МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ім. Ю.М. Потебні
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

КАФЕДРА ЕЛЕКТРОНІКИ, ІНФОРМАЦІЙНИХ СИСТЕМ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Кваліфікаційна робота

перший (бакалаврський)
(рівень вищої освіти)

на тему **Створення веб-сервісу для надання рекламних послуг з використанням фреймворків React та Django**

Виконала: студентка 4 курсу, групи 6.1210-пзс
спеціальності 121 Інженерія програмного забезпечення
(код і назва спеціальності)
освітньої програми Програмне забезпечення систем
(код і назва освітньої програми)
Є. А. Переверзова
(ініціали та прізвище)
Керівник доцент, к.ф.-м.н., доцент каф. ЕІС та ПЗ
І. А. Скрипник
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)
Рецензент
директор ТОВ «Дісітел» П.О. Лютий
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)

Запоріжжя
2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ім. Ю.М. Потебні
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

Кафедра електроніки, інформаційних систем та програмного забезпечення
Рівень вищої освіти _____ перший (бакалаврський)_____
Спеціальність ___121_Інженерія програмного забезпечення_____
(код та назва)
Освітня програма ___Програмне забезпечення систем
(код та назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри____Тетяна КРИТСЬКА_
“ 01 ” березня 2024 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Переверзовій Єві Андріївні
(прізвище, ім’я, по батькові)

1.Тема роботи Створення веб-сервісу для надання рекламних послуг з використанням фреймворків React та Django

керівник роботи Скрипник Ірина Анатоліївна, к.ф.-м.н., доцент
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом ЗНУ від 26.12.2023 № 2215-с
2. Строк подання студентом кваліфікаційної роботи 07.06.2024
3. Вихідні дані бакалаврської роботи
- комплект нормативних документів ;
- технічне завдання до роботи.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
- огляд та збір літератури стосовно теми кваліфікаційної роботи;
- огляд та аналіз існуючих рішень та аналогів;
- дослідження проблеми створення веб-додатків;
- створення програмного продукту та його опис;
- дослідження поставленої проблеми та розробка висновків.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
слайдів презентації

6. Консультанти розділів бакалаврської роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата Завдання прийняв

7. Дата видачі завдання 01.03.2024

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Аналіз предметної області	01.03. - 13.03.24	виконано
2	Формулювання основної задачі дипломної роботи та узгодження її з науковим керівником	14.03.24	виконано
3	Аналіз існуючих методів рішення	15.03 - 19.03.24	виконано
4	Дослідження існуючих засобів для реалізації серверної частини	20.03 - 23.03.24	виконано
5	Огляд технології Django	24.03 - 28.03.24	виконано
6	Дослідження існуючих засобів для реалізації клієнтської частини	29.03 - 31.03.24	виконано
7	Огляд технології React	01.04 - 05.04.24	виконано
8	Проектування архітектури веб-сервісу	06.04 - 13.04.24	виконано
9	Програмна реалізації серверної та клієнтської частин	14.04 - 12.05.24	виконано
10	Перевірка працездатності веб-сервісу	13.05 - 23.05.24	виконано
11	Тестування та виправлення помилок	24.05. - 27.05.24	виконано
12	Оформлення звіту	28.05. - 04.06.24	виконано
13	Оформлення презентації	05.06. - 07.06.24	виконано

Студент _____ Переверзова Є.А.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Скрипник І.А.
(підпис) (прізвище та ініціали)

Нормоконтроль пройдено

Нормоконтролер _____ Скрипник І.А.
(підпис) (прізвище та ініціал

АНОТАЦІЯ

Сторінок: 84

Рисунків: 35

Лістингів: 28

Джерел: 18

Переверзова Є.А. Створення веб-сервісу для надання рекламних послуг з використанням фреймворків React та Django: кваліфікаційна робота бакалавра спеціальності 121 «Інженерія програмного забезпечення» / наук. керівник І.А. Скрипник. Запоріжжя : ЗНУ, 2024. 88с.

Метою кваліфікаційної роботи є розробка зручного веб-сервісу на React та Django з надання рекламних послуг, який забезпечуватиме продуктивну взаємодію між такими групами користувачів, як замовники та виконавці, відповідатиме поставленим вимогам та визначеній архітектурі.

У процесі розробки веб-застосунку досліджено відповідні веб-фреймворки, реляційні системи управління базами даних. Розроблено програмний веб-сервіс, який забезпечує взаємодію між замовниками та виконавцями, надає інтерфейс для роботи модератора та адміністратора сервісу, з відповідними можливостями. Розробка виконана з використанням наступних технологій: Python, Django, SQLite, Node.js, JavaScript, React.

Ключові слова: веб-сервіс, система, замовник, замовлення, виконавець, пропозиція, фреймворк, React, Django.

ABSTRACT

Pages: 84

Drawings: 35

Listings: 28

Sources: 18

Pereverzova Y.A. Creating web-service for providing advertising services using frameworks React and Django: bachelor's thesis in specialty 121 "Software Engineering" / Science. manager I.A. Skrypnyk. Zaporizhzhia : ZNU, 2024. 88p.

The goal of qualification work is create useful web-service with React and Django for providing advertising services, that will provide productive interaction between next groups of users, as customers and performers, will respond to the requirements and determined architecture.

In process of development of web-service researched corresponding web-frameworks, relational database management systems. Created programming web-service, that provides interaction between customers and performers, gives interface for the work of moderator and administrator of service, with corresponding opportunities. The development is made using next technologies: Python, Django, SQLite, Node.js, JavaScript, React.

Keywords: web service, system, customer, order, performer, offer, framework, React, Django.

ЗМІСТ

ВСТУП

Актуальність теми

В сучасному світі реклама відіграє ключову роль в підтримці та просуванні бізнесу. Від залучення нових клієнтів до підвищення уваги до свого бренду – досягти такого результату мають можливість компанії завдяки використанню цього елементу. У продовж останніх кількох років з розвитком технологій почали з'являтися різноманітні та ефективні формати реклами, серед яких більшу популярність здобула саме інтернет-реклама. Одним з її потужних широко використовуваних засобів, який допомагає детальніше донести інформацію про продукт чи послугу до аудиторії, є відеореклама [1].

З розповсюдженням пристроїв, що мають камеру, та загальної доступності глобальної мережі Інтернет, у звичайних людей з'явилася можливість знімати відео контент, зокрема відеорекламу, та розміщувати його на різноманітних веб-ресурсах, таких як соцмережі. В наш час підприємства для створення рекламного контенту залучають професійні рекламні агентства або проводять конкурси, кастинги для заохочення креативного населення з різних куточків світу. Однак, ці варіанти потребують великої кількості витраченого часу та коштів.

Отже, з розвитком технологій та набуттям все більшої затребуваності інтернет-реклами серед інших форм реклами, сучасні компанії для просування свого бізнесу потребують зручний веб-застосунок, який забезпечуватиме їм більш швидкого та зручного отримання рекламного контенту з залученням простого населення, в той час, як надасть можливість звичайним креативним людям реалізувати свій талант. Цей сервіс також заощадить кількість витраченого часу як замовників, так і виконавців.

Мета дослідження

Мета дослідження полягає в розробці зручного веб-сервісу, який забезпечує можливості замовникам залишати замовлення на рекламу, а виконавцям розміщувати свої пропозиції у вигляді завантажених відеороликів.

Завдання дослідження

Завдання дослідження полягає в аналізі засобів, що підходять для реалізації проекту, вивченні React та Django, визначенні переваг поєднання фреймворків React для клієнтської частини та Django для серверної частини.

Об’єкт дослідження

Об’єктом дослідження є процес розробки веб-сервісу для надання рекламних послуг.

Предмет дослідження

Предметом дослідження є технології створення веб-застосунків, аналіз потреб користувачів, визначення технічних рішень для реалізації розроблюваного веб-сервісу.

Методи дослідження

Методами дослідження є огляд літературних джерел, що стосуються технології створення веб-застосунків, аналіз актуальності відеореклами та затребуваність розроблюваного продукту, вивчення інших веб-застосунків.

Практичне значення одержаних результатів

Практичне значення одержаних результатів дослідження полягає у розробці функціоналу, який буде відповідати уподобанням користувачів, матиме більш зручний інтерфейс, порівняно з іншими веб-сервісами для надання рекламних послуг.

Апробація результатів

Результати роботи було представлено на XVII університетській науково-практичній конференції студентів, аспірантів, докторантів і молодих вчених «Молода наука-2024» [18].

Глосарій

Цифровий маркетинг —сукупність стратегій просування продуктів або послуг через цифрові канали для досягнення маркетингових цілей.

Відеомаркетинг — стратегія цифрового маркетингу з використання відеоматеріалів для просування брендів.

Реклама — стратегічний інструмент маркетингу, спрямований на поширення певної інформації про продукт або послуги з метою привернення уваги майбутніх споживачів.

Відео — цифровий запис рухомих яскравих зображень, супроводжуваний звуком, який можна переглядати на деяких пристроях.

Відеореклама — популярна форма реклами, яка є потужним маркетинговим інструментом, що використовує відео для привернення уваги у або послуги можливих споживачів.

Веб-сервіс — компонент інформаційної системи, який надає певні функціональні можливості, доступ до якої можна отримати через Інтернет.

Фреймворк Python — потужний інструмент на Python, який містить різноманітні готові модулі та вбудовані бібліотеки, що значно спрощують розробку серверної частини.

JS-фреймворк — це потужний інструмент на JavaScript, який складається з вже існуючих функцій, класів та іншого, полегшує написання коду складних систем, спрощує взаємодію між веб-додатком та сервером.

1 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

1.1 Огляд літературних джерел

Описувана область сконцентрована на сферах цифрового маркетингу, відеомаркетингу та рекламі, зокрема саме на відеорекламі [2-3].

Перший виток розвитку реклами почався з появи першого телевізійного рекламного відео. З появою всесвітньої мережі Інтернет, відео-реклама почала розвиватися ще стрімкіше та зайняла провідні пропозиції в розвитку та просуванні бізнесу.

У зв’язку з інтенсивним розвитком телекомунікації, Інтернету та персональних гаджетів відео-реклама здобуває все більшу популярність та зачіпає майже кожного, тому на сьогоднішній день вона відіграє важливу роль в цифровому маркетингу та відеомаркетингу.

За прогнозами, які вказані в досліджені [3], витрати на відеорекламу мають сягнути 191,3 мільярда доларів США у 2024 році, а 193,4 мільярда доларів США стягне мобільна відеореклама до 2028 року. Також дослідження [2] вказує на те, що найближчим часом витрати на відеорекламу багатьох різних форматів відео та на різноманітних платформах будуть тільки зростати (див. Рис. 1).

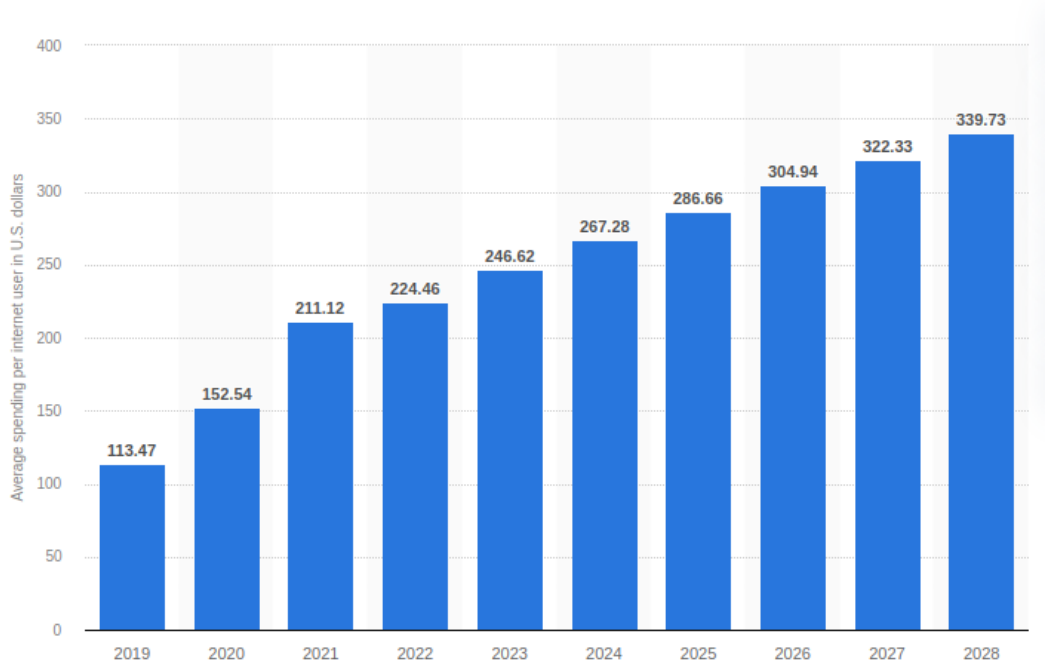

Рисунок *1 — Витрати на відеорекламу на одного користувача Інтернету, млн. дол. США*

1.2 Аналіз програмних продуктів-аналогів

Veed.me

Одним з продуктів-аналогів до розроблюваного веб-сервісу є Veed.me — веб-сервіс, який надає можливості рекламодавцям отримувати відео широкого спектру відео вмісту, що створені професійними відеооператорами та аніматорами. Він був заснований в Ізраїлі, Тель-Авіві в 2012 році та почав стрімко зростати набуваючи все більшої популярності серед брендів та компаній, які потребують відео-контент високої якості [4].

На сьогоднішній день Veed.me функціонує як частина платформи Fiverr, що придбала її у 2017 році. З початку його приєднання до Fiverr, він почав розширювати свої можливості, але продовжує спеціалізуватися на продукції відео.

До основних функцій даної платформи можна віднести відео виробництво різноманітних типів відео та забезпечення допомоги з професійного управління складними проектами від команди підтримки Fiverr.

Veed.me має інтуїтивно зрозумілий користувацький інтерфейс, що дозволяє створювати замовлення та управляти проектами (див. Рис. 2).

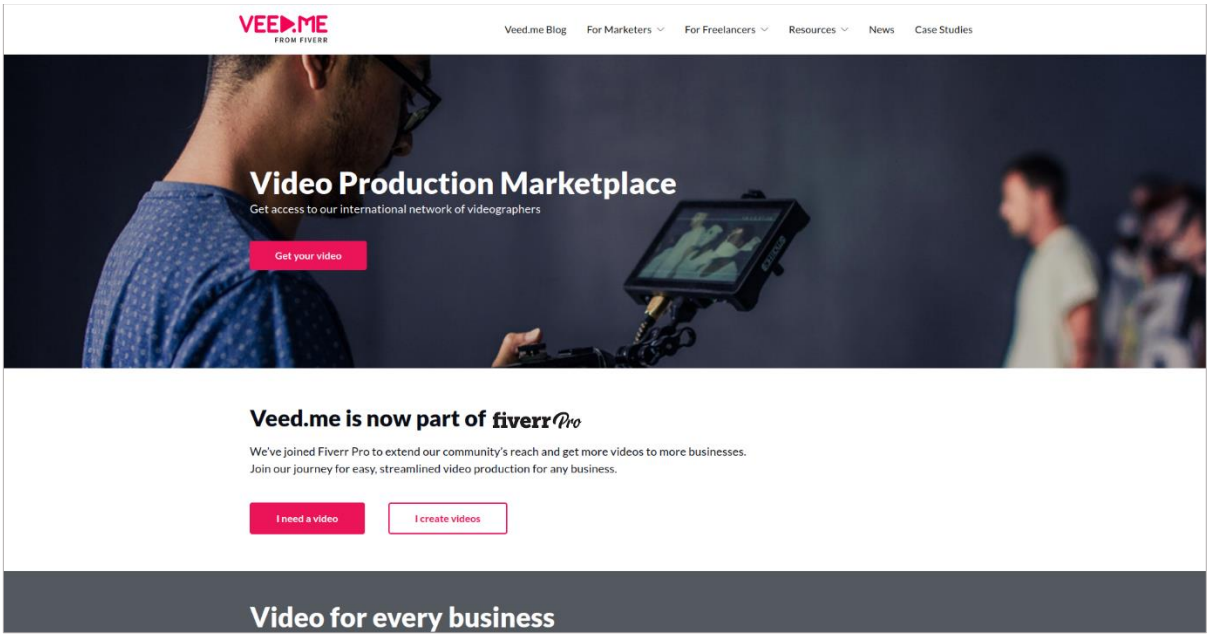

Рисунок *2 — Головна сторінка сайту Veed.me*

Також він забезпечує широкий вибір типів відео для замовлення, каталог прикладів відео-робіт, які допомагають обрати необхідного формату відео, та доступ до міжнародної мережі професійних відеографів з будь-якої точці світу.

Можливим недоліком платформи може бути її залежність від стороннього сервісу — Fiverr, через який проходять всі операції.

90 Seconds

90 Seconds є світовою платформою для створення відео, яка об’єднує бренди з глобальною спільнотою творчих людей з будь-якого куточку світу для розробки високоякісного контенту. Вона надає можливість не тільки створення відео, а також зйомку, монтаж та багато іншого.

З моменту свого заснування у 2010 році, 90 Seconds співпрацює з понад 330 великими брендами, створила понад 47 000 відеороликів та надає доступ до брендів більш ніж 14 000 творцям у понад 110 країнах [5]. Платформа

використовує хмарні технології для надання своїх послуг, що дозволяє легко масштабувати процес виробництва відео.

Основними функціями 90 Seconds є надання інструментів для моніторингу та координації процесу створення відеопродукту з урахуванням індивідуальних потреб замовника, інструменти управління проектами, які надають можливість спілкування між клієнтом та творцем, доступ до глобальної мережі творців з усього світу, можливість замовлення готових пакетів послуг.

Ця платформа має інтуїтивно зрозумілий дизайн (див. Рис. 3) та інтерфейс з різними зручними для користувачів інструментами для управління проектами, для комунікації, та спеціальну активаційну стрічку, яка надає можливість спілкування та оновлення по проекту.

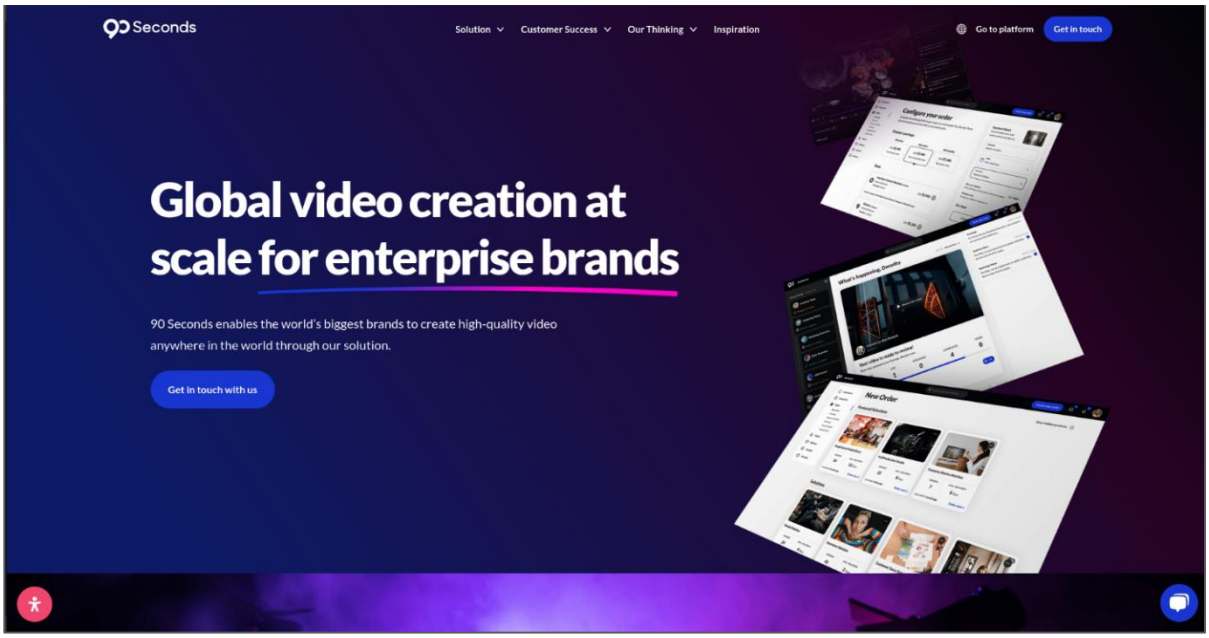

Рисунок *3 — Головна сторінка сайту 90 Seconds*

В ній наявна велика база творців з усього світу, надані можливості координування етапів проекту через єдиний інтерфейс та створювати замовлення як всіх стадій виробництва відео, так і окремих послуг.

