Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

Студента *Куракіна Олексія Станіславовича*
(ПІБ)

академічної групи *122-21ск-1*
(шифр)

Спеціальності *122 Комп'ютерні науки*
(код і назва спеціальності)

освітньої програми *Комп'ютерні науки*
(назва освітньої програми)

на тему: *Розробка веб-додатку для проведення on-line аукціону продажу товарів з використанням фреймворку Angular та мови програмування PHP*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*доц. Спірінцев В.В.*			
розділів:				
Спеціальний	*доц. Спірінцев В.В.*			
Економічний	*доц. Касьяненко Л.В.*			
Рецензент				
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2024

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)
М.О. Алексєєв
(підпис) (прізвище, ініціали)
« » 2024 Року

ЗАВДАННЯ
на кваліфікаційну роботу
бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *122-21ск-1* *Куракіна Олексія Станіславовича*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка веб-додатку для проведення on-line аукціону продажу товарів, з використанням фреймворку Angular та мови програмування PHP*

затверджена наказом ректора НТУ «ДП» від *23.04.2024* № *375-с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів практик та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі. Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***27.05.2024 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***31.05.2024 р.***

Завдання видав _____ *доц. Спірінцев В.В*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Куракін О.С*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2024 р.*

Термін подання кваліфікаційної роботи до ЕК: *04.06.2024 р*

РЕФЕРАТ

Пояснювальна записка: 119 с., 67 рис., 5 дод., 19 джерел, 8 таблиць.

Об'єкт розробки: веб-додаток для проведення on-line аукціону продажу товарів.

Мета кваліфікаційної роботи: розробка та оптимізація роботи web-додатку для проведення аукціону у on-line форматі та полегшення процесу купівлі-продажу товарів користувачами з різних країн світу.

У вступі розглянуто аналіз та сучасний стан проблеми, визначено мету кваліфікаційної роботи та галузь її застосування, наведено обґрунтування актуальності теми та уточнено постановку завдання.

У першому розділі проаналізовано предметну галузь, визначено актуальність завдання та призначення розробки, сформульовано постановку завдання, наведено вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі проаналізовано існуючі рішення, обрано платформу для розробки, виконано проектування і створення веб-орієнтованої інформаційної системи, описано роботу системи, алгоритм і структуру її функціонування, а також процес виклику та завантаження додатку, визначено вхідні та вихідні дані, охарактеризовано склад і параметри технічних засобів.

В економічному розділі визначено трудомісткість розробленої інформаційної системи, проведено розрахунок вартості створення додатку та обчислено час, необхідний для його розробки.

Практичне значення полягає у створенні веб-додатку, що призначений для полегшення процесу купівлі-продажу товарів користувачами з різних країн світу.

Актуальність розробки веб-додатку для проведення онлайн-аукціонів з продажу товарів не викликає сумніву, оскільки сучасний бізнес все більше переходить до онлайн-формату. Існуючі тенденції в електронній комерції вимагають швидких та ефективних рішень для підтримки та розвитку підприємств. Запровадження такого додатку дозволяє автоматизувати процеси купівлі-продажу, забезпечуючи зручність та доступність для користувачів з різних країн світу.

Список ключових слів: WEB-ДОДАТОК, АУКЦІОН, СТОРІНКА, БАЗА ДАНИХ, ANGULAR, КУПІВЛЯ-ПРОДАЖ, ОГОЛОШЕННЯ, МОНІТОРИНГ.

ABSTRACT

Explanatory note: 119 pages, 67 figures, 5 appendices, 19 sources, 8 tables.

Object of development: a web application for conducting online auctions of goods.

The aim of the qualification work is to develop and optimize the operation of a web application for conducting auctions in an online format and facilitating the process of buying and selling goods by users from different countries.

The introduction discusses the analysis and current state of the problem, specifies the purpose of the qualification work and the field of its application, provides justification for the relevance of the topic, and clarifies the formulation of the task.

In the first chapter, the subject area is analyzed, the relevance of the task and the purpose of the development are determined, the task is formulated, and requirements for software implementation, technologies, and software tools are provided.

In the second chapter, existing solutions are analyzed, a platform for development is chosen, the design and creation of a web-oriented information system are carried out, the operation of the system, algorithm, and structure of its functioning are described, as well as the process of calling and loading the application, input and output data are defined, and the composition and parameters of technical means are characterized.

In the economic section, the complexity of the developed information system is determined, the cost of creating the application is calculated, and the time required for its development is estimated.

The practical significance lies in creating a web application designed to facilitate the process of buying and selling goods by users from different countries.

The relevance of developing a web application for conducting online auctions for the sale of goods is not in doubt, as current business is increasingly moving to an online format. The main trends in e-commerce call for quick and effective solutions to support and develop businesses. The provision of such an add-on allows you to automate the purchase and sale processes, ensuring reliability and accessibility for traders from different countries of the world.

Keywords: WEB APPLICATION, AUCTION, PAGE, DATABASE, ANGULAR, BUY-SELL, ANNOUNCEMENT, MONITORING.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних;

СУБД – система управління базами даних;

SQL – Structured Query Language;

FK – Foreign Key;

DOM – Document Object Model;

HTML – Hypertext Markup Language;

CSS – Cascading Style Sheets;

JS – Java Script;

PHP – Hypertext Preprocessor;

ІТ – інформаційні технології;

ПК – персональний комп’ютер;

ПЗ – програмне забезпечення.

ВСТУП

В сучасному світі інтернет-технології стали невід'ємною частиною нашого повсякденного життя, значно змінюючи спосіб ведення бізнесу та комунікацій. Однією з важливих областей, яка зазнала значних змін завдяки інтернету, є торгівля. Веб-додатки для проведення онлайн-аукціонів стали популярними інструментами для купівлі та продажу товарів, пропонуючи зручність, доступність та широкий вибір для користувачів з усього світу.

Метою даної кваліфікаційної роботи є розробка веб-додатку для проведення онлайн-аукціонів з продажу товарів, який використовуватиме сучасні технології Angular для клієнтської частини та PHP для серверної частини. Використання цих технологій дозволяє створити потужний, ефективний та зручний у користуванні додаток, що відповідає всім сучасним вимогам до веб-додатків.

Актуальність теми обумовлена зростаючим попитом на онлайн-торгівлю та необхідністю створення високоякісних платформ, які забезпечують надійність, безпеку та зручність у користуванні. Онлайн-аукціони стали важливим інструментом для малого та середнього бізнесу, дозволяючи їм досягати ширшої аудиторії та збільшувати обсяги продажів. Для покупців такі платформи надають можливість знайти унікальні товари за вигідними цінами.

Основними завданнями даної кваліфікаційної роботи є аналіз потреб користувачів, розробка архітектури системи, створення функціональних модулів для управління аукціонами, забезпечення безпеки та захисту даних. Важливим аспектом є також забезпечення сумісності з різними операційними системами та веб-браузерами, що дозволить забезпечити максимальне охоплення користувачів.

У ході розробки буде використано передові методи та інструменти, що дозволяють забезпечити високу продуктивність та надійність системи. Angular, як сучасний фреймворк для створення односторінкових додатків, забезпечить швидкість та інтерактивність клієнтської частини. PHP, у свою чергу, використовується для створення серверної частини, що відповідає за обробку запитів, управління базами даних та забезпечення логіки додатку.

Таким чином, дана кваліфікаційна робота спрямована на створення інноваційного веб-додатку для проведення онлайн-аукціонів, який відповідатиме всім сучасним вимогам та стане надійним інструментом для користувачів з різних куточків світу.

Практична значущість даного веб-додатку полягає у створенні ефективної платформи для проведення онлайн-аукціонів з продажу товарів. Це сприятиме полегшенню процесу купівлі-продажу, забезпечуючи зручний та доступний інтерфейс для користувачів. Впровадження цього додатку дозволить підприємствам розширити ринок збуту, залучити нових клієнтів та підвищити конкурентоспроможність за рахунок автоматизації та оптимізації торгових процесів. Додаток також забезпечує безпеку та прозорість угод, що є важливим аспектом у сучасній електронній комерції.

РОЗДІЛ 1

АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1. Загальні відомості з предметної галузі

Предметна галузь онлайн-аукціонів з продажу товарів є динамічною та захоплюючою сферою електронної комерції, де учасники мають можливість придбати різноманітні товари через інтернет шляхом участі у ліцитаціях. Наведу загальні відомості з цієї галузі:

Поява онлайн-аукціонів пов'язана з появою Інтернету та розвитком електронної комерції. Перші спроби створити платформи для онлайн-торгівлі відбулися ще в 1990-х роках, коли інтернет став доступним для широкого кола користувачів. Одними з перших популярних платформ стали eBay, який представлено на рис. 1.1 та Amazon, які забезпечили можливість купувати та продавати товари через мережу Інтернет. Згодом з'явилися інші платформи, такі як Allegro, Taobao, та інші, які спеціалізувалися на різних ринках та мають свої особливості.

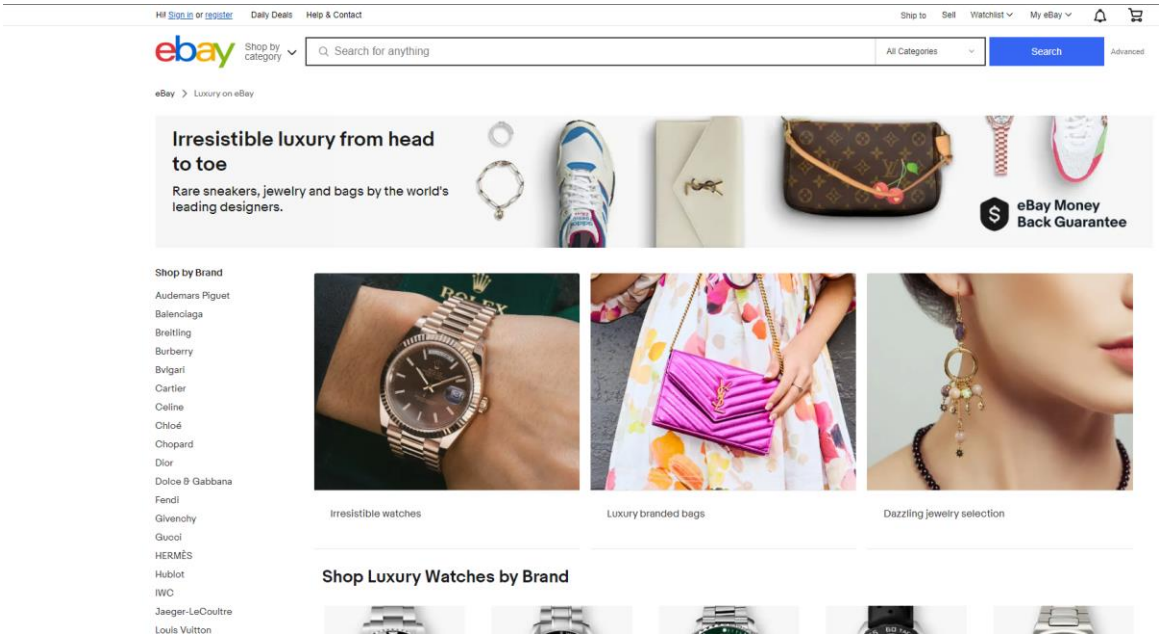

Рис. 1.1. Платформа ebay для продажу товарів

Існує кілька типів аукціонів, які різняться за своєю механікою та правилами. Наприклад, англійський аукціон - це тип аукціону, де ціна постійно зростає з кожним новим ставленням до моменту, коли ніхто не хоче підняти її більше. Голландський аукціон, навпаки, починається з високої ціни, яка поступово зменшується до того часу, коли з'явиться покупець, готовий придбати товар. Також існують аукціони з відкритим та закритим видимим результатом, де учасники можуть бачити ставки один одного або ні.

Учасники аукціонів можуть бути як фізичними особами, так і юридичними особами. Вони можуть виступати як продавці, які пропонують товари або послуги для продажу, або як покупці, які шукають товари або послуги для придбання. Крім того, на аукціонах можуть брати участь посередники, які допомагають у проведенні та координації торгівлі між продавцями та покупцями.

Онлайн-аукціони дозволяють продавцям пропонувати різноманітні товари та послуги для продажу, представлено на рис. 1.2. Це можуть бути споживчі товари, такі як електроніка, одяг, книги тощо, або послуги, такі як туристичні послуги, послуги з ремонту тощо. На деяких аукціонах можуть бути продані навіть унікальні предмети, такі як мистецькі твори, антикваріат або колекційні предмети.

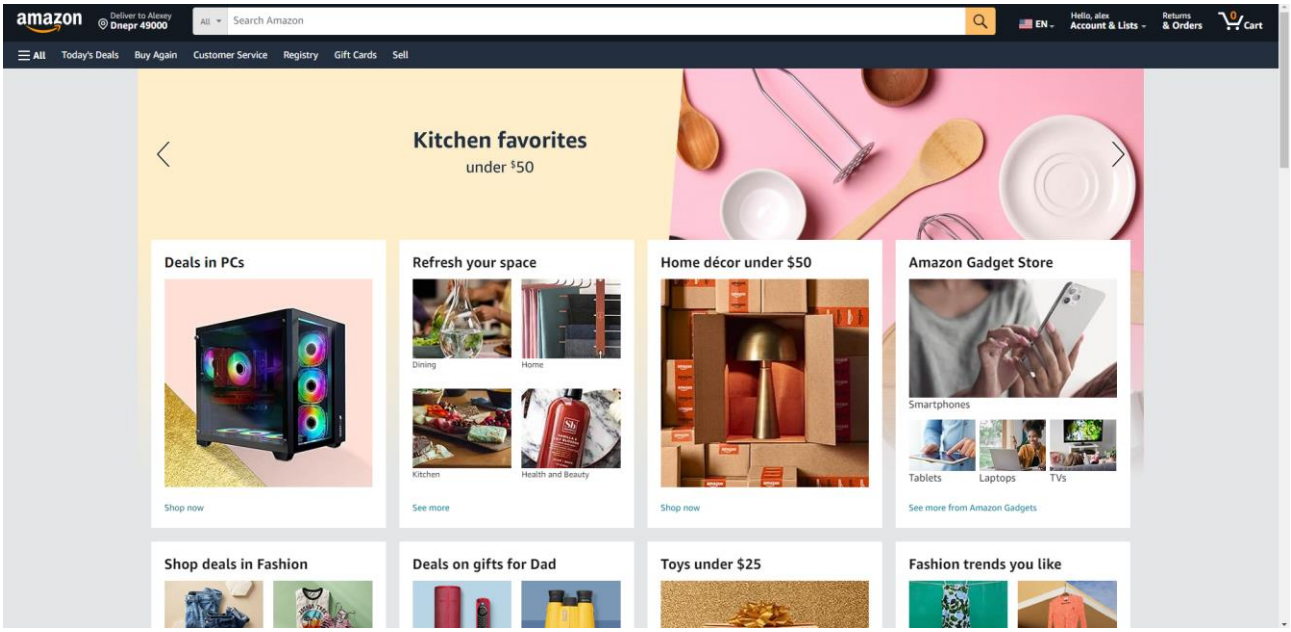

Рис. 1.2. Платформа amazon для продажу різноманітних товарів

Існує кілька бізнес-моделей для онлайн-аукціонів, що визначають, як платформа заробляє гроші. Деякі платформи беруть комісійну винагороду з продажу, інші - плату за участь у торгах, деякі надають преміальні послуги для продавців або покупців. Кожна бізнес-модель має свої переваги та недоліки, і вибір відповідної моделі залежить від стратегії розвитку платформи та специфіки ринку.

Переваги і недоліки існуючих систем можуть включати в себе ефективність та швидкість операцій, доступність для широкого кола користувачів, а також функціональність і можливості інтеграції. Однак такі системи можуть бути складними у використанні, вимагати значних витрат на впровадження та обслуговування, а також мати обмежені можливості налаштування. Дослідження цих аспектів допоможе нам краще розуміти потреби та вимоги користувачів і сприятиме успішному розвитку нашого проекту.

1.2. Призначення розробки та галузь застосування

Розробка веб-додатку для проведення онлайн-аукціонів призначена для створення ефективної, інтерактивної та надійної платформи, яка дозволяє користувачам купувати та продавати товари або послуги через Інтернет. Основними цілями цієї розробки є:

− Автоматизація торгівлі. Веб-додаток автоматизує процеси продажу та купівлі, що дозволяє зменшити часові витрати на організацію аукціонів та участь у них.

− Широкий доступ. Завдяки онлайн-доступу, учасники аукціонів можуть бути з будь-якої точки світу, що значно розширює потенційну аудиторію як для продавців, так і для покупців.

− Зручність та простота використання. Інтуїтивно зрозумілий інтерфейс та зручна навігація дозволяють користувачам легко орієнтуватися на платформі, знаходити потрібні товари та брати участь у торгах.

– Прозорість та безпека. Використання сучасних технологій шифрування та аутентифікації забезпечує безпеку персональних даних користувачів та прозорість процесу торгів.

– Розширені можливості. Додаток надає функціонал для керування аукціонами, відстеження історії торгів, аналізу ринкових тенденцій та іншого.

Розроблений веб-додаток може бути використаний у багатьох галузях, включаючи:

– Електронна комерція. Платформа може використовуватися інтернет-магазинами для продажу товарів через аукціони, що дозволяє збільшити залучення користувачів та отримати вищу ціну за товар.

– Оптова торгівля. Компанії можуть використовувати аукціони для продажу великих партій товарів, що дозволяє отримати найвищу пропозицію від покупців.

– Мистецтво та антикваріат. Платформа ідеально підходить для проведення аукціонів з продажу мистецьких творів, антикваріату та колекційних предметів, де важливі прозорість та довіра, представлено на рис. 1.3.

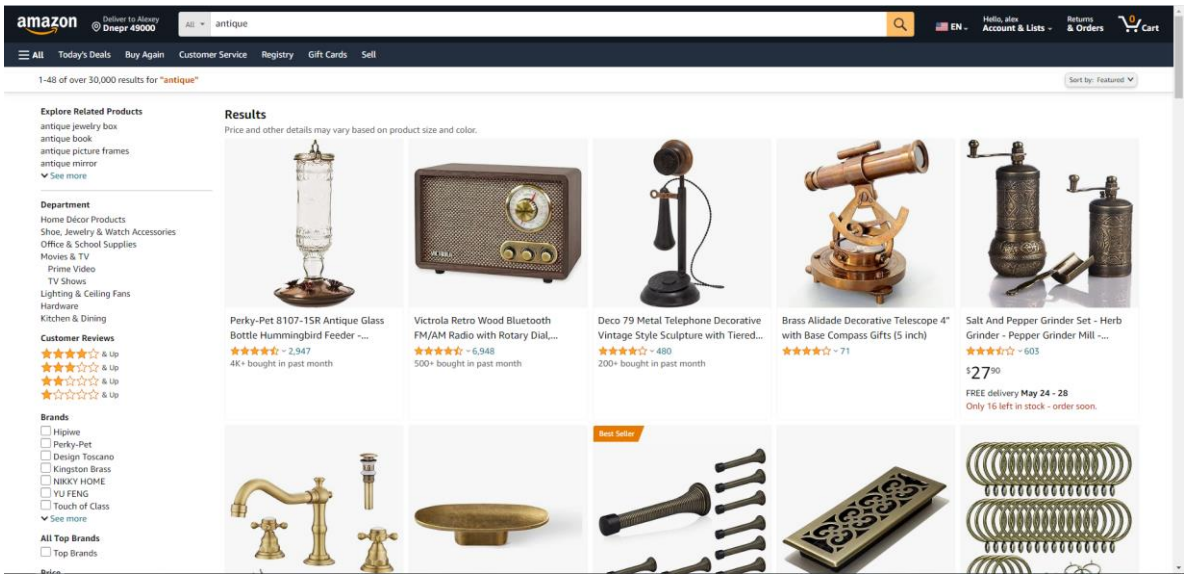

Рис. 1.3. Продаж антикваріату на платформі Amazon

– Автомобільна індустрія. Аукціони автомобілів дозволяють продавцям та покупцям торгувати новими та вживаними автомобілями з максимальною вигодою.

− Нерухомість. Платформа може бути використана для продажу нерухомості через аукціони, що дозволяє потенційним покупцям змагатися за найкращі пропозиції.

Також розробка веб-додатку включає кілька ключових технічних аспектів:

− Клієнтська частина. Використання Angular для розробки інтерфейсу користувача забезпечує динамічний та інтерактивний досвід, зручну навігацію та швидкий доступ до функціоналу.

− Серверна частина. Використання PHP для серверної логіки забезпечує надійну обробку запитів, управління базами даних та виконання бізнес-логіки. PHP є потужним інструментом для розробки веб-додатків завдяки своїй гнучкості та широким можливостям інтеграції.

− Бази даних. Використання реляційних баз даних MySQL для зберігання інформації про користувачів, товари, історію торгів та інші дані.

− Безпека. Впровадження сучасних методів шифрування даних та захисту від несанкціонованого доступу для забезпечення конфіденційності та цілісності інформації.

Таким чином, розробка веб-додатку для проведення онлайн-аукціонів з використанням Angular для клієнтської частини та PHP для серверної частини сприяє створенню ефективної, безпечної та зручної платформи для торгівлі в різних галузях.

1.3. Підстава для розробки

Відповідно до ОПП, згідно навчального плану та графіків навчального процесу, в кінці навчання студент виконує кваліфікаційну роботу (проект). Тема роботи узгоджується з керівником кваліфікаційної роботи, випускаючою кафедрою, та затверджується наказом ректора.

Отже, підставами для розробки (виконання кваліфікаційної роботи) є:

- ОПП за спеціальністю 122 «Комп’ютерні науки»;
- навчальний план та графік навчального процесу;

– наказ ректора Національного технічного університету «Дніпровська політехніка» №375-с від 23.04.2024 р;

– завдання кваліфікаційної роботи на тему «Розробка веб-додатку для проведення on-line аукціону продажу товарів, з використанням фреймворку Angular та мови програмування PHP».

1.4. Постановка завдання

Розробка веб-додатку для проведення онлайн-аукціону має на меті створення комплексного рішення, що забезпечує всі необхідні функції для організації та проведення аукціонів у режимі реального часу. Основною метою розробки є автоматизація торгівлі, надання користувачам зручного та безпечного інструменту для купівлі та продажу товарів і послуг через Інтернет. Додаток повинен бути інтуїтивно зрозумілим, адаптивним та зручним для використання. Інтерфейс користувача, розроблений за допомогою Angular, має підтримувати реєстрацію та авторизацію користувачів, перегляд активних та минулих аукціонів, можливість фільтрації та пошуку товарів, подачу ставок у реальному часі, а також відображення історії ставок та поточного лідера аукціону.

Особистий кабінет користувача забезпечить функціонал для управління профілем, перегляду власних ставок, виграних аукціонів та історії торгів. Angular [1] дозволить додатку бути високопродуктивним та інтерактивним, забезпечуючи оновлення даних у режимі реального часу без необхідності перезавантаження сторінки.

Серверна частина додатку, реалізована за допомогою PHP [2], включає систему реєстрації та авторизації користувачів, управління аукціонами, обробку ставок у реальному часі, збереження та валідацію даних. Для зберігання інформації про користувачів, аукціони та ставки використовуватимуться реляційні бази даних, оптимізовані для високої продуктивності. Важливим аспектом є забезпечення безпеки даних та захист від найпоширеніших веб-загроз, таких як SQL-ін'єкції, CSRF-атаки та XSS-атаки, а також шифрування конфіденційної інформації [3][4][5].

Створення та розробка інформаційної системи повинні включати наступні етапи:

– визначення цілей створення системи та затвердження завдання на її розробку;

– аналіз предметної області та формулювання завдання;

– розгляд технічних аспектів проектування системи, таких як визначення архітектури, файлової та логічної структури сторінок;

– розробка дизайн-макету системи;

– розробка програмного коду, модулів, бази даних та інших елементів, необхідних для проекту;

– заповнення системи контентом;

– тестування та публікація системи в Інтернеті.

1.5. Вимоги до программи або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Розробка веб-додатку для проведення онлайн-аукціонів потребує визначення конкретних функціональних характеристик, які забезпечать ефективну роботу системи та задовольнять потреби користувачів. Основні вимоги до функціональних характеристик включають наступні пункти.

Реєстрація та авторизація користувачів.

– Реєстрація нового користувача. Система повинна забезпечувати можливість реєстрації нових користувачів з введенням обов'язкових даних, таких як ім'я, адреса електронної пошти та пароль.

– Авторизація. Користувачі повинні мати можливість входити до системи за допомогою своїх облікових даних (електронної пошти та пароля).

Управління профілем користувача:

– Редагування профілю. Користувачі повинні мати можливість редагувати свої особисті дані, такі як ім'я, адреса електронної пошти, пароль та контактні дані.

– Перегляд історії торгів. Користувачі повинні мати доступ до історії своїх ставок, виграних аукціонів та інших транзакцій.

Створення та управління аукціонами.

– Створення аукціону. Продавці повинні мати можливість створювати нові аукціони, вводячи інформацію про товар (опис, фотографії, початкова ціна, мінімальний крок ставки, тривалість аукціону).

– Редагування та видалення аукціону. Продавці повинні мати можливість редагувати або видаляти свої аукціони до їх початку.

– Перегляд активних аукціонів. Усі користувачі повинні мати можливість переглядати список активних аукціонів з можливістю фільтрації та пошуку.

Участь у торгах.

– Подача ставок. користувачі повинні мати можливість подавати ставки на активні аукціони у режимі реального часу.

– Оновлення ставок у реальному часі. Інформація про поточні ставки та лідера аукціону повинна оновлюватися в режимі реального часу без необхідності перезавантаження сторінки [6].

1.5.2. Вимоги до інформаційної безпеки

Вимоги до безпеки інформації включають:

– забезпечення недоторканості даних;

– захист конфіденційності особистих даних користувача та його налаштувань;

– захист від несанкціонованого доступу до системи адміністрування.

Для ідентифікації та підтвердження особи користувача використовується унікальна комбінація електронної пошти та пароля, а також токен автентифікації.

1.5.3. Вимоги до складу та параметрів технічних засобів

Рекомендовані вимоги до апаратного та програмного забезпечення серверу.

Вимоги до серверу:

– об’єм оперативної пам’яті: 512МБ;

− операційна система: Ubunty Linux 16;

− підтримка протоколів: HTTP, WebSocket [7];

− HTTP-сервер: Nginx, Gunicorn [8];

− підтримувані бази даних: PostgresSQL версії 9.6 та вище [9].

− підтримка додаткових технологій: PHP.

Рекомендовані вимоги до програмного забезпечення клієнта програми:

− наявність пристрою з ОС 2007 року випуску та пізніше;

− наявність одного з перерахованих браузерів зі вказаною або вищою версією: Google Chrome 16, Opera 12.1, Mozilla Firefox 11, Internet Explorer 10, Apple Safari 5.

Персональний комп'ютер повинен мати доступ до мережі Інтернет.

1.5.4. Вимоги до інформаційної та програмної сумісності

Для забезпечення ефективної роботи веб-додатку для проведення онлайн-аукціонів важливо дотримуватися вимог щодо інформаційної та програмної сумісності. Додаток має підтримувати сучасні операційні системи, включаючи Windows, macOS та Linux [10] на стороні клієнта, що забезпечує широке охоплення користувачів незалежно від використовуваних пристроїв. Важливою є сумісність з останніми версіями основних веб-браузерів, таких як Google Chrome, Mozilla Firefox, Safari та Microsoft Edge, що гарантує коректне відображення та функціонування додатку.

Для зберігання та управління даними використовуються реляційні бази даних MySQL або PostgreSQL [11], які забезпечують надійність та швидкодію. Використання стандартів RESTful API дозволяє легко інтегрувати додаток з іншими сервісами та системами, що робить його більш гнучким та розширюваним. Це сприяє безперебійній роботі додатку та забезпечує його високу продуктивність та стабільність.

РОЗДІЛ 2

ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1. Функціональне призначення програми

Функціональне призначення програми полягає у створенні зручного та ефективного інструменту для проведення онлайн-аукціонів з продажу товарів. Основні функції програми включають:

− Створення аукціонів. Кліент може створювати нові аукціони, встановлювати параметри, такі як тривалість, початкова ціна, мінімальна крок ставки тощо, дивись рис. 2.1.

