МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютеризованих систем управління

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри

_____ Литвиненко О.Є.
"_____" _____2022 р.

КВАЛІФІКАЦІЙНА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ
"МАГІСТР"

Тема: Програмне забезпечення для створення онлайн - оголошень з продажу, покупки або обміну товарами та послугами

Виконавець: Форостюк Юрій Сергійович

Керівник: доцент, к.ф.-м.н. Нечипорук Віталій Володимирович

Нормоконтролер: Тупота Євгеній Вікторович

Київ 2022

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет Кібербезпеки, комп'ютерної та програмної інженерії
Кафедра комп'ютеризованих систем управління
Спеціальність 123 «Комп'ютерна інженерія»
(шифр, найменування)
Освітньо-професійна програма «Системне програмування»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Литвиненко О.Є.
«___»_____2022 р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи

Форостюка Юрія Сергійовича

1. Тема кваліфікаційної роботи (проекту): «Програмне забезпечення для створення онлайн - оголошень з продажу, покупки або обміну товарами та послугами»
   затверджена наказом ректора від 05.10.2022 р. № 1861/ст
2. Термін виконання: 01.09.2022 р. до 15.11.2022 р.
3. Вихідні данні до проекту: для створення засобу використати середовище розробки *PhpStorm*, фреймворк *Laravel 8*, мову програмування *PHP 7.4*, програмування *JavaScript*, СУБД *MySQL*, стандарти ДСТУ та вимоги університету до написання та оформлення дипломних робіт.
4. Зміст пояснювальної записки:
   1. Аналіз предметної області кваліфікаційної роботи
   2. Функціональні можливості програмного засобу.
   3. Архітектура та структура програмного засобу
   4. Розробка програмного засобу.
5. Перелік обов'язкового графічного матеріалу:
   1. Діаграма прецедентів.
   2. Діаграма взаємодії програмної системи.

3. Схема бази даних.
4. Діаграма компонентів.
5. Приклади користувацького інтерфейсу.

6. Календарний план-графік

№ пор	Завдання	Термін виконання	Відмітка про виконання
1.	Ознайомлення з постановкою задачі та вивчення літератури. Написання 1 розділу, представлення керівнику.	08.09.2022 – 17.09.2022	
2.	Написання 2 розділу, представлення керівнику.	18.09.2022 – 10.10.2022	
3.	Написання 3 розділу, представлення керівнику.	11.10.2022 – 01.11.2022	
4.	Загальне редагування та друк пояснювальної записки, графічного матеріалу.	04.11.2022 – 05.11.2022	
5.	Проходження нормоконтролю, перепліт пояснювальної записки.	11.11.2022 – 13.11.2022	
6.	Розробка тексту доповіді. Оформлення графічного матеріалу для презентації.	14.11.2022	
7.	Отримання відгуку керівника, рецензії.	15.11.2022 – 16.11.2022	
9.	Захист кваліфікаційної роботи	22.11.2022- 24.11.2022	

7. Дата видачі завдання 08.10.2022.

Керівник кваліфікаційної роботи _____Нечипорук Віталій Володимирович
(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання _____Форостюк Юрій Сергійович____

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи «Програмне забезпечення для створення онлайн-оголошень з продажу, покупки або обміну товарами та послугами»: 72 с., 35 рис., 1 табл., 30 інформаційних джерел.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, СИСТЕМА КЕРУВАННЯ ОНЛАЙН-ОГОЛОШЕННЯМИ, МЕТРИКИ, СЕРЕДОВИЩЕ РОЗРОБКИ *PHPSTORM*, ФРЕЙМВОРК *LARAVEL*, МОВА ПРОГРАМУВАННЯ *PHP 7.4*, МОВА ПРОГРАМУВАННЯ *JAVASCRIPT*, СУБД *MYSQL*

Об’єкт дослідження – онлайн-оголошення.

Предметом дослідження є система керування онлайн-оголошеннями.

Мета дослідження – розробка програмного засобу для керування онлайн-оголошеннями товарів та послуг.

Методи дослідження – набір інструкцій та команд мов програмування *PHP* та *JavaScript*, сучасні методи розробки вебдодатків, концепція клієнт-серверної архітектури, база даних *MySQL*, сучасний фреймвор *Laravel*.

Розробка засобу проводилася у середовищі розробки *PhpStorm*, з використанням фреймворку *Laravel 8*, мова програмування *PHP 7.4* та *JavaScript*, СУБД *MySQL*.

Результатом виконання кваліфікаційної роботи є розроблений додаток, який призначений для створення онлайн-оголошень. Додаток забезпечує користувачам швидкий пошук актуальних послуг та товарів, зручну навігацію та можливість будь-якому зареєстрованому користувачу додавати свої оголошення.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ. ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ПЗ – програмне забезпечення

ПП – програмний продукт

ПС – програмна система

СУБД - Система управління базами даних

XHTML – англ. *Extensible Hypertext Markup Language* – розширювана мова розмітки гіпертексту.

MySQL – вільна система керування реляційними базами даних.

PHP – PHP: Hypertext Preprocessor — популярна скрипкова мова.

SQL – англ. *Structured Query Language* — мова структурових запитів.

UML – англ. *Unified Modeling Language* — уніфікована мова моделювання.

XML – англ. *EXtensible Markup Language* — розширювана мова розмітки.

БД – база даних.

Version Control System, VCS – система контролю версій

ЕОМ – електронна обчислювальна машина – загальна назва для обчислювальних машин.

ПЗ – Пояснювальна записка.

ВД – вебдодаток

ВНЗ – вищий навчальний заклад

МП – мова програмування

ПЗ ООТП – ПЗ онлайн-оголошень товарів та послуг

ПЗ – програмне забезпечення

HTML – Hyper Text Markup Language

CSS – Cascading Style Sheets

WWW – World Wide Web

JS – Java Script

ВСТУП

Актуальність даної роботи обумовлена тим, що дозволяє скоротити час пошуку товарів або послуг, як для виконавця і продавця, так і для покупця. Також необхідно зазначити, що дана система не призначена для використання одним замовником чи фірмою, а навпаки, може використовуватися декількома, тому що вона може гнучко налаштовуватися в залежності від потреб.

Основними задачами, що розглядаються «ПЗ для створення онлайн - оголошень з продажу, покупки або обміну товарами та послугами», є:

1. Можливість редагування інформаційної спрямованості вебдодатку.
2. Спрощення пошуку, замовлення та додавання певних послуг для різних користувачів.
3. Спрощення пошуку, замовлення та додавання певних товарів для різних користувачів.

Актуальність теми. Слідом за технологічним розвитком сучасного суспільства, пошук товарів чи послуг за допомогою паперових джерел (газети, журнали, білборди, рекламні оголошення і т.д.) втрачає свою актуальність в рамках зростання попиту на ринку. Таким чином, застарілі методи пошуку демонструють загальну неефективність та деяку відсталість, стосовно актуальності різних пропозицій, та їх доступність по місцезнаходженню.

Програмне забезпечення онлайн-оголошень з плином часу стає більш актуальним, і все більше користувачів шукають в мережі *Internet* різні аналоги вебдодатків онлайн-оголошень, або шукають потрібні товари та послуги в різних соціальних мережах. Існуючі рішення дуже часто мають такі недоліки як: не повний функціонал онлайн-оголошень, перенавантаження рекламою, не зручним інтерфейсом, завищеною ціною на підвищення популярності оголошення, або мають інші недоліки, що не так помітні з першого погляду.

