furniture_groups = furniture_groups.filter(state =
0)

        furniture_groups = furniture_groups.order_by('-
creation')

        return furniture_groups

    def get_context_data(self, **kwargs):
        last_fggroups_view = super(LastFurnituregroupsView,
self).get_context_data(**kwargs)
        city = self.request.GET.get('city')
        last_fggroups_view ['city'] = city
        category = self.request.GET.get('category')

```

```

        last_fgroups_view ['category'] = category
        return last_fgroups_view

    def furnituregroup_detail(request, id):
        furniture_group = get_object_or_404(Furnituregroup, pk
= id).prefetch_related("items")
        return render(request,
"ofurnsoft/furnituregroup_detail.html", {"group":
furniture_group })

```

Б.2 Текст файла views_feedback.py

```

from django.shortcuts import get_object_or_404
from .models import Communication
from .models import Auser
from .models import Masterrenovation
from .models import Delivery
from .models import Application
from .models import Deliveryservice
from .models import Renovationplanapplication

import logging
logger = logging.getLogger(__name__)

from ofurnsoft.forms import CommunicationForm
from itertools import chain
from operator import attrgetter

def communication_detail(request, id):
    toauser = get_object_or_404(Auser, pk = id)
    fromauser = get_object_or_404(Auser, pk =
request.user.id)
    communications1 =
Communication.objects.filter(fromauser = fromauser, toauser =
toauser)
    communications2 = Communication.objects.filter(toauser
= fromauser, fromauser = toauser)
    communications = sorted(chain(communications1,
communications2), key = attrgetter('-creation'))
    if request.user.is_active and fromauser.role !=
toauser.role:
        if request.method == "POST":
            communication_form =
CommunicationForm(request.POST)
            if communication_form.is_valid():

```

```

        communication =
communication_form.save(commit = False)
        communication.fromauser = fromauser
        communication.save()
        logger.info("Created comment on Post %d for
user %s", post.pk, request.user)
        return redirect(request.path_info)
    else:
        communication_form = CommunicationForm()
    else:
        communication_form = None
    return render(request, "ofurnsoft/communication.html",
{"communications": communications, "communication_form":
communication_form })

def communications(request):
    fromauser = get_object_or_404(Auser, pk =
request.user.id)
    communications = Communication.objects.filter(fromauser
= fromauser)
    communications = communications.order_by('-
creation').values('toauser').distinct()
    return render(request, "ofurnsoft/communications.html",
{"communications": communications })

def auser_detail(request, id):
    auser = get_object_or_404(Auser, pk = id)
    feedbacks = Feedback.objects.filter(addressant = auser)
    feedbacks = feedbacks.order_by('-creation')
    if auser.role == 2:
        renovations =
Masterrenovation.objects.filter(master = auser)
        return render(request,
"ofurnsoft/communications.html", {"user": auser, "feedbacks":
feedbacks, "renovations": renovations })
    if auser.role == 3:
        deliveries = Delivery.objects.filter(carrier =
auser)
        return render(request,
"ofurnsoft/communications.html", {"user": auser, "feedbacks":
feedbacks, "deliveries": deliveries })
    return render(request, "ofurnsoft/communications.html",
{"user": auser, "feedbacks": feedbacks })

from ofurnsoft.forms import FeedbackForm

```

```

def create_feedback(request):
    auser = Auser.objects.get(user = request.user)
    touser = Auser.objects.get(user__id = id)
    check = auser == 0 and touser != 0
    check = check or auser >= 1 and auser <= 3 and touser
== 0