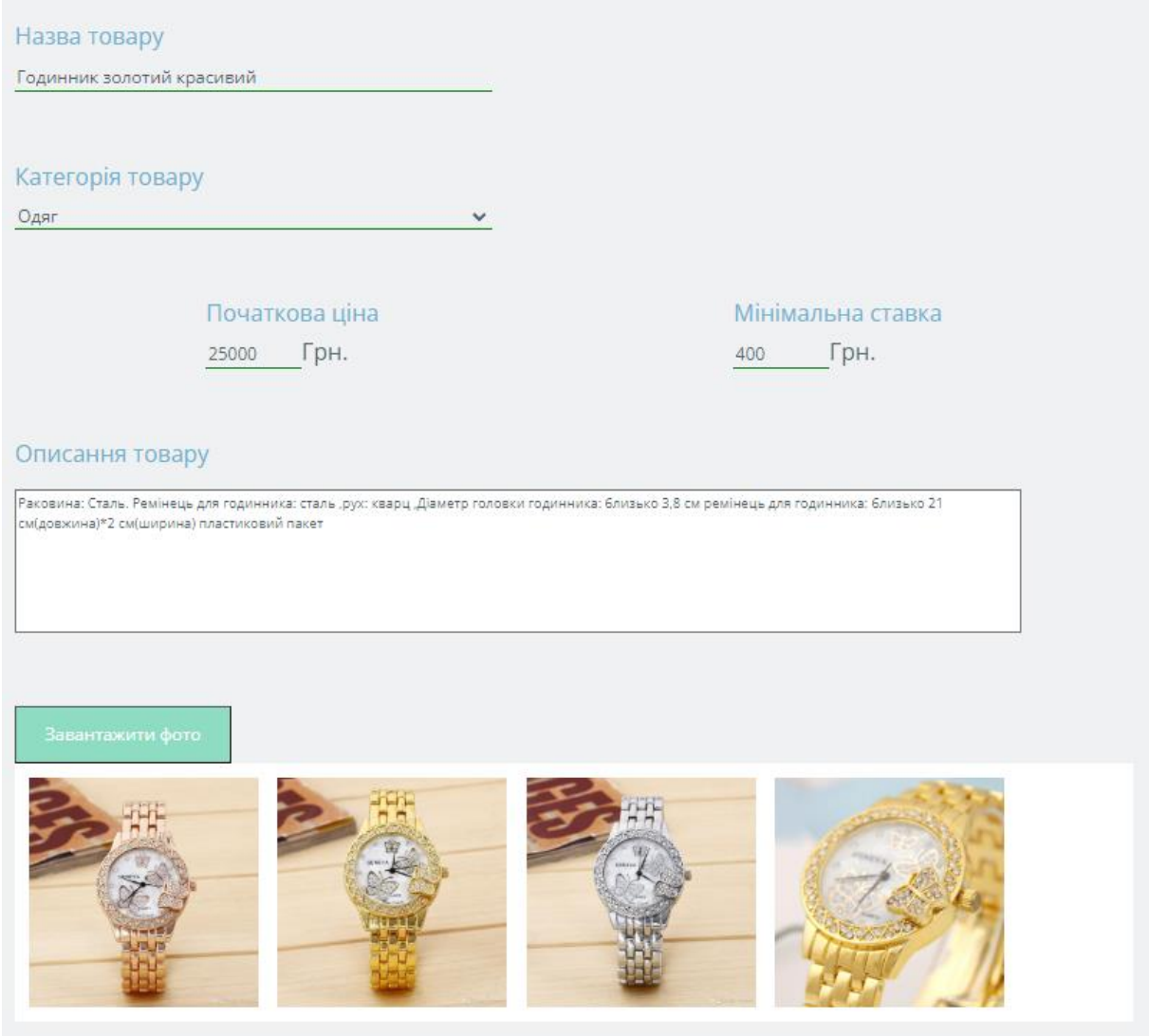

Рис. 2.1. Створення нового аукціону

− Перегляд товарів. Користувачі можуть переглядати доступні товари на аукціоні, ознайомлюватися з їх описом, зображеннями та іншою інформацією.

− Реєстрація користувачів. Користувачі можуть створювати облікові записи для участі в аукціонах, вказуючи свої контактні дані та іншу необхідну інформацію.

− Ставки та ліцитація. Користувачі можуть робити ставки на товари під час тривання аукціону. Система автоматично враховує нові ставки та відображає актуальну ціну.

− Таймер. Для кожного аукціону встановлюється таймер, що відображає час до закінчення. Користувачі можуть спостерігати за змінами та робити ставки в обмежений часовий проміжок.

− Безпека. Забезпечення безпеки та конфіденційності даних користувачів, а також захист від шахрайства та маніпуляцій з боку учасників аукціону.

Реалізація цих функцій дозволить створити потужний та привабливий інструмент для проведення онлайн-аукціонів з продажу товарів, що відповідає сучасним вимогам користувачів та бізнесу.

2.2. Опис застосованих математичних методів

Оскільки особливості предметної області розв'язуваної задачі не передбачають застосування математичних методів, при розробці веб-додатку для проведення онлайн-аукціону з продажу товарів, з використанням технологій Angular для клієнтської частини та PHP для серверної частини, математичні методи не використовувалися.

2.3. Опис використаної архітектури та шаблонів проектування

Для розробки веб-додатку для проведення онлайн-аукціонів з продажу товарів ми використовуємо модульну архітектуру, що базується на принципах Angular Framework для клієнтської частини та підходів MVC (Model-View-Controller) для серверної частини [12], яка реалізована за допомогою PHP, представлено на рис. 2.2.

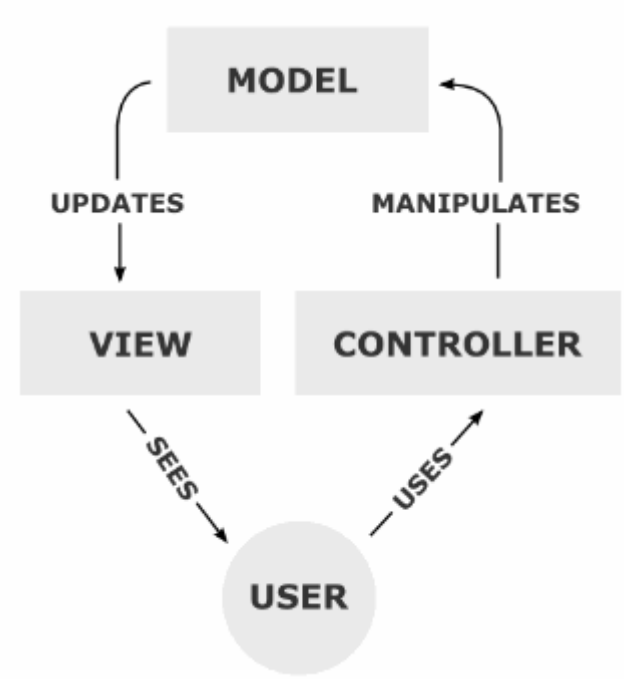

Рис. 2.2. Модульна архітектура MVC

MVC архітектура використовується для структурування серверної частини додатку. Модель відповідає за доступ до даних та логіку бізнес-процесів, Вид - за відображення інформації користувачу, а Controller - за обробку запитів користувача та взаємодію з Model і View.

Клієнтська частина (Angular).

− Модульна структура: Використано модульну структуру Angular для розділення функціональності на окремі модулі, такі як модуль аутентифікації, модуль аукціонів, модуль профілю користувача тощо. Це дозволяє підтримувати чистоту коду та полегшує його масштабованість [13].

− Компонентний підхід: Реалізовано кожен елемент інтерфейсу (компонент) як окремий об'єкт, що дозволяє легко управляти їхнім станом та поведінкою [14].

− Сервіси для зв'язку з сервером: Використано сервіси Angular для взаємодії з сервером через HTTP протокол. Це забезпечує асинхронне отримання та надсилання даних без перезавантаження сторінки.

Серверна частина (PHP).

REST API: Використано RESTful архітектуру для забезпечення зручної та ефективної комунікації між клієнтом та сервером. Кожен ресурс (наприклад,

користувач, лот або ставка) представлений у вигляді унікального URL, і можна взаємодіяти з ним за допомогою стандартних HTTP методів (GET, POST, PUT, DELETE) [15].

Шаблони проектування.

− Facade це шаблон проектування, який використовується для створення простого та зрозумілого інтерфейсу для взаємодії клієнта з системою аукціону, приховуючи складність внутрішньої реалізації. Однією з основних переваг цього шаблону є спрощення використання системи, оскільки фасад надає спрощений інтерфейс для складної системи, роблячи її більш зручною для клієнта. Крім того, фасад зменшує кількість залежностей між клієнтом і складними підсистемами, що сприяє покращенню організації коду. Завдяки фасаду зміни в підсистемах не впливають на клієнтську частину, оскільки клієнт взаємодіє лише з фасадом, що полегшує модифікації системи. Однак фасад має і недоліки, такі як обмежена функціональність, оскільки він може не надавати доступу до всіх функцій підсистеми, що обмежує можливості клієнта. Крім того, додатковий рівень абстракції, що додається з використанням фасаду, може збільшити загальну складність системи. У випадках, коли фасад є єдиною точкою доступу, це може створити "вузьке місце" в системі, що негативно впливає на продуктивність.

− Singleton це шаблон проектування, який забезпечує, що певний клас має лише один екземпляр і надає глобальну точку доступу до цього екземпляра. Це особливо корисно для об'єктів, які повинні існувати в єдиному екземплярі протягом життєвого циклу додатку, таких як сервіси, налаштування або менеджери ресурсів. Основні переваги цього шаблону полягають у контрольованому доступі, оскільки одинак забезпечує єдину точку доступу до об'єкта, що спрощує контроль над його станом і поведінкою. Глобальна доступність екземпляра класу одинак робить його зручним для використання в різних контекстах програми. Використання одного екземпляра об'єкта допомагає зекономити ресурси, особливо в випадках, коли створення нових екземплярів є ресурсомістким. Однак шаблон одинак також має недоліки. Він може ускладнити тестування, оскільки створення та використання глобального екземпляра може

призвести до небажаних залежностей між тестами. Крім того, якщо одинак не реалізований належним чином, можуть виникнути проблеми з багатопоточністю, що може спричинити некоректну роботу програми в умовах багатозадачності..

Використання таких архітектурних підходів та шаблонів проектування дозволяє покращити розширюваність, підтримку та ефективність веб-додатку для проведення онлайн-аукціонів.

2.4. Опис використаних технологій та мов програмування

Для створення програмного продукту було обрано наступні мови програмування: Angular 11 та Php для реалізації сервера та фреймворк Bootstrap 4, який працює з HTML 5 та CSS 3 для створення клієнтського інтерфейсу. Для додавання динаміки на сторінки додатку використано Angular та його інструмент jQuery [16].

Мову програмування Angular було обрано з наступних причин:

Підтримка концепції MTV (Model - Template - View).

– Angular додаток дозволяє розмежувати бізнес-логіку проекту від логіки відображення та збереження даних. MTV розшифровується як:

– M. Модель(Model), шар доступу до даних. Цей шар знає все про дані: як отримати до них доступ, як перевірити їх, як з ними працювати і як дані пов'язані між собою.

– T. Шаблон (Template), шар представлення даних. Цей шар приймає рішення відносно представлення даних: як і що повинно відображатися на сторінці або в іншому типі документу.

– V. Представлення (View), шар бізнес-логіки. Цей шар містить логіку, як діставати доступ до моделей і застосовувати відповідний шаблон. Це «міст» між моделями та шаблонами.

Вбудований ORM (Object-relational mapper)

Механізм об'єктно-орієнтованого відображення, що дозволяє автоматизувати типові взаємодії з базою даних і використовує об'єктно-орієнтований підхід замість інструкцій SQL. Таблиці бази даних представлені у

вигляді класів, де властивості — це поля таблиці, а запис у таблиці — об’єкт класу.

Висока швидкість роботи

Незважаючи на то, що PHP не досить швидка мова програмування. Angular-додаток може витримувати високе навантаження, а також має вбудовані можливості кешування і розподілу навантаження.

Для реалізації відображення web-сторінок було використано HTML (HyperText Markup Language) – спеціальну мову гіпертекстової розмітки, що застосовується при створенні сайтів в інтернеті. HTML-документ задає стандартну структуру документу, що складається з окремих блоків, заданих тегами [17].

Для реалізації відображення web-сторінок було використано CSS фреймворк Bootstrap. Від дозволяє швидко розробляти макети сторінок сайту та містить величезний набір готових рішень і елементів.

Серед найбільших переваг фреймворку слід вказати великий набір можливостей для створення кросбраузерного та адаптивного сайту.

Bootstrap пропонує використання сітки, що допомагає в створенні адаптивного макету. Сітка дозволяє задавати класи для блоків, які вказують, яку ширину повинен займати елемент і як він відображатиметься на різних пристроях [18].

Сітка функціонує як таблиця, в якій є свої ряди і стовпці. Рядки можуть бути поділені на максимум на стовпців 12, що задовольняє більшу частину потреб web-дизайну. Для створення адаптивного дизайну потрібно вказувати яку кількість стовпців займатиме блок на великому, середньому та маленькому екранах. Наприклад, на комп’ютері список новин може відображатися у чотири стовбці, на планшетному-комп’ютері у два, а на мобільному пристрої тільки в один стовбець, що полегшує перегляд та взаємодію з сайтом. Візуальний приклад сітки Bootstrap, представлено на рис. 2.3.

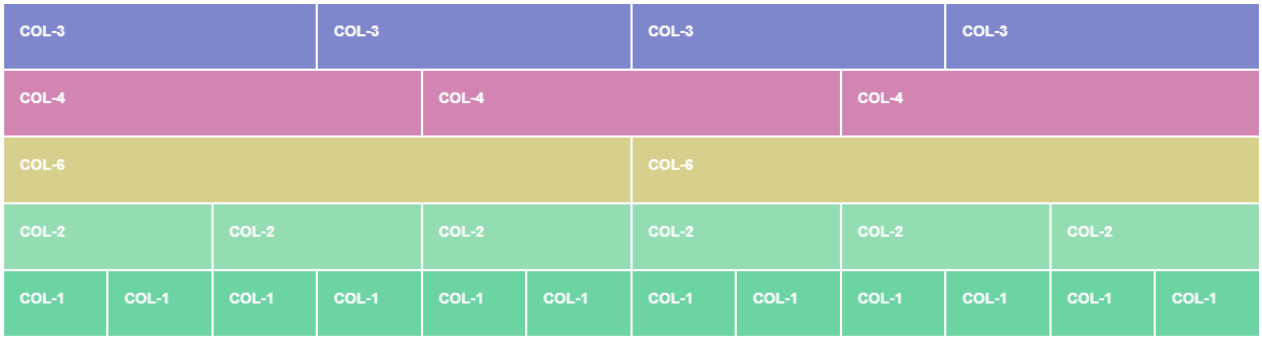

Рис. 2.3. Bootstrap пропонує зручний поділ сторінки на окремі клітини

Окрім сітки в Bootstrap існує величезна кількість всіляких компонентів: навігаційні меню, форми, таблиці, модальні вікна, вкладки, сповіщення, спливаючі підказки і т.д.

Для додання динаміки на сайт було обрано інструмент Angular – jQuery 3.3 — бібліотеку JavaScript, що фокусується на взаємодії JavaScript і HTML. jQuery допомагає легко діставати доступ до будь-якого елементу DOM, звертатися до атрибутів і вмісту елементів DOM, маніпулювати ними. Також бібліотека jQuery надає зручний API по роботі з Ajax [6].

JQuery відділяє поведінку від структури HTML. Наприклад, замість прямої вказівки на обробник події натиснення кнопки, управління передається JQuery, ідентифікуючої кнопки і потім перетворює його в обробник події кліку. Таке розділення поведінки і структури також називається принципом ненав'язливого JavaScript.

У якості середовищ розробки було розглянуто наступні програмні продукти: WebStorm IDE, інтегроване середовище Eclipse з використанням плагіна NodeDev та швидкий кросплатформенний текстовий редактор Sublime Text.

Після дослідження обраних середовищ розробки було обрано WebStorm, який має вбудовану підтримку мови програмування Angular та фреймворка Php, а також включає набір зручних інструментів для проектування, планування та реалізації програмного продукту. Вікно редактору представлено на рис. 2.4.

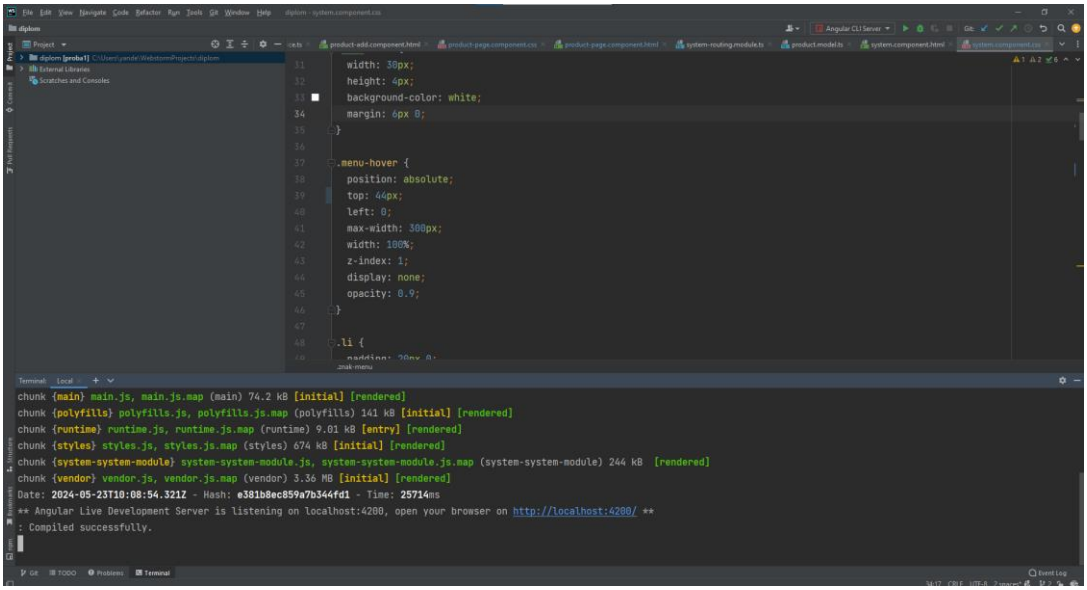

Рис. 2.4. Вікно інтегрованого середовища розробки WebStorm

WebStorm містить панель інструментів з командами для запуску, зупинки програми, запуску відгадчику, збереження змін проекту і т.д. Зліва екрана знаходиться дерево проекту, зручне для навігації та пошуку потрібних тек та файлів.

Внизу екрану знаходиться набір вкладок для відкриття консолі для Angular, php, SSH, відладчика та баз даних [8]. Є можливість створювати декілька терміналів для запуску команд та маніпулювання проектом.

Для перегляду поточної бази даних існує вкладника в правому кінці екрану. Вона дозволяє переглядати та відкривати таблиці БД, будувати ER-діаграми, тощо.

IDE включає потужний рефакторинг коду, який надає широкі можливості по виконанню швидких глобальних змін в проекті [19].

WebStorm включає редактори Javascript, Coffescript, HTML/CSS, SASS, LESS, HAML та дозволяє підключати проект к системі контролю версій.

Це потужний і функціональний редактор коду з підсвічуванням синтаксису, автоформатуванням і автовідступами для підтримуваних мов. WebStorm пропонує просту і потужну навігацію, допомагає при написанні коду, підтримуючи такі функції: автодоповнення, авто-імпорт, шаблони коду, перевірка на сумісність версії інтерпретатора мови і т.д.

2.5 Опис структури програми та алгоритмів її функціонування

Для розробці проекту будо реалізовано базу даних, схема якої наведена на рис. 2.5. Повну схему бази даних наведено у додатку А.

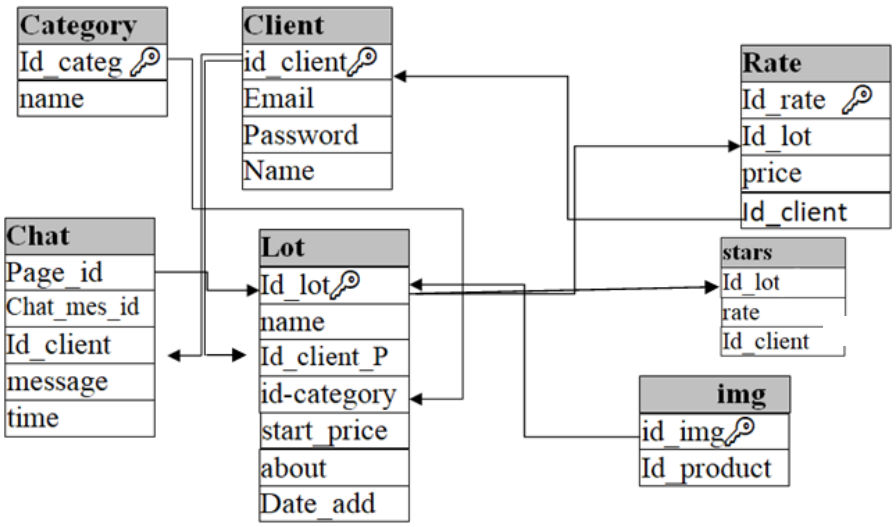

Рис. 2.5. ER-діаграма проекту

ER-діаграма включає в себе 7 таблиць: «Категорія» (Category), «Користувач» (Client), «Ставка» (Rate) , «Бесіда» (Chat), «Оголошення» (Lot) , «Рейтинг лоту» (Stars) та таблицю «Зображення» (img).

Таблиця Chat містить в собі дані про бесіду у оголошенні. Повідомлення та ідентифікатор користувача який іх залишив повідомлення. Має зв'язок з таблицею Lot один до багатьох.

Опис полів всіх Chat таблиці представлено у таблиці 2.1.

Таблиця 2.1

«Бесіда»

Поле	Призначення
Page_id	Ідентифікатор чату
Chat_mes_id	Ідентифікатор повідомлення
Id_client	Ідентифікатор користувача який залишив повідомлення
message	Повідомлення
time	Час відправлення повідомлення

Таблиця Lot включає в себе дані про оголошення, назву оголошення, стартову ціну та додаткову інформацію про оголошення. Опис полів представлено у таблиці 2.2.

Таблиця 2.2

«Оголошення»

Поле	Призначення
Id_lot	Ідентифікатор оголошення
name	Назва оголошення
Id_client_p	Ідентифікатор продавця
Id_category	Ідентифікатор категорії
Start_price	Початкова ціна
about	Інформація про оголошеня
Date_add	Час створення оголошення

Таблиця Category містить список категорій для вибору типа категорії оголошення. Опис полів представлено у таблиці 2.3.

Таблиця 2.3

«Категорія»

Поле	Призначення
Id_category	Ключовий атрибут
name	Назва категорії

Таблиця img містить фото-додатки до оголошення. Так як одне оголошення може мати кілька фото-додатків, між таблицею Lot та img встановлено зв’язок «один до багатьох». Опис полів представлено у таблиці 2.4.

Таблиця 2.4

«Зображення»

Поле	Призначення
Id_product	Вказівник на поле таблиці Lot
Id_img	Ідентифікатор зображення

Таблиця Stars містить дані про оцінку оголошення та має зв’язок «один до багатьох» з таблицею Lot.

Опис полів представлено у таблиці 2.5.

Таблиця 2.5

«Рейтинг лоту»

Поле	Призначення
Lot_id	Вказівник на поле таблиці Lot
Id_client	Ідентифікатор користувача який залишив повідомлення
rating	Оцінка оголошення

Таблиця Rate описує дані про ставки на оголошення. Опис полів представлено у таблиці 2.6.

Таблиця 2.6

«Ставка»

Поле	Призначення
Id_rate	Ідентифікатор ставки
Id_lot	Вказівник на поле таблиці Lot
price	Ставка
Id_client	Вказівник на поле таблиці Client

Таблиця Client описує дані про користувачів web-додатку. Опис полів представлено у таблиці 2.7.

Таблиця 2.7

«Користувач»

Поле	Призначення
Id_Client	Ідентифікатор користувача
Email	Поштова адреса користувача
password	Пароль для входу
Name	Ім’я користувача

Повну ER-діаграму бази даних представлено у додатку А.

2.5.1. Опис проектування інтерфейсу програми

Додаток складається з десяти основних груп відображень. Основні групи відображень являють собою окремі сторінки, або частини сторінок, призначення яких наведено у таблиці 2.8.

Таблиця 2.8

Таблиця відображень сторінки

Назва	Призначення
user/rates	Перегляд ставок на оголошення
user/lots	Створення, редагування, видалення оголошень
user/create/	Створення оголошення користувачами
user/edit/	Редагування оголошень користувачів
reg/registration/	Реєстрація нового користувача
reg/login/	Вхід користувачів у додаток
lot/chat/	Обмін повідомленнями між користувачами
lot /search	Пошук оголошень
lot/info	Інформація про оголошення
lot/rate/	Оцінка оголошень користувачами

Більш розширена карта сайту зображена на рис. 2.6.

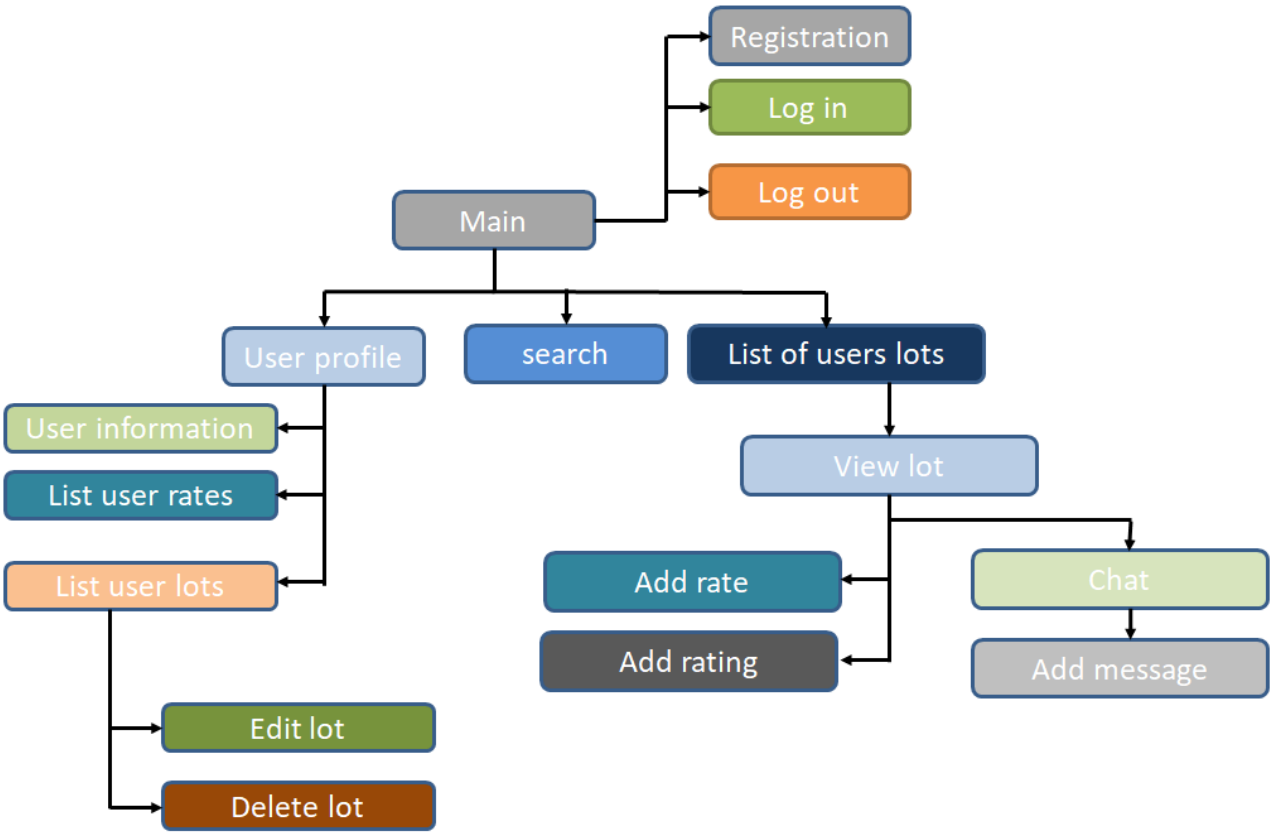

Рис. 2.6. Карта сайту

Карта сайту має чотири основні частини, вони відповідають за такі функції:

1. Сторінка користувача («UserProfile») містить у собі інструменти створення та редагування оголошень, перегляд ставок а також перегляд інфомаціїї про користувача.

2. Модуль аутентинтифікації («Registration») об’єднує функціонал створення сторінки користувача та входження до сторінки користувача.

3. Модуль обміну повідомленнями («Chats») відповідає за організацію комунікації між продавцем та покупцем.

4. Модуль пошуку оголошень («LotsSearch») дозволяє користувачам шукати оголошення використовуючи фільтри.

Повний програмний код web-додатку представлено у додатку Б.

2.5.2. Проектування користувацького інтерфейсу програми

Інтерфейс web-додатку складається з набору web-сторінок, що представляють собою документи, структура яких задана спеціальною мовою гіпертекстової розмітки HTML. Стиль сторінок задається використанням CSS — каскадних таблиці стилів, а за динамику та реакцію на певні дії користувача може відповідати скриптова мова форматування JavaScript.

Для оптимізації роботи сучасних web-сторінок використовується технологія AJAX (Asynchronous JavaScript And XML), що дозволяє здійснювати запити до серверу, не перезавантажуючи цілком сторінку, а лише довантажувати необхідні дані і змінювати потрібні частини сторінки. Це робить взаємодію з користувачем більш інтерактивною і продуктивною.

На web-сторінках можуть розміщуватися форми, за допомогою яких користувач має можливість вносити у систему та відправляти на сервер певні дані або запити. Форма складається з наборів елементів управління (віджетів). Кожен елемент має набір властивостей, що задають його ім’я, тип, стиль та певні обмеження на введені дані.

Інтерфейс web-додатку даного курсового проекту складається наступних модулів:

- Auth_module — модуль авторизації та регістрації користувача;
- login_page — сторінка для введення даних, щоб увійти у свій кабінет;
- registration_page — сторінка для введення даних, щоб створити свій власний кабінет;
- System_module — модуль сторінок функціоналу сайту;
- Main_page — головна сторінка сайту;
- Product_page — сторінка для перегляду інформації про товар;
- Product_add — форма для додавання товару на сайт;
- Search_Product — форма пошуку товарів на сайті;
- System_client — модуль сторінок кабінету користувача;
- Client_Bets — форма зі ставками користувача на товари;

– Client_Wins — форма з виграшами користувача на товари;

– Client_page — головна сторінка кабінету користувача;

– Client_products —сторінка користувача з товарами які він створив.