Таким чином розробка доступного, надійного і повнофункціонального програмного забезпечення для створення онлайн-оголошень є актуальною задачею на даний момент.

Мета і завдання виконання кваліфікаційної роботи. Мета кваліфікаційної роботи – розробка програмного забезпечення для створення онлайн оголошень. Для досягнення поставленої мети потрібно дослідити принципи роботи вебдодатків, а також перенести основну логіку оголошень на функціонал вебдодатку. Потрібно визначити які труднощі виникають в користувачів при використанні сервісів онлайн-оголошень. Проаналізувати існуючі програмні рішення, визначити наскільки вони відповідають потребам користувачів, порівняти їх функціонал, виділити переваги та недоліки.

Розробити архітектуру та алгоритми програмного забезпечення, які будуть відповідати вимогам функціонування онлайн-оголошень, та якісно реалізувати поставлені задачі. Обрати середовище розробки, мови програмування, фреймворки та інші засоби для розробки програмного забезпечення.

Розробити вебдодаток, за допомогою якого буде надано можливість створювати онлайн-оголошення для товарів та послуг. Забезпечити зручний та привабливий інтерфейс для взаємодії користувача з функціоналом вебдодатку. Реалізувати функціонал, який буде вигідно відрізнятися серед існуючих програмних рішень.

Об’єкт і предмет дослідження. Об’єктом дослідження даної роботи є процес створення та пошуку онлайн-оголошень для товарів та послуг. Предметом даної роботи є онлайн-оголошення для товарів та послуг.

Методи дослідження. В ході розробки програмного забезпечення для досягнення поставленої в роботі мети використовується набір інструкцій та команд мови програмування *PHP* та *JavaScript*. Розробка відбувається у інтегрованому середовищі розробки *JetBrains PhpStorm,* при цьому використовуються такі інструменти як *GitHub*, *OpenServer*, *NodeJS*, та *Google Chrome DevTools*. Для забезпечення найбільшого функціонування та безпеки, використовується фреймворк *Laravel,* що наразі є одним з найбільш використовуваних фреймворків мови *PHP*, та надає широкий вибір розширень та пакетів.

В ході розробки програмного забезпечення для створення онлайн-оголошень з продажу, покупки або обміну товарами та послугами значною мірою

використовувалися власні ідеї побудови архітектури проекту та програмні алгоритми. Було використано і розширено один з найпопулярніших архітектурних шаблонів, та різні популярні підходи до розробки програмного забезпечення на базі фреймворку *Laravel*. Також були використані популярні пакети *Laravel* для реалізації допоміжних функцій, наприклад, локалізація текстів та генерування мапи сайту.

Наукова новизна отриманих результатів. В ході виконання кваліфікаційної роботи отримано програмне забезпечення для створення онлайн-оголошень з продажу, покупки або обміну товарами та послугами. Завдяки широкому та потужному функціоналу, зручному інтерфейсу користувачам набагато простіше та зрозуміліше стало додавати, редагувати, видаляти та знаходити онлайн-оголошення для товарів та послуг. За допомогою розробленого програмного забезпечення відбувається взаємодія з базою даних в якій зберігається вся інформація. Також створено механізм відновлення даних, що в разі втрати якихось даних, допоможе повернути останній актуальний стан бази даних. Реалізовано підтримку кроссбраузерності та кроссплатформеності, що дозволить використовувати повний функціонал програмного забезпечення на різних пристроях.

Практичне значення отриманих результатів. Розроблене програмне забезпечення може використовувати будь-який замовник, як рекламне агентство, так і певний приватний підприємець. Ця задача досягається тим, що дане програмне забезпечення може налаштовуватися в широких рамках, що дасть можливість охопити будь-який обсяг користувачів за різними критеріями. Отже, з використанням розробленого програмного забезпечення замовник зможе значно збільшити широту охоплення різних груп користувачів, збільшити популярність онлайн-оголошень і зменшити витрати на різні паперові носії інформації, через які відбувалось поширення оголошень.

РОЗДІЛ 1
АНАЛІЗ ІСНУЮЧОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ПРОДАЖУ ТОВАРІВ ЧЕРЕЗ ІНТЕРНЕТ

1.1. Аналіз та структура вебдодатку

Розробка вебдодатків відбувається за еволюцією, подібною до тієї, що спостерігається для програмних систем: виробництво переходить від художньої фази, заснованої на висококваліфікованих майстрах, до промислової фази, в якій якість контролюється шляхом впровадження структурованого робочого процесу. Така зміна все ще триває для вебдодатків, і існує попит на методології, інструменти та моделі, які будуть використані в рамках нового процесу розробки.

Вебсистеми, як правило, швидко розвиваються і зазнають частих модифікацій через нові технологічні та комерційні можливості, а також відгуки користувачів. З таких причин ітераційні моделі процесу розробки, засновані на уявленні про швидке створення прототипів і безперервні зміни, здається, відповідають умовам, в яких створюються та підтримуються вебсайти.

У контексті ітераційного процесу роль аналізу є вирішальною, оскільки працююча система дуже скоро оживає. Результати аналізу можуть бути використані для підтримки розуміння та модифікації, оскільки вони забезпечують високий рівень уявлення про існуючу систему та можуть відповісти на запитання про її організацію та функції.

В кінці кожної ітерації програма тестується. Коли потрібно досягти високих стандартів якості, може бути доречним доповнити звичайне функціональне тестування «чорного ящика» структурним тестуванням «білого ящика». Тестування білого ящика використовує внутрішню структуру вебпрограми для визначення критеріїв покриття. Тому аналіз є обов’язковою умовою, оскільки він дозволяє визначити точки тестування (наприклад, гілки, потоки даних тощо). Це також відправна точка, коли тестові випадки генеруються автоматично або

напівавтоматично.

Розробка вебресурсу – це процес створення вебсторінок або сайтів. Вебсторінки створюються з використанням *HTML, CSS* і *JavaScript* і т.д. Сторінки можуть містити простий текст і графіку, нагадуючи собою статичний документ. Сторінки також можуть бути інтерактивними або відображати інформацію, що змінюється. Створювати інтерактивні сторінки складніше, але вони дозволяють створювати вебсайти з більш наповненим вмістом. Сьогодні більшість сторінок інтерактивні і надають сучасні інтерактивні послуги, такі як корзини *Internet*-магазинів, динамічна візуалізація та навіть складні соціальні мережі [3] розповсюджена структура вебдодатків (рис. 1.1).