    if auser.role == 2:
        renapplication =
Renovationplanapplication.objects.filter(master = auser,
renovationplan_id__customer = touser, state__in = [2, 4, 7, 10,
12, 13]).order_by('-creation')[0]
        feedback = Feedback.objects.filter(owner = auser,
addressant = touser).order_by('-creation')[0]
        if feedback.creation >= renapplication.lastupdate:
            check = False
    if auser.role == 3:
        delivery =
Deliveryservice.objects.filter(application_id__customer =
touser, delivery_id__carrier = auser, state__in = [2, 4, 6, 8,
9]).order_by('-creation')[0]
        feedback = Feedback.objects.filter(owner = auser,
addressant = touser).order_by('-creation')[0]
        if feedback.creation >= delivery.completedate:
            check = False
    if auser.role == 1:
        application = Application.objects.filter(customer =
touser, furnitureitem_id__furnituregroup_id__owner = auser,
state__in = [2, 4, 5]).order_by('-creation')[0]
        feedback = Feedback.objects.filter(owner = auser,
addressant = touser).order_by('-creation')[0]
        if feedback.creation >= application.lastupdate:
            check = False
    if auser.role == 0 and touser.role == 2:
        renapplication =
Renovationplanapplication.objects.filter(master = touser,
renovationplan_id__customer = auser, state__in = [2, 4, 7, 10,
12, 13]).order_by('-creation')[0]
        feedback = Feedback.objects.filter(owner = auser,
addressant = touser).order_by('-creation')[0]
        if feedback.creation >= renapplication.lastupdate:
            check = False
    if auser.role == 0 and touser.role == 3:
        delivery =
Deliveryservice.objects.filter(application_id__customer = auser,
delivery_id__carrier = touser, state__in = [2, 4, 6, 8,
9]).order_by('-creation')[0]

```

```

        feedback = Feedback.objects.filter(owner = auser,
addressant = touser).order_by('-creation')[0]
        if feedback.creation >= delivery.completdate:
            check = False
        if auser.role == 0 and touser.role == 1:
            application = Application.objects.filter(customer =
auser, furnitureitem_id__furnituregroup_id__owner = touser,
state__in = [2, 4, 5]).order_by('-creation')[0]
            feedback = Feedback.objects.filter(owner = auser,
addressant = touser).order_by('-creation')[0]
            if feedback.creation >= application.lastupdate:
                check = False
        if request.method == "POST" and check:
            feedback_form = FeedbackForm(request.POST)
            if feedback_form.is_valid():
                feedback = feedback_form.save(commit = False)
                feedback.owner = auser
                feedback.save()
                logger.info("Утворено відгук від користувача %s
про користувача %s", feedback.addressant.user, auser.user)
                return redirect(request.path_info)
        elif check:
            feedback_form = FeedbackForm(request.POST)
        else:
            feedback_form = None
        return render(request,
"ofurnsoft/create_furnituregroup.html", {"feedback_form":
feedback_form })

def ready_feedbacks(request):
    auser = Auser.objects.get(user = request.user)
    for_feedbacks = []
    if auser.role == 2:
        renapplications =
Renovationplanapplication.objects.filter(master = auser,
state__in = [2, 4, 7, 10, 12, 13]).order_by('-
creation').values()
        for renapplication in renapplications:
            touser =
renapplication.renovationplan_id.customer
            feedbacks = Feedback.objects.filter(owner =
auser, addressant = touser).order_by('-creation')[0]
            if feedback.creation <
renapplication.lastupdate:
                customer_info = {}
                customer_info['deal'] = renapplication

```

```

        customer_info['customer'] = touser
        for_feedbacks.append(customer_info)
    if auser.role == 3:
        deliveries =
Deliveryservice.objects.filter(delivery_id__carrier = auser,
state__in = [2, 4, 6, 8, 9]).order_by('-creation').values()
        for delivery in deliveries:
            touser = delivery.application_id.customer
            feedback = Feedback.objects.filter(owner =
auser, addressant = touser).order_by('-creation')[0]
            if feedback.creation < delivery.completedate:
                customer_info = {}
                customer_info['deal'] = renapplication
                customer_info['customer'] = delivery
                for_feedbacks.append(customer_info)
    if auser.role == 1:
        applications =
Application.objects.filter(furnitureitem_id__furnituregroup_id__
owner = auser, state__in = [2, 4, 5]).order_by('-creation')
        for application in applications:
            touser = application.customer
            feedback = Feedback.objects.filter(owner =
auser, addressant = touser).order_by('-creation')[0]
            if feedback.creation < application.lastupdate:
                customer_info = {}
                customer_info['deal'] = application
                customer_info['customer'] = touser
                for_feedbacks.append(customer_info)
    if auser.role == 0 and touser.role == 2:
        renapplications =
Renovationplanapplication.objects.filter(renovationplan_id__cust
omer = auser, state__in = [2, 4, 7, 10, 12, 13]).order_by('-
creation')[0]
        for renapplication in renapplications:
            touser =
renapplication.renovationplan_id.customer
            feedback = Feedback.objects.filter(owner =
auser, addressant = touser).order_by('-creation')[0]
            if feedback.creation <
renapplication.lastupdate:
                customer_info = {}
                customer_info['deal'] = renapplication
                customer_info['customer'] = touser
                for_feedbacks.append(customer_info)
    if auser.role == 0 and touser.role == 3:

```