2.5.3. Опис складових частин програми

Модуль auth складається з наступних компонентів:

– Login.component — компонент логіну для входу у свій кабінет;

– Registration.component — компонент реєстрації для створення свого власного кабінету.

Перехід між компонентами модулю реалізовано засобами angular за допомогою роутінг модулю та команди router-outlet.

Форма Login складається з наступних компонентів:

– email — поле для введення email користувача який входить у свій кабінет, рис. 2.7;

– password — поле для введення паролю користувача який входить у свій кабінет, рис. 2.8;

– buttonLogin — кнопка для відправлення даних на сервер, рис. 2.9;

– ClientName — им’я користувача якщо він тільки що зареєструвався, рис. 3.4;

– ToRegistration — посилання на форму регістрації якщо у користувача ще немає свого власного кабінету, рис. 2.10.

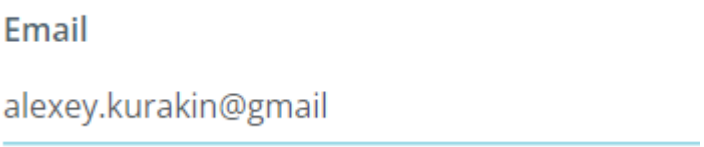

Рис. 2.7. Компонент введення пошти

Рис. 2.8. Компонент введення паролю

Рис. 2.9. Кнопка відправки данних

Рис. 2.10. Посилання на форму реєстрації

Для форми було назначено наступні обробники подій:

− EmailValid() — для перевірки даних у полі «Email» на відповідність шаблону;

− EmptyEmail() — для заборони відправки форми якщо поле «Email» порожне;

− PasswordValid() — для перевірки даних у полі «Пароль» на відповідність шаблону;

− EmptyPassword() — для заборони відправки форми якщо поле «Пароль» порожне;

− SubmitForm() — обробник відправки даних форми на сервер, виконує загальну перевірку введених даних на порожні та некоректні значення.

Усі перевірки на коректність введених даних було реалізовано средствами angular.

Форма Registration складається з наступних компонентів:

− email — поле для введення email користувача який реєструється, аналогічне полю форми Login;

− password — поле для введення паролю користувача який реєструється, аналогічне полю форми Login;

− name — поле для введення імені користувача який реєструється, рис. 2.11;

− Rules_checkBox — перемикач для підтвердження, що користувач згоден з правилами сайту, рис. 2.12;

− ButtonLogin — кнопка для відправлення даних о новом користуваче на сервер, рис. 2.13;

− ButtonLogin — посилання на форму логіну якщо у користувача вже є свій власний кабінет, рис. 2.14.

Рис. 2.11. Компонент введення ім’я користувача

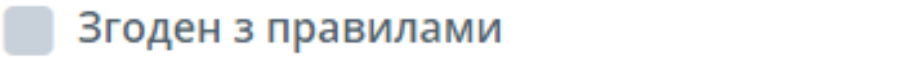

Рис. 2.12. Перемикач згоди з правилами сайту

Рис. 2.13. Кнопка відправки данних

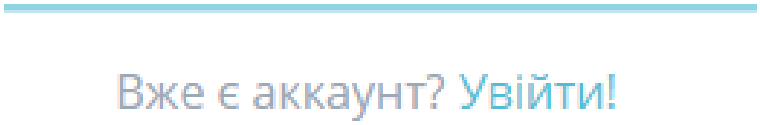

Рис. 2.14. Посилання на форму логіну

Для форми було назначено наступні обробники подій:

− EmailValid() — для перевірки даних у полі «Email» на відповідність шаблону;

− EmptyEmail() — для заборони відправки форми якщо поле «Email» порожне;

− PasswordValid() — для перевірки даних у полі «Пароль» на відповідність шаблону;

− EmptyPassword() — для заборони відправки форми якщо поле «Пароль» порожне;

− EmptyName() — для заборони відправки форми якщо поле «Имя» порожне;

− ValidationChecked() — для заборони відправки форми якщо поле «Имя» порожне;

− SubmitForm() — обробник відправки даних форми на сервер, виконує загальну перевірку введених даних на порожні та некоректні значення.

Модуль system складається з наступних компонентів:

− system.component — загальний компонент, містить верх та низ сторінки;

− client-page.component — компонент власного кабінету користувача;

− main-page.component — компонент виводу головної сторінки;

− product-add.component — компонент додавання товару на сайт;

− product-page.component — компонент виводу інформації про товар.

Форма System складається з наступних компонентів:

− categorySlider — випадаючий список вибору категорії товару, рис. 2.15;

− Logo — посилання на головну сторінку, рис. 2.16;

− SearchProduct — поле для пошуку товарів на сайті, рис. 2.17;

− Profile — посилання на профіль, рис. 2.18;

− info — посилання на сторінку о компанії, рис. 2.19.

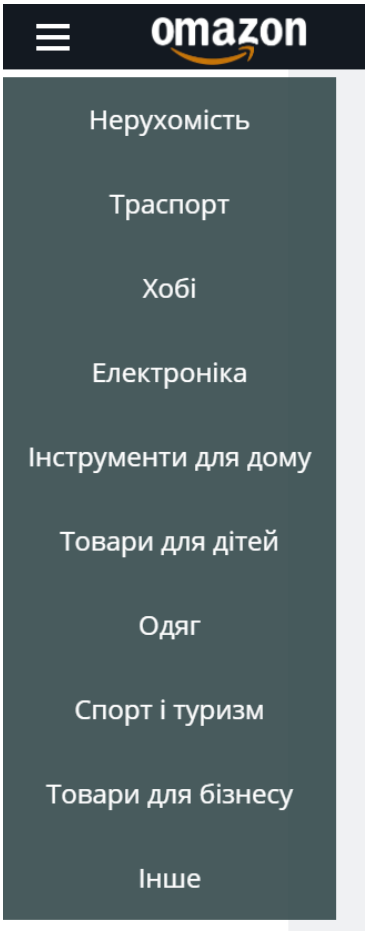

Рис. 2.15. Випадаючий список

Рис. 2.16. Посилання на головку

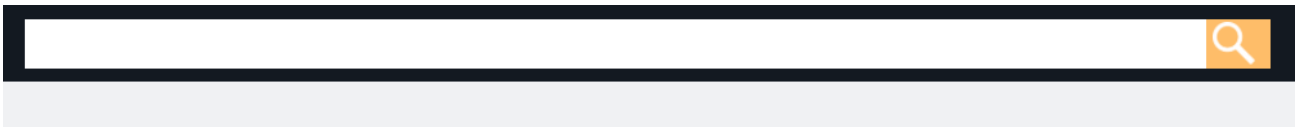

Рис. 2.17. Поле пошуку товарів

Рис. 2.18. Посилання на профіль

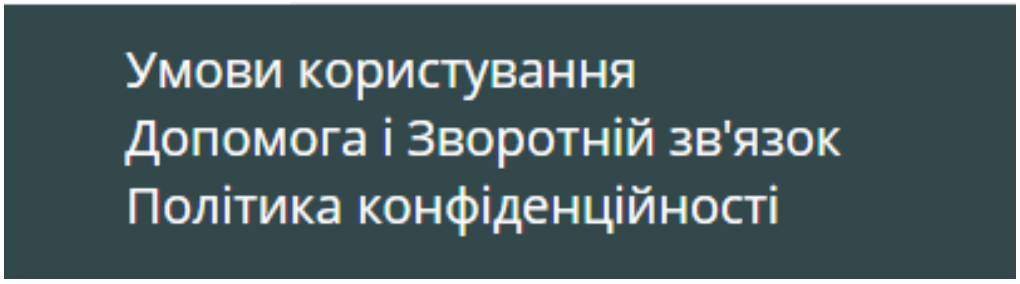

Рис. 2.19. Посилання на сторінку о компанії

Для форми було назначено наступні обробники подій:

− EnableCategory() — для виведення випадаючого списку меню категорій товарів;

− SearchBy() — для пошуку товарів на сайті.

Компонент Main складається з наступних компонентів:

− BlockCategory — блок, що складається з переліку декількох товарів відповідних категорії цього блоку з посиланням на весь перелік товарів цієї категорії, рис. 2.20;

− productHref — зображення з посиланням на відповідний товар, рис. 2.21;

− ToCategory — посилання на відповідну блоку категорію, рис. 2.22.

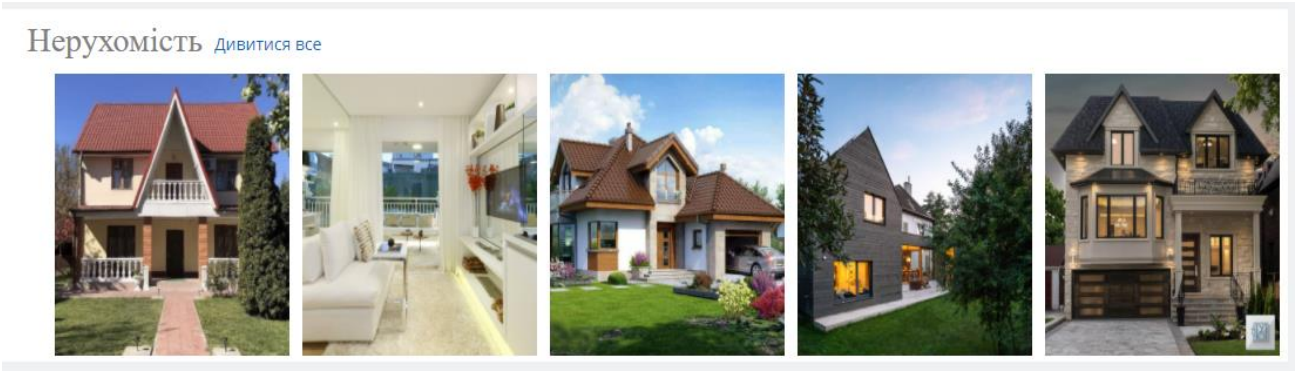

Рис. 2.20. Блок товарів

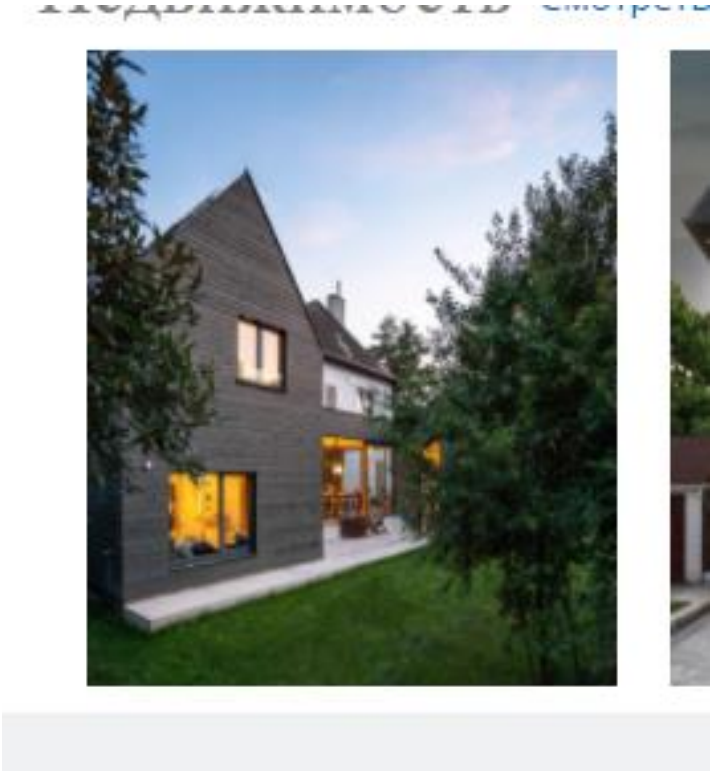

Рис. 2.21. Зображення з посиланням на товар

Рис. 2.22. Посилання на категорію товарів

Для форми було назначено наступні обробники подій:

– ToProduct() — для того щоб при кліці на картинку користувач попадав на відповідний цієї картинці товар.

Засобами Angular панель виведення товарів була організована за допомогою циклу for.

Компонент Product складається з наступних компонентів:

– ImagesProd — Зображення товару, рис. 2.23;

– BuyLot — блок товару з можливістю підняти ставку, рис. 2.24;

– AboutProduct — блок товару з інформацією про товар, рис. 2.25;

– ClientProd — блок товару з інформацією про продавця, рис. 2.26.

Рис. 2.23. Зображення товару

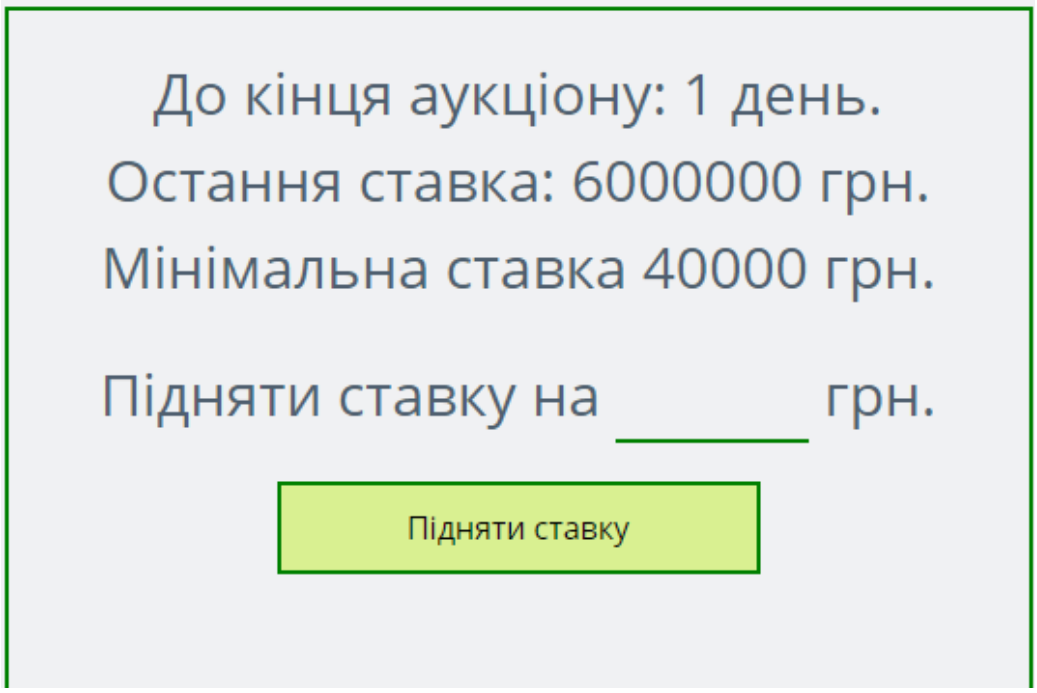

Рис. 2.24. Блок торгів за товар

Рис. 2.25. Блок інформації про товар

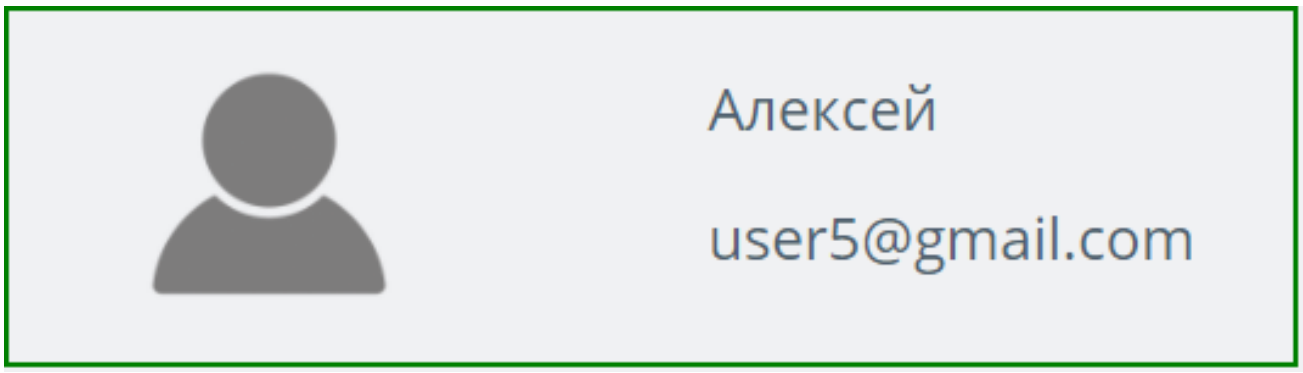

Рис. 2.26. Блок інформації про продавця

Для форми було назначено наступні обробники подій:

– Bet() — для добавлення своєї ставки на товар.

Компонент Client складається з наступних компонентів:

– NavigationClient — блок з посиланнями на додавання товару, перегляд виграшів, перегляд товарів на які зроблена ставка та ще йдуть торги, перегляд об’яв клиенту, рис. 2.27;

– ClientBio — блок з переглядом інформації про користувача, рис. 2.28;

– Bets — блок з переглядом інформації про останні 5 ставок користувача, рис. 2.29;

– Wins — блок з переглядом інформації про останні 5 виграшів користувача, рис. 2.30.

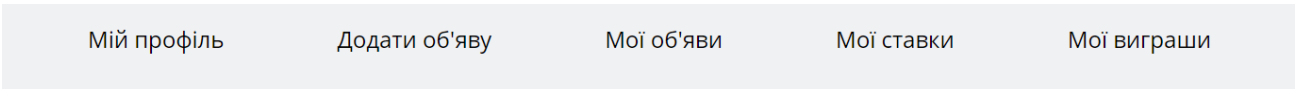

Рис. 2.27. Блок навігації

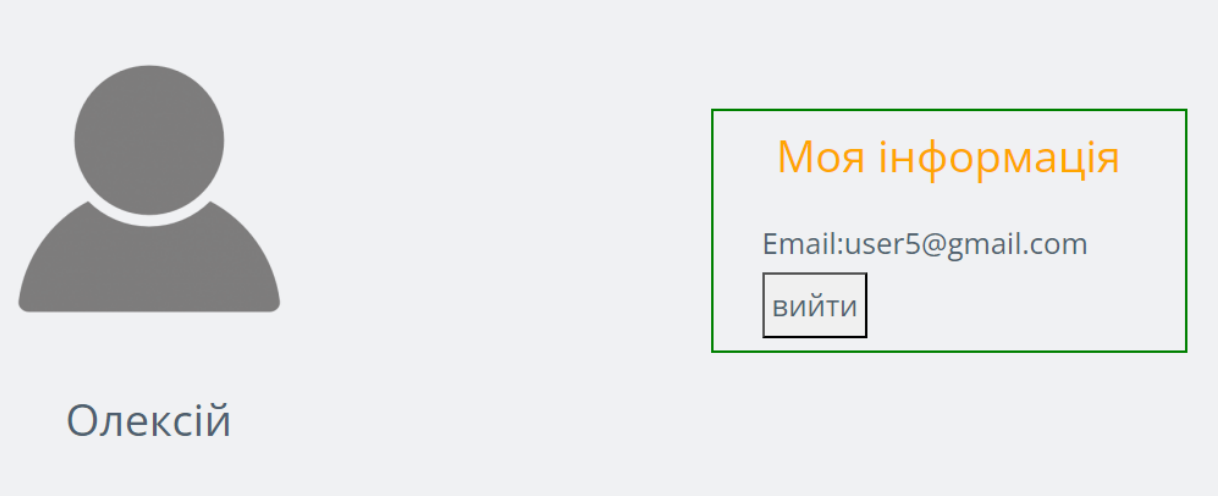

Рис. 2.28. Блок інформації про продавця

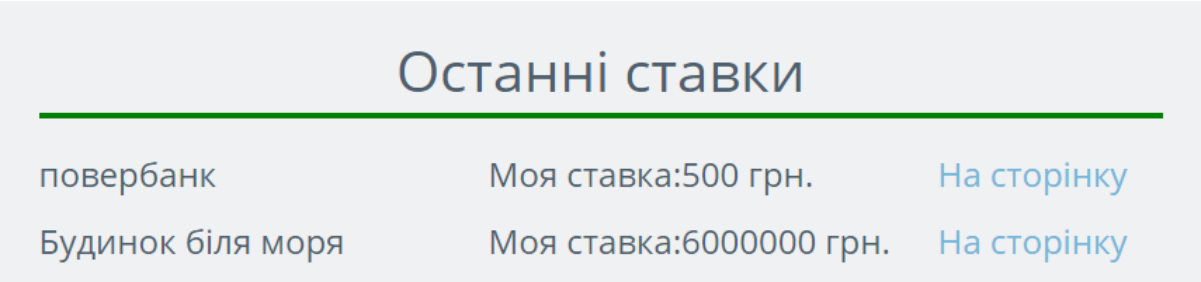

Рис. 2.29. Блок останніх ставок

Рис. 2.30. Блок останніх виграшів

Для форми було назначено наступні обробники подій:

- GetClient() — для отримання інформації про користувача;
- GetBets() — для отримання інформації про останні 5 ставок користувача;

– GetWins() — для отримання інформації про останні 5 виграшів користувача.

Компонент ProductAdd складається з наступних компонентів:

– NameProduct — поле для введення назви товару який додається, рис. 2.31;

– CategorySelect — поле для вибору віподвідної категорії для товару, рис. 2.32;

– StartPrice — поле для вибору початкової ціни для товару, рис. 2.33;

– MinPrice — поле для вибору мінімальної ставки для підняття ціни товару, рис. 2.34;

– About — поле введення інформації про товар(опис, характеристики тощо), рис. 2.35;

– DownloadImg — кнопка для завантаження фотографій, рис. 2.36;

– SendData — кнопка для завантаження фотографій, рис. 2.37.

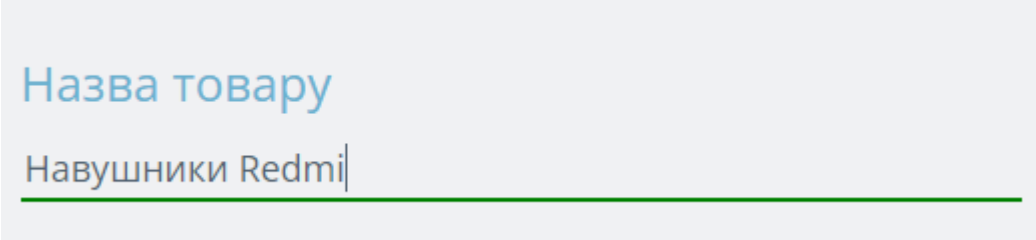

Рис. 2.31. Компонент введення назви товару

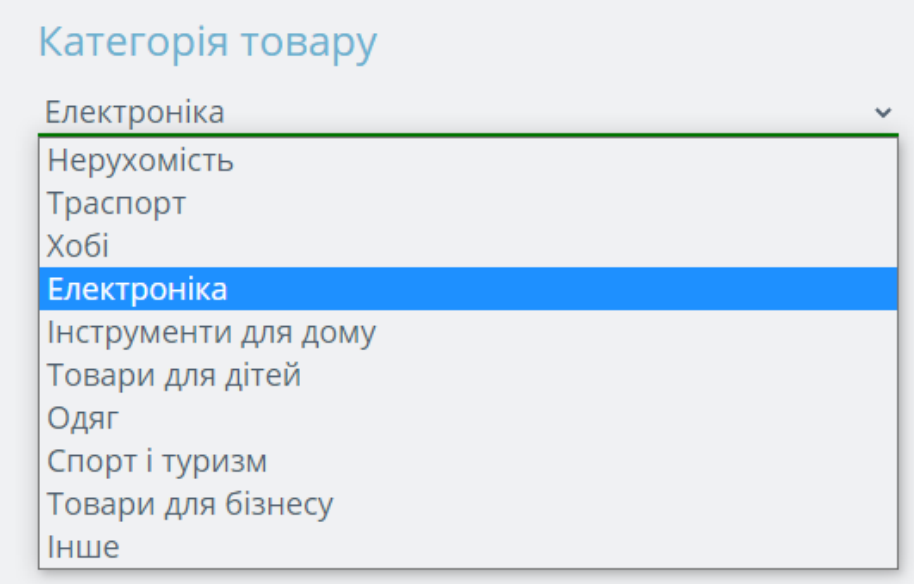

Рис. 2.32. Компонент вибору категорії

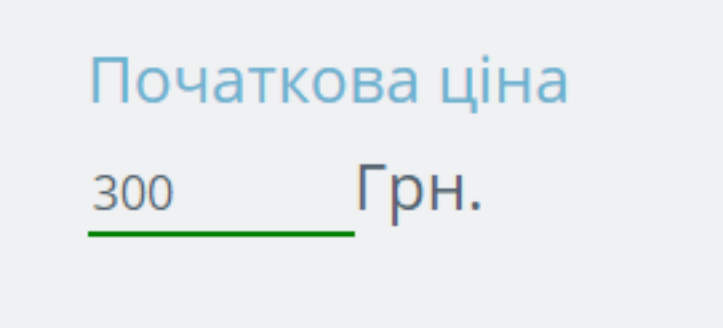

Рис. 2.33. Компонент введення початкової ставки

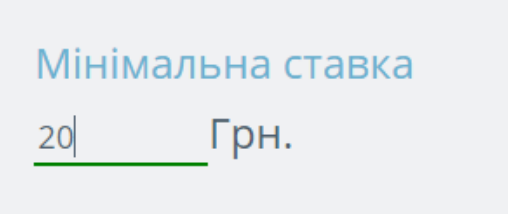

Рис. 2.34. Компонент введення мінімальної ставки

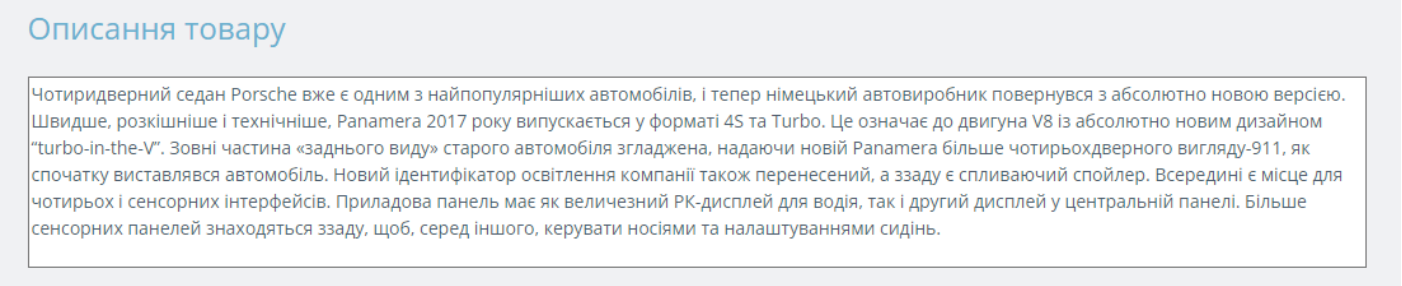

Рис. 2.35. Компонент введення інформації про товар

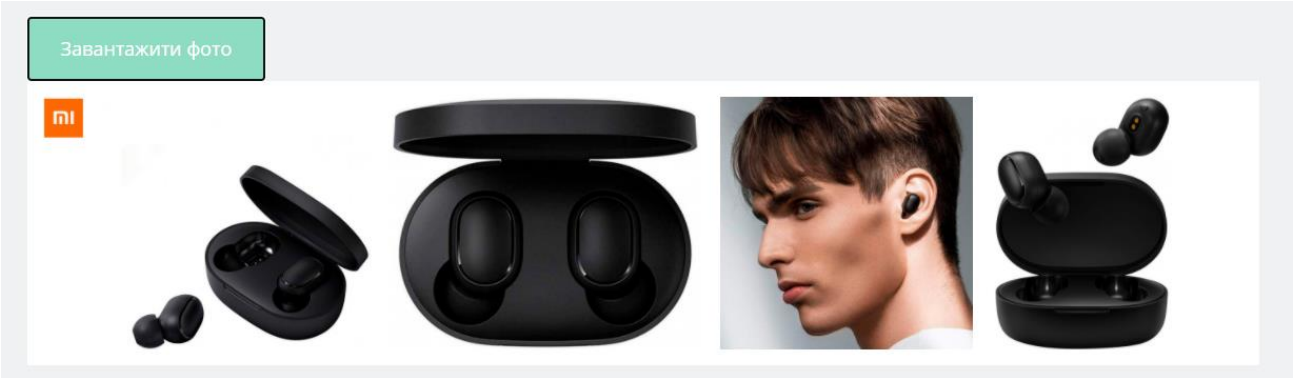

Рис. 2.36. Компонент для додавання фото товару

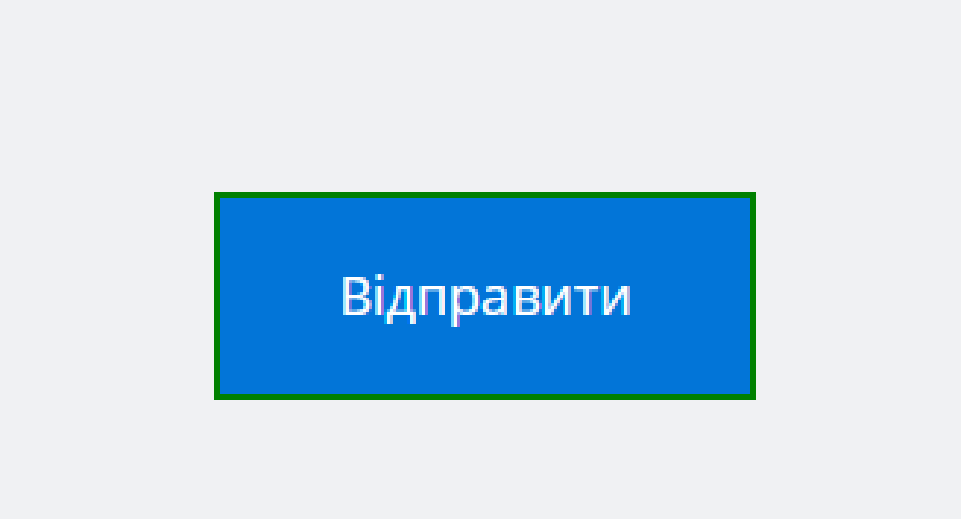

Рис. 2.37. Кнопка для відправлення даних на сервер

Для форми було назначено наступні обробники подій:

− NameValid() — для перевірки даних у полі «название товара» на відповідність шаблону;

− EmptyName() — для заборони відправки форми якщо поле «название товара» порожне;

− EmptyCategory() — для заборони відправки форми якщо поле «Категория товара» порожне;

− PriceValid() — для перевірки даних у полі «ціна» на відповідність шаблону;

− EmptyPrice() — для заборони відправки форми якщо поле «ціна» порожне;

− MinBetValid() — для перевірки даних у полі «мінімальна ставка» на відповідність шаблону;

− EmptyMinBet() — для заборони відправки форми якщо поле «мінімальна ставка» порожне;

− EmptyAbout() — для заборони відправки форми якщо поле «Описание» порожне;

− EmptyImage() — для заборони відправки форми якщо не завантажено ні одного зображення.

