Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мережних технологій факультету інформатики

Розробка веб-сайту для пункту збору речей, пожертвуваних на благодійність

Текстова частина до курсової роботи

за спеціальністю „Інженерія програмного забезпечення ” 121

Керівник курсової роботи
кандидат фізико-математичних наук, старший викладач
Гречко А. В.

_____

(Підпис)
“___” _____ 2022 року
Виконала студентка ІПЗ-БП3
Колохіна К.І.
“___” _____ 2022 року

Київ 2022

Зміст_Toc105118226

Вступ *3*

РОЗДІЛ 1 Аналіз предметної області. Постановка завдання курсової роботи *5*

1.1 Аналіз сучасного стану питання та обґрунтування теми **5**

1.2 Огляд існуючих аналогів розробки **7**

1.3 Постановка задачі **8**

РОЗДІЛ 2 Теоретичні відомості *9*

2.1 MVC **9**

РОЗДІЛ 3 Опис реалізації програмного продукту *12*

3.1 Аналіз технічного завдання **12**

3.2. Обґрунтування вибору архітектури розробки **13**

3.3. Обґрунтування вибору засобів розробки **14**

3.4. Опис структури проекту **14**

3.5. Опис бази даних **16**

3.6. Тестування програми і результати її виконання **19**

3.6.1. Тестування авторизації **19**

3.6.2. Тестування сторінки «Отримати допомогу» **19**

3.6.3. Тестування додавання продукту адміністратором **20**

3.6.4. Тестування додавання нового продукту до заявки **21**

3.7. Інструкція користувача **23**

Висновки *40*

Список використаної літератури *41*

Додатки *42*

Додаток А. Файл з залежностями проекту. **42**

Додаток Б. Налаштування авторизації, автентифікації та доступів. **44**

Додаток С. Контроллер AuthController. **45**

Вступ

Декілька років тому у світі став популярним рух усвідомленого споживання. Передумовами для його появи стали збільшення обсягу речей, які купують люди, що в свою чергу призвело до екологічної проблеми, пов'язаної з великою кількістю утилізованих речей, які не знайшли собі господаря.

Одним з проявів цього руху стали магазини «другого життя». Такі магазини умовно поділяються на декілька категорій, одній з яких і присвячена тема роботи, а саме – благодійні магазини.

Поняття **благодійних магазинів** з'явилося ще задовго до виникнення проблеми перенасичення речами, однак здобуло свого піка популярності лише декілька років тому. Зазвичай благодійні магазини мають багато пунктів прийому речей, розміщених по місту, та декілька торгівельних приміщень, де виставляються на продаж зібрані речі або роздаються людям.

На жаль, теперішня ситуація у країні призвела до збільшення потоку людей до таких магазинів у пошуку потрібних речей. Більшість благодійних магазинів зараз перетворилися на волонтерські центри, які забезпечують своїми товарами велику кількість людей. Асортимент товарів став значно ширшим. Також, збільшилась і кількість заявок від людей на отримання допомоги. Проте не всі благодійні магазини мають електронні системи внутрішнього обліку товарів та прийому заявок від клієнтів, що призводить до збільшення часових витрат персоналу на виконання замовлень та знижує якість обслуговування. Таким чином створення для благодійних магазинів зручних систем управління розподілом безоплатних речей серед клієнтів зумовлює **актуальність даного дослідження.**

Метою роботи є створення веб-сайту благодійного магазину, з можливістю обліку речей, отриманих з пунктів прийому, та обробки великої кількості заявок на отримання допомоги.

Завданням роботи є розроблення веб-сайту благодійного магазину для інформування людей про можливу допомогу та подання відповідних заявок,

створення системи керування складом отриманих речей з пунктів прийому або акцій пожертвування з метою систематизації та обліку отриманих товарів.

Об’єктом дослідження є огляд існуючих веб-сайтів та додатків для вторинного прийому або продажу речей.

Предметом дослідження є створення відповідного веб-сайту благодійного магазину з можливістю подання заявок на допомогу онлайн та системи обліку товарів складу.

Для розробки веб-сайту використано наступне програмне забезпечення: фреймворк Spring Web для розробки клієнт-серверної частини; бібліотека Bootstrap та шаблонізатор Thymeleaf для оформлення сторінок.

Робота складається зі вступу, трьох основних розділів, висновків та додатків до них. Перший розділ висвітлює огляд існуючих аналогів та постановку задачі. У другому розділі розглядаються методи та підходи до її розв’язку. У третьому розділі описується реалізація створення веб-застосунку.

РОЗДІЛ 1 Аналіз предметної області. Постановка завдання курсової роботи

1.1 Аналіз сучасного стану питання та обґрунтування теми

Проблема надмірного виробництва та, як наслідок, надмірного споживання є актуальною в світі останні двадцять років, оскільки призводить до передчасного виснаження природних ресурсів, збільшення забруднення ґрунту, повітря та води.

Майже кожна людина насправді купує набагато більше речей ніж їй потрібно. Наприклад, за результатами досліджень британських вчених [1] кожна дитина у віці десяти років у середньому має 238 іграшок, але грається щодня лише з 12-ма. Кількість торгових центрів перевищує кількість середніх шкіл, а 93% дівчат-підлітків вважають шопінг своїм улюбленим заняттям.

Однак, більшість придбаних речей згодом утилізується за своєю непотрібністю. Згідно з даними дослідників, в середньому за рік по всьому світі викидається 92 млн тон одягу. Експерт прогнозують, що через декілька років ця цифра може зрости принаймні у півтора рази.

Дану проблему можна вирішити, якщо зменшити обсяги покупок та перейти на так зване «раціональне споживання».

З 2020 року даний тренд набув досить швидкого поширення. На думку вчених [2]: «У його основі лежить ідея споживання, що задовольняє потреби нашого і прийдешніх поколінь економічно, соціально та екологічно збалансованим шляхом. Таке збалансоване споживання передбачає зменшення обсягу споживання, а отже, і відходів; надання переваги екологічно чистій продукції; відмову від використання агресивних хімікатів тощо. Змінюється й загальна модель споживання: споживачі надають перевагу товарам місцевого виробництва, товарам довготривалого використання, вводиться «міра» у споживанні, тобто виважене ставлення до придбання нових речей та використання вже вживаних речей».

Ідея повторного використання вживаних речей вже реалізується шляхом відкриття магазинів, що продають товари секонд хенд. За останні роки кількість таких магазинів швидко збільшується. В Україні окрім класичних магазинів з продажу вживаних речей здобули свого розвитку ще й благодійні.

Зазвичай такі магазини мають декілька пунктів прийому речей від населення, розміщених у містах та один або декілька безпосередньо фізичних магазинів, де виставляються на продаж зібрані речі (точніше ті, які підлягають продажу). В Україні таких благодійних магазинів існує декілька, усі вони мають сайти-візитівки, однак ніхто з них не запровадив онлайн торгівлю.

У ситуації війни, яка зараз триває в Україні, торгівля стала другорядним призначенням благодійних магазинів. Більшість з тих, що продовжують працювати, перетворилися на волонтерські центри для прийому та видачі речей вимушеним переселенцям. Під час війни рівень взаємо підтримки в українському суспільстві значно підвищився. Українці активно допомагають один одному і передають у благодійні магазини велику кількість своїх речей, які можуть знадобитися тим, хто був змушений покинути свої домівки чи втратив своє майно через бойові дії.

В той же час в існуючих благодійних магазинах системи обліку та управління товарообігом, процедури прийняття і обробки заявок на допомогу не були розраховані на великий потік людей. З огляду на це, гостро постало питання автоматизації роботи.

Тому, створення веб-сайту зі спеціалізованою системою обліку отриманих речей та єдиним алгоритмом обробки отриманих заявок – актуальна розробка, яка допоможе благодійним магазинам вийти на новий рівень та пришвидшити свою роботу.

1.2 Огляд існуючих аналогів розробки

У рамках цієї роботи було проаналізовано дві системи складського обліку, 1С:Підприємство та BAS. Для аналізу були використані матеріали [3 – 4].

«1С:Управління нашою фірмою 8» («1С:УНФ») – готове комплексне рішення для управління та обліку у малому бізнесі. Все необхідне міститься в одній програмі:

- CRM;
- Закупівлі та продажу;
- Склад;
- Виробництво;
- Фінанси та зарплата;
- Регламентована звітність;
- Аналіз стану підприємства.

УНФ підходить усім підприємствам, які працюють у сферах:

- оптової та роздрібної торгівлі;
- надання послуг;
- інтернет-торгівлі;
- дрібносерійного та позамовного виробництва;
- виконання підрядних робіт.

"BAS Управління торгівлею" – сучасний інструмент для підвищення ефективності бізнесу торгового підприємства. За допомогою цієї програми можливо:

- збільшити КПД праці служб торгового підприємства
- зберігати та опрацьовувати інформацію, яка відображає поточний стан підприємства;
- швидко формувати звіти у зручному вигляді;
- вести детальний та оперативний облік товарів на усіх складах;

- мати повний контроль над кількістю товарів підприємства.

Мінуси у обох продуктів:

- Досить висока ціна конфігурації та дороге щомісячне обслуговування;
- Суворі системні вимоги;
- Достатньо складний інтерфейс, що вимагає додаткового навчання;
- Відсутність пробної версії для безкоштовного тестування функціоналу;
- Безліч обслуговуючих компаній. Дуже складно вибрати кваліфіковану супроводжуючу компанію, яка б надавала якісну та оперативну підтримку.
- Платні поновлення.

На жаль, популярні готові продукти для управління та обліку у малому бізнесі не підходять для благодійних організацій та магазинів, адже є дорогими у щомісячному обслуговуванні та дуже навантажені зайвим функціоналом. Потрібна система, яка буде наближеною до аналогів та легкою в управлінні.

1.3 Постановка задачі

В результаті проведеного дослідження існуючих на ринку систем аналогів вирішено, що для створення системи ведення обліку пожертвуваних речей і обробки запитів на їх видачу необхідно реалізувати такі функціональні можливості:

- Реєстрація нових користувачів (людей, потребуючих допомоги) в системі;
- Автентифікація користувачів у системі;
- Формування користувачами запитів на отримання певних речей з наявних у БД пропозицій;
- Ведення облік пожертвуваних речей за певними категоріями та відповідно до їх типу, сезону, статі;
- Контролювання обсягів отриманих речей.

РОЗДІЛ 2 Теоретичні відомості

2.1 MVC

MVC(абрев. Model–View– Controller) [5] – це спосіб організації коду, який ділить застосунок на пов’язані між собою компоненти, які відповідають за виконання різних задач.

Model (Модель) – компонент даних. Містить всю інформацію, важливу для відображення, її вимоги щодо доступу та інших перевірок.

View (Представлення) – це інтерфейс, усі сторінки. Його задача це взаємодія з користувачем.

Controller (Контролер) – головний елемент, що пов’язує між собою Model та View, обробляє усі запити користувача, які надходять з інтерфейсу.

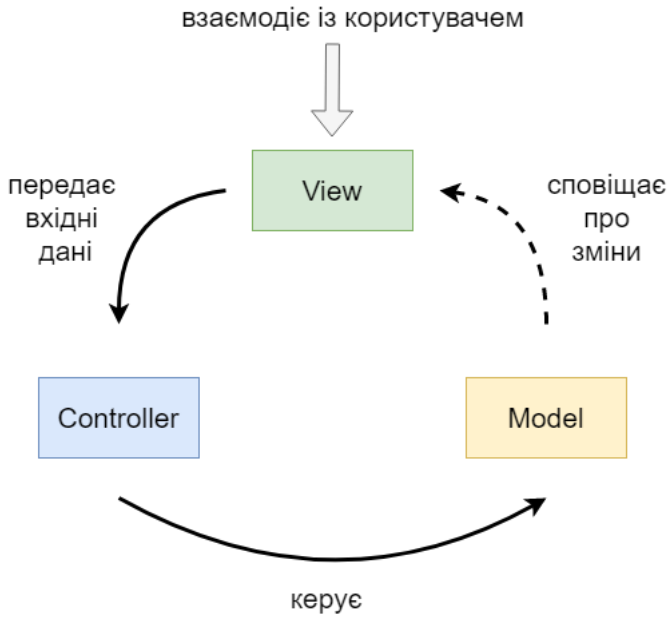

Рис. 1 - Діаграмма взаємодії між компонентами шаблону MVC

Цикл роботи сайту згідно з цим шаблоном виглядає наступним чином:

1. Користувач заходить на сайт за посиланням. Це відправляє відповідний запит до контролеру.
2. Контролер починає оброблювати запит, звертається до моделі.
3. Модель звертається до бази даних, отримує потрібну інформацію, вона передається знову до контролеру.

4. Контролер підставляє отримані дані у представлення (зазвичай це шаблон веб-сторінки з підготовленими полями для даних з моделі).
5. Представлення (веб-сторінка) відображається користувачу.

Існує багато реалізацій цього шаблону у різних мовах програмування, однак для нашого застосунку було обрано фреймворк Java Spring MVC.

Детально логіка роботи фреймворку описана у статті [6]. Загалом, вона побудована навколо DispatcherServlet який опрацьовує усі HTTP-запити з інтерфейсу і їх відповіді. Шлях від запиту користувача до результуючої веб-сторінки у цьому випадку трохи відрізняється від описаної раніше абстрактної схеми і виглядає наступним чином:

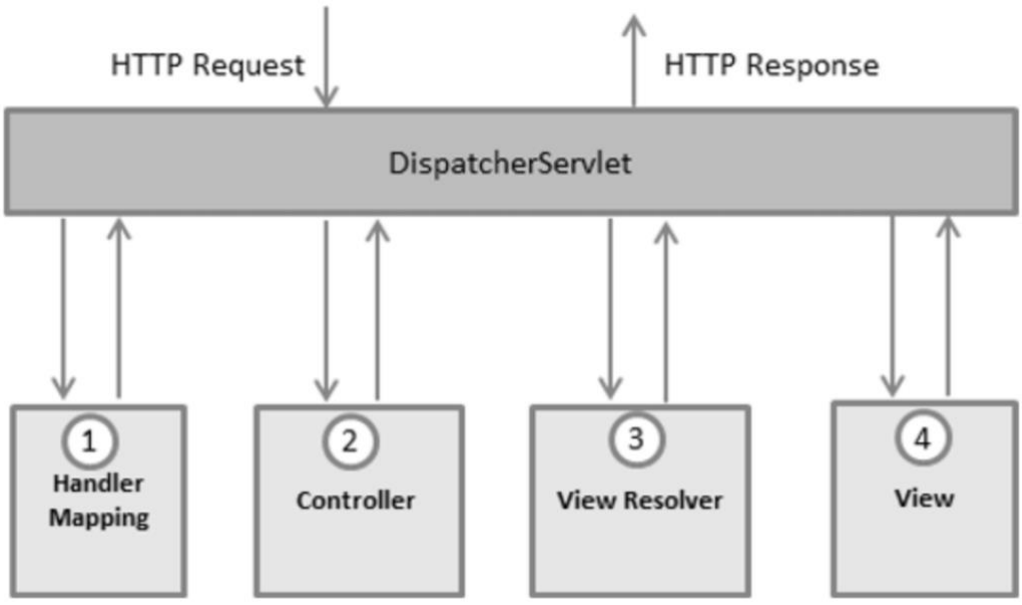

Рис. 2 - Схема роботи шаблону у фреймворці Spring MVC

1. Запит потрапляє до DispatcherServlet, який звертається до інтерфейсу HandlerMapping. Він визначає контролер, якому повинен бути переданий запит.
2. В залежності від запиту і його типу викликається певний метод контроллера.
3. Контролер у рамках обраного методу звертається до сервісів, щоб отримати дані, потрібні для визначення пов’язаної з методом контролера

моделі і вказує яке відображення треба побудувати у результаті. Модель з ім’ям відображення передаються у DispatcherServlet.

4. Базуючись на імені відображення інтерфейс ViewResolver повертає потрібний шаблон веб-сторінки.
5. Після цього, DispatcherServlet відправляє дані моделі у вигляді атрибутів у шаблон сторінки.
6. Сторінка будується (у вигляді об’єкту Response, який надсилається браузеру) і відображається користувачу.

РОЗДІЛ 3 Опис реалізації програмного продукту

3.1 Аналіз технічного завдання

Розроблене в даній курсовій роботі веб-застосування складається з двох частин – системи для ведення обліку пожертвуваних речей та інтерфейсу прийому, обробки та обліку заявок на отримання гуманітарної допомоги.