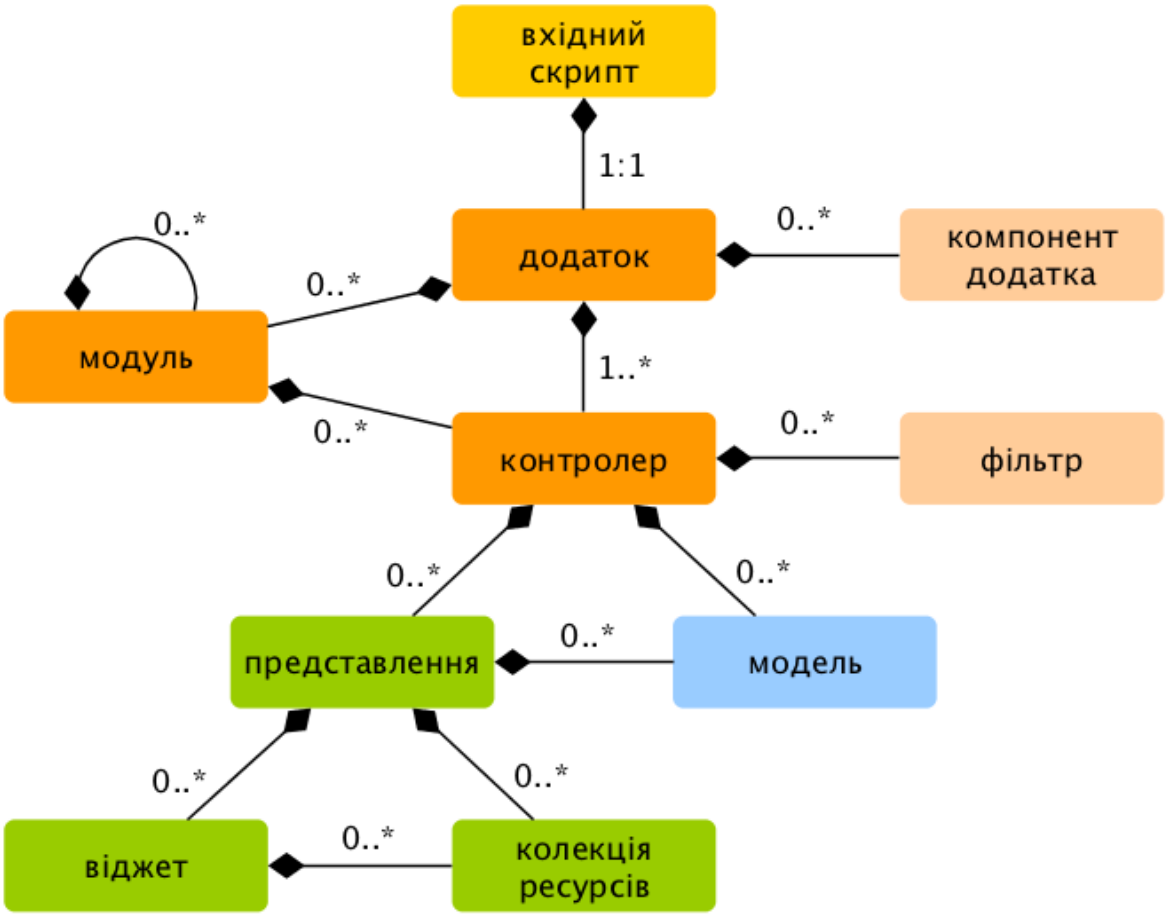

Рис. 1.1. Структура вебдодатку, в більшості фреймоврків

Створенню конкретного вебресурсу передує детальний комплексний аналіз, визначаючий критерії, яким він має відповідати. Процес створення включає шість основних етапів:

1. визначення цілей та задач;
2. розробка структури;
3. розробка дизайн-макетів;
4. *HTML*-верстка;
5. програмування та контроль якості;
6. запуск та оптимізація.

Кожен з перелічених етапів самодостатній, що дозволяє обирати схему роботи і виконавця для кожного з них окремо. Розглянемо кожний етап більш детально.

На етапі проектування формуються бізнес-цілі проекту, що створюється, визначаються вимоги, яким він повинен відповідати, розробляється загальна концепція. Під час роботи на цьому етапі уточнюються вимоги замовника, формується технічне завдання, виконується аналіз цільової аудиторії. Для більш глибокого аналізу можна запросити у замовника відповідні матеріали: брошури, щорічні звіти, зразки продукції, інші супутні дані - все, що допоможе скласти уявлення про те, хто і з якою метою буде відвідувати сайт, які завдання будуть виконуватися на сайті.

Важливо з'ясувати технічні можливості майбутньої основної користувацької аудиторії – пропускну спроможність каналів зв'язку, які використовуються *Internet*-браузери і т.д.

Для знаходження цільової аудиторії доцільно увійти в роль кваліфікованого користувача на аналогічно створюваних *Internet*-ресурсах. Це допоможе виявити нові креативні концепції для того, щоб сайт був більш конкурентоспроможним і «не загубився» серед безлічі інших.

Коли цілі визначені, приступають до складання розширеного плану проекту, що відображає скільки часу, грошей та інших коштів знадобиться для виконання робіт на кожному з наступних етапів. Такий план часто містить інформацію про бюджет проекту, графік робіт (з відповідним розподілом ролей між веброзробниками), технічну документацію, а також розділ «деталей і уточнень», де

обумовлені конкретні аспекти можливих спірних питань. У цей розділ також включають пропозиції по готових розробках і шаблонах.

Наступний етап – розробка структури. Включає в себе зміст сторінки в якій буде розміщено вебресурс, а також – інформаційну стратегію, яка визначає, як організована подача інформації, щоб майбутні користувачі могли легко та зручно використовувати надані ресурси. Головною задачею цього етапу розробки є створення карти, яка відображає зв'язок сторінок та їх найбільш значущі функціональні можливості. Її подають у вигляді блок схеми, на якій кожна сторінка відображається окремим прямокутником, зв’язки між ними означають переходи між сторінками.

Дизайн-макет – це графічне, наглядне зображення елементів сайту. Дизайн-макет повністю втілює візуальну концепцію сайту. Його розробка виконується в одній з графічних програм (у переважній більшості випадків - в *Adobe Photoshop*). У процесі розробки дизайнер керується письмовою угодою на створення дизайн-макету, який заповнюється замовником і містить побажання до дизайну: тип, кольорова гама, наявність тих чи інших графічних елементів, тощо. На цій стадії створюються всі елементи вебдизайну відповідно до стилю подачі інформації та загальної концепції. Головним при дизайні сайту є вміння розробити графічні об'єкти, які б швидко завантажувалися і добре виглядали, незалежно від використовуваного *Internet*-браузера. Часто вдаються до використання готових дизайн-шаблонів, які широко представлені в мережі *Internet* або є вбудованими в різні графічні редактори, такі як, *Microsoft FrontPage* або *Adobe Photoshop*. За допомогою подібних шаблонів сайт створюється за максимально стислий час. Однак слід зазначити, що у такого рішення є ряд істотних недоліків, головний з яких - повторюваність і не унікальність дизайну. Шаблон є оболонкою з мінімальною кількістю інтерактивних елементів і корисних модулів. Тому при виборі шаблону варто звертати увагу не тільки на дизайн, але і на функціональність. Важливим елементом дизайну є графіка, яку умовно розділяють на три категорії:

1. ілюстративна графіка – пояснювальні зображення, схеми та графіки, фото;

2. функціональна графіка – кнопки навігації, лічильники та інші елементи управління;
3. декоративна графіка – естетичні елементи дизайну, такі як фоновий малюнок, заголовки, баннери, рамки.