```

        deliveries =
Deliveryservice.objects.filter(application_id__customer = auser,
state__in = [2, 4, 6, 8, 9]).order_by('-creation')
        for delivery in deliveries:
            touser = delivery.application_id.customer
            feedback = Feedback.objects.filter(owner =
auser, addressant = touser).order_by('-creation')[0]
            if feedback.creation < delivery.completedate:
                customer_info = {}
                customer_info['deal'] = delivery
                customer_info['customer'] = touser
                for_feedbacks.append(customer_info)
        if auser.role == 0 and touser.role == 1:
            applications = Application.objects.filter(customer
= auser, state__in = [2, 4, 5]).order_by('-creation')
            for application in applications:
                touser = application.customer
                feedback = Feedback.objects.filter(owner =
auser, addressant = touser).order_by('-creation')[0]
                if feedback.creation >= application.lastupdate:
                    customer_info = {}
                    customer_info['deal'] = application
                    customer_info['customer'] = touser
                    for_feedbacks.append(customer_info)
        return render(request, "ofurnsoft/feedbacks.html",
{"feedbacks": for_feedbacks })

```

ДОДАТОК В
Слайди презентації

Національний університет «Запорізька політехніка»
кафедра програмних засобів

Дипломна кваліфікаційна робота бакалавра на тему:
«Програмне забезпечення комплексної підтримки
продажу старих меблів»

Виконав: студент групи КНТ-229

Давид УШАКОВ