Отже, маємо три категорії користувачів: адміністратори застосунку, незареєстровані користувачі, зареєстровані користувачі.

Адміністратори можуть:

- Реєструвати надходження нових пожертвуваних речей;
- Переглядати поточний список речей на складі, вносити зміни, видаляти;
- Створювати заяви замість користувачів;
- Переглядати та обробляти заявки користувачів на допомогу, списувати потрібні речі у рамках цих заявок;
- Редагувати та створювати нові категорії товарів, які може приймати організація.

Незареєстровані користувачі можуть :

- Переглядати основну інформацію про організацію (що за організація, яка місія, де розташовані пункти прийому та графік їх роботи, як можна матеріально допомогти);
- Подати заявку на допомогу (тим самим ставши зареєстрованим користувачем);

Зареєстровані користувачі мають ті ж самі можливості що і незареєстровані, але з невеличким доповненням у вигляді авторизації та перегляду статусу своїх заявок.

Усі данні про користувачів, адміністраторів, речі на складі та заявки зберігаються у базі даних.

Загалом, застосунок можна поділити на декілька розділів:

«Інформація про організацію» – набір статичних сторінок «Про нас», «Де знайти», «Як підтримати» та динамічна сторінка «Отримати допомогу».

«Панель адміністратора» – набір динамічних сторінок для керування складом та заявками, перегляду поточної інформації по складу, управління заявками та їх перегляд. Панель захищено попередньою авторизацією та автентифікацією через форму входу.

«Панель користувача» – перегляд поточних заявок, внесення до них змін. Панель захищено попередньою авторизацією та автентифікацією через форму входу.

3.2. Обґрунтування вибору архітектури розробки

Враховуючи те, що головною перевагою розробленого веб-додатку повинна бути швидкість роботи і те, що заздалегідь не відомо обчислювальні потужності пристроїв, на яких буде використовуватись додаток, для розробки була обрана багаторівнева архітектура (а саме трирівнева – клієнт, сервер та БД).

Вказана архітектура передбачає перенесення всіх операцій з клієнту на сервер, що суттєво зменшує вимоги до пристроїв користувача.

Також, розроблений веб-застосунок передбачає зберігання персональної і іншої важливої інформації, локальне зберігання якої на пристроях адміністраторів неможливе за законом. Трирівнева ж архітектура вимагає зберігання усієї інформації у базі даних та отримання її за потреби, що обмежує доступ, тип самим захищає дані користувачів.

Розроблене веб-застосування обов’язково повинно працювати без перебоїв та затримок, що багаторівнева архітектура може забезпечувати шляхом збільшенні кількості серверів і використанням, наприклад, балансування або розділення потоків роботи.

Чимало важливо і те, що додаток розроблений за цією архітектурою легко масштабувати і налаштовувати за рахунок ізольованості рівнів.

3.3. Обґрунтування вибору засобів розробки

Веб-застосування побудовано на основі патерну багаторівневої архітектури і включає три рівні - клієнт, сервер та БД [7].

База даних

З огляду на те, що інформація, яку потрібно зберігати є дуже структурованою і містить достатньо багато зв'язків, було обрано реляційну БД, а саме MySQL.

Клієнт-серверна частина

Нажаль, через брак досвіду роботи з frontend фрймворками, від варіанту проектування клієнтської частити додатку використовуючи Node.Js або Angular довелося відмовитись.

Натомість, клієнт-серверну частину було реалізовано за допомогою фрейм-ворку Spring Web (і додаткових бібліотек Spring Security, Spring Data JPA) , адже він містить усі необхідні компоненти для створення веб-застосунку.

Також, для реалізації сторінок було використано шаблонізатор ThymeLeaf (для динамічної генерації сторінок) та бібліотеку Bootstrap (задля пришвидшення розробки самих сторінок та зберігання єдиного стилю їх оформлення).

Аргументів для вибору описаних вище засобів розробки було декілька – наявність великого досвіду роботи з вказаними засобами та їх велика популярність й актуальність на ринку.

3.4. Опис структури проекту

Клієнт-серверну частину застосування було реалізовано за допомогою фрейм-ворку Spring Web, відповідні файли зберігаються разом у одному проекті, структуру якого наведено на рис.3.

Модуль com.naukma.kursova відповідає за роботу сервера. У папці resourses-tempates зберігаються шаблони сторінок веб-застосування (рис.4).

У папці entity зберігаються представлення відповідних реляцій бази даних і трансферних об'єктів (рис.5). Зв'язок з базою реалізовано через репозиторії

(пакет com.naukma.kursova.repository). Налаштування підключення до бази даних прописані у файлі application.properties.

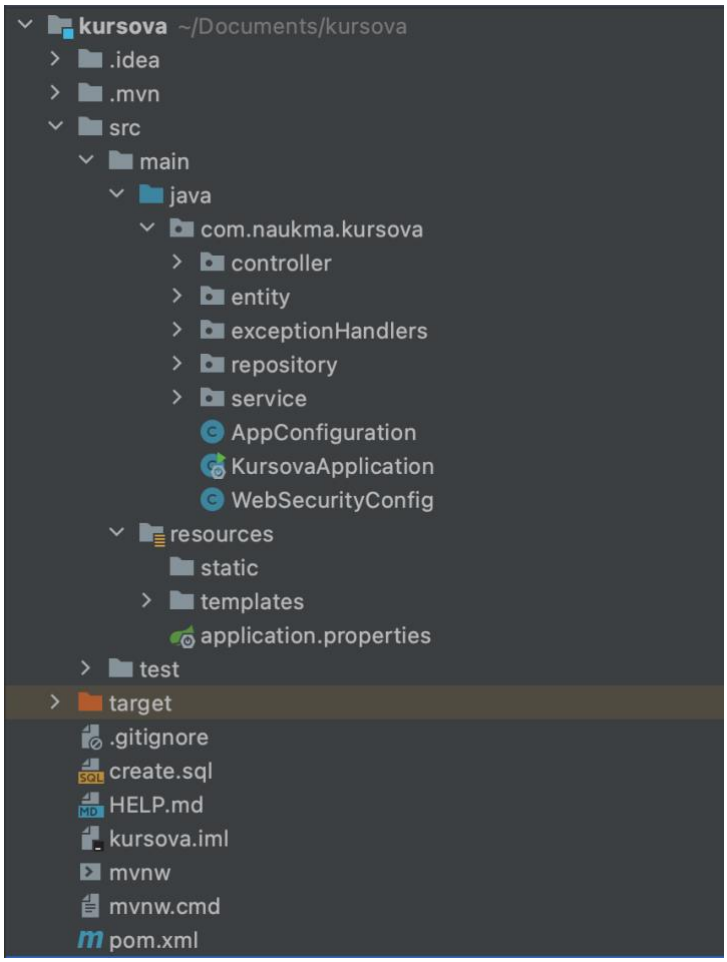

Рис. 3. Структура проекту

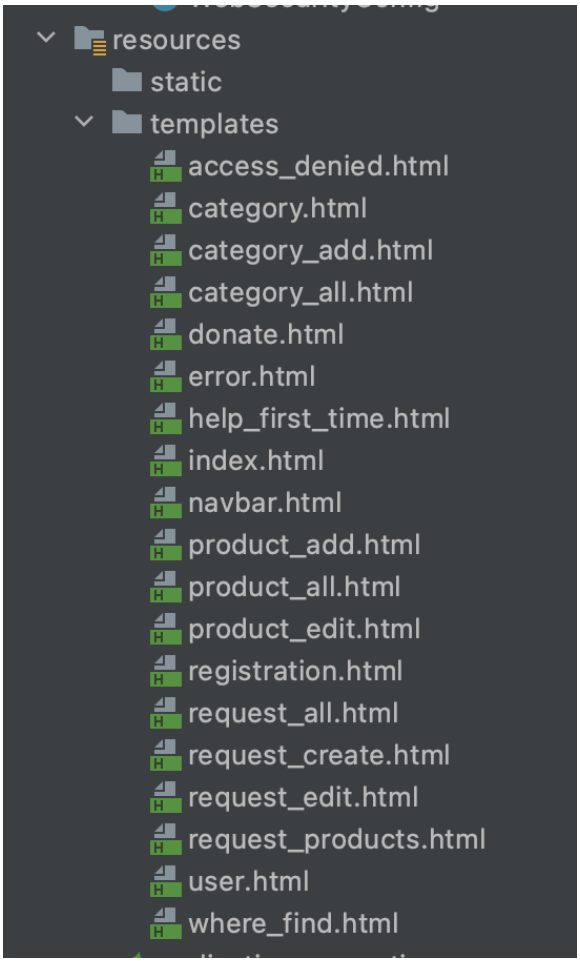

Рис. 4. Шаблони сторінок

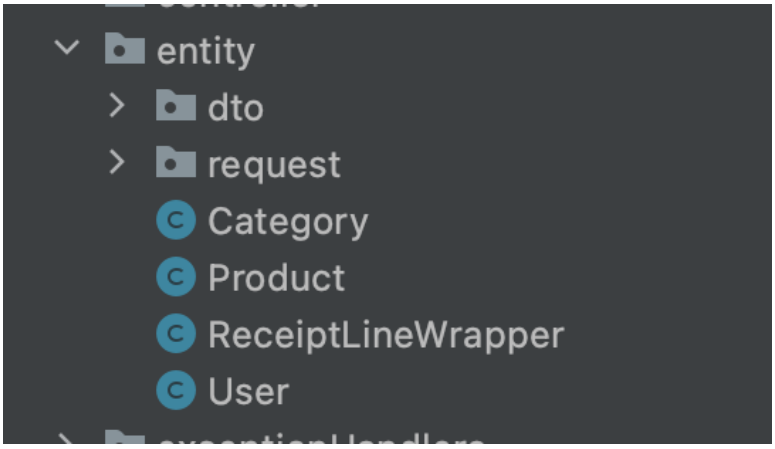

Рис. 5. Представлення реляцій БД у проекті

Схема взаємодії сторінок інтерфейсу зображена на рис. 6.

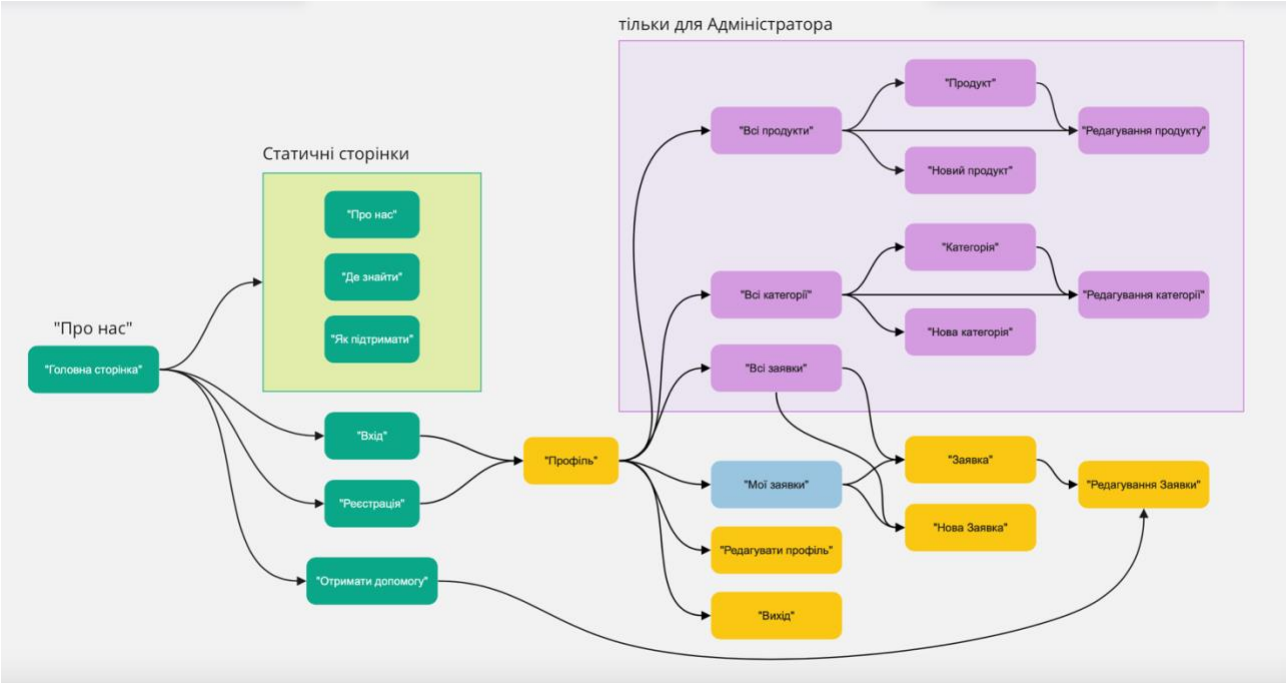

Рис. 6. Схема взаємодії сторінок інтерфейсу

3.5. Опис бази даних

Схему бази даних та зв’язки між її елементами зображено на рис.7.

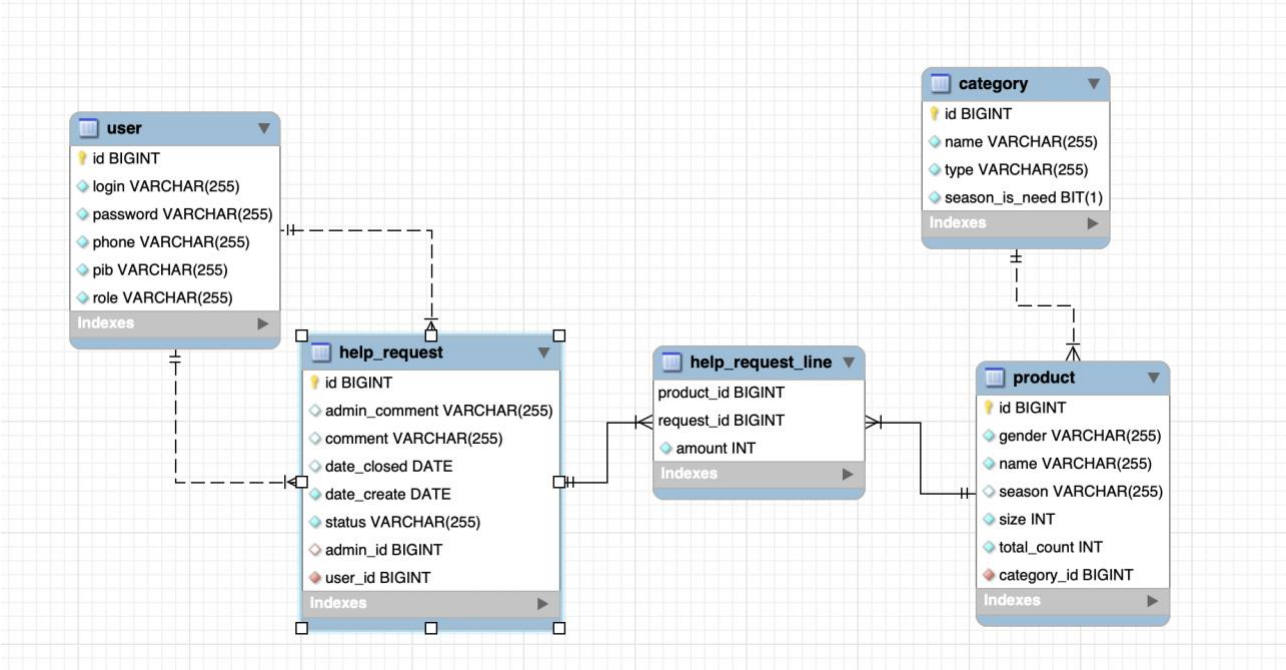

Рис. 7. Схема Бази Даних

Опис таблиць:

1. **“user”** – таблиця, що містить загальні дані про адміністраторів та користувачів сайту. Первинний ключ – id. Альтернативні ключі –login та pib.

   Атрибути:

   id –унікальний номер користувача у БД;

   pib – ПІБ користувача;

   login – пошта або номер телефону;

   password – пароль (зашифрований технологією BCrypt);

   phone – контактний телефон;

   role – роль користувача – ADMIN або USER.

2. **“help_request”** – таблиця, що містить загальні дані про заявки на гуманітарну допомогу. Первинний ключ – id. Зовнішні ключі – user_id, admin_id.