90 Seconds має такі слабкі сторони, як складність освоєння платформи новими користувачами через великий набір різноманітних функцій, висока вартість послуг при замовленнях індивідуальних проектів, якість створеного продукту залежить від творця, який виконував замовлення.

Tongal

Раніше відома як Newdio, Tongal — це одна з глобальних платформ для створення контенту, що була заснована в Нью-Йорку в 2009 році [6]. Вона дозволяє різноманітним популярним бізнесам залучати талановитих творців для створення відео різноманітного формату, які відповідатимуть їх потребам.

Станом на сьогодні Tongal є провідною платформою для створення різноманітного контенту, яка активно співпрацює з великими студіями.

Серед основних можливостей цієї платформи можна виділити можливості створення різноманітного контенту на замовлення, залучаючи творців з різних куточків світу, управління проектами за допомогою спеціальних інструменти, які координують роботу між замовниками та креаторами.

Tongal містить простий у використанні інтерфейс (див. Рис. 4) та інструменти для керування та моніторингу розвитку проектів.

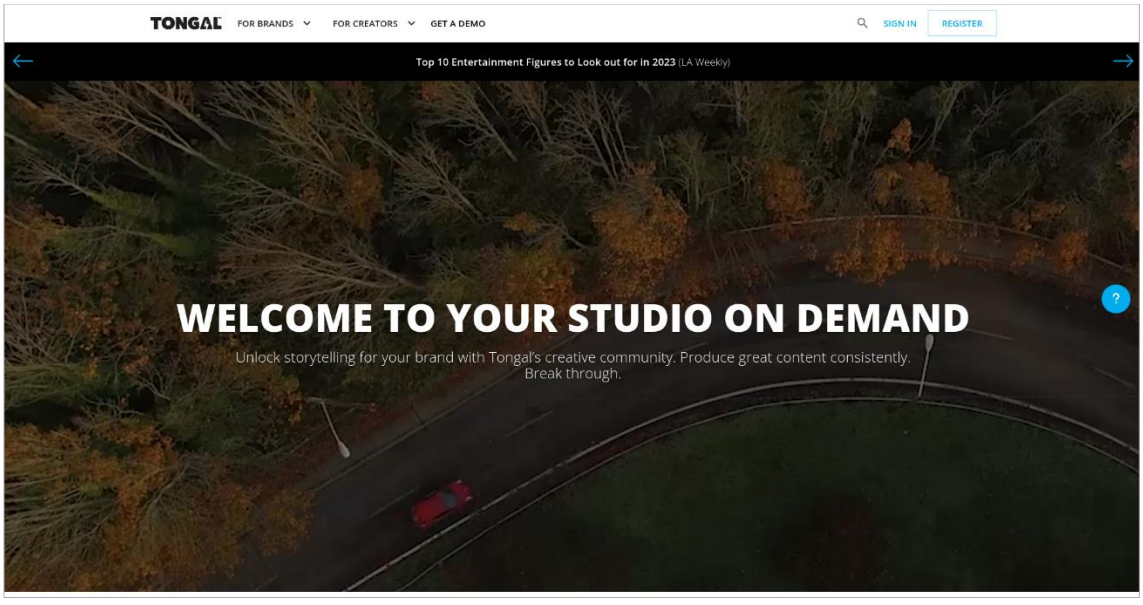

Рисунок *4 — Головна сторінка сайту Tongal*

Ця платформа наділена доступом до великої кількості творчих людей зі світової мережі, можливостями швидкого доступу до креативного контенту, замовлення контенту різноманітного формату.

До слабких сторін Tongal можна віднести різний рівень якості, який залежить від досвіду та навичок творців, ускладнення вибору найкращих пропозицій через велику конкуренцію серед креаторів, можлива висока вартість для невеликих компаній.

1.3 Постановка завдання

Мета кваліфікаційної роботи полягає в розробці зручного веб-сервісу для надання рекламних послуг, який забезпечить можливість безпосередньої взаємодії між замовниками та виконавцями.

Для успішного досягнення мети треба вирішити ряд наступних завдань:

1. Провести аналіз предметної області.
2. Визначити функціональні, нефункціональні та технічні вимоги до системи розроблюваного веб-сервісу.
3. Визначити архітектуру системи та розробити її структуру.
4. Програмно реалізувати серверну частину веб-сервісу, за допомогою технологій Django та SQLite.
5. Розробити клієнтську частину веб-сервісу, використовуючи технологію React.
6. Провести тестування розробленого функціоналу.

2 ДОСЛІДЖЕННЯ ПРОГРАМНИХ ЗАСОБІВ РЕАЛІЗАЦІЇ

Основою в переважної більшості веб-сайтів та веб-додатків є серверна та клієнтська частини.

2.1 Аналіз сучасних технологій для розробки серверної частини

Серверна частина (Backend) є невидимим для користувачів програмним механізмом веб-додатка, який відповідає за створення його основного функціоналу. До її основних функцій відноситься створення основи для сайту, взаємодія з зовнішніми API та сервісами, обробка даних, керування та забезпечення працездатності бази даних , розробка адміністративної частини, інтеграція з різними додатковими системами.

До технологій, що часто використовуються для розробки серверної частини відносяться:

1. Python — це високорівнева інтерпретована мова програмування загального призначення, яка широко використовується в веб-розробці, підтримує функціональне, процедурне та об'єктно-орієнтоване програмування, підходить для машинного навчання.
2. Java — об'єктно-орієнтована мова програмування, яка надає можливість створювати надійні масштабовані додатки, підтримує багатопотоковість, має широке використання розробці корпоративних програм та веб-додатків.
3. Node.js — це серверне середовище виконання, яке забезпечує розробникам можливість використання JavaScript для створення високопродуктивних та масштабованих серверних додатків.

Щоб спростити створення структур проектів, маршрутизації та багато інших функцій, деякі проекти використовують фреймворки Python. Сьогодні лідируючі позиції у розробці серверної частини серед фреймворків на Python займають Django та Flask.

Django

Одним з провідних фреймворків є Django - високорівневий фреймворк на Python з модульною структурою та великою кількістю вбудованих компонентів, має відкритий відкритий вихідний код, розроблений у 2003 році компанією World Company для створення безпечних масштабованих веб-додатків [7-9].

Перевагами Django є:

1. Безпека. Django має багато вбудованих механізмів та засобів безпеки, які сприяють забезпеченню безпеки та роблять додатки більш захищеними.
2. Інтерфейс адміністратора. Django включає в себе потужну вбудовану адміністративну панель, яка автоматично генерується та створює адміністративні модулі та інтерфейс з управління даними, що значно спрощує адміністрування.
3. Швидкий розвиток. В Django вбудовані функції значно прискорюють розробку додатків завдяки багатому набору бібліотек та різноманітних інструментів, таких як ORM, інтерфейс адміністратора, аутентифікація, маршрутизація. В той час як Flask потребує розширення для додаткових функцій.
4. Масштабованість. Django є зручним у використанні для розробки великих проектів, за допомогою своєї структурованості та широкому набору інструментів, він забезпечує високу масштабованість, коли Flask підходить від невеликих до середніх проектів через забезпечення масштабування, яке потребує підключення додаткових налаштувань та розширень.

Цей фреймворк має наступні недоліки:

1. Складність навчання. Через велику кількість функціональності, іноді занадто велику та важку для навігації документацію, цей фреймворк може бути складним для освоєння початківцям та збільшити їх час на навчання.
2. Монолітність. Django являє собою монолітний фреймворк, де частини програми тісно інтегровані, що може бути недоліком для великих проектів, які потребують незалежного масштабування сервісів та більшої гнучкості.

Flask

Flask є легким мікрофреймворком на основі Python з відкритим вихідним кодом, який є досить простим та зручним, використовується розробки як простих, так і складних веб-додатків [7-8, 10].

До переваг Flask можна віднести:

1. Гнучкість. В порівнянні з Django, Flask є досить гнучким фреймворком, який відрізняється своїм мінімалізмом, надає розробникам необхідний набір інструментів, без зайвих функцій, що робить його дуже корисним у використанні в розробці малих та середніх проектів.

2. Легкість навчання та використання. Flask забезпечує для розробників тільки основні функції, відрізняється простою архітектурою та синтаксисом, що робить його досить зручним для використання, легким для налаштування та простим для освоєння, на відмінність від складного для навчання Django через значно більшу кількість вбудованих функцій.

3. Інтеграція. Цей веб-фреймворк має чудове поєднання з іншими інструментами та різними бібліотеками в Python, такими як Jinja2, SQLAlchemy та інші.

До недоліків Flask відносяться:

1. Відсутність вбудованих рішень. Flask не містить багатьох вбудованих інструментів, таких як адмін-панель, аутентифікація, ORM та інші, тому він потребує самостійної розробки власних рішень або підключення вже існуючих сторонніх розширень.

2. Невелика підтримка великих проектів. Flask є менш ефективний в розробці та підтримці великих проектів через мінімальний набір вбудованих структур та інструментів, що може негативно вплинути на їх масштабування.

2.2 Аналіз сучасних реляційних систем управління базами даних

До серверної частини також мають відношення бази даних (БД) — організована сукупність структурованих даних, яка існує протягом тривалого

періоду часу. Вони бувають реляційними - БД які підтримують SQL, зберігаються у форматі суворо структурованих таблиць, які пов'язані між собою; та нереляційними - не підтримують SQL, зберігаються у вигляді даних, які не є чітко структурованими, має динамічну схему.

SQL (Structured query language) — стандартна декларативна мова запитів, яка забезпечує можливіть взаємодіяти користувачу з реляційними базами даних через формування запитів.

Системи управління базами даних (СУБД) є призначений для ефективного управління БД комплекс програмних засобів, який охоплює різні типи баз даних. В наш час дуже розповсюдженими є СУБД які керують саме реляційними базами даних. Серед них високу популярність мають MySQL, PostgreSQL, SQLite.

MySQL

Однією з популярних СУБД є MySQL — відкрита система управління реляційними базами даних, яка розроблена компанією TcX, використовується для розробки різноманітних програм, має можливості роботи з великими БД, складними запитами та іншими функціями [11].

MySQL володіє перевагами, такими як:

1. Швидкість розвитку. MySQL є легконалаштовуємою СУБД, яка має широкий набір інструментів для управління.
2. Масштабованість. MySQL, як і PostgreSQL, підтримує масштабованість великих та складних проектів та веб-додатків.
3. Підтримка складних запитів. Порівняно з SQLite має значно більшу підтримку складних запитів, але, в порівнянні з PostgreSQL, має обмеження з рекурсивними запитами.
4. Продуктивність. Порівняно високу продуктивність для великої кількості веб-додатків з високим навантаженням на читання.

MySQL включає в себе наступні недоліки:

1. Споживання ресурсів. Для високонавантажених проектів та великих БД MySQL споживає багато ресурсів системи.

2. Складність налаштування. MySQL, на відміну від SQLite, вимагає налаштування серверного середовища, що може бути не зручним для малих проектів.

PostgreSQL

PostgreSQL є об'єктно-реляційною СУБД з відкритим вихідним кодом, яка підтримує широкий спектр можливостей роботи з базами даних, розширені модулі та функції для забезпечення доступності даних [12].

PostgreSQL охоплює наступні переваги:

1. Висока підтримка складних запитів. PostgreSQL, на відміну від SQLite та MySQL, має повну підтримку складних запитів.
2. Розширені можливості. PostgreSQL є багатофункціональним, але порівняно з MySQL та SQLite, може потребувати більше налаштувань.
3. Безпека. PostgreSQL має високій рівень безпеки, завдяки широкому набору вбудованих функцій безпеки та багатій кількості розширень для дотакового захисту.
4. Масштабованість. Як і MySQL, PostgreSQL підходить для багатьох різноманітних складних систем.

В PostgreSQL зустрічаються такі недоліки:

1. Складність налаштувань. PostgreSQL, на відміну від MySQL та SQLite, потребує більше налаштувань через наявність великої кількості конфігураційних параметрів.
2. Сумісність. В порівнянні з MySQL, перехід з інших систем баз даних на PostgreSQL може бути складним через відмінності в функціях та SQL-синтаксисі кожної з систем.

SQLite

SQLite — компактна серверна система управління базами даних, забезпечує створення та зберігання даних у вигляді одного файлу, без необхідності налаштування та запуску серверного середовища [13].

SQLite характеризується перевагами, зокрема:

1. Швидкий розвиток. SQLite є більш простим для навчання, ніж PostgreSQL та MySQL, та не потребує додаткових налаштувань.
2. Продуктивність. SQLite має дуже високу продуктивність для невеликих проектів.
3. Портативність та сумісність. SQLite є більш портативним та сумісним з багатьма операційними системами та мовами програмування, ніж інші розглянуті СУБД.
4. Вбудованість. SQLite ідеально підходить для вбудованих застосунків, на відміну від MySQL та PostgreSQL, завдяки невеликому розміру та легкості.

Недоліки SQLite полягають у:

1. Безпека. Відносно інших СУБД, SQLite є менш безпечним через відсутність серверного середовища.
2. Обмежена підтримка складних запитів. SQLite, порівняно з PostgreSQL та MySQL, не призначений для виконання складних запитів.
3. Масштабованість. SQLite, в порівнянні з PostgreSQL та MySQL, не призначений для масштабованих та складних проектів.

2.3 Аналіз сучасних технологій розробки клієнтської частини

Клієнтська частина (Frontend) є публічним компонентом програмного забезпечення, який відображається на веб-сторінці та взаємодіє з користувачем напряму. ЇЇ основними функціями є створення та відображення користувацького інтерфейсу, обробка дій користувача, виконання запитів на веб-сервер та обробка відповідей на ці запити.

Головними технологіями розробки клієнтської частини є:

1. HTML (HyperText Markup Language) — це стандартна мова гіпертекстової розмітки, що визначає структуру веб-документів та відображення вмісту у браузері, визначають різні елементи за допомогою тегів та атрибутів,.

2. CSS (Cascading Style Sheets) — мова стилізації, яка впливає на відображення елементів з встановленими стилями на веб-сторінці, визначаючи її зовнішній вигляд.

3. JavaScript — це одна з найпоширеніших високорівневих мов програмування, що використовується для динамічної генерації веб-сторінок, реалізації інтерактивної взаємодії з користувачем, дозволяє створювати різноманітні ефекти на сторінках.

Для полегшення роботи над створенням інтерфейсу користувача (GUI) також можуть використовуватися такі технології GUI бібліотеки, як Bootstrap, Reactstrap та інші.

Щоб прискорити створення стандартних елементів користувача у більшості проектів використовуються JS-фреймворки. На сьогоднішній день досить поширеними JS-фреймворками є React, Angular та Vue.

React

Найпоширенішою бібліотекою на Java Script через легкість створення веб-додатків [14] є React — широко функціональна JS - бібліотека з відкритим кодом для розробки інтерфейсів користувача на основі компонентів, яка була розроблена компанією Facebook у 2013 році [15], використовується для розробки широкого спектра веб-проектів.

Серед переваг React можна виділити:

1. Модульність. React є досить гнучким інструментом, який завдяки модульному підходу надає розробникам можливість використовувати сторонні бібліотеки для роботи з багатьма функціями, порівняно з Angular.

2. Компонентний підхід. React надає можливість як самостійно створювати взаємодіючі компоненти, так і використовувати вже існуючі, дозволяючи створювати зручні інтерфейси з використанням повторюваного коду.

3. Продуктивність. Використання Virtual DOM надає React можливість ефективного оновлення інтерфейсу, тим самим підвищуючи продуктивність веб-додатків.

4. JSX. React містить розширення синтаксису JSX, яке поєднує JavaScript та HTML, є дуже зручним у створені інтуїтивно зрозумілого коду, але він може бути незвичним та заплутаним для початкових розробників.

Недоліками даного фреймворку є:

1. Швидкість змін. Через швидкий розвиток, React може отримувати меншу підтримку через часті оновлення.

2. Відсутність вбудованих рішень. React, на відміну від Angular та Vue.js, не містить вбудованих рішень, тому для вирішення деяких задач він потребує підключення сторонніх бібліотек, таких як Redux, React Router та інші, або додаткові налаштування.

Vue.js

Vue.js являє собою прогресивний зовнішній фреймворк, який використовується для створення користувацьких інтерфейсів, динамічних та односторінкових веб-додатків, за допомогою компонентного підходу, є легким для вивчення та має просте API [16].

Переваги Vue.js включають:

1. Простота навчання та використання. Vue.js, серед фреймворків React та Angular, має простіший та більш інтуїтивно-зрозумілий синтаксис, що робить його легким для освоєння.

2. Вбудовані рішення. Vue.js більш зручний для швидкого старту, ніж Angular та React, завдяки наявності вбудованих рішень, таких як Vuex, Vue Router та інших, що значно спрощують розробку.

3. Продуктивність. Через використання Virtual DOM, Vue.js зменшує навантаження на браузер та сприяє їх швидкому завантаженню сторінок та оновленню веб-інтерфейсу.

4. Реактивність. Vue.js містить потужну систему реактивності даних, яка автоматично запускає оновлення інтерфейсу та відображає на ньому зміни, що спрощує створення динамічних веб-додатків.

До недоліків Vue.js можна віднести:

1. Невелика екосистема. Екосистема Vue.js є меншою за екосистеми Angular та React, що, в деяких випадках, ускладнює пошук рішення для деяких завдань у великих проектах.

Angular

Angular —потужний веб-фреймворк на TypeScript, розроблений компанією Google, який має відкритий вихідний код та використовується для розробки різноманітних односторінкових та динамічних додатків [17].

До переваг Angular можна віднести :

1. TypeScript: Angular, на відміну від React та Vue.js, побудований на TypeScript, який додає статичну типізацію та багато корисних функціональних можливостей, які допомагають виявляти помилки.

2. Повна функціональність. Angular пропонує повний набір необхідних інструментів для реалізації проектів.

3. Колективна розробка. Angular, завдяки наявності повного набору необхідних інструментів та структурованості, добре підходить для колективної розробки корпоративних проектів.

4. Тестування. Angular, відносно React та Vue.js, не потребує додаткових інструментів та налаштувань тестування, завдяки вбудованій підтримці для тестувань.

Недоліками Angular є:

1. Складність навчання. Порівняно з Vue.js та React, Angular є складнішим для навчання через свою багатофункціональність, велику кількість інструментів, складний синтаксис та вбудований TypeScript.

2. Гнучкість. Angular, у відмінності від інших розглянутих фреймворків, має строго структуровану архітектуру, що зменшує може перешкоджати розробникам адаптуватися до деяких специфічних вимог проекту.

2.4 Аналіз використання комбінації обраних технологій розробки

Для створення зручного та легкозрозумілого для нових користувачів інтерфейсу у клієнтській частині, була обрана одна з популярних відкритих бібліотек для веб-інтерфейсів на JavaScript — React.

Для взаємодії з базами даних та створення різних типів користувачів, адміністративної панелі у серверній частини, був обраний популярний високорівневий веб-фреймворк на Python — Django, а для швидкого початку розробки та обробки даних полегшена реляційна система керування базами даних — SQLite.

