Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБКА ВЕБЗАСТОСУНКУ «ДОШКА ОГОЛОШЕНЬ ДЛЯ ПРОДАЖУ ТОВАРІВ»
(тема)

Виконав:
здобувач 4 року навчання,
групи ІТІНФ-21-1
Іващенко А.О.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Тітова О.В.
(посада, прізвище, ініціали)

Допускається до захисту

Завідувач кафедри інформатики _____ Кобилін О. А.
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту

Кафедра Інформатики

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Іващенку Артему Олексійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка вебзастосунку «Дошка оголошень для продажу товарів»

затверджена наказом університету від 19 травня 2025 року № 381Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 27 травня 2025 р.
3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, дані інтернет-мережі

4. Перелік питань, що потрібно опрацювати в роботі
1. Аналіз поведінки користувача на платформах онлайн оголошень та C2C ринку.
2. Огляд аналогічних застосунків на ринку дошок онлайн оголошень.
3. Проєктування зручного потоку оформлення та створення оголошень.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) актуальність проблеми вибору дошки онлайн оголошень, постановка задачі, тестові зображення.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	07.04.2025	
2	Аналіз завдання, підбір літератури	08.04.25-10.04.25	
3	Аналіз літератури з досліджуваної проблеми	11.04.25-14.04.25	
4	Аналіз технічних засобів	15.04.25-20.04.25	
5	Розробка методу	21.04.25-27.04.25	
6	Програмна реалізація	04.05.25-20.05.25	
7	Оформлення пояснювальної записки	07.05.25-24.05.25	
8	Перевірка на нормоконтроль	21.05.25-01.06.25	
9	Перевірка на плагіат	21.05.25-01.06.25	
10	Рецензування	21.05.25-01.06.25	
11	Підготовка презентації та доповіді	21.05.25-18.06.25	
12	Занесення роботи в електронний архів	02.06.25-18.06.25	
	Попередній захист кваліфікаційної роботи	02.06.25-18.06.25	

Дата видачі завдання 7 квітня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ _____доц. Тітова О.В._____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 82 с., 10 табл., 41 рис., 31 джерело.

ОНЛАЙН ДОШКА ОГОЛОШЕНЬ, КУПІВЛЯ, ПРОДАЖ, Е-КОМЕРЦІЯ, ТОВАР, TYPESCRIPT, NESTJS, REACT, NEXTJS, SEQUELIZE, POSTGRESQL.

Об’єктом роботи є купівля-продаж товарів через онлайн-платформи.

Метою роботи є розробка онлайн-сервісу для пошуку та розміщення оголошень про продаж та купівлю товарів.

У ході роботи розглядаються реальні аналоги для маркетингу, а також особливості поведінки користувача при користуванні даними онлайн-платформами. Досліджується ефективність та доцільність системи як способу взаємодії між учасниками торговельного процесу.

У результаті роботи було реалізовано програмний застосунок з використанням сучасних технологій та інтуїтивним інтерфейсом, орієнтованим на простий користувацький досвід.

ONLINE CLASSIFIEDS MARKETPLACE, BUYING, SELLING, E-COMMERCE, PRODUCT, TYPESCRIPT, NESTJS, REACT, NEXTJS, SEQUELIZE, POSTGRESQL.

The aim of the work is to develop an online service for searching and posting advertisements for the sale and purchase of goods, as well as to effectively engage interested parties in the process.

The study explores real-world marketing analogues and examines user behavior when interacting with such online platforms. It also investigates the effectiveness and feasibility of the system as a means of interaction between participants in the trading process.

As a result, a software application was developed using modern technologies, featuring an intuitive interface focused on a simple and user-friendly experience.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Маркетплейс – онлайн-платформа, що виступає посередником між покупцями, які мають ціль знайти, порівняти та купити товар по певним критеріям, а продавці – виставити товар на продаж

E-commerce – вид діяльності в Інтернеті, спрямований на отримання прибутку від купівлі-продажу товарів або послуг за допомогою інтернет ресурсів

C2C – це вид створення продукту чи послуги, ключову роль якої відіграє взаємодія між користувачами, переймання досвіду з певними товарами та взаємна торгівля

Фреймворк – набір інструментів, бібліотек та правил, який використовується для створення застосунків

ВСТУП

Торгівля завжди була невід’ємною частиною людства упродовж всього свого існування. Завдяки розподілу поділу праці люди зосереджувались на своїй частині роботи і могли у разі потреби просто докупити те, що не виробляли самі.

У наш час торгівля стала доступною завдяки онлайн-сервісам. Якщо у давні часи люди торгували на обмеженій території, то зараз будь-який користувач Інтернету може побачити та купити майже будь-який існуючий товар, незалежно від місцезнаходження. Такі сервіси стали популярними завдяки своїй доступності та все більше людей користуються ними.

Актуальність роботи полягає у високій оптимізації торговельних процесів, таких як доступність у будь-якій частині світу, можливість залучатись у процес у будь-який час. Завдяки таким системам люди формують вторинний ринок, що економить час та ресурси для усіх сторін. Це дозволить покупцям швидко знаходити потрібні товари по заданим критеріям, а продавцям – економити на зусиллі пропозиції, керувати ціною та робити товарообіг доступним.

Дошки онлайн оголошень є доступним і водночас безпечним середовищем для товарообігу, оскільки такі системи виступають посередником між зацікавленими сторонами, контролюючи прозорість процесу, що знижує ризик шахрайства

Робота спрямована на кінцевого користувача, що хоче поліпшити процес купівлі-продажу. Для цього вивчається його поведінка задля створення простого та інтуїтивного інтерфейсу, так само як і для розширеного функціоналу платформи.

1 АНАЛІЗ МЕТОДІВ СТВОРЕННЯ ОНЛАЙН-СЕРВІСУ ДОШКИ ОГОЛОШЕНЬ ДЛЯ КУПІВЛІ І ПРОДАЖУ ТОВАРІВ

1.1 Роль дошок онлайн-оголошень у сучасному світі

Розвиток інтернету та e-commerce як його частини у 1990-х дав поштовх до створення вторинного ринку, де люди напряму могли напряму купувати та продавати товари без будь-яких перешкод. Такий вид діяльності став називатись «C2C», що розшифровується як «customer-to-customer» – сервіс від користувача до користувача. Тільки у 2023 більш ніж 250 мільйонів європейців зробили таких покупок онлайн [1].

У першу чергу, це гарний спосіб заробити додаткові кошти, позбувшись непотрібних речей. Аналогічно ніхто не забороняє шукати такі ж речі, дешевші, ніж аналоги у звичайних онлайн магазинах з новими товарами. Не дивно, чому така популярність таких сайтів виникла саме у Індії, оскільки велика доля населення живе за межею бідності. В основному в країні торгують одягом, що займає 72% усіх продажів на ринку. Але з іншої сторони, у розвинених країнах теж стає дедалі популярним такого роду товарообіг, наприклад у Північній Америці та Європі, де і виникли такі комерційні гіганти, як Amazon, Depop та Etsy [2]. Вони дають можливість користувачам виставляти на продаж, купляти, або ж рекламувати за певну плату іншим користувачам свої товари.

Підкріпило популярність таких платформ пандемія COVID 19. Явище дало великий поштовх для використання цифровізованих платформ в цілому, заміняючи при цьому звичайну торгівлю. Оскільки рух на вулицях був обмежений, люди стали шукати альтернативи знайти собі товари, не наражаючи себе на небезпеку. Так, фірми, що переважно працювали офлайн, стали міняти свою бізнес-модель і переходити на онлайн варіант, для того, щоб вижити на ринку. Таким сервісам стало легше знаходити покупців, а пізніше це все почало перетікати у вторинний ринок. Платформам стало

вигідно брати або маленьку комісію між користувачами та їх транзакціями, або ж повністю покривати ресурс рекламою [3].

Більш доступне входження в Інтернет завдяки використанню смартфонів дало поштовх людям об’єднуватись навколо цифрових платформ. Аналогічно можна провести паралель з бажанням цифровізуватись, ніж витрачати час на пошук бажаного в реальному житті. Очевидною перевагою електронних ресурсів є те, що такий підхід дає рівну можливість отримати бажаний продукт усім користувачам платформ, що є критичним у покупці, оскільки попит на будь-якого роду товари стає все більш гострим з популяризацію трендів.

Популярність C2C ресурсів принесли саме покоління Z. Для «зумерів» стала звична культура вторинного ринку, або більш зрозуміле поняття «second-hand» магазини. Найбільш поширена категорія таких покупок це одяг, оскільки мода стала популяризованою на сьогодення, і C2C сервіси підхоплюють цей тренд. Дохід від такого ринку прогнозується приблизно у 448 мільярдів доларів станом на 2025 рік.

Важливим чинником, що впливає на розвиток таких систем, – це філософія спільного користування. Ядром цієї філософії є думка, що використання товарів і послуг може бути ефективніше, якщо велика кількість людей буде спільно впроваджуватись у процес. Наприклад, ті ж самі дошки оголошень спонукають користувачів постійно переоцінювати ресурси і формувати культуру розумного споживання. Замість того, щоб просто викидати речі, є можливість їх продати людям, котрі цього реально потребують. Це підвищує ефективність використання ресурсів: час, гроші, зусилля, а також знижує кількість відходів та негативний вплив на довкілля. Але і тут є недоліки – все ж таки складніше довіряти стану вживаних речей, ніж нових, тому є ризик придбати пошкоджені або ж неякісні речі, що суттєво додає складність верифікувати інших користувачів та їх доброчесність. Всі ці фактори складують певний образ про те, як потрібно користуватись дошками оголошень та C2C платформами в цілому, тому

важливо враховувати поведінку користувача, його критерії підбору платформи та цілі при дизайні таких систем [4, 5].

1.2 Аналіз поведінки користувача під час користування дошкою онлайн оголошень

Онлайн-платформи оголошень мають значний вплив на поведінку користувачів, особливо на етапах, що передують купівлі або продажу. Пошук необхідного товару на таких платформах зазвичай йде за двома шляхами.

Перший передбачає ситуацію, коли споживач не має чітко сформульованої потреби. У цьому випадку, бажання щось придбати виникає під час перегляду наявних товарів. У такому випадку рекомендації та їх алгоритми напряму впливають на стимуляцію до покупок. Чим краще платформа розуміє, кому варто рекомендувати певного роду товари, тим більша ймовірність виникнення бажання щось замовити. Також грає роль кнопок «Зберегти оголошення», або ж показувати повторно користувачам вже те, що вони бачили – це все сприяє зацікавленості у платформі та її пропозиціях. Класичним у науковій сфері є дослідження, що більшість прибутку приносять саме користувачі, що повертаються на платформу, аніж ті, які зайшли вперше та імпульсивно щось купили [6]. Наприклад можливим інструментом може бути система нагадувань через пошту або push-повідомлення. Тому візуально приємний інтерфейс, налаштована категоризація, широка можливість досліджувати вебресурс, розвинений функціонал, безперечно, відіграють дуже важливу роль при створенні C2C платформи.

Інший випадок характеризується тим, чи має користувач певну потребу у певному товарі (наприклад «Купити iPhone 11 менш ніж за 16 тисяч гривень»). Вагітній жінці доцільніше буде замовити коляску для майбутньої дитини, ніж волейбольний м’яч чи дриль. Тому розглядаючи даний вид

поведінки користувача, критичним є розвиненість пошуку на сайті та різноманітна фільтрація товарів. Користувачі очікують швидко дістатись до бажаного товару, використовуючи ключові слова, категорії, цінові проміжки, локацію та інші атрибути. Очевидно, що повільність та неправильний пошук будуть тільки дратувати користувача, і, він, ймовірно піде шукати іншу платформу для свого повсякденного користування [7].

Існують також люди, що не користуються пошуком, а надають перевагу навігації. Наприклад, такі користувачі можуть переходити по великій кількості посилань без специфічного запиту. Вони переглядають оголошення з цікавості, або просто подивитись, які позиції доступні. Дуже критичним є момент урахування таких шаблонів поведінки задля кращої інтеграції користувацького досвіду при створенні дошки онлайн оголошень.

Коли покупець вирішує, чи варто відповідати на пропозицію, декілька ключових факторів грають роль.

По-перше, це якість оголошення. Після вибору потенційного товару, важливим фактором, що впливає на рішення споживача, є інформація про його характеристики та стан. Чіткий та повний опису, а також якість зображень, безпосередньо впливають на сприйняття товару потенційним покупцем. Замість опису товару «Старий велосипед, нормальний стан» можна сказати «Гірський велосипед Kinetic Crystal, рама 18 дюймів, колеса 29", алюмінієва рама, 21 передача, гальма дискові. Експлуатувався 2 роки, зберігався в приміщенні, є незначні подряпини на рамі. Повністю справний, ланцюг змащено, колеса не спущені». Детальна та зрозуміла інформація дозволяє користувачеві сформувати адекватне уявлення про стан товару, що, у свою чергу, позитивно впливає на його готовність до здійснення покупки. Тому є великий сенс при створенні дошки онлайн оголошень робити розгорнуті форми для заповнення найрізноманітнішої інформації, від опису товару до тегів, по яким можна знайти більше схожих позицій [8].

По-друге, ціна. Це одна з вирішальних метрик, яка впливає на вибір покупця. Зазвичай відвідувачі мають специфічний бюджет. Тому якщо товар

має занадто високу ціну, він може бути проігнорований. Багато C2C платформ мають фільтрацію та сортування через це, щоб користувач міг налаштовувати контент під власні потреби.

Критичним фактором є довіра та безпека. Користувачі йдуть до маркетплейсів та платформ, де вони відчувають себе в безпеці. Якщо вебсайти мають репутаційний функціонал, як от, рейтинги, то відвідувачі дійсно будуть покладатись на цю метрику. Принцип соціального затвердження заохочує нові позитивні угоди на основі позитивних старих [9]. Для цього може бути використаний функціонал підтверджених користувачів платформи за допомогою телефону або інших контактних даних – з'являється відповідна галочка біля профілю або напис «Підтверджений користувач». Це стимулює інших користувачів довіряти один одному, що підвищує шанси створення продуктивних взаємин. Однак, система має бути захищена від маніпуляцій з підставними відгуками. Ранній позитивний чи негативний відгук створює так званий «bandwagon effect» — ефект наслідування. Це варто брати до уваги при дизайні системи відгуків. У таблиці 1.1 показано основні фактори з конкретними прикладами функціоналу, які впливають на довіру при користуванні дошками онлайн оголошеннями [10].

Таблиця 1.1 – Фактори, які впливають на довіру

Фактор	**Опис ключової ідеї**	**Приклад функціоналу**
1	**2**	**3**
Репутація продавця/ відгуки	Рейтинги (зірочки) та коментарі від інших покупців показують, чи можна довіряти продавцю (чи він чесний, надійний, виконує обіцянки).	Після покупки можна поставити продавцю оцінку (1-5 зірок) і написати відгук. У профілі продавця видно його середній рейтинг та всі відгуки. Система також намагається виявити підроблені відгуки.

Продовження таблиці 1.1

1	2	3
Гарантії платформи	Правила та послуги самої платформи, які захищають і покупця, і продавця.	Функція «Безпечна оплата»: гроші покупця «заморожуються» платформою і передаються продавцю тільки після того, як покупець підтвердить отримання товару. Є кнопка «Відкрити спір», якщо щось пішло не так.
Безпечні платіжні механізми	Можливість безпечно оплатити карткою чи іншим способом через відомі платіжні системи	При оплаті видно значок замка (HTTPS). Платформа використовує відомі сервіси типу LiqPay або WayForPay. Дані картки надійно захищені і не зберігаються на сайті напряму.
Верифікація, ідентифікація продавця	Можливість для продавця підтвердити свою особу. Це зменшує анонімність і показує, що продавець серйозно налаштований.	У профілі продавця є розділ «Верифікація». Він може завантажити фото документів або пройти перевірку через BankID/Дію. Після успішної перевірки біля його імені з'являється значок «Перевірений продавець».
Чітка та оперативна комунікація	Можливість швидко і зручно спілкуватися з продавцем прямо на платформі (в чаті).	Вбудований чат на сайті, де видно історію листування, час повідомлень, чи прочитав їх продавець. Можна надсилати фото. Приходять сповіщення про нові повідомлення.

Продовження таблиці 1.1

1	2	3
Детальні та точні оголошення	Оголошення з детальним описом товару, його стану та якісними фотографіями. Це допомагає зрозуміти, що саме купляється, і уникнути розчарувань.	Продавець заповнює поля: назва, категорія, стан (новий/вживаний), ціна, детальний опис. Можна завантажити багато фото. Чим краще заповнене оголошення, тим вище довіра.
Історія транзакцій	Інформація про те, як довго продавець на платформі і скільки угод він вже успішно завершив. Це показує його досвід та надійність.	У профілі продавця видно дату реєстрації («На сайті з 2018 року») та кількість успішних продажів (наприклад, «50+ успішних угод»).

Ці фактори необхідно враховувати під час розроблення застосунка для того, щоб гарантувати не тільки технічну правильність виконання роботи, а й психологічну адаптованість продукту до користувачів.

1.3 Аналіз провідних аналогічних застосунків для купівлі та продажу товарів

Існує багато дошок онлайн оголошень, але було обрано три провідні платформи: OLX, Etsy та eBay. З них трьох специфікованою дошкою оголошень можна назвати тільки OLX, в той час інші розглянуті платформи мають більш широкий спектр застосування. Вони мають різні стратегії в сфері C2C, тому вибір впав саме на набір цих систем. OLX є прикладом платформи, орієнтованої саме на локалізовані оголошення, оскільки для населення різних країн є різні економічні та матеріальні потреби. Etsy

спеціалізується на виробах ручної роботи, вінтажних речах, та в цілому тісно пов’язана з творчістю, що створює специфічну нішу. eBay ж навпроти є прикладом глобального маркетплейсу, з величезною кількістю онлайн-аукціонів, що пропонує різні формати продажів [11].

1.3.1 OLX

Платформа OLX була заснована у 2006 році, зареєстрована у Нідерландах зі штаб-квартирою у Амстердамі. В Україні спочатку бренд називався «Slando», поки у 2014 році не був перейменований на OLX. На платформу припадало 40% усіх онлайн-замовлень у країні. OLX має козир у вигляді локалізації та широкому охопленні категорій. Система відображає традиційні офлайн C2C відносини, але з цифровізованим напрямком. Наприклад, на рисунку 1.1 наведено приклад типового оголошення.

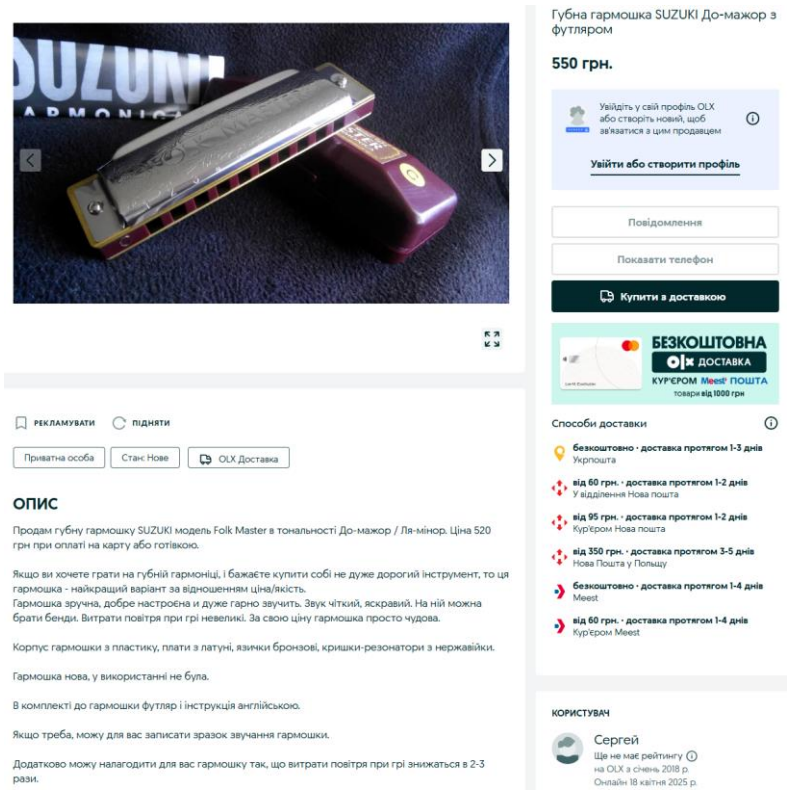

Рисунок 1.1 – Типове оголошення на OLX

Дошка оголошень має таку бізнес модель, що дозволяє розміщувати базові оголошення у багатьох категоріях безкоштовно, що сприяє залученню великої кількості користувачів. Але дохід генерується за допомогою преміальних онлайн оголошень, що у пошуці займають найперші місця. Важливим є зв'язок між покупцем і продавцем, що досягається за допомогою розвиненої системи чатів.

Особливу увагу варто звернути на OLX доставку, яка дозволяє замовляти товари онлайн через партнерські служби, такі як Укрпошта та Нова Пошта. Коли покупець сплачує товар, платформа виступає посередником і утримує кошти, поки замовник не отримає його. Тільки після огляду товару на партнерських службах пошти покупець підтверджує отримання, і лише тоді кошти перераховуються продавцю.

В цілому платформа робить зусилля з вертикалізації, тобто створює досвід для окремих категорій, наприклад для роботи чи нерухомості. Це дає змогу заходити на різноманітніші ринки та отримувати віддану аудиторію.

1.3.2 Etsy

Etsy виникла у результаті потреби ринку у спеціалізованому онлайн-майданчику для продажі товарів ручної роботи. Назва «Etsy» походить від італійського «etsi», що означає «о, так!», відображаючи захоплення від знахідки унікальних речей. Аудиторія специфічна, зазвичай це жінки та молодші покоління. Тому персоналізація є дуже важливою, оскільки велика частка припадає на продаж товарів, що є у єдиному екземплярі [13].

Продавці на Etsy мають можливість створювати власні «вітрини», або ж простіше магазини (рис. 1.2), де вони можуть представляти свій бренд. На відміну від OLX, лістинг тут не безкоштовний, тобто за кожне оголошення стягується плата у вигляді $0,20 USD. Платформа має свою платіжну систему, яка тісно інтегрована з Visa, MasterCard, та PayPal. Etsy пропонує

рекламу, так звану «offside ads». Це принцип, коли оголошення рекламується на зовнішніх вебсайтах, але продавець сплачує комісію у 12-15%. Це буває дуже зручно для термінових оголошень.