   Атрибути:

   id –унікальний номер заявки у БД;

   date_create – дата створення;

   date_closed –дата закриття;

   user_id – зовнішній ключ, вказує на користувача, що оформив заявку;

   admin_id – зовнішній ключ, вказує на адміністратора, що завершив заявку чи останнім вносив зміни до неї;

   comment – додаткові дані;

   admin_comment – коментарі адміністратора;

   status – статус заявки (OPEN, IN PROGRESS, CLOSED);

3. **“help_request_line” –** таблиця, що містить інформацію про потрібні речі, вказані у заявці. Первинний ключ (product_id, request_id). Зовнішні ключі – request_id, product_id.

   Атрибути:

   product_id – частина первинного ключа, зовнішній ключ, вказує на номер продукту;

   request_id – частина первинного ключа, зовнішній ключ, вказує на номер заявки;

   amount – кількість вказаного товару, потрібна людині.

4. **“product”** – таблиця, що містить загальні дані про усі речі, передані організації. Первинний ключ – id. Зовнішні ключі – category_id

   Атрибути:

   id –унікальний номер одиниці одежі у БД;

   name – назва речі.

   category_id – зовнішній ключ, посилання на категорію товару;

   gender – гендер;

   season – сезон (осінь, зима, весна, літо);

   size – розмір;

   total_count – загальна кількість на складі.

5. **“category”** – таблиця, що містить усі категорії речей. Первинний ключ – id.

   Атрибути:

   id –унікальний номер категорії у БД;

   name – назва;

   type – тип речі (взуття або одяг).

   season_is_need – булеве поле, вказує чи потрібно обирати сезон для продуктів, які відносяться до цієї категорії.

3.6. Тестування програми і результати її виконання

3.6.1. Тестування авторизації

Номер тесту	A1
Відповідальний за тест	Колохіна Катерина
Дата проведення тесту	21.04.2022
Послідовність дій	• Відкрити головну сторінку системи, натиснути кнопку «Вхід» у верхньому меню. • Ввести логін та пароль для авторизації адміністратора чи користувача. • Натиснути кнопку «Увійти» (рис.17).
Очікуваний результат	Якщо введено некоректний логін або пароль, сторінка перезавантажиться і над формою з’явиться попередження - з’явиться повідомлення «Bad Credentials». Якщо дані для входу введені вірно, система переведе користувача на сторінку профілю Адміністратора або Користувача, в залежності від ролі (рис. 18 або рис. 35).

3.6.2. Тестування сторінки «Отримати допомогу»

Номер тесту	A2
Відповідальний за тест	Колохіна Катерина

Дата проведення тесту	22.04.2022
Послідовність дій	• Відкрити головну сторінку системи, перейти у меню за посиланням «Отримати допомогу». (рис. 11). • Заповнити форму внизу сторінки (рис.12). • Натиснути кнопку «Далі».
Очікуваний результат	• Якщо введено некоректні данні, з’явиться спливаюче віконце з описом проблеми (рис.13). • Якщо користувач з таким логіном вже існує, користувача буде переведено на сторінку помилки (рис.14). • Якщо дані для входу введені вірно, система зареєструє користувача та переведе на сторінку перегляду нової заявки (рис.15).

3.6.3. Тестування додавання продукту адміністратором

Номер тесту	А3
Відповідальний за тест	Колохіна Катерина
Дата проведення тесту	19.04.2022
Послідовність дій	• Відкрити головну сторінку системи, натиснути кнопку «Вхід» у верхньому меню. • Авторизуватись, як Адміністратор.

	• У меню обрати «Продукти» (рис. 24). Натиснути кнопку «Створити новий». • Послідовно заповнити потрібні поля (рис.25). • Натиснути кнопку «Створити».
Очікуваний результат	• Якщо намагатись створити вже існуючий товар (введена кількість на складі не важлива), то користувач побачить вікно помилки (рис. 28). • Якщо товар був створений успішно, користувача поверне на сторінку «Усі продукти» (рис.24), а продукт буде відображатись у списку.

3.6.4. Тестування додавання нового продукту до заявки

Номер тесту	A4
Відповідальний за тест	Колохіна Катерина
Дата проведення тесту	19.04.2022
Послідовність дій	• Відкрити головну сторінку системи, натиснути кнопку «Вхід» у верхньому меню. • Авторизуватись, перейди до сторінки «Всі заявки/Мої заявки» через верхнє меню. Обрати заявку та натиснути «Редагувати».

	• У другому розділі «Додати продукт» (рис.33) обрати зі списку продукт та ввести бажану кількість. • Натиснути кнопку «Зберегти» яка знаходиться у вказаному розділі.
Очікуваний результат	• Якщо вказана кількість більша, ніж товару наявно на складі, з’явиться спливаюче віконце з попередженням (рис. 34). • Якщо обрана кількість менше або дорівнює кількості товару на складі, товар буде успішно додано, користувач побачить сторінку «Перегляд заявки», де товар з’явиться у таблиці.

3.7. Інструкція користувача

Після відкриття сайту у браузері, перша сторінка яку побачить користувач, буде «Про нас».

«Про нас» - головна сторінка, містить опис діяльності організації (рис.8) .

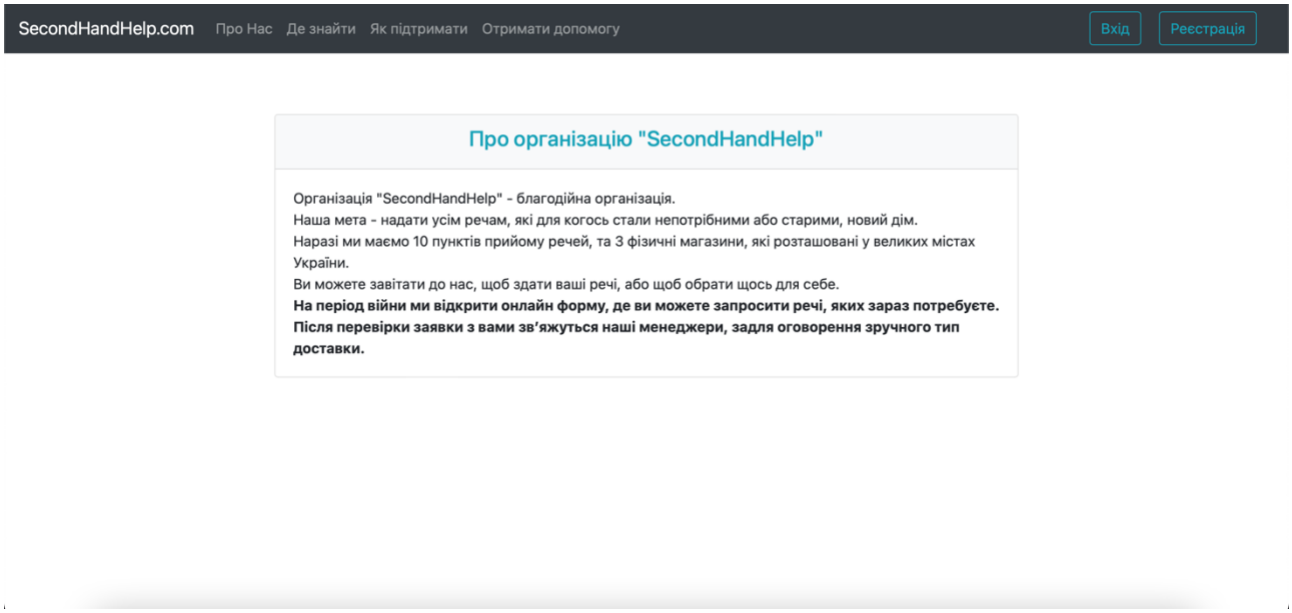

Рис. 8.Сторінка "Про нас"

Меню для незареєстрованого користувача складається з посилань на три статичні сторінки («Про нас», «Де знайти», «Як підтримати») та з посилання на динамічну сторінку - «Отримати допомогу». А також містить дві кнопки – «Вхід» та «Реєстрація».

«**Де нас знайти** » - статична сторінка, містить адреси та години роботи прийомних пунктів та магазинів організації (рис. 9) .

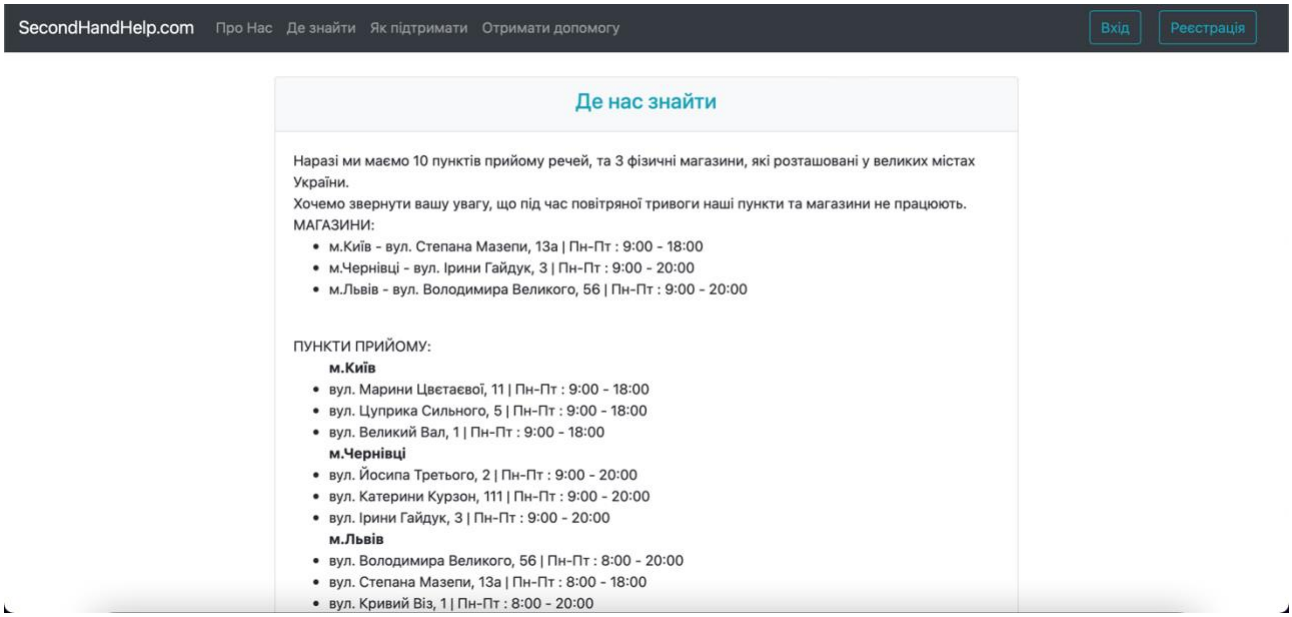

Рис. 9. Сторінка "Де нас знайти"

«Як підтримати? » - статична сторінка, містить реквізити для грошової підтримки організації (рис.10).

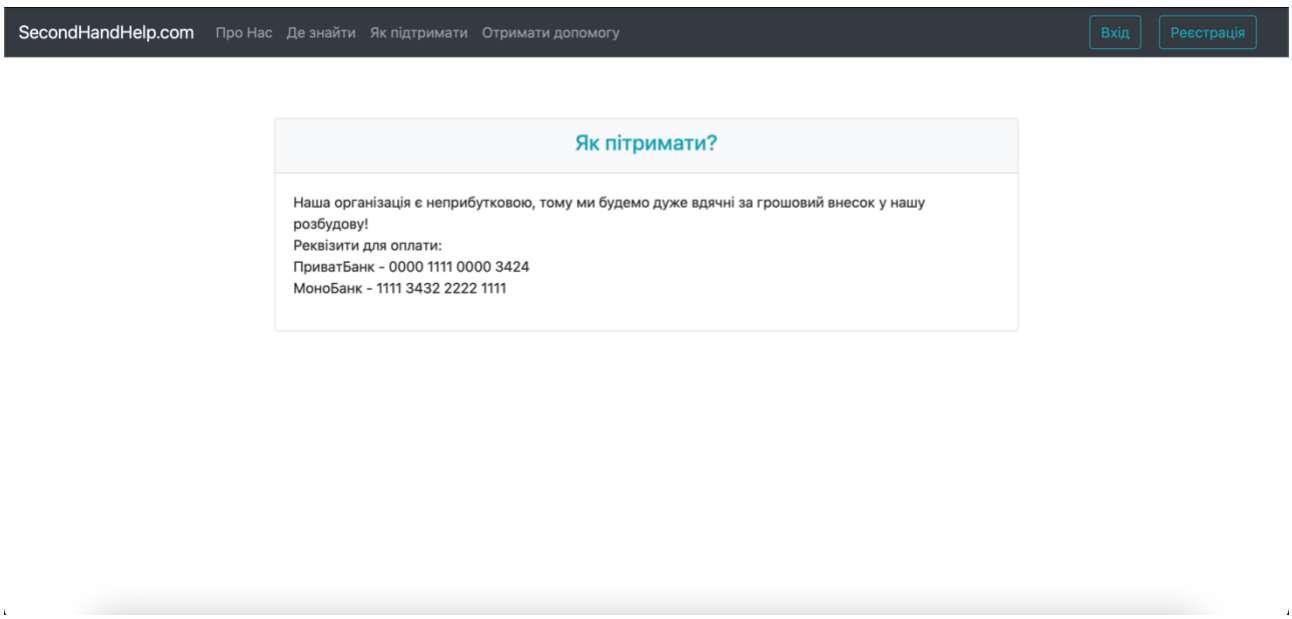

Рис. 10. Сторінка "Як підтримати"

«Отримати допомогу» - сторінка, що містить детальну інструкцію як оформили заявку на допомогу вперше та що робити, якщо користувач вже раніше оформлював у організації заявку (рис.11).

Алгоритм оформлення заявки вперше складається з реєстрації користувача у системі. Для цього потрібно ввести логін (це може бути як електрона пошта так і номер телефону) і надійний пароль. Для зручності користувачів, форму реєстрації було продубльовано у низу сторінки (рис.12).

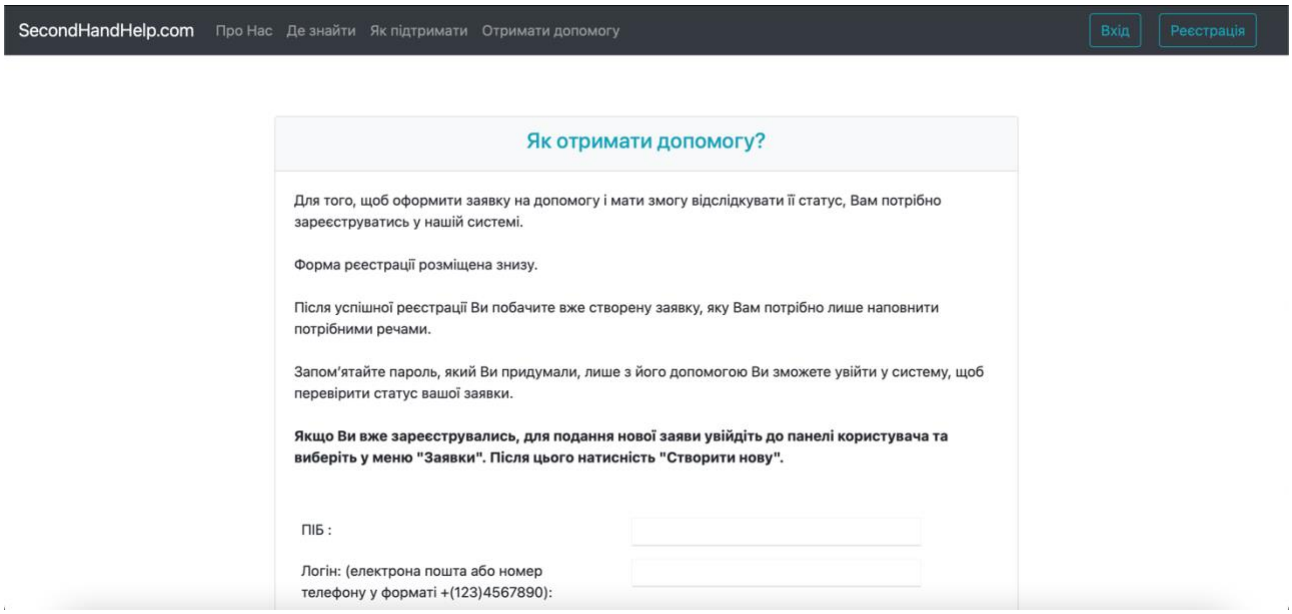

Рис. 11. Сторінка "Отримати допомогу"

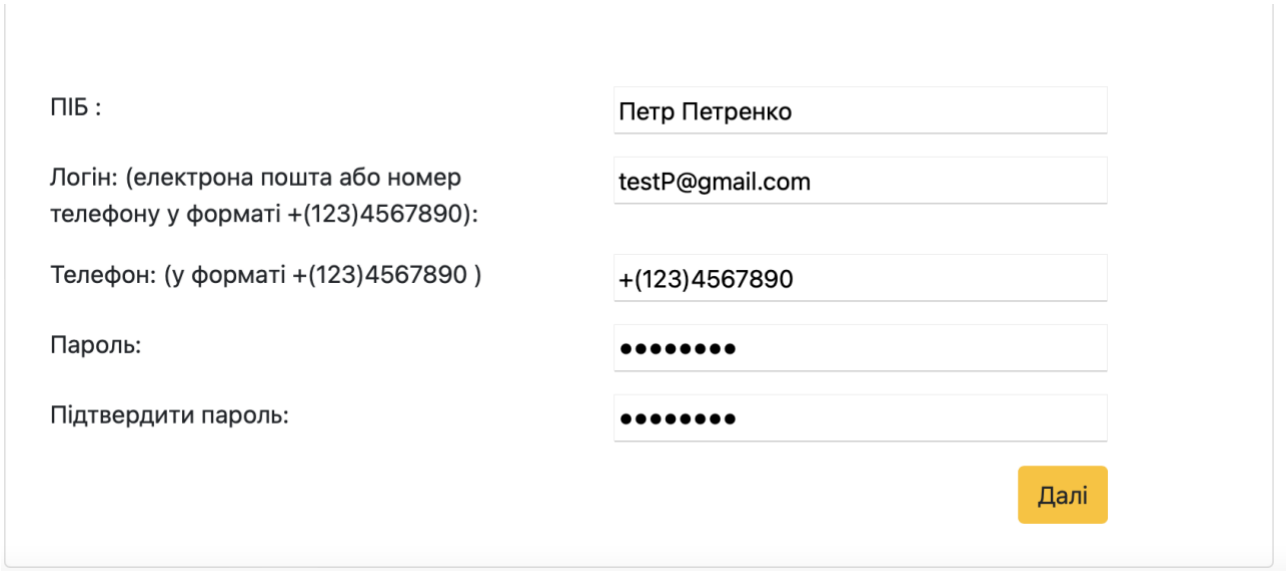

Рис. 12. Форма реєстрації нового користувача

Якщо введені дані будуть містити помилки, користувач побачить вікно попередження (рис.13).