Використання комбінації цих інструментів нададуть створеній системі наступні переваги:

1. Для клієнтів Django забезпечує безпеку даних та гарантію стабільної роботи сервісу, в той час як React забезпечує швидке завантаження компонентів на сторінці, а SQLite швидкий доступ до даних та пошук у базі даних.

2. Для розробників цими перевагами в Django є автоматична генерація створених на Python моделей в SQL-запити, проста взаємодія з базою даних та її зручна міграція, вбудований адміністративний розділ, який дозволяє керувати користувачами та групами користувачів, зв'язками між моделями, виконувати операції з реляційними даними через веб-інтерфейс. В React це можливість написання своїх компонентів та їх багаторазове використання або використання вже існуючих різноманітних компонентів, можливість розміщувати компоненти в окремих файлах, що робить код більш лаконічним та легкочитаємим. В SQLite це те, що він не потребує складної конфігурації та при необхідності може легко бути замінений на більш потужне рішення.

3 РОЗРОБКА ВЕБ-СЕРВІСУ ДЛЯ НАДАННЯ РЕКЛАМНИХ ПОСЛУГ

3.1 Опис предметної області

Предметна область кваліфікаційної роботи містить створення веб-сервісу з надання рекламних послуг та охоплює знання програмування, інструментів та засобів веб-розробки.

Мета розробленого веб-сервісу полягає в наданні достатніх можливостей для ефективної взаємодії між замовниками та виконавцями, забезпечені для них зручного та інтуїтивно зрозумілого користувацького інтерфейсу.

Дана робота зосереджена на розробці веб-сервісу на основі Django та SQLite для серверної частини, React для клієнтської частини, спрямована на розширення знань в створенні веб-застосунків.

В ході виконання дипломної роботи будуть розглянуті загальні принципи розробки веб-додатків, як функціональні та нефункціональні вимоги, архітектура, програмна реалізація та інше.

3.2 Вимоги до системи

При розробці будь-якого проекту важливо мати опис вимог до системи, які визначають основні аспекти розробки веб-сервісу, необхідні для успішного завершення проекту, створення безпечної, надійної та зручної для використання системи, яка задовольняє потребам користувачів.

Функціональні вимоги:

1. Типи користувачів:
    - Адміністратор. Максимальні права.
    - Модератор. Право на перегляд та зміну статусу пропозицій/замовлень.

- Замовник. Права на додавання, редагування та видалення замовлень, зміна статусу, редагування свого профілю.
- Виконавець. Права на додавання, редагування та видалення пропозицій, зміна статусу, редагування свого профілю.

2. Можливості для будь-якого типу користувача:
    - Реєстрація та авторизація:
        - Реєстрація. Система має надати можливість реєстрації для незареєстрованих користувачів.
        - Вхід до системи. Користувачам важливо мати можливість на вхід до своїх облікових записів.
        - Вихід з системи. Користувачам обов'язково повинна бути надана можливість виходити зі своїх облікових записів.
        - Перегляд профілю. Має бути надана можливість для зареєстрованих користувачів переглядати їх профілі та редагувати деяку інформацію.
    - Перегляд загальнодоступних сторінок. Користувачі, незалежно від стану їх авторизації, повинні мати можливість переглядати загальнодоступні сторінки.
    - Фільтрація замовлень. Система має надати можливість будь-яким з користувачів фільтрувати замовлення за деякими частинами його вмісту.
3. Можливості для типу користувача Адміністратор:
    - Створення модератора. Адміністратору необхідно можливість створювати облікові записи для модераторів.
    - Слідкування за роботою модераторів. Адміністратор повинен мати можливість бачити дій, які робить модератор.
    - Активація/деактивація користувачів. Користувачу з роллю адміністратор має мати можливість активізувати або деактивувати будь-якого з користувачів.

4. Можливості для типу користувача Модератор:
    - Перегляд замовлень/пропозицій. Модератору повинна бути надана можливість переглядати замовлення та пропозиції, які за ним закріплені.
    - Зміна статусу замовлення/пропозиції. Користувачу з роллю модератора повинно бути надано можливість підтверджувати або відхиляти закріплені до нього замовлення/пропозиції.
5. Можливості для типу користувача Замовник:
    - Дії стосовно замовлення. Замовник повинен мати можливості виконувати наступні дії стосовно замовлення:
        - Створення замовлення. Для користувача з роллю замовника повинна існувати можливість створити замовлення з відповідної сторінки.
        - Редагування замовлення. Замовнику слід забезпечити можливість редагування замовлення.
        - Видалення замовлення. Рекомендовано надати можливість видалення замовлення.
    - Перегляд пропозицій. обов'язково наданою можливістю для замовника є перегляд підтверджених модератором пропозицій під його підтвердженими замовленнями.
    - Зміна статусу замовлення. Замовнику цілком доцільно мати можливість змінювати статус під замовлення.
6. Можливості для типу користувача Виконавець:
    - Дії стосовно замовлення. Виконавець повинен мати можливості виконувати наступні дії стосовно пропозиції:
        - Створення пропозиції. Для користувача з роллю виконавця повинна існувати можливість створити пропозицію під цікавим для нього замовленням.

        - Редагування замовлення. Виконавцю слід забезпечити можливість редагування пропозиції.
        - Видалення замовлення. Рекомендовано надати можливість видалення замовлення.
    - Зміна статусу пропозиції. Виконавцю цілком доцільно мати можливість

Нефункціональні вимоги:

1. Безпека. Система повинна бути захищеною від доступу до інформації тим, хто немає достатніх прав.
2. Користувацький інтерфейс. Інтерфейс користувача повинен бути інтуїтивно зрозумілим, зручним та забезпечувати усі необхідні користувачу функції.
3. Продуктивність. Час відповіді на дію користувача повинен бути прийнятним та не створювати дискомфорт користувачу.
4. Сумісність. Система повинна мати можливість розгортатися на Ubuntu та Windows, працювати стабільно з усіма найпопулярнішими версіями браузерів.

3.3 Архітектура системи

Веб-сервіс з надання рекламних послуг містить дві основні складові частини:

1. Backend відповідає за додавання, модифікацію, вибірку, видалення даних за критеріями з бази даних. Забезпечує безпеку сервісу через аутентифікацію, авторизацію та перевірку доступу користувачів. Виконує деякі логіки, такі як пошук, фільтрація. Надає відповідний інтерфейс, який забезпечує користувачам доступ взаємодії з даними через клієнтську частину, через API.
2. Frontend є відповідальним за інтуїтивно зрозумілий користувацький інтерфейс, взаємодію з користувачем через цей інтерфейс, створення та

відправку запитів до сервера, отримання відповідей на них у вигляді даних, їх відображення.

Клієнтська частина є доповненням до серверної частини, що, в результаті, утворює повноцінну функціональну систему розроблюваного веб-сервісу.

3.4 Програмна реалізація веб-сервісу

Для програмної реалізації клієнтської частини проекту використовується React, для серверної частини - Django та SQLite.

3.4.1 Реалізація серверної частини

Першим етапом розробки серверної частини є встановлення Django та додаткових модулів, таких як rest_framework, та інших, створення Django проєкту website (див. Рис. 5) за допомогою команди startproject.

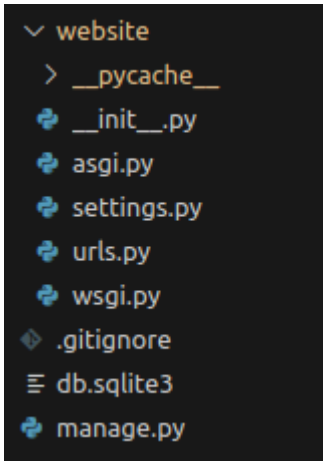

Рисунок 5 — *Директорії проекту website*

Далі йде створення інфраструктури проєкту, необхідної для побудови додатку arena (див. Рис. 6) за допомогою команди startapp.

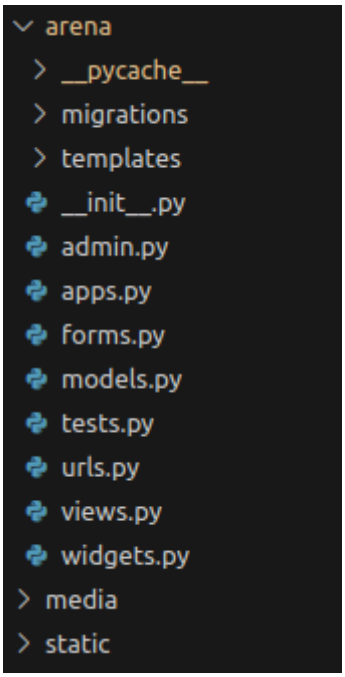

Рисунок *6 — Директорії додатку arena*

При проектуванні бази даних, другим етапом є створення моделей (див. Лістинг 1, Лістинг 2) в Python коді файлу models.py, відповідно до майбутньої схеми бази даних.

Лістинг *1 Модель для створення таблиці arena_advertorder*

```
class AdvertOrder(models.Model):
    STATUS_CHOICES = (
        ('draft', 'Чернетка'),
        ('publish', 'Опублікувати'),
        ('moderating', 'На модерації'),
        ('rejected', 'Відхилено'),
        ('confirmed', 'Підтверджено'),
        ('suspended', 'Призупинити'),
    )
        title = models.CharField(max_length=250,
verbose_name="Заголовок")
    author = models.ForeignKey(
        User, related_name='+', on_delete=models.CASCADE,
verbose_name="Автор")
```

```
        moderator = models.ForeignKey(
            User, related_name='+', on_delete=models.CASCADE,
verbose_name="Модератор")
        body = models.TextField(verbose_name="Опис")
        moderated = models.DateTimeField(
            default=timezone.now, verbose_name="Модеровано")
        publish = models.DateTimeField(
            auto_now_add=True, verbose_name="Опубліковано")
        created = models.DateTimeField(auto_now_add=True,
verbose_name="Створено")
        updated = models.DateTimeField(auto_now=True,
verbose_name="Оновлено")
        status = models.CharField(
            max_length=10, choices=STATUS_CHOICES,
default='draft', verbose_name="Статус")
        class Meta:
            verbose_name = 'Замовлення'
            verbose_name_plural = 'Замовлення'
        def __str__(self):
            return "Замовлення " + str(self.pk) + ": " +
self.title)
```

Модель AdvertOrder має наступні поля:

1. title — заголовок замовлення.
2. author — користувач, який створив замовлення.
3. moderator — модератор, закріплений за замовленням.
4. body — зміст замовлення.
5. moderated — дата та час, коли замовлення було модеровано.
6. publish — дата та час, коли замовлення було опубліковано.
7. created — дата та час, коли замовлення було створено.
8. updated — дата та час, коли замовлення було оновлено.
9. status — статус замовлення.

Лістинг *2 Модель для створення таблиці arena_advertoffer*

```
class AdvertOffer(models.Model):
    STATUS_CHOICES = (
        ('draft', 'Чернетка'),
        ('publish', 'Опублікувати'),
        ('moderating', 'На модерації'),
        ('rejected', 'Відхилено'),
        ('confirmed', 'Підтверджено'),  # success
    )
    order = models.ForeignKey(AdvertOrder, default=0,
on_delete=models.CASCADE, verbose_name="Замовлення")
    author = models.ForeignKey(
        User, related_name='+', on_delete=models.CASCADE,
verbose_name="Автор")
    moderator = models.ForeignKey(
        User, related_name='+', on_delete=models.CASCADE,
verbose_name="Модератор")
    body = models.TextField(verbose_name="Коментар")
    moderated = models.DateTimeField(
        default=timezone.now, verbose_name="Модеровано")
    video = models.CharField(max_length=20, default='',
verbose_name="Відео")
    publish = models.DateTimeField(
        default=timezone.now, verbose_name="Опубліковано")
    created = models.DateTimeField(auto_now_add=True,
verbose_name="Створено")
    updated = models.DateTimeField(auto_now=True,
verbose_name="Оновлено")
    status = models.CharField(
        max_length=10, choices=STATUS_CHOICES,
default='draft', verbose_name="Статус")
    selected = models.IntegerField(default=0,
verbose_name="Обрано")
    class Meta:
        verbose_name = 'Пропозиція'
```

```
            verbose_name_plural = 'Пропозиції'
        def __str__(self):
            return "Пропозиція " + str(self.pk) + ": " +
re.sub('<[^<]+?>', '', self.body)
```

Модель AdvertOffer містить поля:

1. order — замовлення, для якого створено пропозицію.
2. author — користувач, який створив пропозицію.
3. moderator — модератор, закріплений за пропозицією.
4. video — відео-пропозиція.
5. body — коментар до завантаженого відео-пропозиції.
6. moderated — дата та час, коли пропозицію було модеровано.
7. publish — дата та час, коли пропозицію було опубліковано.
8. created — дата та час, коли пропозицію було створено.
9. updated — дата та час, коли пропозицію було оновлено.
10. status — статус пропозиції.
11. selected — прапор, що означає чи було обрано пропозицію замовником, чи ні.

Наступним етапом є створення міграції та її застосування до бази даних за допомогою відповідних команд Python:

1. makemigrations — команда створення міграції.
2. migrate - синхронізація бази даних з моделлю.

В результаті були автоматично створені таблиці, які забезпечують функціонування адміністративної та фронтальної частин. Наприклад: деякі з них використовуються додатково для розмежування доступу користувачів різних типів. Схема бази даних зображена на рисунку 7.

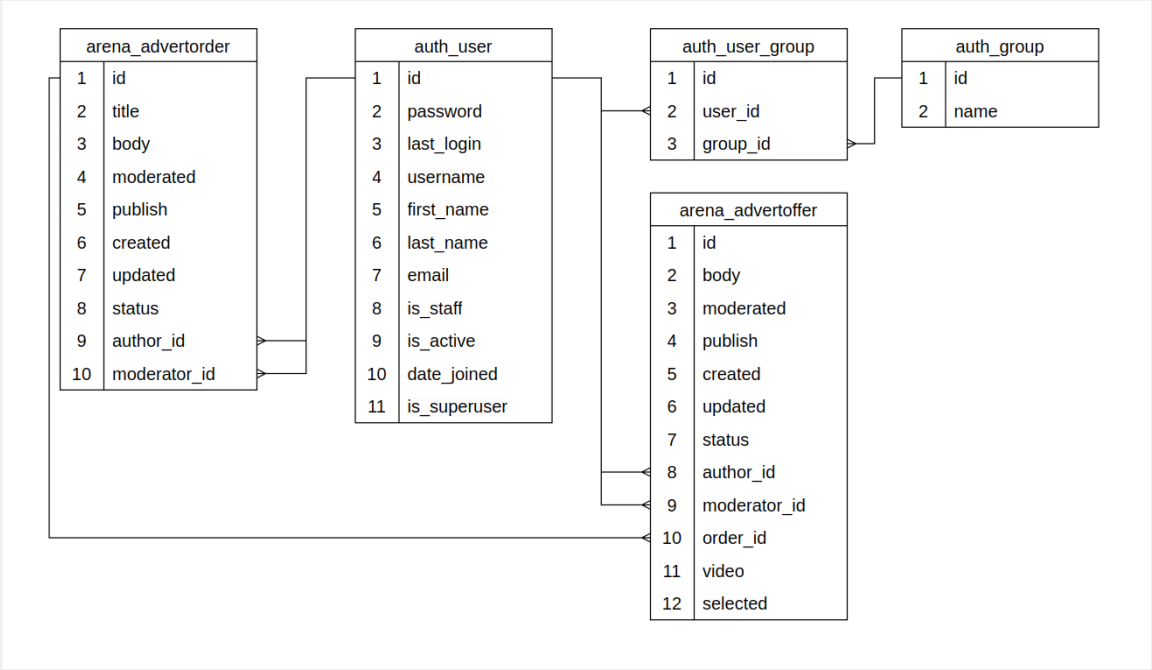

Рисунок *7* — *Схема бази даних веб-сервісу*

За допомогою команди createsuperuser створюється адміністратор системи з повноваженнями згідно рисунку 8.

Рисунок *8* — *Діаграма головних прецедентів для адміністратора*

Далі адміністратор в адмін-панелі створює наступні групи користувачів:

1. moderator (Модератор) — персонал, який має доступ з адмін-панелі див. Рис. 9):

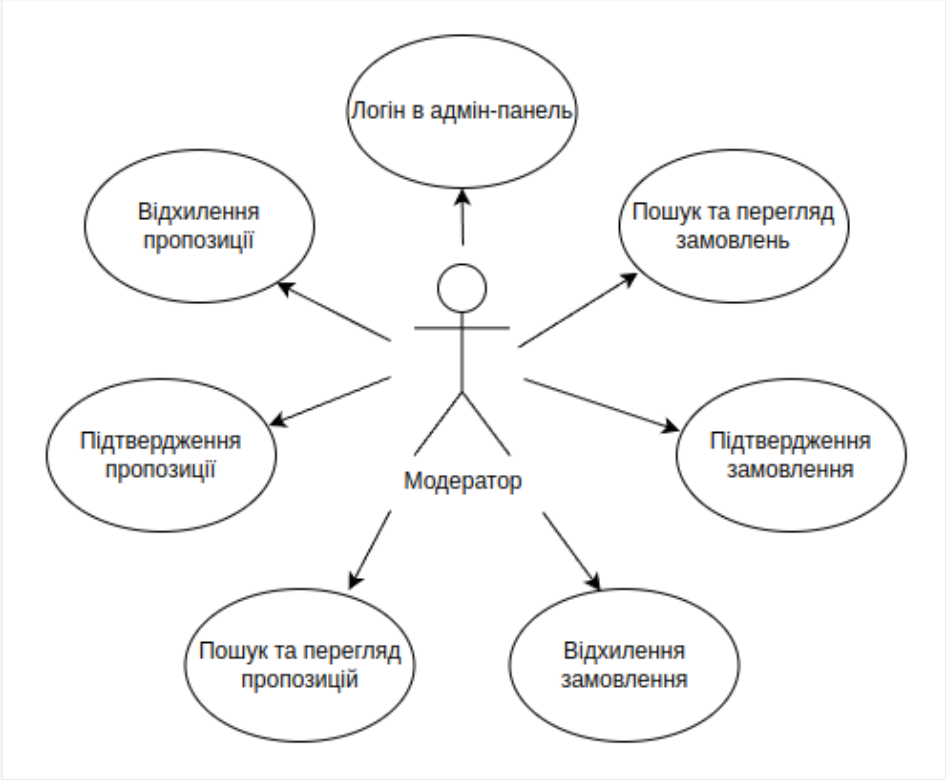

Рисунок 9 — *Діаграма прецедентів для модератора*

2. customer (Замовник) — користувачі, які мають доступ до фронтальної частини див. Рис. 10):