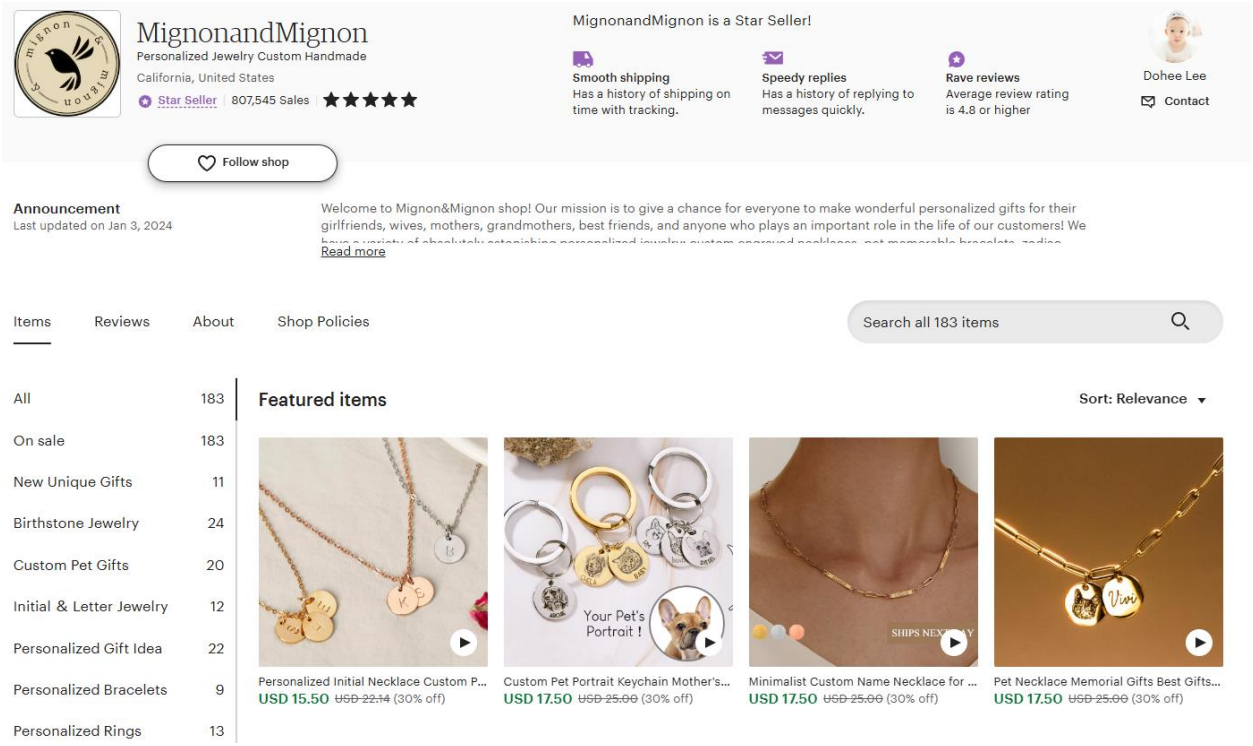

Рисунок 1.2 – Персоналізована «вітрина» на Etsy

Платформа залишається однією з найбезпечніших, оскільки вона використовує шифрування TLS, має спеціальну команду Trust and Safety, систему повідомлень про помилки та механізми для повідомлення користувачів про підозрілу активність.

1.3.3 eBay

eBay був заснований у 1995 році П’єром Омідьяром під назвою AuctionWeb. Хоча і сайт починався як аукціонний сайт, але потім на ринку стали відігравати все ж таки продажі з фіксованою ціною, тому довелось

компанії перекваліфікуватись і виходити на більш широкий ринок у 1998 році. Тобто два основні напрями платформи це аукціони та продаж товарів.

Аудиторія eBay різноманітна, тому не вийде виокремити певну категорію, яка домінує. Компанія має глобальний масштаб і працює на 190 ринках світу [14].

Особливістю платформи є, як зазначалось раніше, аукціони (рис. 1.3).

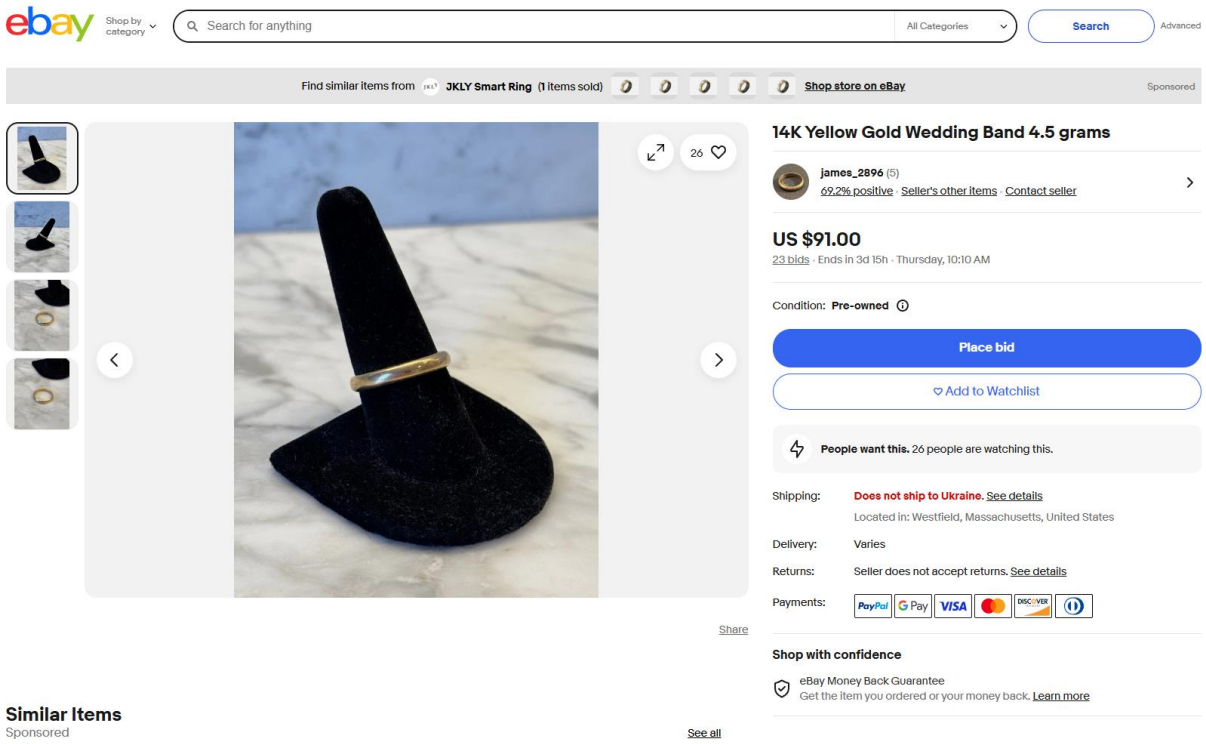

Рисунок 1.3 – Аукціон на eBay

Платформа часто супроводжує продавців, надаючи широкий функціонал та корисні послуги. Наприклад, налаштування опції міжнародної доставки, вказування країни та вартості. Це супроводжується захистом продавця від міжнародної комісії.

Важливу роль відіграє мобільний застосунок, який є особливо популярним серед молодої аудиторії.

Система довіри, як і на інших платформах, базується на рейтинговій системі, де користувачі можуть оцінювати один одного та залишати коментарі, формуючи публічний рейтинг.

Платформа залишається однією з найпопулярніших у світі та продовжує рости.

1.4 Огляд технологій та інструментів

Для створення клієнтської та серверної частини з використанням бази даних розглянемо наступні технології:

TypeScript – статично типізована мова програмування з компілятором, який транслює код у JavaScript. TypeScript зберігає повну сумісність з існуючим кодом на мові JavaScript. Завдяки своїм особливостям, TypeScript має велику перевагу над JavaScript, а саме:

– читабельність коду: вказані типи допомогають зрозуміти розробнику, як саме можно використовувати класи та об’єкти;

– безпека: завдяки процесу компіляції, помилки в написані програми можна побачити на етапі розробці.

NodeJS – платформа для виконання JavaScript коду на сервері. Має рушій Google V8, який використовується в сучасних браузерах на основі Chromium, та підтримку модулів CommonJS. Для керування модулямі використовується пакетний менеджер NPM.

Для створення клієнтської частини проєкту, існує комбінація фреймворку Next та бібліотеки React з використанням Tailwind.

NextJS – фреймворк для NodeJS, на основі React, для створення вебзастосунків з рендерингом вебсторінок як на клієнті, так і на сервері. Завдяки своєму великому набіру зручних для використання бібліотек, NextJS виділяється з ряду інших фреймворків.

React – бібліотека для створення інтерфейсів у декларативному стилі на основі компонентів. Оскільки React є лише бібліотекою, більшість функціоналу досягається використанням надбудов або фреймворків.

Tailwind – CSS фреймворк для стилізації вебзастосунку. В основі - кодогенерація допоміжних класів для елементів, що дозволяє при розробці легко та зручно поєднувати стилі для створення унікального стилю.

Необхідно створити серверну частину проєкту та розробити зручний інтерфейс для взаємодії з клієнтом. Для цього використовується фреймворк Nest та JSON REST API з протоколом передачі даних HTTP.

Nest – фреймворк для NodeJS, на базі серверного фреймворка ExpressJS, для створення серверних застосунків.

REST (Representational State Transfer) – арітектурний підхід для створення інтерфейсу для надання доступу до ресурсів системи. REST не залежить від формату даних чи мережевого прошарку, тому можна використовувати будь-які формати даних та протоколи, наприклад JSON чи XML з поєднанням HTTP. В реалізації REST має бути підтримка кешування, оскільки залежність від кліента може зробити систему менш стійкою до змін.

Дані, створені користувачем необхідно зберігати. Для цього використвується реляційна база даних PostgreSQL. Для більш зручної взаємодії, можна використати ORM.

ORM (object-relational mapping, об'єктно-реляційне відображення) – технологія, яка зв’язує бази даних та мови програмування методами об'єктно-орієнтовного програмування. ORM спрощує роботу в написані коду, представляючи комплексні зв’язки в базах даних зручними в використані класами та об’єктами.

Sequelize – ORM бібліотека з підтримкою реляційних баз даних. З особливостей можна виділити:

– підтримку транзакцій;

– керування міграціями та сідами;

– реплікація.

PostgreSQL – реляційна система керування базами даних на основі SQL. PostgreSQL має відкритий код, а також розробкою займаються багато людей з різних куточків світу. Відкритість коду допомогає впроваджувати

новий функціонал, що в свою чергу, в порівнянні з іншими системами, робить PostgreSQL найсучаснішою системою. До можливостей можна віднести:

– вбудована мова програмування;

– великий вибір вбудованих типів даних, таких як: геометричні примітиви, IP та IPv6 адреси, JSON, XML дані, масиви;

– вбудована підтримка безпечних протоколів SSL та Kerberos.

PostgreSQL повністю дотримується принципів ACID.

1.5 Постановка задачі

Таким чином, створення дошки онлайн оголошень є актуальним завданням для об’єднання користувачів за торговими потребами. Тому ставиться завдання розробки сучасної онлайн-платформи, з надійною архітектурою та інтуїтивним інтерфейсом.

Об’єктом роботи є купівля-продаж товарів через онлайн-платформи.

Метою роботи є розробка онлайн-сервісу для пошуку та розміщення оголошень про продаж та купівлю товарів.

Для досягнення мети необхідно вирішити такі завдання

– створити зручний користувацький інтерфейс для взаємодії з платформою, що передбачає пошук товарів та публікацію оголошень;

– розробити архітектуру системи, що включає схему баз даних та принцип взаємодії з API;

– реалізувати зручний процес угоди між користувачами

– розробити персоналізований профіль для відображення інформації про угоди та їх статуси.

2 ПРОЄКТУВАННЯ СИСТЕМИ З ПРОГРАМНОЇ ТА ІНТЕРФЕЙСНОЇ СТОРІН

2.1 Інфраструктура системи та вимоги

Системний формат архітектури є базовим фундаментом для будь-якого застосунку, особливо для проєктів на початковому етапі формування. Для цього важливо підходити до планування своєї роботи, оскільки це напряму впливає на подальшу продуктивність, масштабованість та зручність системи.

Оскільки дана робота зосереджена саме на початковому формуванні C2C застосунків, або ж дошок онлайн оголошень, основні метрики підбору формату будуть дещо змінені від загальноприйнятних метрик для комерційних проєктів.

Домінуючими підходами є монолітний та мікросервісний, які пропонують різні рішення щодо вищезгаданих метрик.

Монолітний підхід є традиційним, де весь програмний код застосунку знаходиться в єдиному місці. Сюди входить бізнес-логіка, користувацький інтерфейс, доступ до бази даних. Зазвичай, частини програми залежні одні від одного, що є як і перевагою, так і недоліком.

Моноліт зручний для початкової простоти, коли треба зробити усе швидко для розробки, тестування та розгортання, оскільки все знаходиться у одному місці. Це називається MVP, що розшифровується як «Most Viable Product» і означає легкий застосунок з базовим функціоналом, або ж прототип.

Недоліком такого підходу є вразливість до відмов, тобто при помилці в одній частині системи може бути ураженим весь застосунок. З часом проєкт стає складно масштабувати, оскільки моноліти зазвичай прив’язані до одного технологічного стеку, що призводить до уповільнення розробки.

Мікросервісний підхід являє собою систему з невеликих, незалежних, та слабко зв’язаних модулів. Кожен сервіс відповідає за певну функцію, що має власну кодову базу і розгортається незалежно від інших мікросервісів.

Чіткість та розділення по принципу Single Responsibility дає велику гнучкість у створенні окремої бізнес-логіки та функціоналу. Такий підхід забезпечує стійкість до відмов частин проєкту, а менші кодові бази легше зрозуміти.

Однак, розробка стає складною, а управління комплексним. Окремими проблемами можуть бути проблеми з мережевими затримками, складністю комунікації між цими простими частинками. Нерідко мікросервісні проєкти стають дуже дорогими для підтримування [15, 16].

Ознайомившись з даними підходами, можна проаналізувати вимоги до системи дошок онлайн-оголошень. Вони бувають функціональними та нефункціональними.

Функціональні вимоги визначають пряму взаємодію з користувачами, часто входять сюди вимоги до інтерфейсу та можливостей платформи. Вони, як правило, ставляться на початку розробки та є основними у постановці вимог. Наприклад, це зав’язано з «User story» – короткі вимоги, які одним реченням описують, що очікує користувач. Шаблон може бути таким: «Як користувач, я хочу створювати оголошення для продажу товарів».

Нефункціональні вимоги не пов’язані напряму з функціоналом, але описують, як задовольнити користувача та його потреби. Це може бути масштабованість, зручність користування, безпека тощо [17-18].

Функціональні вимоги для платформи онлайн оголошень:

− реєстрація та вхід в систему: користувачі мають можливість зайти на платформу під своїми даними, що розблокує велику частину функціоналу платформи;

− пошук оголошень: система повинна шукати товари по ключовим словам та відображати товари, які підпадають під критерії пошуку. Це

включає в себе пошук по категоріям, можливість фільтрування та сортування товарів у кожній з категорій;

− створення оголошень: користувач має змогу створити своє оголошення та як завгодно його редагувати. Це включає в себе активацію і деактивацію, що відповідно дозволяє іншим користувачам показувати це саме оголошення, чи приховати у списку доступних товарів;

− профіль користувача: користувач має бачити статистику по продажам і купівлі товарів, власні контактні дані та панель з особисті оголошення для подальшого корегування. Тут має відображатись уся інформація по замовленням товарів, а також продажів. Користувач повинен мати змогу редагувати свій профіль та аватар;

− сторінка оголошення: покупці мають бачити повну інформацію про товар та коротку інформацію про продавця. Система повинна давати розрізняти власника оголошення та звичайного покупця;

− створення замовлень: система має надавати можливість купити товар для авторизованих користувачів. Це включає заповнення особистої інформації та створення спеціального статусу замовлення з підтвердженням як для замовника, так і для продавця.

Оглянувши функціональні вимоги, можна створити список нефункціональних вимог:

− безпека користування: користувачі мають доступ тільки до свого профілю і оголошень. Система повинна забороняти будь-які дії, що стосуються процесу продажі або купівлі, незареєстрованим користувачам, чи користувачам, які хочуть виконувати дії від імені інших користувачів;

− адаптивність інтерфейсу: система має надавати можливість заходити з будь-якого пристрою, з різним розширенням екрану. Включає в себе мобільні пристрої, планшети, комп’ютери та ноутбуки. Інтерфейс має підлаштовуватись під різні розміри екрану;

− зручність використання: сюди входить мінімізація кліків на сторінці задля досягнення мети користувача у функціоналі, приємна кольорова гама, простота у використанні.

Враховуючи вищезазначені вимоги, а також специфіку MVP, очевидним вибором є монолітна архітектура. Оскільки треба швидко запустити готовий проєкт та показати основний функціонал дошки платформ оголошень, можна пропустити на даний момент комплексний підхід до проєктування в цілому, налагодження зв’язків, як от у мікросервісній архітектурі. Ця простота виправдовується тим, що до розробки даного проєкту було залучено лише одну людину. Як наслідок, можливість зосереджуватись на усьому, але малими етапами, дає змогу швидко зробити «скелет» застосунку. Це дає безперечну перевагу у інкрементальних покращеннях, гнучких до змін. Очевидною перевагою є також і процес розробки, оскільки не треба додаткових налаштувань для комунікації між частинами програми. Це суттєво спрошує код та дає змогу використовувати час ефективніше [19].

2.2 Компоненти інтерфейсу системи

Інтерфейс напряму впливає на користувацький досвід, що є дуже важливим для комерційних платформ. У випадку C2C важливим контекстом є створення взаємопов’язаної довіри. Оскільки дошки онлайн оголошень не мають власних товарів, а кожен користувач може відчути себе малим підприємцем, важливо налагодити довіру не тільки між самими користувачами, а й до самої платформи. У сукупності «трикутник» взаємодії може відіграти дуже значну роль у користуванні платформою.

Розділення системи на сторінки суттєво впливає на навантаження контентом та сприйняття інформації. Сюди входить принцип «чистого макету», де використовується зрозумілий і простий макет, з достатнім «білим

простором». Вибір шрифту також впливає на сприйняття, тому не варто застосовувати велику кількість різноманітних шрифтів. Для цього система буде мати один базовий шрифт від бібліотеки Material Tailwind на клієнтській частині. Також навігація між сторінками має включати в себе мінімальну кількість зусиль, які можна виміряти в кліках. Це стосується також функціоналу, по принципу «менше дій – більше результату» [20-21]. В цілому, користувач повинен розуміти, де він знаходиться і що йому робити далі.

Також великою частиною роботи є розпізнавання бренду пересічними користувачами. Для цього є безліч способів, але у даному випадку доцільно використати впізнаваний логотип. На рисунку 2.1 наведено логотип дошки онлайн оголошень.

Рисунок 2.1 – Логотип дошки онлайн оголошень

Назва «Exbuy» походить від симбіозу двох англійських слів: «ex», що означає «колишній» або «минулий», та уособлює специфіфкацію платформи на вживаних товарах, та «buy» − «купувати». Назва водночас проста, легко запам’ятовується та має простий сенс − «Купити вживані речі».

Сторінки, які будуть присутні на дошці онлайн оголошень, можна виокремити: «Реєстрація», «Логін», «Профіль», «Головна», «Створення оголошення», «Оголошення», «Редагування оголошення», «Категорія», «Замовлення». Важливо зазначити, що дошку онлайн оголошень, яка не має локалізації, варто розробляти саме міжнародною мовою – англійською. Це дозволить охопити якомога більше користувачів. Конкретні назви, які

розглянуті українською у даній роботі, мають синонімічний переклад англійською на самому вебсервісі.

Сторінка «Реєстрація» надає користувачу можливість вводити свої персональні дані та бути зареєстрованим на платформі. Сторінка має форму з полями, які можна заповнити. Основним полем, яке буде відображати сутність користувача буде електрона пошта. Отже, на сторінці мають бути розташовані поля електронної пошти та пароля, кнопка «Зареєструватись». Форма також має валідацію, що не дозволяє створити користувача з існуючою електронною адресою. При будь-якій помилці можна побачити «toast» – специфічне повідомлення у вигляді картки у правому нижньому кутку екрану. Цього було б достатньо на даному етапі, оскільки дозаповнити усю інформацію користувач міг би в профілі. Саме мінімальна кількість полів має особливу причину – не перенавантажувати реєстрацію та не фруструвати користувача великою формою. На сторінці також розташована кнопка «Повернутись до головної», яка перенаправляє користувача на головну сторінку. Також на сторінці має бути логотип дошки онлайн оголошень, щоб позначити приналежність бренду. Зазвичай, такі сторінки мають також посилання на сторінку для логіну, тому має бути кнопка «Маєш акаунт? Увійти».

Сторінка «Логін» абсолютно повторює вигляд сторінки «Реєстрація», але з кнопкою «Не маєш акаунту? Зареєструватись», яка перенаправляє на реєстрацію. Сторінка дає змогу увійти на платформу після успішного логіну, та перенаправляє на головну сторінку сайту.

Головна сторінка важлива для першого враження користувача, тому має показати доступ до основного функціоналу платформи. Мають бути розташовані заохочувальні банери та «каруселі» із привабливими товарами. У даному випадку ці віджети мають великий сенс, оскільки основною ціллю головної сторінки – затримати користувача на платформі якомога довше. Тут повинна бути кнопка «Створити оголошення», різні категорії товарів. Пошук

та іконка профілю у шапці сайту забезпечить найкращу орієнтованість для відвідувача платформи.

«Профіль» – одна з найголовніших для користувача сторінок. Тут мають показуватись усі оголошення, створені користувачем, та оголошення, за якими цей ж користувач купляє товари. Тут має бути вичерпна інформація про самі оголошення та пов’язані замовлення, наприклад статуси та дати опрацювання. Також користувач має побачити основну статистику діяльності на платформі. Це включає в себе дату реєстрації, кількість відгуків, проданих та куплених товарів тощо. Основна інформація про користувача – це його ім’я, прізвище, електронна пошта та локація, які також мають відображатись у профілі. Цю основну інформацію одразу можна ж оновити за допомогою інтегрованої форми. Такий підхід дозволяє персоналізувати профіль та насичити його усією інформацією про користувача.

Сторінка «Створення оголошення» – це проста сторінка-форма з полями назви оголошення, ціною, та описом, які є обов’язковими. Варто зазначити, що усі форми повинні мати валідацію, що забезпечить правильність даних. Необов’язковим полем є поле з зображеннями, але якщо його заповнити, мають бути видні усі зображення під цим полем. Кнопка «Очистити» та «Створити» мають бути у кінці форми. Після створення платформа має показати повідомлення про статус створення, і у разі успіху перенаправити на сторінку «Оголошення»

Сторінка «Оголошення» має на меті показати детальну інформацію про оголошення та запропонувати користувачеві купити товар. Важливо розподіляти секції на «картки». У даній роботі таких секцій буде три: зображення товару та детальний опис, коротка і важлива інформація, та секція про продавця. Першим чином треба показати зображення та опис товару, щоб потенційний покупець мав можливість одразу оцінити подальші дії. Секція з найважливішою інформацією має включати дату опублікування оголошення, назву, ціну, статус (активний, неактивний) та кнопка «Купити». У секції про продавця варто зосередити зусилля на довірі покупця, тому

відображення підтвердженого продавця має допомагати покупцю зрозуміти валідність оголошення. Фото продавця, а також справжнє ім’я та рейтинги також суттєво допомагають у вибудовуванні довіри. Тут ж показано кількість успішних транзакцій, локацію, та номер телефону.