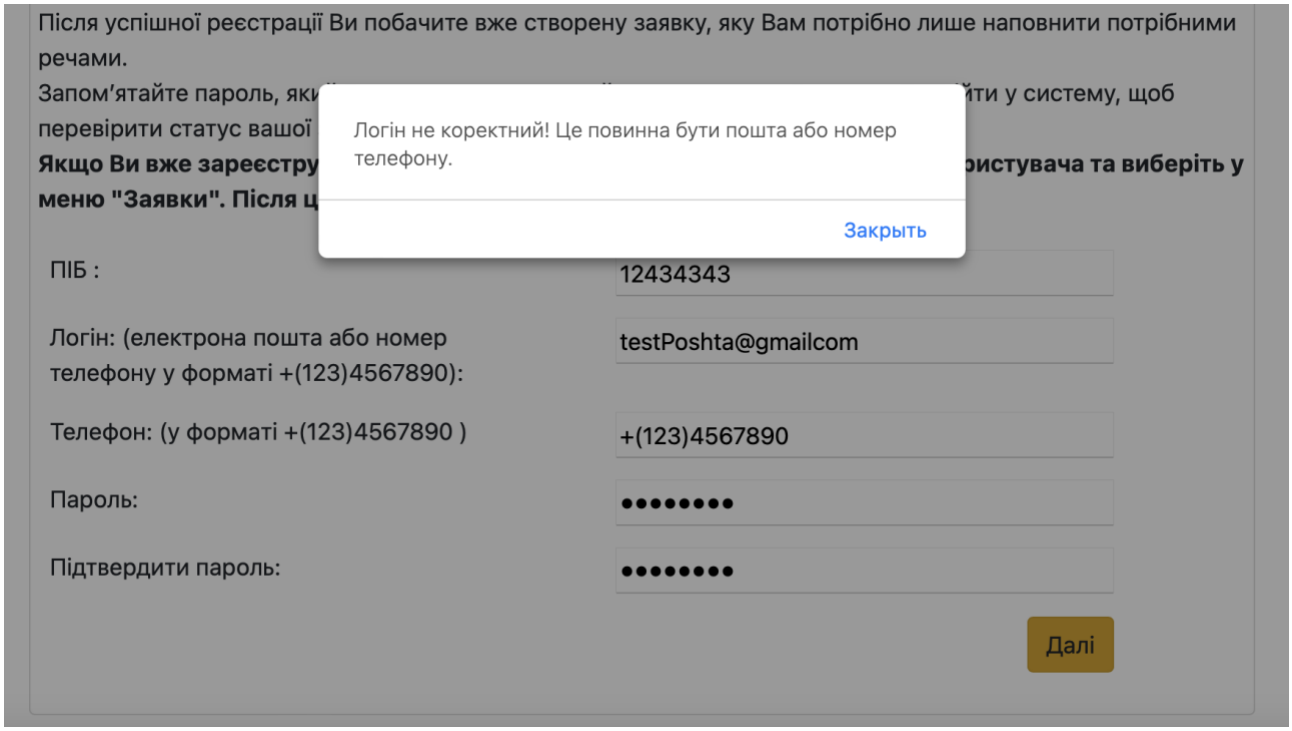

Рис. 13. Вікно попередження при некоректних даних

Якщо користувач з таким логіном вже зареєстрований у системі, користувач побачить сторінку помилки (рис.14).

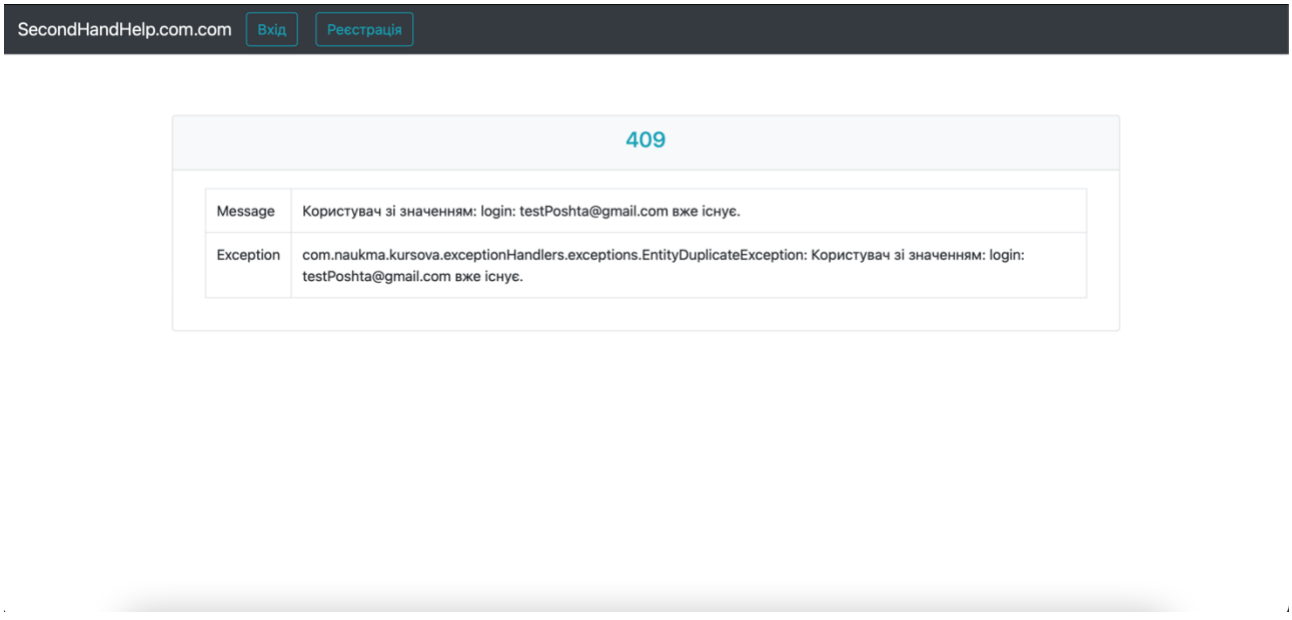

Рис. 14. Вікно помилки "Такий користувач вже існує"

Після успішної реєстрації, заявка створиться автоматично, а користувача буде переадресовано одразу до сторінки її перегляду (рис.15).

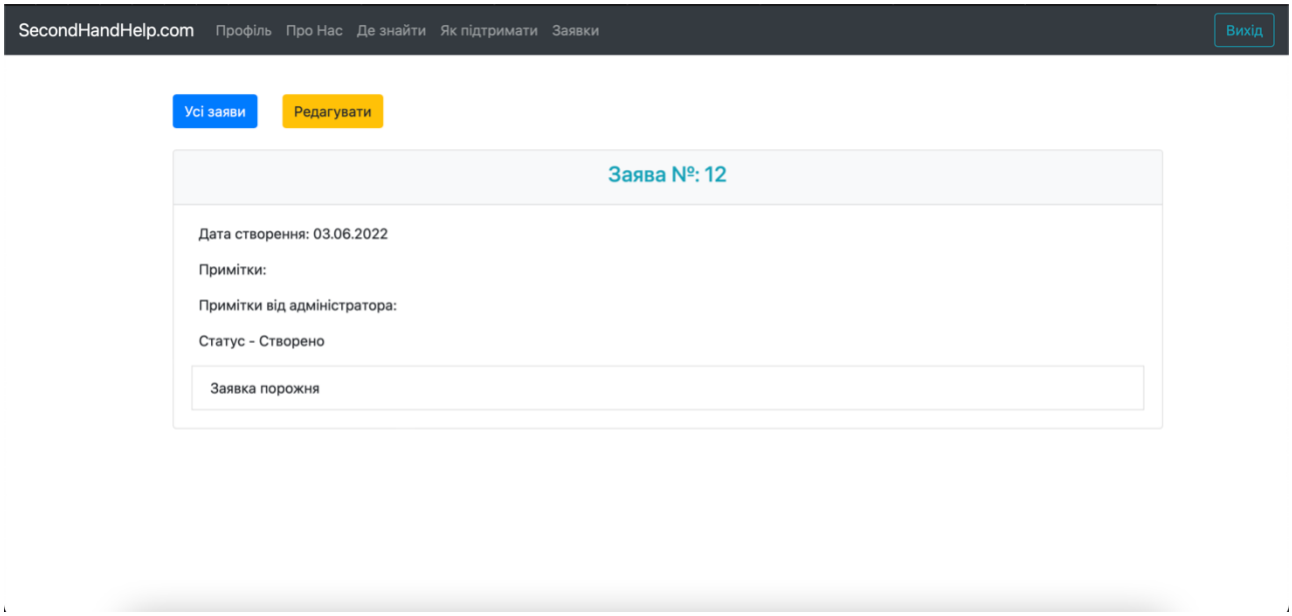

Рис. 15. Перегляд нової заявки

За кліком на кнопку «Реєстрація» можливо зареєструватись у системі для подання заявок та подальшого відслідковування їх статусів. (рис.16).

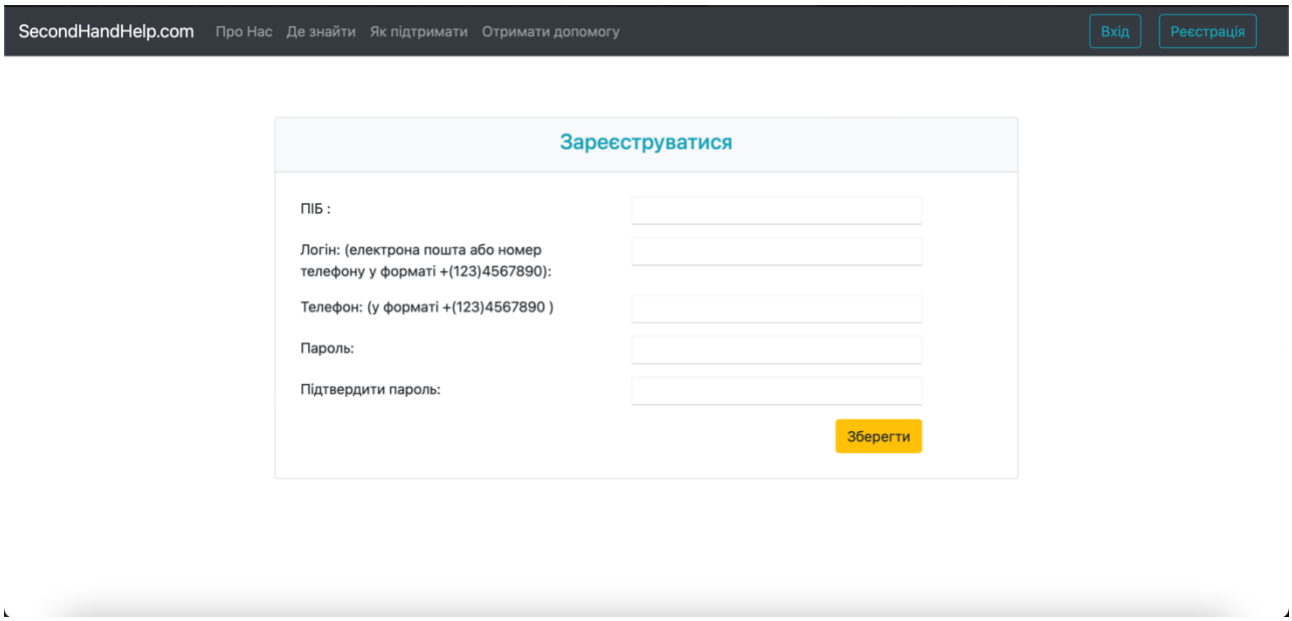

Рис. 16. Сторінка "Реєстрація"

За кліком на кнопку «Вхід» можливо здійснити вхід до панелі користувача або адміністратора (рис. 17).

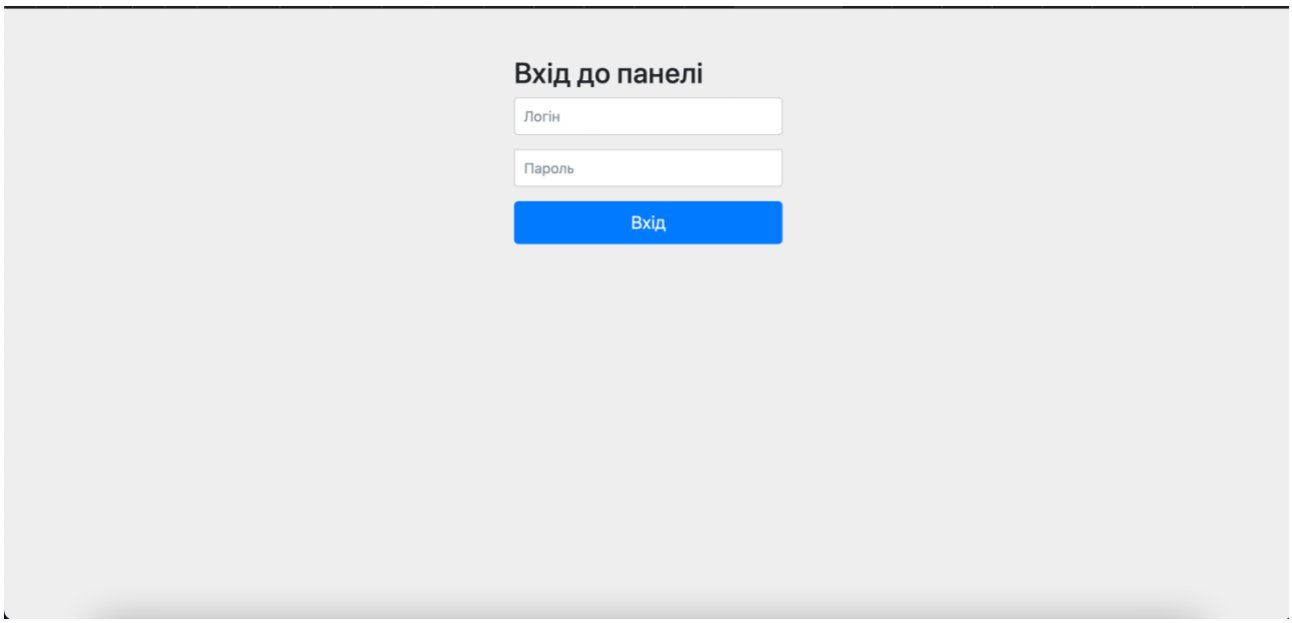

Рис. 17. Сторінка "Вхід"

Після успішного авторизації, відповідно до ролі (звичайний користувач чи адміністратор) користувач потрапить у потрібну панель.