Така класифікація передбачає чітке розмежовування форматів і передбачає використання певних категорій по максимуму (наприклад, без використання функціональної графіки навігація буде не зручною в той час як без декоративного оформлення сайт буде більш швидко завантажуватися і буде більш простим для сприйняття, не буде перевантажений графікою). *HTML*-верстка макета є наступним кроком після розробки сайту. Верстка - це перетворення створених дизайнером графічних макетів сторінок в *HTML*-код, який би відображався в *Internet*-браузері в точній відповідності до вихідного макету. Складність верстки залежить від складності дизайну. Основними завданнями при верстці є:

- коректність відображення сторінок сайту при різних розширеннях екрану;
- кросбраузерність – однакове відображення сторінок сайту в найбільш популярних браузерах - *Internet Explorer, Mozilla Firefox, Opera, Chrome.*

Програмування - це практична реалізація проекту, інтеграція напрацювань за окремими напрямками. Іншими словами, це процес побудови функціональних інструментів для наповнення і обробки даних. Програмування визначає наскільки стабільним і захищеним буде функціонування сайту. Вибір платформи, технологій і грамотного підходу до програмування відіграє істотну роль. На даному етапі важливо визначитися з підходом до створення вебреусурсу: чи буде він статичним або динамічним. Створення вебресурсу, як і будь-якого іншого програмного продукту, зіштовхується з проблемою постійної зміни даних та файлів. Контроль за змінами, що вносяться в проект, допомагають забезпечити системи управління версіями *VCS*, які зберігають попередні версії вихідних файлів проекту, відстежують вироблені в файлах зміни, забезпечують спільну командну роботу над проектом. До найбільш популярних на поточний момент *VCS* відносяться : *SVN, GIT, Microsoft VSS.*

Використання системи контролю версій піднімає загальний рівень якості розробки. По завершенні етапу активного програмування починається етап тестування коректності функціонування створеного вебресурсу: перевірки на наявність граматичних помилок, пропущених картинок, непрацюючих посилань і т.д., а також перевірки функціонування сайту в різних веббраузерах.

Розробка вебресурсу - це комплексний багатокроковий процес, що вимагає знання безлічі різних технологій і мов програмування, вміння працювати з базами даних, використовувати безліч інструментальних засобів і програмних пакетів. Підсумовуючи все вищесказане, слід при створенні вебресурсу пройти всі перелічені етапи та визначити задачі, параметри та цільову аудиторію.

1.2. Призначення та застосування «ПЗ для створення онлайн – оголошень з продажу, покупки або обміну товарами та послугами»

Сьогодні інформаційні технології все більше використовуються у всіх сферах життя та діяльності. Не минув цей процес таку сферу, як оголошення про різні товари та послуги.

Однак наразі більшість людей все рівно веде пошук потрібних товарів або послуг за допомогою паперових джерел, або через знайомства. Що негативно відображається на швидкості пошуку, актуальності існуючих пропозицій.

Процедура продажу, купівлі або обміну товарів та послуг має бути легкою, та доступною для більшості користувачів.

Програмне забезпечення для створення онлайн-оголошень, це складна система, що об’єднує в собі різноманітні процеси керування даними, і в кінцевому результаті спрощує користувачу пошук задовольняючих його варіантів послуг чи товарів.

Важливою частиною програмного забезпечення є користувацький інтерфейс, та кількість активних користувачів. Зручний інтерфейс забезпечує користувачам простий та зрозумілий пошук цікавих товарів чи послуг, та додавання власних

пропозицій. А від кількості користувачів, в підсумку, залежить наповненість ресурсу різними пропозиціями.

Також однією з найважливіших частин програмного забезпечення є адміністративна панель, з якої власники ресурсу зможуть гнучко налаштувати тематику та базове наповнення інформацією.

Програмне забезпечення для створення онлайн-оголошень, це складна система, що об’єднує в собі різноманітні процеси керування даними, і в кінцевому результаті спрощує користувачу пошук задовольняючих його варіантів послуг чи товарів.

Важливою частиною програмного забезпечення є користувацький інтерфейс, та кількість активних користувачів. Зручний інтерфейс забезпечує користувачам простий та зрозумілий пошук цікавих товарів чи послуг, та додавання власних пропозицій. А від кількості користувачів, в підсумку, залежить наповненість ресурсу різними пропозиціями.

Також однією з найважливіших частин програмного забезпечення є адміністративна панель, з якої власники ресурсу зможуть гнучко налаштувати тематику та базове наповнення інформацією.

В даний час на ринку України є декілька ресурсів, що розділивши «зони відповідальності» утримують лідируючі позиції, і не дозволяють невеликим за обсягом, або зосередженим на якійсь певній частині споживачів компаніям увійти на даний ринок.

Метою випускної кваліфікаційної роботи є розробка вебдодатка онлайн-оголошень, що зможе гнучко налаштовуватися під різні групи користувачів.

В ході розробки повинні були вирішені наступні завдання:

1. Аналіз існуючих аналогів вебдодатків онлайн-оголошень.
2. Збір і аналіз інформації про сучасні технології управління даними та рішень на базі яких створені існуючі аналоги.
3. Проектування архітектури та інтерфейсу вебдодатку, що задовольняє функціоналу аналогів, та буде інтуїтивно зрозумілий користувачу.
4. Реалізація вебдодатку «*VarAn*» (скороч. від англ. *various announcements*).

Розглядаючи аналоги різних дошок онлайн-оголошень, можливо виділити основні елементи інтерфейсу, що допоможе взаємодії з користувачем:

1. Каталог категорій.

Каталог включає в себе відображення різнорівневих категорій, при переході на яку застосовується фільтр, і відображаються товари або послуги тільки потрібних категорій, а також дає можливість більш точно обирати потрібні категорії.

2. Фільтр пошуку.

Фільтр включає в себе різні обмеження, по типу ціни, дати, категорій, що допомагають користувачу визначитися з потрібними обмеженнями для пошуку.

3. Різні типи товарів чи послуг.

Під типами мається на увазі позначення певних товарів чи послуг такими мітками, як «ТОП», «Популярне» і т.п..

4. Локалізація усього контенту.

Дуже важлива частина, тому що не завжди користувач може в повній мірі володіти мовою, що є основною, і в такому випадку має бути альтернатива.

5. Взаємодія з адміністрацією.

Важливий елемент, тому що у вебдодатку в обов’язковому порядку має бути функціонал зв’язку для улагодження різних ситуацій.

Проект програмного забезпечення для створення онлайн-оголошень має в обов’язковому порядку реалізувати такі функції, та мати гнучкість налаштування під різні типи оголошень та товари.

1.3. Порівняльний аналіз «ПЗ для створення онлайн - оголошень з продажу, покупки або обміну товарами та послугами»

OLX (англ. ***O**n**L**ine e**X**change*) — платформа онлайн-оголошень, яка об'єднує людей для покупки, продажу або обміну товарами та послугами. Станом на 2018 рік на майданчику зареєстровано 1,5 млн продавців, розміщено понад 11 млн оголошень і кожну хвилину додається близько 100 нових. Оголошення

класифікуються за такими категоріями, як «Дитячий світ», «Нерухомість», «Транспорт», «Запчастини для транспорту», «Робота», «Тварини», «Дім і сад», «Електроніка», «Бізнес та послуги», «Мода і стиль», «Хобі, відпочинок і спорт», «Віддам безкоштовно» і «Обмін». *OLX* є найбільш відвідуваним сервісом оголошень в Україні. Кожен другий інтернет-користувач з України відвідує ресурс мінімум один раз на місяць.

Переваги:

- Найбільша платформа України, а отже велика кількість оголошень.
- Умовна безкоштовність.