Сторінка «Редагування оголошення» доступна тільки для автора оголошення. Цю сторінку можна знайти через саму сторінку оголошення, або ж через профіль. Тут продавець може активувати, або деактивувати оголошення, а також змінити назву, опис, та ціну, разом із зображеннями. Деактивація означає приховування оголошення для усіх користувачів, але не видаляє повністю з бази даних. Це зроблено для відслідковування платформою контенту, який публікує користувач. У такому випадку автор зможе побачити таке оголошення, але вже під іншим статусом.

Сторінка «Категорія» показує основні товари для певної категорії. Таких сторінок може бути декілька, оскільки платформа не матиме централізованого каталогу. Це пояснюється малою кількістю наповнення різноманіттям на початкових етапах розвитку дошки онлайн оголошень. Тут доступна фільтрація та сортування за певними параметрами, а також коротка інформація про товар у вигляді картки.

Сторінка «Замовлення» відображає процес створення замовлення. Має вигляд сторінки-форми. На відміну від профіля, тут усі поля контактної інформації обов’язкові. Це включає в себе також оформлення доставки, адресу і вибір провайдера. Після успішного оформлення, замовлення має з’явитись у профілі.

Опис даних сторінок значною мірою полегшує планування реалізації цілих частин платформи, та дає певну «інструкцію» для користувача щодо платформи та її специфікації. Для кращого розуміння використовують User flow.

User flow – це шлях користувача, який візуалізує його кроки для виконання певної дії під час навігації вебсайтом або застосунком. Як правило, user flow відображається у вигляді блок-схеми або діаграми. По суті

це план навігації продуктом, створений з перспективи користувача на платформі, конкретним інструментом у UX (user experience) – користувацькому досвіді [22]. Для кращого розуміння на рисунку 2.2 наведено схему User flow для замовлення товару на дошці онлайн оголошень «Exbuy».

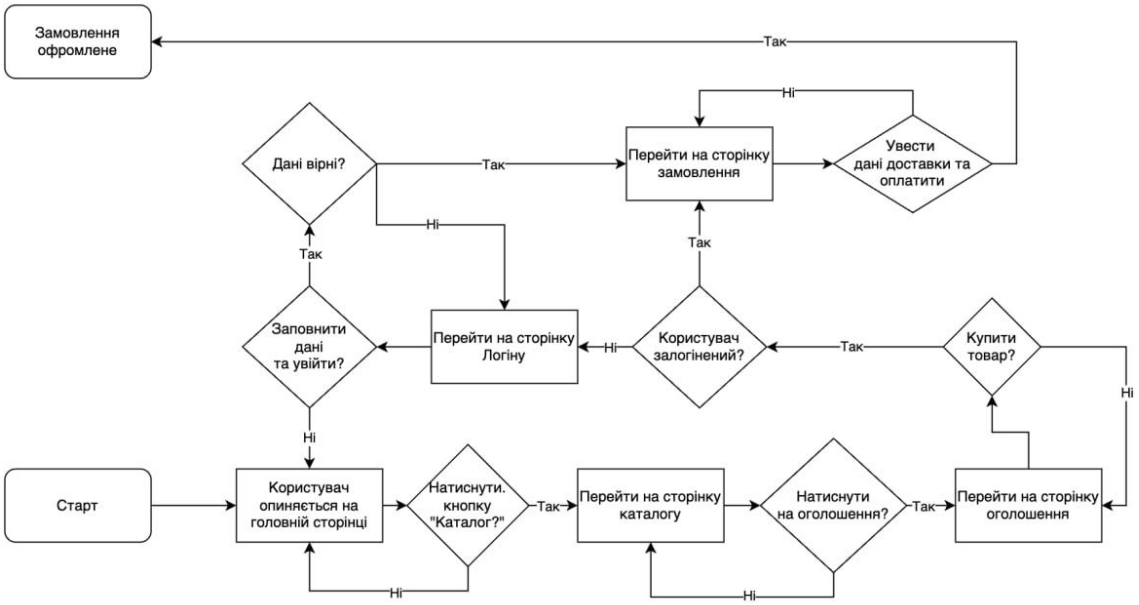

Рисунок 2.2 – User flow схема для процесу купівлі товару на платформі

Загалом, user flow дуже корисно описувати для усіх процесів на платформі для кращого поширення контексту між співробітниками платформи. Але на даному етапі надлишкова документація не потрібна, оскільки платформа не досягла великих масштабів, та сам потік дій користувача достатньо зрозумілий.

2.3 Планування структури бази даних

Протягом усієї історії програмування бази даних були великою частиною інформаційних систем. Бази даних спеціалізуються на

організованому зберіганні та віддачі або прийнятті даних. Одна з найпотужніших баз даних є PostgreSQL.

PostgreSQL – потужна об’єктно-реляційна система керування базами даних з відкритим кодом. Однією з найбільших переваг є наслідування принципу ACID (Atomicity, Consistency, Isolation, Durability), що в основному використовується для систем, які вимагають точність даних.

PostgreSQL підтримує багато типів даних, які розширюють звичайні типи SQL. Наприклад, тип «jsonb» − об’єкт з довільною структурою, і часто використовується для позначення конфігурацій. Це дозволяє поєднувати найкращі практики з реляційних і нереляційних баз даних. Система також підтримує перелічувані типи (ENUM). Саме через розширюваність типів та точність даних, що є критичним для C2C платформ, було обрано саме PostgreSQL [22-24].

Схема бази даних застосунку «Exbuy» матиме декілька табличок: «Users», «Images», «Listings», «Categories», «Orders».

Таблиця «Users» відповідає за зберігання користувачів нашої системи. У моделі нашої системи, саме по обов’язковому полю пошти (email) ми будемо розрізняти користувачів на платформі.

Поле хеш пароля (passwordHash) є зашифрованими з міркувань безпеки. Інші поля, такі як ім’я (firstName), прізвище (lastName), номер телефону (phoneNumber), локація (location), дата реєстрації (createdAt). Ця модель використовується практично на усіх сторінках, оскільки користувач залучений у усі процеси на платформі. Ідентифікатор зображення (imageId) використовується як аватар та посилається на іншу табличку «Images» як зовнішній ключ. Детальну схему користувача можна побачити у таблиці 2.1.

Таблиця 2.1 – Детальна схема таблиці «Users»

Назва поля	**Тип**	**Значення поля**
1	**2**	**3**
id	number	Ідентифікатор користувача

Продовження таблиці 2.1

1	2	3
email	string	Електрона пошта
passwordHash	string	Хеш пароля у зашифрованому вигляді
firstName	string	Ім’я
lastName	string	Прізвище
phoneNumber	string	Номер телефону
location	string	Місцезнаходження, точка продажу
imageId	number	Ідентифікатор картинки (аватару)
createdAt	Date	Дата створення акаунту
updatedAt	Date	Дата оновлення профілю

Табличка зображень «images», яка була зв’язана з табличкою «Users» відповідає за зберігання зображень в цілому, а не тільки фото профілю користувачів. Варто відзначити зберігання зображення ідентифікатору оголошення (listingId). На відміну від таблички «Users», оголошення можуть мати декілька фото, тому у сутності оголошень не буде на пряму зазначено ідентифікатор зображення, натомість, усі дані про зображення оголошень зберігаються саме у цій табличці. Варто відмітити поле шляху до зображення (path). Фактично зберігатись данні будуть у кодовій базі, а дана табличка лише слугує навігатором для знаходження правильного зображення.

Інші поля – дата створення картинки (createdAt) та оновлення картинки (updatedAt) також присутні. Детально розглянути схему можна у таблиці 2.2.

Таблиця 2.2 – Детальна схема таблиці «Images»

Назва поля	Тип	Значення поля
1	2	3
id	number	Ідентифікатор картинки

Продовження таблиці 2.2

1	2	3
userId	number	Ідентифікатор користувача
path	string	Шлях до картинки у кодовій базі
createdAt	Date	Дата створення картинки
updatedAt	Date	Дата оновлення профілю

Табличка «Listings» відображає оголошення, яке є основою платформи. Одне оголошення має лише зв’язок з одним користувачем, який є автором, за допомогою зовнішнього ключа ідентифікатора (userId). Так само одне оголошення може бути тільки в одній категорії, тому маємо ще один зовнішній ключ (categoryId). Тут мають зберігатись дані про назву (title), опис товару (description), ціну (price), статус оголошення (isActive) та дату активації оголошення (activeAt). Детальна схема зображена у таблиці 2.3.

Таблиця 2.3 – Детальна схема таблиці «Listings»

Назва поля	Тип	Значення поля
id	number	Ідентифікатор оголошення
userId	number	Ідентифікатор користувача
categoryId	number	Ідентифікатор категорії
title	string	Заголовок
description	string	Опис
price	number	Ціна
isActive	boolean	Прапорець активності оголошення
activeAt	Date	Дата оновлення оголошення, після активації та деактивації
createdAt	Date	Дата створення оголошення
updatedAt	Date	Дата оновлення оголошення

Наступною табличкою є категорії «Categories», яка відображає логічні частини каталогу. Кожна категорія матиме свою сторінку, де є багато оголошень. Варто зазначити, що ця таблиця групує оголошення по логічному розділенню. Схему наведено на таблиці 2.4

Таблиця 2.4 – Схема таблиці «Categories»

Назва поля	**Тип**	**Значення поля**
id	number	Ідентифікатор категорії
name	number	Назва категорії
createdAt	Date	Дата створення категорії
updatedAt	Date	Дата оновлення категорії

Останньою є таблиця замовлень «Orders», що відображає замовлення до одного оголошення. Варто відзначити, що декілька користувачів можуть замовити однин і той самий товар, але продавець має підтвердити лише одне замовлення. Замовлення зв’язане з таблицею користувачів через ідентифікатор (userId) та саме оголошення (listingId). Статус може мати початкове значення «review» – розглядається продавцем, «pending» – замовлення прийнято і в прогресі, «completed» – замовлення доставлено і виконано, «canceled» – замовлення відмінено продавцем чи покупцем. Можна описати схему таблицею 2.5.

Таблиця 2.5 – Схема таблиці «Orders»

Назва поля	**Тип**	**Значення поля**
1	**2**	**3**
id	number	Ідентифікатор замовлення
listingId	number	Ідентифікатор оголошення
userId	number	Ідентифікатор користувача

Продовження таблиці 2.5

1	2	3
status	enum	Може мати значення review, pending, completed, delivered, canceled
createdAt	Date	Дата створення оголошення
updatedAt	Date	Дата оновлення оголошення (зазвичай через статус)

Огляд структури бази даних дає можливість краще зрозуміти зв’язок між окремими частинами системи, оскільки в самому коді буде додаткова логіка, яка ускладнює процес розуміння. На початковому етапі розробки вебзастосунку структура бази даних безперечно впливає на подальше масштабування, тому варто робити її максимально простою та зрозумілою.

2.4 Архітектура клієнтської та серверної частин

Структура проєкту включає в себе реалізацію клієнтської частини, де користувачі напряму взаємодіють платформою, та серверної частини, де відбувається вся основна логіка обчислень.

Для структуризації серверної частини було обрано шаблон проєктування «Controller – Service – Repository». Суть полягає в тому, що він структурує кодову базу на логічні шари, які відповідають лише за одну конкретну частину функціональності:

− controller відповідає за прийняття, початкову обробку, та відправку запитів із зовнішнім світом. У даній роботі буде застосована взаємодія через HTTP за допомогою архітектурного стилю REST API;

− service відповідає за бізнес-логіку програми, інкапсулює її від прямої взаємодії. Тут в основному зберігається уся логіка застосунку;

− repository відповідає виключно за доступ до даних, достатньо абстрактна частина, яка розширюється сервісом [25].

У даній роботі зазвичай уся логіка буде прив’язана до логічних частин платформи – таблиць, які розглянуто вище. Але є деякі модулі, такі як авторизація, які потребують особливої логіки і мають суміжне представлення між існуючими таблицями.

Важливо зазначити, що для серверної частини буде використовуватись Sequelize, який в свою чергу дає легко звертатись до бази даних. Але для цього треба описувати схему моделей, щоб синхронізувати базу даних з серверною частиною. Для цього і використовується Nest JS з великою кількістю декораторів.

Система зображень передбачає те, що самі зображення будуть зберігатись у кодовій базі сервера, оскільки це найпростіший варіант обробки зображень. Поки система ще достатньо мала на початкових етапах, цього буде достатньо.

Клієнтська частина має вбудовану архітектуру, яку пропонує Next JS. Технологія буде використана через свою простоту у використанні маршрутизації, де кожна папка у директорії «/app» − це окрема сторінка.

Також авторизація буде реалізована через поширення контексту на усю клієнтську частину – це пояснюється тим, що платформа орієнтується на підтверджених і зареєстрованих користувачів. Для цього буде використано React Context API, яка дозволяє передавати дані через увесь застосунок. Авторизація реалізована через JWT – спеціальна система токенів, яка не залежить від серверної частини. Клієнтська частина розділена на сторінки і компоненти в ній − модулі.

Для того, щоб приймати і відсилати запити на серверну частину використовується архітектурний стиль REST API, який підкріплюється зручним інструментом Axios, який дозволяє будувати запити. Як відомо, проблема таких систем полягає в тому, що типізувати це буває складно, тому

важливо тримати типи усіх існуючих таблиць, результатів запитів та інших типів у тісній синхронізації.

Для інтерфейсної частини використовується Tailwind CSS та Material Tailwind, що репрезентують вбудовані класи зі стилями та компонентами відповідно. Це суттєво прискорює задачу верстки, оскільки не треба робити інтерфейс з нуля.

Симбіоз клієнтської та серверної частини дає змогу краще розпланувати та масштабувати проєкт, тому правильні налаштування відіграють ключову роль у розгортанні.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАСТОСУНКУ ДОШКИ ОНЛАЙН-ОГОЛОШЕНЬ

3.1 Програмна реалізація технічної частини

Для розробки застосунку було використано WebStorm – середовище програмного забезпечення (IDE). Webstorm популярний через широкі функції для JavaScript, Typescript та різних вебфреймфорків. Окрім цього надає великі можливості по роботі з базами даних, включаючи PosgreSQL. Для цього використовується плагін «Database Tools and SQL». Інтерфейс середовища наведений на рисунку 3.1.

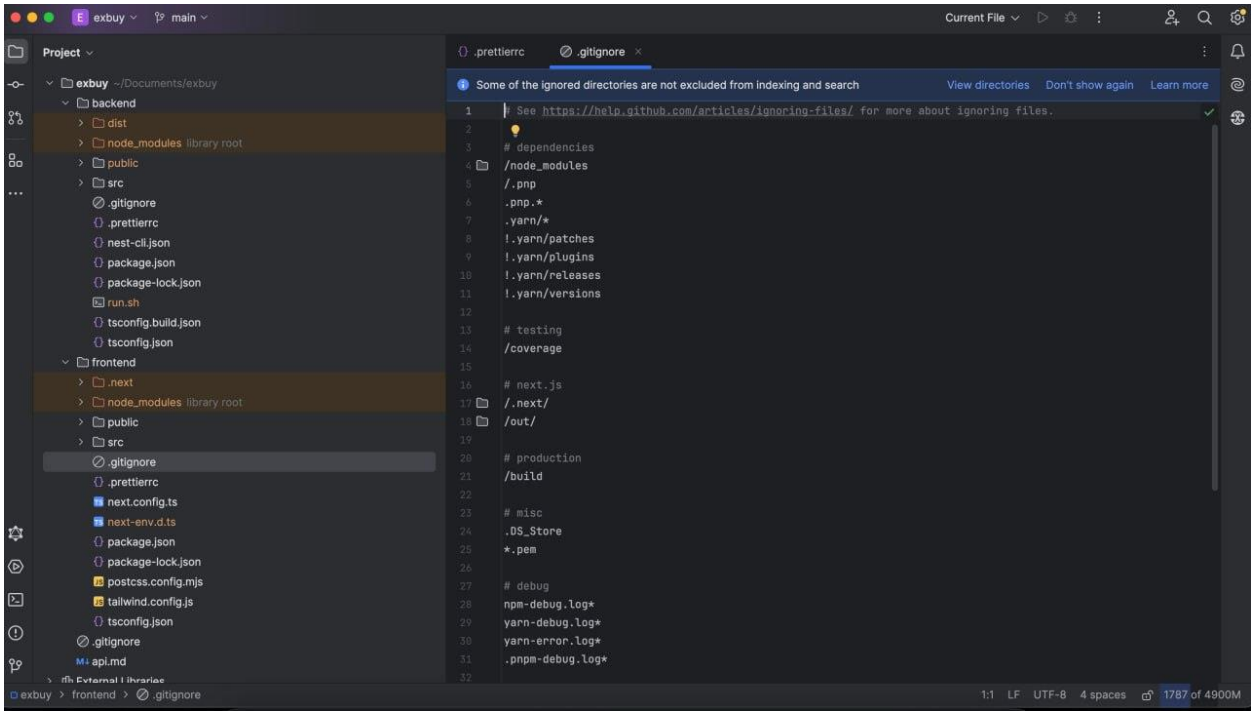

Рисунок 3.1 – Інтерфейс програмного середовища WebStorm

Webstorm зручний своїм автодоповненням коду, підсвіченням синтаксису, виявленням помилок, що покращує та пришвидшує знаходження необхідної інформації та помилок у проєкті. Особлива функція, яку варто відмітити – широкий пошук по директоріям, частинам слова, файловим маскам. Цікавий факт, що у Webstorm присутні два пошуки – один

мінімалістичний для пошуку по частинам слова, а інший – більш розгорнутий, де основним фунціоналом є пошук по частинам проєкту.

Бази даних є важливою частиною програми, тому для них існує безліч графічних клієнтів, що відображають дані у приємному для ока вигляді. Саме такий сучасний клієнт бази даних Beekeper Studio було використано для перевірки даних.

Процес підключення до PostgreSQL в Beekeeper Studio є інтуїтивно зрозумілим. Користувачеві потрібно надати стандартні параметри з'єднання, такі як хост, порт, ім'я користувача, пароль та назва бази даних (рис. 3.2). У даному випадку використовується локальна база даних PostgreSQL.

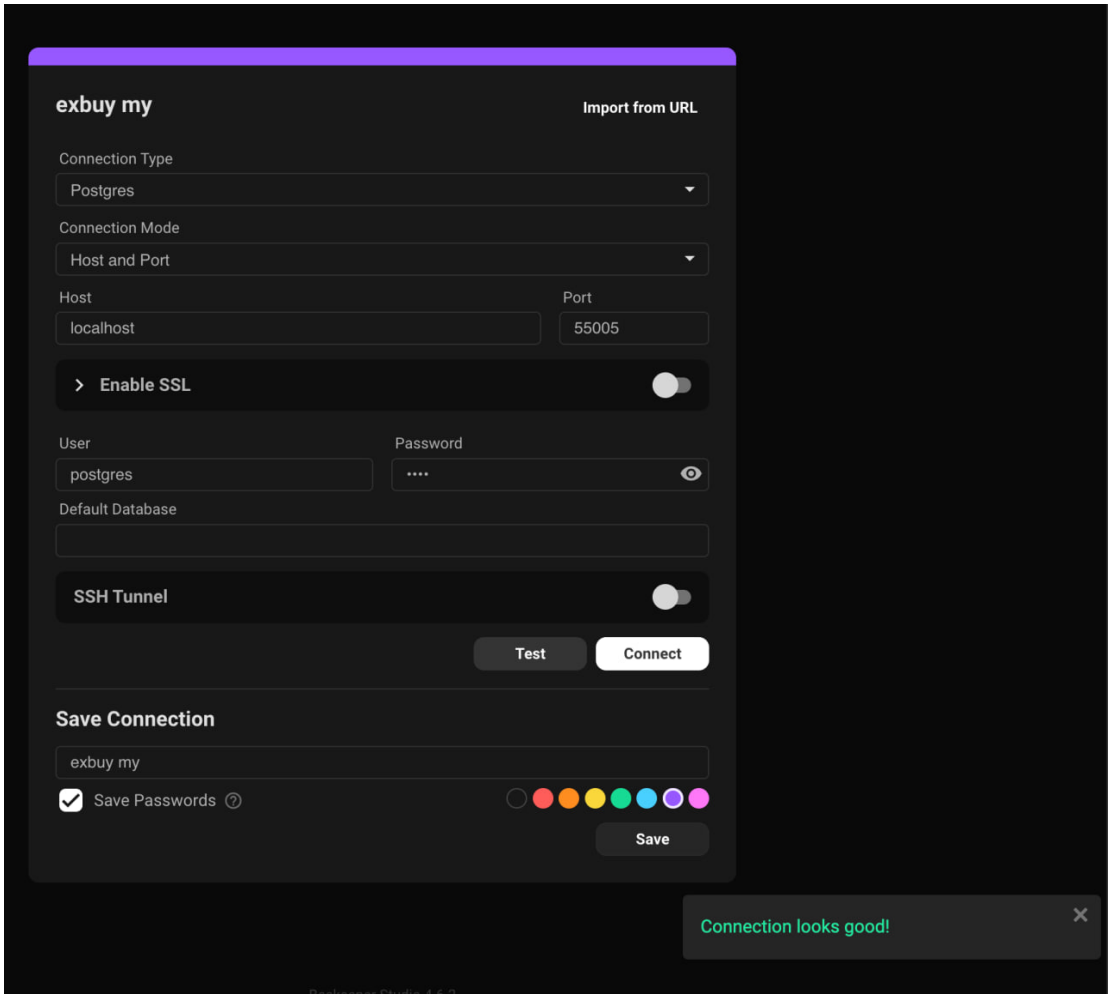

Рисунок 3.2 – Підключення до бази даних у Beekeper Studio

Однією з ключових переваг Beekeeper Studio, яку часто відзначають користувачі, є його сучасний, чистий та інтуїтивно зрозумілий інтерфейс (рис. 3.3). Розробники свідомо уникали складних конструкцій на кшталт «вкладок у вкладках» та застарілих елементів інтерфейсу, характерних для деяких старих Java Swing додатків. Результатом став легкий та приємний у використанні інструмент.