«Панель адміністратора»

Головною сторінкою панелі адміністратора є сторінка «Профіль» (рис. 18), яка містить основну інформацію та кнопку переходу до сторінки її редагування.

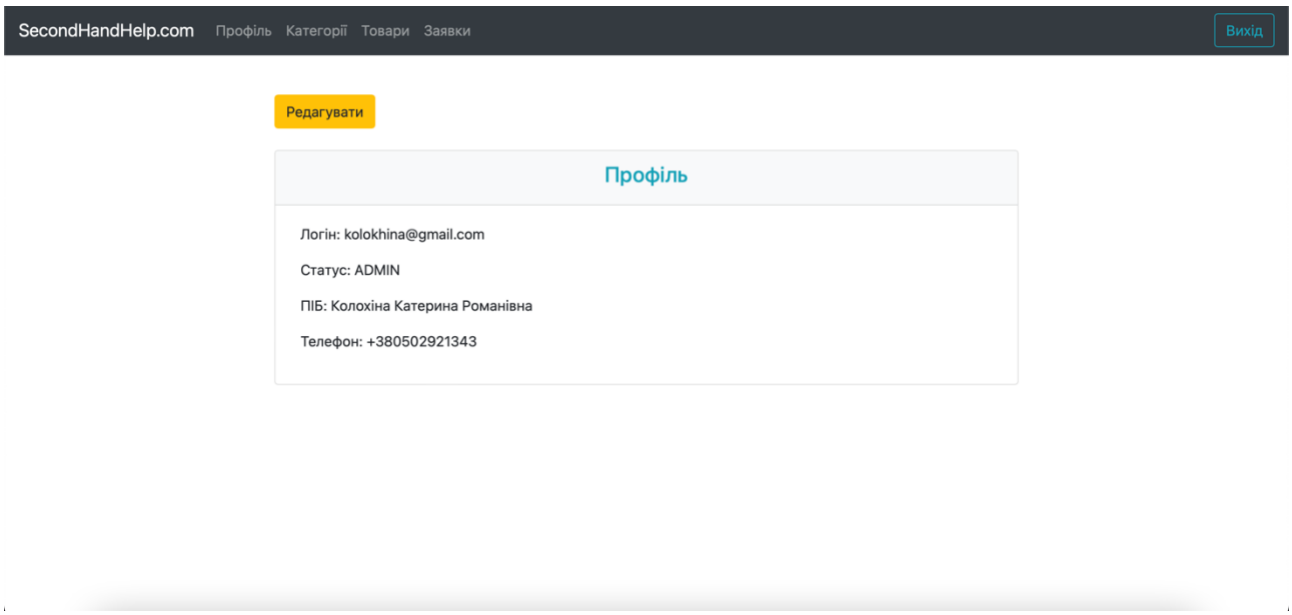

Рис. 18. Сторінка "Профіль"

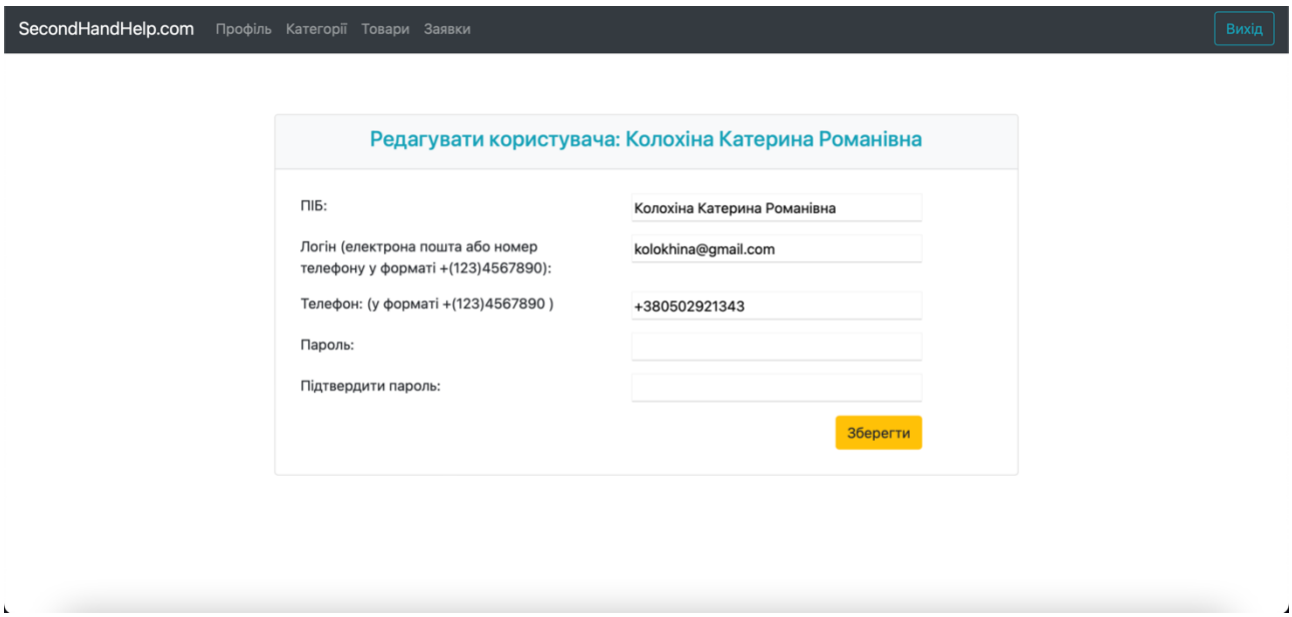

Рис. 19. Сторінка "Редагування профілю"

Меню панелі адміністратора (рис. 20) складається з трьох посилань на відповідні розділи та кнопки виходу.

Рис. 20. Меню панелі адміністратора

За кліком на «Категорії» у меню, користувач потрапить на сторінку усіх категорій (рис. 21).

На сторінці окрім інформації про категорії також наявні: кнопка для видалення певної категорії, кнопка для переходу на сторінку редагування категорії та кнопка переходу на сторінку створення нової категорії.

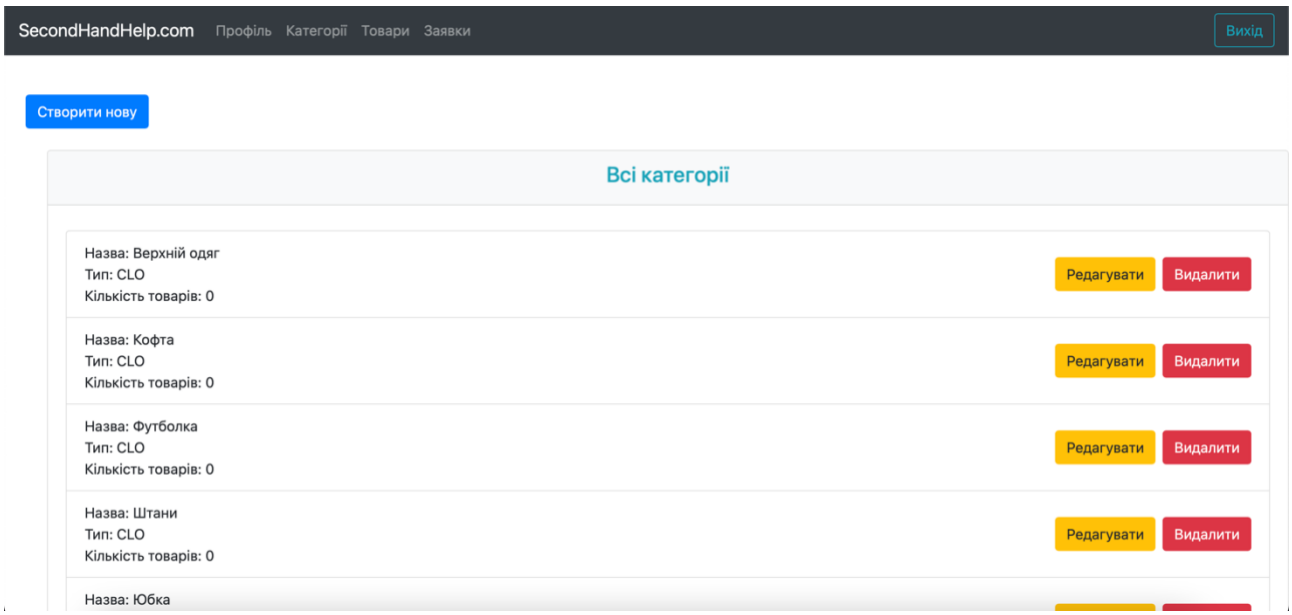

Рис. 21. Сторінка "Всі категорії"

Під час створенні категорії (рис. 22), окрім назви потрібно також обрати тип речі, до якого вона відноситься (одяг чи взуття) і чи потребує річ вказання сезону (верхній одяг, наприклад, може бути або зимовим, або осіннім).

Сторінка редагування категорії (рис. 23) аналогічна до сторінки створення, однак, при першому завантаженні сторінки поля форми заповнюються даними обраної для редагування категорії.

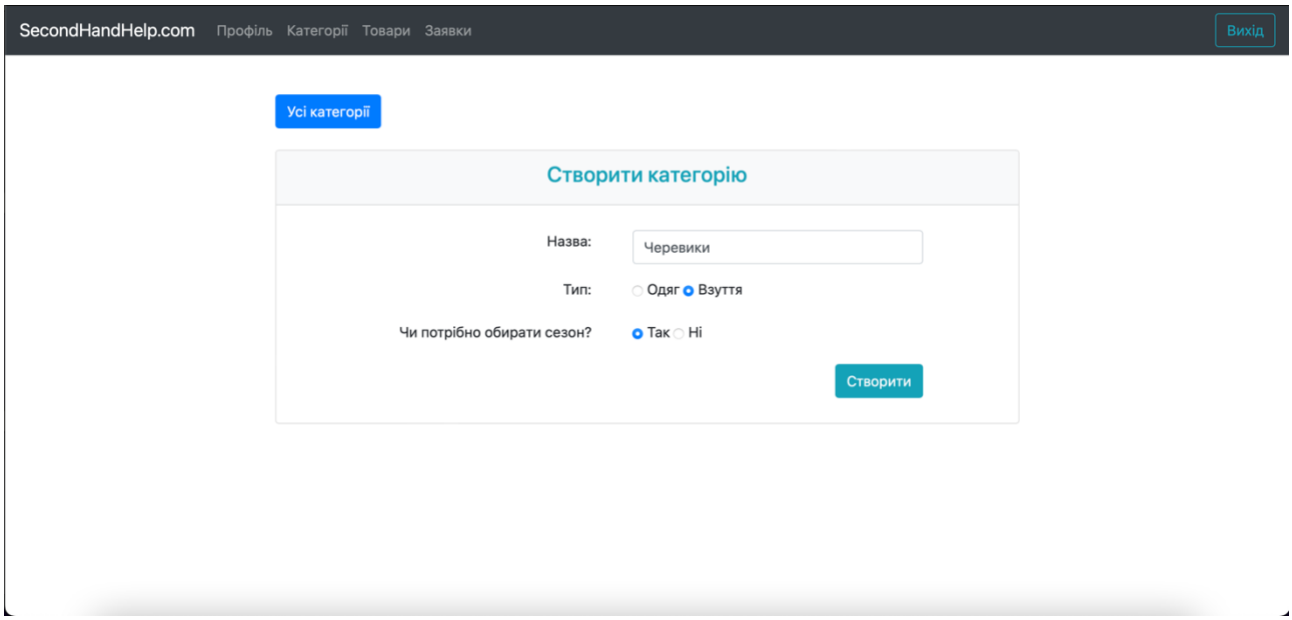

Рис. 22. Сторінка "Нова категорія"

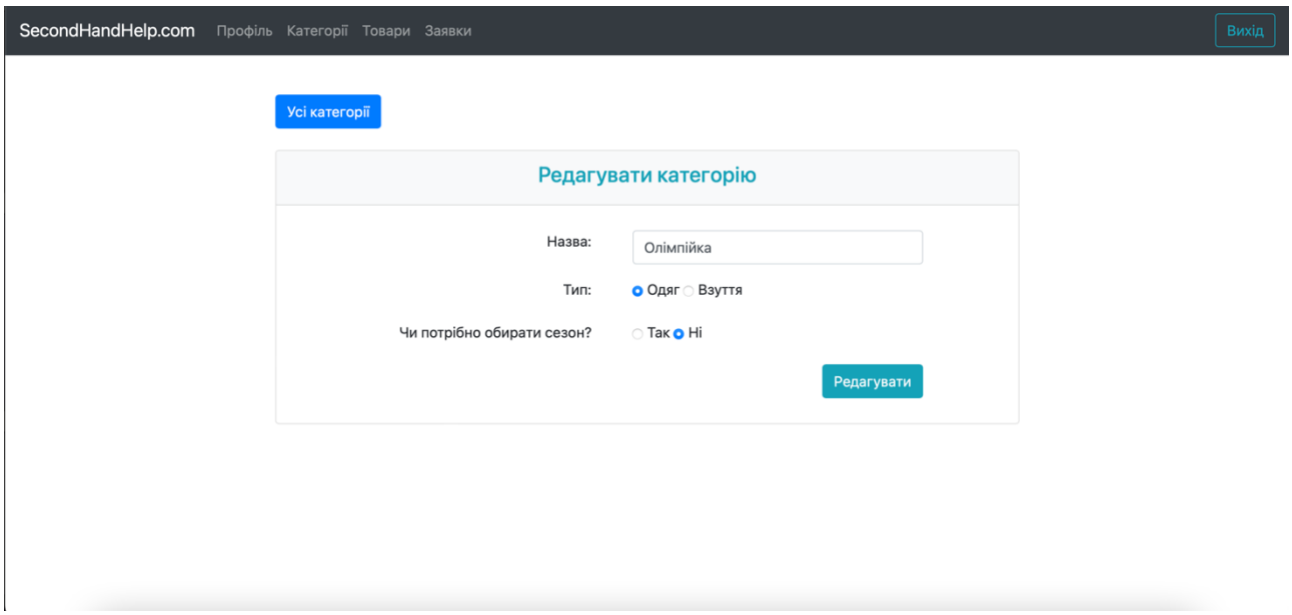

Рис. 23. Сторінка "Редагувати категорію"

За кліком на «Товари» у меню, користувач потрапить на сторінку усіх товарів (рис. 24).

На сторінці окрім інформації про товари також наявні: кнопка для видалення певного товару , кнопка для переходу на сторінку редагування товару та кнопка переходу на сторінку створення нового товару.

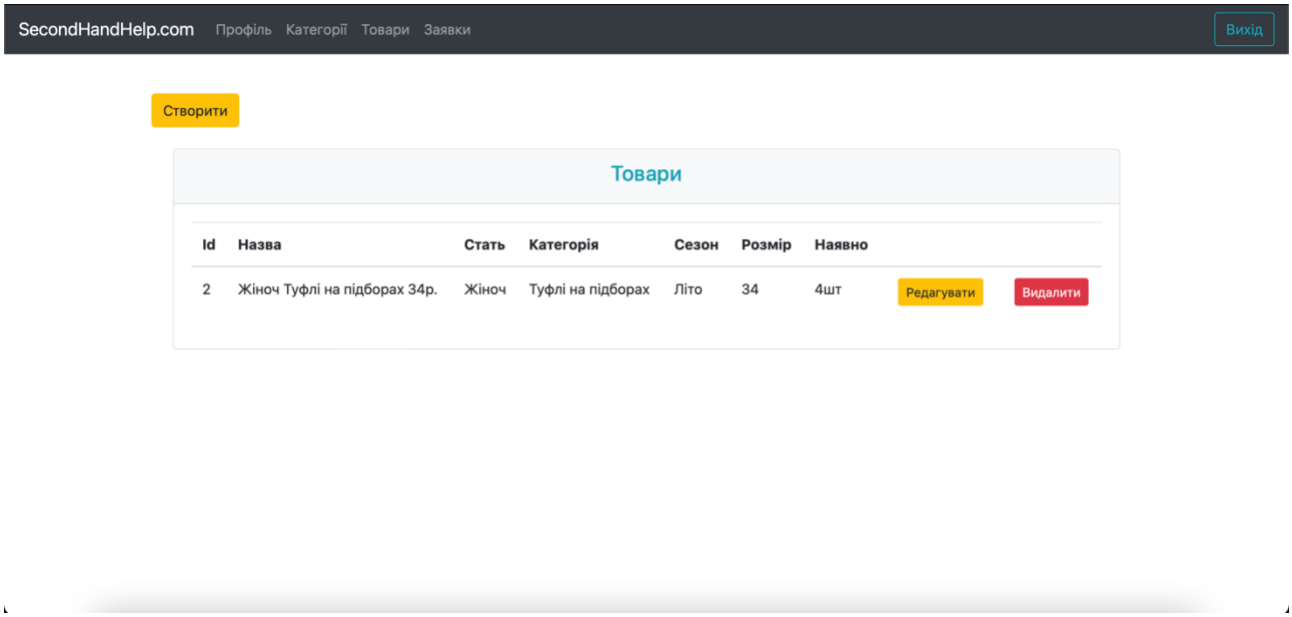

Рис. 24. Сторінка "Усі товари"

Сторінка створення товару (рис. 25) містить два «розумних» поля – категорія та розмір. Категорію можна обрати лише після вибору типу товару, та лише таку, яка до цього типу відноситься (рис. 26). Аналогічно і з розміром – його можна обрати лише після вибору статі та лише у межах вказаних для цієї розмірів (рис. 27).