Компонент ClientProducts складається з наступних компонентів:

– ProductInfo — блок інформації про товар який користувач виставив, рис. 2.38;

– EditButton — кнопка для редагування товару, рис. 2.39;

– DeleteButton — кнопка для видалення товару, рис. 2.40.

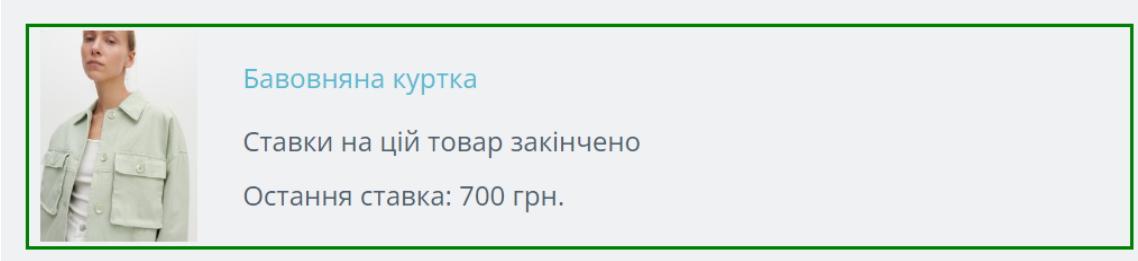

Рис. 2.38. Блок інформації про товар

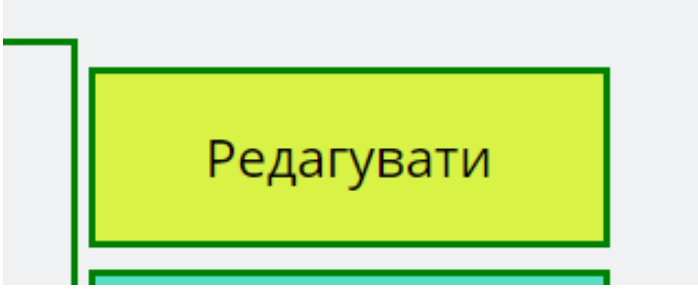

Рис. 2.39. Кнопка редагування товару

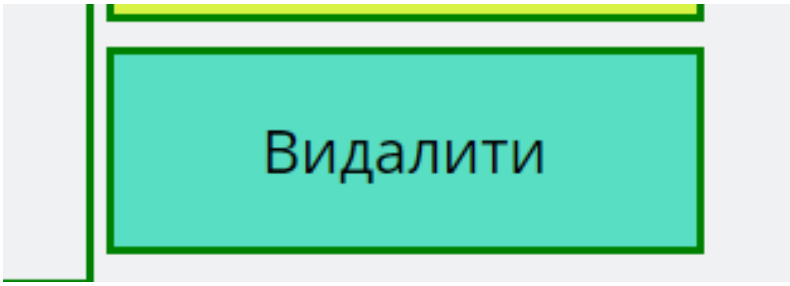

Рис. 2.40. Кнопка видалення товару

Для форми було назначено наступні обробники подій:

– GoEdit() — для переходу на сторінку редагування товару;

– GoDelete() — для видалення товару;

Вивід блоків позицій товару, редагування та видалення товару здійснено засобами Angular ngFor, routerLink, (click).

Компонент ClientWins складається з наступних компонентів:

– ProductInfo — блок інформації про товар який користувач виграв, аналогічне блоку форми ClientProducts;

– ToProduct — кнопка для переходу на сторінку товару, рис. 2.41.

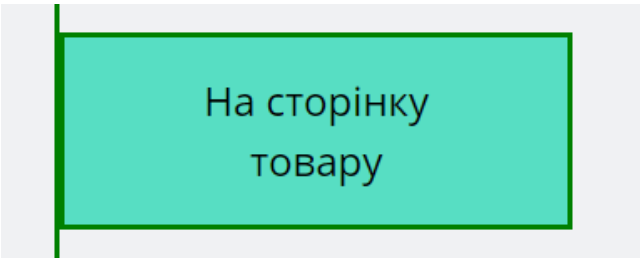

Рис. 2.41. Кнопка переходу на сторінку товару

Для форми було назначено наступні обробники подій:

– GoToProduct() — для переходу на сторінку товару;

Компонент ClientBets використовує той самий компонент що і ClientWins.

Компонент SearchProduct та CategoryProducts складається з наступних компонентів:

BlockOfFilters — блок фільтрації товарів за параметрами: категорія товару, срок ставок, сумма ставки, рис. 2.42;

ProductInfo — блок інформації про товар який відповідає пошуку та фільтрам, рис. 2.43.

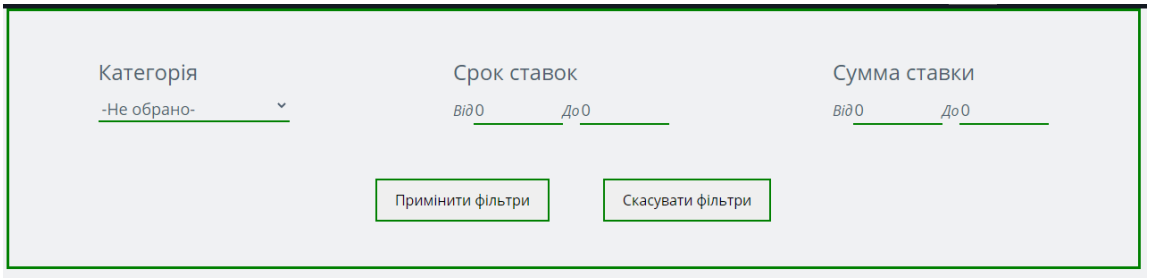

Рис. 2.42. Блок фільтрації товарів

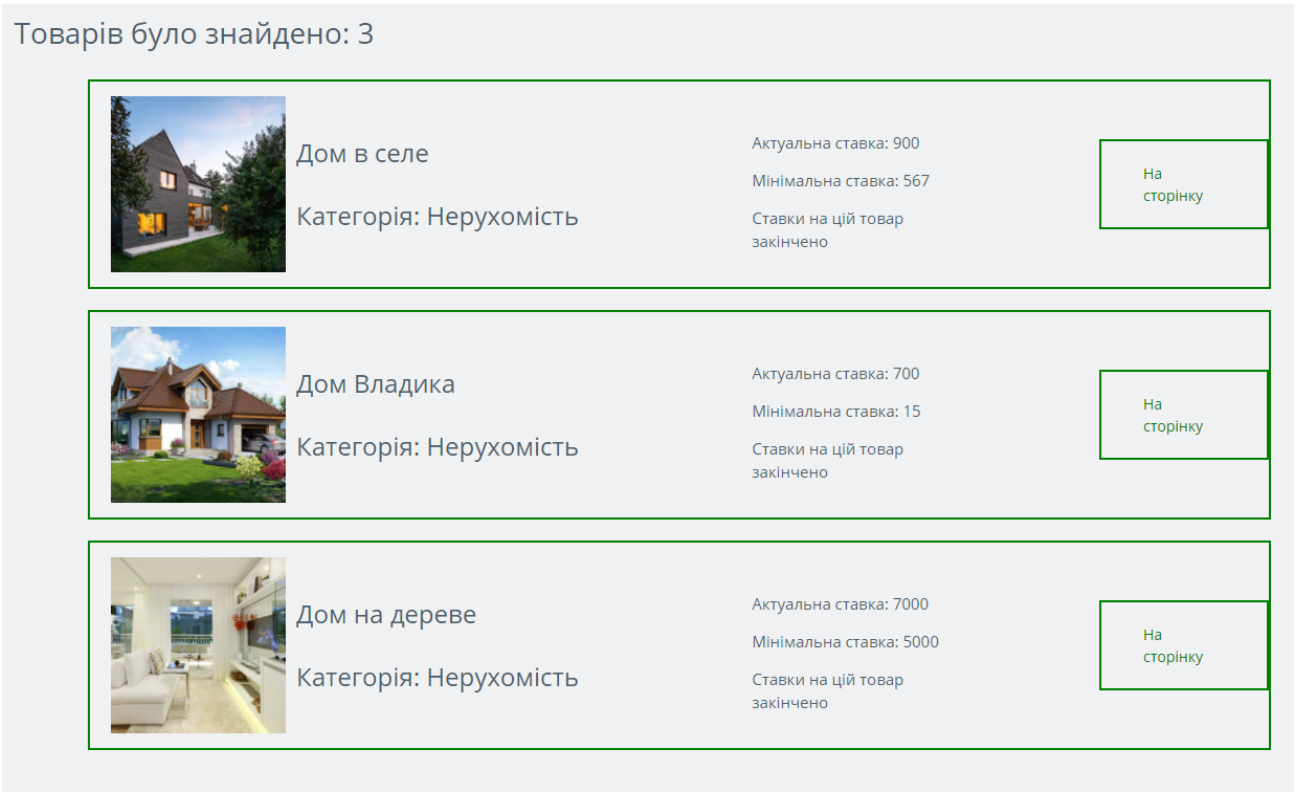

Рис. 2.43. Блок товарів за результатами пошуку

2.5.4 Схема взаємозв'язку модулів

Основні модулі програмної системи та схему їх взаємозв'язку представлено на рис. 2.44.

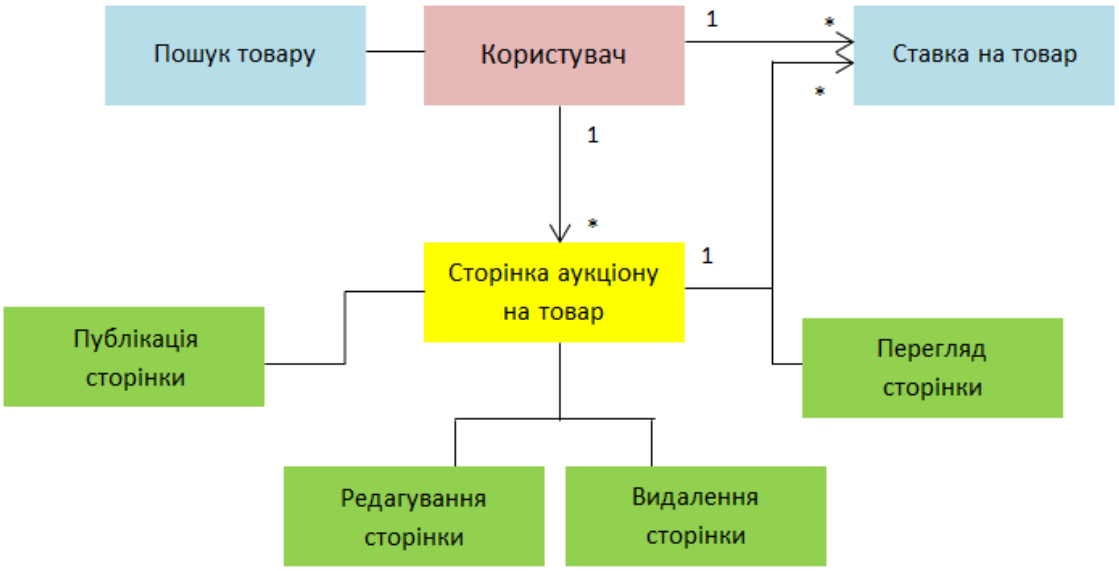

Рис. 2.44. Схема взаємозв'язку модулів

2.6. Обґрунтування та організація вхідних та вихідних даних програми

Web-додаток повинен задовольняти наступний функціонал:

− публікація оголошень про товар, що продається, редагування видалення ціх оголошень;

− можливість зробити ставку на цей товар;

Детальний опис функціонала:

1. Користувачі повинні мати можливість публікувати оголошення про річ, що продається. Один користувач може розмістити декілька оголошень.

2. Перегляд розміщених оголошень з пошуком за наступними параметрами: тип товару, ціна, назва товару, час продажу.

3. Результати пошуку повинні бути відсортовані за датою додавання або вартістю.

4. Якщо параметрів пошуку не обрано, на сторінці повинні відображатися десять перших оголошень, які мають найбільшу популярність.

5. Після закінчення торгів товар потрібен попадати у кабінет користувача який дав найбільшу ставку. У нього буде 2 тижні щоб сплатити ціну за цей товар.

Вхідні дані:

1. Товар:

- назва товару;
- тип товару;
- перша ставка;
- ім’я продавця;
- мінімальна ставка;
- опис товару;
- фото товару.
- дата додання товару.

2. Користувач:

- пароль;
- email;

– ім’я.

3. Категорія:

– назва категорії;

4. Фотографія:

– назва товару;

– фотографії.

5. Ставка:

– ім’я користувача який зробив ставку;

– назва товару;

– ставка.

2.7. Опис розробленого програмного продукту

2.7.1. Використані технічні засоби

Для розробки та функціонування веб-додатку для проведення онлайн-аукціонів необхідно використовувати відповідні технічні засоби як на стороні сервера, так і на стороні клієнта. У цьому підрозділі наведено основні вимоги до технічних засобів, необхідних для роботи програмного забезпечення.

Серверні засоби.

Основні забезпечення серверу для роботи програмного забезпечення включають наступні компоненти:

– Операційна система: Windows 10, яка забезпечує стабільність та безпеку серверного середовища.

– Підтримка протоколів: HTTP для обміну даними між сервером та клієнтом.

– Підтримувані бази даних: MySQL, що використовується для зберігання та управління даними аукціонів.

– Об'єм оперативної пам'яті: 4 ГБ, що забезпечує достатню продуктивність для обробки запитів.

– HTTP-сервер: Node 17 для обслуговування веб-запитів та забезпечення високої продуктивності.

Клієнтські засоби.

Для роботи програмного забезпечення на стороні клієнта необхідні наступні технічні засоби:

– Пристрій: Комп'ютер або мобільний пристрій, що забезпечує достатню потужність для запуску сучасних веб-додатків.

– Веб-браузер: Один з наступних браузерів або їх більш нові версії:

– Google Chrome;

– Opera;

– Mozilla Firefox;

– Internet Explorer;

– Інтернет-з'єднання: Персональний комп'ютер або інший пристрій повинен мати доступ до мережі Інтернет для стабільного обміну даними з сервером та забезпечення коректної роботи веб-додатку.

Використання цих технічних засобів забезпечує надійність, продуктивність і масштабованість веб-додатку для проведення онлайн-аукціонів. Дотримання зазначених вимог дозволяє створити стабільне середовище для роботи як серверної, так і клієнтської частин програмного забезпечення, що є ключовими факторами для успішної реалізації проєкту.

2.7.2. Використані програмні засоби

Для виконання проекту було обрано наступні мови програмування:

Клієнтська частина:

– Angular 16;

– HTML 5;

– CSS 4;

– Bootstrap 4.

Серверна частина:

– Php 6;

– Mysql 5;

Середовище розробки:

– WebStorm 2021;

2.7.3. Виклик та завантаження програми

Для завантаження програми потрібно зробити наступні дії:

1. Завантажити openserver 5 версії, та запустити його.
2. Скачати та запустити mysql database 5 версії.
3. Відкрити консоль розробника та перейти в директорію проекту, після цього в консоль написати команду ng serve.
4. Відкрити браузер та перейти на локальний сервер http://localhost:4200/

2.7.4. Опис інтерфейсу користувача

Переглядати головну сторінку сайту можуть як авторизовані, так і не авторизовані користувачі. Для неавторизованого користувача сторінка зі списком оголошень буде мати вигляд, рис. 2.45.

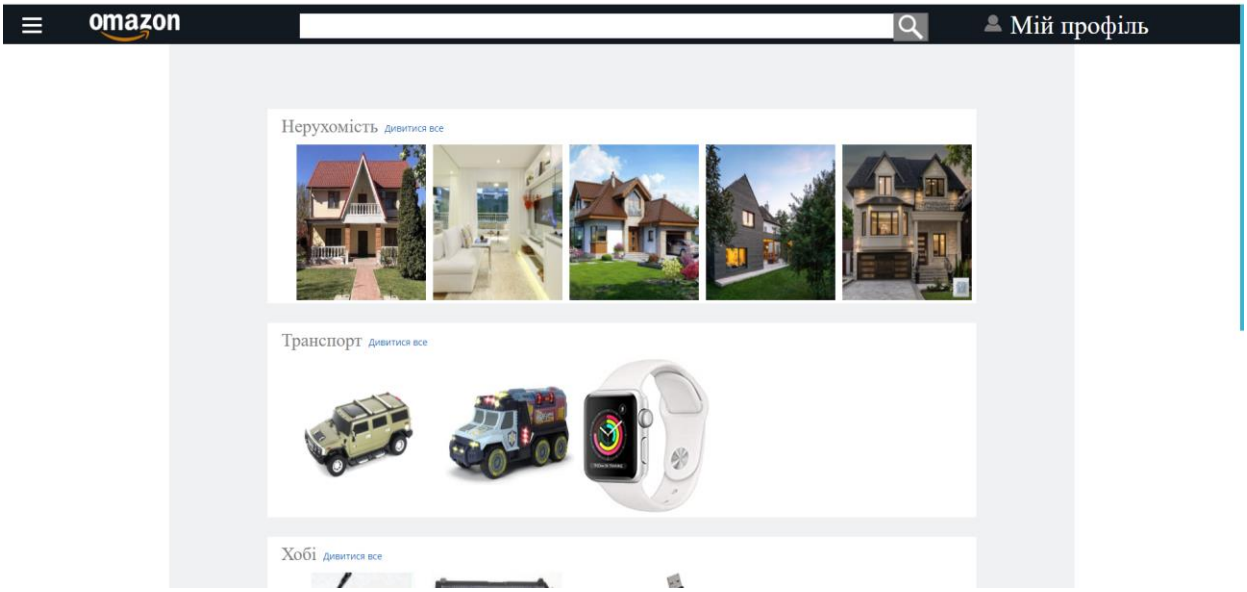

Рис. 2.45. Вигляд головної сторінки якщо користувач не авторизований

Якщо неавторизований користувач спробує зайти в свій кабінет, то його буде направлено на страніцу реєстрації, рис. 2.46.

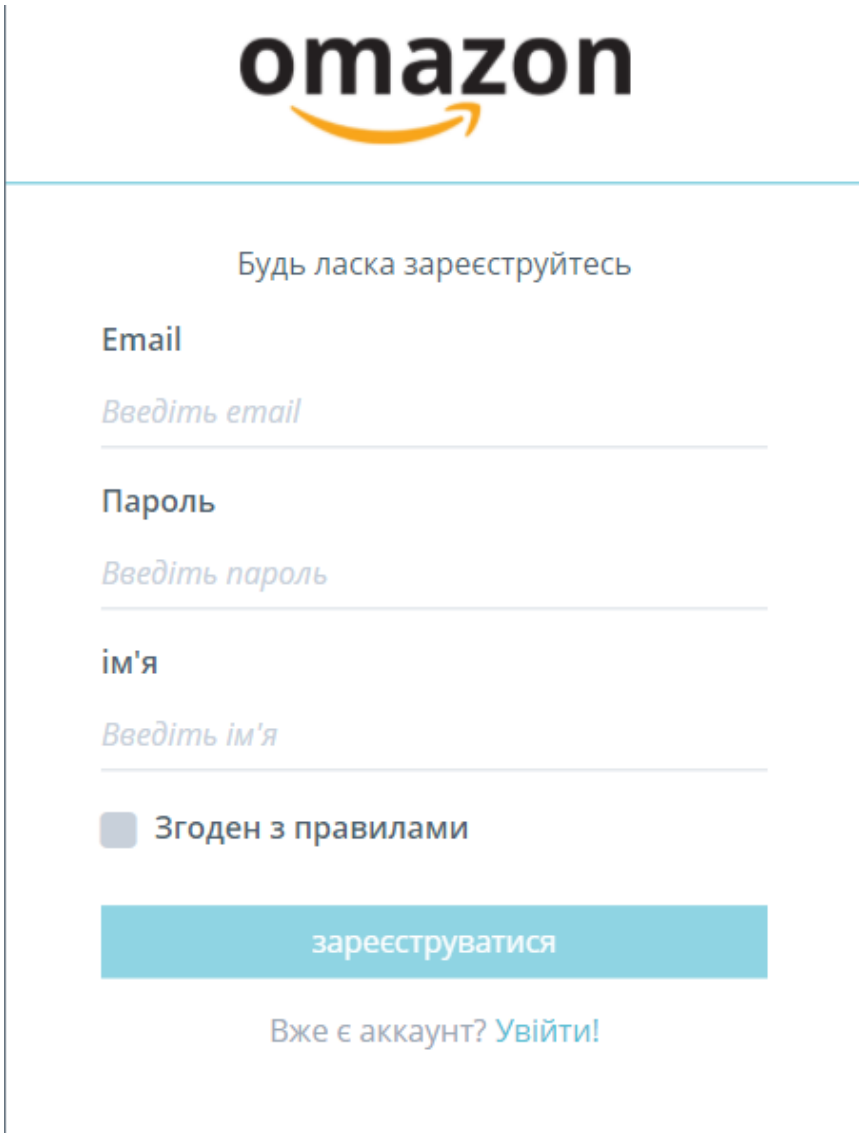

Рис. 2.46. Сторінка реєстрації

Якщо якесь з полів було введено некоректно або не було заповнено, буде відображено відповідні помилки, рис. 2.47.

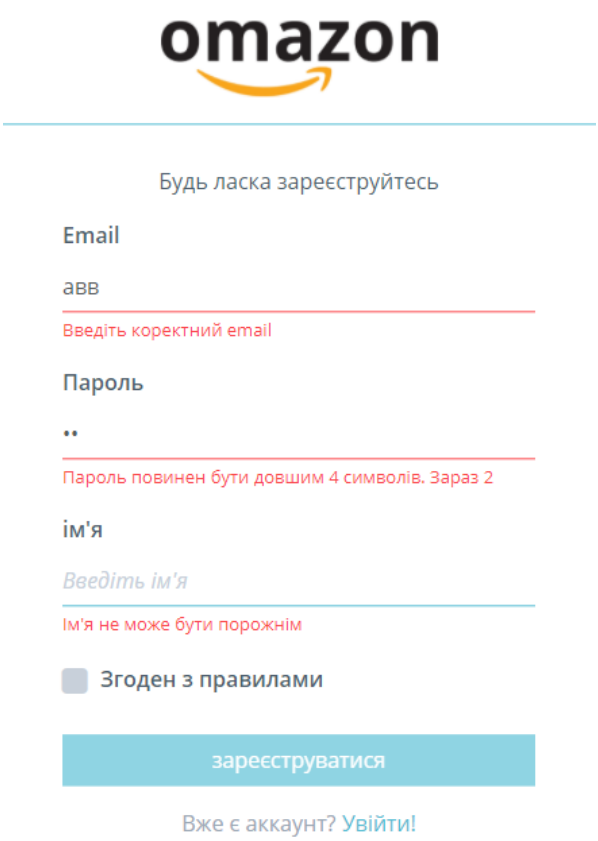

Рис. 2.47. Якщо поля email або пароль не відповідають шаблону

Після натискання кнопки «зареєструватися» користувача буде направлено на форму авторізації, рис. 2.48.

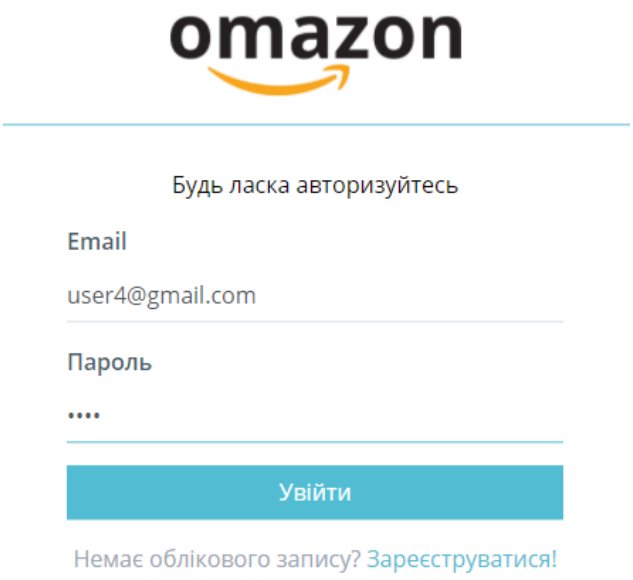

Рис. 2.48. Сторінка авторизації

Якщо при авторизації пароль або логін не співпадають з даними у базі даних, то буде виведено відповідне повідомлення, рис. 2.49.

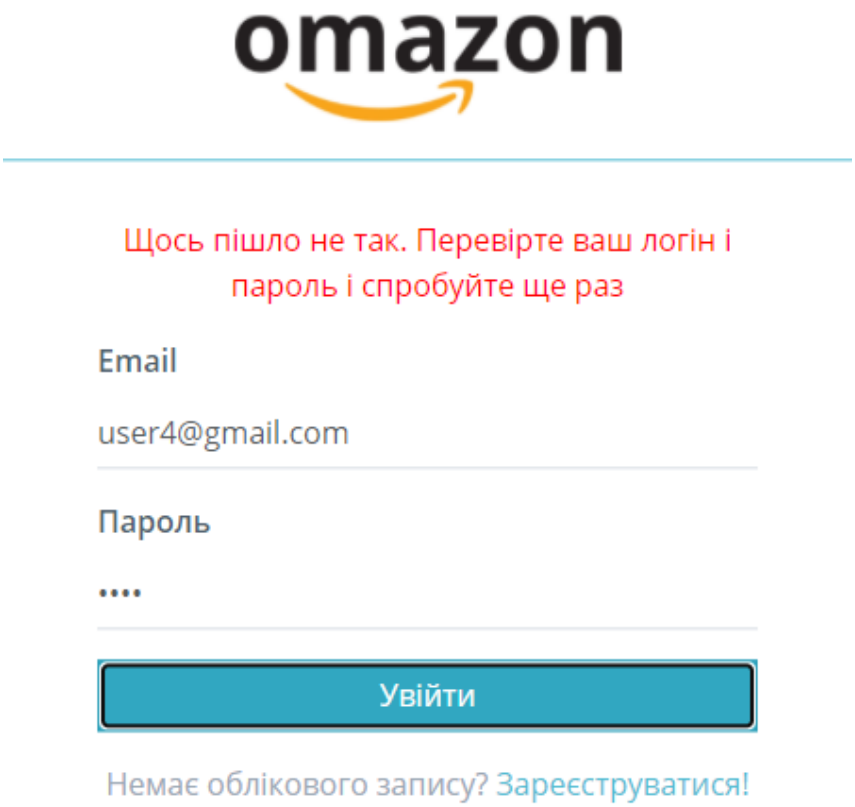

Рис. 2.49. Сторінка авторизації

Коли користувач буде авторизован головна сторінка зі списком оголошень буде мати вигляд, рис. 2.50.

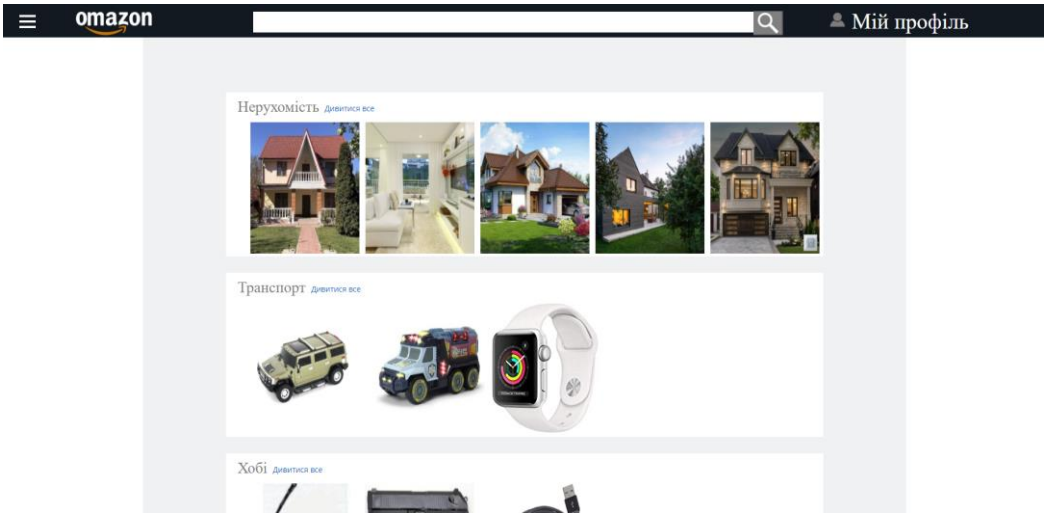

Рис. 2.50. Вигляд головної сторінки якщо користувач не авторизований

Авторизований користувач має можливість перейти у свій особистий кабінет, рис. 2.51.