Керівник: к.т.н., доцент

Валерій ЛЬОВКІН

2023



Рисунок В.1 – Слайд титульний

Об'єкт, предмет, мета роботи

Об'єкт роботи – процес продажу старих меблів.

Предмет роботи – програмне забезпечення продажу старих меблів.

Мета роботи – розробити програмне забезпечення комплексної підтримки продажу старих меблів для підвищення зручності придбання меблів, які були у використанні, через інтеграцію з реставрацією.

Рисунок В.2 – Мета роботи, об'єкт, предмет

Завдання роботи

- аналіз програмних додатків, придатних для продажу старих меблів;
- проектування програмного забезпечення комплексної підтримки продажу старих меблів;
- реалізація і тестування програми за вимогами для продажу старих меблів.

3

Рисунок В.3 – Завдання роботи

Аналіз програмних додатків для продажу старих меблів

Характеристика	Kaiyo	Etsy	Clairish	Розробка
Продаж і покупка старих меблів	Так	Так	Так	Так
Придбання старих меблів від продавців	Ні	Так	Так	Так
Оцінювання продавців	Ні	Так	Так	Так
Пропонування ціни покупцем	Ні	Ні	Так	Так
Реновація старих меблів майстром	Ні	Ні	Ні	Так

4

Рисунок В.4 – Аналіз програмних додатків для продажу старих меблів

Вибір мови програмування для розробки

Показник	Python	C#
Підтримувана ліцензія	Відкрита	Відкрита
Складність реалізації об'єктно-орієнтованої парадигми програмування	Нижча	Вища
Вебфреймворк з інтегрованим фреймворком для роботи з даними	Наявний	Наявний
Швидкість створення базових рішень	Вища	Нижча

Рисунок В.5 – Вибір мови програмування для розробки

Діаграми прецедентів

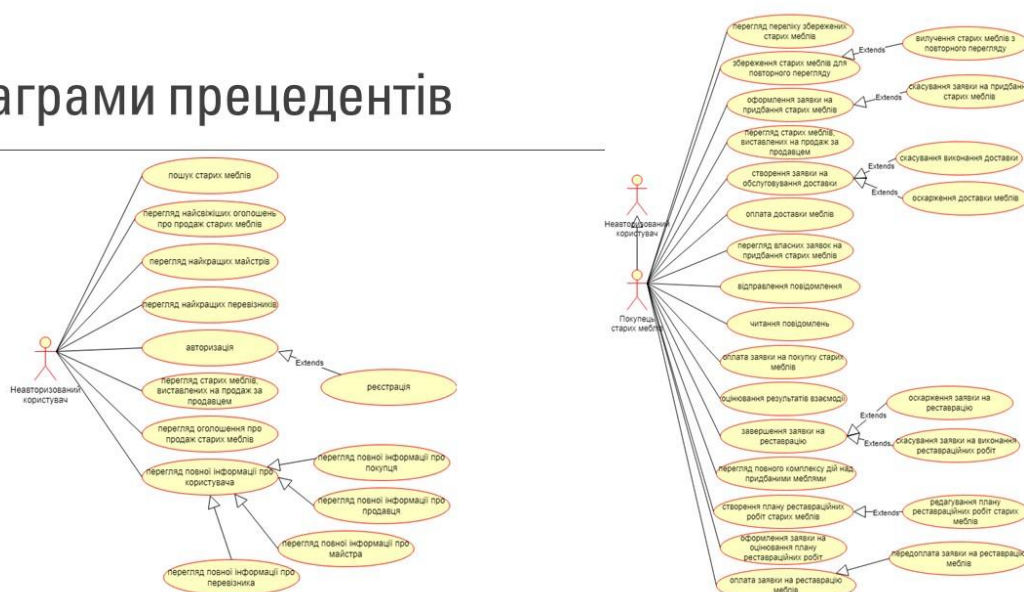