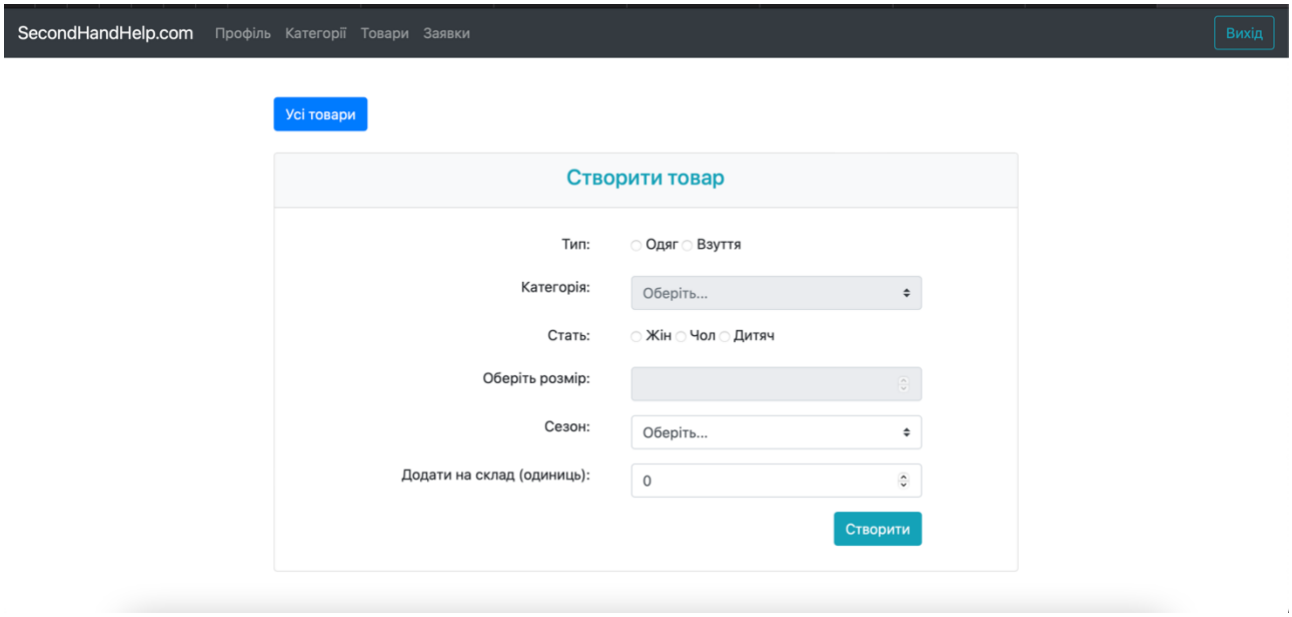

Рис. 25. Сторінка "Створити товар"

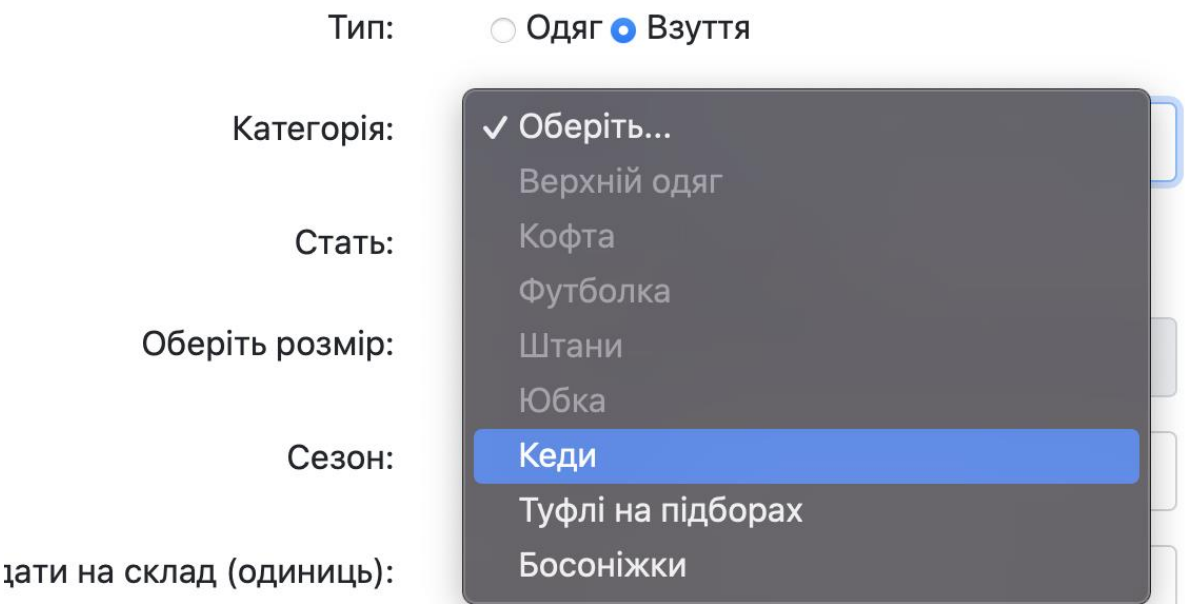

Рис. 26. Розумне поле вибору категорії товару

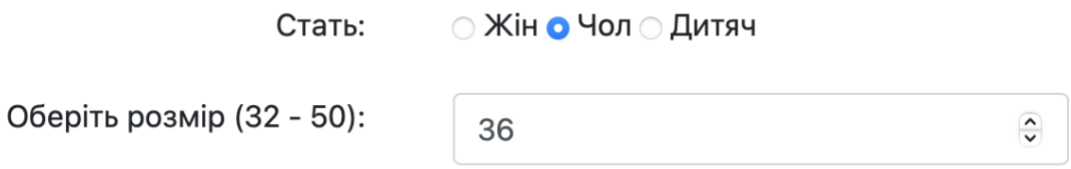

Рис. 27. Розумне поле вибору розміру товару

Якщо користувач створить товар, який вже існує, він побачить вікно помилки (рис. 28).

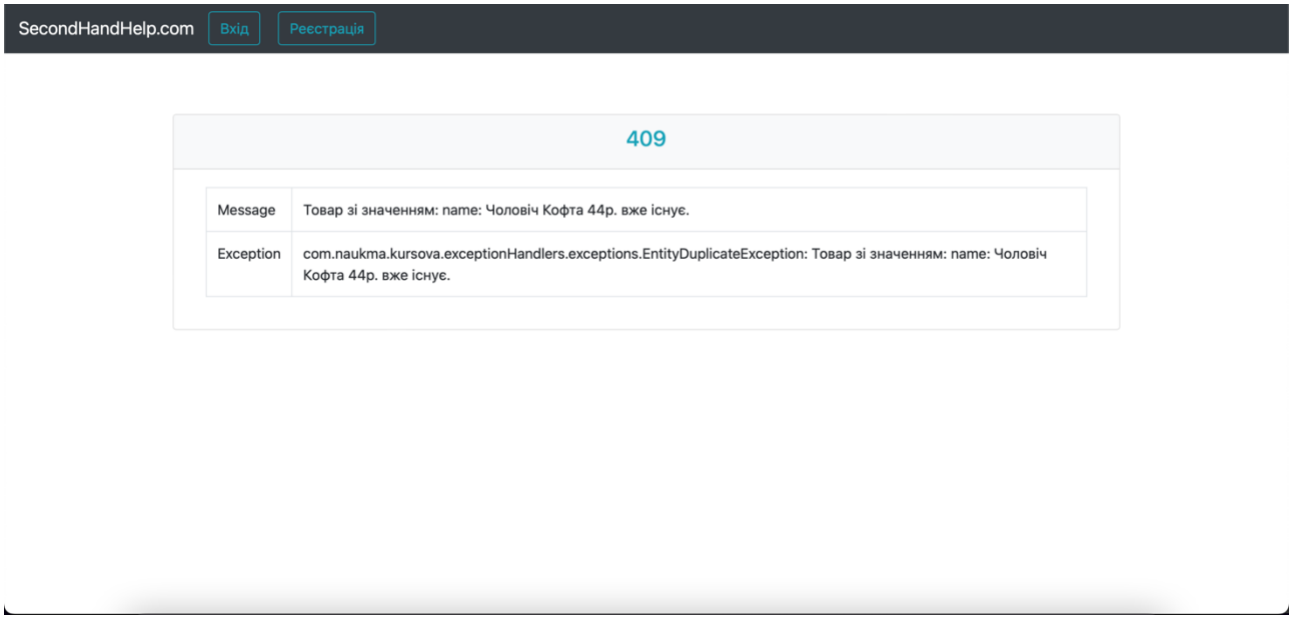

Рис. 28. Помилка "Такий товар вже існує"

Сторінка редагування товару (рис. 29) аналогічна до сторінки «Створення товару», але ще містить функцію списання та зарахування товару на склад.

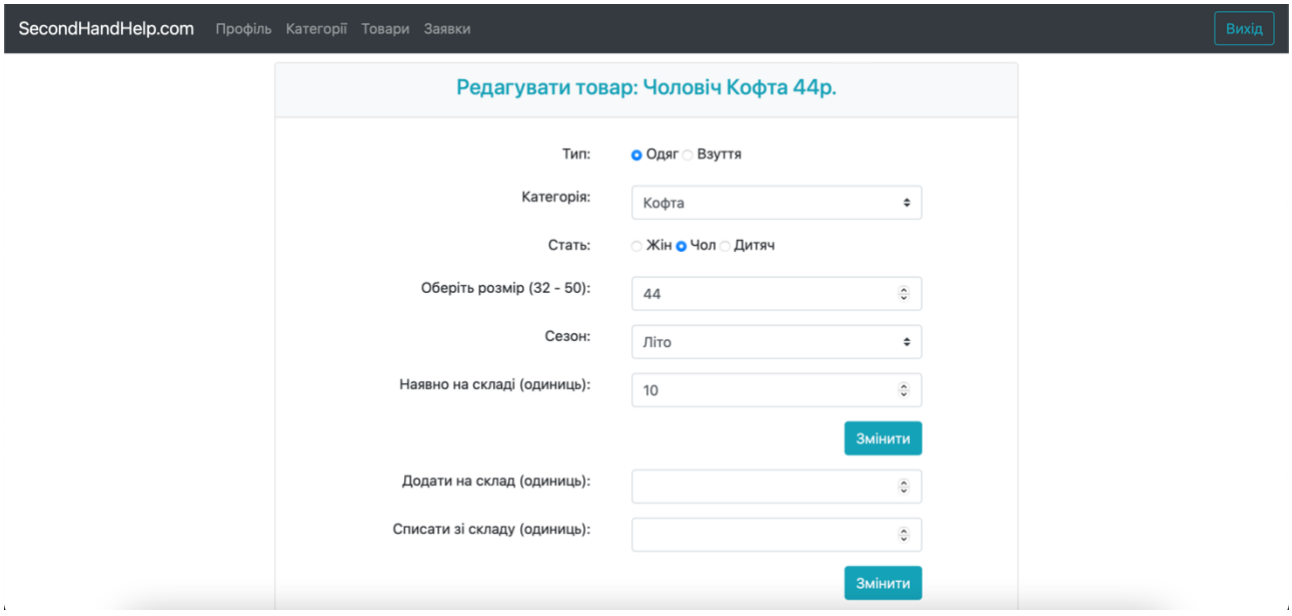

Рис. 29. Сторінка "Редагувати товар"

За кліком на «Заявки» у меню, користувач потрапить на сторінку усіх заявок на допомогу (рис. 30).

На сторінці окрім інформації про заявки також наявні: кнопка для видалення певної заявки (якщо заявка вже завершена, кнопка не буде активною), кнопка для переходу на сторінку перегляду заявки, кнопка для переходу на сторінку редагування заяви та кнопка переходу на сторінку створення нової заяви певному користувачу.

Також, є можливість сортувати заявки на сторінці за номером, ПІБом замовника, датою створення та статусом.

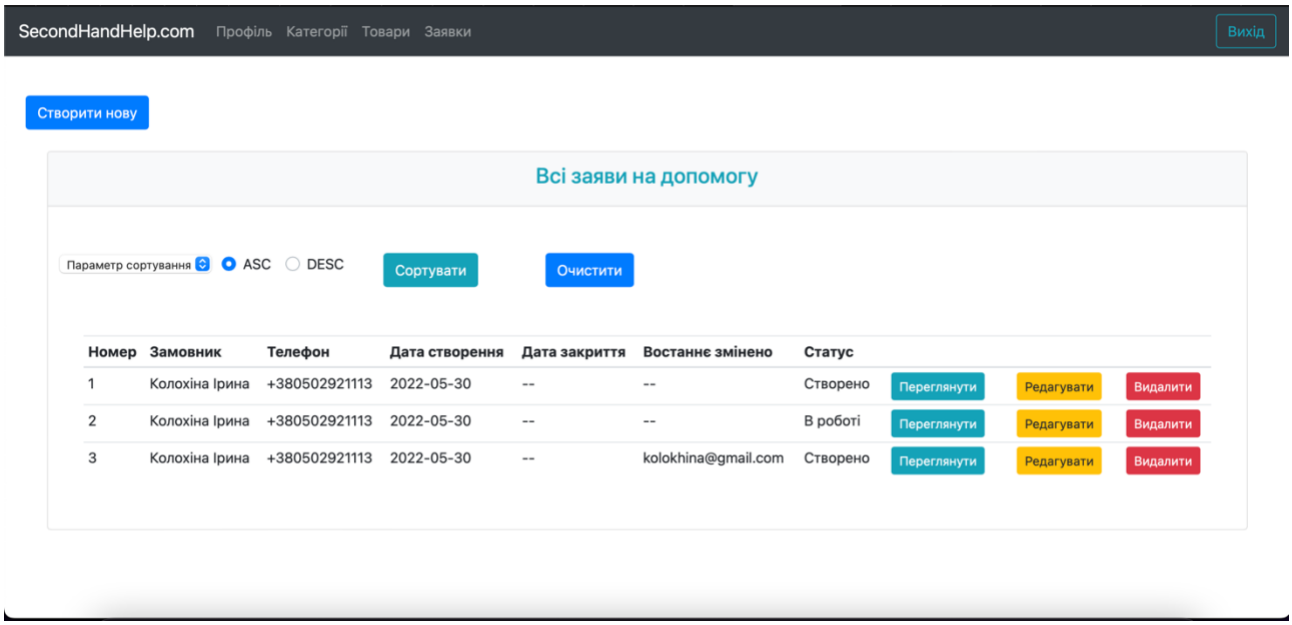

Рис. 30. Сторінка "Всі заяви на допомогу"

За кліком на кнопку «Переглянути», користувача переадресує на відповідну сторінку з усією важливою інформацією про заявку включаючи список товарів, які входять до її складу (рис. 31). Окрім перегляду інформації є можливість видалити певні позиції з заявки (завдяки кнопкам «Видалити» навпроти кожної позиції) або перейти на сторінку редагування заявки (рис.32).

Також, адміністратор може змінити статус заявки за допомогою відповідних кнопок «У роботі» та «Завершити». В залежності від поточного статусу, деякі з них можуть бути не активні.

Варто зазначити, що переведення заявки у статус «Завершено» запускає алгоритм списання зі складу кількості вказаних у списку товарів, блокує заявку для її редагування та видалення і автоматично виставляє «Дату завершення».