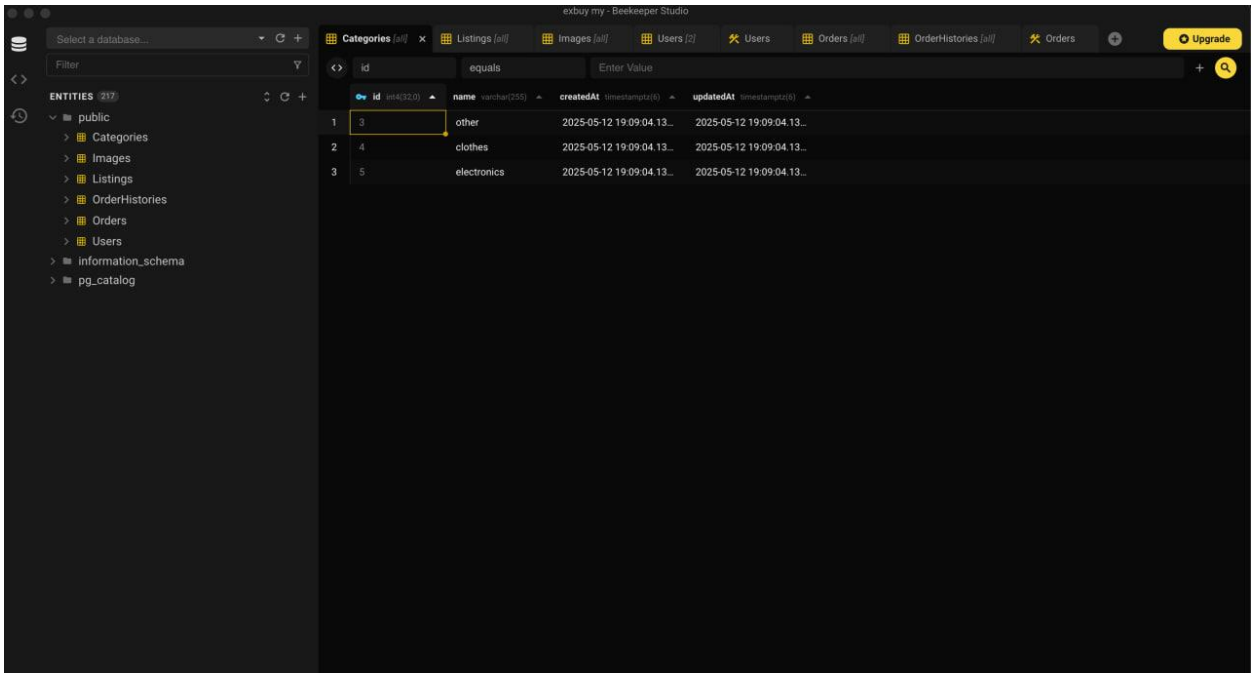

Рисунок 3.3 – Підключення до бази даних у Beekeper Studio

Основним інструментом є TypeScript – надбудова над мовою програмування JavaScript, зі своєю типізацію під час виконання коду. TypeScript використовується як і для серверної, так і для клієнтської частини.

3.1.1 Реалізація серверної частини

Для створення серверної частини було використано NestJS – фреймворк для NodeJS, який під капотом використовує прості HTTP запити для обробки даних.

У NestJS основними структурними одиницями є модулі. Вони групують пов’язаний функціонал у логічні блоки. Модулі – це класи з декоратором «@Module()» [26], що можуть імпортувати інші модулі, контролери, провайдери (логіка застосунку), так само як і експортувати. На лістингу 3.1 зображено основні модулі застосунку.

Лістинг 3.1 Зібрані модулі застосунку у глобальний модуль:

```
@Module({
  imports: [
    AuthModule,
    UserModule,
    ListingModule,
    OrderModule,
    OrderHistoryModule,
    ImageModule,
    CategoryModule,
  ]})
export class AppModule {}
```

Розглянемо основну файлову систему серверної частини. Вона складається в основному з тих самих модулів, але є також і організаційні файли (рис. 3.4).

Важливим фактором є налаштування об’єктно-реляційного відображення, яким і виступає Sequelize – сучасна ORM з підтримкою вбудованих методів, які заміняють прямі запити в базу даних [27]. За налаштування відповідає папка «database». Підключення до бази даних – це теж окремий модуль з власною ініціалізацію у провайдері. Це дуже зручний підхід налаштовувати усе модулями, оскільки це не потребує додаткових інструментів для ініціалізації.

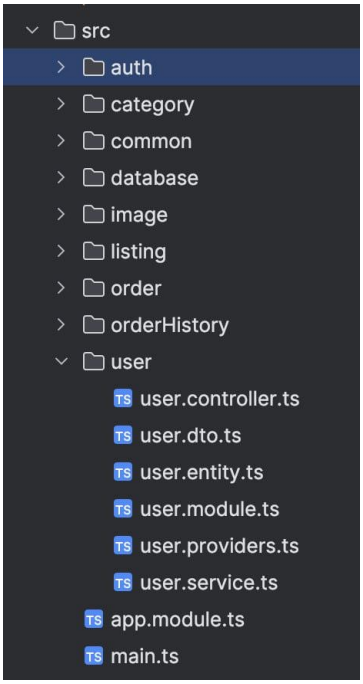

Рисунок 3.4 – Файлова структура серверної частини

Перейдемо до конкретного прикладу розглядання модулю. Найвдалішим варіантом є користувач, оскільки є основним актором системи.

Файли з закінченням «entity.ts» позначають репрезентацію структури таблички бази даних у вигляді схеми. Це потрібно для синхронізації кодової частини та даних. Схема описується за допомогою декораторів NestJS, таких як «@Table» та «@Column». Потім ця схема передається Sequelize, який у свою чергу вже напряму працює з базою даних. Поля бази даних були описані у пункті 2.3.

Варто відмітити зв’язки схеми з іншими схемами. У файлі можна побачити *@HasMany(() => Listing)* та *@HasMany(() => Order)*, що означає те, що користувач може мати багато замовлень та оголошень. Зовнішній ключ *@ForeignKey(() => Image)* означає те, що користувач може мати лише одне відображуване фото аватару.

Файли з закінченням «service.ts» описують усю логіку застосунку. Тут можна побачити різні функції. Конвенційним є те, що напряму до репозиторію саме цього модулю можна підключатись тільки звідси за допомогою *private userRepository: typeof User* у конструкторі. Це зроблено

для розмежування відповідальності кожної частини застосунку. На лістингу 3.2 можна побачити приклад функції всередині сервісу.

Лістинг 3.2 Реалізація функції пошуку користувача:

```
async find(id: number) {
   return this.userRepository.findOne({
     where: { id },
     attributes: { exclude: ['passwordHash'] },
   });
 }
```

Функція *getUserProfileStats* відповідає за обчислення статистики по діяльності на платформі. Сюди входить обчислення кількість виконаних замовлень, як і з сторони продавця, так і покупця. Також по цим даним обчислюється скільки користувач всього заробив та витратив грошей. Реалізація логіки всередині логічного модулю замовлень та оголошень, тому розглянемо імплементацію пізніше. Загалом, звернутись до логіки іншого сервісу можна, наприклад для обчислення кількості проданих товарів *this.orderService.countSellerOrdersByStatusAndUserId(...)*. Дані, отримані у результаті, використовуються для статистики у профілі.

Функції *findWithListing* та *findWithOrders* знаходять користувача, приєднуючи до нього усі оголошення та замовлення відповідно. Цей функціонал досягається за допомогою метода Sequelize – «include», який приєднує моделі одна до одної. Результат функції використовується на сторінці оголошення та звмовлення.

Функція *orders* використовується для повернення усіх замовлень користувача. Використовується на сторінці профілю.

Функція *findByEmail* використовується для знаходження користувача за допомогою електронної пошти. Застосовується для валідації при створенні користувача.

Функція *create* використовується для створення користувача на формі реєстрації. Приймає тільки електрону пошту та пароль.

Функція *update* використовується для оновлення користувацього профілю.

Файли з закінченням «controller.ts» позначають частини програми, які напряму приймають запити. Тут можна вказувати тип запиту який буде приймати контролер за допомогою декораторів «@Put», «@Get» та «@Post». Деякі контролери мають захист у вигляді *@UseGuards(JwtAuthGuard).* Це означає те, що тільки авторизовані користувачі зможуть користуватись даним функціоналом, інакше буде помилка. Для цього використовується перевірка за допомогою JWT (JSON Web Token) – система спеціальних токенів доступу [28]. Токени генеруються сервером, підписуються секретним ключем і передаються клієнту. Клієнт далі використовує цей токен для підтвердження своєї особи. Токени будуть генеруватись при авторизації та зберігатись у локальному сховищі браузера. На рисунку 3.5 наведено приклад такого зберігання.

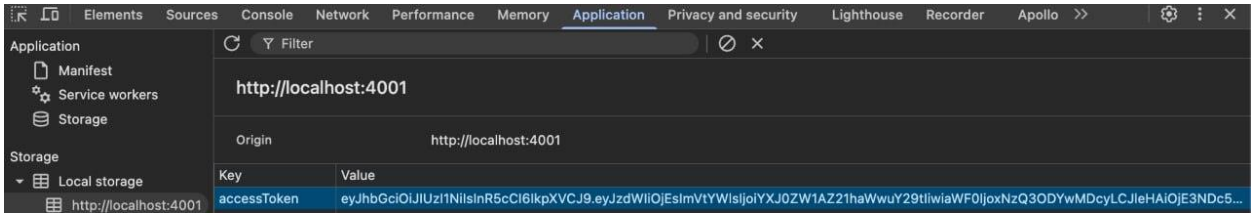

Рисунок 3.5 – Зберігання токену у локальному сховищі браузера

Структура контролерів полягає у ініціалізації базового шляху, наприклад для користувача це буде *@Controller('user').* Далі ініціалізуємо самі шляхи, передавши у «@Put», «@Get» та «@Post» параметри. На лістингу 3.3 наведено приклад такого контролеру.

Для того щоб потім на клієнті звернутись за певною частиною контролера, треба скласти базовий шлях, у нашому випадку це «localhost:4001», назву контролера «user», та назву самої частини контролеру,

куди будемо звертатись, наприклад «me». Складемо це все разом і отримаємо адресу, куди можна звернутись за даними «localhost:4001/user/me».

Лістинг 3.3 Контролер для пошуку даного користувача у системі

```
@UseGuards(JwtAuthGuard)
 @Get('me')
 async me(@Request() req) {
  return this.userService.find(req.user.id);
 }
```

Контролери зазвичай відображають певну функцію в сервісі. У даному випадку це *stats, find (':id/profile'), listings(':id/listing),* та *orders,* що знаходять статистику користувача, самого користувача по ідентифікатору, його оголошення та замовлення відповідно.

Варто виокремити контролер оновлення користувача, що валідує вхідні дані за допомогою декоратора, що використовує Zod схему *@UsePipes(new ZodValidationPipe(updateUserSchema)).*

Опис структури валідації контролерів, або спеціальних представлень даних знаходяться у файлах з закінченням «dto.ts». На лістингу 3.4 зазначений приклад такої валідації за допомогою Zod

Лістинг 3.4 Об'єкт валідації для контролера оновлення користувача.

```
export const updateUserSchema = z.object({
 firstName: z.string().nullable(),
 lastName: z.string().nullable(),
 phoneNumber: z.string().nullable().refine(
    (value) => value === null || value === '' || /^\+380\d{9}$/.test(value),
),
 location: z.string().nullable()});
```

Файли з закінченням «module.ts» та «providers.ts» визначають, які частини підключати до даного модулю. Це можуть бути або інші модулі, або сервіси.

Оскільки було розглянуто основні принципи написання архітектури проєкту, інші модулі будуть надані у вигляді таблиці основної логіки з найважливішою інформацією.

Модуль авторизації створений окремо від користувача, щоб зберігати модульність. Якщо для користувача це тільки маніпуляція його особистими даними, то авторизація зачіпає інші аспекти програми, такі як JWT. У таблиці 3.1 наведено основний опис методів сервісу авторизації.

Таблиця 3.1 – Опис методів сервісу авторизації

Назва	**Основний код**	**Опис**
validate	*const isMatch = await compare (password, user.passwordHash)*	Використовується для валідації логіну. Перевіряється валідність паролю та чи існує користувач.
login	*return {accessToken: await jwtService.signAsync()}*	Логін до платформи. Валідуються введені дані. Генерується валідний JWT токен
register	*Password = hash(password) This.userService.updateLogin*	Генерується новий хеш для паролю, записується у базу даних

Авторизація є максимально полегшеною та простою, тому пропустимо розгляд контролерів авторизації – вони повністю реплікують логіку сервісу, за винятком викидання помилок.

Наступним модулем є зображення. Важливо відзначити те, що вони будуть зберігатись фактично у «public» папці серверної частини, тоді як посилання на шлях буде знаходитись у базі даних. Для цього використовується інтегрований модуль *ServeStaticModule*, якому ми передаємо бажаний шлях для зберігання у аргументи *serveRoot: ‘public’* та

rootPath: join(__dirname, '..', '../public'). У таблиці 3.2 розписаний короткий опис методів сервісу зображень.

Аналогічно до контролеру авторизації, контролер зображень реплікує поведінку свого сервісу, тому розглядати його немає сенсу.

Таблиця 3.2 – Опис методів сервісу зображень

Назва	**Основний код**	**Опис**
findByListingId	*this.imageRepository .findAll({ where: { listingId } });*	Використовується для знаходження зображення оголошення по всьому застосунку: оголошення, профіль, каталог
findByUserId	*return this.imageRepository .findOne ({ where: { userId }, order: [['createdAt', 'DESC']] });*	Пошук найбільш останнього зображення користувача. Використовується на сторінці оголошення та у профілі при перегляду замовлень
createBulkListing	*imageRepository.bulkCreate(paths.map((path) => ({ listingId, path })));*	Створення зображення для оголошення
createUser	*this.imageRepository .create({ userId, path });*	Створення зображення для користувача

Модуль категорій максимально короткий. Ця сутність була створена суто для групування оголошень.

Оскільки у програми немає адмінської панелі та розділення ролей, категорії мають бути створені у базі даних та не мінятись. У таблиці 3.3 наведено методи сервісу категорій.

Таблиця 3.3 – Опис методів сервісу зображень

Назва	Основний код	Опис
findAll	*return this.categoryRepository.findAll();*	Знайти усі каталоги для створення оголошення
findOne	*Return this.categoryRepository .findOne({ where: {id },});*	Знайти специфічне оголошення

Модуль оголошення є основним у застосунку. Варто відзначити, що у одного оголошення можу бути лише один автор, багато зображень, багато замовлень, та одна категорія. Сюди входить пошук, підрахунок користувацької статистики по оголошенням, пагінація для каталогу товарів, активація та деактивація оголошень. У таблиці 3.4 розглянуто лише найважливіші методи, обминаючи звичайні методи по пошуку, створення та оновлення.

Таблиця 3.4 – Опис методів сервісу оголошень

Назва	Основний код	Опис
1	**2**	**3**
search	*where: {title: { [Op.iLike]: `%${query}%` }, isActive: true }, order: [['activeAt', 'DESC']]*	Пошук активних недавніх оголошень по частині слова. Використовується для глобального пошуку
Count MoneyEarned	*const earnFromListing = ordersCount * listing.price; return acc + earnFromListing;*	Підрахунок коштів, зароблених за весь час від успішних угод. Рахується сумою кількості замовлень, помножених на ціну товару

Продовження таблиці 3.4

1	2	3
Count MoneySpent	*return acc + listing.price;*	Підрахунок витрачених коштів на товари. Обираються усі угоди користувача, і шукають ціну у оголошенні
findBy CategoryName	*where: { isActive: true, title: {[Op.iLike]:`%${title}%` ,},}), { price: priceFilter }), activeAt:{[Op.gte]:dateFrom,}*	Пагінація для пошуку товарів за різними параметрами: назва, ціна, дата.
activate	*await this.listingRepository .update({ isActive: true, activeAt: new Date() }*	Активація оголошення, перемкнення прапорцю видимості для усіх користувачів
deactivate	*this.listingRepository.update({ isActive: false })*	Деактивація оголошення означає прихованість для користувачів, крім автора оголошення

Наступним модулем є замовлення. Користувачі можуть управляти різними статусами. Покупці можуть лише запросити замовлення (review статус), та у разі успішного замовлення, підтвердити отримання (completed статус). Продавець навпаки, може прийняти замовлення (in progress статус) та доставити замовлення (deliver статус). Обидві сторони можуть відмінити замовлення у будь який момент (canceled статус). При будь-якій дії з оголошеннями, історія буде відслідковуватись в окремий модуль історій статусів замовлень. У таблиці 3.5 наведено основні методи по роботі з замовленнями

Таблиця 3.5 – Опис методів сервісу замовлень

Назва	**Основний код**	**Опис**
1	**2**	**3**
findBuying AllByUserId	*this.orderRepository .findAll({ where: { userId }*	Пошук замовлень користувача, де він виступає покупцем. Використовується у профілі
Find Selling AllByUserId	*{model: User, required: true, where: { id: userId }},*	Пошук замовлень користувача, де він виступає продавцем. Використовується у профілі
countSeller OrdersByStatus AndUserId	*this.orderRepository .count({ include: { model: Listing, where: { userId } },*	Обрахунок кількості замовлень по статусу та продавцю. Зазвичай використовується по успішним замовленням. Для статистики користувача
countBuyerOrders ByStatus AndUserId	*return this.orderRepository. count (where: {userId, status});*	Обрахунок статистики:: кількості замовлень по статусу та покупцю.
approve	*await order.update({ status: OrderStatus.IN_PROGRESS },*	Прийняти замовлення як продавець. Перевірка на те, чи попердній статус був у статусі review
cancel	*await order.update({ status: OrderStatus.Caneled },*	Відмінити замовлення з перевіркою користувача на учасника угоди
deliver	*await order.deliver({ status: OrderStatus.Delivered },*	Доставити замовлення може тільки продавець.

Продовження таблиці 3.5

1	2	3
complete	*await order.deliver({ status: OrderStatus.Compeleted },*	Підтвердити отримання замовлення може лише покупець.

Останнім модулем є історія статусів замовлення. Як і модуль категорій, статуси замовлення є демонстраційними. Вони мають лише інформацію про час, коли статус був активним. Цей функціонал зроблений заради відслідковування замовлення по часу як продавцем, так і покупцем.

3.1.2 Реалізація клієнтської частини

Для створення клієнтської частини виступає JavaScript фреймворк NextJS – надбудова над React зі своєю системою серверних та клієнтських компонентів. Next JS був обраний саме через свою просту маршрутизацію, щоб полегшити процес навігації користувача між сторінками [29].

Для маршрутизації використовується файлова система у папці «app». Кожна папка – це певний шлях. Кожна папка має мати файл сторінки «page.tsx». Особливо допомагають лейаути – шаблони для сторінок з певною розміткою. Це дозволяє економити час, не переписуючи знову існуючий код. Папки у квадратних дужках означають те, що вони можуть приймати параметри у посиланні. У круглих дужках позначаються певні групи шляхів нашого застосунку. Зазвичай вони використовуються, щоб розділити розмітку для різних сторінок. Розглянемо файлову структуру на рисунку 3.6.

Одразу варто відмітити дві шляхові групи – автентифікація та власне платформа. Логін і реєстрація будуть використовувати один і той самий шаблон розмітки, тому було прийнято рішення їх згрупувати. Платформа ж має категорії, оголошення та профіль.

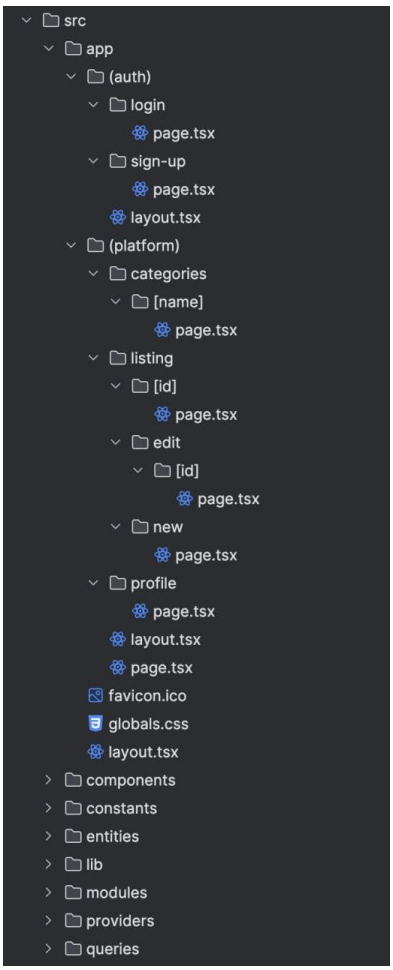

Рисунок 3.6 – Файлова структура клієнтської частини застосунку

До прикладу, категорії визначаються за допомогою імені, тому було додано подвійну вкладеність. Шлях буде виглядати таким чином: «/categories/(назва категорії)». Зауважимо, що сторінки «/categories» не існує і ми отримаємо помилку.

Оголошення мають три розділення – *[id]* позначає саму сторінку оголошення, де є уся детальна інформація. *edit/[id]* позначає сторінку редагування оголошення, а *new* – створення нового оголошення.

Профіль має лише загальну сторінку, у якій буде багато розділень саме у інтерфейсі.

Інші папки проєкту мають свої функції. Папка *components* зберігає у собі компоненти сторінки, що можуть бути перевикористані на різних сторінках. Папка *constants* позначає певні константні значення, які не змінюються під час виконання програми. Наприклад список дат для фільтру

каталогу є сталим значенням. Папка *entities* позначає репрезентацію моделей, які були описані у серверній частині – зазвичай це типи та інтерфейси. Приклад такої репрезентації наведено на лістингу 3.5. Папка *lib* позначає допоміжні функції, що виконують одну конкретну задачу. Зазвичай це функції форматування. Папка *modules* містить усю логіку сторінок, це зроблено для того, щоб розбити сторінку на менші логічні частини. Папка *queries* позначає запити, які будуть використовуватись для отримання даних з серверної частини.

Лістинг 3.5 Репрезентація моделі користувача за допомогою інтерфейсу

```
export interface User {
  id: number;
  email: string;
  firstName: string | null;
  lastName: string | null;
  passwordHash: string;
  phoneNumber: string | null;
  location: string | null;
  createdAt: Date;
  updatedAt: Date;
}
```

Наступним кроком є розгляд структури окремих сторінок. Сторінка реєстрації складається з валідованої форми пошти та паролю. Для валідації використовується бібліотека «react-hook-form» у поєднанні зі схемами Zod. У Zod описується структура валідації та поля, а «react-hook-form» лише проводить зв’язок із розміткою та відслідковує її різноматні стани. Для цього використовується *useForm,* який приймає схему Zod та повертає *register*,

який треба передати у розмітку. Таку взаємодію можна побачити на лістингу 3.6.

Важливо зазначити, що зі сторони стилів застосовано Tailwind Material та Tailwind CSS, що надають зручні готові компоненти та атомарні стилі відповідно. Замість створення файлів CSS, стилі напряму інтегруються в HTML.