Рисунок В.6 – Діаграми прецедентів покупця старих меблів та неавторизованого користувача

Діаграми прецедентів

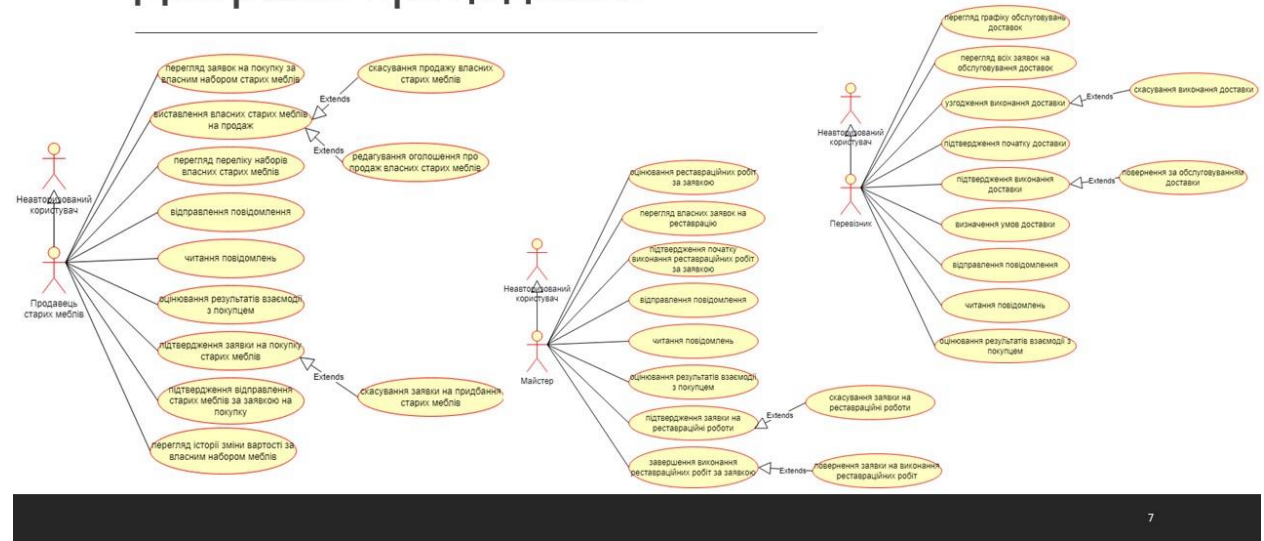


Рисунок В.7 – Діаграми прецедентів майстра, перевізника та продавця старих меблів

Схема бази даних

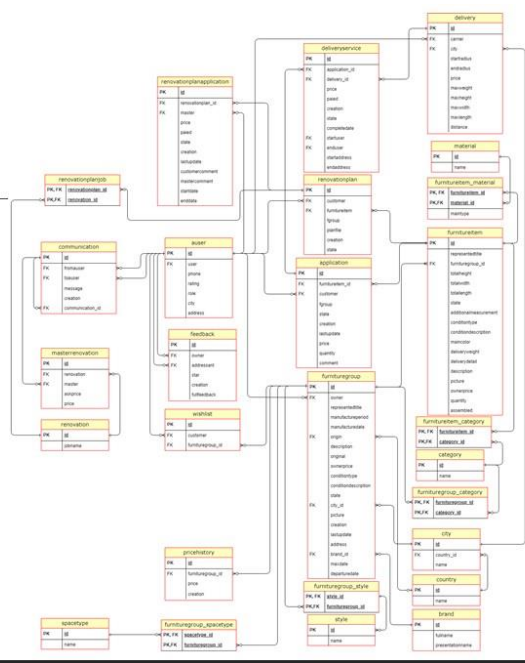


Рисунок В.8 – Схема БД

Діаграми станів

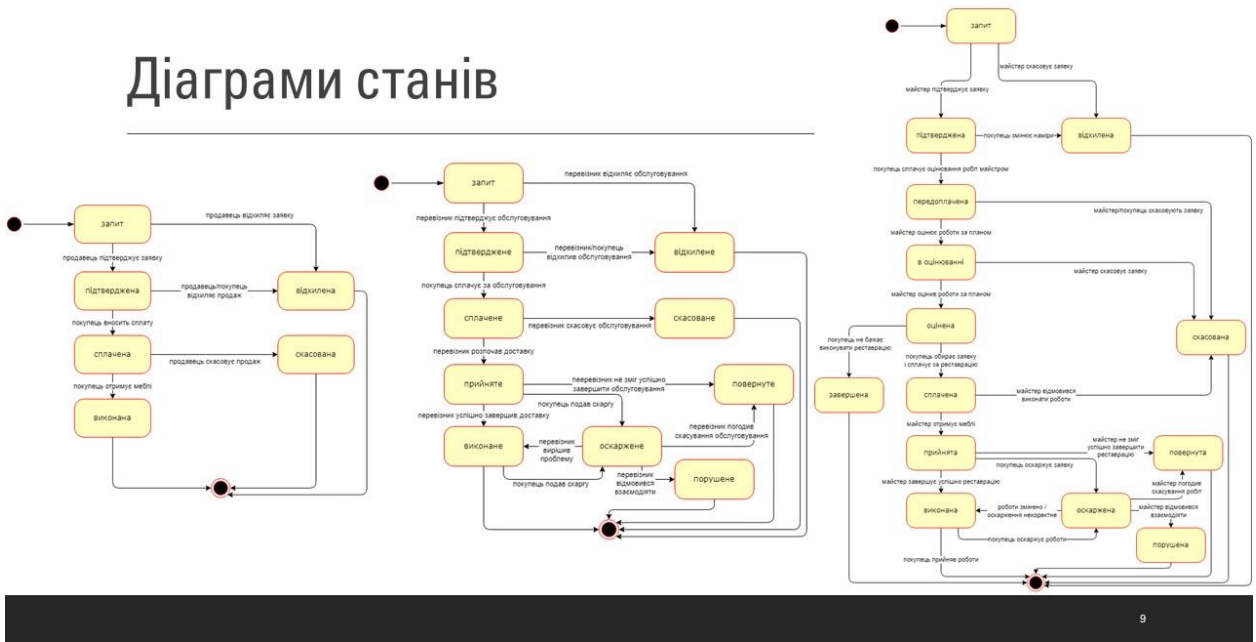


Рисунок В.9 – Діаграми станів

Робота з програмою



Рисунок В.10 – Робота з програмою: продавець старих меблів

Робота з програмою

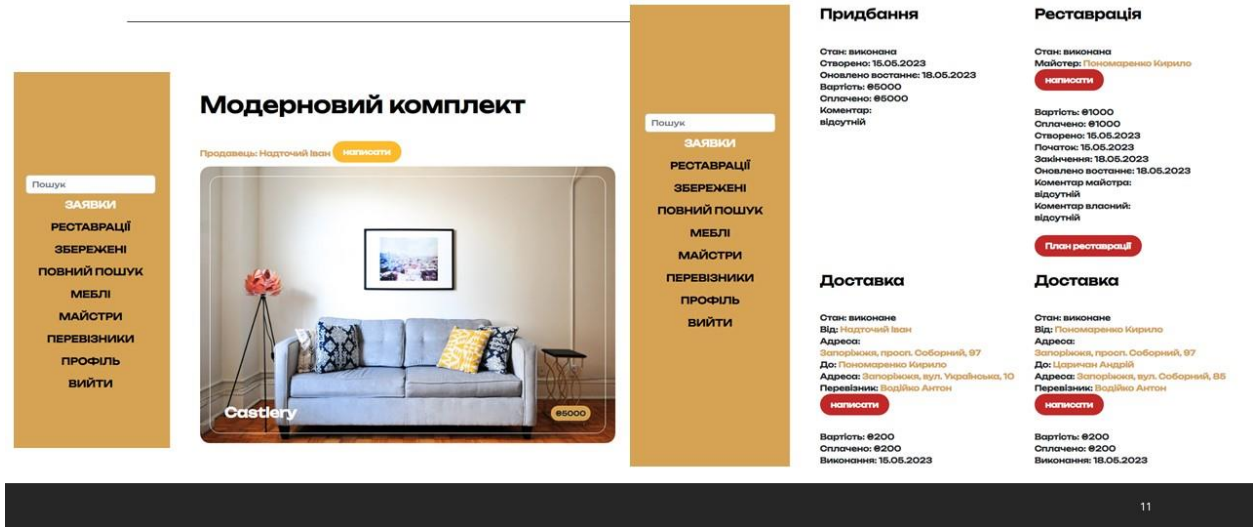


Рисунок В.11 – Робота з програмою: покупець старих меблів

Робота з програмою

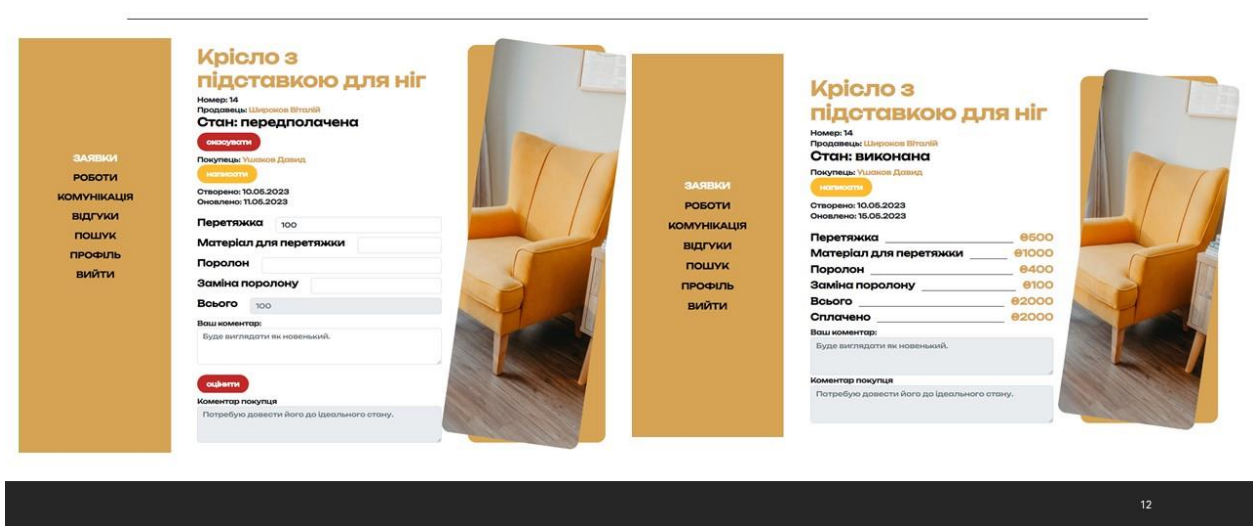


Рисунок В.12 – Робота з програмою: майстер

Тестування програми

Функція	Неавторизований користувач	Покупець	Продавець	Майстер	Перевізнак
1	2	3	4	5	6
Пошук старих меблів	+	+	+	+	+
Перегляд оголошення про продаж старих меблів	+	+	+	+	+
Перегляд найсвіжіших оголошень про продаж старих меблів	+	+	+	+	+
Перегляд найкращих майстрів	+	+	+	+	+
Передплата заявки на реставрацію меблів		+			
Оплата заявки на реставрацію меблів		+			
Підтвердження початку виконання реставраційних робіт за заявкою				+	
Повернення заявки на виконання реставраційних робіт				+	
Завершення виконання реставраційних робіт за заявкою				+	
Підтвердження виконання доставки					+
Узгодження виконання доставки					+
Скасування виконання доставки		+			+
Підтвердження початку доставки					+

1	2	3	4	5	6
Перегляд найкращих перевізників	+	+	+	+	+
Перегляд заявок на покупку за власним набором старих меблів		+			
Виставлення власних старих меблів на продаж		+			