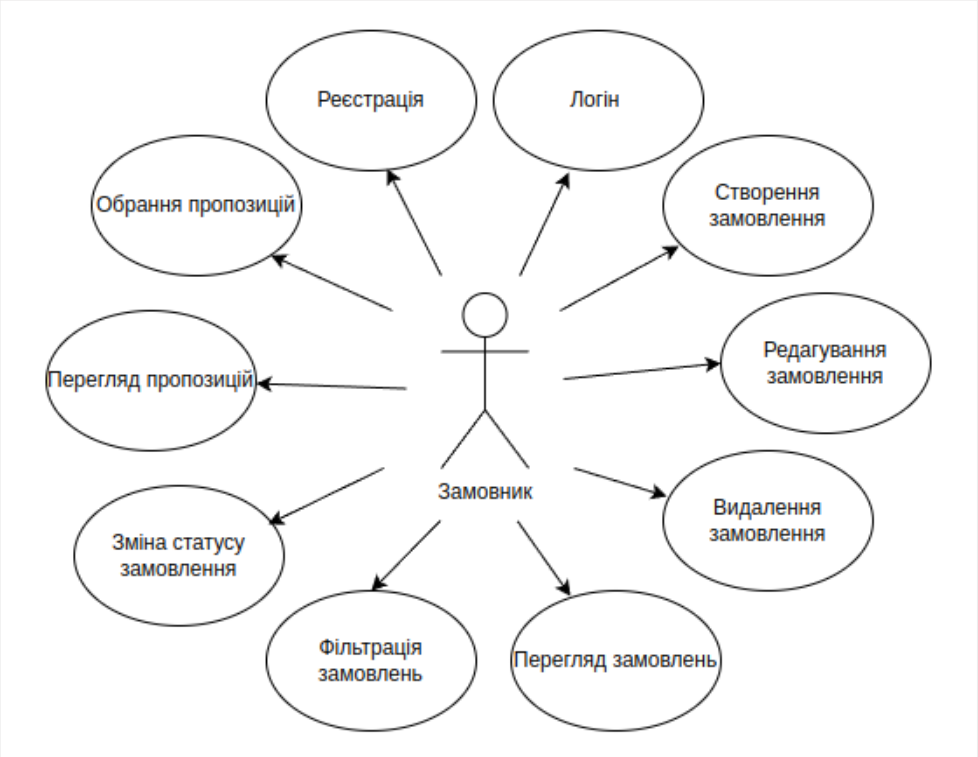

Рисунок *10 — Діаграма прецедентів для замовника*

3. performer (Виконавець) — користувачі, які мають доступ до фронтальної частини (див. Рис. 11):

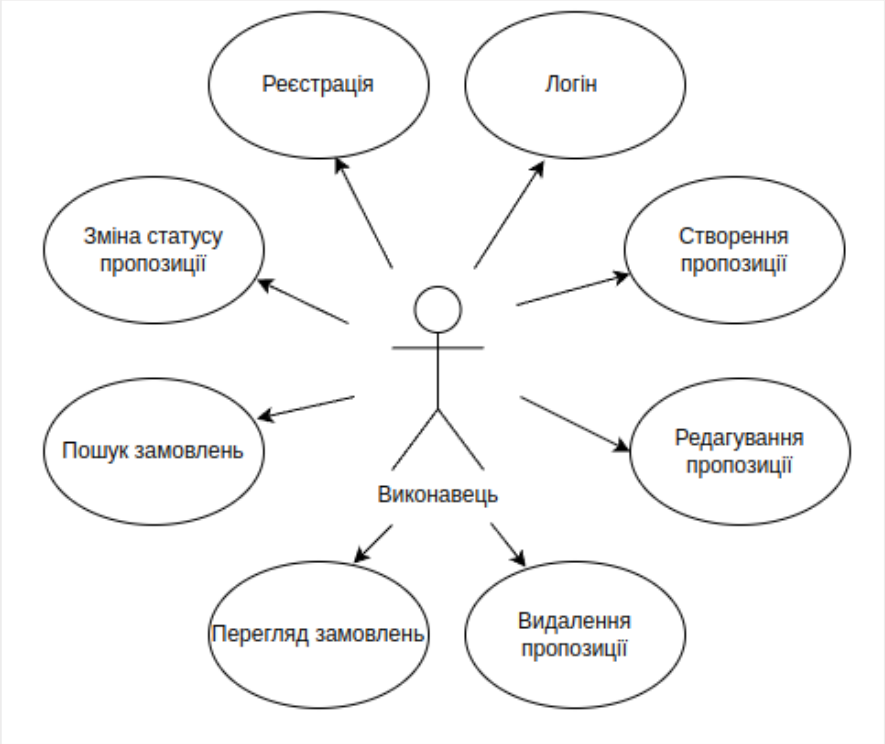

Рисунок *11 — Діаграма прецедентів для виконавця*

В лістингу 3 наведено функцію реєстрації, де відбувається перевірка правильності введення даних та наявності користувача з таким логіном, користувач додається в відповідну до обраної ролі групу.

Лістинг *3 Функція реєстрації*

```
def registration(request):
    if request.user.is_authenticated:
        return ErrResponse(
            request, "Ви вже увійшли як зареєстрований користувач, спочатку вийдіть перед реєстрацією"
        )
    username = request.POST["username"]
    first_name = request.POST["first_name"]
    last_name = request.POST["last_name"]
    email = request.POST["email"]
    password = request.POST["password"]
    repeatpassword = request.POST["repeatpassword"]
    gridRadios = request.POST["gridRadios"]
    try:
        if password != repeatpassword:
            raise Exception("Повтор пароля не співпадає з введеним паролем")
        try:
            User.objects.get(username=username)
            error = True
        except:
            error = False
        if error == True:
            raise Exception(
                "Таке ім'я користувача вже використовується, виберіть будь ласка інше ім'я"
            )
```

```
            user = User.objects.create_user(username, email,
password)
            user.first_name = first_name
            user.last_name = last_name
            if gridRadios == "customer":
                user.groups.add(1)
            else:
                user.groups.add(2)
            user.save()
            return SuccessResponse(request, {})
        except Exception as ex:
            return ErrResponse(request, str(ex))
```

Авторизація для користувачів, перевірка наявності облікового запису та створення сесії представлені в лістингу 4.

Лістинг *4 Функція авторизації*

```
def loginUser(request):
    try:
        json_data = json.loads(request.body)
        user = authenticate(
            request,
            username=json_data["loginUsername"],
            password=json_data["loginPassword"],
        )
        if user is None:
            raise Exception("Користувача не знайдено")
        userType = getUserType(user)
        if userType == "":
            raise Exception("Невірна група користувача")
        login(request, user)
        userData = {
            "id": user.id,
            "username": user.username,
```

```
            "first_name": user.first_name,
            "last_name": user.last_name,
            "user_type": userType,
        }
        return SuccessResponse(request, userData)
    except Exception as ex:
        return ErrResponse(request, str(ex))
```

Отримання певних даних про аутентифікованого користувача наведено у лістингу 5.

Лістинг *5 Функція отримання даних користувача*

```
def getUser(request):
    userType = getUserType(request.user)
    if userType == "":
        return NotAuthedResponse(request)
    userData = {
        "id": request.user.id,
        "username": request.user.username,
        "first_name": request.user.first_name,
        "last_name": request.user.last_name,
        "user_type": userType,
    }
    return SuccessResponse(request, userData)
```

Оновлення таких даних користувача, як ім’я та прізвище, у разі їх зміни, виконується у функції updateProfile (див. Лістинг 6).

Лістинг *6 Редагування даних користувача*

```
def updateProfile(request):
    userType = getUserType(request.user)
    if userType == "":
        return NotAuthedResponse(request)
```

```
        first_name = request.POST["first_name"]
        last_name = request.POST["last_name"]
        if (not first_name or not last_name):
            return ErrResponse(request, "Введіть коректні Ім`я та
Прізвище")
        request.user.first_name = first_name
        request.user.last_name = last_name
        request.user.save()
        userData = {
            "id": request.user.id,
            "username": request.user.username,
            "first_name": request.user.first_name,
            "last_name": request.user.last_name,
            "user_type": userType,
        }
        return SuccessResponse(request, userData)
```

Вихід з системи та завершення сесії здійснюються за допомогою функції logoutUser (див. Лістинг 7).

Лістинг *7 Функція виходу з системи (завершення сесії) користувача*

```
def logoutUser(request):
    logout(request)
    return SuccessResponse(request, {})
```

В функції saveOrder (див. Лістинг 8) виконується додавання нового або оновлення даних вже існуючого замовлення, відбувається перевірка наявності прав на виконання цієї дії користувачем. При додаванні нового замовлення, для нього підбирається модератор викликом функції getModeratorId.

Лістинг *8 Функція збереження даних замовлення*

```
def saveOrder(request):
    userType = getUserType(request.user)
```

```
        try:
            if (userType != "customer"):
                raise Exception(errorNotAllowed)
            mode = request.POST["mode"]
            title = request.POST["title"]
            body = request.POST["text"]
            if mode == "add":
                AdvertOrder.objects.create(
                    title=title,                        body=body,
author_id=request.user.id, moderator_id=getModeratorId()
                )
            else:
                order_id = request.POST["id"]
                order_instance                                  =
AdvertOrder.objects.get(id=order_id)
                if          (order_instance.author_id          !=
request.user.id):
                    raise Exception(errorNotAllowed)
                order_instance.title = title
                order_instance.body = body
                order_instance.save()
            return SuccessResponse(request, {})
        except Exception as ex:
            return ErrResponse(request, str(ex))
```

Отримання списку усіх замовлень відповідно до фільтру, поточної сторінки та прав користувача для відображення їх на головній сторінці. Для незареєстрованого користувача та виконавця повертаються тільки замовлення, затверджені модератором, а для замовника додатково - всі ті, що належать йому (див. Лістинг 9).

Лістинг *9 Функція отримання списку замовлень*

```
def orders(request):
    page_number = request.GET.get("page")
```

```
        pp = request.GET.get("pp")
        title = request.GET.get("title").strip()
        body = request.GET.get("body").strip()
        my_orders = True if request.GET.get(
            "my_orders").strip() == "true" else False
        my_offers = True if request.GET.get(
            "my_offers").strip() == "true" else False
        userType = getUserType(request.user)
        if userType == "customer":
            if my_orders:
                criteria = Q(author=request.user.id)
            else:
                criteria      =      Q(status="confirmed")      |
Q(author=request.user.id)
        else:
            criteria = Q(status="confirmed")
            if title:
                criteria = criteria & Q(title__icontains=title)
            if body:
                criteria = criteria & Q(body__icontains=body)
        orders = (
            AdvertOrder.objects.filter(criteria)
            .values("id", "publish", "author", "status", "title",
"body")
            .order_by("id")
        )
        for order in orders:
            count_user_offers = 0
            count = 0
            info(order)
            if (userType == "performer"):
                offers = AdvertOffer.objects.filter(
                    order=order['id']).values("author",
"status")
                for offer in offers:
```

```
                    if offer["author"] == request.user.id:
                        count_user_offers = count_user_offers +
1
                        count = count + 1
                    elif offer["status"] == "confirmed":
                        count = count + 1
            else:
                offers = AdvertOffer.objects.filter(
                    status="confirmed", order=order['id'])
                count = len(offers)
            order["count_user_offers"] = count_user_offers
            order["count"] = count
        paginator = Paginator(orders, pp)
        page_obj = paginator.get_page(page_number)
        data = {"list": list(page_obj), "total": len(orders)}
        return SuccessResponse(request, data)
```

Отримання даних окремого замовлення відповідно до прав користувача (див. Лістинг 10).

Лістинг *10 Функція отримання даних замовлення*

```
def order(request, order_id):
    try:
        orderDetail = AdvertOrder.objects.get(id=order_id)
        orderDetail.publish                                    =
orderDetail.publish.strftime("%-d %b %Y %H:%M")
        user = User.objects.get(username=orderDetail.author)
        offers                                                 =
AdvertOffer.objects.filter(order=orderDetail.id).values(
            "id", "order", "publish", "author", "status",
"body", "selected"
        )
        order = {
            "id": orderDetail.id,
```

```
                "publish": orderDetail.publish,
                "author": user.username,
                "author_id": orderDetail.author.id,
                "title": orderDetail.title,
                "status": orderDetail.status,
                "body": orderDetail.body,
                "count": len(offers.filter(status="confirmed")),
            }
            userType = getUserType(request.user)
            order["count"] = len(offers)
            order["offers"] = []
            for offer in offers:
                if userType == "performer" and offer["author"] ==
request.user.id:
                    order["offers"].append(offer)
                elif userType == "customer" and offer["status"]
== "confirmed" and order["author_id"] == request.user.id:
                    order["offers"].append(offer)
                info(offer)
            return SuccessResponse(request, order)
        except Exception as ex:
            return ErrResponse(request, str(ex))
```

Видалення замовлення відповідно до прав користувача наведено в лістингу 11.

Лістинг *11 Функція видалення замовлення*

```
def deleteOrder(request, order_id):
    userType = getUserType(request.user)
    try:
        if (userType != "customer"):
            raise Exception(errorNotAllowed)
        order_instance                                    =
AdvertOrder.objects.get(id=order_id)
```

```
            if (order_instance.author_id != request.user.id):
                raise Exception(errorNotAllowed)
            order_instance.delete()
            return SuccessResponse(request, {})
        except Exception as ex:
            return ErrResponse(request, str(ex))
```

В функції saveOffer (див. Лістинг 12) виконується додавання нової або оновлення даних вже існуючої пропозиції, відбувається перевірка наявності прав на виконання цієї дії користувачем. При додаванні нової пропозиції, завантажується обраний користувачем файл з відео, а також створюється превью зниженої якості та накладеним логотипом (водяним знаком) для нього. Також для нової пропозиції підбирається модератор викликом функції getModeratorId.

Лістинг *12 Функція збереження даних пропозиції*

```
def saveOffer(request):
    userType = getUserType(request.user)
    try:
        if (userType != "performer"):
            raise Exception(errorNotAllowed)
        mode = request.POST["mode"]
        body = request.POST["comment"]
        if mode == "add":
            video = handle_uploaded_file(
                settings.MEDIA_ROOT, request.FILES["file"])
            order_id = request.POST["order_id"]
            AdvertOffer.objects.create(
                video=video,  order_id=order_id,  body=body,
author_id=request.user.id, moderator_id= getModeratorId()
            )
        else:
            offer_id = request.POST["id"]
```

```
                offer_instance                                    =
AdvertOffer.objects.get(id=offer_id)
                if          (offer_instance.author_id          !=
request.user.id):
                    raise Exception(errorNotAllowed)
                offer_instance.body = body
                offer_instance.save()
            return SuccessResponse(request, {})
        except Exception as ex:
            return ErrResponse(request, str(ex))
```

Отримання даних окремої пропозиції відповідно до прав користувача (див. Лістинг 13).

Лістинг *13 Функція отримання даних пропозиції*

```
def offer(request, offer_id):
    if not request.user.is_authenticated:
        return NotAuthedResponse(request)
    try:
        offerDetail = AdvertOffer.objects.get(id=offer_id)
        offer = {
            "id": offerDetail.id,
            "body": offerDetail.body,
        }
        return SuccessResponse(request, offer)
    except Exception as ex:
        return ErrResponse(request, str(ex))
```

Функція playVideo (див. Лістинг 14) забезпечує передачу відео до фронтенду з урахуванням рівня доступу. Якщо у користувача немає прав доступу, або запрошений файл не існує, то йому буде відображено відео NoPermissions.mp4. Для замовників буде відображено превью, а для виконавців додатково - оригінальне відео.

Лістинг *14 Функція передачі відео*

```
def playVideo(request, offer_id, is_preview):
    userType = getUserType(request.user)
    try:
        if (userType == ""):
            raise Exception(errorRequireLogin)
        offer = AdvertOffer.objects.get(id=offer_id)
        if (offer == None):
            raise Exception("Пропозицію не знайдено")
        path = settings.MEDIA_ROOT
        if is_preview:
            path = path + "/small"
        filepath = getFullPath(path, offer.video)
        if (not os.path.isfile(filepath)):
            raise Exception("Відео не знайдено")
        if (userType == "customer"):
            order =
AdvertOrder.objects.get(id=offer.order.id)
            # тільки на свої замовлення зі статусом
пропозиції confirmed
            if (order.author_id != request.user.id or
offer.status != "confirmed"):
                raise Exception("Немає прав для перегляду
відео")
        elif (userType == "performer"):
            if (offer.author_id != request.user.id):
                raise Exception("Немає прав для перегляду
відео")
        ext = os.path.splitext(filepath)[1]
    except Exception as ex:
        filepath = getFullPath(settings.MEDIA_ROOT,
"NoPermissions.mp4")
        ext = ".mp4"
    file = FileWrapper(open(filepath, 'rb'))
```

```
        response = HttpResponse(file, content_type='video/' +
ext)
        response['Content-Disposition'] = 'attachment;
filename=video-offer-' + \
            str(offer_id) + ext
        return response
```

Функцію deleteOffer для видалення пропозиції наведено в лістингу 15.

Лістинг *15 Функція видалення пропозиції*

```
    def deleteOffer(request, offer_id):
        userType = getUserType(request.user)
        try:
            if (userType != "performer"):
                raise Exception(errorNotAllowed)
            offer_instance =
AdvertOffer.objects.get(id=offer_id)
            if (offer_instance.author_id != request.user.id):
                raise Exception(errorNotAllowed)
            offer_instance.delete()
            return SuccessResponse(request, {})
        except Exception as ex:
            return ErrResponse(request, str(ex))
```

Функція selectOffer (див. Лістинг 16) змінює поле selected пропозиції, яке означає, що пропозиція обрана замовником.

Лістинг *16 Функція обрання пропозиції*

```
    def selectOffer(request, offer_id):
        userType = getUserType(request.user)
        try:
            if (userType != "customer"):
                raise Exception(errorNotAllowed)
            offer_instance =
AdvertOffer.objects.get(id=offer_id)
```

```
            if (offer_instance.order.author_id !=
request.user.id):
                raise Exception(errorNotAllowed)
            offer_instance.selected = 1
            offer_instance.save()
            return SuccessResponse(request, {})
        except Exception as ex:
            return ErrResponse(request, str(ex))
```

Зміна статусу замовлення/пропозиції з перевіркою ролі та прав користувача на виконання цієї дії наведено в лістингу 17.

Лістинг *17 Функція зміни статусу замовлення/пропозиції*

```
def status(request):
    userType = getUserType(request.user)
    if (userType == ""):
        return NotAuthedResponse(request)
    user_id = request.user.id
    try:
        json_data = json.loads(request.body)
        if userType == "customer":
            instance =
AdvertOrder.objects.get(id=json_data["id"])
        if userType == "performer":
            instance =
AdvertOffer.objects.get(id=json_data["id"])
        status = instance.status
        newStatus = json_data["status"]
        isAllowed = instance.author_id == user_id and (
            status == "draft"
            and newStatus == "publish"
            or status == "confirmed"
            and newStatus == "suspended"
            or status == "suspended"
```

```
            and newStatus == "confirmed"
        )
        if isAllowed:
            instance.status = newStatus
            instance.save()
            return SuccessResponse(request, {})
        else:
            return ErrResponse(request, errorNotAllowed)
    except Exception as ex:
        return ErrResponse(request, str(ex))
```

3.4.2 Реалізація клієнтської частини

Основні пакети та залежності для клієнтської частини:

- axios: 1.6.8
- bootstrap: 5.3.3
- draft-js: 0.11.7
- draft-js-export-html: 1.4.1
- draft-js-import-html: 1.4.1
- html-react-parser: 5.1.10
- react: 18.3.1
- react-dom: 18.3.1
- react-draft-wysiwyg: 1.15.0
- react-player: 2.16.0
- react-redux: 9.1.2
- react-router: 6.23.1
- react-router-dom: 6.23.1
- react-scripts: 5.0.1
- reactstrap: 9.2.2
- redux: 5.0.1
- redux-thunk: 3.1.0

Весь код клієнтської частини розбитий на компоненти та розподілений по окремих директоріях (див. Рис. 12).