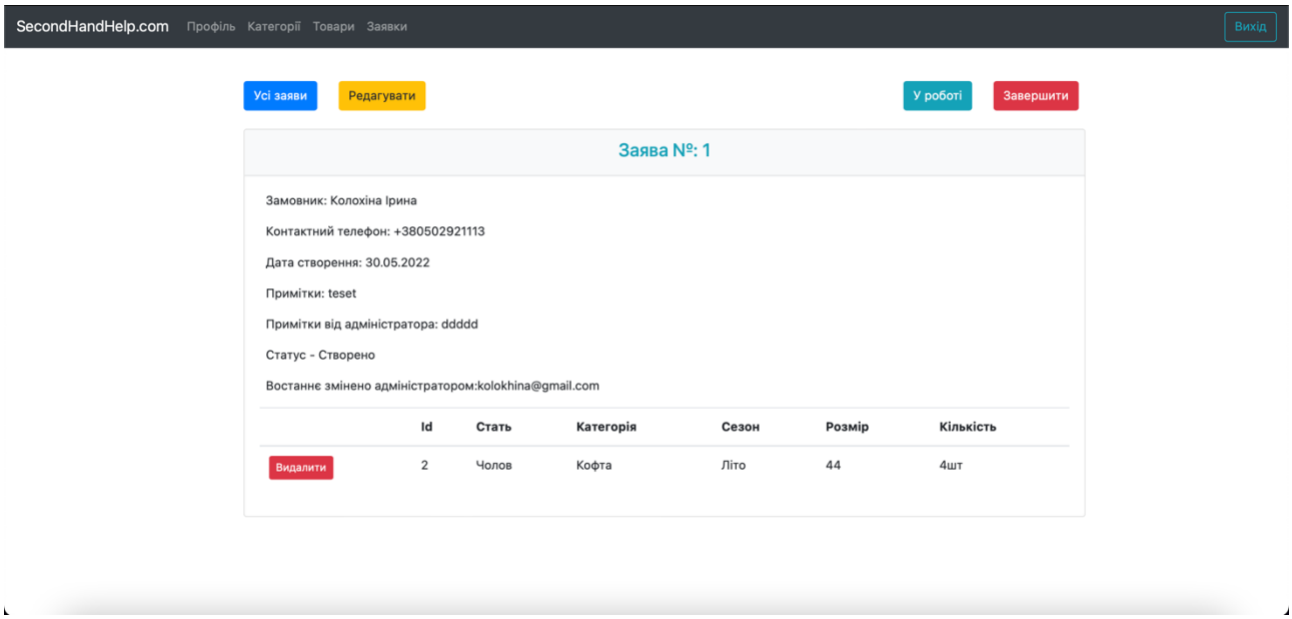

Рис. 31. Сторінка "Перегляд заявки»

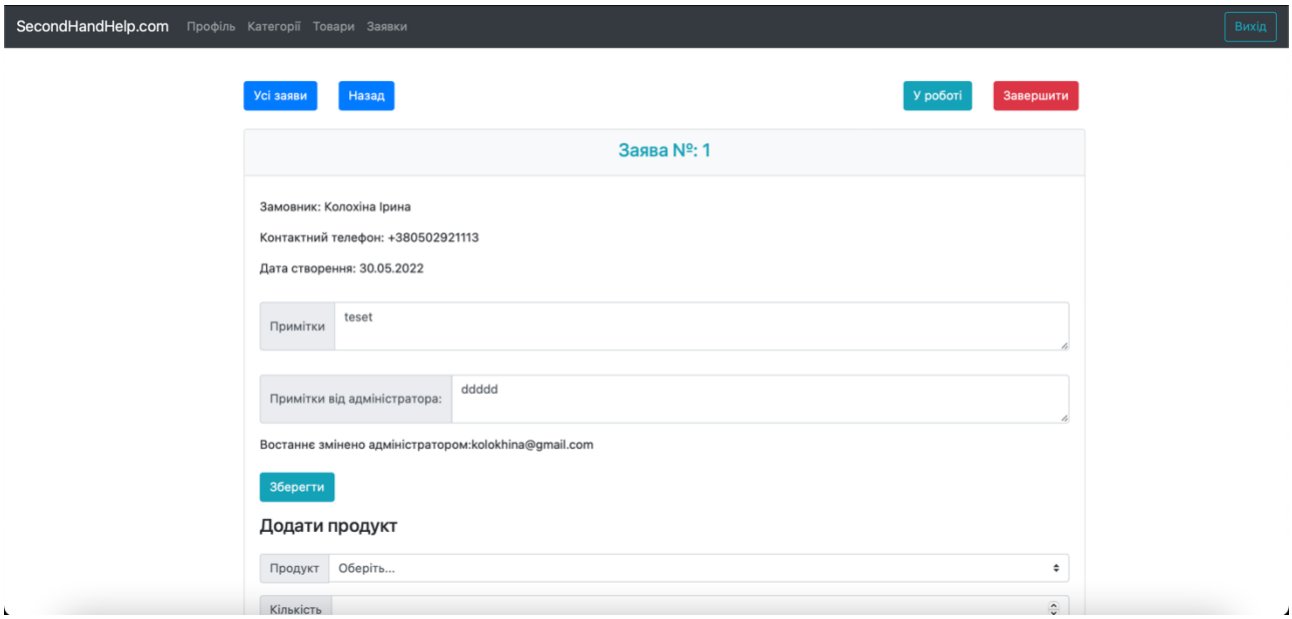

Рис. 32. Сторінка "Редагування заявки"

На сторінці редагування заявки (рис.32, рис.33) адміністратор може редагувати примітки, наповнення заявки та кількість обраних товарів і змінювати статус заявки. Для зручності використання сторінку було поділено на три форми: редагування загальної інформації, додавання нового продукту, зміна кількості товарів у заявці.

Варто зазначити що на поле введення кількісті товарів (при додаванні нового, або редагуванні наявного у заявці) обмежене кількістю товару, що знаходиться на складі. Якщо користувач або адміністратор буде намагатись ввести більшу кількість, він отримає попередження у спливаючому вікні (рис. 34).

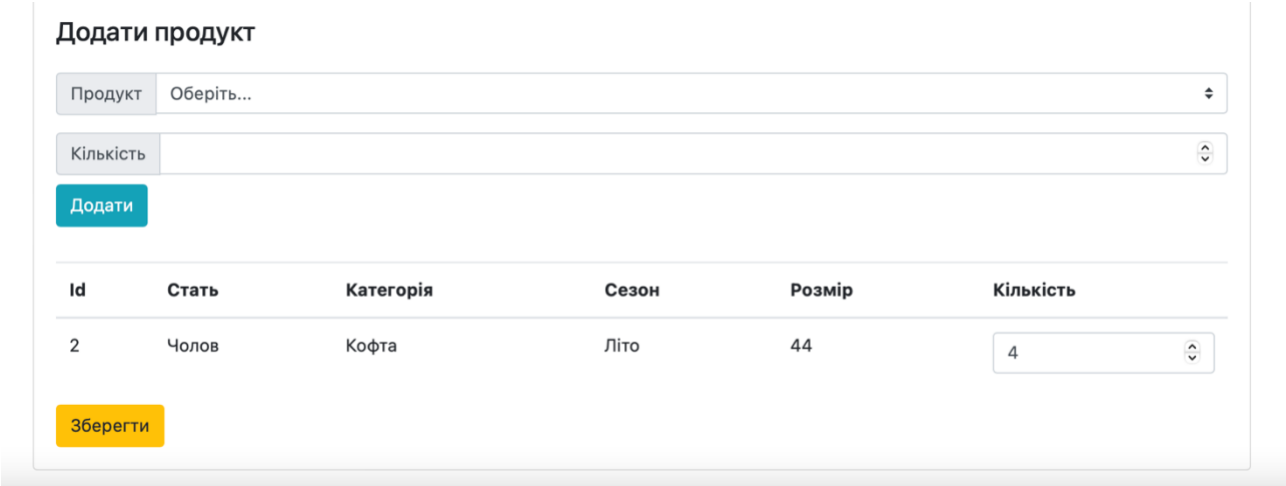

Рис. 33. Частина сторінки "Редагування заявки"

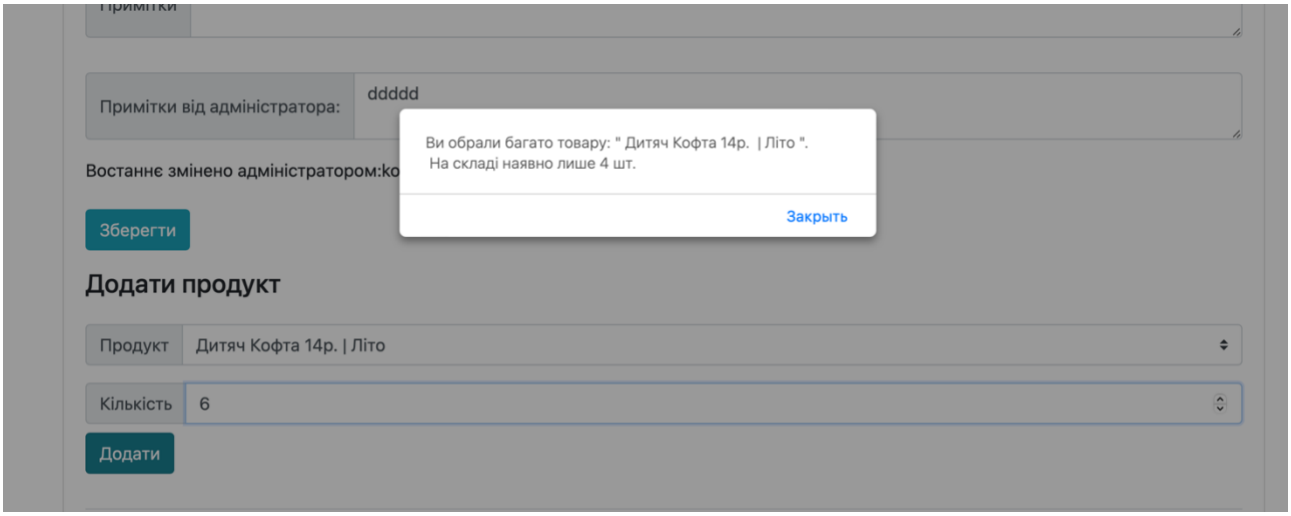

Рис. 34. Попередження "Обрано забагато одиниць товару."

«Панель користувача»

Як і у панелі Адміністратора, стартовою сторінкою є профіль користувача (рис.35). Користувач, як і адміністратор, може вносити зміни до свого профілю (рис.36).

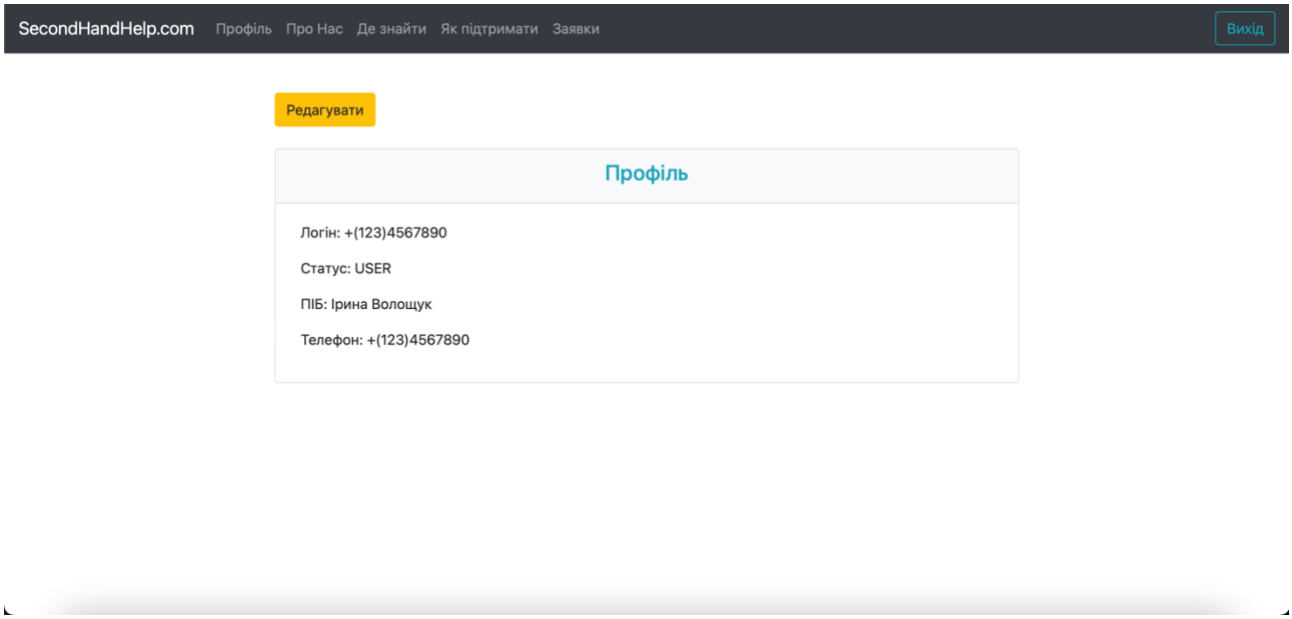

Рис. 35. Сторінка "Профіль" користувача

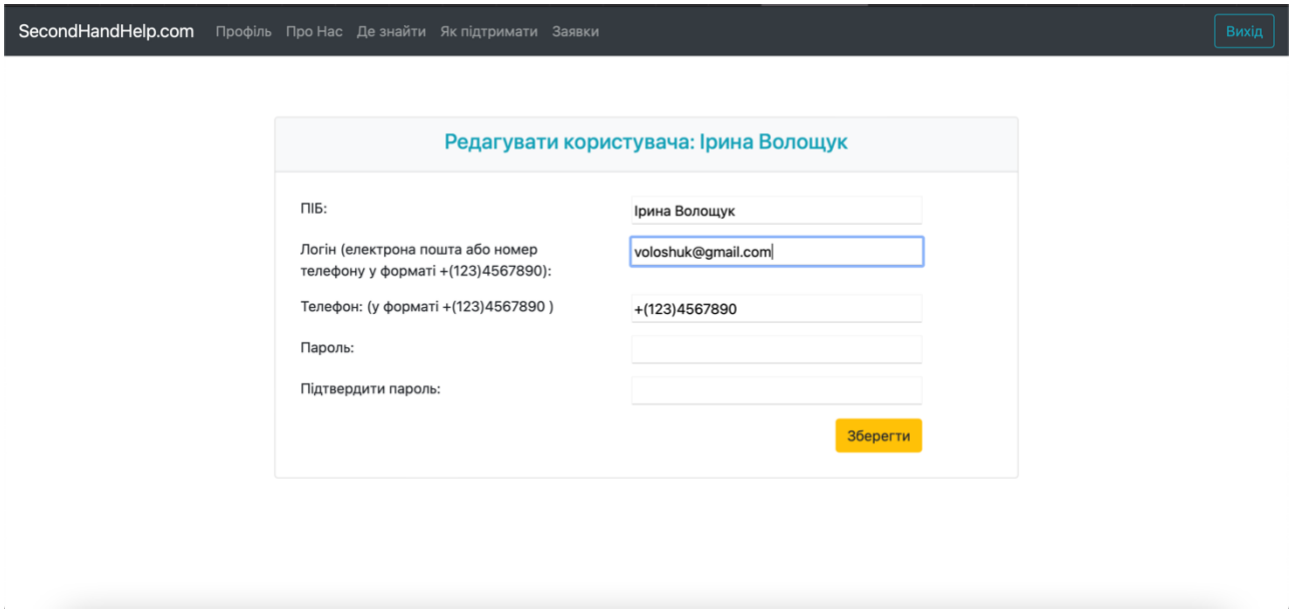

Рис. 36. Сторінка "Редагування профілю" користувача

Меню панелі користувача (рис. 37) складається з трьох посилань на відповідні розділи та кнопки виходу.

Рис. 37. Меню панелі Користувача

На відміну від адміністратора, користувач має обмежений функціонал. Користувач може лише переглядати свій профіль та вносити до нього зміни, переглядати статичні сторінки з інформацією про організацію, та свої заявки (рис.38).