Перегляд переліку наборів власних старих меблів		+			
Скасування продажу власних старих меблів		+			
Редагування оголошення про продаж власних старих меблів		+			
Збереження старих меблів для повторного перегляду		+			
Вилучення старих меблів з повторного перегляду		+			
Перегляд переліку збережених старих меблів		+			
Оформлення заявки на придбання старих меблів		+			
Скасування заявки на придбання старих меблів		+			
Авторизація	+	+	+	+	+
Рестрація	+	+	+	+	+
Перегляд повної інформації про перевізника	+	+	+	+	+
Перегляд повної інформації про майстра	+	+	+	+	+
Перегляд повної інформації про продавця	+	+	+	+	+
Перегляд повної інформації про покупця	+	+	+	+	+
Перегляд старих меблів, виставлених на продаж за продавцем	+	+	+	+	+
Перегляд графіку обслуговувань доставок				+	
Перегляд всіх заявок на обслуговування доставок				+	

1	2	3	4	5	6
Повернення за обслуговуванням доставки					+
Створення заявки на обслуговування доставки		+			
Оплата доставки меблів		+			
Оскарження доставки меблів		+			
Визначення умов доставки					+
Перегляд історії зміни вартості за власним набором меблів			+		
Відправлення повідомлення		+	+	+	+
Читання повідомлень		+	+	+	+
Оцінювання результатів взаємодії		+	+	+	+
Перегляд повного комплексу дій над придбаними меблями		+			
Підтвердження заявки на покупку старих меблів		+			
Підтвердження відправлення старих меблів за заявкою на покупку		+			
Оплата заявки на покупку старих меблів		+			
Перегляд власних заявок на придбання старих меблів		+			
Створення плану реставраційних робіт старих меблів		+			
Редагування плану реставраційних робіт старих меблів		+			