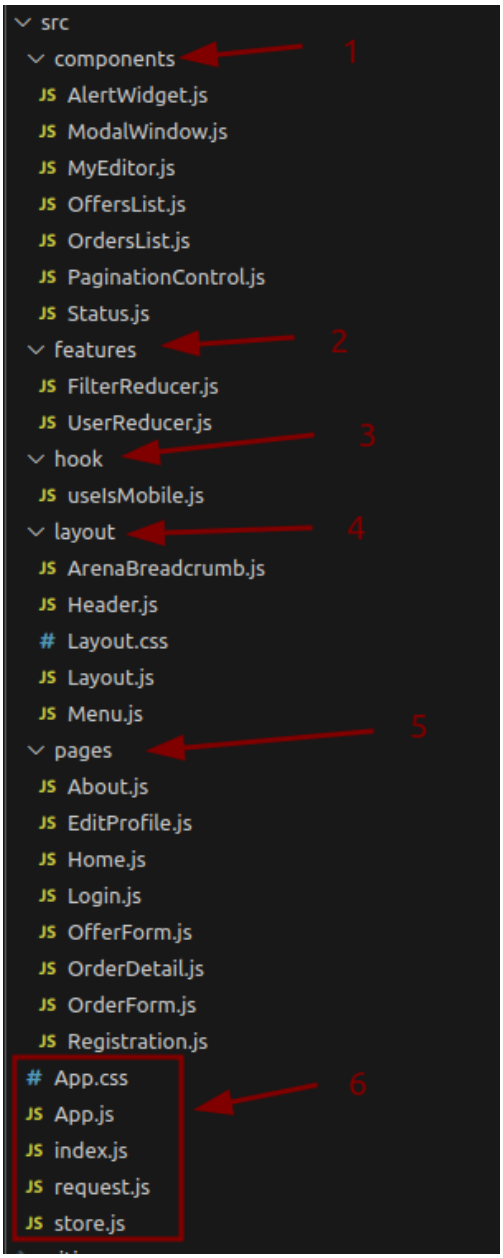

Рисунок *12 — Директорії reactapp*

Головним файлом проекту є index (див. Рис. 12, 6, див. Лістинг 18), в якому підключається компонент додатку App (див. Рис. 12, 6, Лістинг 19) та обрамлений компонентом Provider (від Redux), в який передається сховище store (див. Рис. 12, 6, Лістинг 20), що підключає 2 редьюсера (див. Рис. 12, 2, Лістинг 21, Лістинг 22)

Лістинг *18 Головний файл проекту index.js*

```
import App from "./App";
import store from "./store";
const root =
ReactDOM.createRoot(document.getElementById("root"));
root.render(
  <React.StrictMode>
    <Provider store={store}>
      <App />
    </Provider>
  </React.StrictMode>
);
```

Лістинг *19 Компонент всього додатку App.js*

```
const router = createBrowserRouter([
  {
    path: "/",
    element: <Layout><Home /></Layout>,
  }, {
    path: "/about",
    element: <Layout><About /></Layout>,
  }, {
    path: "/login",
    element: <Layout><Login /></Layout>,
  }, {
    path: "/profile",
    element: <Layout><EditProfile /></Layout>,
  }, {
    path: "/registration",
    element: <Layout><Registration /></Layout>,
  }, {
    path: "/addorder",
    element: <Layout><OrderForm mode="add" /></Layout>,
  }, {
```

```
      path: "/order/:id",
      element: <Layout><OrderDetail /></Layout>,
    }, {
      path: "/editorder/:id",
      element: <Layout><OrderForm mode="edit" /></Layout>,
    }, {
      path: "/addoffer/:order_id",
      element: <Layout><OfferForm mode="add" /></Layout>,
    }, {
      path: "/editoffer/:order_id/:id",
      element: <Layout><OfferForm mode="edit" /></Layout>,
    },
]);
  function App() {
    const dispatch = useDispatch();
    useEffect(() => {
      request
        .get("/user/")
        .then((response) => {

          if (response.status === "success") {
            dispatch(setUser(response.data));
          }
        })
        .catch((error) => {});
    });
    return <RouterProvider router={router} />;
  }
```

Лістинг *20 Сховище store.js*

```
  export default configureStore({
    reducer: {
      user: UserReducer,
      filter: FilterReducer,
    },
```

```
})
```

Лістинг *21 UserReducer.js*

```
export const UserReducer = createSlice({
  name: 'user',
  initialState: {
    value: null,
  },
  reducers: {
    setUser: (state, action) => {
      state.value = action.payload
    },
  },
})
```

Лістинг *22 FilterReducer.js*

```
const defaultFilterValue = {
  title: "",
  body: "",
  my_orders: false,
  my_offers: false,
};
export const FilterReducer = createSlice({
  name: "filter",
  initialState: {
    value: defaultFilterValue,
  },
  reducers: {
    setFilter: (state, action) => {
      state.value = action.payload;
    },
    resetFilter: (state, action) => {
      state.value = defaultFilterValue;
    },
  },
```

```
});
```

В компоненті додатку App формується роутер для навігації по сторінках. В кожному елементі роутера підключаємо компонент контенту окремої сторінки, такі як About, EditProfile, Home, та інші (див. Рис. 12, 5), обрамлений компонентом Layout (див. Рис. 12, 4, Лістинг 23), який формує розташування загальних елементів на сторінці, таких як Menu (див. Лістинг 24), ArenaBreadcrumb (див. Лістинг 25) та відповідні контейнери.

В компоненті Menu використовується хук useIsMobile (див. Рис. 12, 3), що виконується при зміні розміру вікна браузера та при ширині менше 800 пікселів переключає режим меню у зручний стан для мобільних пристроїв.

Лістинг *23 Компонент Layout.js*

```
import "./Layout.css";
const Layout = ({ children }) => {
  return (
    <React.Fragment>
      <Menu />
      <Container>
        <Row className="row align-items-center">
          <Col className=""></Col>
          <Col className="p-4 col-8">
            <ArenaBreadcrumb />
            {children}
          </Col>
          <Col></Col>
        </Row>
      </Container>
    </React.Fragment>
  );
};
```

Лістинг *24 Компонент Menu.js*

```
const Menu = () => {
  const navigate = useNavigate();
  const user = useSelector((state) => state.user.value);
  const [isOpen, setIsOpen] = useState(true);
  const toggle = () => setIsOpen(!isOpen);
  const isMobile = useIsMobile();
  const userType = user ? user.user_type : "";
  let menu;
  switch (userType) {
    case "customer":
      menu = [{ path: "/addorder", text: "Додати замовлення"
}];
      break;
    default:
      menu = [];
      break;
  }
  menu.push({ path: "/about", text: "Про сервіс" });
  const args = {
    color: "dark",
    light: false,
    dark: true,
    full: "false",
    expand: !isMobile,
    container: "fluid",
  };
  return (
    <Navbar {...args}>
      <NavbarBrand onClick={() =>
navigate("/")}>AdvertArena</NavbarBrand>
      <NavbarToggler onClick={toggle} />
      <Collapse isOpen={isOpen} navbar>
        <Nav className="me-auto" navbar>
          {menu.map((item) => (
            <NavItem key={item.path}>
```

```
                  <NavLink onClick={() =>
navigate(item.path)}>{item.text}</NavLink>
                </NavItem>
              ))}
            </Nav>
            <NavbarText>
              <Profile user={user} />
            </NavbarText>
          </Collapse>
        </Navbar>
      );
    };
```

Лістинг *25 Компонент ArenaBreadcrumb.js*

```
const ArenaBreadcrumb = () => {
  const navigate = useNavigate();
  const paths = window.location.pathname.split("/");
  const path = "/" + paths[1];
  const mapPageName = {
    "/": "Домівка",
    "/about": "Про сервіс",
    "/registration": "Реєстрація",
    "/login": "Вхід",
    "/profile": "Профіль",
    "/order": "Замовлення",
    "/addorder": "Додавання замовлення",
    "/editorder": "Редагування замовлення",
    "/addoffer": "Додавання пропозиції",
    "/editoffer": "Редагування пропозиції",
  };
  const parts = ["/"];
  if (path === "/editorder") {
    parts.push("/order/" + paths[2] + "/");
  }
  if (path === "/editoffer") {
```

```
        parts.push("/order/" + paths[2] + "/");
      }
      if (path !== "/" && mapPageName[path]) {
        parts.push(path);
      }

      return (
        <Container>
          <Row>
            <Col className="p-1 col-24">
              <Breadcrumb listTag="div">
                {parts.map((url) => {
                  const urlPath = "/" + url.split("/")[1];
                  const isActive = path !== urlPath;
                  return (
                    <BreadcrumbItem
                      key={url}
                      href={url}
                      active={isActive}
                      tag="li"
                      className={isActive ? "breadcrumb-link" :
""}
                      onClick={() => isActive && navigate(url)}
                    >
                      {mapPageName[urlPath]}
                    </BreadcrumbItem>
                  );
                })}
              </Breadcrumb>
            </Col>
          </Row>
        </Container>
      );
    };
```

Контент головної сторінки формується в компоненті Home, представленою в лістингу 26 та лістингу 27. Здійснюється взаємодія з сервером за допомогою AJAX запитів через розроблений компонент request (див. Лістинг 28). Хук useEffect спрацьовує при завантаженні сторінки та при зміні номера сторінки (page), фільтрів (filter) та виконує запит до сервера для завантаження списку замовлень. При отриманні помилки від сервера (status: “error”) вона відображається за допомогою AlertWidget (див. Рис. 12, 1), який розроблений на UncontrolledAlert з модуля reactstrap. Таким самим чином реалізовано відображення помилок на інших сторінках, що взаємодіють з сервером.

Лістинг *26 Компонент Home. Взаємодія з сервером, обробка дій користувача*

```
const Home = () => {
 const titleRef = useRef(null);
 const bodyRef = useRef(null);
 const myOrdersRef = useRef(false);
 const myOffersRef = useRef(false);
 const [orders, setOrders] = useState([]);
 const [page, setPage] = useState(1);
 const [total, setTotal] = useState(1);
 const [error, setError] = useState("");
 const filter = useSelector((state) => state.filter.value);
 const user = useSelector((state) => state.user.value);
 const dispatch = useDispatch();

 useEffect(() => {
   request
     .get("/orders/", { page, pp, ...filter })
     .then((response) => {
       if (response.status === "success") {
         setOrders(response.data.list);
         setTotal(response.data.total);
```

```
      } else {
        setError(response.msg);
      }
    })
    .catch((error) => setError(error));
}, [page, filter]);

const userType = user ? user.user_type : "";
const handleSubmitFilter = (e) => {
  e.preventDefault();
  const { title, body, my_orders, my_offers } = e.target;

  dispatch(
    setFilter({
      title: title.value,
      body: body.value,
      my_orders: my_orders ? my_orders.checked : false,
      my_offers: my_offers ? my_offers.checked : false,
    })
  );
};

const resetForm = () => {
  dispatch(resetFilter());
  titleRef.current.value = "";
  bodyRef.current.value = "";
  if (myOrdersRef.current) {
    myOrdersRef.current.checked = false;
  }
  if (myOffersRef.current) {
    myOffersRef.current.checked = false;
  }
};
```

Лістинг *27 Компонент Home. Генерація HTML*

```
          <Card body>
            <CardTitle tag="h5">Фільтр</CardTitle>
            <Form                  onSubmit={handleSubmitFilter}
id={orderFilterFormId}>
              <AlertWidget message={error} setMessage={setError}
/>
              <Row>
                <Col md={6}>
                  <FormGroup>
                    <Label         for="filterTitleId">Заголовок
містить</Label>
                    <Input
                      id="filterTitleId"
                      innerRef={titleRef}
                      name="title"
                      type="text"
                      defaultValue={filter.title || ""}
                    />
                  </FormGroup>
                </Col>
                <Col md={6}>
                  <FormGroup>
                    <Label              for="filterBodyId">Опис
містить</Label>
                    <Input
                      id="filterBodyId"
                      innerRef={bodyRef}
                      name="body"
                      type="text"
                      defaultValue={filter.body || ""}
                    />
                  </FormGroup>
                </Col>
              </Row>
              <Row>
```

```
                    <Col md={6}>
                      {userType === "customer" && (
                        <FormGroup check inline>
                          <Input
                            id="filterMyOrdersId"
                            innerRef={myOrdersRef}
                            name="my_orders"
                            type="checkbox"
                          />
                          <Label check for="filterMyOrdersId">
                            Тільки мої замовлення
                          </Label>
                        </FormGroup>
                      )}
                      {userType === "performer" && (
                        <FormGroup check inline>
                          <Input
                            id="filterMyOffersId"
                            innerRef={myOffersRef}
                            name="my_offers"
                            type="checkbox"
                          />
                          <Label check for="filterMyOffersId">
                            Тільки з моїми пропозиціями
                          </Label>
                        </FormGroup>
                      )}
                    </Col>
                    <Col md={6}>
                      <div style={{ textAlign: "right" }}>
                        <Button    color="dark"    onClick={()    =>
resetForm()}>
                          Очистити
                        </Button>{" "}
                        <Button color="dark" type="submit">
```

```
                    Шукати
                  </Button>
                </div>
              </Col>
            </Row>
          </Form>
        </Card>
        <Row>
          <Col className="p-3">
            <OrdersList orders={orders} />
          </Col>
        </Row>
        <PaginationControl    setPage={setPage}    page={page}
pp={pp} total={total} />
```

Лістинг *28 Компонент request.js*

```
const getApiUrl = (path) => "/api" + path;
const request = {
  get: (path, params = []) => {
    const url = getApiUrl(path);
    const config = {
      headers: {
        "X-CSRFToken": getCookie("csrftoken"),
      },
      params: params,
    };
    return    axios.get(url,    config).then((response)    =>
response.data);
  },
  post: (path, data) => {
    const url = getApiUrl(path);
    const config = {
      headers: {
        "Content-Type": "application/json",
        "X-CSRFToken": getCookie("csrftoken"),
```

```
          },
        };
        return axios.post(url, data, config).then((response) =>
response.data);
      },
      postForm: (path, formId) => {
        const url = getApiUrl(path);
        const config = {
          headers: {
            "X-CSRFToken": getCookie("csrftoken"),
          },
        };
        return axios
          .postForm(url, document.querySelector("#" + formId),
config)
          .then((response) => response.data);
      },
    };
```

3.5 Практичне використання додатку

3.5.1 Користувацький інтерфейс для неавторизованого користувача

Коли користувач запускає веб-додаток, він автоматично потрапляє на головну сторінку. На головній сторінці для неавторизованого користувача відображається наступний користувацький інтерфейс (див. Рис. 13) , який містить:

1. Меню з наступними елементами:
    - AdvertArena (див. Рис. 13, 1) — назва розробленого веб-сервісу, а також посилання на головну сторінку.
    - Про сервіс (див. Рис. 13, 2) — посилання на сторінку з інформацією про веб-сервіс.

- Кнопки (див. Рис. 13, 3), які ведуть на сторінки авторизації та реєстрації.

2. Шлях до поточної сторінки (див. Рис. 13, 4). На головній сторінці він містить тільки назву сторінки.
3. Фільтр (див. Рис. 13, 5), за яким користувач може здійснювати фільтрацію та пошук даних за певними критеріями.
4. Перелік замовлень (див. Рис. 13, 6).
5. Пагінація (див. Рис. 13, 7), яка значно спрощує перегляд великої кількості замовлень на веб-сторінці.
6. Номера записів на сторінці та загальна кількість записів (див. Рис. 13, 8).

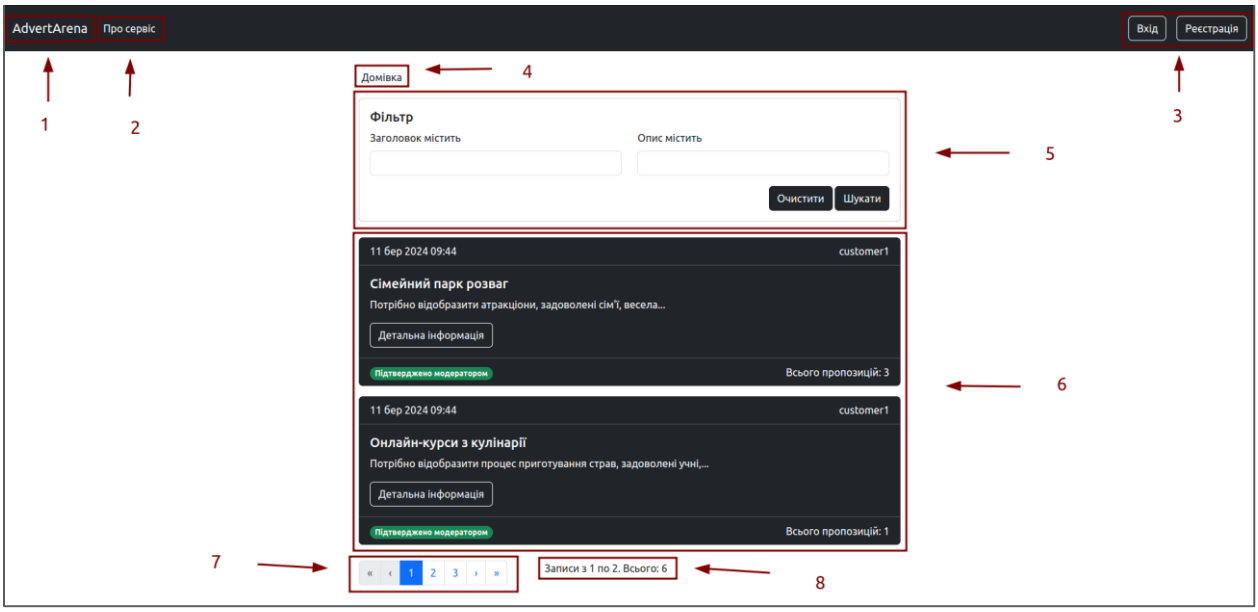

Рисунок *13 — Головна сторінка веб-сервісу для неавторизованого користувача*

Неавторизованому користувачу надана можливість фільтрації за двома критеріями: за заголовком та за описом замовлення (див. Рис. 14). Він може здійснювати фільтрацію як за двома критеріями одночасно, так і за окремим. Для цього йому необхідно заповнити потрібні поля форми та натиснути кнопку Шукати. Для очищення вмісту полів форми користувач має натиснути кнопку Очистити.

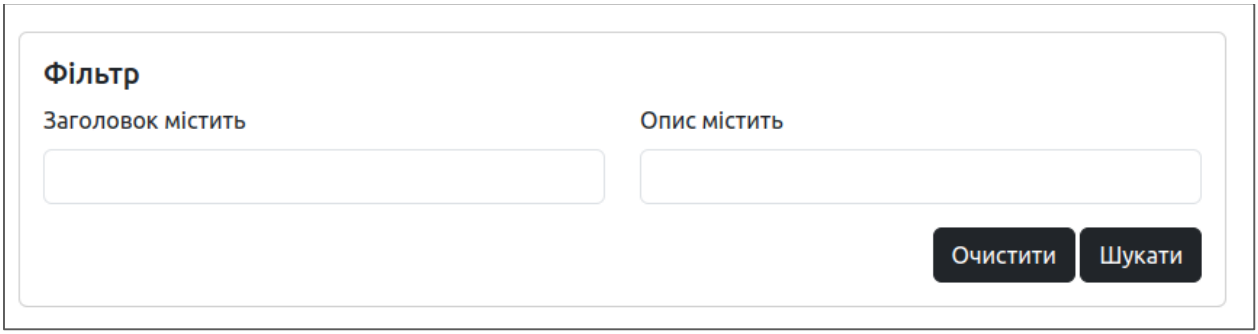

Рисунок *14 — Фільтр пошуку замовлень*

На головній сторінці в секції замовлення (див. Рис. 14, 6) користувач може бачити наступну інформацію про замовлення (див. Рис. 15):

1. Точні дата і час створення замовлення (див. Рис. 15, 1).
2. Ім’я користувача (див. Рис. 15, 2), який опублікував замовлення.
3. Заголовок (див. Рис. 15, 3), який описує головну тему замовлення.
4. Невеликий уривок змісту замовлення (див. Рис. 15, 4), що містить початок опису замовлення.
5. Статус (див. Рис. 15, 5), який відображає стан замовлення.
6. Кількість залишених пропозицій (див. Рис. 15, 6) до цього замовлення.

Також секція замовлення має кнопку Детальна інформація (див. Рис. 15, 7), при натисканні якої користувач потрапляє до сторінки з більш детальною інформацією про замовлення.