Лістинг 3.6 Поле введення пошти з контрольованою логікою за допомгою *register*

```
<input
    placeholder='name@mail.com'
    className={cn('border border-gray-800 p-2 outline-0 rounded-md', {
        'border-red-200 placeholder-red-200': errors.email,
    })}
    {...register('email')}
/>
```

Далі, при відправці форми відбувається взаємодія з серверною частиною. Для взаємодії використовується бібліотека Axios, у яку передаються шлях, дані та опції запиту. У папці *queries* розташовано запит під назвою *signUpQuery*. Цей запит можна побачити на лістингу 3.7. Функція *pathBuilder* з'єднує базовий шлях вебзастосунку з шляхом контролера, який буде приймати запит.

Лістинг 3.7 Запит на реєстрацію

```
export type SignUpQueryData = { email: string; password: string}
export type SignUpQueryResponse = User;
export const signUpQuery = (data: SignUpQueryData): Promise<AxiosResponse<SignUpQueryResponse, any>> => {
    return axios.post(pathBuilder('auth/register'), data)}
```

Сторінка логіну має таку ж саму структуру, як і сторінка реєстрації. Але різниця полягає у тому, що після реєстрації та логіну записується повернута інформація з серверу про користувача та його токен у локальному сховищі. Застосунок після цього дізнається про дані за допомогою ReactContext API – частина екосистеми React, що дозволяє ділитися певними даними у певній частині застосунку або повністю [30]. У будь-якій частині застосунку можна викликати функцію *const { user, setUser } = useAuthUser()*, яка дозволяє дістати або змінити користувача. Всередині себе контекст виконує логіку перевірки користувача при першому заході на сторінку. Якщо є збережений JWT токен у локальному середовищі – можна дістати інформацію про користувача і автоматично його авторизувати. Якщо токену немає, то повинен здійснитись вихід з системи на клієнті за допомогою *setUser(null)* та очисткою локального сховища. Після цього відправляється запит до бази даних щоб ідентифікувати користувача за допомогою токену.

Створити оголошення зареєстрований користувач може за допомогою відповідної сторінки. Зазвичай на сторінках напряму не використовуються запити, оскільки React динамічно перемальовує структуру, і відповідні запити відправляються ще раз. Щоб цього уникнути, використовується хук *useEffect(() => {...}, [])* з пустим масивом залежностей, що означає, що код всередині відпрацює один раз. По усьому додатку використовується саме такий підхід. На лістингу 3.8 наведений приклад обробки запиту на список категорій. Код огорнутий в *try ... catch,* що дозволяє ловити помилки і показувати користувачу за допомогою функції *toast* з бібліотеки Toastify, яка показує спливаючі повідомлення.

Лістинг 3.8 Приклад обробки результату запитів для категорій прямо на сторінці з розміткою

```
useEffect(() => {
  const getCategoriesList = async () => {
    try { const response = await getAllCategoriesQuery();
```

if(!response.data.length) {toast.error(`No categories found, cannot create listing`, toastOptions); return;}

reset({ categoryId: `${response.data[0].id}})

setCategoriesList(response.data)

} catch { toast.error(`No categories found, cannot create listing`, toastOptions);}} getCategoriesList();

}, [])

На сторінці потрібно заповнити поля назви товару, ціни, опису. Також дозволяється додати декілька зображень та обрати категорію. Після успішного створення, за допомогою NextJS користувач перенаправляється до сторінки щойно створеного оголошення за допомогою *router.push('/listing/${id}').*

На сторінці оголошення при першому завантаженні вивантажується багато інформації. Ідентифікувати оголошення можна по переданим ідентифікатором у посиланні. Його можна дістати за допомогою функції *const param = useParam()* з NextJS. За допомогою ідентифікатора витягуємо повну інформацію з сервера запитом. Розглядається два випадки: якщо користувач це автор оголошення або ж покупець. У першому варіанті кнопка замовлення є вимкненою за замовчуванням, та продавець може побачити своє оголошення, навіть якщо воно деактивоване, звідси ж і перейти до редагування. У покупця є доступ до перегляду тільки активних оголошень.

Після завантаження оголошення вивантажуються усі фото оголошення *getAllImagesQuery* та детальна інформація про замовника *getSellerQuery.* За допомогою компонента *OrderDialog* можна замовити товар, зображеного на лістингу 3.9. Всередині заповнюється форма з контактними даними користувача: адресою, коментарем, типом оплати та доставки. Модальне вікно має таку саму структуру форми, розглянутих при реєстрації. Після успішного замовлення користувач перенаправляється до профілю, щоб

побачити детальну інформацію. Користувач не може замовити той самий товар ще раз, оскільки замовлення вже існує.

Лістинг 3.9 Відображення модального вікна при наявності оголошення

```
{listing && (
    <OrderDialog
        isOpen={openOrder}
        listing={listing}
        userId={user?.id || 0}
        handleOpen={handleOrderModal}
    />)}
```

Продавець також може перейти на сторінку редагування оголошення за допомогою компоненту *<Link href={`/listing/edit/${listing?.id}`}>*. Сторінка повністю повторює форму створення, за винятком фунціоналу активації та деактивації. За допомогою відповідного запиту *await deactivateListingQuery({id: listing.id})* можна деактивувати оголошення. При цьому треба оновити існуючу інформацію, це а робиться за допомогою React State – зберігає та дає доступ до редагування інформації. Наприклад, за допомогою const *[listing, setListing] = useState<Listing>().* Після успішного запиту ми синхронізуємо стан, на лістингу 3.10 зображена така взаємодія. Міняється лише поле *isActive*, все інше залишається таким ж як і було, оскільки відомо, що нічого більше не змінилось.

Лістинг 3.10 Синхронізація клієнтської інформації з отриманими даними з сервера

```
setListing(prev => !!prev
    ? ({...prev, isActive: false })
    : prev);
```

На сторінці категорій відіграє велику роль пагінація. Вона буде використовувати параметри з посилання. Це зроблено для того, щоб при перезавантаженні сторінки не втратити налаштовані фільтри. Кількість сторінок для пагінації визначається як загальна кількість оголошень до кількості видимих оголошень *const totalPagesCount = Math.ceil(listingsCount / DEFAULT_ITEMS_ON_PAGE)*. Щоб отримати список пагінованих та відфільтрованих оголошень, потрібно передати опціонально: назву категорії, назву товару, дату публікації, мінімальну та максимальну ціну, пошук тільки з фото, ліміт, та кількість вже отриманих оголошень. Цю взаємодію наведено на лістингу 3.11.

Лістинг 3.11 Передача параметрів до запиту пагінованих та відфільтрованих оголошень

```
const response = await getCategoryListingsByName({
                name: (Array.isArray(name) ? (name[0] ?? '') : name) ?? '',
                title: defaultValues.title || undefined,
                date: date === CategoriesDateFilters.All ? undefined : date,
                priceMin: defaultValues.priceMin ?
Number(defaultValues.priceMin) : undefined,
                priceMax: defaultValues.priceMax ?
Number(defaultValues.priceMax) : undefined,
                onlyWithPhoto: defaultValues.onlyPhoto || false,
                limit: DEFAULT_ITEMS_ON_PAGE,
                offset: (page - 1) * DEFAULT_ITEMS_ON_PAGE,
              });
```

Після цього відбувається одразу сортування по найбільш дешевим, новим та старим критеріям. Варто зауважити, що запит відбувається тільки тоді, коли міняються пошукові параметри у посиланні. Змінюються вони при відправки форми, де налаштовуються фільтри.

Профіль користувача складається з його статистики та табуляції Табуляція включає в себе замовлення, які в свою чергу розділяються на покупки та продажі. Також табуляція включає в себе список товарів користувача та редагування контактної інформації.

Статистика включає в себе також і аватар користувача. Усі картинки зберігаються у папці «public» на серверній частині. Але за допомогою функції *getImageByPath* дістається повний шлях до зображення, а не тільки його ідентифікатор. Все що робить ця функція це повертає зібраний рядок *return `http://localhost:3000/public/${path}`*.

У секції оголошень користувача зібрані картки з короткою інформацією. Звідси також можна перейти до редагування оголошення. Це досягається за допомогою перевірки, чи є користувач автором *const isCurrentUserSeller = listing.userId === user?.id.*

У секції редагування профілю можна також оновлювати свої зображення. Оскільки «react-hook-form» контролює лише поля з базовими типами, маємо логічне розділення по оновленню зображень та користувача в цілому. Для завантаження зображень використовується «react-dropzone». Тут можна специфікувати який формат зображення потрібен за допомогою поля *accept: {'image/png': ['.png']}*. Щоб відокремити логіку оновлення, використовується спеціальний стан *isDirty* з «react-hook-form». Він означає, якщо поля були змінені і відрізняються від стандартних, то це правдиве значення. Важливо зазначити, що після оновлення профілю треба перезавантажити сторінку за допомогою *window.location.reload().* Це зроблено для того, щоб оновити інформацію по зображенням на сервері.

У секції замовлень ми зберігаємо поточне замовлення, яке буде відображати повну інформацію поза списком. Це може бути як і замовлення де користувач виступає як замовник, так і як продавець. Для того, щоб відрізняти тип замовлення, при витягуванні даних додатково додається тип замовлення. На лістингу 3.12 продемонстрована така логіка додавання додатково з станом поточного замовлення.

Лістинг 3.12 Додавання ідентифікатора типу замовлення для відображення

```
const [selectedOrder, setSelectedOrder] = useState<
    ProfileBuyingOrderType
    | ProfileSellingOrderType
    | null
  >(null);
const response = await getUserBuyingOrdersQuery();
        setOrders(response.data.map(order => ({
          ...order,
          type: OrderType.Buying
        })));
```

Якщо користувач не має замовлень, то показується відповідне повідомлення «No orders found».

При обраному замовленні, залежно від типу, відображається панель користувача або продавця з детальною інформацією про замовлення. Панелі відображають однакову інформацію, за винятком логіки дій з замовленням. В залежності від даного статусу замовлення та чи користувач продавець чи покупець, користувач має різний контроль над замовленням. На лістингу 3.13 зображений компонент, який визначає видимі кнопки. Варто зауважити, що кнопка відміни замовлення буде показуватись завжди, окрім коли замовлення виконане та вже скасоване.

Лістинг 3.13 Компонент контролю дій замовлення

```
<div className="flex gap-2 items-center grow">
      {!( currentStatus === OrderStatus.CANCELED
```

```
        || currentStatus === OrderStatus.COMPLETED ) && (
            <ActionButton orderId={orderId} action={cancelAction}
onRerender={onRerender}/> )}
            {nextAction && <ActionButton orderId={orderId}
action={nextAction} onRerender={onRerender}/>}
        </div>
```

Користувач також може переглядати історію статусів у вигляді списку. Це досягається за допомогою вбудованого компоненту *TimeLine* від Material Tailwind.

В цілому клієнтська частина одноманітна через схожість функціоналу та компонентів. Більшість сторінок мають форми з полями та одноманітний шаблон витягування даних. Наступним кроком є огляд інтерфейсу.

3.2 Огляд інтерфейсу

Інтерфейс є основою взаємодії користувача та програми. Платформа сконструйована так, щоб користувач інтуїтивно орієнтувався серед існуючого функціоналу.

Розглянемо головну сторінку (рис. 3.7), де користувач може побачити декораційні банери, що мотивують створювати нові оголошення. Звідси можна перейти до категорій, натиснувши на відповідну іконку: електроніка, одяг та інші товари.

Також на сторінці зображений динамічний віджет останніх створених оголошень. Кожні пару секунд віджет пролистується автоматично, та користувач може побачити більше нових товарів. Варто зауважити, що натиснувши на один з оголошень, можна подивитись детальну інформацію.

В цілому, на платформі завжди можна побачити шапку профілю, окрім сторінок реєстрації та логіну.

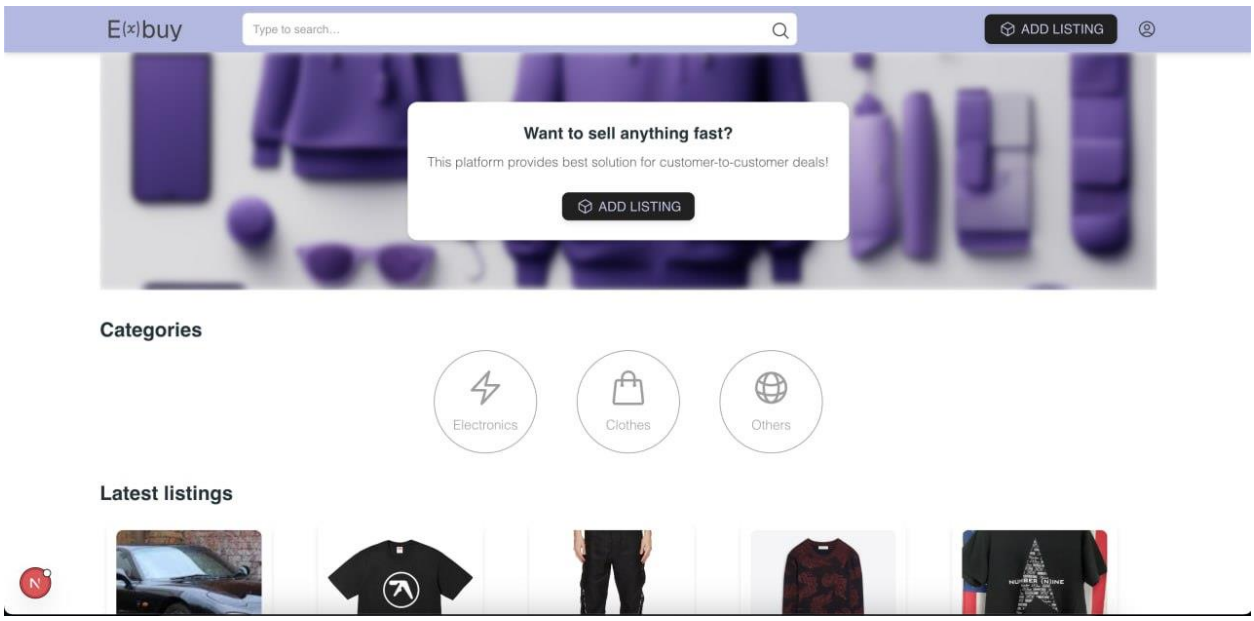

Рисунок 3.7 – Головна сторінка сайту

Перейти до форми реєстрації та логіну можна, натиснувши на іконку користувача з правої верхньої сторони у шапці сайту. Відкривається випадаючий список, де треба натиснути на кнопку профілю (рис. 3.8).

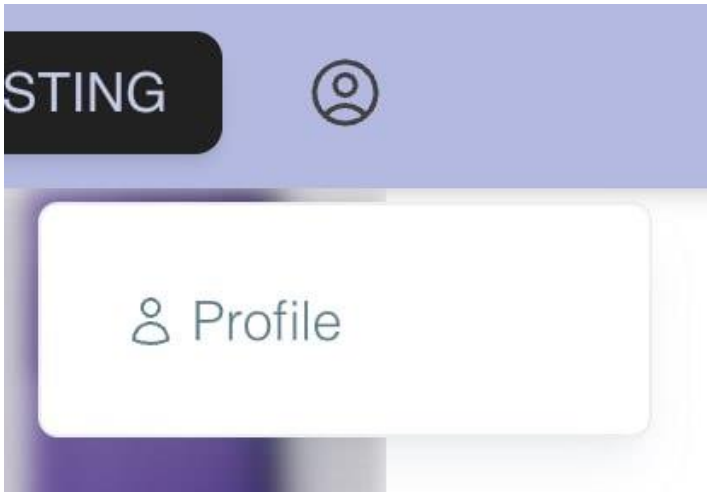

Рисунок 3.8 – Випадаючий список

На сторінці реєстрації працює валідація – це означає перевірку даних під час відправки форми. Спроба відправити пусту форму зображена на рисунку 3.9.

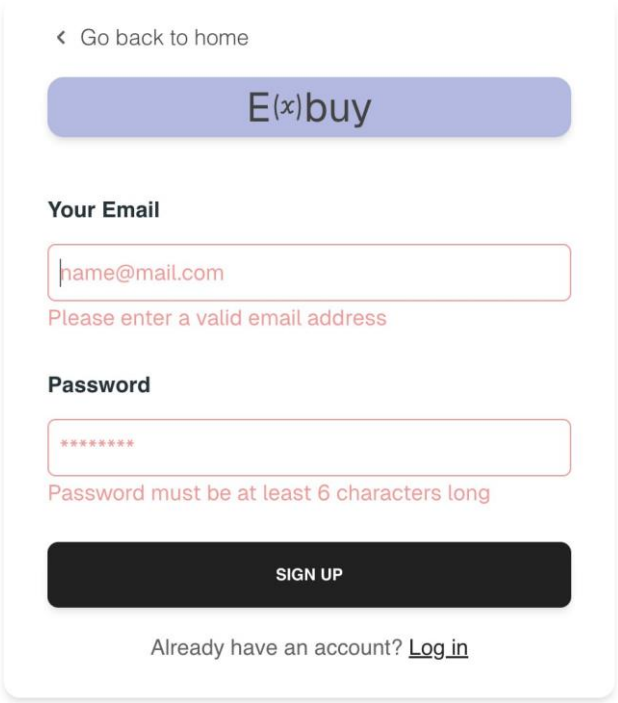

Рисунок 3.9 – Спроба зареєструватись з пустою формою

Зареєструємось та перейдемо до сторінки категорій одягу (рис. 3.10). Тут можна фільтрувати товари по назві, ціні, даті публікації та фото.

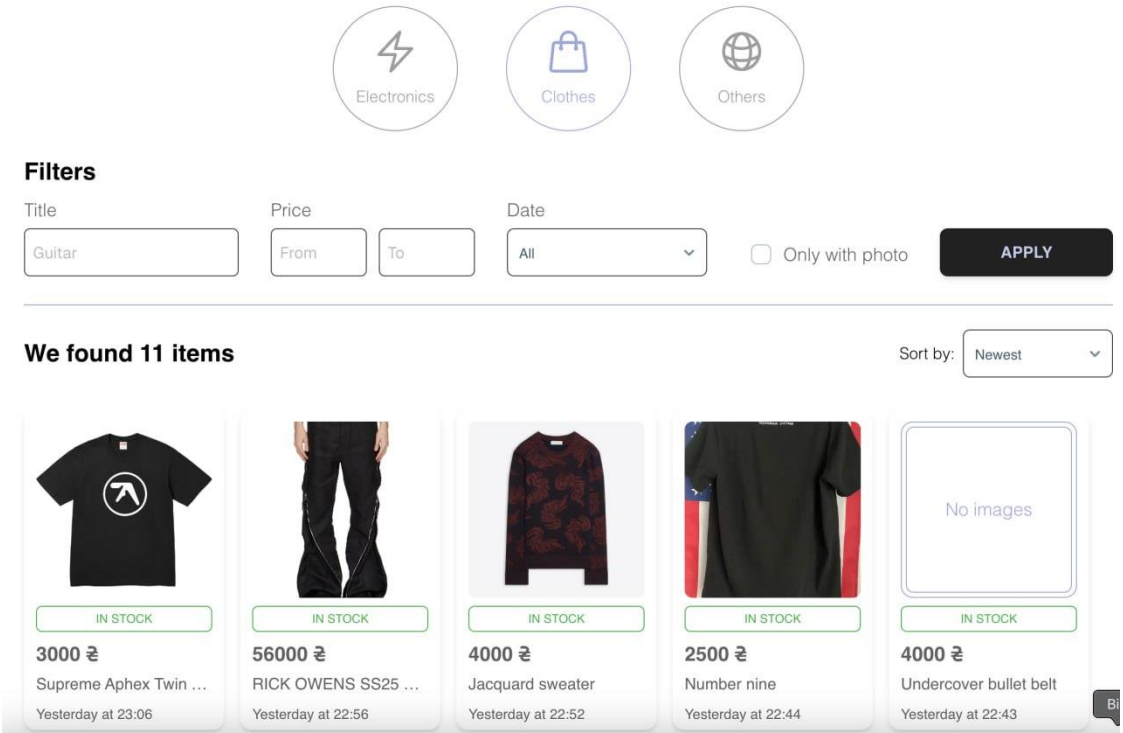

Рисунок 3.10 – Сторінка категорії одягу

На рисунку 3.11 показані результати по довільним заповненим фільтрам та обраним сортуванням. Для цього після вибору самих опцій треба натиснути кнопку «Apply», щоб застосувати зміни.

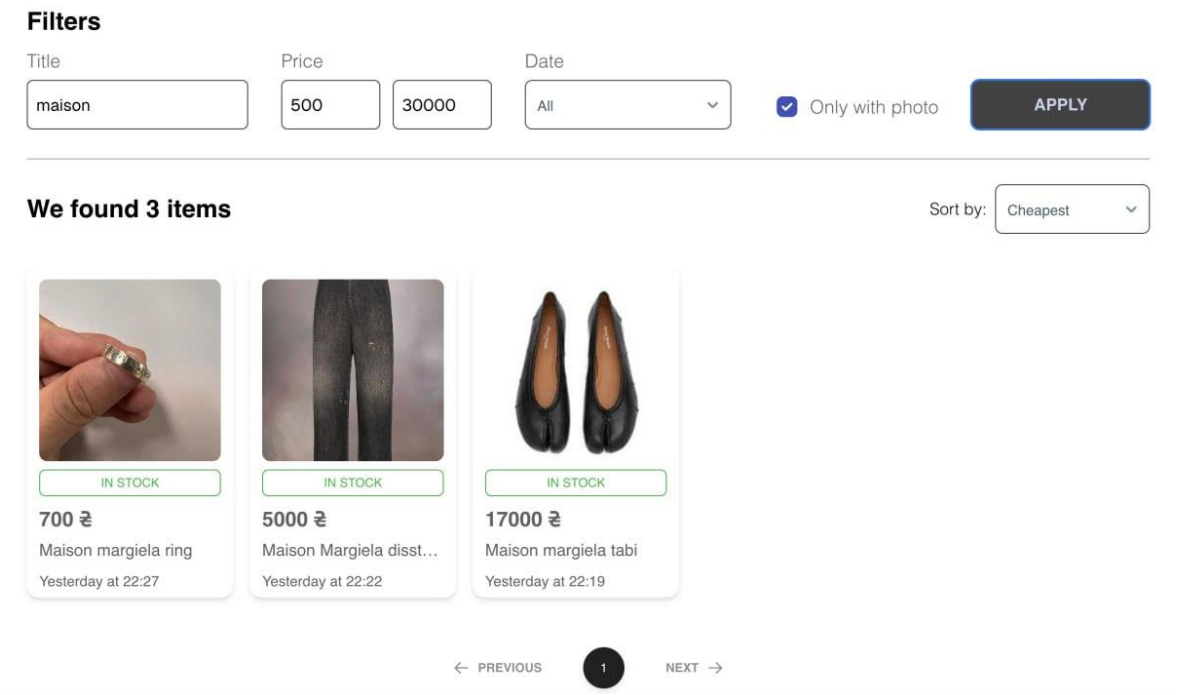

Рисунок 3.11 – Застосовані фільтри та сортування на сторінці каталогу одягу

Після вибору оголошення, треба натиснути на картку, щоб перейти на сторінку з детальною інформацією. На рисунку 3.12 зображена сторінка оголошення. Тут можна переглядати фото за допомогою динамічного віджету каруселі. Детальна інформація позначена знизу, щоб не завантажувати верхню частину сторінки великим контентом.