Недоліки:

- Велика кількість оголошень, мінус в тому, що при такому об'ємі неможливо виділити цікаві пропозиції
- Велика кількість шахраїв
  - Менші шанси продати товар при безкоштовному розміщенні
- Відсутність функціоналу послуг

RIA.com Marketplaces — естонська технологічна компанія зі штаб-квартирою в Таллінні, яка розробляє і керує онлайн-платформами *AUTO.RIA.com, DOM.RIA.com, RIA.com, RIA.by і DomRIA.eu*.

RIA.com Marketplaces OÜ займає зі своїм порталом оголошень 23-е місце в світовому рейтингу порталів, і 5-е місце серед незалежних порталів. Володіє двома дочірніми компаніями в Естонії і Україні з офісами в Таллінні, Києві та Вінниці.

Переваги:

- Розділення порталу на різні направлення, автомобілі, будинки, тощо.
- Умовна безкоштовність.

Недоліки:

- Велика кількість шахраїв.
- Неоптимізований інтерфейс
- Відсутність функціоналу послуг

OGOLOSHA.UA – онлайн-платформа, яка була створена для: покупки, продажу або обміну товарами; пошуку і реклами послуг; найму персоналу і пошуку роботи для себе.

Переваги:

– Наявність функціоналу послуг.

– Швидкість роботи ресурсу

Недоліки:

– Низький рейтинг серед інших платформ.

– Низька швидкість роботи адміністрації.

– Проблеми роботи інтерфейсу на різних платформах

Для формування висновків щодо розробки нової системи необхідно провести порівняння наявних рішень (табл. 1.1) за споживчими характеристиками програмного забезпечення для автоматизованої СУПЗ за вказаною рейтинговою системою:

– Функціональність – оцінюється здатність програмного забезпечення задовольнити функціональні потреби користувача під час виконання задач, пов'язаних із наданням сервісних послуг. Оцінюється балом від 1 (найнижча оцінка) до 10 (найвища оцінка).

– Ергономічність інтерфейсу – одна з основних характеристик при виборі та користуванні програмним засобом, оцінюється як ступінь зручності (розташування елементів керування, палітра кольорів) програмного засобу під час його використання. Оцінюється балом від 1 (найнижча оцінка) до 10 (найвища оцінка).

– Вартість – оцінюється вартість програмного забезпечення за умови вибору найменшого та найдешевшого плану із мінімально можливою кількістю користувачів, замовлень та каналів продажу. Оцінюється балом від 1 (найдорожче) до 10 (найдешевше).

– Розширення – оцінюється здатність програми надавати можливість створювати необмежену кількість продуктів або послуг. Оцінюється за фактом підтримки функції - так чи ні.

– Підтримка користувача – оцінюється за наявністю супровідної документації, цілодобової онлайн-підтримки та можливості пропозиції нових функцій. Оцінюється балом від 1 (відсутність підтримки) до 10 (бездоганна підтримка).

Таблиця 1.1

Рейтингова оцінка

Критерій / ПЗ	*OLX*	*RIA.com Marketplaces*	*OGOLOSHA.UA*
Функціональність	7	6	8
Ергономічність інтерфейсу	9	8	7
Вартість	8	7	8
Розширення	Так	Так	Так
Підтримка користувача	4	5	4

Проаналізувавши існуючі рішення ПЗ ООТП, виникає необхідність розробити і впровадити ПЗ ООТП ВД, за допомогою якого користувач зможе оперативно та з будь-якого пристрою отримувати коректну інформацію про список існуючих товарів та послуг, купувати та обмінювати їх, а також матиме можливість сам створювати нові оголошення товарів та послуг.

«Програмне забезпечення для створення онлайн - оголошень з продажу, покупки або обміну товарами та послугами» розробляється з метою простого та безкоштовного доступу до всього функціоналу ВД для всіх користувачів, а також можливість управління створеними товарами та послугами. Користувачі даного ВД також мають можливість створити свій товар або послугу.

Впровадження ВД:

– Спрощення процесу пошуку актуальних пропозицій на ринку;

– Інформування користувачів про надходження нових замовлень на їх послуги або товари;
– Забезпечення доступу до інформації з будь-якого пристрою;
– Можливість створення товарів та послуг.
– Можливість запропонувати іншим користувачам обмін товарами або послугами.

ВД повинен відповідати наступним критеріям:

– Містити функціонал для менеджменту товарами та послугами;
– Мати можливість фільтрування товарів та послуг за певними критеріями;
– Форма додавання нових товарів та послуг має бути зручною та зрозумілою;
– Надавати користувачу можливість приймати або відхиляти пропозиції обміну, або купівлі його товарів та послуг.

1.4. Висновки до розділу

У даному розділі було проаналізовано та ретельно вивчено усі особливості ПП, які пропонує ІТ-ринок у сфері систем онлайн-оголошень, а також досліджено існуючі системи онлайн-оголошень. Враховуючи переваги та недоліки існуючих рішень, були зроблені висновки щодо проектування та розробки майбутнього програмного забезпечення, а саме – щодо вимог до ВД та його архітектури.

РОЗДІЛ 2
ВИМОГИ ДО РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1. Загальний принцип роботи вебдодатків

Вебдодатки використовують архітектуру «клієнт - сервер». Веб -додаток перебуває на сервері і обробляє запити, які передають йому через Інтернет численні клієнти.

На стороні клієнта вебдодаток працює в браузері, наприклад в *Google Chrome*. Користувальницький інтерфейс вебдодатка передається на клієнтську машину у вигляді сторінок мовою *HTML*, де браузер інтерпретує й відображає їх (рис 1.2).

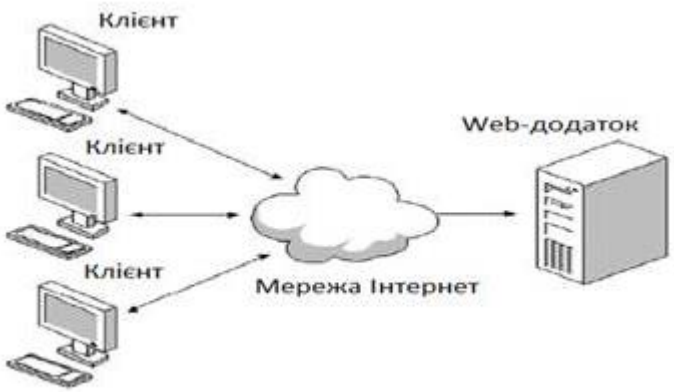

Рис. 1.2. Принцип роботи вебдодатків

На стороні сервера вебдодаток працює під керуванням *IIS* (*Internet Information Services*). *IIS* управляє роботою додатка, передає йому клієнтські запити й повертає клієнтам результати виконання їхніх запитів. Запити і результати їх виконання передаються через Інтернет протоколом *HTTP*.

Протокол - це набір правил, що регламентують взаємодію двох і більше сутностей, що реалізується через середовище, таке, як Інтернет.

Вебдодаток компонує відгук із серверних ресурсів, до яких відноситься:

- код, що працює на сервері (те, що традиційно вважається «додатком» в *Windows*-програмуванні);

– Вебформи, *HTML*-сторінки, графічні файли й інший уміст, що становить інформаційне наповнення додатків.

Вебдодатки багато в чому нагадують традиційні вебсайти, але на відміну від них відображають користувачеві динамічний уміст, який генерується кодом, додатка, а не статичні сторінки, що зберігаються на сервері в готовому вигляді.