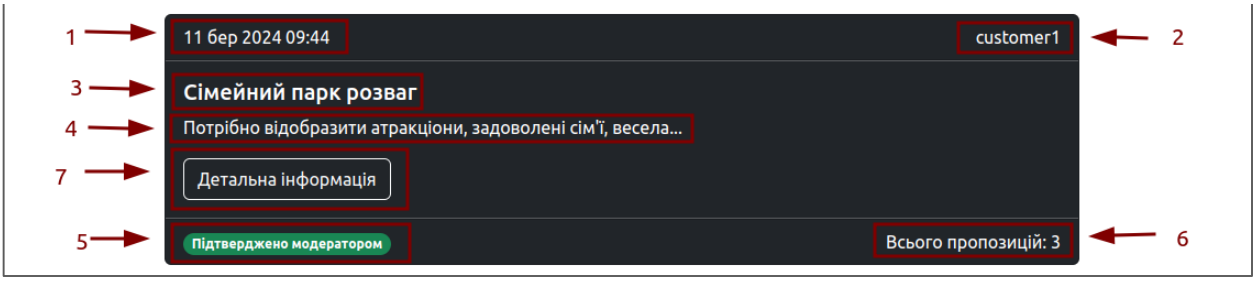

Рисунок *15 — Зовнішній вигляд замовлення на головній сторінці*

Незареєстрований користувач має можливість зареєструватися на сайті натиснувши в меню на кнопку Реєстрація, яка зображена на рисунку 16.

Для зручності навігації, наприклад, на попередню сторінку або на головну, було розміщено з лівого крайнього боку посилання на попередню сторінку та назва сторінки (див. Рис 16 1), яку на цей час переглядає користувач. При заповнені необхідних даних до наданої форми (див. Рис 16 2), користувач обов’язково повинен обрати роль (див. Рис 16 3), в залежності від дій, які він хоче виконувати, створювати замовлення або надсилати свої пропозиції. Після закінчення заповнення даних, користувач має підтвердити свій намір зареєструватися натиснувши на кнопку Зареєструватися (див. Рис 16 4).

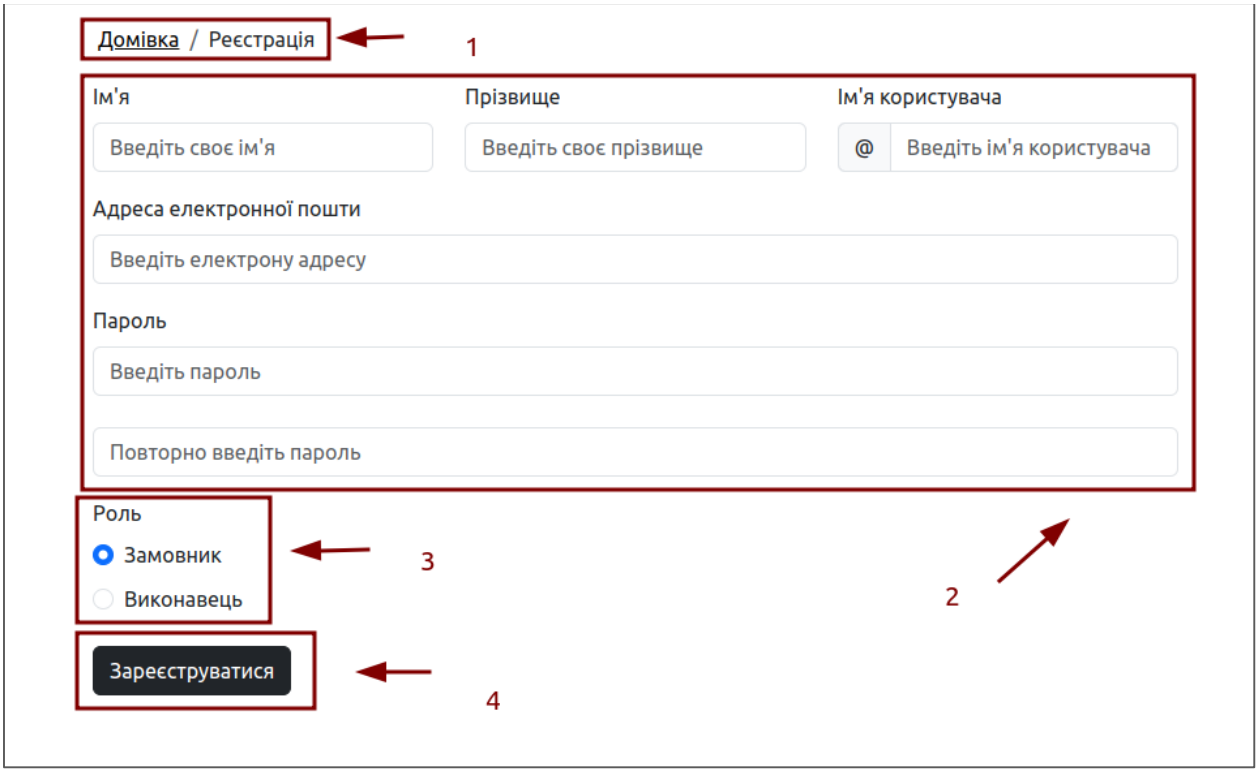

Рисунок *16 — Форма реєстрації для незареєстрованого користувача*

Якщо при реєстрації введені користувачем паролі не співпадають, або логін вже існує (чи виникають інші), то система відображає відповідний компонент з описом помилки (див. Рис. 17).

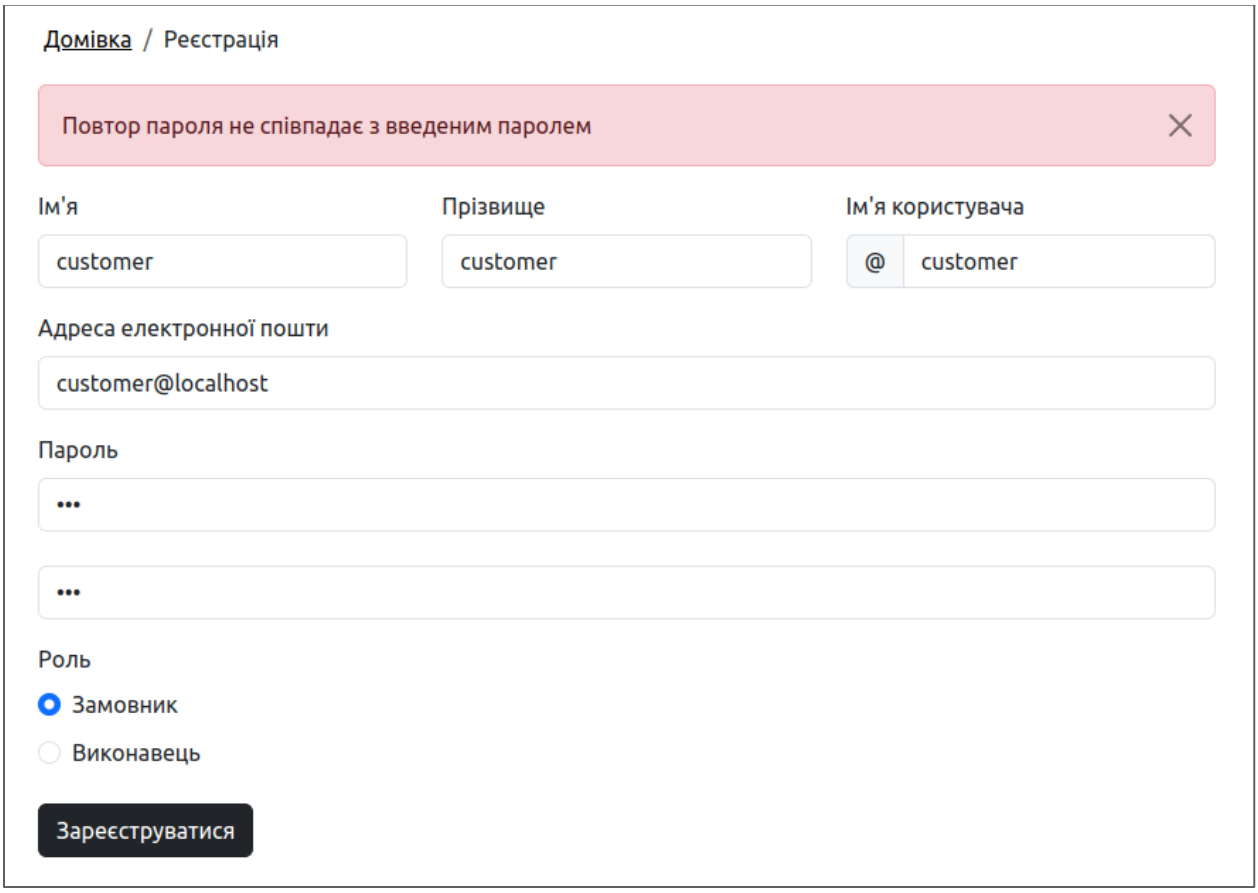

Рисунок *17 — Відображення помилки*

Таким самим чином при виникненні будь-яких з помилок у веденні даних користувачем на всіх інших сторінках, що взаємодіють з серверною частиною, система відобразить ці помилки, як було зображено на рисунку 17.

По закінченню процесу реєстрації, користувач потрапляє на сторінку входу до облікового запису (див. Рис 18), де для авторизації йому необхідно заповнити поля, ввівши свій логін та пароль. Щоб увійти до системи, по закінченню заповнення необхідних даних, користувач має натиснути на кнопку Увійти.

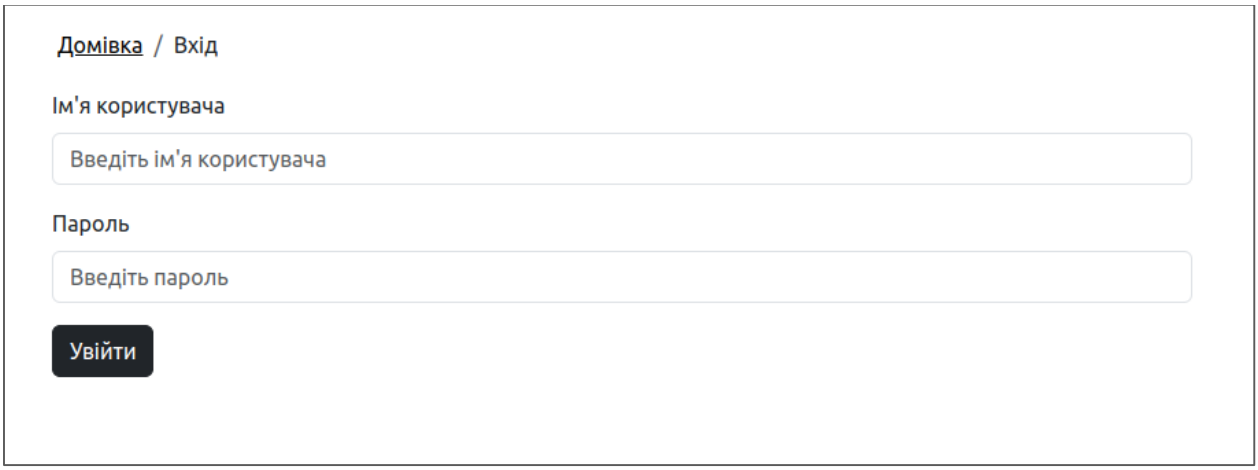

Рисунок *18 — Форма авторизації для незареєстрованого користувача*

3.5.2 Користувацький інтерфейс для замовника

Якщо авторизований користувач відноситься до групи замовників, головна сторінка для нього виглядає як на рисунку 19.

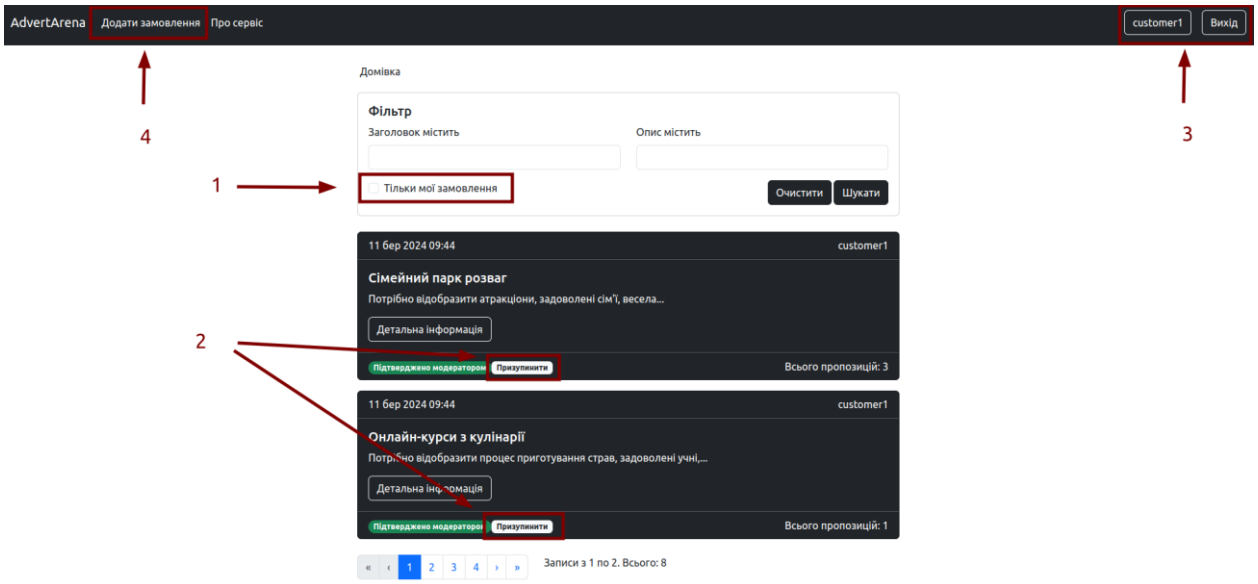

Рисунок *19 — Головна сторінка для користувача групи замовників*

В фільтрах для замовника додатково доступний фільтр Тільки мої замовлення (див. Рис. 19, 1), який здійснює пошук тих замовлень, які були створені цим замовником.

В замовленнях, власником яких є автентифікований замовник, біля статусу замовлення з’являється кнопка для можливого змінення статусу (див. Рис. 19, 2).

В меню для замовника відображаються нові елементи:

1. Кнопки (див. Рис. 19, 3):
    - Кнопка з ім'ям користувача, яка, при натисканні, навігує на сторінку Профіль (див. Рис. 20).
    - Кнопка Вихід, яка здійснює вихід з системи.
2. Додати замовлення (див. Рис. 19, 4) - посилання на сторінку додавання замовлення.

Сторінка Додавання замовлення (див. Рис. 20) містить форму, для створення замовлення.

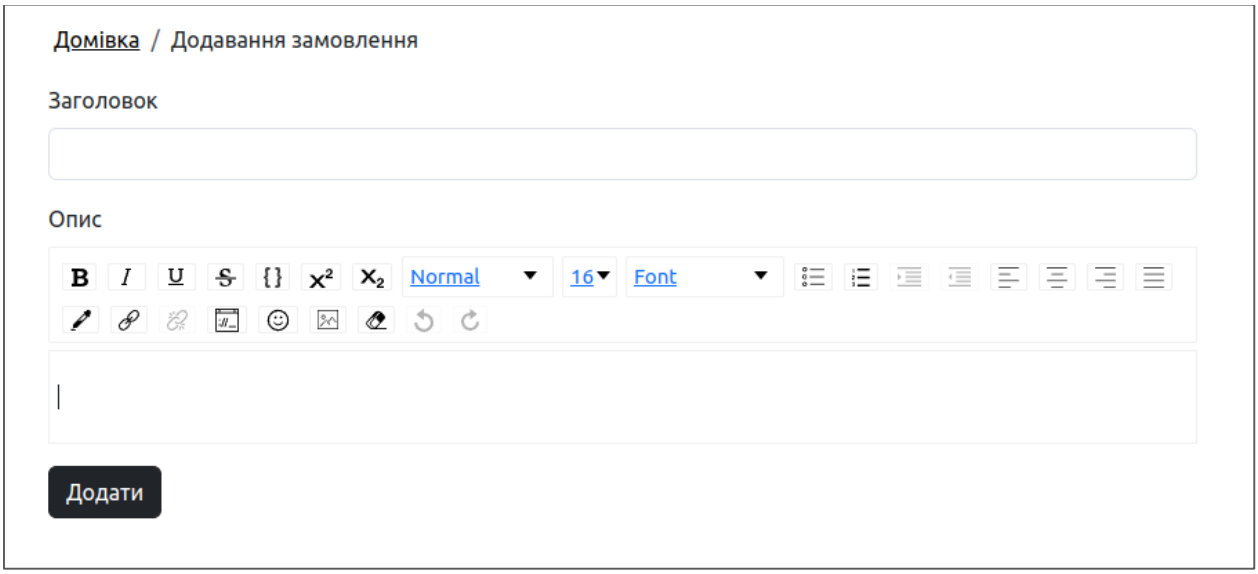

Рисунок *20 — Сторінка додавання замовлення*

Щоб потрапити до сторінки з детальною інформацією про замовлення (див. Рис. 21), замовнику необхідно натиснути на кнопку Детальна інформація (див. Рис. 15 7) до замовлення, яке його цікавить.

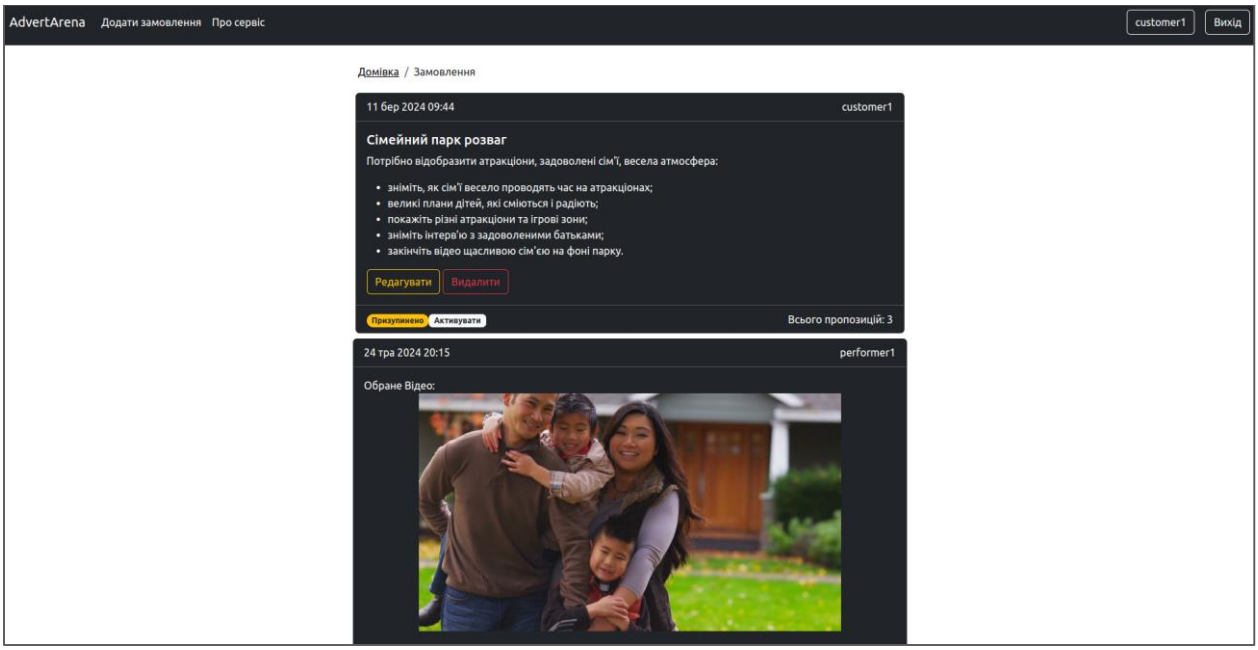

Рисунок *21 — Сторінка з детальною інформацією про замовлення для замовника*

Якщо замовлення, яке переглядає замовник, належить йому, він побачить, окрім детального змісту, кнопки для редагування та видалення (див. Рис. 22, 1) замовлення, а також його статус та кнопку для зміни статусу (див. Рис. 22, 2).