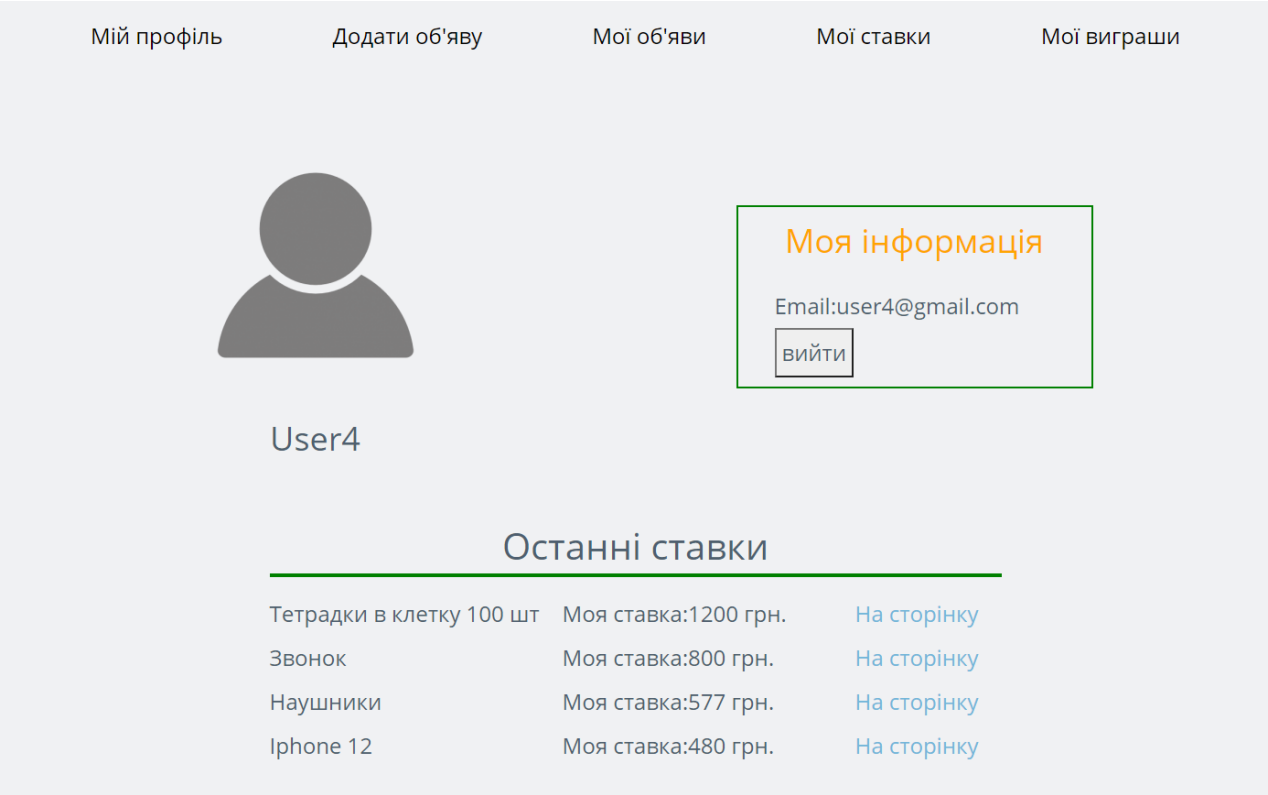

Рис. 2.51. Особистий кабінет користувача

У своєму кабінеті користувач має можливість натиснути кнопку «Додати об’яву», рис. 2.52.

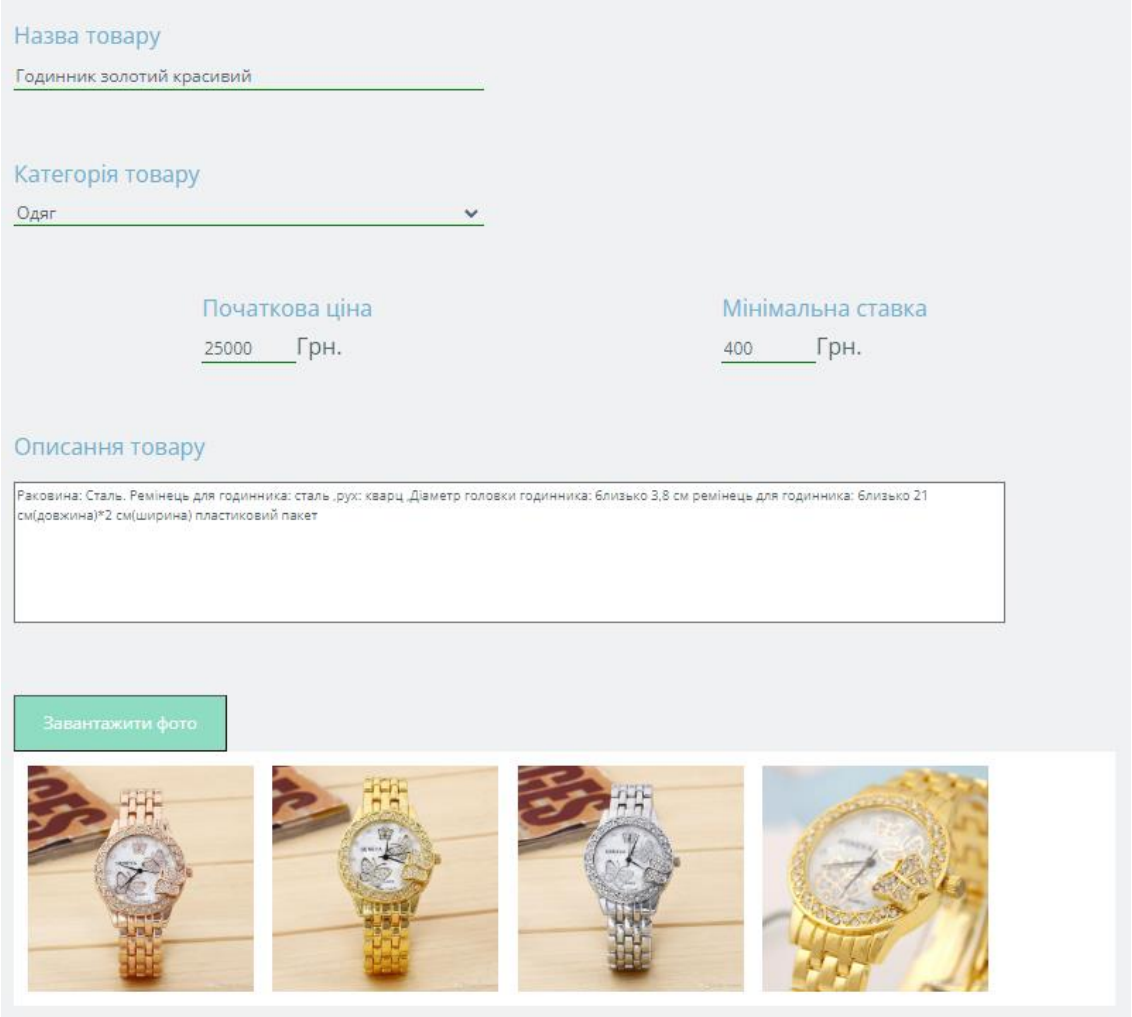

Рис. 2.52. Додавання користувачем нової об’яви про товар

Після додавання товару для нього генерується унікальний id, в особистому кабінеті натиснувши «Мої об’яви» можна побачити всі створені об’яви, рис. 2.53.

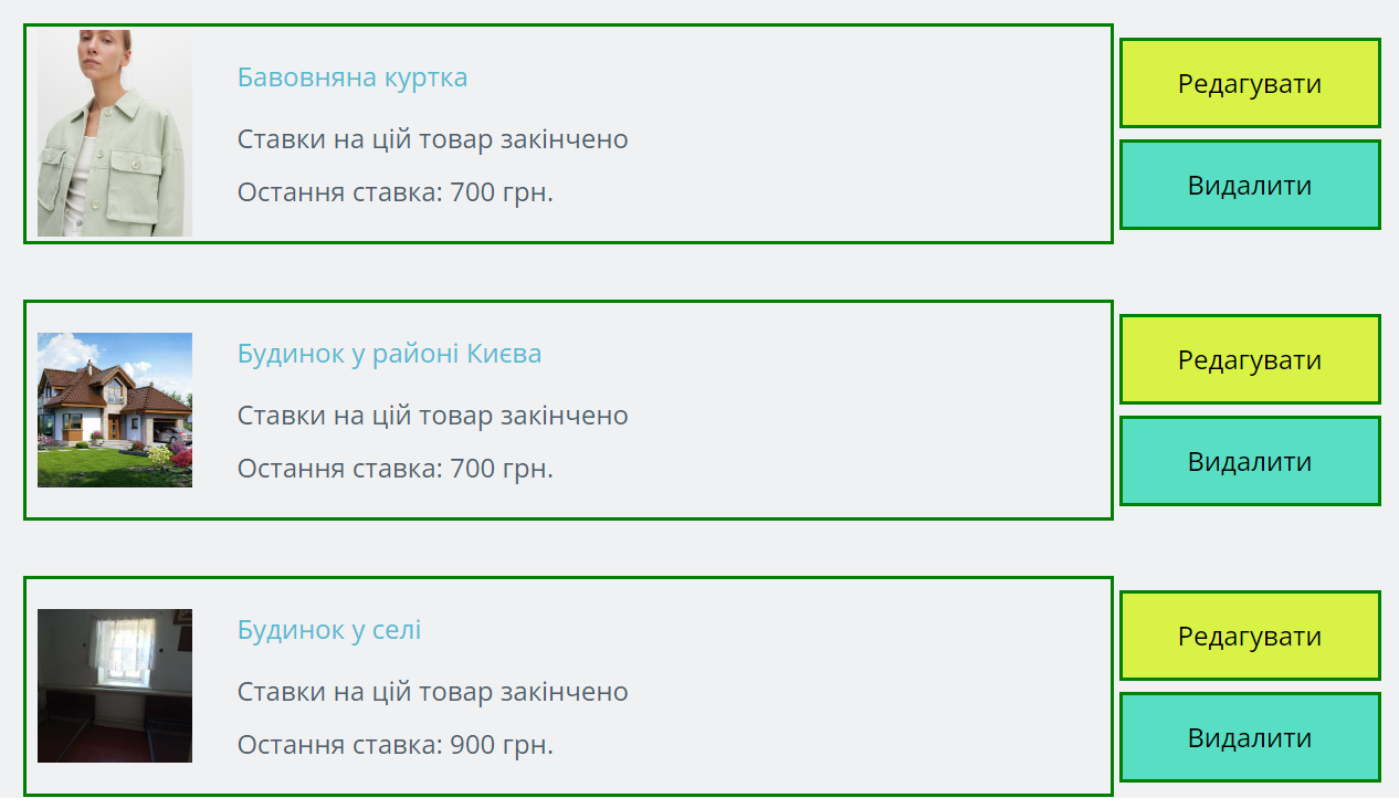

Рис. 2.53. Перегляд користувачем товарів які він створив

На сторінці створених товарів на сайті можна відредагувати створений товар натиснувши кнопку «Редагувати», рис. 2.54.

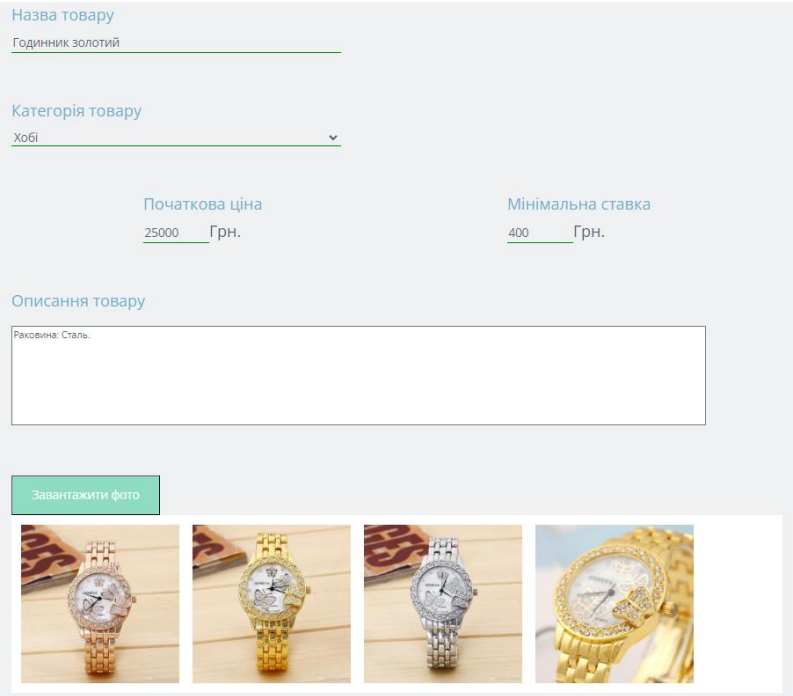

Рис. 2.54. Редагування створеної сторінки

Після редагування можно переглянути сторений товар перейшовши на сторінку цього товару, рис. 2.55.

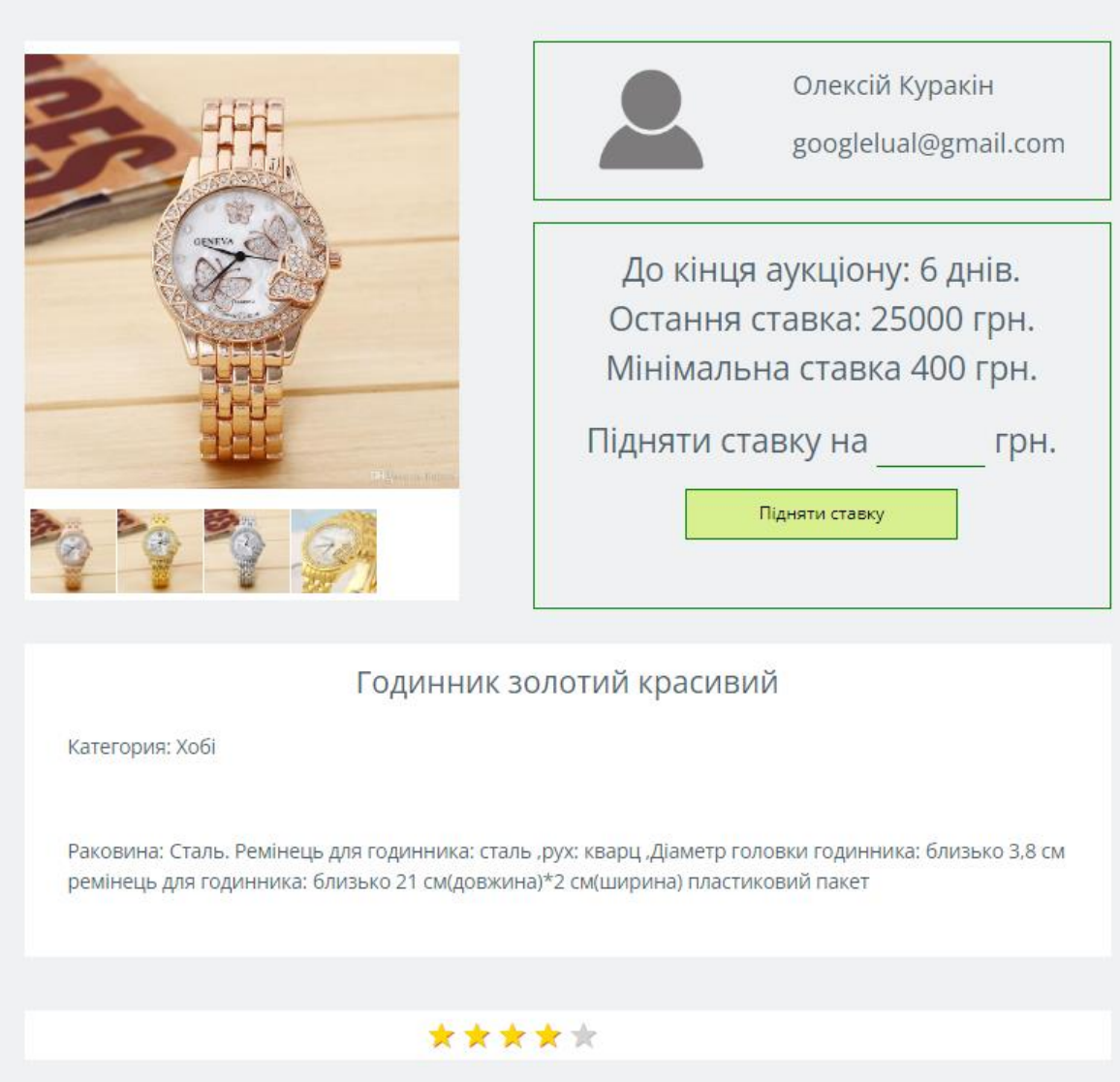

Рис. 2.55. Перегляд створенного оголошення

Після перегляду оголошення його можно видалити, рис. 2.56.

Рис. 2.56. Видалення оголошення

Для пошуку товарів потрібно ввести у поле пошуку потрібний текст та натиснути кнопку пошуку, рис. 2.57.

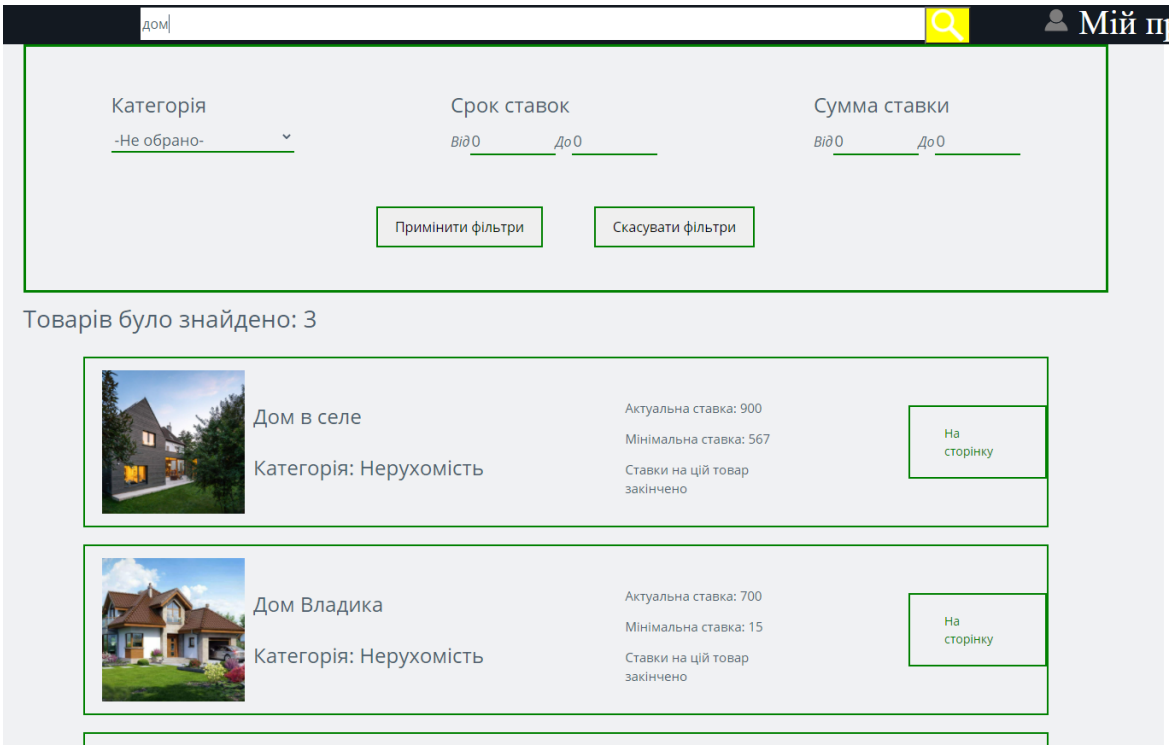

Рис. 2.57. Пошук оголошень

Можна відфільтрувати товари по категоріям, терміну ставок, суми ставки, рис. 2.58.

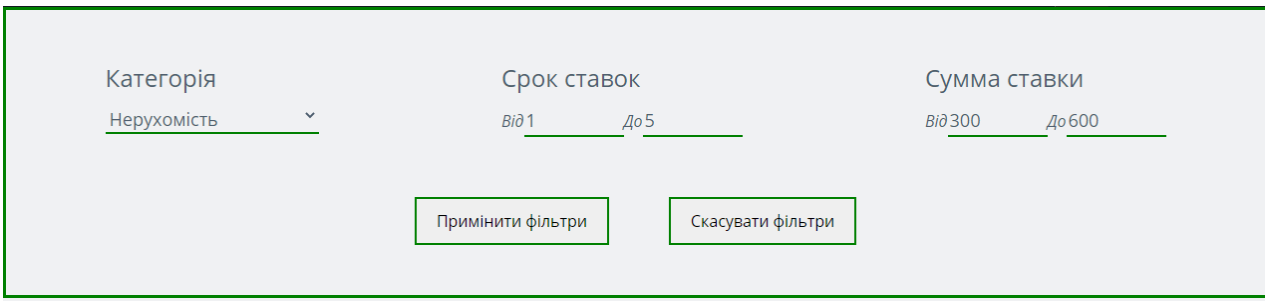

Рис. 2.58. Фільтрація товарів

Щоб скасувати виставлені фільтри потрібно натиснути кнопку «Скасувати фільтри», рис. 2.59.

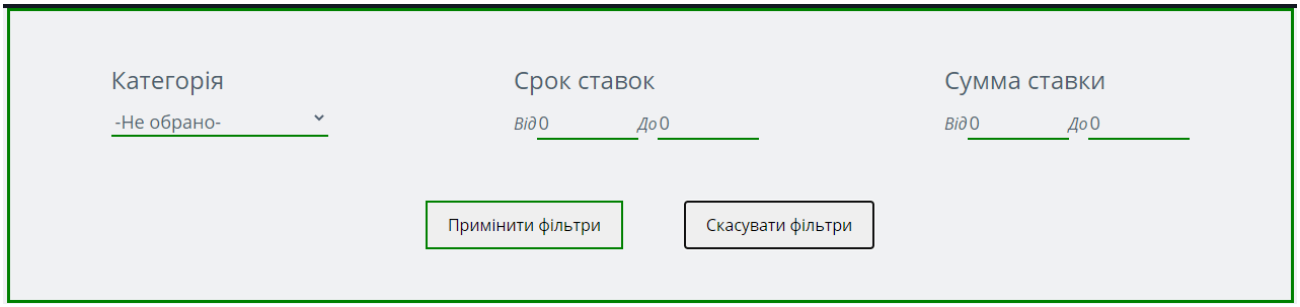

Рис. 2.59. Скасовані фільтри

Зробимо ставку на товар, перейшовши на сторінку товару введемо потрібну ставку, рис. 2.60.

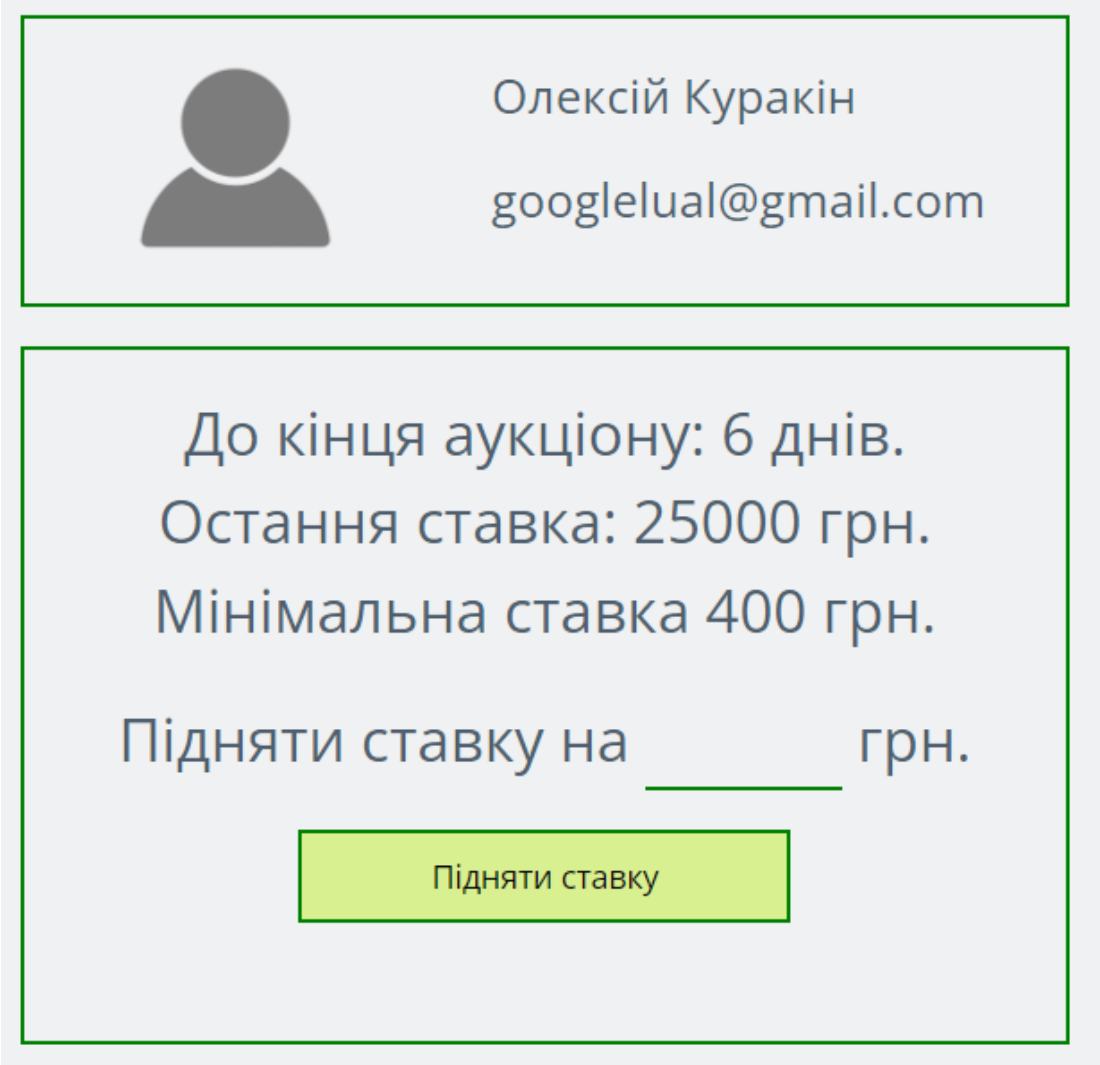

Рис. 2.60. Підняття ставки

Ставка потрібна бути введена коректно, якщо ставку була введена некоректно буде виведено відповідне повідомлення, рис. 2.61.

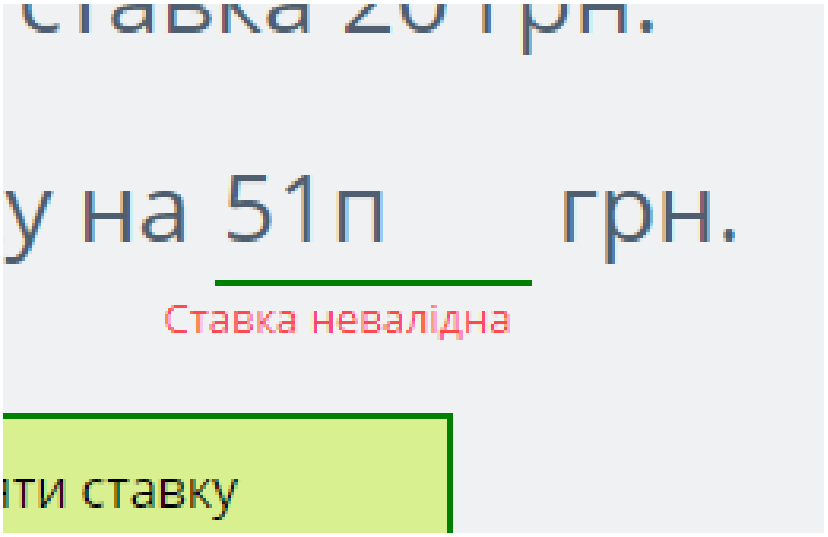

Рис. 2.61. Повідомлення про неможливість підняти ставку

Зроблені ставки можно подивитись у своєму кабінеті у розділі «Мої ставки», рис. 2.62.

Рис. 2.62. Розділ зроблених ставок

Якщо після закінчення ставок на товар ставка користувача виявилась найбільшою, то це можна побачити у своєму кабінеті у розділі «Мої виграші», рис. 2.63.