Частина вебдодатка, що виконується, здатна робити багато чого з того, чого не можуть статичні вебсайти, а саме:

– приймати дані від користувача й зберігати їх на сервері;
– виконувати для користувача різні дії: розміщати замовлення, робити складні обчислення і витягати інформацію з баз даних (БД);
– ідентифікувати користувача і відображати інтерфейс, побудований відповідно до його профілю;
– відображати постійно змінюваний уміст, наприклад інвентарні списки, оброблювані замовлення і відомості про товари, що відвантажують.

Цей перелік далеко не повний. У принципі, вебдодатки здатні вирішити будь-які уявлені завдання, доступні і клієнт-серверним додаткам. Особливість вебдодатків у тім, що взаємодія між клієнтом і сервером здійснюється через Інтернет.

2.2. Принципи роботи вебдодатку

Будь який користувач, що зареєструвався у вебдодатку має право на безкоштовне розміщення оголошень про товари та послуги в усіх представлених категоріях, для цього він має відкрити відповідну форму, заповнити усі обов’язкові поля, і при необхідності додати пояснювальні зображення та файли.

Після заповнення форми нового проекту до ВД, інформація одразу стає доступна усім користувачам.

Для того, щоб додати нову послугу, наприклад, послуги перукарів з можливістю виїхати до замовника, користувач має заповнити форму додавання нової послуги, позначити, що тип послуги стоїть «З можливість виїзду», при

бажанні додати сертифікати про проходження курсів, потім обрати категорію, до якої має відноситися дана послуга і зберегти зміни, після цього послуга стане доступною для перегляду для всіх користувачів.

Після заповнення форми нового товару або задачі до ВД, інформація відразу стає доступна усім користувачам.

Для того, щоб в рейтингу користувачів він оцінювався як більш кращий спеціаліст, то в користувача є можливість додати до свого профілю різні сертифікати про проходження курсів, тощо.

Адміністрація ВД має можливість гнучко налаштувати всі категорії, що будуть представлені та доступні для всіх користувачів. А також відслідковувати усі виконані послуги та куплені товари за допомогою інтерфейсу, та бачити статистику популярності певних категорій.

У разі необхідності, існує опція редагування інформації про існуючі товари або послуги, з можливим внесенням коректівів до опису.

Накопичення товарів та послуг надає можливість аналізувати діяльність користувачів у вебдодатку та популярність певних категорій і інформації, і на основі даних проводити реформи у інформації вебдодатку.

ПЗ ООТП являє собою складну систему автоматизації процесів менеджменту товарів та послуг у сфері онлайн оголошень є сучасною формою системи онлайн-оголошень.

Відмінною перевагою ПЗ ООТП має бути оперативність доступу до інформації з будь-якого типу пристрою, а також можливість пропонувати різним користувачам влаштовувати обмін товарами або послугами.

2.3. Ключові вимоги

Проаналізувавши ринок Програмне забезпечення для створення онлайн - оголошень з продажу, покупки або обміну товарами та послугами, були сформовані основні критерії, яким має відповідати ПП:

– Інтегрованість – відповідність основній структурі і змісту;

– Адаптивність – можливість подальшого удосконалення й адаптації ВД СУПЗ до вимог окремого замовника;

– Поширюваність – використання ВД на безоплатній основі та невибагливість до технічних характеристик пристроїв;

– Зрозумілість – простота у використанні, наявність інших корисних функцій, використання підказок для спрощення взаємодії з ВД;

– Безпечність – розмежування прав для доступу до інформації та внесення змін;

– Збережуваність – резервне копіювання даних про заходи у разі виходу ВД з ладу;

При розробці ПЗ необхідно забезпечити дотримання наступних вимог:

– Сумісність з усіма сучасними веббраузерами;

– Підтримка Інтернет-технологій;

– Забезпечення належної роботи ВД за умови збільшення потоку інформації;

– Захищеність даних від викрадення;

– Можливість роботи у режимі реального часу;

2.4. Рівні доступу

Для запобігання внесення несанкціонованих змін інформації, ВД передбачає стандартне розмежування прав доступу. В даному ВД реалізоване наступне розмежування:

– «Адміністратор» - особа, яка має можливість повністю керувати функціями ВД, створювати нові категорії для товарів та послуг, вносити зміни до даних, також має доступ до всіх описаних сервісів та додаткових модулів адміністрування БД;

– «Користувач» - особа, яка зареєструвалась у ВД, має стандартний набір користувацьких можливостей;

Для отримання необхідного доступу із ВД, користувачеві необхідно авторизуватися у системі шляхом введення відповідних до рівня доступу логіна і пароля.

2.5. Функціональні можливості

Відповідно до специфіки роботи онлайн-оголошень було розроблено наступний перелік функціональних можливостей, якими повинен володіти майбутній ВД:

- Перегляд списку товарів та послуг;
- Створення нових товарів та послуг;
- Створення нових категорій товарів та послуг;
- Редагування даних про товар або послугу;
- Пропозиція обміну товаром або послугою;
- Підтвердження/відхилення пропозиції обміну;
- Видалення товарів або послуг;
- Перегляд списку користувачів;
- Створення нового користувача;
- Реєстрація своїми силами у ВД;
- Редагування інформації про користувача;
- Видалення користувача;
- Перегляд списку сертифікатів користувачів;
- Подача звернення до адміністрації;
- Пошук товарів та послуг за назвою;
- Фільтрування товарів та послуг та задач за датою, статусом та відображенням;
- Сповіщення про пропозиції обміну або купівлі товарів та послуг

2.6. Впровадження та функціонування системи

Для інтеграції ПЗ ООТП в систему роботи необхідно дотримуватися наступних пунктів:

- Виділення та переніс на сервер для функціонування ПЗ;
- Призначення адміністратора ВД;
- Заповнення інформації про новини та акції.
- Заповнення інформації про можливі категорії товарів та послуг у ВД
- Заповнення інформації про товари та послуги, якщо такі є в наявності
- Створення реклами, що зацікавить та приведе користувачів у ВД
- Забезпечення функціонування ВД у рамках компанії замовника як структурної одиниці;
- Своєчасне оновлення інформації;

2.7. Інтерфейс вебсервісу онлайн-оголошень

Інтерфейс ВД складається із меню для вибору сторінки, на якій знаходиться певна інформація про товари, послуги, перемикання мов ВД, пошук на всіх сторінках, товарах та послугах, або тільки в певній категорії, окремих сторінок: редагування власного профілю, звернення до адміністрації, що також знаходяться в меню.

Передбачена можливість вибору статусу товару або послуги для інформування користувача про доставку, регіон виконання послуги, наявність товару, тощо.

Форма товару включає в себе можливість вибору статусу з наступних:

- В продажі;
- Куплено;

Форма товару включає в себе можливість вибору типу доставки з наступних:

- На пошту
- Самовивіз

Форма послуги включає в себе можливість вибору типу з наступних:

– Виїзна;
– Віддалена;
– Договірна;
– Не виїзна;

Це дає користувачам можливість орієнтування в списках товарів та послуг, та замовляти товари і послуги, що підходять саме йому. А також вказувати ті параметри, які він буде надавати в своєму товарі або послузі.

У ВД в користувача з рівнем доступу «Адміністратор», присутні всі вище названі функції ВД, а також є доступ до адміністративної панелі, де відбувається додавання категорій товарів та послуг, відбувається перегляд користувачів, існуючих товарів та послуг, а також статистика куплених і замовлених послуг.