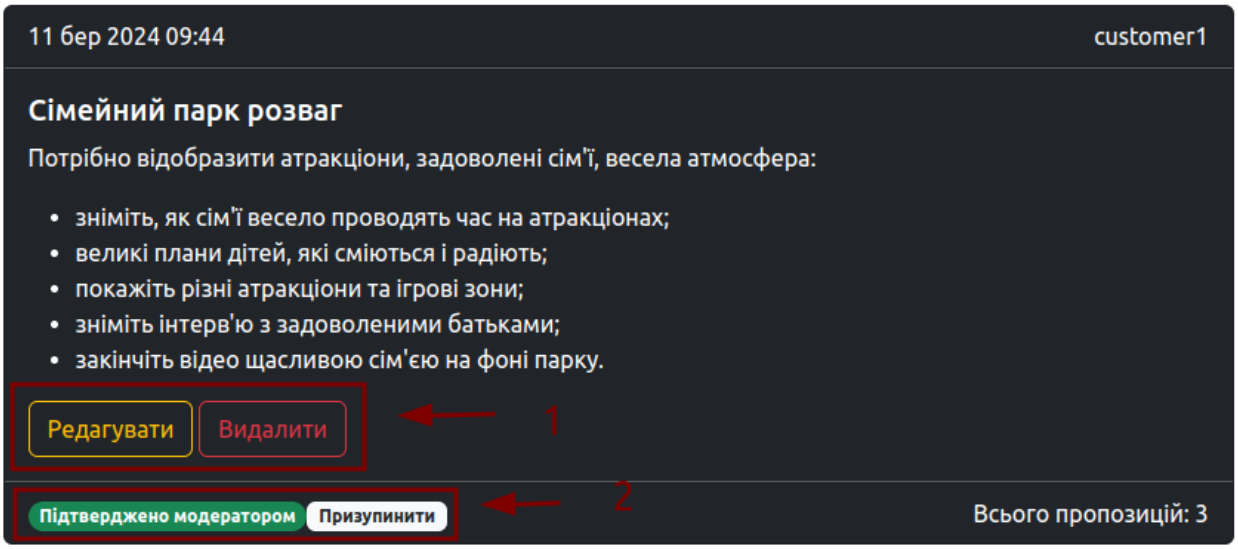

Рисунок *22 — Детальна інформація про замовлення*

Також, замовник може переглядати затверджені модератором відео-пропозиції, додані виконавцем до його замовлення (див. Рис. 21). Якщо пропозиція ще не була обрана замовником (див. Рис. 23), то для неї відображається Превью (див. Рис. 23 1), відео низької якості з логотипом сервісу посередині (див. Рис. 23 2). Щоб обрати відео-пропозицію, користувачу необхідно натиснути кнопку Обрати (див. Рис. 23 3).

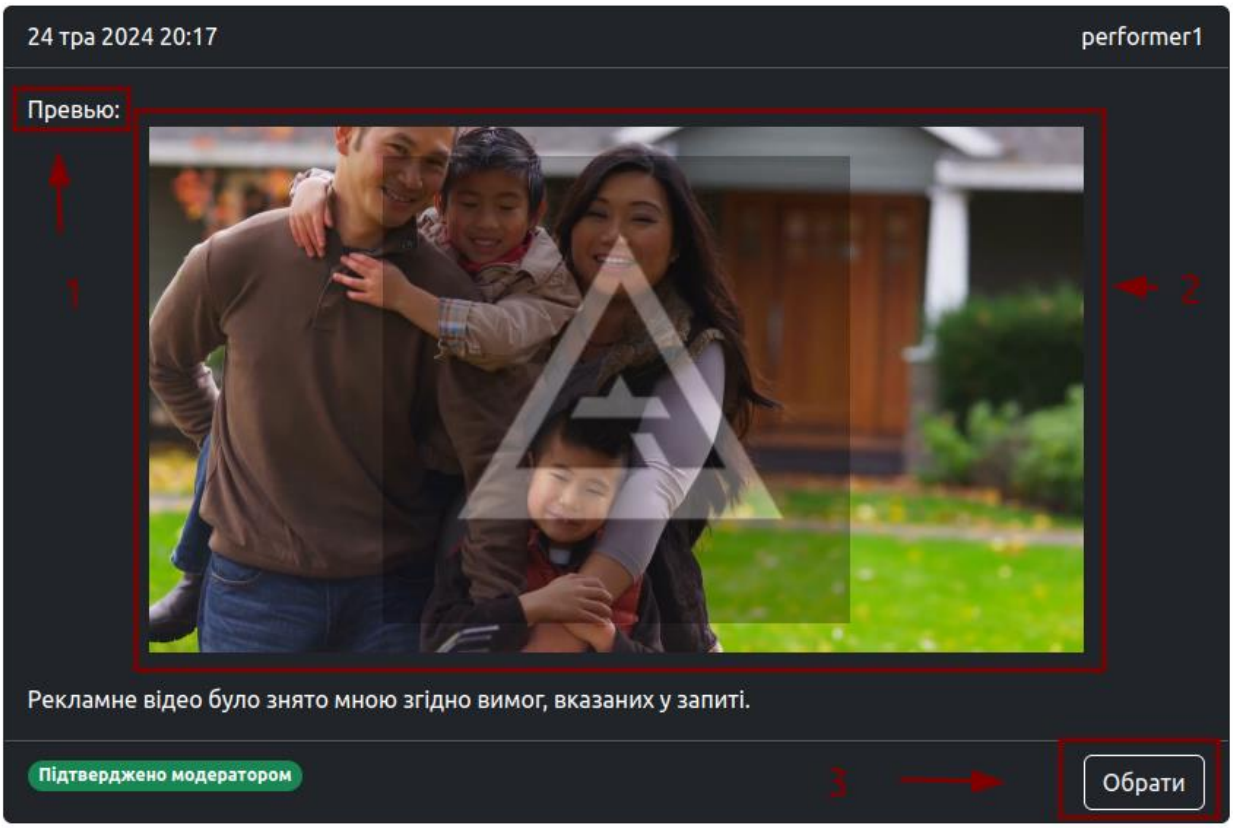

Рисунок *23 — Не обрана пропозиція до замовлення*

Після того, як замовник обирає відео, Превью (див. Рис. 23) змінюється на оригінальне високої якості Обране відео (див. Рис. 24).

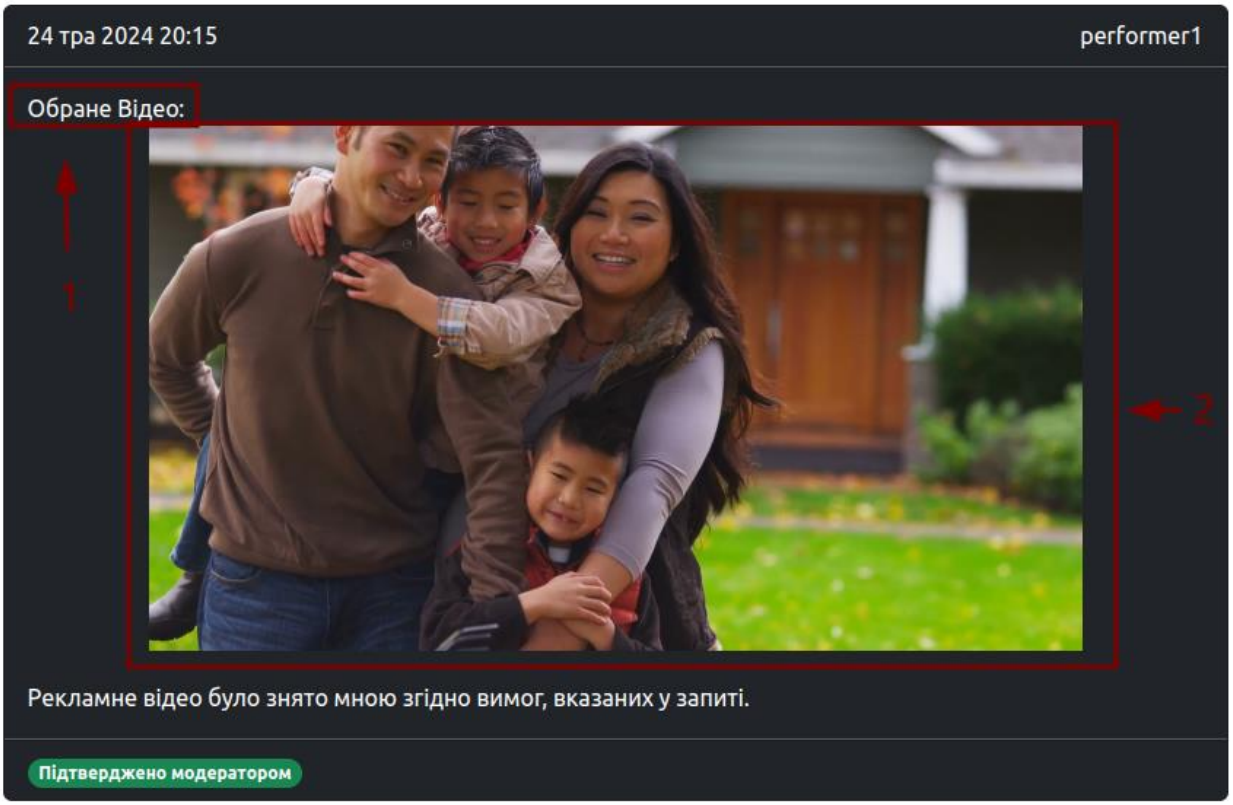

Рисунок *24 — Обрана пропозиція до замовлення*

3.5.2 Користувацький інтерфейс для виконавця

Головна сторінка для авторизованого користувача, який відноситься до групи виконавців, майже не відрізняється від головної сторінки для неавторизованих користувачів (див. Рис. 13), крім додаткового фільтру (див. Рис. 25, 1) та меню, на якому відображаються кнопки: на сторінку Профіль та вихід з системи (див. Рис. 25, 2).

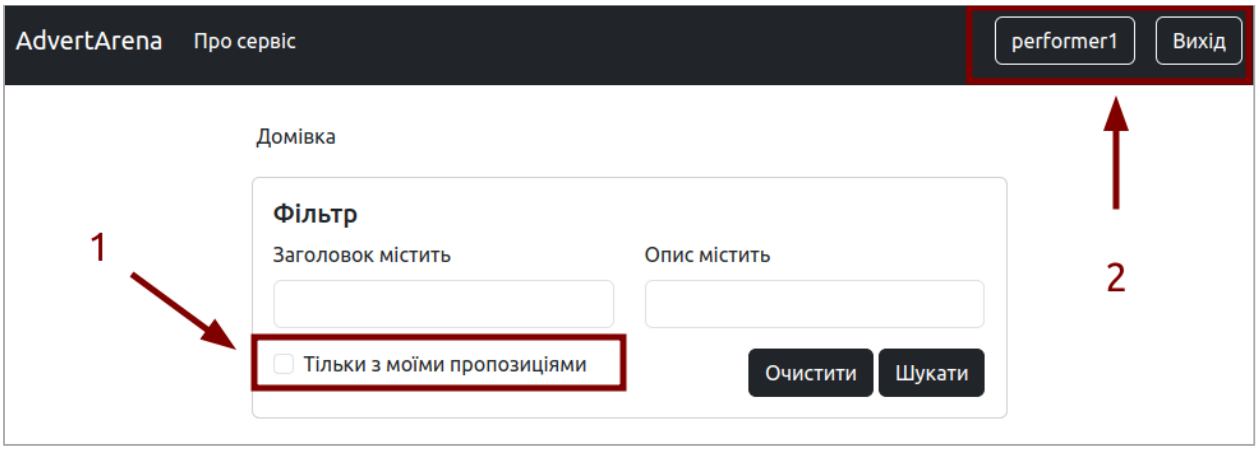

Рисунок *25 — Обрана пропозиція до замовлення*

Виконавець має можливість додати пропозицію до певного замовлення замовника на сторінці з деталями замовлення через кнопку Додати пропозицію (див. Рис. 26).

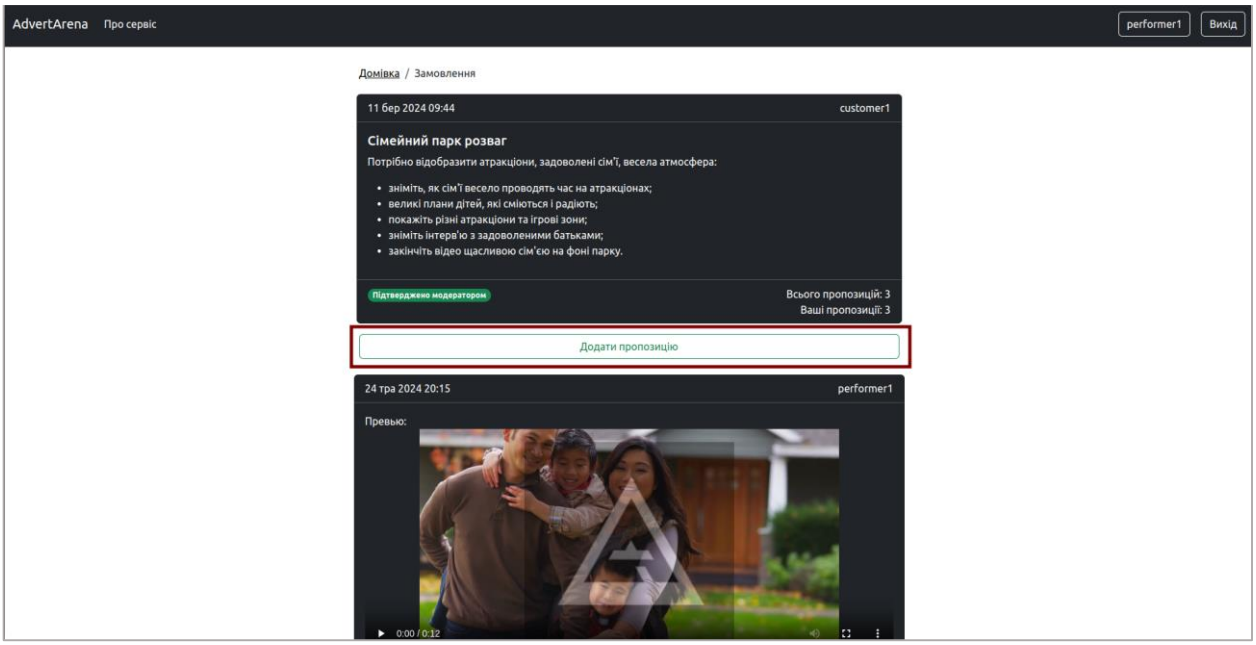

Рисунок *26 — Детальна інформація про замовлення для виконавця*

Форма додавання пропозиції зображена на рисунку 27.

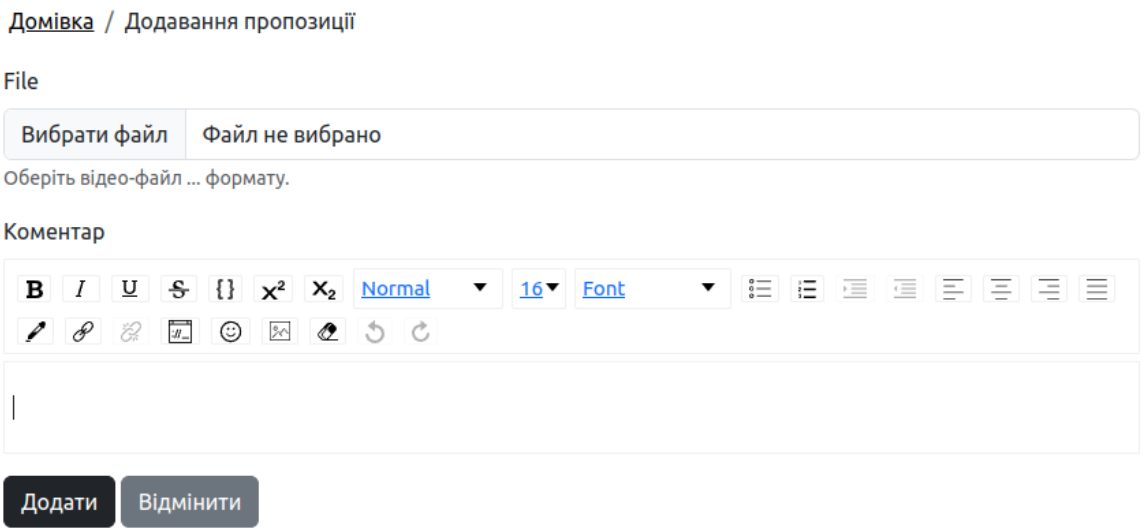

Рисунок *27 — Форма додавання пропозиції*

3.5.3 Додаткові функції, які можуть здійснювати замовники та для виконавці

Сторінка Профіль (див. Рис. 28) для авторизованого користувача налічує в собі інформацію, надану користувачем при реєстрації:

1. Імя та Прізвище (див. Рис. 28, 1). Інформація, яку користувач може редагувати.
2. Email та Роль (див. Рис. 28, 2). Дані, які користувач може лише переглянути.

Для здійснення редагування даних, користувачу необхідно замінити текст в полях форми, які доступні для редагування, після чого натиснути кнопку Зберегти.

Рисунок *28 — Сторінка профілю користувача*

Якщо авторизований користувач належить до групи замовників, то в замовленнях, власником яких є автентифікований замовник, біля статусу замовлення відобразиться кнопка для зміни його статусу (див. Рис. 29).

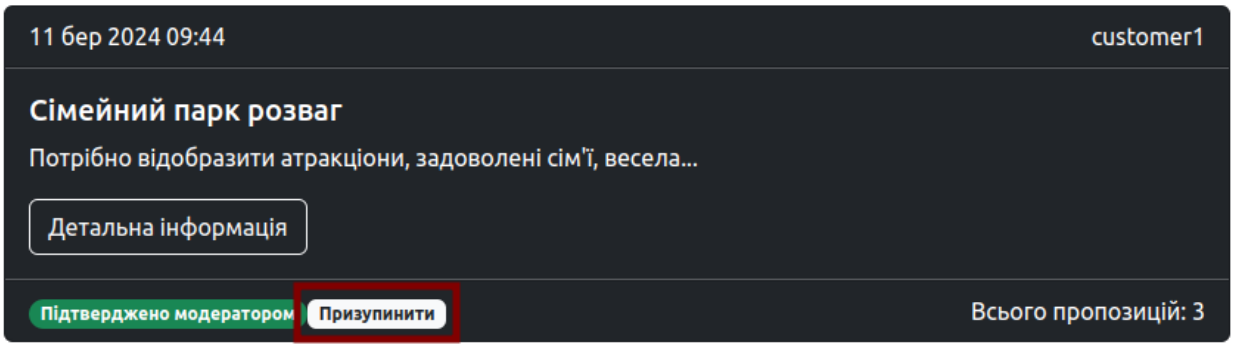

Рисунок *29 — Відображення пропозиції для автентифікованого виконавця-власника*

Для авторизованого користувача, який належить до групи виконавців, в пропозиціях, власником яких є автентифікований виконавець, біля статусу пропозиції відобразиться кнопка для зміни його статусу (див. Рис. 30).

Для статусів Відправлено на модерацію, На модерації та Відхилено модератором, якими займається модератор, кнопка для зміни статусу відображатися не буде.

При натисканні на кнопку зміни статусу, як для замовлення, так і для пропозиції, випливає вікно з запитом на підтвердження цієї дії (див. Рис. 31).