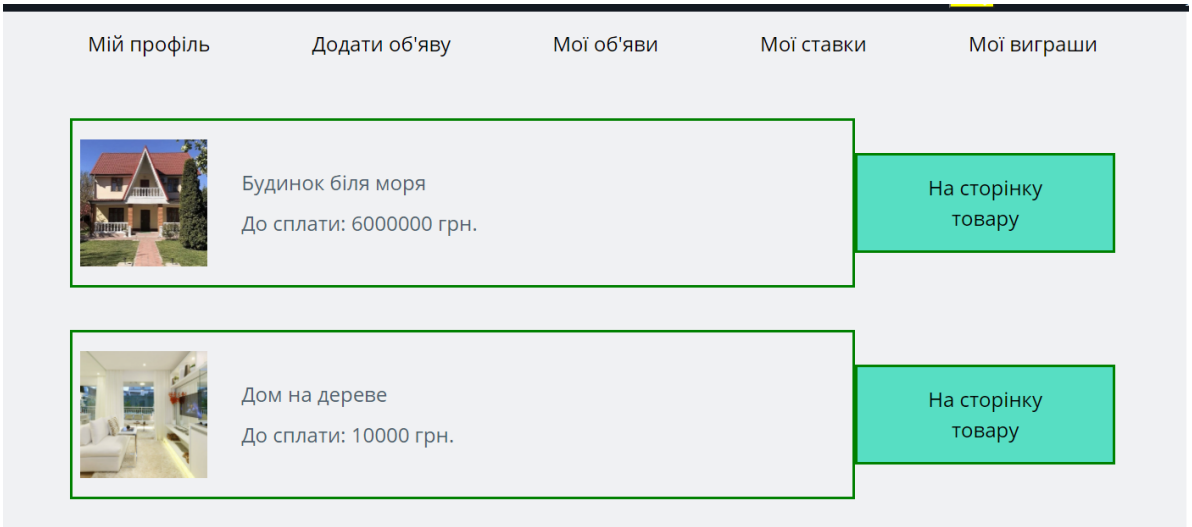

Рис. 2.63. Розділ виграних товарів

Якщо користувач хоче вийти зі свого облікового запису він повинен у своєму кабінеті натиснути кнопку «Вийти», після цього користувач буде направлений на головну сторінку, рис. 3.64.

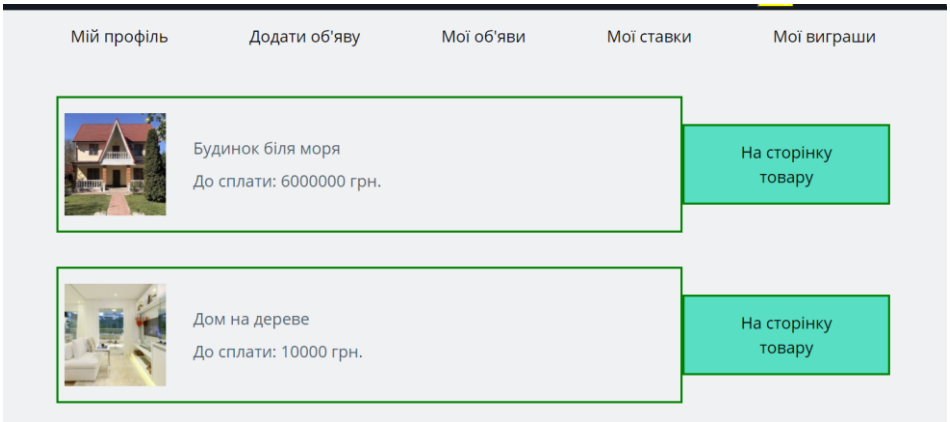

Рис. 2.64. Розділ виграних товарів

Повний програмний код web-додатку представлено у додатку Г.

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Початкові дані:

1. Передбачуване число операторів програми - 600;
2. Коефіцієнт складності програми (1,25...2,0) - 1,3;
3. Коефіцієнт корекції програми в ході її розробки (0,05...0,1) - 0,09;
4. Годинна заробітна плата програміста - 5000 грн/год;

Згідно зі статистикою платформи найму "Djinni" на середину 2024 року, середня зарплата Junior Angular розробника в Україні варіюється в діапазоні від 400$ до 1000$ на місяць. Виходячи з цих даних, можна розрахувати середню місячну заробітну плату, яка становить приблизно 700$ на місяць.

Ураховуючи офіційний курс валют Національного банку України на початок травня 2024 року, де один американський долар еквівалентний 38,90 гривням, середня місячна зарплата Junior Angular/Php розробника в Україні складає 27230 гривень. Якщо припустити стандартний робочий графік з 176 годинами на місяць, то зарплата за годину становитиме приблизно 154 гривнів.

5. Коефіцієнт збільшення витрат праці в наслідок недостатнього опису задачі (1,2...1,5) - 1,5;
6. Коефіцієнт кваліфікації програміста, обумовлений від стажу роботи зданої спеціальності (від 3-х до 5 років) – 1.3;
7. Вартість машино-години ЕОМ - 30 грн/год.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д \text{, людино-годин,} \qquad (3.1)$$

де t_o - витрати праці на підготовку й опис поставленої задачі (приймається50 людино-годин);

t_u - витрати праці на дослідження алгоритму рішення задачі;

t_a - витрати праці на розробку блок-схеми алгоритму;

t_n - витрати праці на програмування по готовій блок-схемі;

$t_{отл}$ - витрати праці на налагодження програми на ЕОМ;

$t_д$ - витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у програмному забезпеченні, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C \cdot (1 + p), \tag{3.2}$$

де q - передбачуване число операторів програми (q =600);

C - коефіцієнт складності програми (C =1,3);

p - коефіцієнт корекції програми в ході її розробки (p =0,09).

Звідси умовне число операторів в програмі:

$$Q = 600 \cdot 1{,}3 \cdot (1 + 0{,}09) = 850{,}2.$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \cdot B}{(75..85) \cdot k}, \text{ людино-годин}, \tag{3.3}$$

де B - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

k - коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності. При стажі роботи від 3 до 5 років він складає 1,1.

Приймемо збільшення витрат праці в наслідок недостатнього опису завдання не більше 50% (B = 1.5). З урахуванням коефіцієнта кваліфікації k = 1,3, отримуємо витрати праці на вивчення опису завдання:

$$t_u = (850{,}2 \cdot 1{,}5) / (80 \cdot 1{,}3) = 12{,}2625 \text{ людино-годин}.$$

Витрати праці на розробку алгоритму рішення задачі визначаються за формулою:

$$t_a = \frac{Q}{(20\ldots25)\cdot k}, \text{ людино-годин,} \qquad (3.4)$$

де Q – умовне число операторів програми;

k – коефіцієнт кваліфікації програміста.

Підставивши відповідні значення в формулу (3.4), отримаємо:

$$t_a = 850{,}2 / (21 \cdot 1{,}3) = 31{,}14 \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20\ldots25)\cdot k}, \text{ людино-годин,} \qquad (3.5)$$

$$t_n = 850{,}2 / (23 \cdot 1{,}3) = 28{,}43 \text{ людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

– за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4..5)\cdot k}, \text{ людино-годин,} \qquad (3.6)$$

$$t_{отл} = 850{,}2 / (4 \cdot 1{,}3) = 163{,}5 \text{ людино-годин.}$$

– за умови комплексного налагодження завдання:

$$t^k_{отл} = 1{,}5 \cdot t_{отл}, \text{ людино-годин,} \qquad (3.7)$$

$$t^k_{отл} = 1{,}5 \cdot 163{,}5 = 245{,}25 \text{ людино-годин.}$$

Витрати праці на підготовку документації визначаються за формулою:

$$t_д = t_{др} + t_{до}, \text{ людино-годин,} \qquad (3.8)$$

де $t_{др}$ - трудомісткість підготовки матеріалів і рукопису:

$$t_{др} = \frac{Q}{(15..20)\cdot k}, \text{ людино-годин,} \qquad (3.9)$$

$t_{до}$ - трудомісткість редагування, печатки й оформлення документації:

$$t_{до} = 0{,}75 \cdot t_{др}, \text{ людино-годин.} \qquad (3.10)$$

Підставляючи відповідні значення, отримаємо:

$t_{др}$ = 850,2 / (15 · 1,3) = 43,6 людино-годин.

$t_{до}$ = 0,75 · 43,6 = 32,7 людино-годин.

$t_{д}$= 43,6 + 32,7 = 76,3 людино-годин.

Повертаючись до формули (3.1), отримаємо повну оцінку трудомісткості розробки програмного забезпечення:

t = 50 + 12,2625 + 31,14 + 28,43 + 163,5 + 76,3 = 361,63 людино-годин.

3.2. Рахунок витрат на створення програми

Витрати на створення ПЗ $К_{ПО}$ включають витрати на заробітну плату виконавця програми $З_{ЗП}$ і витрати машинного часу, необхідного на налагодження програми на ЕОМ:

$$К_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн..} \quad (3.11)$$

Заробітна плата виконавців визначається за формулою:

$$З_{ЗП} = t \cdot С_{ПР}, \text{ грн,}$$

де: t - загальна трудомісткість, людино-годин;

$С_{ПР}$ - середня годинна заробітна плата програміста, грн/година

З урахуванням того, що середня годинна зарплата програміста становить 131 грн / год, отримуємо:

$З_{ЗП}$ = 361,63 · 154 = 55 691,02 грн.

Вартість машинного часу, необхідного для налагодження програми на ЕОМ, визначається за формулою:

$$З_{мв} = t_{отл} \cdot С_{мч}, \text{ грн,} \quad (3.12)$$

де $t_{отл}$ - трудомісткість налагодження програми на ЕОМ, год;

$С_{мч}$ - вартість машино-години ЕОМ, грн/год (14 грн/год).

Підставивши в формулу (3.12) відповідні значення, визначимо вартість необхідного для налагодження машинного часу:

$З_{мв}$= 163,5 · 30 = 4 905 грн.

Звідси витрати на створення програмного продукту:

$$K_{ПО} = 55\ 691{,}02 + 4\ 905 = 60\ 596{,}02 \text{ грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p}, \text{ міс.} \qquad (3.13)$$

де B_k - число виконавців (дорівнює 1);

F_p - місячний фонд робочого часу (при 40 годинному робочому тижні

F_p=176 годин).

Звідси витрати на створення програмного продукту:

$$T = 628{,}59 / 1 \cdot 176 = 3{,}57 \text{ міс.}$$

Висновок

Програмне забезпечення призначене для реалізації інформаційної системи з метою онлайн аукціону. Вартість даного програмного забезпечення становить 60 596,02 грн і включає витрати на заробітну плату виконавця та машинний час, необхідний для налагодження програми. Очікуваний час розробки складає приблизно 4 місяця. Цей термін обумовлений значним числом операторів і включає час на дослідження та розробку алгоритму вирішення поставленого завдання, програмування за готовим алгоритмом, налагодження програми і підготовку документації.

ВИСНОВКИ

Метою даної кваліфікаційної роботи є розробка веб-додатку для проведення онлайн-аукціонів з продажу товарів, який використовуватиме сучасні технології Angular для клієнтської частини та PHP для серверної частини. Використання цих технологій дозволяє створити потужний, ефективний та зручний у користуванні додаток, що відповідає всім сучасним вимогам до веб-додатків.

Актуальність теми обумовлена зростаючим попитом на онлайн-торгівлю та необхідністю створення високоякісних платформ, які забезпечують надійність, безпеку та зручність у користуванні. Онлайн-аукціони стали важливим інструментом для малого та середнього бізнесу, дозволяючи їм досягати ширшої аудиторії та збільшувати обсяги продажів. Для покупців такі платформи надають можливість знайти унікальні товари за вигідними цінами.

Основними завданнями даної кваліфікаційної роботи є аналіз потреб користувачів, розробка архітектури системи, створення функціональних модулів для управління аукціонами, забезпечення безпеки та захисту даних. Важливим аспектом є також забезпечення сумісності з різними операційними системами та веб-браузерами, що дозволить забезпечити максимальне охоплення користувачів.

У ході розробки буде використано передові методи та інструменти, що дозволяють забезпечити високу продуктивність та надійність системи. Angular, як сучасний фреймворк для створення односторінкових додатків, забезпечить швидкість та інтерактивність клієнтської частини. PHP, у свою чергу, використовується для створення серверної частини, що відповідає за обробку запитів, управління базами даних та забезпечення логіки додатку.

Таким чином, дана квалифікаційна робота спрямована на створення інноваційного веб-додатку для проведення онлайн-аукціонів, який

відповідатиме всім сучасним вимогам та стане надійним інструментом для користувачів з різних куточків світу.

Розробка веб-додатку для проведення онлайн-аукціонів досягла своєї мети та вирішила поставлені задачі, підтверджуючи актуальність та доцільність вибору теми дослідження. Отримані результати можуть бути використані для подальших досліджень та вдосконалення існуючих систем електронної комерції.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. WebStorm Product [Електроний ресурс] / інтелектуальна Angular IDE. – Режим доступу до ресурсу : https://jetbrains/products/webstorm/.
2. Angular [Електронний ресурс] // Вікіпедія. Вільна енциклопедія – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Angular_(web_framework)
3. Angular Documentation [Електронний ресурс] // Офіційний сайт – Режим доступу до ресурсу: https://angular.io/docs
4. PHP [Електронний ресурс] // Вікіпедія. Вільна енциклопедія – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/PHP.
5. PHP Documentation [Електронний ресурс] // Офіційний сайт – Режим доступу до ресурсу: https://www.php.net/docs.php
6. MySQL [Електронний ресурс] // Вікіпедія. Вільна енциклопедія – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/MySQL
7. MySQL Documentation [Електронний ресурс] // Офіційний сайт – Режим доступу до ресурсу: https://dev.mysql.com/doc/
8. Nginx [Електронний ресурс] // Вікіпедія. Вільна енциклопедія – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Nginx
9. Nginx Documentation [Електронний ресурс] // Офіційний сайт – Режим доступу до ресурсу: https://nginx.org/en/docs/
10. Gunicorn [Електронний ресурс] // Вікіпедія. Вільна енциклопедія – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Gunicorn
11. Gunicorn Documentation [Електронний ресурс] // Офіційний сайт – Режим доступу до ресурсу: https://docs.gunicorn.org/en/stable/
12. Ubuntu Linux [Електронний ресурс] // Вікіпедія. Вільна енциклопедія – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Ubuntu

13. Ubuntu Linux Documentation [Електронний ресурс] // Офіційний сайт – Режим доступу до ресурсу: https://ubuntu.com/server/docs

14. HTTP [Електронний ресурс] // Вікіпедія. Вільна енциклопедія – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/HTTP

15. HTTP/1.1 [Електронний ресурс] // Офіційна документація – Режим доступу до ресурсу: https://tools.ietf.org/html/rfc2616

16. Google Chrome [Електронний ресурс] // Вікіпедія. Вільна енциклопедія – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Google_Chrome

17. Opera [Електронний ресурс] // Вікіпедія. Вільна енциклопедія – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Opera

18. Mozilla Firefox [Електронний ресурс] // Вікіпедія. Вільна енциклопедія – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Firefox

19. Internet Explorer [Електронний ресурс] // Вікіпедія. Вільна енциклопедія – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Internet_Explorer

ДОДАТОК А

ПОВНА ER-ДІАГРАМА БАЗИ ДАНИХ

Повна ER-діаграма бази даних web-додатку Omazon представлена на рис. А.1

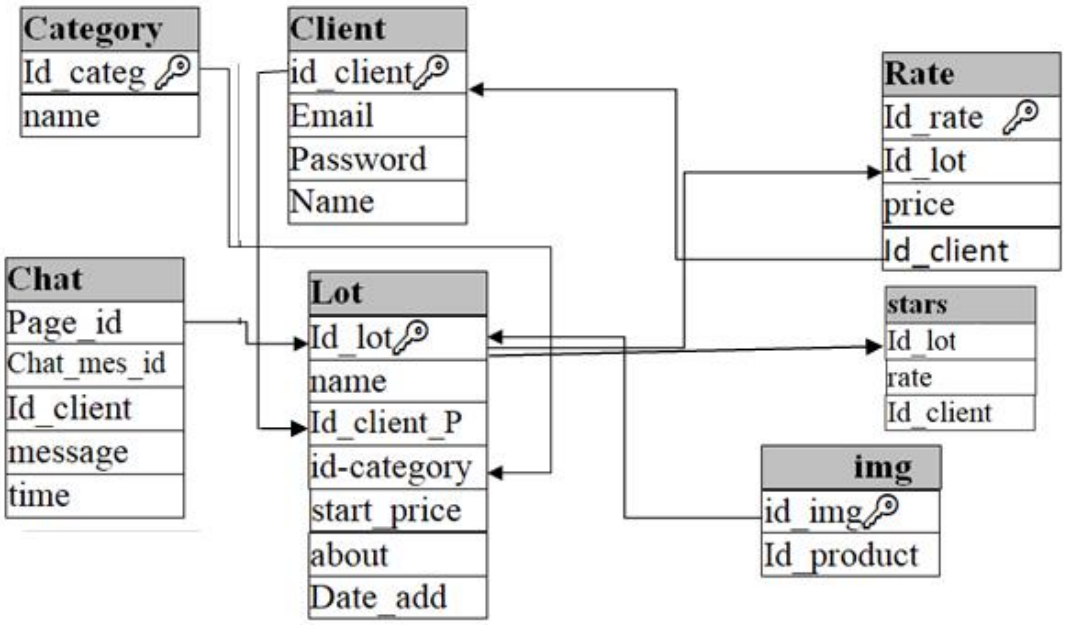

Рис. А.1. Повна ER-діаграма бази даних

ДОДАТОК Б

ТАБЛИЦІ БАЗИ ДАНИХ

Структури таблиць бази даних «OmazonApp» зображені на рисунках Б.1-Б.7.

Поле id в таблицях передбачено в якості первинного ключа.

#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Комментарии	Дополнительно	Действие
1	id_client	mediumint(25)		UNSIGNED	Нет	Нем			
2	Email	varchar(40)	utf8mb4_unicode_ci		Нет	Нем			
3	password	varchar(16)	utf8mb4_unicode_ci		Нет	Нем			
4	Name	varchar(255)	utf8mb4_unicode_ci		Нет	Нем			

Рис. Б.1. Структура таблиці «Користувач»

#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Комментарии	Дополнительно	Действие
1	id_tovar	int(11)			Нет	Нем			
2	id_client_prod	int(11)			Нет	Нем			
3	id_category	int(11)			Нет	Нем			
4	name	varchar(250)	utf8mb4_unicode_ci		Нет	Нем			
5	min_rate	float			Нет	Нем			
6	about	text	utf8mb4_unicode_ci		Нет	Нем			
7	date_add	datetime			Да	current_timestamp()		ON UPDATE CURRENT_TIMESTAMP()	

Рис. Б.2. Структура таблиці «Товар»

#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Комментарии	Дополнительно	Действие
1	id_category	int(1)			Нет	Нем			
2	name	varchar(12)	utf8mb4_unicode_ci		Нет	Нем			

Рис. Б.3. Структура таблиці «Категорія»

Структура таблицы Связи

#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Комментарии	Дополнительно	Действие
1	id_stavka	int(11)			Нет	Нем			
2	id_tovar	int(11)			Нет	Нем			
3	id_client	int(11)			Нет	Нем			
4	samaStavka	int(11)			Нет	Нем			

Рис. Б.4. Структура таблиці «Ставка»

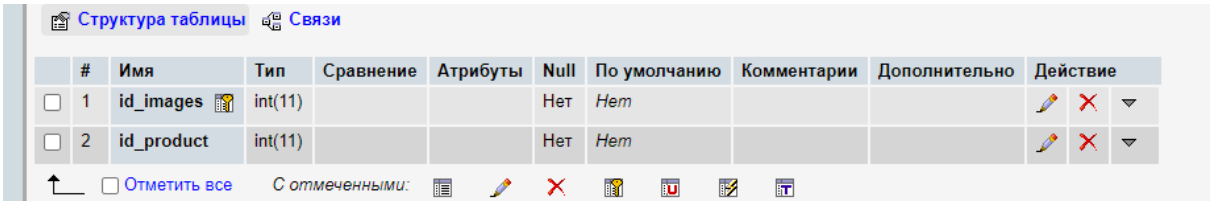

Рис. Б.5. Структура таблиці «Зображення»

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файла	**Опис**
Пояснювальні документи	
Диплом_ Куракін.doc	Пояснювальна записка до дипломного проекту. Документ Word.
Диплом_ Куракін.pdf	Пояснювальна записка до дипломного проекту в форматі PDF
Програма	
Program.rar	Архів. Містить коди програми і откомпільовану програму
Презентація	
Презентація_ Куракін.pptx	Презентація дипломного проекту

ДОДАТОК Г

КОД ПРОГРАМИ

Фрагменти програмного коду наведено в лістингах Г.1-Г.50.

Лістинг Г.1 — Текст програмного компоненту для реєстрації користувача

```
import {Component, OnInit} from '@angular/core';
import {FormControl, FormGroup, Validators} from '@angular/forms';
import {RegistrationService} from './registration.service';
import {User} from '../../shared/models/user.model';
import {Router} from '@angular/router';

@Component({
  selector: 'app-registration',
  templateUrl: './registration.component.html',
  styleUrls: ['./registration.component.css']
})
export class RegistrationComponent implements OnInit {
  NewEmail = false;
  form: FormGroup;

  constructor(private registrationService: RegistrationService,
              private router: Router) {
  }

  ngOnInit(): void {
    this.form = new FormGroup({
      'email': new FormControl(null, [Validators.required, Validators.email]),
      'password': new FormControl(null, [Validators.required,
Validators.minLength(4), Validators.maxLength(12)]),
      'name': new FormControl(null, [Validators.required]),
      'agree': new FormControl(false, [Validators.required,
Validators.requiredTrue])
    });
  }

  onSubmit() {
    const {email, password, name} = this.form.value;
    const user = new User(email, password, name);
    // tslint:disable-next-line:no-shadowed-variable
    this.registrationService.createNewUser(user).subscribe((user: number) => {
      // tslint:disable-next-line:triple-equals
      if (user == 0) {
        this.NewEmail = false;
        this.router.navigate(['./login'], {
          queryParams: {
            NowCanLogin: this.form.value.name
          }
        });
      } else {
        this.NewEmail = true;
      }
    });}}
```

Лістинг Г.2 — Текст шаблону компоненту для реєстрації користувача

```
<p class="text-xs-center">Будь ласка зареєструйтесь</p>
<form [formGroup]="form" (ngSubmit)="onSubmit()">
  <div class="form-group"
       [ngClass]="{'has-error': form.get('email').invalid &&
form.get('email').touched}"
```

```
  >
    <label for="email">Email</label>
    <input
      type="text"
      class="form-control underlined"
      id="email"
      placeholder="Введіть email"
      formControlName="email"
    >
    <span *ngIf="NewEmail; else succesStatus" style="color: red">Даний email вже
використовується</span>
    <ng-template #succesStatus>
      <span class="form-help-text"
            *ngIf="form.get('email').invalid && form.get('email').touched"
      >
        <span *ngIf="form.get('email')['errors']['required']">email не може бути
порожнім</span>
        <span *ngIf="form.get('email')['errors']['email']">Введіть коректний
email</span>

      </span>
    </ng-template>
  </div>
  <div class="form-group"
       [ngClass]="{'has-error': form.get('password').invalid &&
form.get('password').touched}"
  >
    <label for="password">Пароль</label>
    <input
      type="password"
      class="form-control underlined"
      id="password"
      placeholder="Введіть пароль"
      formControlName="password"
    >
    <span class="form-help-text"
          *ngIf="form.get('password').invalid && form.get('password').touched"
    >
        <span *ngIf="form.get('password')['errors']['required']">Пароль не може
бути порожнім</span>
        <span *ngIf="form.get('password')['errors']['minlength']">
          Пароль повинен бути довшим
          {{form.get('password')['errors']['minlength']['requiredLength']}}
символіГ.
          Зараз
{{form.get('password')['errors']['minlength']['actualLength']}}</span>
         <span *ngIf="form.get('password')['errors']['maxlength']">
         Пароль повинен бути коротше
           {{form.get('password')['errors']['maxlength']['requiredLength']}}
символіГ.
          Зараз
{{form.get('password')['errors']['maxlength']['actualLength']}}</span>
      </span>
  </div>
  <div class="form-group"
       [ngClass]="{'has-error': form.get('name').invalid &&
form.get('name').touched}"
  >
    <label for="name">ім'я</label>
    <input
      type="text"
      class="form-control underlined"
      id="name"
      placeholder="Введіть ім'я"
      formControlName="name"
    >
    <span class="form-help-text"
```

```
          *ngIf="form.get('name').invalid && form.get('name').touched"
    >
              Ім'я не може бути порожнім
      </span>
  </div>
  <div
    class="form-group"
    [ngClass]="{'has-error': form.get('agree').invalid &&
form.get('agree').touched}"
  >
    <label for="agree">
      <input class="checkbox"
             id="agree"
             type="checkbox"
             formControlName="agree"
      >
      <span>Згоден з правилами</span>
    </label>
  </div>
  <div class="form-group">
    <button
      type="submit"
      class="btn btn-block btn-primary"
      [disabled]="form.invalid"
    >
      зареєструватися
    </button>
  </div>
  <div class="form-group">
    <p class="text-muted text-xs-center">
      Вже є аккаунт?
      <a [routerLink]="'/login'">
        Увійти!
      </a>
    </p>
  </div>
</form>
```

Лістинг Г.3 — Текст сервису компоненту для реєстрації користувача

```
import {Injectable} from '@angular/core';
import {HttpClient} from '@angular/common/http';
import {Observable} from 'rxjs';
import {User} from '../../shared/models/user.model';
import {map} from 'rxjs/operators';

@Injectable({
  providedIn: 'root'
})
export class RegistrationService {
  bul = 0;

  constructor(private http: HttpClient) {
  }

  createNewUser(user: User): Observable<any> {
    console.log(user);
    const formData = new FormData();
    Object.keys(user).forEach(key => formData.append(key, user[key]));
    return this.http.post<any>(`http://kyrsovoi/registration.php`, formData).pipe(
      map((res) => {
        this.bul = res['data'];
        return this.bul;
      }));
  }
}
```

Лістинг Г.4 — Текст програмного компоненту для авторізації користувача

```
import {Component, OnInit} from '@angular/core';
import {FormControl, FormGroup, Validators} from '@angular/forms';
import {User} from '../../shared/models/user.model';
import {LoginService} from './login.service';
import {ActivatedRoute, Router} from '@angular/router';
import {AuthService} from '../../auth.service';
import {CookieService} from 'ngx-cookie-service';
import {Login} from './login';

@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css']
})
export class LoginComponent implements OnInit {
  loginName = '';
  message = 'Будь ласка авторизуйтесь';
  form: FormGroup;
  messageColor = 'black';

  constructor(
    private loginService: LoginService,
    private router: Router,
    private route: ActivatedRoute,
    private auth: AuthService,
    private cookieService: CookieService
  ) {
  }

  ngOnInit(): void {
    this.route.queryParams.subscribe((params) => {
      if (params['NowCanLogin']) {
        this.loginName = params['NowCanLogin'];
      }
      if (this.loginName != '') {
        this.message = this.loginName + ' будь ласка авторизуйтесь';
      }
    });
    this.form = new FormGroup({
      email: new FormControl(null, [Validators.required, Validators.email]),
      password: new FormControl(null, [Validators.required,
Validators.minLength(4)])
    });
  }

  onSubmit() {
    const {email, password} = this.form.value;
    const user = new User(email, password, 'AAA');

    // tslint:disable-next-line:no-shadowed-variable
    this.loginService.getUser(user).subscribe((baf: Login) => {
      if (baf.match != 0) {
        this.changeAuthStatus('login');
        console.log(baf);
        this.cookieService.set('user', String(1));
        this.cookieService.set('id', String(baf.id_client));
        this.router.navigate(['./system/main'], {
          queryParams: {
            nowCanLoggin: true
          }
        });
      } else {
        this.messageColor = 'red';
        this.message = 'Щось пішло не так. Перевірте ваш логін і пароль і
спробуйте ще раз';
```

```
      }
    });
  }

  changeAuthStatus(status: string) {
    if (status === 'login') {
      this.auth.logIn();
    } else {
      this.auth.logOut();
    }
  }
}
```

Лістинг Г.5 — Текст програмного сервісу для авторізації користувача

```
import {Injectable} from '@angular/core';
import {HttpClient} from '@angular/common/http';
import {Observable} from 'rxjs';
import {User} from '../../shared/models/user.model';
import {map} from 'rxjs/operators';

@Injectable({
  providedIn: 'root'
})
export class LoginService {
  bul = 0;

  constructor(private http: HttpClient) {
  }

  getUser(user: User): Observable<any> {
    const formData = new FormData();
    Object.keys(user).forEach(key => formData.append(key, user[key]));
    return this.http.post<any>(`http://kyrsovoi/get_user.php`, formData).pipe(
      map((res) => {
        this.bul = res['data'];
        return this.bul;
      }));
  }
}
```

Лістинг Г.6 — Текст програмного класу для авторізації користувача

```
import {Injectable} from '@angular/core';
import {HttpClient} from '@angular/common/http';
import {Observable} from 'rxjs';
import {User} from '../../shared/models/user.model';
import {map} from 'rxjs/operators';

@Injectable({
  providedIn: 'root'
})
export class LoginService {
  bul = 0;

  constructor(private http: HttpClient) {
  }

  getUser(user: User): Observable<any> {
    const formData = new FormData();
    Object.keys(user).forEach(key => formData.append(key, user[key]));
    return this.http.post<any>(`http://kyrsovoi/get_user.php`, formData).pipe(
      map((res) => {
        this.bul = res['data'];
        return this.bul;
      }));
  }
}
```