2.8. Висновки до розділу

В даному розділі подано опис принципу роботи системи, сформовано основні функціональні та нефункціональні вимоги, яких необхідно дотримуватися при розробці ВД. Визначено основні рівні доступу до ВД.

Описано найімовірніший алгоритм впровадження системи у роботу та поради до забезпечення його функціонування.

РОЗДІЛ 3
СТРУКТУРА ТА АРХІТЕКТУРА ВЕБДОДАТКУ

3.1. Принципи проектування архітектури програмної системи

У зв'язку з тим, що комп'ютер-сервер знаходиться окремо від клієнта, для проектування ВД було обрано архітектуру «клієнт-сервер» [8] (рис. 3.1).

У цій архітектурі ІС ділиться на неоднорідні частини - сервер і клієнт БД.

Сервер - це сама СУБД. Він підтримує усі основні функції СУБД: визначення даних, обробку даних, захист даних, підтримку цілісності даних і т.д.

Клієнт - це додаток користувача. Для отримання даних клієнт формує і надсилає запит до віддаленого сервера, на якому розміщена БД. Запит формується на мові *SQL*, що є стандартним засобом доступу до сервера при використанні реляційних моделей даних. Після отримання запиту віддалений сервер направляє його до *SQL*-сервера (сервера БД) спеціальною програмою, що управляє віддаленою БД і забезпечує виконання запиту і видачу його результатів клієнту.

Таким чином, в архітектурі «клієнт-сервер» клієнт надсилає запит на надання даних і отримує лише ті дані, які дійсно були затребувані. Вся обробка запиту виконується на віддаленому сервері.

«Клієнт-серверна» архітектура має наступні переваги:

1. Зниження навантаження на мережу, оскільки тепер в ній циркулює тільки потрібна інформація.
2. Підвищення безпеки інформації, так як обробка запитів всіх клієнтів виконується єдиною програмою, розташованою на сервері. Сервер встановлює загальні для всіх користувачів правила використання БД, управляє режимами доступу клієнтів до даних, забороняючи, зокрема, одночасні зміни одного запису різними користувачами.
3. Зменшення складності клієнтських додатків за рахунок відсутності в них коду, пов'язаного з контролем БД і розмежуванням доступу до неї.

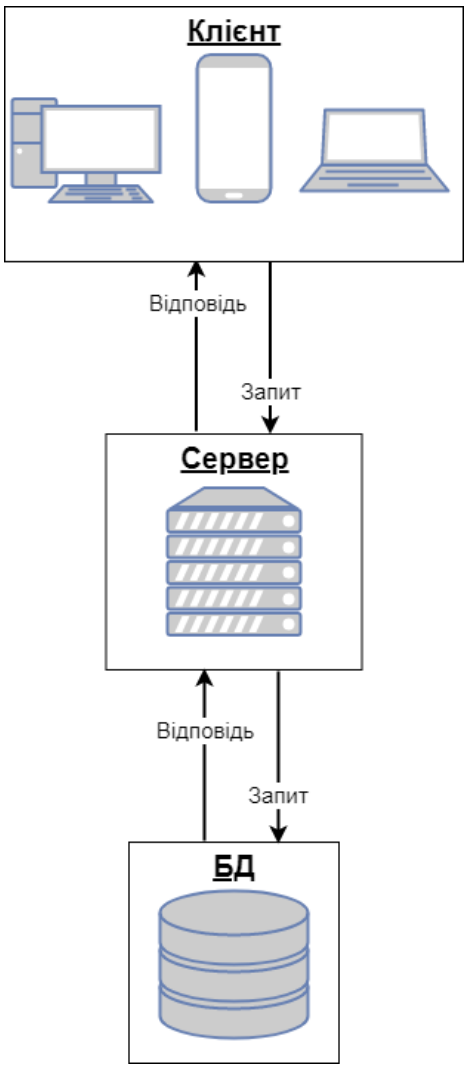

Рис. 3.1. Архітектура програмного комплексу

3.2. Модель клієнта вебдодатку системи онлайн-оголошень

Для доступу до тих чи інших мережевих сервісів використовуються клієнти, можливості яких характеризуються поняттям «товщини». Товщина клієнта визначається конфігурацією устаткування і ПЗ, що є у клієнта.

«Тонкий клієнт» [8] - термін, що визначає клієнта, обчислювальних ресурсів якого достатньо лише для запуску необхідного Інтернет-додатку через *web*-інтерфейс. Інтерфейс такого додатку формується засобами статичного *HTML* (виконання *JS* не обов'язкове), вся прикладна логіка виконується на сервері.

Для роботи «тонкого клієнта» (рис. 3.2) досить лише забезпечити можливість запуску *web*-браузера, у вікні якого і здійснюються всі дії. З цієї причини *web*-браузер часто називають "універсальним клієнтом".

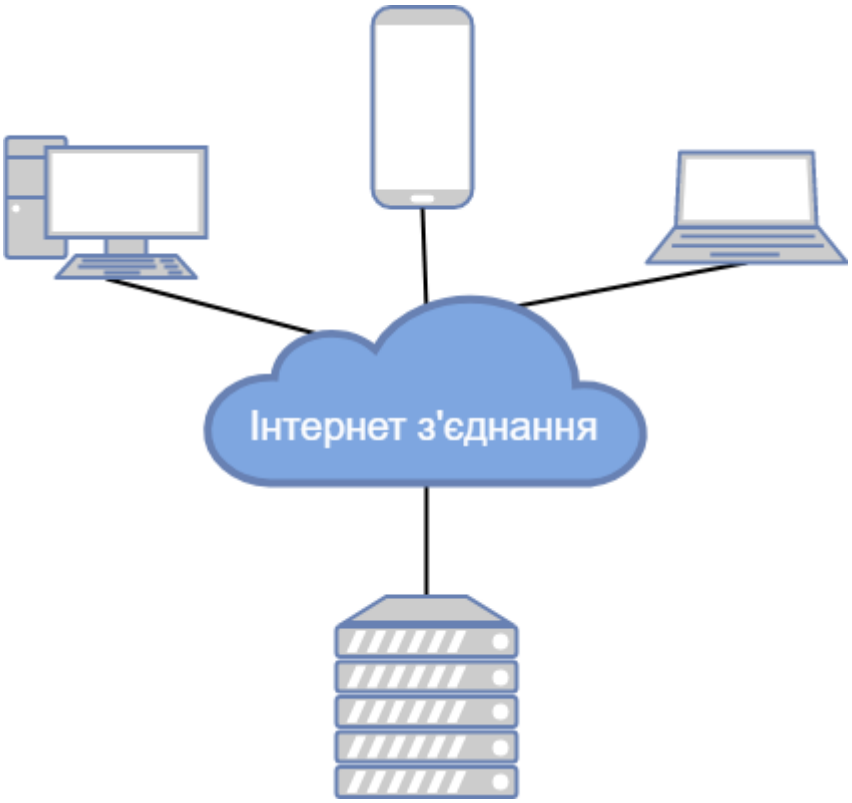

Рис. 3.2. Принцип роботи клієнта

3.3. Архітектура вебсервісу оголошень

За допомогою мови *UML* [9] буде представлено архітектуру ВД, що розробляється, у вигляді наступних діаграм:

Структурні діаграми:

1. Діаграма класів.
2. Діаграма компонентів.
3. Діаграма розгортання.

Діаграма класів - статичне представлення структури моделі. Відображає статичні елементи, такі як: класи, типи даних, їх зміст та відношення.