Оформлення заявки на оцінювання плану реставраційних робіт		+			
Оцінювання реставраційних робіт за заявкою					+
Скасування заявки на реставраційні роботи		+	+		
Завершення заявки на реставрацію		+			
Оскарження заявки на реставрацію		+			
Перегляд власних заявок на реставрацію					+

13

Рисунок В.13 – Тестування програми

Висновки

У дипломній кваліфікаційній роботі розроблено програмне забезпечення комплексної підтримки продажу старих меблів для підвищення зручності придбання меблів, які були у використанні, через інтеграцію з реставрацією.

Для реалізації програми було використано набір технологій, що включають мову Python, фреймворки Django і Bootstrap, мови HTML, CSS, JavaScript, технологію AJAX, систему керування базами даних PostgreSQL. Усі вони в підсумку сприяли швидшій розробці та підтримці на актуальному технологічному рівні тієї концепції, яка була закладена в роботу. Перед реалізацією для цього було виконано докладний аналіз, проєктування на основі діаграм прецедентів, схеми БД, що було в подальшому підтримано процесом реалізації з використанням діаграм станів для вираження послідовності роботи, закладеної у процес придбання меблів з реставрацією.

Розроблена програма дозволяє інтегрувати необхідні допоміжні складові для виконання реставрації меблів до процесу придбання старих меблів, що в підсумку сприяє збільшенню ринку реставраційних послуг, допомагаючи майстрам, та збільшенню зручності покупців. Для підтвердження працездатності програми було виконано тестування, що зафіксувало успішну реалізацію всіх запланованих функцій.

14

Рисунок В.14 – Висновки за роботою