У боковій панелі відділено інформацію про сам товар та продавця. Товар має інформацію про дату публікації оголошення, назву, ціну, та статус активності з кнопкою замовлення.

Інформація про продавця зображена нижче, та виводить його фото, ім’я, та локацію. Це зроблено для того, щоб покупець розумів, з якою людиною контактувати. Зазвичай незаповнений профіль продавця насторожує користувачів, тому було прийнято рішення використати усю заповнену інформацію по максимуму.

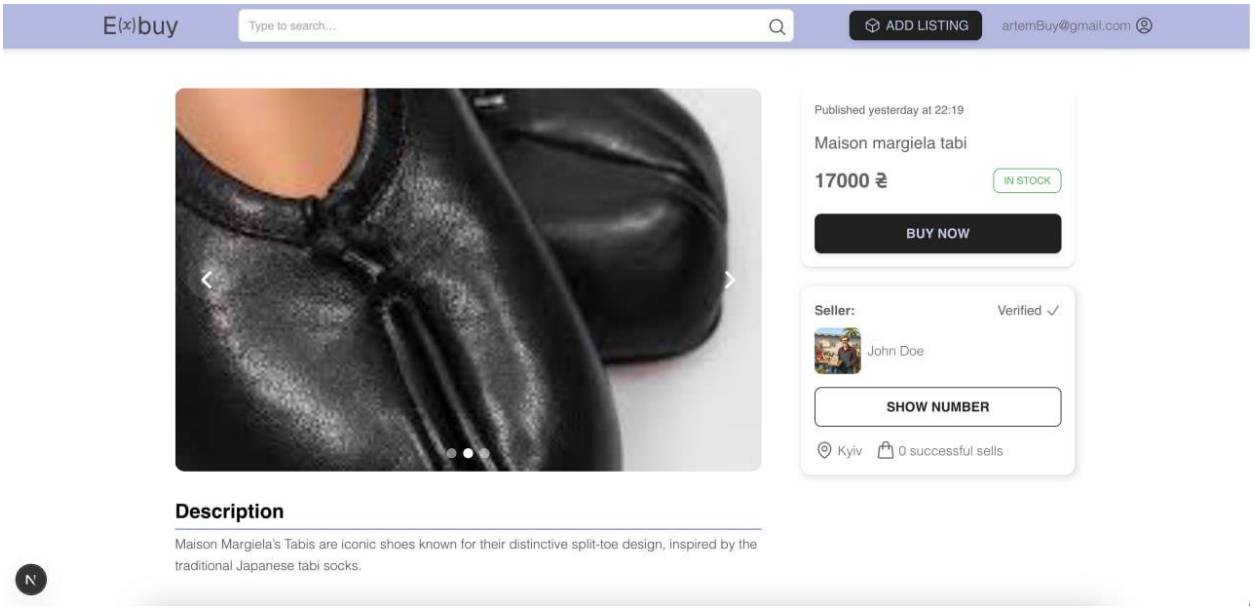

Рисунок 3.12 – Сторінка оголошення

Якщо продавець вказав номер свого телефону, то можна переглянути його, натиснувши на кнопку «Show number» (рис 3.13). Номер можна легко скопіювати, натиснувши на білу кнопку з правої сторони номера.

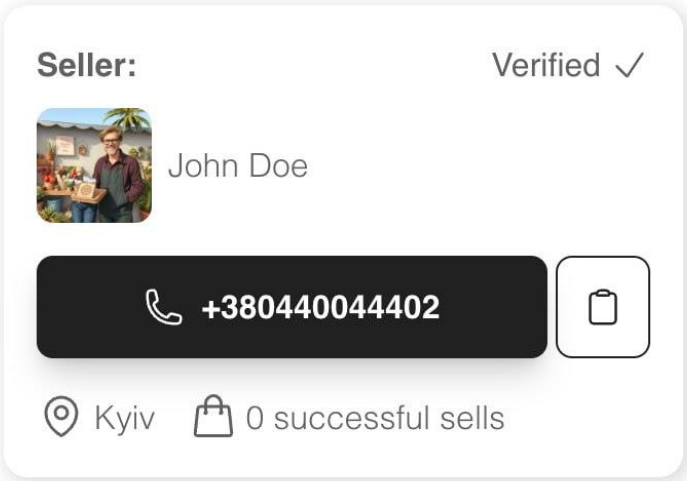

Рисунок 3.13 – Віджет продавця з заповненим телефоном

Замовити замовлення можна натиснувши кнопку «Buy now». Після цього відкриється модальне вікно для заповнення деталей замовлення (рис. 3.14). Тут користувач має ввести дані про тип оплати, адресу і тип доставки та опціонально коментар.

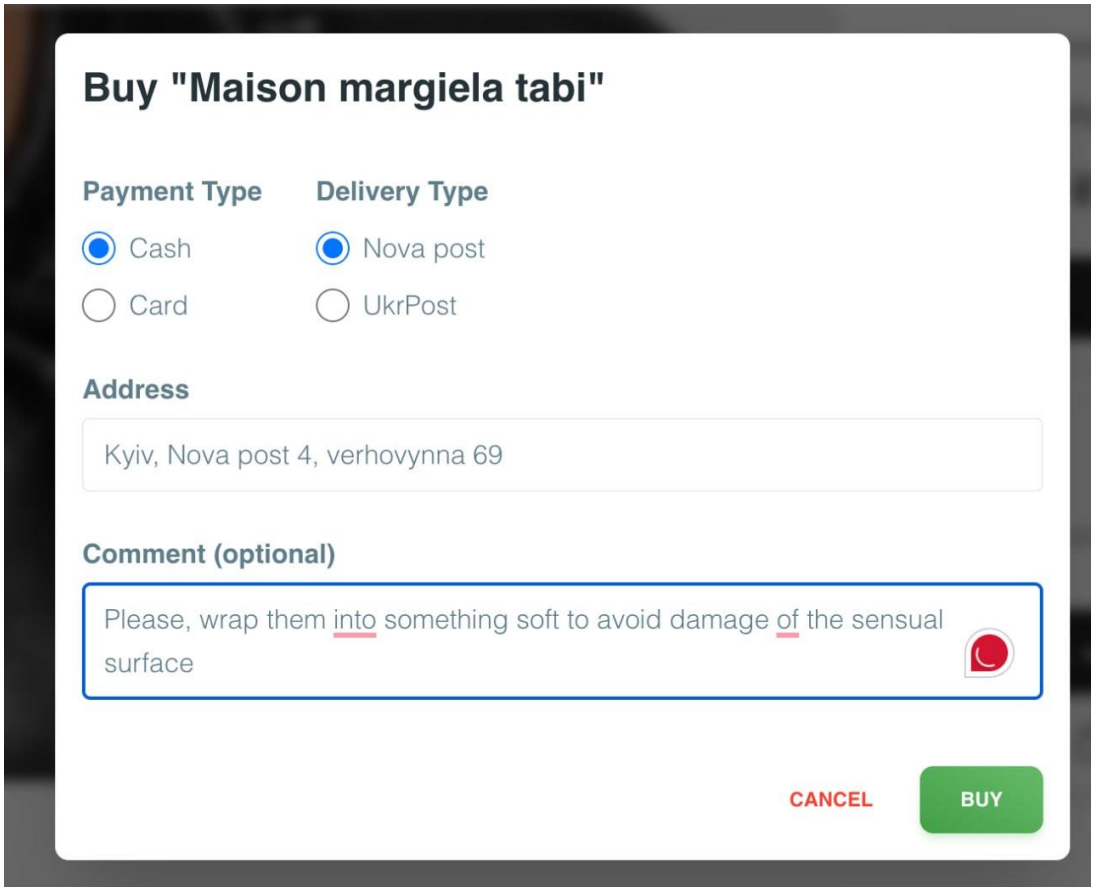

Рисунок 3.14 – Модальне вікно заповнення інформації про замовлення

Після цього користувач перенаправлений до сторінки профілю, щоб переглянути замовлення та побачити список інших замовлень. На рисунку 3.15 зображена сторінка профілю. Профіль має три підсекції – замовлення, власні оголошення та редагування профілю. Замовлення у свою чергу діляться на ті, що користувач виступає у ролі покупця та продавця.

На даному етапі ми не будемо розглядати інші секції, оскільки користувач ще не має власних оголошень та замовлень, де виступає продавцем.

Коротка інформація про замовлення з лівої сторони включає в себе перше фото (якщо існує), назву, ціну, ім’я продавця (або ж пошти, якщо ім’я не зазначене) та адресу доставки, поруч зі статусом замовлення.

Натиснемо на картку, щоб подивитись детальну інформацію про оголошення та статуси замовлення (рис. 3.16).

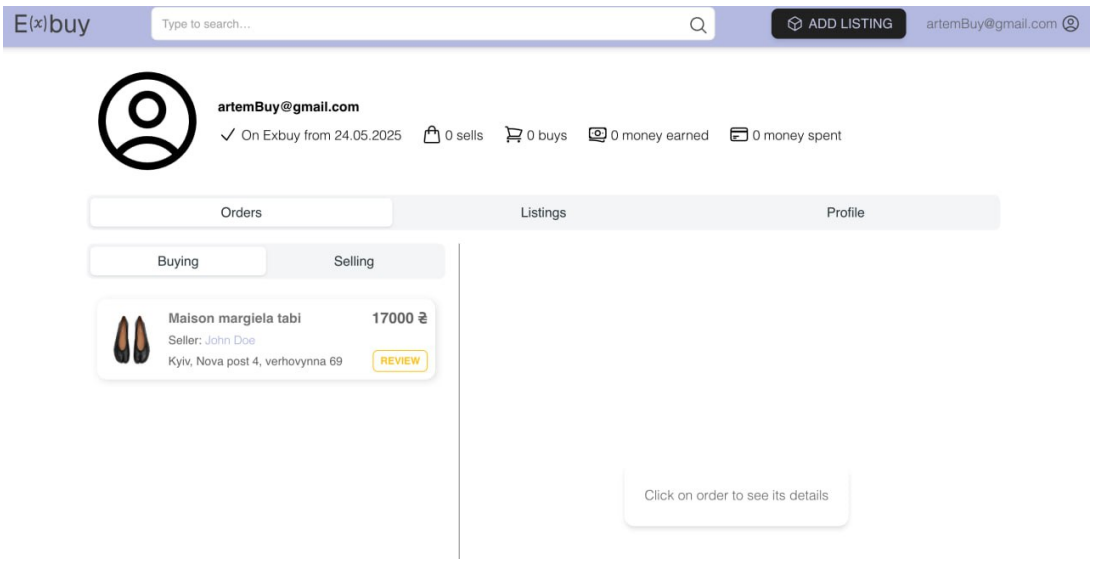

Рисунок 3.15 – Сторінка профілю

Детальна інформація про оголошення містить у собі всю ту інформацію, що заповнював покупець при відправці форми оформлення замовлення. Тут зображена кнопка «Cancel order», що позначає відміну замовлення. З правої сторони від цієї кнопки є кнопка переходу на сторінку оголошення, до якого прив’язана угода. Під час кожної ітерації замовлення можна побачити підказки щодо угоди. З правої сторони буде відображатись хронологія статусів замовлення.

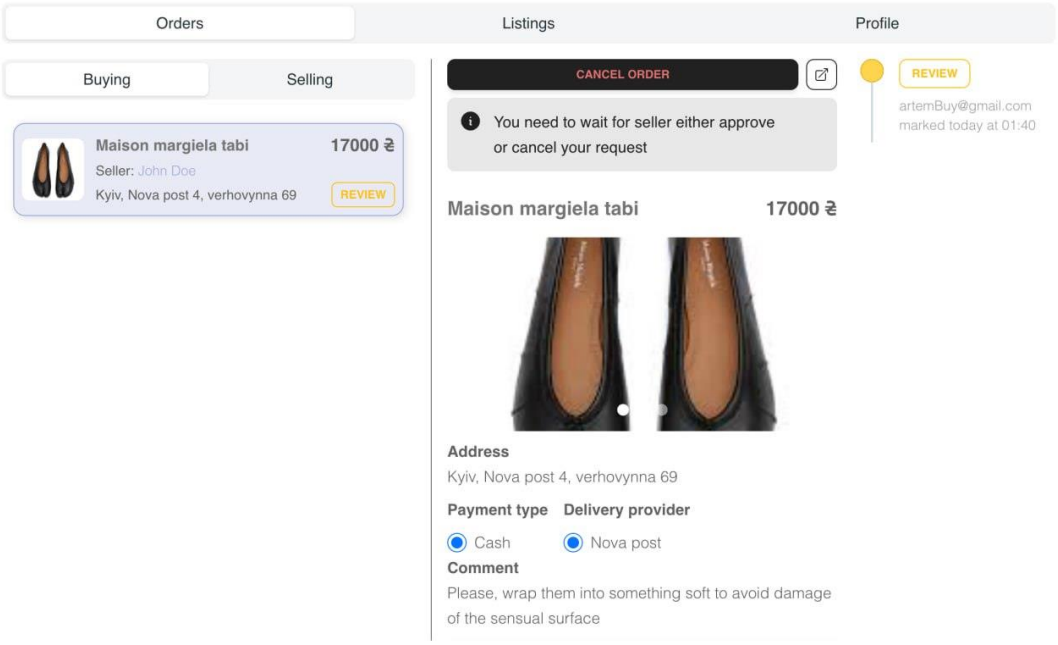

Рисунок 3.16 – Детальна інформація про замовлення

На даному етапі продавець повинен підтвердити замовлення, або ж у будь-який момент скасувати. Для цього нам потрібно вийти з акаунту покупця та зайти у акаунт продавця. Це можна зробити, натиснувши на іконку користувача, і у випадаючому списку натиснути кнопку «Log out» (рис. 3.17).

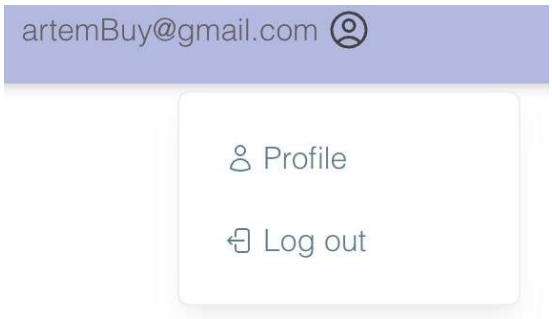

Рисунок 3.17 – Вихід з акаунту

Зайдемо у акаунт користувача, натиснувши кнопку того ж самого віджету у шапці профілю та перейдемо до сторінки логіку, де треба ввести дані продавця та зайти в акаунт (рис. 3.18).

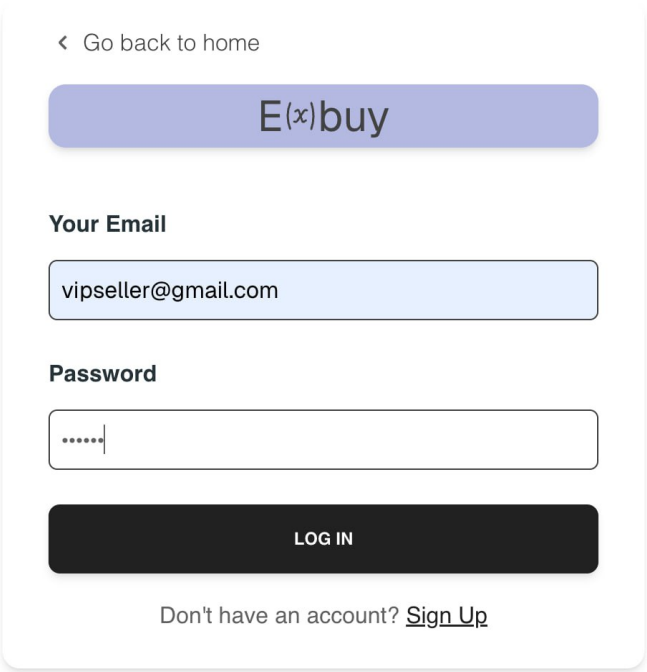

Рисунок 3.18 – Автентифікація продавця на сторінці логіну

Далі треба перейти на сторінку профілю, але тепер у вкладку «Selling», що позначає угоди, де користувач виступає продавцем (рис. 3.19).

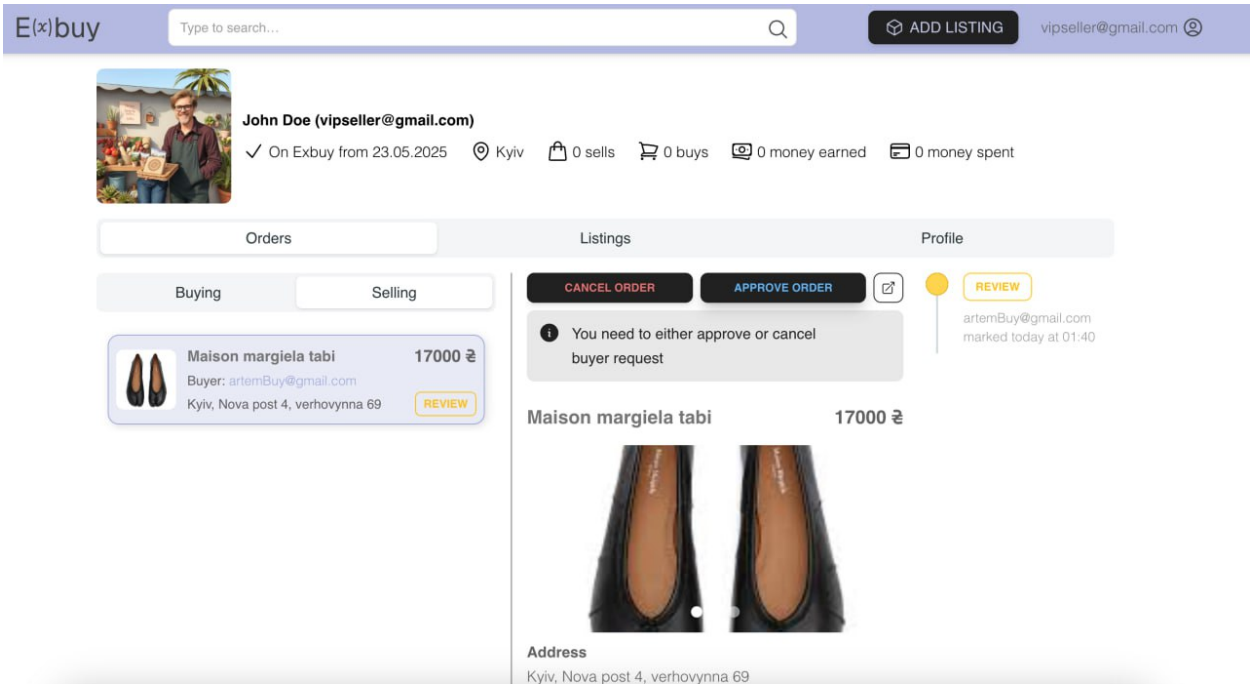

Рисунок 3.19 – Сторінка профілю продавця з його угодами

Натиснемо на кнопку «Approve order», щоб прийняти замовлення (рис. 3.20). Можна побачити, що статус замовлення змінився на той, що у прогресі. Також оновилась історія статусів, де можна відслідкувати час зміни статусу та того, хто змінив цей статус.

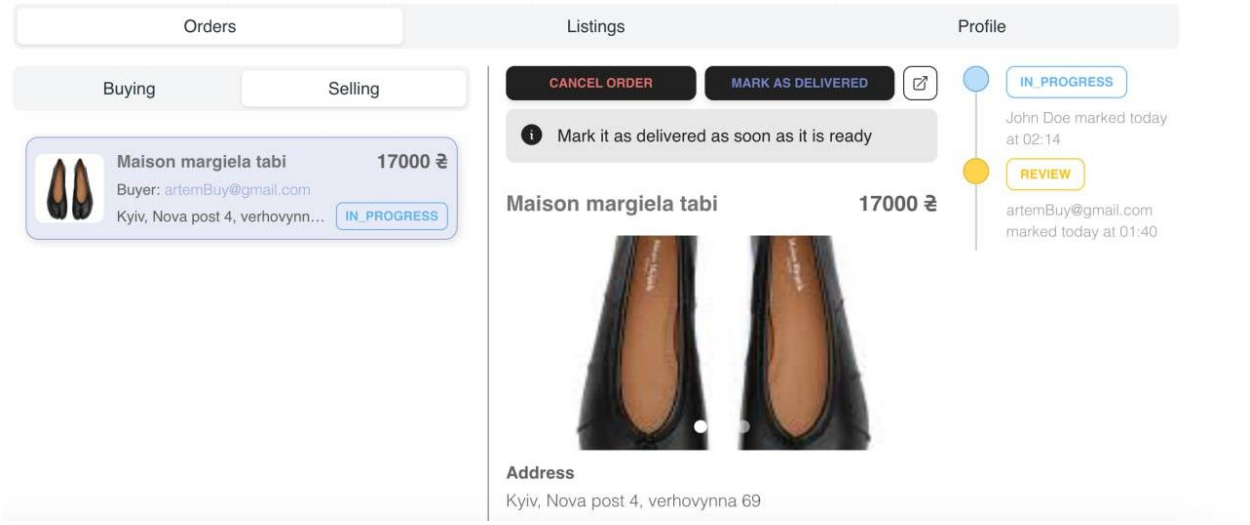

Рисунок 3.20 – Замовлення у процесі виконання

Коли замовлення буде доставлено у відділення пошти, продавець має відмітити статус замовлення як доставлене. Для цього треба натиснути кнопку «Mark as delivered» (рис. 3.21).