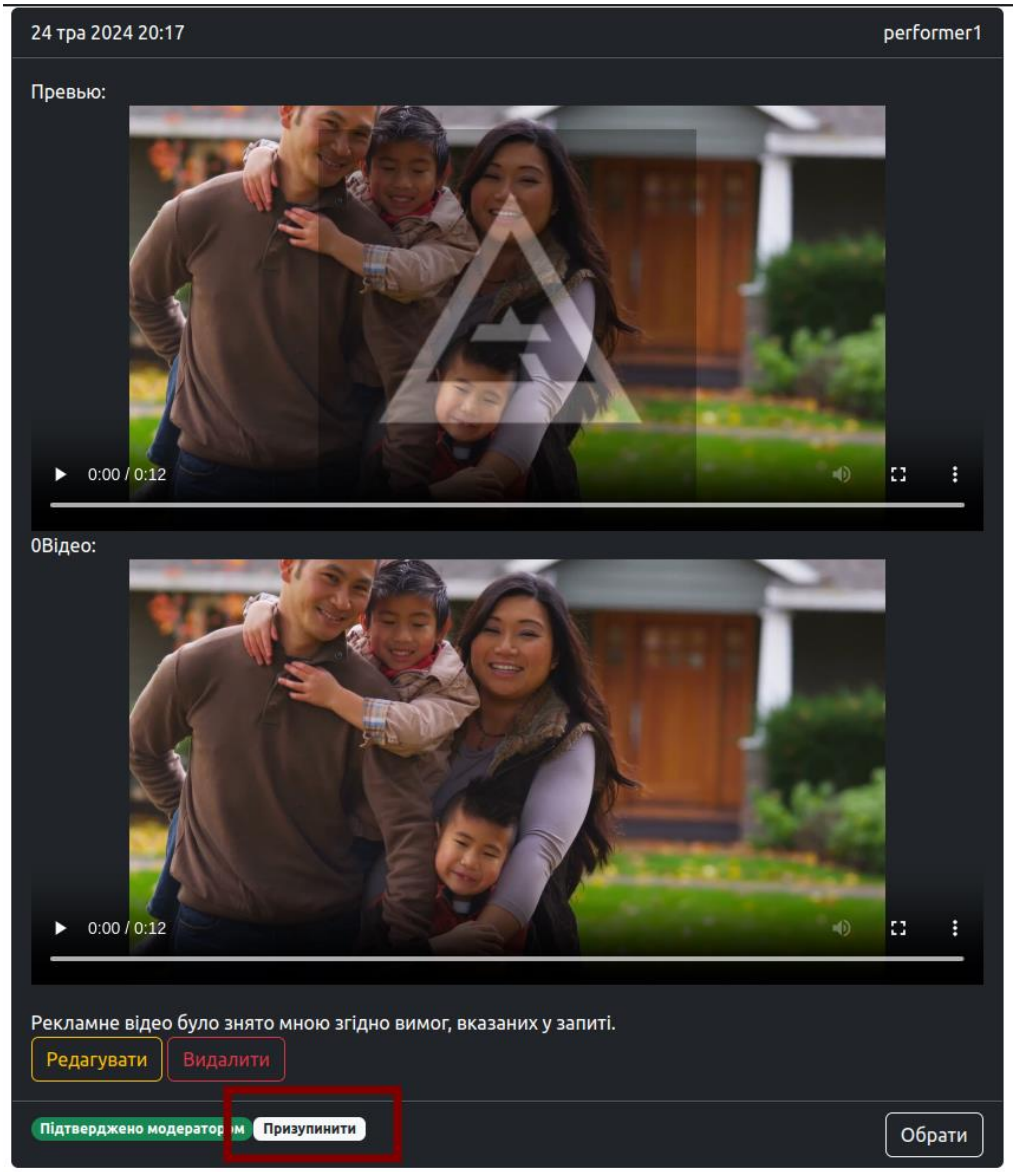

Рисунок *30 — Відображення пропозиції для автентифікованого виконавця-власника*

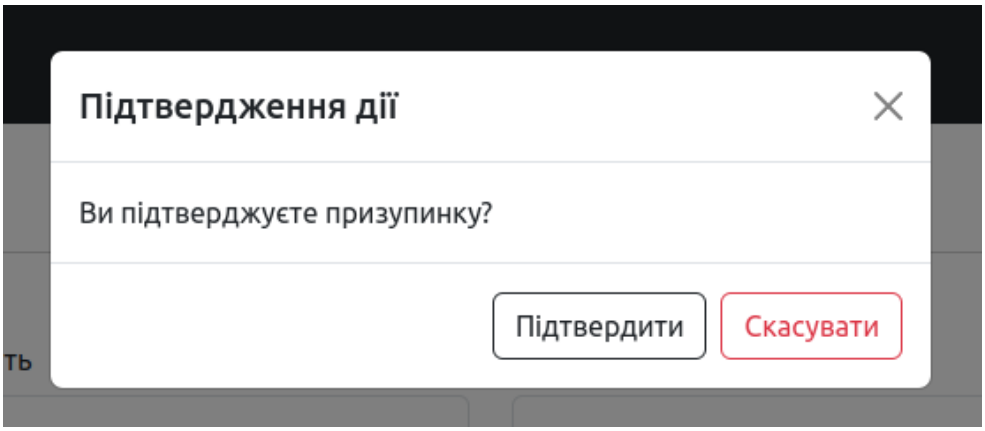

Рисунок *31* — *Спливаюче вікно підтвердження зміни статусу*

Якщо користувач підтвердить дію, то статус буде успішно замінено та відобразиться новий (див. Рис. 32). При скасуванні вікно з запитом закриється.

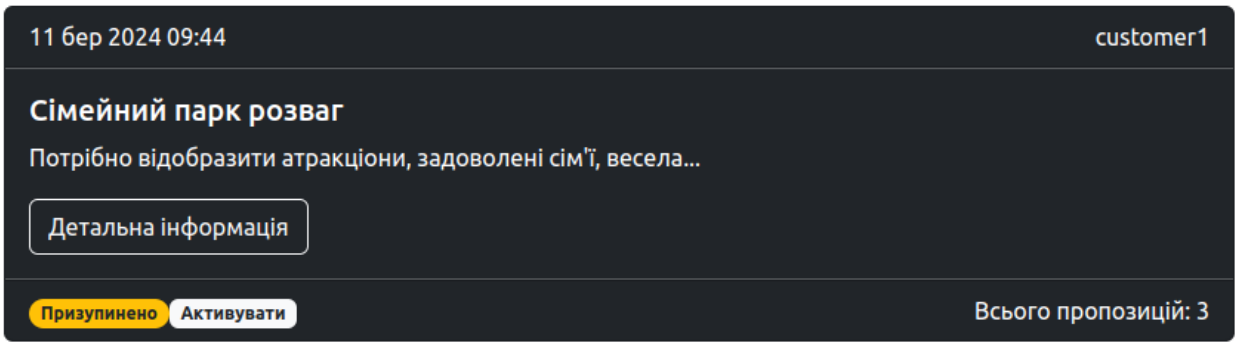

Рисунок *32* — *Приклад оновленого статусу*

В адмін-панелі модератору надана можливість передивлятися списки замовлень та пропозицій (див. Рис. 33, 1) фільтровані за статусами (див. Рис. 33, 3). Далі для редагування він обирає її (див. Рис. 33, 2).

Рисунок 33 — *Інтерфейс модератора (списки та фільтр пропозицій/замовлень)*

На формі перегляду/редагування модератор переглядає всю інформацію в режимі читання та може змінити тільки статус на: На модерації, Відхилено, Підтверджено (див. Рис. 34).

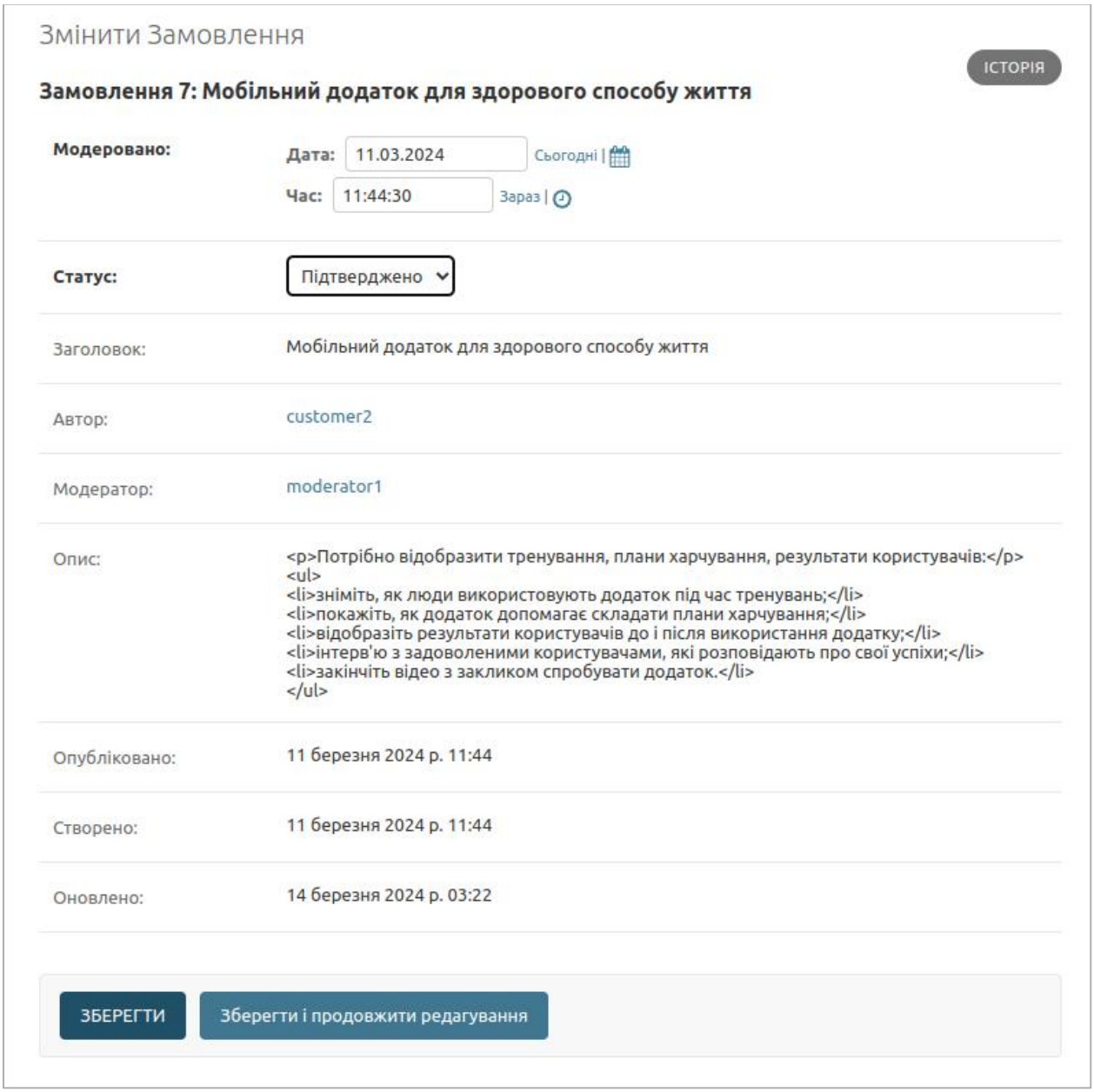

Рисунок *34 — Інтерфейс модератора (перегляд замовлення та зміна статусу)*

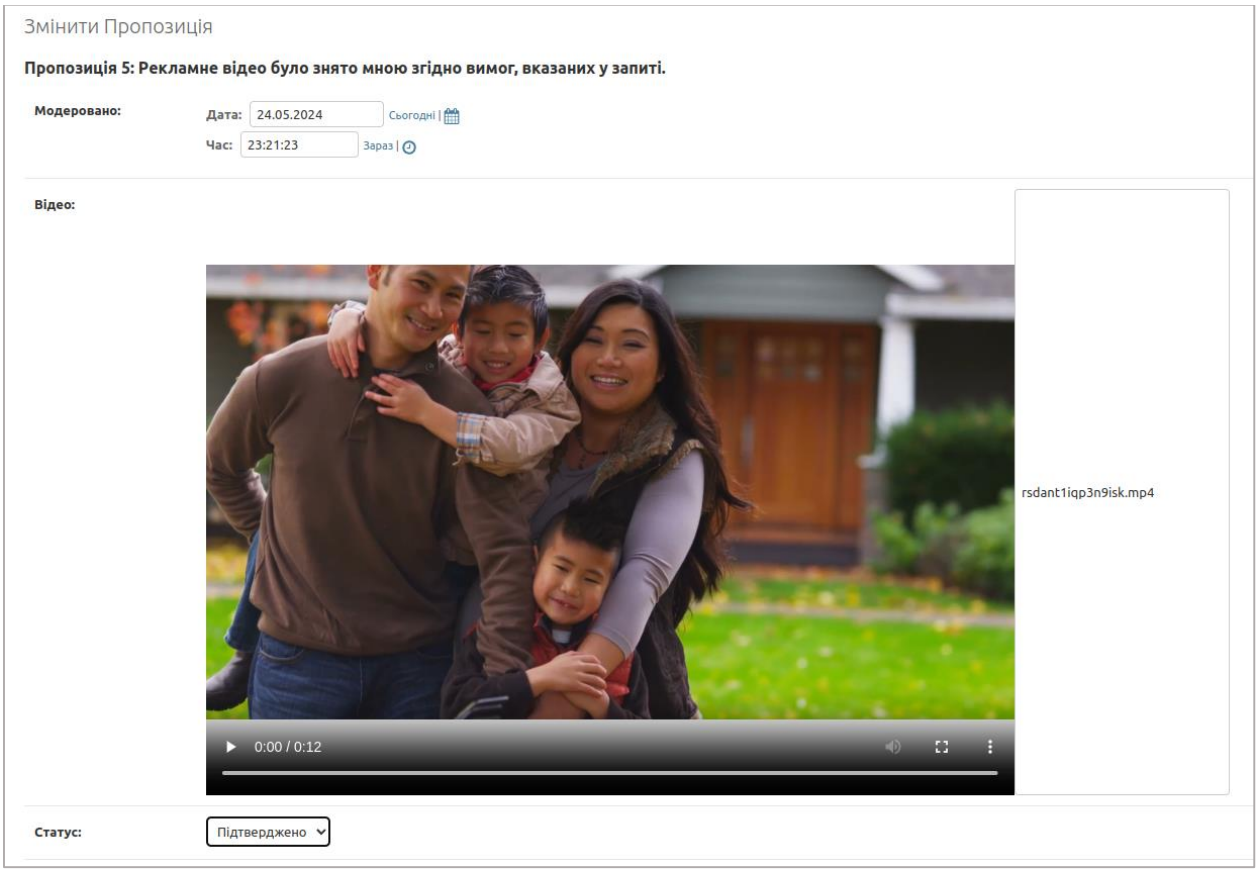

Рисунок *35 — Інтерфейс модератора (перегляд пропозиції та зміна статусу)*

ВИСНОВКИ

1. Зроблено огляд існуючих аналогічних продуктів з метою визначення вимог для розробки зручного веб-сервісу для користувачів.
2. Проаналізовано існуючі засоби розробки для вибору оптимального рішення.
3. Досліджено переваги та недоліки популярних фреймворків для розробки серверної та клієнтської частин, сучасних реляційних систем управління базами даних. Вивчено React та Django, визначені переваги комбінації обраних технологій для розробки веб-сервісу.
4. Застосовано засоби розробки для серверної частини на Python/Django, що взаємодіє з базою даних SQLite, та для клієнтської частини на JavaScript/React.
5. Розроблено програмний веб-застосунок, який забезпечує взаємодію замовників та виконавців в рамках поставленої задачі, надає можливості замовнику створювати замовлення на рекламу, а виконавцю завантажувати відео-контент в якості пропозиції на замовлення замовника.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Нові технології реклами в інтернеті 2023 році: веб-сайт. URL: https://www.mistrakov.eu/2023/03/28/novi-tehnologiyi-reklami-v-interneti-2023-roci/ (дата звернення: 25.05.2024 р.)
2. Video Marketing Statistics: Trends for 2024 That You Can’t Ignore: веб-сайт. URL: https://quickframe.com/blog/video-marketing-statistics/ (дата звернення: 25.05.2024 р.)
3. What is video advertising and why should marketers care?: веб-сайт. URL: https://martech.org/why-we-care-about-video-advertising/ (дата звернення: 25.05.2024 р.)
4. About Veed.me: веб-сайт. URL: https://pitchbook.com/profiles/company/98927-47#overview/ (дата звернення: 28.05.2024 р.)
5. About 90 Seconds: веб-сайт. URL: https://90seconds.com/about/ (дата звернення: 28.05.2024 р.)
6. About Tongal: веб-сайт. URL: https://pitchbook.com/profiles/company/55929-79/ (дата звернення: 28.05.2024 р.)
7. Вступ до Python фреймворків: які вони бувають і для чого використовуються: веб-сайт. URL: https://foxminded.ua/freimvorky-python/ (дата звернення: 29.05.2024 р.)
8. Differences Between Django vs Flask: веб-сайт. URL: https://www.geeksforgeeks.org/differences-between-django-vs-flask/ (дата звернення: 29.05.2024 р.)
9. Antonio Melé. Django 5 By Example - Fifth Edition: Build powerful and reliable Python web applications from scratch. Packt Publishing, 2024. 820p.
10. Miguel Grinberg. Flask Web Development: Developing Web Applications with Python 2nd Edition. O'Reilly Media, 2018. 312p.

11. Jesper Wisborg Krogh. MySQL Connector/Python Revealed: SQL and NoSQL Data Storage Using MySQL for Python Programmers 1st ed. Edition. Apress, 2018. 538p.
12. Luca Ferrari and Enrico Pirozzi. Learn PostgreSQL - Second Edition: Use, manage and build secure and scalable databases with PostgreSQL 16 2nd ed. Edition. Packt Publishing, 2023. 744 p.
13. S BASU, Learn SQLite with Python in 24 hours For Beginners - Simple, Concise & Easy Guide To Using Database with Python Paperback. May 20, 2021. S BASU, 2021. 80p.
14. ТОП-10 фреймворків JavaScript у 2023 році: веб-сайт. URL: https://spacelab.ua/articles/top-10-frejmvorkiv-javascript-u-2023-roci/ (дата звернення: 01.06.2024 р.)
15. Alex Banks and Eve Porcello. Learning React: Functional Web Development with React and Redux 1st edition. O'Reilly Media, 2017. 348p.
16. Документація до Vue: веб-сайт. URL: https://vuejs.org/guide/introduction.html (дата звернення: 01.06.2024 р.)
17. Документація до Angular: веб-сайт. URL: https://angular.dev/overview (дата звернення: 01.06.2024 р.)
18. Переверзова Є.А., студентка 4 курсу ІННІ ім. Ю.М. Потебні ЗНУ. Наук. кер.: к.ф.-м.н., доц. Скрипник І.А. «Створення веб-сервісу для надання рекламних послуг з використанням фреймворків React та Django». Збірник наукових праць студентів, аспірантів і молодих вчених «Молода наука-2024». Запоріжжя : ЗНУ, 2024. С.199-200.

Декларація
академічної доброчесності
здобувача ступеня вищої освіти ЗНУ

Я, Переверзова Єва Андріївна, студентка 4 курсу, форми навчання денної, Інженерного навчально-наукового інституту, спеціальність 121 Інженерія програмного забезпечення, адреса електронної пошти ipz20bd-118@stu.zsea.edu.ua, — підтверджую, що написана мною кваліфікаційна робота на тему «**Створення веб-сервісу для надання рекламних послуг з використанням фреймворків React та Django**» відповідає вимогам академічної доброчесності та не містить порушень, що визначені у ст.42 Закону України «Про освіту», зі змістом яких ознайомлений.

- заявляю, що надана мною для перевірки електронна версія роботи є ідентичною її друкованій версії;

згоден на перевірку моєї роботи на відповідність критеріям академічної доброчесності у будь-який спосіб, у тому числі за допомогою інтернет-системи, а також на архівування моєї роботи в базі даних цієї системи.

Дата 17.06.2024 _____ Переверзова Єва Андріївна
(студент)

Дата 17.06.2024 _____ Скрипник Ірина Анатоліївна
(науковий керівник)