Лістинг Г.7 — Текст шаблону компоненту для авторізації користувача

```
<p class="text-xs-center" style="color: {{messageColor}}">{{message}}</p>
<form [formGroup]="form" (ngSubmit)="onSubmit()">
  <div class="form-group"
       [ngClass]="{'has-error': form.get('email').invalid &&
form.get('email').touched}"
  >
    <label for="email">Email</label>
    <input
      type="text"
      class="form-control underlined"
      id="email"
      placeholder="Введіть ваш email"
      formControlName="email"
    >
    <span class="form-help-text"
          *ngIf="form.get('email').invalid && form.get('email').touched"
    >
        <span *ngIf="form.get('email')['errors']['required']">email не може бути
порожнім</span>
        <span *ngIf="form.get('email')['errors']['email']">Введіть коректний
email</span>
      </span>
  </div>
  <div
    class="form-group"
    [ngClass]="{'has-error': form.get('password').invalid &&
form.get('password').touched}"
  >
    <label for="password">Пароль</label>
    <input
      type="password"
      class="form-control underlined"
      id="password"
      placeholder="Пароль"
      formControlName="password"
    >
    <span class="form-help-text"
          *ngIf="form.get('password').invalid && form.get('password').touched"
    >
        <span *ngIf="form.get('password')['errors']['required']">Пароль не може
бути порожнім</span>
        <span *ngIf="form.get('password')['errors']['minlength']">
          Пароль повинен бути довшим
{{form.get('password')['errors']['minlength']['requiredLength']}} символіГ.
          Зараз
{{form.get('password')['errors']['minlength']['actualLength']}}</span>
      </span>
  </div>
  <div class="form-group">
    <button

      type="submit"
      class="btn btn-block btn-primary"
      [disabled]="form.invalid"

    >Увійти
    </button>
  </div>
  <div class="form-group">
    <p class="text-muted text-xs-center">
      Немає облікового запису? <a
[routerLink]="'/registration'">Зареєструватися!</a>
    </p>
  </div>
</form>
```

Лістинг Г.8 — Текст програми модулю авторизації користувача

```
import {Component, OnInit} from '@angular/core';
import {Router} from '@angular/router';

@Component({
  selector: 'app-auth',
  templateUrl: './auth.component.html'
})
export class AuthComponent implements OnInit {
  constructor(private router: Router) {
  }

  ngOnInit() {
    //this.router.navigate(['/login', '/registration']);
  }
}
```

Лістинг Г.9 — Текст роутінг модулю авторизації користувача

```
import {NgModule} from '@angular/core';
import {RouterModule, Routes} from '@angular/router';

import {LoginComponent} from './login/login.component';
import {RegistrationComponent} from './registration/registration.component';
import {AuthComponent} from './auth.component';

const routes: Routes = [
  {
    path: '', component: AuthComponent, children: [
      {path: 'login', component: LoginComponent},
      {path: 'registration', component: RegistrationComponent}
    ]
  }

];

@NgModule({
  imports: [RouterModule.forChild(routes)],
  exports: [RouterModule]
})
export class AuthRoutingModule {
```

Лістинг Г.10 — Текст програмного коду error компоненту

```
import {Component, OnInit} from '@angular/core';

@Component({
  selector: 'app-not-found',
  templateUrl: './not-found.component.html',
  styleUrls: ['./not-found.component.css']
})
export class NotFoundComponent implements OnInit {

  constructor() {
  }
  ngOnInit(): void {
  }

}
```

Лістинг Г.11 — Текст шаблону коду error компоненту

```
<h1 style="color: red;">Error 404</h1>
```

Лістинг Г.12 — Текст програмного коду роутинг модулю головного модулю

```
import {NgModule} from '@angular/core';
import {RouterModule, Routes} from '@angular/router';
import {LoginComponent} from './auth/login/login.component';
import {NotFoundComponent} from './not-found/not-found.component';

const routes: Routes = [
  {path: 'login', component: LoginComponent},
  {path: 'system', loadChildren: () => import('./system/system.module').then(mod
=> mod.SystemModule)},
  {path: 'error', component: NotFoundComponent},
  // {path: '**', redirectTo: '/error'},

];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule {

}
```

Лістинг Г.13 — Текст програмного коду модулю головного модулю

```
import {BrowserModule} from '@angular/platform-browser';
import {NgModule} from '@angular/core';

import {AppComponent} from './app.component';
import {AuthModule} from './auth/auth.module';
import {AppRoutingModule} from './app.routing-module';
import {UsersService} from './shared/services/users.service';
import {HttpClientModule} from '@angular/common/http';
import {RouterModule} from '@angular/router';
import {RegistrationService} from './auth/registration/registration.service';
import {NotFoundComponent} from './not-found/not-found.component';
import {AuthGuard} from './auth-guard.service';
import {AuthService} from './auth.service';

@NgModule({
  declarations: [
    AppComponent,
    NotFoundComponent
  ],
  imports: [
    BrowserModule,
    AuthModule,
    RouterModule,
    AppRoutingModule,
    HttpClientModule,
  ],
  providers: [UsersService, RegistrationService, AuthService, AuthGuard,],
  bootstrap: [AppComponent]
})
export class AppModule {
}
```

Лістинг Г.14 — Текст програмного коду компоненту system

```
import {Component} from '@angular/core';
import {CookieService} from 'ngx-cookie-service';
import {FormControl, FormGroup, Validators} from '@angular/forms';
import {Router} from '@angular/router';
import {SearchService} from './search/search.service';

//import {SystemService} from './system.service';

@Component({
```

```
  selector: 'app-system',
  templateUrl: 'system.component.html',
  styleUrls: ['./system.component.css']
})
export class SystemComponent {
  id = this.cookieService.get('id');
  form: FormGroup;
  categoryNames = [ // ydali  i napishi normalniy zapros na bekend
    "Нерухомість",
    "Траспорт",
    "Хобі",
    "Електроніка",
    "Інструменти для дому",
    "Товари для дітей",
    "Одяг",
    "Спорт і туризм",
    "Товари для бізнесу",
    "Інше"
  ];
  constructor(
    private cookieService: CookieService,
    private router: Router,
    private searchService: SearchService,
    // private systemService: SystemService
  ) {
  }

  ngOnInit() {
    this.form = new FormGroup({
      searchValue: new FormControl(null, [Validators.required,
Validators.minLength(1)])
    });
  }

  onSubmit() {
    console.log(this.form.value);
    this.searchService.updateSearch(this.form.value.searchValue);
    this.router.navigate(['./system/search/', this.form.value.searchValue]);
  }

  IfAuth() {
    return this.cookieService.get('user');
  }}
```

Лістинг Г.15 — Текст шаблону компоненту system

```
<!DOCTYPE html>

<head>
  <meta charset="utf-8">
  <title></title>
</head>
<header>
  <div class="navigation">
    <div class="menu">
      <div class="menu-znak">
        <div class="znak-menu"></div>
        <div class="znak-menu"></div>
        <div class="znak-menu"></div>
      </div>
      <div class="menu-hover">
        <a *ngFor="let it of categoryNames" [routerLink]="['/system/categories',
it]" class="li">{{it}}</a>
      </div>
    </div>
    <div id="icon">
      <a [routerLink]="['/system/main']">
```

```
        <img src="assets/logo_omazon.png" width="75%">
      </a>
    </div>
    <form [formGroup]="form" (ngSubmit)="onSubmit()">
      <div id="find">
        <input type="" name="finder" id="find-bar" formControlName="searchValue">
        <button type="submit" [disabled]="form.invalid" class="buttonSeacrh">
          <img src="assets/loupe.png" width="38px"></button>
      </div>
    </form>
    <div id="my-profile">
      <img src="http://kyrsovoi/S2.png" width="38x">
      <h1>
        <a *ngIf="IfAuth(); else succesStatus"
[routerLink]="['/system/clientSys/client', id]">Мій профіль</a>
        <ng-template #succesStatus>
          <a [routerLink]="['../registration']">Мій профіль</a>
        </ng-template>

      </h1>
    </div>
  </div>
</header>
<div class="container" style="height: auto">
  <router-outlet></router-outlet>
</div>

<footer>
  <div id="information">
    <a [routerLink]="['/system/main']"><h5>Умови користування</h5></a>
    <a [routerLink]="['/system/main']"><h5>Допомога і Зворотній зв'язок</h5></a>
    <a [routerLink]="['/system/main']"><h5>Політика конфіденційності</h5></a>
  </div>
</footer>
```

Лістинг Г.16 — Текст стилів компоненту system

```
header {
  position: relative;
}

* {
  box-sizing: border-box;
}

h1, h2, h3, h4, h5 {
  margin: 0;
  padding: 0;
}

.navigation {
  background-color: #131921;
  width: 100%;
  max-width: 2000px;
  height: 6.5vh;
}

.menu {
  padding: 20px 0;
  padding-left: 30px;
}

.znak-menu:first-child {
  margin-top: 0;
}

.znak-menu {
```

```
  width: 30px;
  height: 4px;
  background-color: white;
  margin: 6px 0;
}

.menu-hover {
  position: absolute;
  top: 68px;
  left: 0;
  max-width: 300px;
  width: 100%;
  z-index: 1;
  display: none;
  opacity: 0.9;
}

.li {
  padding: 20px 0;
  background-color: #33484b;
  display: block;
  text-decoration: none;
  text-align: center;
  color: white;
  font-size: 24px;
}

.li:hover {
  background-color: #33484b;
  color: red;
  transition: background-color 0.2s;
}

.menu:hover .menu-hover {
  display: inherit;
}

#icon {
  position: absolute;
  width: 10%;
  height: 4%;
  left: 130px;
  top: 13px;
}

#find {
  position: absolute;
  height: 5%;
  left: 24%;
  top: 13px;
  width: 60%;

}

#find-bar {
  width: 80%;
  height: 38px;
  outline: none;
  border: none;
}

#button-find {
  position: absolute;
  width: 50px;
  height: 38px;
  color: yellow;
  background-color: #febd69;
```

```
}

#find > a > img {
  position: absolute;
  padding-left: 5px;
  padding-top: 2px;
}

#my-profile {
  left: 78%;
  position: absolute;
  font-size: 10px;
  top: 10px;
  display: flex;
}

#my-profile > img {
  margin-left: 20px;
  position: absolute;
}

#my-profile > h1 {
  font-family: Playfair Display;
  margin-left: 65px;
}

#my-profile > h1 :hover {
  color: yellow;
}

#my-profile > h1 > a {
  color: white;
  text-decoration: none;
}

.container {
  background-color: #f0f1f3;
  max-width: 1400px;
  min-height: 165vh;
  margin: auto;
}

footer {
  background-color: #33484b;
  height: 15vh;
  position: relative;
}

#information {
  left: 10%;
  top: 10px;
  position: absolute;
}

#information > a {
  font-size: 20px;
  color: white;
  text-decoration: none;
  margin-top: 5px;
  display: flex;
}

@media (max-width: 1650px) {
  .find {
    max-width: 700px;
    width: 100%;
  }
```

```
}

.buttonSeacrh {
  background-color: yellow;
}

.buttonSeacrh:disabled {
  background-color: gray;
}
```

Лістинг Г.17 — Текст модулю компоненту system

```
import {NgModule} from '@angular/core';
import {CommonModule} from '@angular/common';
import {SharedModule} from '../shared/shared.module';
import {SystemRoutingModule} from './system-routing.module';
import {MainPageComponent} from './main-page/main-page.component';
import {SystemComponent} from './system.component';
import {ProductAddComponent} from './product-add/product-add.component';
import {ClientPageComponent} from './client-system/client-page/client-
page.component';
import {ProductPageComponent} from './product-page/product-page.component';
import {ClientProductsComponent} from './client-system/client-products/client-
products.component';
import {ClientBetsComponent} from './client-system/client-bets/client-
bets.component';
import {ClientWinsComponent} from './client-system/client-wins/client-
wins.component';
import {ClientSystemComponent} from './client-system/client-system.component';
import {SearchComponent} from './search/search.component';


@NgModule({
  imports: [
    CommonModule,
    SharedModule,
    SystemRoutingModule
  ],
  declarations: [
    MainPageComponent,
    SystemComponent,
    ProductAddComponent,
    ClientPageComponent,
    ProductPageComponent,
    ClientProductsComponent,
    ClientBetsComponent,
    ClientWinsComponent,
    ClientSystemComponent,
    SearchComponent
  ]


})
export class SystemModule {

}
```

Лістинг Г.18 — Текст роутінг модулю компоненту system

```
import {NgModule} from '@angular/core';
import {RouterModule, Routes} from '@angular/router';
import {SystemComponent} from './system.component';
import {MainPageComponent} from './main-page/main-page.component';
import {ProductAddComponent} from './product-add/product-add.component';
import {ClientPageComponent} from './client-system/client-page/client-
page.component';
import {ProductPageComponent} from './product-page/product-page.component';
import {AuthGuard} from '../auth-guard.service';
```

```
import {NotFoundComponent} from '../not-found/not-found.component';

import {ClientProductsComponent} from './client-system/client-products/client-
products.component';
import {ClientBetsComponent} from './client-system/client-bets/client-
bets.component';
import {ClientWinsComponent} from './client-system/client-wins/client-
wins.component';
import {ClientSystemComponent} from './client-system/client-system.component';
import {SearchComponent} from './search/search.component';


const routes: Routes = [
  // {path: 'main', component: MainPageComponent},
  {
    path: '', component: SystemComponent, children: [
      {path: 'main', component: MainPageComponent},
      {path: 'add', component: ProductAddComponent, canActivate: [AuthGuard]},
      {path: 'edit/:id_tov', component: ProductAddComponent, canActivate:
[AuthGuard]},

      {path: 'product', redirectTo: 'error'},
      {path: 'product/:id', component: ProductPageComponent},
      {path: 'categories', redirectTo: 'error'},
      {path: 'error', component: NotFoundComponent},
      {path: 'search', redirectTo: 'error'},
      {path: 'search/:name', component: SearchComponent},
      {path: 'categories/:idCategory', component: SearchComponent},
      {
        path: 'clientSys', component: ClientSystemComponent, canActivate:
[AuthGuard], children: [
          {path: 'client', redirectTo: 'error'},
          {path: 'client/:id', component: ClientPageComponent, canActivate:
[AuthGuard]},
          {path: 'client_products', component: ClientProductsComponent,
canActivate: [AuthGuard]},
          {path: 'my-bets', component: ClientBetsComponent, canActivate:
[AuthGuard]},
          {path: 'my-wins', component: ClientWinsComponent, canActivate:
[AuthGuard]},
        ]
      },

    ]
  }
];

@NgModule({
  imports: [RouterModule.forChild(routes)],
  exports: [RouterModule]

})
export class SystemRoutingModule {

}
```

Лістинг Г.19 — Текст роутінг модулю компоненту system

```
import {NgModule} from '@angular/core';
import {RouterModule, Routes} from '@angular/router';
import {SystemComponent} from './system.component';
import {MainPageComponent} from './main-page/main-page.component';
import {ProductAddComponent} from './product-add/product-add.component';
import {ClientPageComponent} from './client-system/client-page/client-
page.component';
import {ProductPageComponent} from './product-page/product-page.component';
import {AuthGuard} from '../auth-guard.service';
import {NotFoundComponent} from '../not-found/not-found.component';
```

Лістинг Г.20 — Текст програми компоненту search

```
import {Component, OnInit} from '@angular/core';
import {productSearch} from './productSearch';
import {SearchService} from './search.service';
import {ActivatedRoute} from '@angular/router';
import {CategoryService} from '../product-add/category.service';
import {Product} from '../main-page/product';
import {Categories} from '../product-add/category';
import {FormControl, FormGroup, Validators} from '@angular/forms';

@Component({
  selector: 'app-search',
  templateUrl: './search.component.html',
  styleUrls: ['./search.component.css']
})
export class SearchComponent implements OnInit {

  form: FormGroup;
  products: productSearch[];
  error: string;
  searchName = '';
  category: Categories[];
  private mess: string;

  constructor(
    private searchService: SearchService,
    private route: ActivatedRoute,
    private categoryService: CategoryService
  ) {
  }

  ngOnInit(): void {
    this.form = new FormGroup({
      'category': new FormControl(0, [Validators.required]),
      'minDay': new FormControl(0, [Validators.required, Validators.pattern('[1-
6]+')]),
      'maxDay': new FormControl(0, [Validators.required, Validators.pattern('[2-
7]+')]),
      'rateMin': new FormControl(0, [Validators.required, Validators.pattern('[0-
9]+')]),
      'rateMax': new FormControl(0, [Validators.required, Validators.pattern('[0-
9]+')])
    });
    this.getCategories();
    if (this.route.snapshot.params.name){
      this.searchName = this.route.snapshot.params.name;
    }else{
      this.form.value.category = this.route.snapshot.params.idCategory;
      this.form.patchValue({
        category: this.route.snapshot.params.idCategory
      });
    }
    this.searchService.updateSearch(this.searchName);
    this.searchService.search$.subscribe(queryString => {
      this.searchName = queryString;
      this.getProductSearch();

    });
  }

  onSubmit() {
    this.getProductSearch();
  }

  getProductSearch(): void {
```

```
    this.searchService.getProducts(this.searchName, this.form).subscribe(
      (res: productSearch[]) => {
        console.log(res);
        this.products = res.map(item => new productSearch({...item}));
        console.log(this.products);
        console.log(this.products[0].id_images);
      },
      (err) => {
        this.error = err;
      }
    );
  }

  Days(day: number): string {
    // tslint:disable-next-line:label-position no-unused-expression

    if (day >= 5) {
      this.mess = 'днів';
    }
    if (day >= 2 && day <= 4) {
      this.mess = 'дні';
    }
    if (day <= 1) {
      this.mess = 'день';
    }
    return this.mess;
  }

  getCategories(): void {
    this.categoryService.getCategories().subscribe(
      (res: Product[]) => {
        this.category = res;
      },
      (err) => {
        this.error = err;
      }
    );
  }

  deleteFilters() {
    this.form.patchValue({
      category: 0,
      minDay: 0,
      maxDay: 0,
      rateMin: 0,
      rateMax: 0
    });
    this.getProductSearch();
  }
}
```

Лістинг Г.21 — Текст програми шаблону компоненту search

```
<div class="filters">
  <form [formGroup]="form" (ngSubmit)="onSubmit()">
    <div class="filter1">

      <div class="filter-in-category">
        <span>Категорія</span>
        <select formControlName="category">
          <option value="0">-Не обрано-</option>
          <option *ngFor="let item of category"
value="{{item.name}}">{{item.name}}</option>
        </select>
      </div>

      <div class="filter-in-day">
        <span>Срок ставок</span>
```

```
        <div class="start-end-day">
          <i>Від</i>
          <input type="text" name="" class="enter-day-min"
formControlName="minDay">
          <i>До</i>
          <input type="text" name="" class="enter-day-max"
formControlName="maxDay">
        </div>
      </div>
      <div class="filter-in-sum">
        <span>Сумма ставки</span>
        <div class="start-end-sum">
          <i>Від</i>
          <input type="text" name="" class="inputput-sum-min"
formControlName="rateMin">
          <i>До</i>
          <input type="text" name="" class="input-sum-max"
formControlName="rateMax">
        </div>
      </div>
    </div>
    <div class="enter">
      <button type="submit" class="enter-filter">Примінити фільтри</button>
      <button class="drop-filter" (click)="deleteFilters()">Скасувати
фільтри</button>
    </div>
  </form>
</div>
<h3 class="naideno">Товарів було знайдено: {{products?.length}}</h3>
<div class="list-tovar">

  <div class="tovar" *ngFor="let items of products">
    <div class="img">
      <img src="http://kyrsovoi/photos/{{items.id_images}}.jpg">
    </div>
    <div class="name-tovar">
      <span>{{items.name}}</span>
      <span>Категорія: {{items.category}}</span>
    </div>
    <div class="info">
      <span>Актуальна ставка: {{items.samaStavka}}</span>
      <span>Мінімальна ставка: {{items.minRate}}</span>
      <span *ngIf="7 - items.dayloss > 0">Дата окінчення: {{7 - items.dayloss}}
{{Days(7 - items.dayloss)}}</span>
      <span *ngIf="7 - items.dayloss <= 0">Ставки на цій товар закінчено</span>
    </div>
    <div class="go-in-tovar">
      <a [routerLink]="['/system/product', items.id_tovar]">На сторінку</a>
    </div>
  </div>

</div>
```

Лістинг Г.22 — Текст програми сервісу компоненту search

```
import {Injectable} from '@angular/core';
import {HttpClient, HttpErrorResponse} from '@angular/common/http';

import {BehaviorSubject, Observable, throwError} from 'rxjs';
import {catchError, map} from 'rxjs/operators';
import {productSearch} from './productSearch';
import {FormGroup} from '@angular/forms';


@Injectable({
  providedIn: 'root'
})
export class SearchService {
```

```
  constructor(private http: HttpClient) {
  }

  baseUrl = 'http://kyrsovoi';
  products: productSearch[];
  search$: BehaviorSubject<string> = new BehaviorSubject<string>(null);

  getProducts(name: string, form: FormGroup): Observable<productSearch[]> {
    const formData = new FormData();
    formData.append('name', name);
    console.log(form);
    Object.keys(form.value).forEach(key => formData.append(key, form.value[key]));

    return this.http.post(`${this.baseUrl}/get_SearchProduct.php`, formData)
      .pipe(
        map((res) => {
          this.products = res['data'];

          return this.products;
        }),
        catchError(this.handleError));
  }

  private handleError(error: HttpErrorResponse) {
    console.log(error);

    // return an observable with a user friendly message
    return throwError('error');
  }

  updateSearch(qwery: string): void {
    this.search$.next(qwery);
  }
}
```

Лістинг Г.23 — Текст програми компоненту product-page

```
import {Component, OnInit} from '@angular/core';

import {Product} from './product';
import {productPageService} from './product-page.service';
import {ActivatedRoute} from '@angular/router';
import {FormControl, FormGroup, Validators} from '@angular/forms';
import {CookieService} from 'ngx-cookie-service';
import {imageg} from './imageg';
import {toNumbers} from '@angular/compiler-
cli/src/diagnostics/typescript_version';

@Component({
  selector: 'app-product-page',
  templateUrl: './product-page.component.html',
  styleUrls: ['./product-page.component.css']
})
export class ProductPageComponent implements OnInit {
  // tslint:disable-next-line:no-shadowed-variable
  product: Product;
  images: imageg[];
  error = '';
  success = '';
  id: string;
  activeProduct = 1;
  form: FormGroup;
  idClient = this.cookieService.get('id');

  constructor(
    private productPageService: productPageService,
    private route: ActivatedRoute,
    private cookieService: CookieService
```

```
  ) {
  }



  ngOnInit() {
    this.id = this.route.snapshot.params.id;

    this.getProduct();
    this.getImages();

  }

  getProduct(): void {
    this.productPageService.getAll(this.id).subscribe(
      (res: Product) => {
        console.log(res);
        this.product = new Product({...res});
        this.form = new FormGroup({
          bet: new FormControl(null, [Validators.required, Validators.pattern('[0-
9]+'), Validators.min(this.product.min_rate)])
        });
      },
      (err) => {
        this.error = err;
      }
    );
  }

  getImages(): void {
    this.productPageService.getImages(this.id).subscribe(
      (res: imageg[]) => {
        this.images = res.map(item => new imageg({item}));
        // console.log(this.images[0].item['id_images']);
      },
      (err) => {
        this.error = err;
      }
    );

  }

  onSubmit() {
    console.log(+this.product.samaStavka + 1 * this.form.value.bet);
    this.productPageService.postBet(this.id, (+this.product.samaStavka + 1 *
this.form.value.bet).toString(), this.idClient).subscribe((id: string) => {
      this.getProduct();
    });

  }

  CheckActive() {
    if (this.product.dayloss >= 7) {
      this.activeProduct = 0;
    }
    return this.activeProduct;
  }

  private mess: string;

  Days(day: number): string {
    // tslint:disable-next-line:label-position no-unused-expression

    if (day >= 5 || day == 0) {
      this.mess = 'днів';
    }
    if (day >= 2 && day <= 4) {
```

```
      this.mess = 'дні';
    }
    if (day === 1) {
      this.mess = 'день';
    }
    return this.mess;
  }

  IfAuth() {
    return this.cookieService.get('user');
  }

  selectImage(id_images: string): void {
    this.product.image = id_images;
  }
}
```

Лістинг Г.24 — Текст програми шаблону компоненту product-page

```
<form *ngIf="product">

  <div class="img-profile-rate">

    <div class="img-slider">
      <div class="img">
        <img src="http://kyrsovoi/photos/{{product.image}}.jpg">
      </div>
      <div class="slider">
        <li  class="imageproduct">
         <img *ngFor="let items of images"
src="http://kyrsovoi/photos/{{items.item.id_images}}.jpg"
(click)="selectImage(items.item.id_images)" alt="">
        </li>
      </div>
    </div>

    <div class="profile-rate">

      <div class="img-profile-name">
        <img src="http://kyrsovoi/S2.png" width="30%" height="30%">
        <div class="name-email">
          <span>{{product.cliName}}</span>
          <span>{{product.cliEmail}}</span>
        </div>
      </div>


      <div class="rate">
        <form [formGroup]="form" (ngSubmit)="onSubmit()">
          <div class="text-rate">

            <span *ngIf="CheckActive()">До кінця аукціону: {{7 - product.dayloss}}
{{Days(7 - product.dayloss)}}.</span>
            <span>Остання ставка: {{product.samaStavka}} грн.</span>
            <span>Мінімальна ставка   {{product.min_rate}} грн.</span>
            <div *ngIf="CheckActive(); else succesStatus" id="add-rate">



                        <span class="upBet">Підняти ставку на



                        <input
                type="text"
                name=""
                id="input_rate"
```

```
                formControlName="bet"
              > грн.
               <p class="form-help-text" *ngIf="form.get('bet').invalid &&
form.get('bet').touched">
                 Ставка невалідна
              </p>
              </span>

            </div>
            <div *ngIf="CheckActive()" id="but">
              <button *ngIf="IfAuth(); else goRegister"
                      type="submit"
                      class="add_rate"
                      [disabled]="form.invalid"
              >
                Підняти ставку
              </button>
              <ng-template #goRegister>
                <a [routerLink]="['//registration']">
                  <button

                    class="add_rate"
                  >
                    Підняти ставку
                  </button>
                </a>
              </ng-template>
            </div>

          </div>

          <ng-template #succesStatus>
            <p>Ставки на цій товар закінчено</p>
          </ng-template>
        </form>
      </div>
    </div>


  </div>

  <div class="name-desc">
    <h3>{{product.name}}</h3>
    <p>Категория: {{product.catName}}</p>
    <span><p>{{product.about}}</p></span>
  </div>

</form>
```

Лістинг Г.25 — Текст програми стилів компоненту product-page

```
/*             */
.img-profile-rate {
  max-width: 1000px;
  width: 100%;
  margin: 0 auto;
  padding-top: 100px;
  display: flex;
  justify-content: space-between;
}

.img-slider {
  max-width: 400px;
  width: 100%;
}
```

```
.img {
  width: 100%;
  height: 400px;
}

.img > img {
  width: 100%;
  height: 100%;
}

.slider {
  width: 100%;
  height: 80px;
  background-color: white;
}

.slider > h3 {
  font-weight: 50;
  font-size: 40px;
  text-align: center;
  padding: 10px 0;
}

.profile-rate {
  max-width: 532px;
  width: 100%;
  height: 355px;
}

.img-profile-name {
  display: flex;
  justify-content: space-between;
  padding: 5px 40px;
  border: 2px solid green;
}

.img-profile-name > span {
  font-size: 40px;
  display: block;
  padding-top: 40px;
}

.name-email span {
  display: block;
  font-size: 24px;
  padding-top: 17px;
}

.rate {
  margin-top: 20px;
  height: 100%;
  border: 2px solid green;
}

.text-rate {
  text-align: center;
  padding-top: 20px;
}

.text-rate span {
  font-weight: 400;
  line-height: 1.5;
  font-size: 30px;
  display: block;
}

.text-rate .upBet {
```

```
  padding-top: 20px;
}

#input_rate {
  border: none;
  border-bottom: 2px solid green;
  width: 100px;
  font-size: 30px;
  background-color: #f0f1f3;
}

.add_rate {
  padding: 10px 40px;
  display: block;
  text-align: center;
  background-color: #d9f091;
  width: 250px;
  margin: 0 auto;
  margin-top: 20px;
  /*font-size: 24px;*/
  border: 2px solid green;
  text-decoration: none;
  color: black;
}

.name-desc {
  height: 250px;
  background-color: white;
  max-width: 1000px;
  margin: 0 auto;
  margin-top: 50px;
}

.name-desc > h3 {
  text-align: center;
  padding-top: 20px;
}

.name-desc > span {
  margin-top: 30px;
  padding: 0 40px;
  display: block;
  font-size: 18px;
}

/*                    */
@media (max-width: 800px) , (max-device-width: 800px) {

  .img-profile-rate {
    flex-wrap: wrap;
    width: 600px;
    justify-content: center;
  }

  .profile-rate {
    margin-top: 30px;
  }

  .name-desc {
    position: relative;
    top: 130px;
  }
}
.imageproduct > img {
  max-height: 80px;
padding: 1px;
}
```

```
.imageproduct {
  padding: 5px;
  background-color: #fff;
  min-width: 100%;
  min-height: 10px;
  list-style-type: none;
  display: flex;
  overflow-x: auto;
}
.form-help-text {
  padding-top: 1px;
  color: #fb494d;
  font-size: 13px;
  text-align: right;
  padding-right: 120px;
}
```