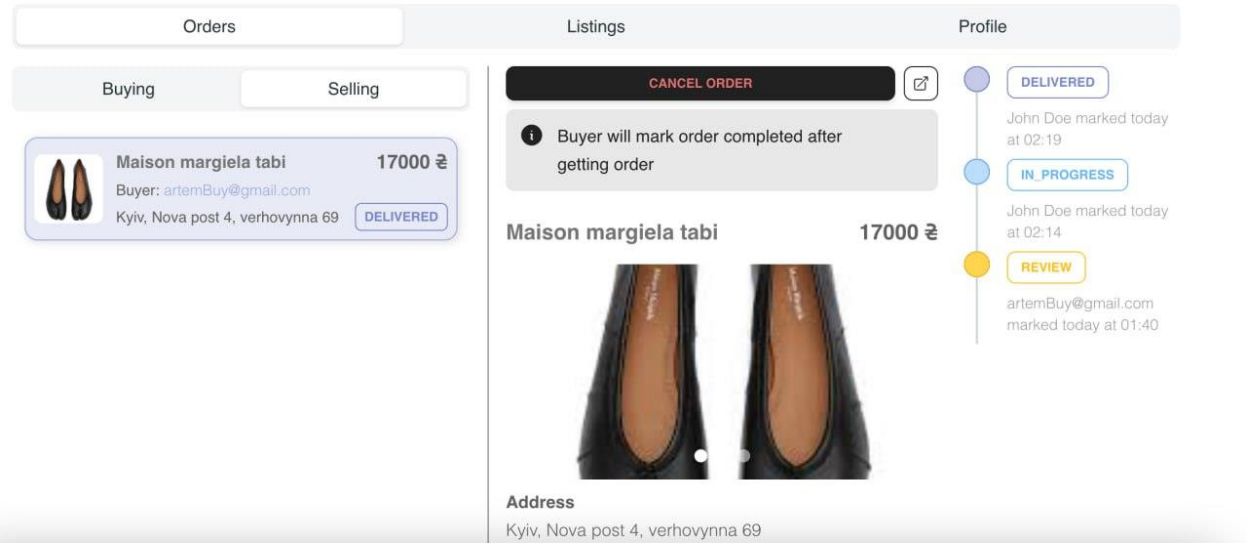

Рисунок 3.21 – Замовлення відзначено продавцем як доставлене

Наступним кроком є підтвердження отримання замовлення користувачем. Для цього вийдемо з акаунту продавця і назад увійдемо до акаунту покупця (рис. 3.22).

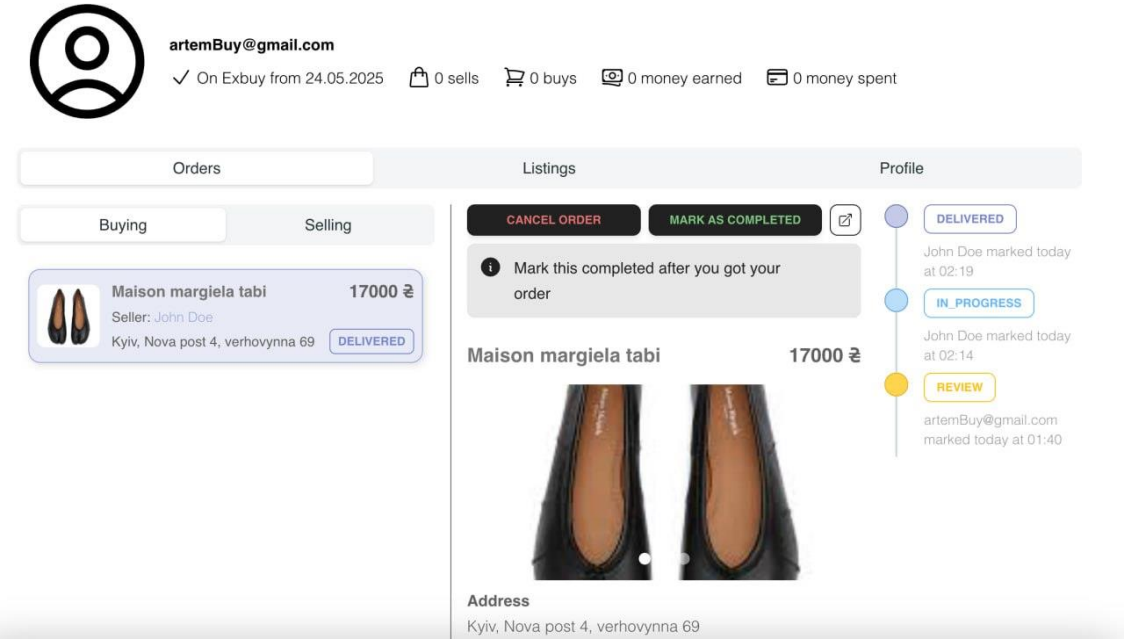

Рисунок 3.22 – Замовлення після доставки у кабінеті користувача

Натиснемо «Approve order», щоб підтвердити замовлення. На рисунку 3.23 замовлення вважається виконаним. Варто зауважити, що статистика користувача теж оновилась – тепер користувач має одне успішне замовлення та кількість витрачених грошей рівним ціні замовлення. На рисунку 3.24 зображена статистика покупця після завершення замовлення.

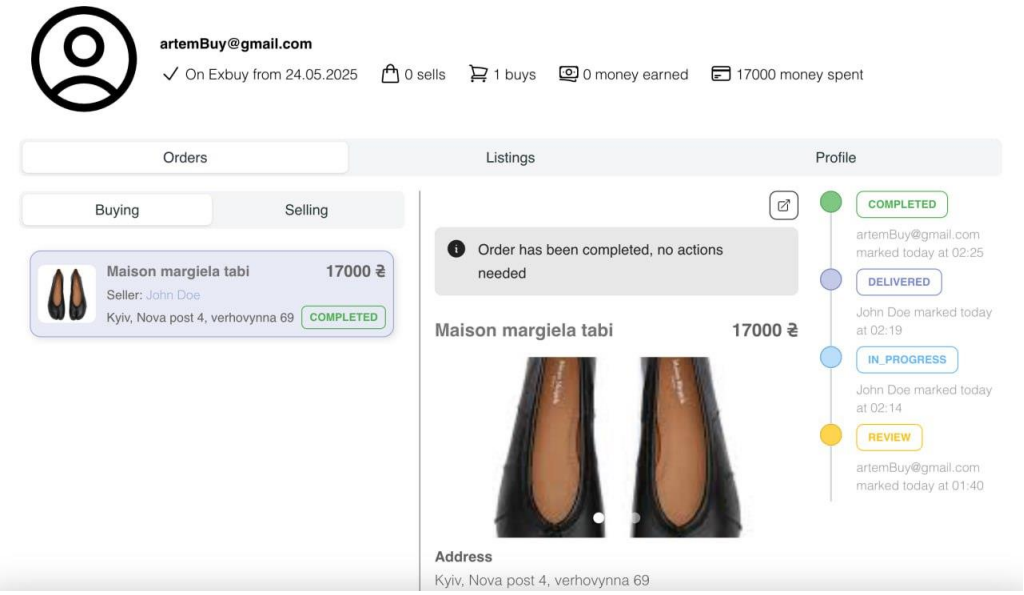

Рисунок 3.23 – Замовлення виконане у кабінеті покупця

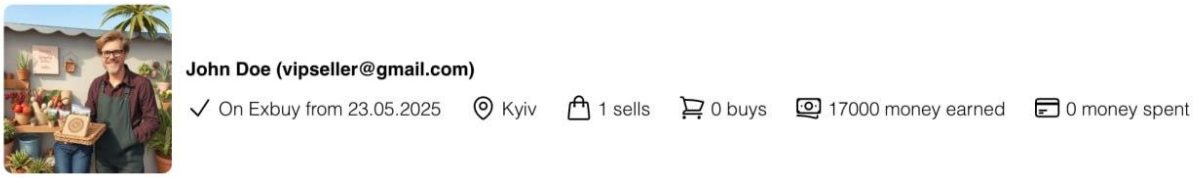

Рисунок 3.24 – Статистика продавця після виконання замовлення

Розглянемо випадок, коли замовлення відміняється. Для цього створимо замовлення зі сторони покупця, та перейдемо у акаунт продавця та натиснемо кнопку «Cancel order» (рис. 3.25). Після відміни замовлення неможливо продовжити зміну статусів, тому замовлення вважається архівним.

Зауважимо, що список замовлень стає більш наповненим – замовлення сортуються за останньою датою зміни статусу замовлення. Це допомгагає відслідковувати поточні замовлення, не шукаючи їх у списку.

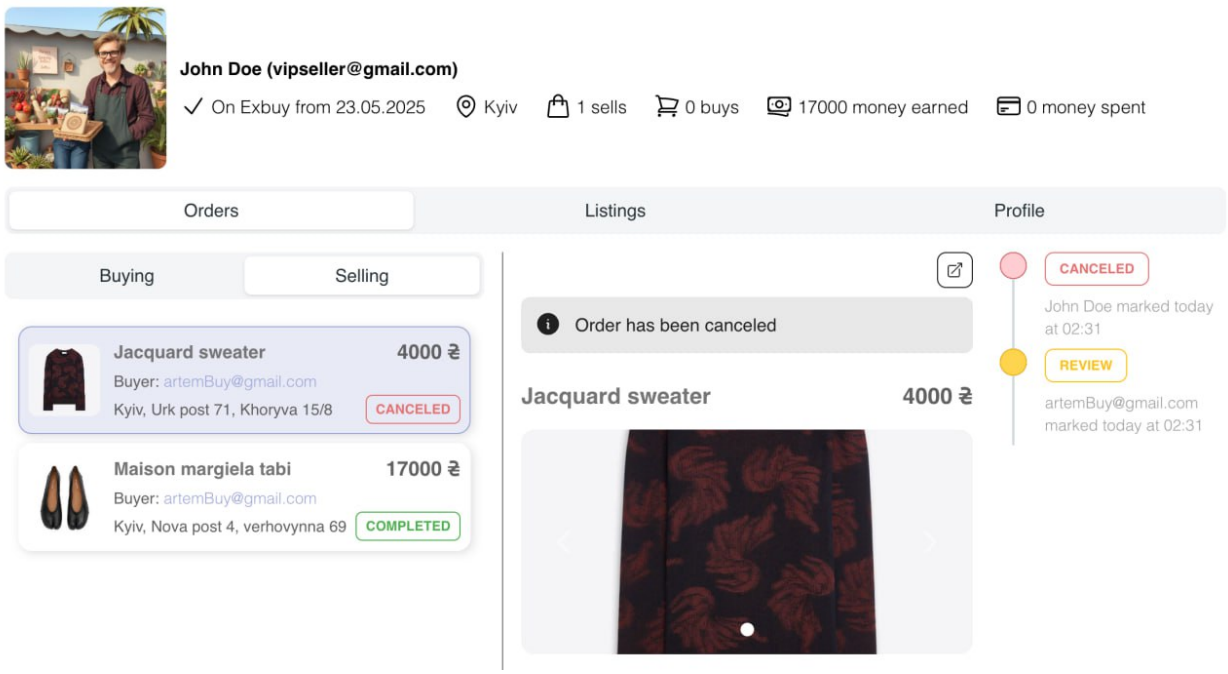

Рисунок 3.25 – Скасоване замовлення

Оскільки платформа одразу мала деякі оголошення, було розглянуто потік дій при замовленні. Але необхідним є доступ користувачів до створення власних оголошень. Для цього доступна кнопка «Add listing» завжди у шапці профілю (рис. 3.26).

Рисунок 3.26 – Кнопка створення оголошення

Натиснувши на кнопку, перейдемо до сторінки створення оголошення. На рисунку 3.27 показана форма для заповнення інформації. На рисунку 3.28 показана валідація форми.

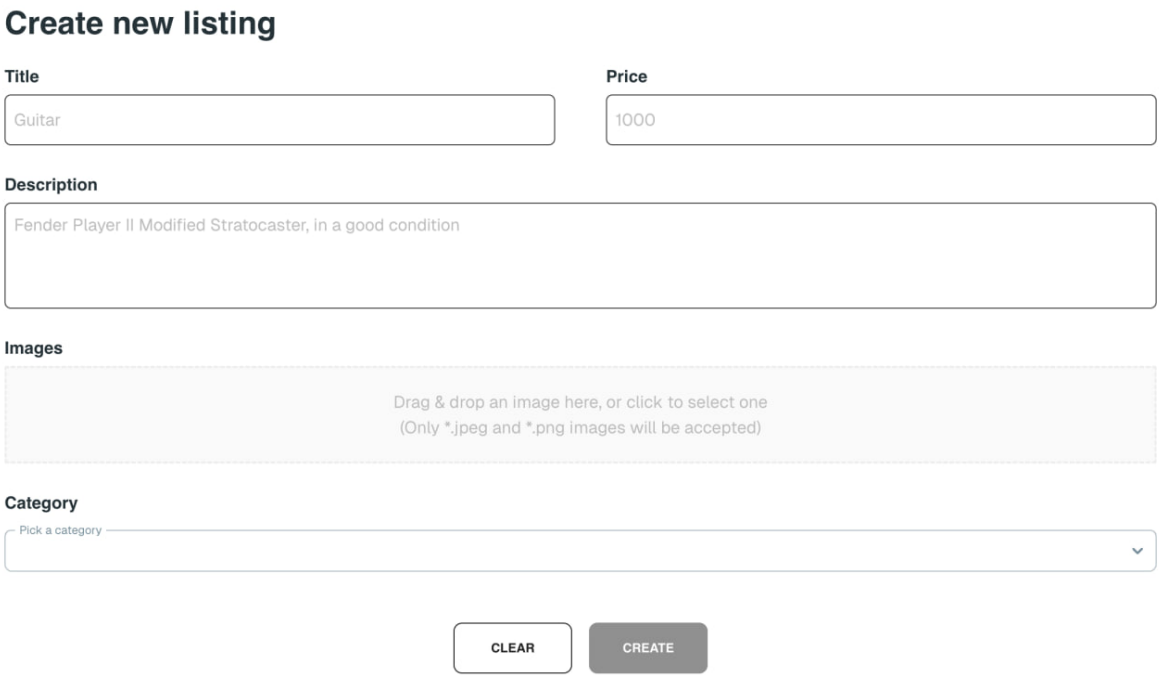

Рисунок 3.27 – Сторінка створення оголошення

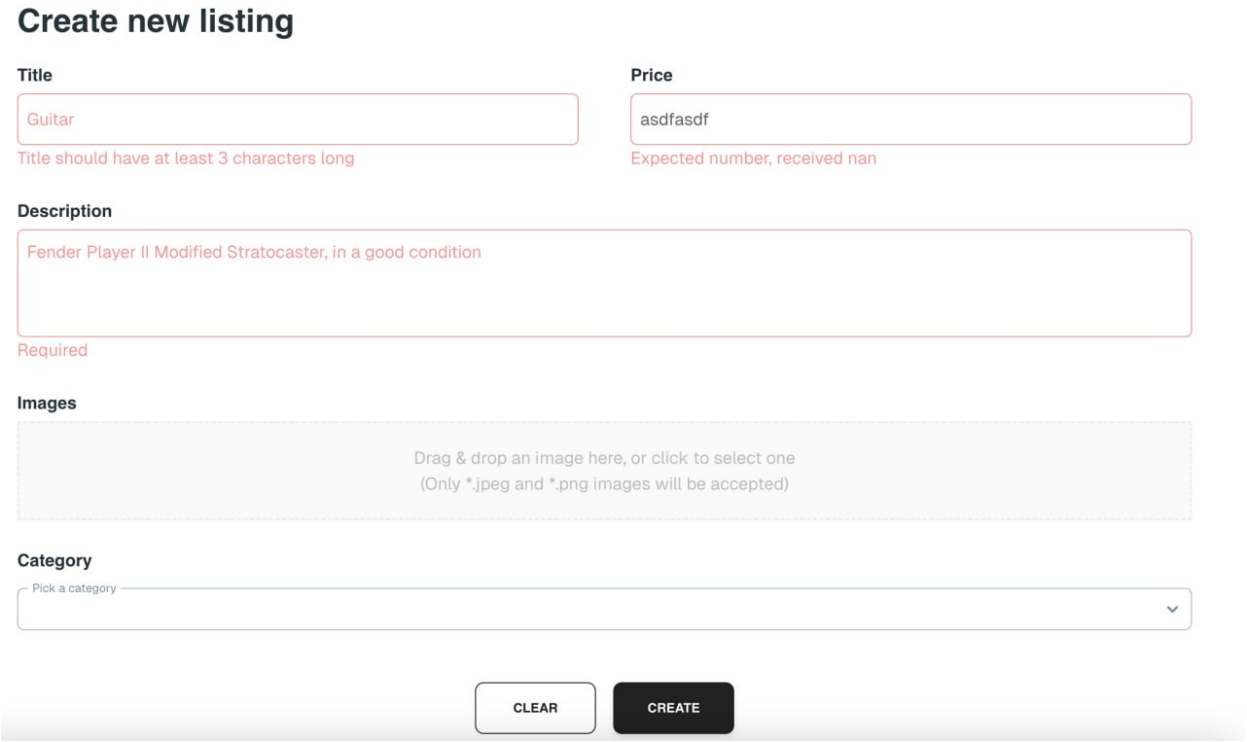

Рисунок 3.28 – Валідація при створенні оголошення

Заповнимо форму валідними даними, обравши бажані зображення (рис. 3.29). Їх може бути декілька, щоб покупці мали більше контексту про товар.

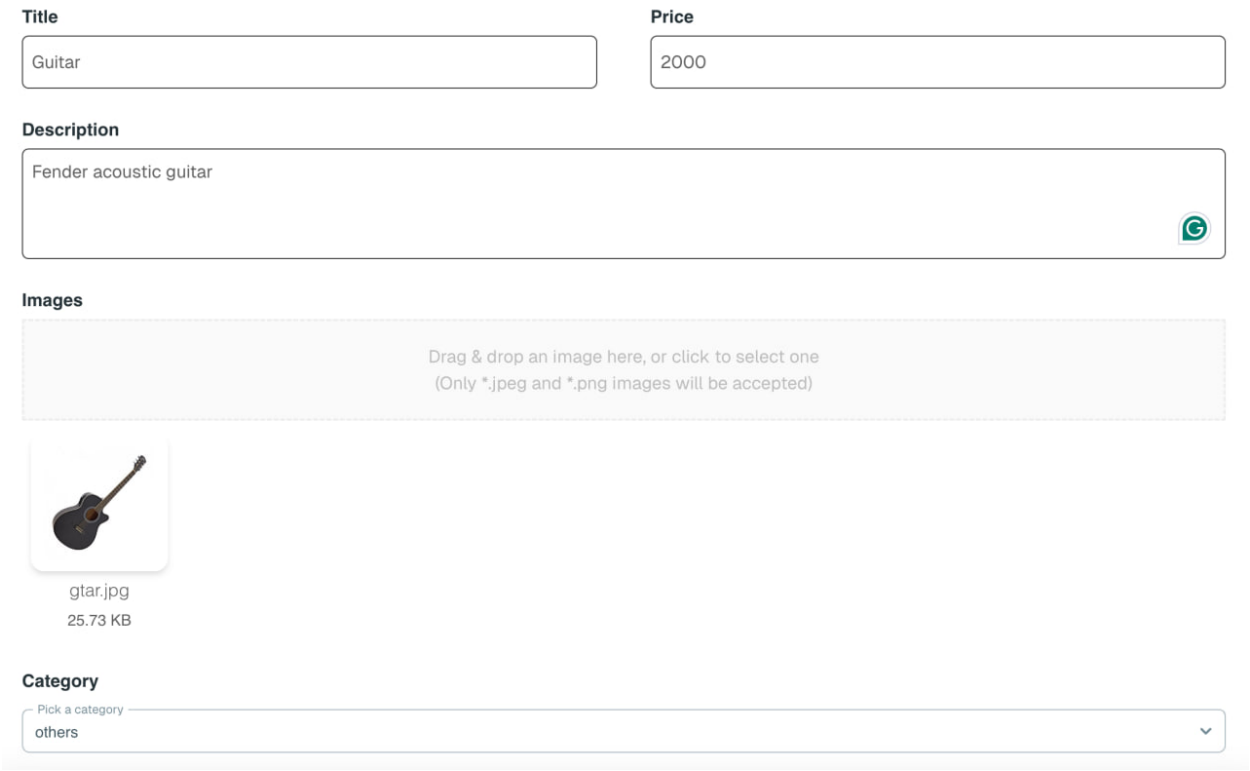

Рисунок 3.29 – Створення оголошення

Після цього продавець переадресований на сторінку щойно створеного оголошення (рис. 3.30). Зауважимо, що продавець не може купити власний товар. Кнопка відображається суто для показу того, що ця дія можлива лише для покупців. Як можемо зауважити, після виконання успішного замовлення, частина статистики відображається і в картці продавця на самій сторінці оголошення – кількість успішних замовлень.

Біля дати публікації оголошення позначений маркер у вигляді олівця, що означає можливість продавця редагувати своє оголошення. При чому, якщо на це оголошення зайде покупець, він не побачить цього маркеру

Перейдемо на сторінку з редагуванням оголошенням та побачимо форму для зміни. Заповнивши її зміненою інформацією (рис. 3.31), натиснемо на кнопку збереження.