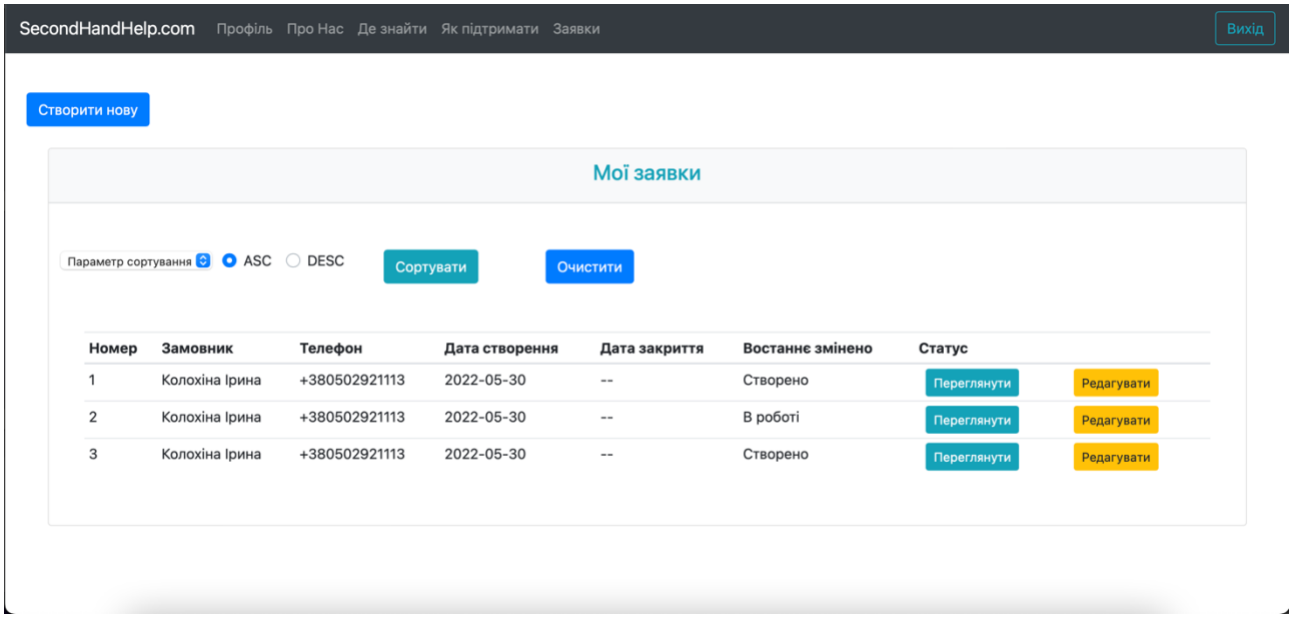

Рис. 38. Сторінка "Мої заявки"

Також, на відміну від адміністратора, користувач не може видаляти заявки, але може створювати нові, переглядати та редагувати їх.

Сторінка перегляду заявки виглядає майже так само як і на панелі Адміністратора (рис.39).

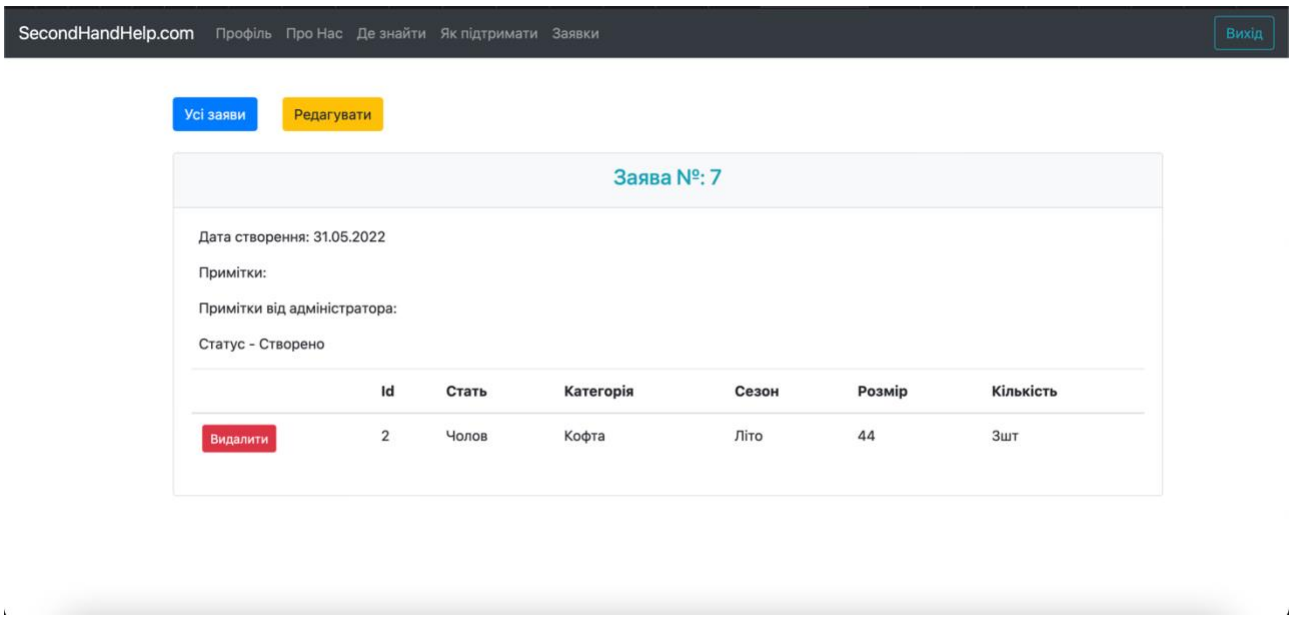

Рис. 39. Сторінка "Перегляд заявки" користувача

З приміток, користувач може редагували лише свої (рис.32).

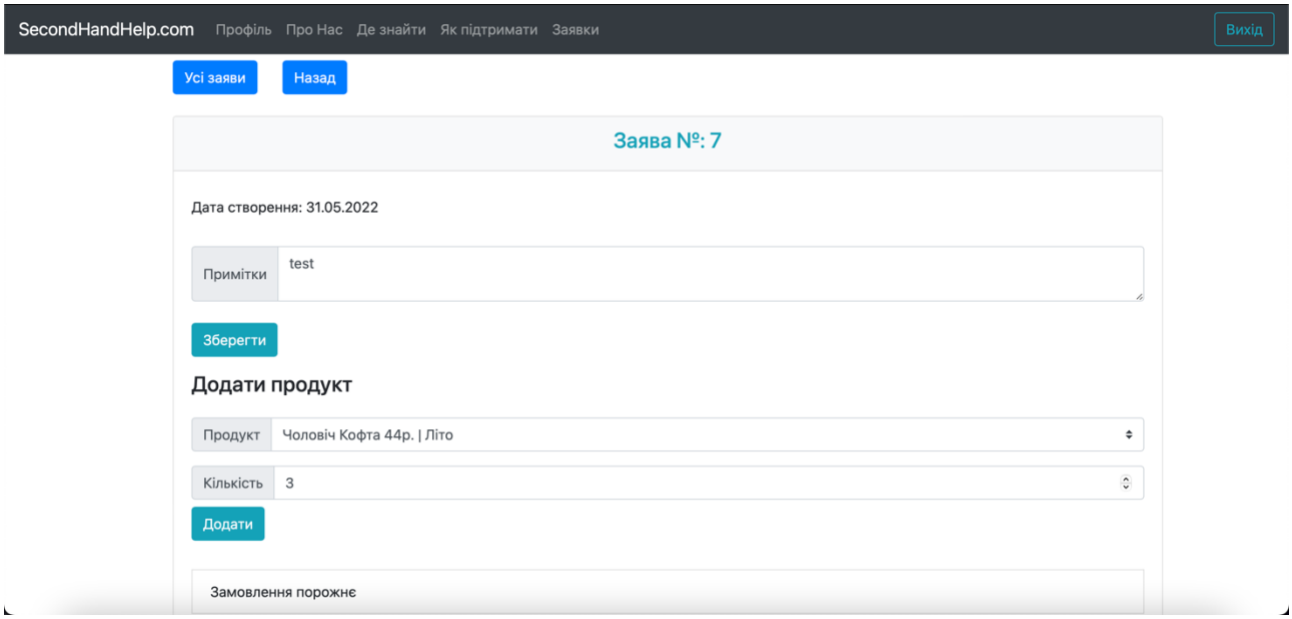

Рис. 40. сторінка "Редагування заявки" користувача

Варто також зазначити, що з метою забезпечення безпеки та конфіденційності адміністраторів, користувач не бачить хто востаннє з адміністраторів вносив зміни до його заявки, чи змінював її статус.

Висновки

Отже, поставлена задача була виконана: розроблено веб-застосунок для благодійної організації з додатковою системою обліку отриманих речей та заявок про допомогу.

Для цього було розглянуто та проаналізовано популярні на сьогодні системи для управління складом та виокремлено найбільш пріоритетні вимоги до подібних спеціалізованих систем для благодійних магазинів.

На основі поставленого технічного завдання було підібрано набір сучасних технологій для реалізації застосунку. В якості прикладу було обрано благодійний магазин (наразі волонтерський центр), що спеціалізується на зборі одягу та взуття.

У результаті створено зручний та простий застосунок, який допоможе працівникам контролювати обсяг речей, які їм надходять, та забезпечить єдину базу та алгоритм опрацювання заявок на допомогу.

Завдяки використанню сучасного фреймворку та нормалізованої бази даних, систему можна розвивати й надалі. Наприклад, можливе розширення асортименту товарів, інформація про які зберігатиметься у системі. Також, існує низка операцій зі складом та корисних функцій, які можливо реалізувати у системі, задля збільшення швидкості роботи (поповнення складу шляхом імпорту списку речей з готового файлу або додавання одразу декількох товарів до заявки).

Список використаної літератури

1. Шокуюча статистика про кількість речей, яку ми маємо.[Електронний ресурс] https://www.becomingminimalist.com/clutter-stats/
2. Поняття про раціональне споживання. [Електронний ресурс] https://uahistory.co/pidruchniki/bakka-civil-education-10-class-2018/42.php
3. «1С:УНФ». Плюси та мінуси. [Електронний ресурс] https://gendalf.ru/news/store/1c-unf-plusy/
4. «BAS Управління торгівлею». Загальна інформація. [Електронний ресурс] https://www.bas-soft.eu/soft/bas-mass/bas-trade-management/#recentPosts
5. MVC визначення та опис технології. [Електронний ресурс] https://uk.wikipedia.org/wiki/Модель-вид-контролер
6. Spring MVC. Основні принципи [Електронний ресурс] https://habr.com/ru/post/336816/
7. Richards M. Software Architecture Patterns. O'Reilly Media, Inc., 2015. 54 p. [Електронний ресурс] https://www.safaribooksonline.com/library/view/software-architecture-patterns/9781491971437/

Додатки

Додаток А. Файл з залежностями проекту.

```
?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.6.8-SNAPSHOT</version>
    <relativePath/>
  </parent>
  <groupId>com.naukma</groupId>
  <artifactId>kursova-kolokhina</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>kursova</name>
  <description>Demo project for kursova robota of Kateryna Kolokhina</description>
  <properties>
    <java.version>1.8</java.version>
  </properties>
  <dependencies>

    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-jdbc</artifactId>
    </dependency>

    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
      <groupId>mysql</groupId>
      <artifactId>mysql-connector-java</artifactId>
      <scope>runtime</scope>
    </dependency>

    <dependency>
      <groupId>org.projectlombok</groupId>
      <artifactId>lombok</artifactId>
      <optional>true</optional>
    </dependency>

    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-thymeleaf</artifactId>
    </dependency>

    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
```

```
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
  </dependency>

  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-validation</artifactId>
  </dependency>

  <dependency>
    <groupId>org.thymeleaf.extras</groupId>
    <artifactId>thymeleaf-extras-springsecurity5</artifactId>
  </dependency>

  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
  </dependency>

</dependencies>

<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <configuration>
        <source>11</source>
        <target>11</target>
      </configuration>
    </plugin>
  </plugins>
</build>
<repositories>
  <repository>
    <id>spring-milestones</id>
    <name>Spring Milestones</name>
    <url>https://repo.spring.io/milestone</url>
    <snapshots>
      <enabled>false</enabled>
    </snapshots>
  </repository>
  <repository>
    <id>spring-snapshots</id>
    <name>Spring Snapshots</name>
    <url>https://repo.spring.io/snapshot</url>
    <releases>
      <enabled>false</enabled>
    </releases>
  </repository>
</repositories>
<pluginRepositories>
  <pluginRepository>
    <id>spring-milestones</id>
    <name>Spring Milestones</name>
    <url>https://repo.spring.io/milestone</url>
    <snapshots>
```

```
        <enabled>false</enabled>
      </snapshots>
    </pluginRepository>
    <pluginRepository>
      <id>spring-snapshots</id>
      <name>Spring Snapshots</name>
      <url>https://repo.spring.io/snapshot</url>
      <releases>
        <enabled>false</enabled>
      </releases>
    </pluginRepository>
  </pluginRepositories>

</project>
```

Додаток Б. Налаштування авторизації, автентифікації та доступів.

```
package com.naukma.kursova;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

import javax.sql.DataSource;

@EnableWebSecurity
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {

    @Autowired
    private DataSource dataSource;

    /**
     * Encoder of password
     */
    @Bean
    public BCryptPasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }

    /**
     * Method builder for security configuration
     */
    @Override
    public void configure(AuthenticationManagerBuilder auth) throws Exception {
        auth.jdbcAuthentication()
                .dataSource(dataSource)
                .usersByUsernameQuery(
                        "select login, password, 'true' from user " +
                                "where login=?")
                .authoritiesByUsernameQuery(
                        "select login, role from user " +
                                "where login=?");
    }

    /**
```

```
    * Method for security configuration
    */
   @Override
   protected void configure(HttpSecurity http) throws Exception {
      http
            .csrf().disable()
            .logout()
            .and()
            .authorizeRequests()
            .antMatchers("/index", "/where-find", "/donate", "/help").permitAll()
            .antMatchers("/category/**", "/product/**").hasAuthority("ADMIN")
            .antMatchers("/user/**", "receipt/**").hasAnyAuthority("USER", "ADMIN")
            .and().exceptionHandling().accessDeniedPage("/accessDenied")
            .and()
            .formLogin()
            .defaultSuccessUrl("/")
            .defaultSuccessUrl("/success_login")
            .permitAll();
   }

}
```

Додаток С. Контроллер AuthController.

```
package com.naukma.kursova.controller;

import com.naukma.kursova.entity.User;
import com.naukma.kursova.entity.UserDto;
import com.naukma.kursova.exceptionHandlers.exceptions.EntityDuplicateException;
import com.naukma.kursova.exceptionHandlers.exceptions.UserNotFoundException;
import com.naukma.kursova.repository.UserRepository;
import com.naukma.kursova.service.UserService;
import lombok.extern.log4j.Log4j2;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.view.RedirectView;

import java.security.Principal;

@Controller
@Log4j2
public class AuthController {

   @Autowired
   private UserService userService;

   @Autowired
   private UserRepository userRepository;

   @GetMapping("/registration")
   public ModelAndView registration() {
      ModelAndView mav = new ModelAndView();
      mav.setViewName("registration");
```

```
        UserDto user = new UserDto();
        mav.addObject("user", user);
        return mav;
    }

    @PostMapping("/registration")
    public RedirectView registUser(@ModelAttribute("user") UserDto user) {
        log.info(user);
        if (userRepository.findUserById(user.getId()).isPresent()) {
            throw new EntityDuplicateException("Користувач", "id", user.getId().toString());
        }
        userService.addUser(user);
        Long id = userRepository.findByLogin(user.getLogin()).get().getId();
        return new RedirectView("/user/" + id);
    }

    @GetMapping("/success_login")
    public RedirectView successLogin(Principal principal) {
        User user = userService.findUserByLogin(principal.getName());
        return new RedirectView("/user/" + user.getId());
    }

    @GetMapping("/accessDenied")
    public ModelAndView accessDeniedException() {
        ModelAndView mav = new ModelAndView();
        mav.setViewName("access_denied");
        return mav;
    }

    @ModelAttribute("userId")
    public String getCurrentUserIpn(Authentication authentication) {
        UserDetails det = (authentication == null) ? null : (UserDetails) authentication.getPrincipal();
        if (det == null) {
            return "";
        } else {
            String login = det.getUsername();
            if (!userRepository.findByLogin(login).isPresent()) {
                throw new UserNotFoundException(login);
            } else
                return userRepository.findByLogin(login).get().getId().toString();
        }
    }

    @ModelAttribute("currentUser")
    public UserDetails getCurrentUser(Authentication authentication) {
        return (authentication == null) ? null : (UserDetails) authentication.getPrincipal();
    }
}
```