Лістинг Г.26 — Текст програми сервісу компоненту product-page

```
import {Injectable} from '@angular/core';
import {HttpClient, HttpErrorResponse} from '@angular/common/http';

import {Observable, throwError} from 'rxjs';
import {catchError, map} from 'rxjs/operators';

import {Product} from './product';
import {imageg} from './imageg';

@Injectable({
  providedIn: 'root'
})
// tslint:disable-next-line:class-name
export class productPageService {
  baseUrl = 'http://kyrsovoi';
  product: Product;
  images: imageg[];

  constructor(private http: HttpClient) {
  }

  getAll(id: string): Observable<Product> {
    const formData = new FormData();
    formData.append('id', id);
    return this.http.post(`${this.baseUrl}/get_product.php`, formData)
      .pipe(
        map((res) => {
          this.product = res['data'];

          return this.product;
        }),
        catchError(this.handleError));
  }

  private handleError(error: HttpErrorResponse) {
    console.log(error);

    // return an observable with a user friendly message
    return throwError('error');
  }

  // tslint:disable-next-line:typedef
  postBet(id: string, betts : string, idClient) {
    const formData = new FormData();
    formData.append('id', id);
    formData.append('idClient', idClient);
    formData.append('bet', betts);
```

```
    return this.http.post<any>(`${this.baseUrl}/post_bet.php`, formData);
  }

  getImages(id: string): Observable<imageg[]> {
    const formData = new FormData();
    formData.append('id', id);
    return this.http.post(`${this.baseUrl}/get_img_of_product.php`, formData)
      .pipe(
        map((res) => {
          this.images = res['data'];
          return this.images;
        }),
        catchError(this.handleError));

  }


}

//const formData = new FormData();
//Object.keys(product).forEach( key => formData.append(key, product[key]));
```

Лістинг Г.27 — Текст класу ProductEdit компоненту product-add

```
export class ProductsEdit {
  name: string;
  id_category: string;
  samaStavka: number;
  about: string;
  min_rate: number;
  photos: [];
  constructor(
    values: object
  ) {
    Object.keys(values).forEach(key => {
      this[key] = values[key];
    });
  }
}
```

Лістинг Г.28 — Текст програми сервісу компоненту product-add

```
import {Injectable} from '@angular/core';
import {HttpClient, HttpErrorResponse} from '@angular/common/http';

import {Observable, throwError} from 'rxjs';
import {catchError, map} from 'rxjs/operators';
import {Categories} from './category';

@Injectable({
  providedIn: 'root'
})
export class CategoryService {
  baseUrl = 'http://kyrsovoi/get_category.php';
  category: Categories[];

  constructor(private http: HttpClient) {
  }

  getCategories(): Observable<Categories[]> {
    return this.http.get(`${this.baseUrl}`).pipe(
      map((res) => {
        this.category = res['data'];
        return this.category;
      }),
      catchError(this.handleError));
```

```
  }

  private handleError(error: HttpErrorResponse) {
    console.log(error);

    // return an observable with a user friendly message
    return throwError('privetkakdela');
  }
}
```

Лістинг Г.29 — Текст програми шаблону компоненту product-add

```
<link
href="https://fonts.googleapis.com/css2?family=Playfair+Display&display=swap"
rel="stylesheet">


<div class="add-new-tovar-text">Додавання нового товару</div>
<form [formGroup]="form" (ngSubmit)="onSubmit()">
  <div class="block-add-pole">
    <div class="name-tovar">
      <span class="name-text">Назва товару</span>
      <input
        type="text"
        name=""
        class="add-name"
        formControlName="name"
      >
      <span class="form-help-text"
            *ngIf="form.get('name').invalid && form.get('name').touched"
      >
        <span class="form-help-text"
              *ngIf="form.get('name')['errors']['required']">Назва товару не може
бути порожньою</span>
         <span class="form-help-text"
*ngIf="form.get('name')['errors']['maxlength']">
         пароль повинен бути коротше
           {{form.get('name')['errors']['maxlength']['requiredLength']}} символіГ.
          Зараз {{form.get('name')['errors']['maxlength']['actualLength']}}</span>
      </span>
    </div>
    <div class="category-tovar">
      <span class="name-text">Категорія товару</span>
      <select class="add-category" formControlName="category">
        <option id="type" *ngFor="let item of category"
value="{{item.id_category}}">{{item.name}}</option>
      </select>
    </div>
    <div class="price">
      <div class="price-tovar">
        <span class="name-text">Початкова ціна</span>
        <input
          type="text"
          name=""
          class="add-price"
          formControlName="price"
        >
        <div class="grn">Грн.</div>
        <span class="form-help-text"
              *ngIf="form.get('price').invalid && form.get('price').touched"
        >
           <span class="form-help-text"
*ngIf="form.get('price')['errors']['required']">Початкова ціна товару не може бути
порожньою</span>
          <span class="form-help-text"
*ngIf="form.get('price')['errors']['pattern']">Початкова ціна невалідна</span>
```

```
 </span>

      </div>

      <div class="price-min">
        <span class="name-text">Мінімальна ставка</span>
        <input
          type="text"
          name=""
          class="add-price"
          formControlName="min_price"
        >
        <div class="grn">Грн.</div>
        <span class="form-help-text"
              *ngIf="form.get('min_price').invalid &&
form.get('min_price').touched"
        >
           <span class="form-help-text"
*ngIf="form.get('min_price')['errors']['required']">Мінімальна ціна товару не може
бути порожньою</span>
          <span class="form-help-text"
                *ngIf="form.get('min_price')['errors']['pattern']">Мінімальна ціна
невалідна</span>
           </span>
      </div>
    </div>
    <div class="desc-tovar">
      <span class="name-text">Описання товару</span>
      <textarea
        class="add-desc"
        formControlName="about"
      >

      </textarea>
      <span class="form-help-text"
            *ngIf="form.get('about').invalid && form.get('about').touched"
      >
        Описання не може бути порожнім
          </span>
    </div>

    <div class="add-photo">
      <input
        style="display: none"
        #fileInput
        multiple
        type="file" (change)="onFileSelected($event)"

      >
      <button type="button" (click)="fileInput.click()">Завантажити фото</button>
    </div>
    <li class="imageproduct">
      <img *ngFor="let link of links" id="" src="{{ link }}" alt="">
    </li>
    <div class="add-lot">
      <button class="button" type="submit" name="names"
[disabled]="form.invalid">Відправити</button>
    </div>
  </div>
</form>
```

Лістинг Г.30 — Текст програми компоненту product-add

```
import {Component, OnInit} from '@angular/core';

import {Categories} from './category';
import {CategoryService} from './category.service';
```

```
import {FormControl, FormGroup, Validators} from '@angular/forms';
import {ProductAddService} from './product-add.service';
import {Products} from './product.model';
import {HttpClient} from '@angular/common/http';
import {CookieService} from 'ngx-cookie-service';
import {ActivatedRoute, Router} from '@angular/router';
import {ProductsEdit} from './ProductEdit';

@Component({
  selector: 'app-product-add',
  templateUrl: './product-add.component.html',
  styleUrls: ['./product-add.component.css']
})
export class ProductAddComponent implements OnInit {
  form: FormGroup;
  category: Categories[];
  edId = '0';
  error = '';
  success = '';
  selectedFile: File[] = [];
  productEdit: ProductsEdit;
  links: Array<string | ArrayBuffer> = [];

  constructor(
    private cookieService: CookieService,
    private route: ActivatedRoute,
    private categoryService: CategoryService,
    private productAddService: ProductAddService,
    private http: HttpClient,
    private router: Router) {
  }


  ngOnInit() {
this.ifEdit();
    this.getCategories();
    this.form = new FormGroup({
      'name': new FormControl(null, [Validators.required,
Validators.maxLength(50)]),
      'category': new FormControl(1, [Validators.required]),
      'price': new FormControl(null, [Validators.required, Validators.pattern('[0-
9]+')]),
      'about': new FormControl(null, [Validators.required]),
      'photos': new FormControl(false, [Validators.required]),
      'min_price': new FormControl(null, [Validators.required,
Validators.pattern('[0-9]+')]),
      'id': new FormControl(false, [Validators.required])
    });
  }

  onSubmit() {
    this.form.value.photos = this.selectedFile;
    this.form.value.id = this.cookieService.get('id');
    const {name, category, price, about, photos, min_price, id} = this.form.value;
    const product = new Products(name, category, price, about, photos, min_price,
id);
    if(this.route.snapshot.params.id_tov){
      alert("Товар успішно відредаговано");

    }else{
      alert("Товар успішно додано");
    }
    // tslint:disable-next-line:no-shadowed-variable
    console.log(this.form.value);
    this.productAddService.addProduct(product, this.edId).subscribe((product:
Products) => {
      console.log(this.form.value);
```

```
    });
    this.router.navigate(['/system/clientSys/client_products']);
  }

  getCategories(): void {
    this.categoryService.getCategories().subscribe(
      (res: Categories[]) => {
        this.category = res;
      },
      (err) => {
        this.error = err;
      }
    );
  }

  onFileSelected(event) {
    this.selectedFile = event.target.files;
    this.links = [];
    for (let i = 0; i < this.selectedFile.length; i++) {
      const fr = new FileReader();

      fr.addEventListener('load', () => {
        this.links.push(fr.result);
      }, false);

      fr.readAsDataURL(this.selectedFile[i]);
    }
  }

  ifEdit(){
    if(this.route.snapshot.params.id_tov){
this.EditProduct();
this.edId = this.route.snapshot.params.id_tov;
    }
  }
  EditProduct(){
this.productAddService.getEdit(this.route.snapshot.params.id_tov).subscribe(
  (res: ProductsEdit) => {
    console.log(res);
    this.productEdit = new ProductsEdit({...res});
    this.links = [];
    for (let i of this.productEdit.photos){
      this.links.push("http://kyrsovoi/photos/"+i+".jpg");
    }
    this.form.patchValue({
      name: this.productEdit.name,
      category: this.productEdit.id_category,
      price: this.productEdit.samaStavka,
      min_price: this.productEdit.min_rate,
      about: this.productEdit.about
    });
  },
  (err) => {
    this.error = err;
  }
);
  }
}
```

Лістинг Г.31 — Текст сервісу компоненту product-add

```
import {Injectable} from '@angular/core';
import {HttpClient, HttpErrorResponse} from '@angular/common/http';
import {Observable, throwError} from 'rxjs';

import {Products} from './product.model';
import {catchError, map} from 'rxjs/operators';
```

```
import {ProductsEdit} from './ProductEdit';

@Injectable({
  providedIn: 'root'
})
export class ProductAddService {

  productEdit: ProductsEdit;
  constructor(private http: HttpClient) {
  }

  addProduct(product: Products, editId: string): Observable<any> {
    console.log(product);
    const formData = new FormData();
    Object.keys(product).forEach(key => formData.append(key, product[key]));
    Object.keys(product['photos']).forEach(key => formData.append(key,
product['photos'][key]));
    formData.append('editId', editId);
    return this.http.post<any>(`http://kyrsovoi/addProduct.php`, formData);
  }

  getEdit(tov: string ): Observable<ProductsEdit>{
    console.log(tov);
    const formData = new FormData();
    formData.append('tov', tov);
    return this.http.post(`http://kyrsovoi/get_editProduct.php`, formData)
      .pipe(
        map((res) => {
          this.productEdit = res['data'];

          return this.productEdit;
        }),
        catchError(this.handleError));
  }
  private handleError(error: HttpErrorResponse) {
    console.log(error);

    // return an observable with a user friendly message
    return throwError('error');
  }
}
```

Лістинг Г.32 — Текст шаблону компоненту main-page

```
<div class="categories">
  <div id="category">
    <h3>Нерухомість</h3>
    <a [routerLink]="['/system/categories', this.categoryNames[0]]">Дивитися
все</a>
    <div class="images-tovars">
      <ul class="imageproduct" href="">
        <li *ngFor="let item of products">
          <a [routerLink]="['/system/product', item.id_tovar]">
            <img *ngIf="item.id_category==1"
src="http://kyrsovoi/photos/{{item.image_tovar}}.jpg" class="img-prod"
                 width="15%" height="15%">
          </a>
        </li>
      </ul>
    </div>
  </div>
  <div id="category">
    <h3>Транспорт</h3>
    <a [routerLink]="['/system/categories', this.categoryNames[1]]">Дивитися
все</a>
    <div class="images-tovars">
      <ul class="imageproduct" href="">
```

```
        <li *ngFor="let item of products">
          <a [routerLink]="['/system/product', item.id_tovar]">
            <img *ngIf="item.id_category==2"
src="http://kyrsovoi/photos/{{item.image_tovar}}.jpg" width="15%"
                 height="90%">
          </a>
        </li>
      </ul>
    </div>
  </div>
  <div id="category">
    <h3>Хобі</h3>
    <a [routerLink]="['/system/categories', this.categoryNames[2]]">Дивитися
все</a>
    <div class="images-tovars">
      <ul class="imageproduct" href="">
        <li *ngFor="let item of products">
          <a [routerLink]="['/system/product', item.id_tovar]">
            <img *ngIf="item.id_category==3"
src="http://kyrsovoi/photos/{{item.image_tovar}}.jpg" width="15%"
                 height="15%">
          </a>
        </li>
      </ul>
    </div>
  </div>
  <div id="category">
    <h3>Електроніка</h3>
    <a [routerLink]="['/system/categories', this.categoryNames[3]]">Дивитися
все</a>
    <div class="images-tovars">
      <ul class="imageproduct" href="">
        <li *ngFor="let item of products">
          <a [routerLink]="['/system/product', item.id_tovar]">
            <img *ngIf="item.id_category==4"
src="http://kyrsovoi/photos/{{item.image_tovar}}.jpg" width="15%"
                 height="90%">
          </a>
        </li>
      </ul>
    </div>
  </div>
</div>
```

Лістинг Г.33 — Текст компоненту main-page

```
import {Component, OnInit} from '@angular/core';
import {Product} from './product';
import {MainPageService} from './main-page.service';
import {ActivatedRoute} from '@angular/router';

@Component({
  selector: 'app-main-page',
  templateUrl: './main-page.component.html',
  styleUrls: ['./main-page.component.css']
})
export class MainPageComponent implements OnInit {
  products: Product[];
  error = '';
  success = '';
categoryNames = [// ydali  i napishi normalniy zapros na bekend
  "Нерухомість",
  "Траспорт",
  "Хобі",
  "Електроніка",
  "Інструменти для дому",
  "Товари для дітей",
```

```
  "Одяг",
  "Спорт і туризм",
  "Товари для бізнесу",
  "Інше"
];
  constructor(
    private productService: MainPageService,
    private route: ActivatedRoute
  ) {
  }

  ngOnInit() {
    this.getProducts();
  }

  getProducts(): void {
    this.productService.getAll().subscribe(
      (res: Product[]) => {
        this.products = res;
      },
      (err) => {
        this.error = err;
      }
    );
  }

}
```

Лістинг Г.34 — Текст сервісу компоненту main-page

```
import {Injectable} from '@angular/core';
import {HttpClient, HttpErrorResponse} from '@angular/common/http';

import {Observable, throwError} from 'rxjs';
import {catchError, map} from 'rxjs/operators';

import {Product} from './product';

@Injectable({
  providedIn: 'root'
})
export class MainPageService {
  baseUrl = 'http://kyrsovoi/get_products.php';
  products: Product[];

  constructor(private http: HttpClient) {
  }

  getAll(): Observable<Product[]> {
    return this.http.get(`${this.baseUrl}`).pipe(
      map((res) => {
        this.products = res['data'];
        console.log(this.products);
        return this.products;
      }),
      catchError(this.handleError));
  }

  private handleError(error: HttpErrorResponse) {
    console.log(error);

    // return an observable with a user friendly message
    return throwError('privetkakdela');}}
```

Лістинг Г.35 — Текст класу product компоненту main-page

```
export class Product {
  constructor(
```

```
    id_tovar: number,
    image_tovar: string,
    id_category: string
  ) {

  }
}
```

Лістинг Г.36 — Текст програми компоненту модулю client-system

```
import {Component, OnInit} from '@angular/core';
import {CookieService} from 'ngx-cookie-service';

@Component({
  selector: 'app-client-system',
  templateUrl: './client-system.component.html',
  styleUrls: ['./client-system.component.css']
})
export class ClientSystemComponent implements OnInit {

  id = this.cookieService.get('id');

  constructor(private cookieService: CookieService,
              // private systemService: SystemService
  ) {
  }

  ngOnInit(): void {
  }

}
```

Лістинг Г.37 — Текст програми шаблону компоненту модулю client-system

```
<div class="navigation-profile">
  <a [routerLink]="['client', id]" id="nav">Мій профіль</a>
  <a [routerLink]="['/system/add']" id="nav">Додати об'яву</a>
  <a [routerLink]="['client_products']" id="nav">Мої об'яви</a>
  <a [routerLink]="['my-bets']" id="nav">Мої ставки</a>
  <a [routerLink]="['my-wins']" id="nav">Мої виграши</a>
</div>
<router-outlet></router-outlet>
```

Лістинг Г.38 — Текст програми компоненту client-bets

```
import {Component, OnInit} from '@angular/core';
import {CookieService} from 'ngx-cookie-service';
import {ClientPageService} from '../client-page/client-page.service';

import {clientbets} from '../client-page/clientbets';

@Component({
  selector: 'app-client-bets',
  templateUrl: './client-bets.component.html',
  styleUrls: ['../client-products/client-products.component.css']
})
export class ClientBetsComponent implements OnInit {

  prodBets: clientbets[];
  ClientId = this.cookieService.get('id');
  error: string;

  constructor(
    private cookieService: CookieService,
    private clientPageService: ClientPageService) {
  }
```

```
  ngOnInit(): void {
    this.getBets();
  }

  getBets(): void {
    this.clientPageService.getBets(this.ClientId).subscribe(
      (res: clientbets[]) => {
        this.prodBets = res.map(item => new clientbets({...item}));
        console.log(this.prodBets);
      },
      (err) => {
        this.error = err;
      }
    );
  }

}
```

Лістинг Г.39 — Текст програми щаблону компоненту client-bets

```
<div class="tovar" *ngFor="let items of prodBets">
  <div class="tovar__inner">
    <img src="http://kyrsovoi/photos/{{items.img}}.jpg">
    <div class="text">
      <span>{{items.name}}</span>
      <span>Моя ставка: {{items.samaStavka}} грн.</span>
    </div>
  </div>
  <div class="tovar__edit">
    <a [routerLink]="['/system/product', items.id_tovar]">На сторінку товару</a>
  </div>
</div>
```

Лістинг Г.40 — Текст програми класу client компоненту client-page

```
export class Client {
  Email: string;
  Name: string;

  constructor(
    values: object
  ) {
    Object.keys(values).forEach(key => {
      this[key] = values[key];
    });
  }
}
```

Лістинг Г.41 — Текст програми шаблону компоненту client-page

```
<form *ngIf="client">

  <!--   Image and information profile   -->
  <div class="img-info">
    <div class="img-name">
      <img src="http://kyrsovoi/S2.png">
      <div class="name">{{client.Name}}</div>
    </div>
    <div class="my-info">
      <div class="my-info-text">Моя інформація</div>
      <div class="info">
        <div class="email">Email:{{client.Email}}</div>
        <button (click)="LogOut()">вийти</button>
      </div>
    </div>
  </div>
  <!--   Latest 5 rate   -->
```

```
  <div class="latest-rate">
    <div class="text-rate">Останні ставки</div>
    <div class="rates">

      <div class="rate" *ngFor="let items of prodBets">
        <div class="name-rate">{{items.item.name}}</div>
        <div class="status">Моя ставка:{{items.item.samaStavka}} грн.</div>
        <div class="src-page"><a [routerLink]="['/system/product',
items.item.id_tovar]">На сторінку</a></div>
      </div>
    </div>
  </div>
  <!--   Latest 5 win   -->
  <div class="latest-win">
    <div class="text-win">Останні виграші</div>
    <div class="wins">
      <div class="win" *ngFor="let items of prodWins">
        <div class="name-win">{{items.item.name}}</div>
        <div class="src-page">Ціна: {{items.item.samaStavka}}грн.</div>
        <div class="src-page"><a
href="https://www.donationalerts.com/r/olesha2020">Сплатити</a></div>
      </div>
    </div>
  </div>
</form>
```

Лістинг Г.42 — Текст програми компоненту client-page

```
import {Component, OnInit} from '@angular/core';
import {ActivatedRoute} from '@angular/router';
import {ClientPageService} from './client-page.service';
import {Client} from './client';
import {productClientwins} from './product-clientwins';
import {clientbets} from './clientbets';
import {CookieService} from 'ngx-cookie-service';


@Component({
  selector: 'app-client-page',
  templateUrl: './client-page.component.html',
  styleUrls: ['./client-page.component.css']
})
export class ClientPageComponent implements OnInit {

  id: string;
  client: Client;
  prodWins: productClientwins[];
  prodBets: clientbets[];
  error = '';
  success = '';

  constructor(
    private route: ActivatedRoute,
    private clientPageService: ClientPageService,
    private cookieService: CookieService,
  ) {
  }

  ngOnInit(): void {
    this.id = this.route.snapshot.params['id'];
    this.getClient();
    this.getWins();
    this.getBets();
  }

  getClient(): void {
    this.clientPageService.getClient(this.id).subscribe(
```

```
      (res: Client) => {
        console.log(res);
        this.client = new Client({...res});
      },
      (err) => {
        this.error = err;
      }
    );
  }

  getWins(): void {
    this.clientPageService.getWins(this.id).subscribe(
      (res: productClientwins[]) => {
        this.prodWins = res.map(item => new productClientwins({item}));
        console.log(this.prodWins);
      },
      (err) => {
        this.error = err;
      }
    );
  }

  getBets(): void {
    this.clientPageService.getBets(this.id).subscribe(
      (res: clientbets[]) => {
        this.prodBets = res.map(item => new clientbets({item}));
        console.log(this.prodBets);
      },
      (err) => {
        this.error = err;
      }
    );
  }

  LogOut() {
    this.cookieService.delete('user');
    this.cookieService.delete('id');
  }
}
```

Лістинг Г.43 — Текст програми сервісу компоненту client-page

```
import {Injectable} from '@angular/core';
import {Observable, throwError} from 'rxjs';

import {catchError, map} from 'rxjs/operators';
import {HttpClient, HttpErrorResponse} from '@angular/common/http';
import {Client} from './client';
import {productClientwins} from './product-clientwins';
import {clientbets} from './clientbets';

@Injectable({
  providedIn: 'root'
})
export class ClientPageService {
  baseUrl = 'http://kyrsovoi';
  client: Client;
  prodWins: productClientwins[];
  prodBets: clientbets[];

  constructor(private http: HttpClient) {
  }

  getClient(id: string): Observable<Client> {
    const formData = new FormData();
    formData.append('id', id);
    return this.http.post(`${this.baseUrl}/get_client.php`, formData)
```

```
      .pipe(
        map((res) => {
          this.client = res['data'];

          return this.client;
        }),
        catchError(this.handleError));
  }

  getWins(id: string): Observable<productClientwins[]> {
    const formData = new FormData();
    formData.append('id', id);
    return this.http.post(`${this.baseUrl}/get_wins.php`, formData)
      .pipe(
        map((res) => {
          this.prodWins = res['data'];

          return this.prodWins;
        }),
        catchError(this.handleError));
  }

  getBets(id: string): Observable<clientbets[]> {
    const formData = new FormData();
    formData.append('id', id);
    return this.http.post(`${this.baseUrl}/get_bets.php`, formData)
      .pipe(
        map((res) => {
          this.prodBets = res['data'];

          return this.prodBets;
        }),
        catchError(this.handleError));
  }


  private handleError(error: HttpErrorResponse) {
    console.log(error);

    // return an observable with a user friendly message
    return throwError('error'); }}
```

Лістинг Г.44 — Текст програми шаблону компоненту client-products

```
<div class="tovar" *ngFor="let items of cliProds">
  <div class="tovar__inner">
    <img src="http://kyrsovoi/photos/{{items.img}}.jpg">
    <div class="text">
      <span><a [routerLink]="['/system/product',
items.id_tovar]">{{items.name}}</a></span>
      <span *ngIf="7 - items.dayloss > 0">До закінчення ставок: {{7 -
items.dayloss}} {{Days(7 - items.dayloss)}}</span>
      <span *ngIf="7 - items.dayloss <= 0">Ставки на цій товар закінчено</span>
      <span>Остання ставка: {{items.samaStavka}} грн.</span>
    </div>
  </div>
  <div class="tovar__edit">
    <a [routerLink]="['/system/edit', items.id_tovar]">Редагувати</a>
    <a (click)="deleteProduct(items.id_tovar)">Видалити</a>
  </div>
</div>
```

Лістинг Г.45 — Текст програми компоненту client-products

```
import {Component, OnInit} from '@angular/core';
import {CookieService} from 'ngx-cookie-service';
import {ClientProductsService} from './client-products.service';
import {clientProducts} from './clientProducts';
```

```
@Component({
  selector: 'app-client-products',
  templateUrl: './client-products.component.html',
  styleUrls: ['./client-products.component.css']
})
export class ClientProductsComponent implements OnInit {

  error: string;
  ClientId = this.cookieService.get('id');
  cliProds: clientProducts[];

  constructor(
    private cookieService: CookieService,
    private clientProductsService: ClientProductsService
  ) {
  }



  ngOnInit(): void {
    this.getClientProducts();
  }

  getClientProducts(): void {
    this.clientProductsService.getClientProductsS(this.ClientId).subscribe(
      (res: clientProducts[]) => {
        this.cliProds = res.map(item => new clientProducts({...item}));
        console.log(this.cliProds);
      },
      (err) => {
        this.error = err;
      }
    );
  }

  private mess: string;

  Days(day: number): string {
    // tslint:disable-next-line:label-position no-unused-expression

    if (day >= 5) {
      this.mess = 'днів';
    }
    if (day >= 2 && day <= 4) {
      this.mess = 'дні';
    }
    if (day <= 1) {
      this.mess = 'день';
    }
    return this.mess;
  }

  deleteProduct(id_tovar: number) {
    let confirm1 = confirm('Ви дійсно хочете видалити цей товар');
    if (confirm1){
      this.clientProductsService.deleteProduct(id_tovar.toString()).subscribe((id:
string) => {
        this.getClientProducts();
      });
    }
    else{
      console.log('ничего не делаем');
    }
  }
}
```

Лістинг Г.46 — Текст програми сервісу компоненту client-products

```
import {Injectable} from '@angular/core';
import {HttpClient, HttpErrorResponse} from '@angular/common/http';
import {Observable, throwError} from 'rxjs';
import {clientProducts} from './clientProducts';
import {catchError, map} from 'rxjs/operators';

@Injectable({
  providedIn: 'root'
})
export class ClientProductsService {
  baseUrl = 'http://kyrsovoi';
  cliProds: clientProducts[];
  constructor(private http: HttpClient) {
  }

  getClientProductsS(id: string): Observable<clientProducts[]> {
    const formData = new FormData();
    formData.append('id', id);
    return this.http.post(`${this.baseUrl}/get_cliProds.php`, formData)
      .pipe(
        map((res) => {
          this.cliProds = res['data'];

          return this.cliProds;
        }),
        catchError(this.handleError));
  }
  deleteProduct(idProd: string) {
    console.log('ddd');
    const formData = new FormData();
    formData.append('id', idProd);
    return this.http.post(`${this.baseUrl}/delete_product.php`, formData);
  }

  private handleError(error: HttpErrorResponse) {
    console.log(error);

    // return an observable with a user friendly message
    return throwError('error');
  }
}
```

Лістинг Г.47 — Текст програми класу clientProducts компоненту client-products

```
export class clientProducts {
  id_tovar: number;

  Name: string;
  daysLost: string;
  imgAddr: string;
  SamaStavka: number;

  constructor(
    values: object
  ) {
    Object.keys(values).forEach(key => {
      this[key] = values[key];
    });
  }
}
```

Лістинг Г.48 — Текст програми компоненту client-wins

```
import {Component, OnInit} from '@angular/core';
import {ClientPageService} from '../client-page/client-page.service';
import {productClientwins} from '../client-page/product-clientwins';
```

```
import {CookieService} from 'ngx-cookie-service';

@Component({
  selector: 'app-client-wins',
  templateUrl: './client-wins.component.html',
  styleUrls: ['../client-products/client-products.component.css']
})
export class ClientWinsComponent implements OnInit {

  prodWins: productClientwins[];
  ClientId = this.cookieService.get('id');
  error: string;

  constructor(
    private cookieService: CookieService,
    private clientPageService: ClientPageService) {
  }


  ngOnInit(): void {
    this.getWins();
  }

  getWins(): void {
    this.clientPageService.getWins(this.ClientId).subscribe(
      (res: productClientwins[]) => {
        this.prodWins = res.map(item => new productClientwins({...item}));
        console.log(this.prodWins);
      },
      (err) => {
        this.error = err;
      }
    );
  }

}
```

Лістинг Г.49 — Текст програми шаблону компоненту client-wins

```
<div class="tovar" *ngFor="let items of prodWins">
  <div class="tovar__inner">
    <img src="http://kyrsovoi/photos/{{items.img}}.jpg">
    <div class="text">
      <span>{{items.name}}</span>
      <span>До сплати: {{items.samaStavka}} грн.</span>
    </div>
  </div>
  <div class="tovar__edit">
    <a [routerLink]="['/system/product', items.id_tovar]">На сторінку</a>
    <a href="https://www.donationalerts.com/r/olesha2020">Сплатити</a>
  </div>
</div>
```

Лістинг Г.50 — Текст програми гуард сервісу компоненту app

```
import {ActivatedRouteSnapshot, CanActivate, RouterStateSnapshot, UrlTree} from
'@angular/router';
import {Observable} from 'rxjs';
import {AuthService} from './auth.service';
import {Injectable} from '@angular/core';
import {CookieService} from 'ngx-cookie-service';

@Injectable()
export class AuthGuard implements CanActivate {
  constructor(private auth: AuthService,
              private cookieService: CookieService
  ) {
  }
```

```
  // tslint:disable-next-line:max-line-length
  canActivate(route: ActivatedRouteSnapshot, state: RouterStateSnapshot):
Observable<boolean | UrlTree> | Promise<boolean | UrlTree> | boolean | UrlTree {
    return this.auth.isAuth().then((isLoggedIn: boolean) => {
      if (this.cookieService.get('user')) {
        return true;
      } else {
        return false;
      }
    });
        }

}
```

ДОДАТОК Д

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