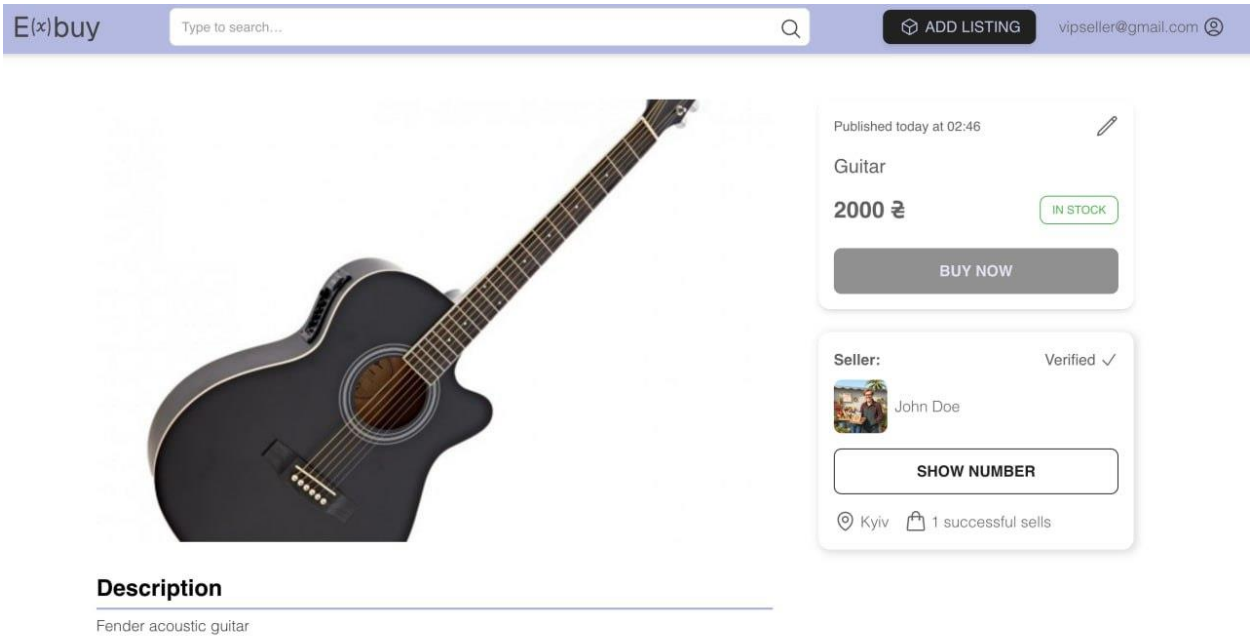

Рисунок 3.30 – Вид власного оголошення

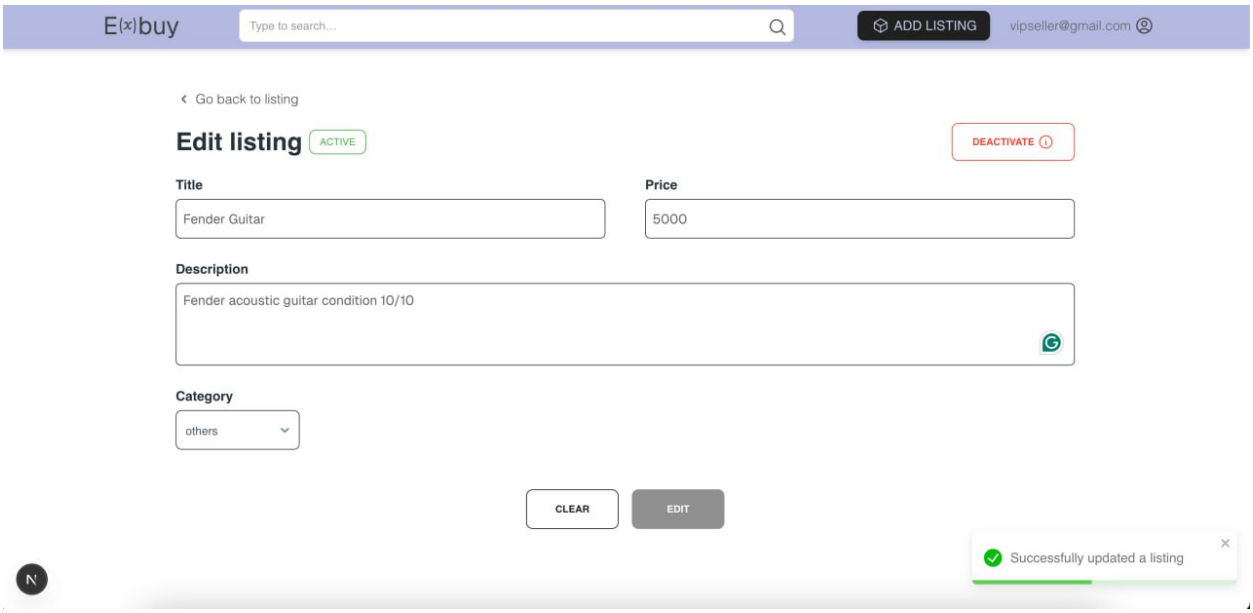

Рисунок 3.31 – Вид власного оголошення

Аналогічним чином можливо деактивувати оголошення (рис. 3.32). У такому випадку сторінку з самим оголошенням буде бачити тільки продавець. Для покупця не відобразиться це оголошення у категоріях, на головній сторінці у найновіших оголошеннях. Але деактивоване оголошення все ж таки може мати замовлення. Вони не видаляються через те, що продавець може деактивувати оголошення саме через існуюче замовлення.

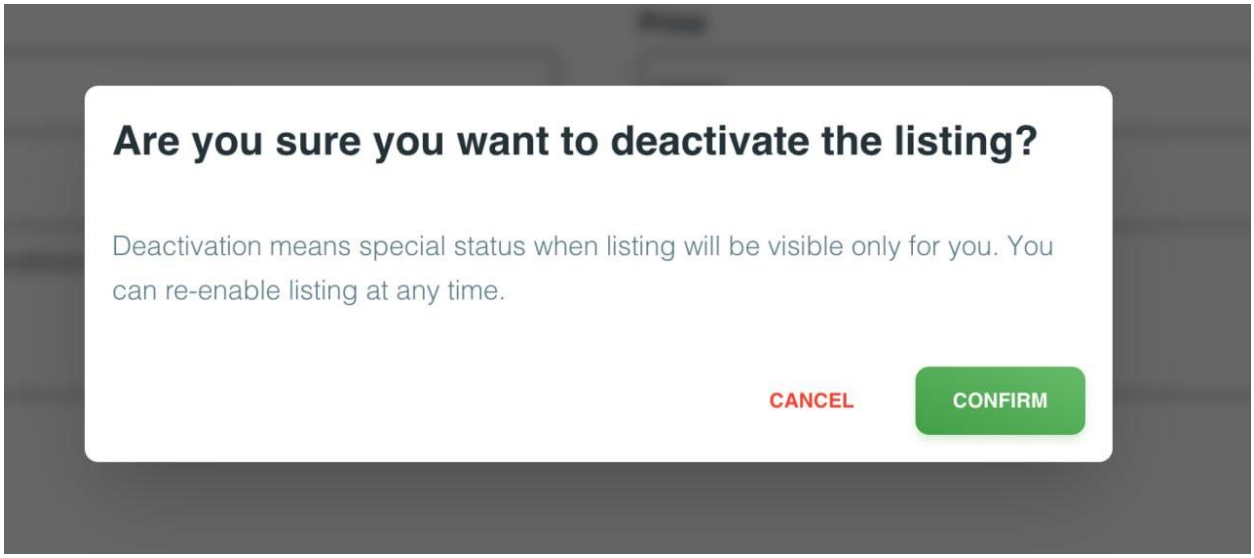

Рисунок 3.32 – Підтвердження деактивації оголошення

Спробуймо знайти оголошення у акаунті покупця через глобальний пошук (рис 3.33). Він відкривається тоді, коли є хоча б 1 символ, введений в навігацію в шапці профілю.

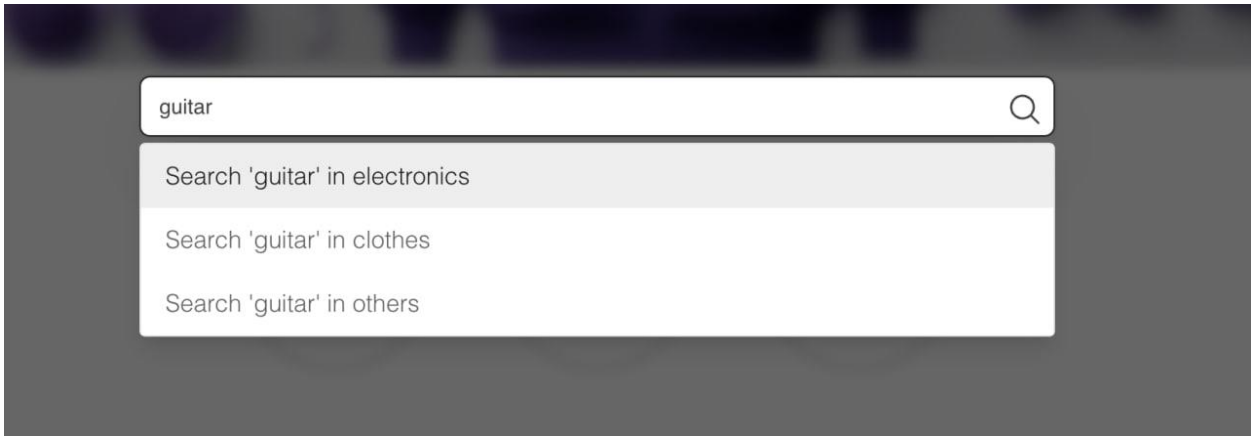

Рисунок 3.33 – Глобальний пошук

На рисунку 3.34 можна побачити, що пошук не дав результатів. Функціонал деактивації оголошення дуже зручний, коли треба редагувати оголошення, але через деякий проміжок часу. Це підкріплюється тим, якщо оголошення не є валідним через застарілу інформацію, тож простіше за все

його приховати від інших користувачів, щоб уникнути непорозумінь у процесі.

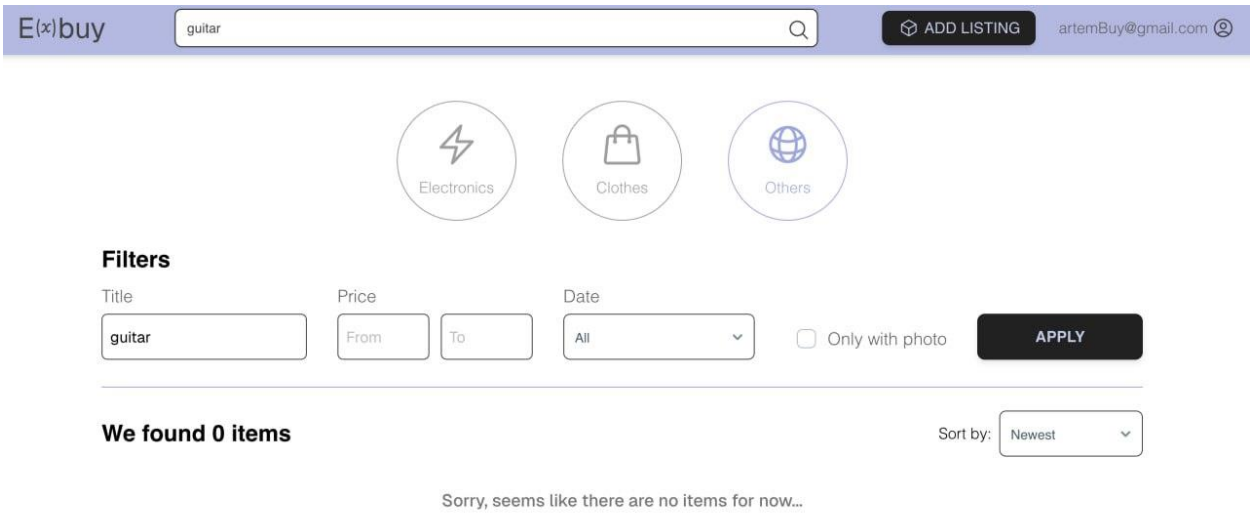

Рисунок 3.34 – Безрезультатний пошук

Останнім функціоналом є редагування профілю. Перейдемо до сторінки профілю і побачимо секцію редагування (рис 3.35).

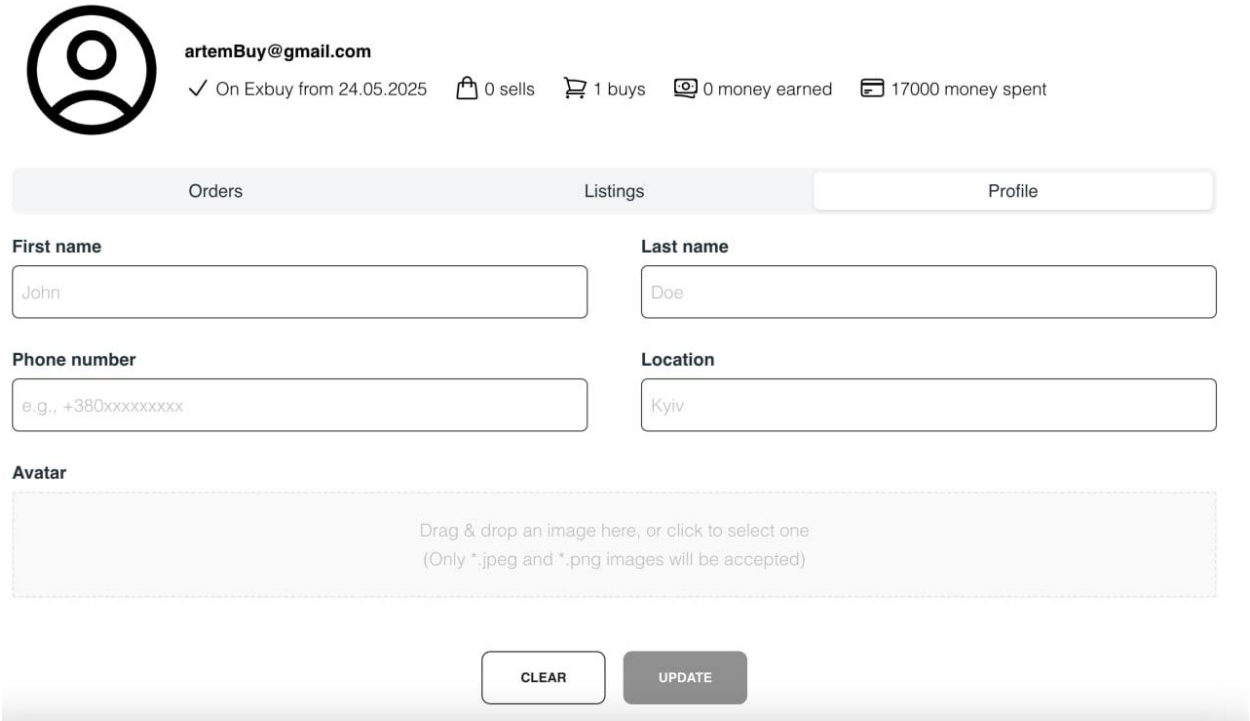

Рисунок 3.35 – Секція редагування профілю

Заповнимо профіль контактною інформацією та натиснемо кнопку «Update», щоб зберегти зміни. Результат оновлення можна побачити на рисунку 3.36.

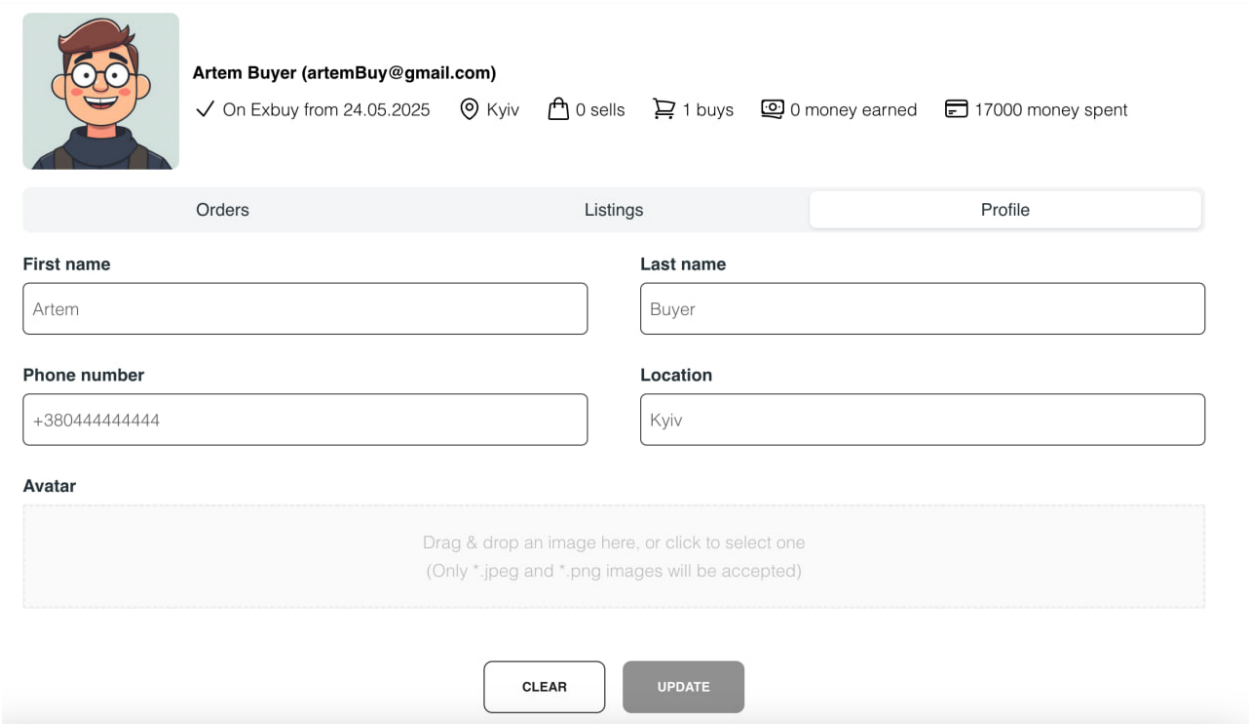

Рисунок 3.36 – Оновлений профіль

Таким чином, користувач має декілька шляхів того, як користуватись платформою. Користувач може як і перейти одразу до замовлень після швидкої реєстрації, так і створювати власні оголошення та редагувати свій профіль.

ВИСНОВКИ

У рамках кваліфікаційної роботи було розроблено та реалізований вебзастосунок дошки онлайн оголошень.

Метою кваліфікаційної роботи була розробка онлайн-сервісу для зручного пошуку та розміщення оголошень про продаж та купівлю товарів. Під час розробки було використано сучасні технології, такі як NextJS для клієнтської частини, NestJS для серверної частини, та PostgreSQL для менеджменту бази даних. Застосунок був успішно створений зі зручним користувацький інтерфейсом та сучасним дизайном. Роботу було успішно проаналізовано, як зі сторони продавця, так і покупця. Застосунок допомагає зручно шукати і замовляти товари, а процес створення оголошення є інтуїтивно простим, що економить час у процесі оформлення угод. Робота відповідає усім сучасним вимогам та може бути повноцінно використана для купівлі та продажу онлайн-товарів.

Результати роботи апробовано у вигляді тези доповіді під час Міжнародного молодіжного форуму «Радіоелектроніка і молодь у XXI столітті» [31].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Ūsas, A., Jasinskas, E., & Štreimikienė, D. (2024). Impact of website quality and user satisfaction on consumer loyalty in Lithuanian C2C E-commerce platforms. Management & marketing., 19(4), 710-724.

2. JHAWAR, Apoorav. C2C is a Business Model, but not popular. Why?. 2022.

3. Miao, L., So, K. K. F., Im, J., & Jiang, T. (2022). The pandemic's effects on customer-to-customer engagement in hospitality consumption: A multi-country investigation. *International Journal of Hospitality Management*, *102*, 103158.

4. Tokita, S. (2022, August). Institutional Logics Shaping the Behavior of Actors in C2C Handmade Market. In *2022 Portland International Conference on Management of Engineering and Technology (PICMET)* (pp. 1-10).

5. Castillo-Sotomayor, S., Guimet-Cornejo, N., & Lodeiros-Zubiria, M. L. (2023). C2C e-Marketplaces and How Their Micro-Segmentation Strategies Influence Their Customers. *Data*, *8*(2), 26.

6. Amplitude. (n.d.). *User behavior: What it is, types, and how to analyze it*. Amplitude. URL: https://amplitude.com/blog/user-behavior (дата звернення 10.04.2025).

7. Chen, Y., Lu, Y., Gupta, S., & Pan, Z. (2020). Understanding "window" shopping and browsing experience on social shopping website: An empirical investigation. *Information Technology & People*, *33*(4), 1124-1148.

8. Haugtvedt, C. P., Machleit, K. A., & Yalch, R. (2005). *Online consumer psychology: Understanding and influencing consumer behavior in the virtual world*. Psychology Press.

9. Mayayise, T. O. (2024). Investigating factors influencing trust in C2C e-commerce environments: A systematic literature review. Data and Information Management, 8(1), 100056.

10. Duong, N. T., Lin, H. H., Wu, T. L., & Wang, Y. S. (2024). Understanding Consumer Trust Dynamics and Purchase Intentions in a

Multichannel Live Streaming E-Commerce Context: A Trust Transfer Perspective. *International Journal of Human–Computer Interaction*, 1-14.

11. Orienteed. (n.d.). *Features of C2C eCommerce: The complete guide*. URL: https://orienteed.com/en/features-ecommerce-c2c/ (дата звернення 15.04.2025).

12. Wikipedia contributors. (n.d.). *OLX*. Wikipedia. URL: https://uk.wikipedia.org/wiki/OLX (дата звернення 16.04.2025).

13. Canvas Business Model. (n.d.). *Etsy: Brief history*. URL: https://canvasbusinessmodel.com/blogs/brief-history/etsy-brief-history (дата звернення 17.04.2025).

14. Encyclopaedia Britannica. (n.d.). *eBay*. Britannica. URL: https://www.britannica.com/money/eBay (дата звернення 19.04.2025).

15. Cortex. Monoliths vs. Microservices: Differences, Pros & Cons, and Choosing the Right Architecture. URL: https://www.cortex.io/post/monoliths-vs-microservices-whats-the-difference (дата звернення 24.04.2025).

16. Kalske, M. (2017). Transforming monolithic architecture towards microservice architecture. *University of Helsinki*.

17. Zhou, J. Y. (2004). *Functional requirements and non-functional requirements: a survey* (Doctoral dissertation, Concordia University).

18. Laplante, P. A., & Kassab, M. (2022). *Requirements engineering for software and systems*. Auerbach Publications.

19. Art of BA. (n.d.). *Основні типи архітектури програмного забезпечення*. URL: https://www.artofba.com/uk/post/main-types-of-software-architecture (дата звернення 02.05.2025).

20. Hamidli, N. (2023). Introduction to UI/UX design: key concepts and principles. *Preuzeto*, *28*, 2024.

21. CMSmart. (n.d.). *Designing a user-friendly marketplace interface*. URL: https://cmsmart.net/community/designing-a-user-friendly-marketplace-interface (дата звернення 03.05.2025).

22. Novak, T. P., & Hoffman, D. L. (1997). Measuring the flow experience among web users. *Interval Research Corporation*, *31*(1), 1-35.

23. Worsley, J., & Drake, J. D. (2002). *Practical PostgreSQL*. " O'Reilly Media, Inc.".

24. PostgreSQL Global Development Group. (n.d.). About PostgreSQL. PostgreSQL. URL: https://www.postgresql.org/about (дата звернення 05.05.2025).

25. Michaels, P. (n.d.). *The service-repository pattern*. P. Michaels. URL: https://pmichaels.net/service-repository-pattern/ (дата звернення 05.05.2025).

26. NestJS. (n.d.). *Modules*. NestJS. URL: https://docs.nestjs.com/modules (дата звернення 06.05.2025).

27. NestJS. *SQL (Sequelize)*. URL: https://docs.nestjs.com/recipes/sql-sequelize (дата звернення 08.05.2025).

28. DevZone. Як використовувати JSON Web Tokens (JWT) для автентифікації. URL: https://devzone.org.ua/post/iak-vykorystovuvaty-json-web-tokens-jwt-dlia-avtentyfikatsiyi (дата звернення 09.05.2025).

29. FreeCodeCamp. (n.d.). Routing in Next.js – How to Use App Router in your Next Apps. URL: https://www.freecodecamp.org/news/routing-in-nextjs/ (дата звернення 09.05.2025).

30. Meta Platforms, Inc. (n.d.). *createContext – React*. URL: https://react.dev/reference/react/createContext (дата звернення 10.05.2025).

31. Іващенко А.О. (2025) Розробка вебзастосунку «Дошка оголошень». *Радіоелектроніка і молодь у XXI столітті: тези доповідей 29-го Міжнародного молодіжного форуму (Харків, 16–19 квітня 2025 р.).* Харків: ХНУРЕ, 2025. Т.7. С. 55-57.