Відповідно до архітектури «клієнт-сервер», структуру ВД можна представити за допомогою діаграм класів (рис. 3.3-3.6), які відображають логіку та зв’язки у програмному комплексі.

За розподілом обов’язків між клієнтом та сервером архітектуру ВС можна поділити на 3 рівні: (рис. 3.1):

1. Рівень представлення – цей шар відповідає за реалізацію клієнтської частини, містить інтерфейс користувача та методи і класи збору первинних вхідних даних.

2. Прикладний рівень – цей шар реалізує основний функціонал програмного комплексу, пов’язаний із обробкою вхідних даних, а також містить інтерфейси передачі даних на шари представлення та даних.

3. Рівень даних – цей шар відповідає за обробку вхідних запитів та повернення результатів виконання запитів через інтерфейс.

Рис. 3.3. Діаграма класів у загальному вигляді

Кожен з рівнів включає в себе відповідні класи:

1. Рівень представлення включає в себе підсистему «Відображення проектів, задач, користувачів та ролей користувачів», яка збирає дані та передає їх до прикладного рівня (рис. 3.4).

Рис. 3.4. Діаграма класів рівня представлення

2. Рівень представлення включає в себе підсистему «Обробка даних», яка здійснює обробку інформації отриманої з рівня представлення та посилає запити до рівня даних. Оброблені запити з рівня даних передає на рівень представлення (рис. 3.5).

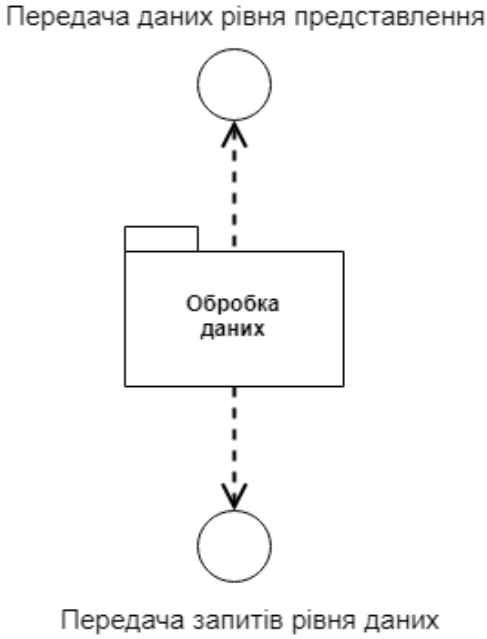

Рис. 3.5. Діаграма класів рівня підсистем шару «Бізнес-логіка»

3. Рівень даних складається з підсистеми «Обробка запитів», яка обробляє запити, отримані з прикладного рівня, та повертає результати виконання запитів через інтерфейс передачі запитів рівня даних до прикладного рівня (рис. 3.6).

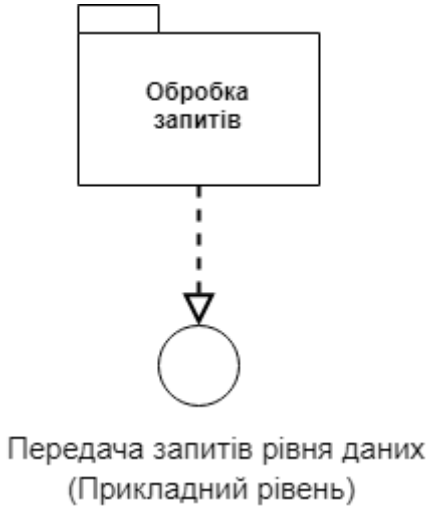

Рис. 3.6. Діаграма класів рівня даних

Діаграма компонент відображає залежності між компонентами програмного забезпечення, включаючи компоненти вихідних кодів, бінарні компоненти, та компоненти, що можуть виконуватись.

Діаграма компонентів, на відміну від раніше розглянутих діаграм, описує особливості фізичного представлення системи.

Діаграма компонентів розробляється для наступних цілей:

– Візуалізації загальної структури вихідного коду ПС.

– Специфікації виконуваного варіанту програмної системи.

– Забезпечення багаторазового використання окремих фрагментів програмного коду.

– Уявлення концептуальної і фізичної схем БД.

Відповідно до наведеної логіки ПЗ, архітектуру ВД можна представити у вигляді діаграми компонентів (рис. 3.7), яка відображає компоненти ВД та їх зв’язки.

Компоненти ВД:

1. Пакет «Клієнт»:

– Компонент «Авторизація»;
– Компонент «Інтерфейс користувача»;
– Компонент «Збір вхідних даних»;

2. Пакет «Прикладний рівень»:
   – Компонент «Обробка вхідних даних»;
   – Компонент «Формування запитів»;
   – Компонент «Перевірка рівня доступу»;
   – Компонент «Обробка результатів виконання запитів»;
   – Інтерфейс «Передача даних рівня представлення»
   – Інтерфейс «Передача даних рівня даних»;
3. Пакет «Рівень даних»:
   – Обробка запитів;
   – Формування пакету результатів виконання запиту;
4. БД

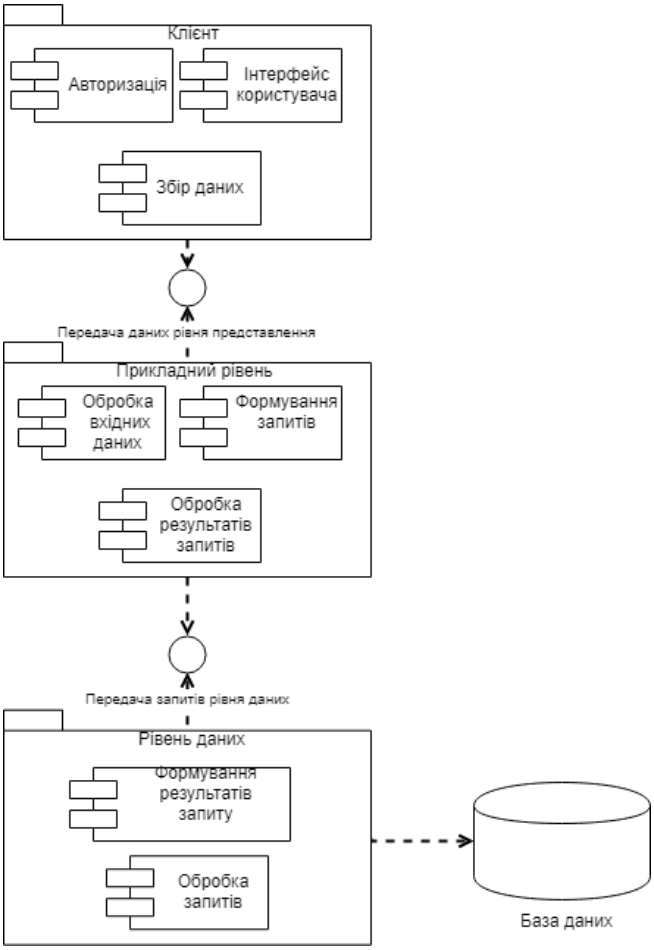

Рис. 3.7. Діаграма компонентів рівня реалізації ВД

Діаграма розгортання - діаграма в *UML*, на якій відображаються обчислювальні вузли під час роботи програми, компоненти, та об'єкти, що виконуються на цих вузлах.

Для візуалізації фізичного розташування компонентів програмного комплексу серед апаратних засобів доцільно використати діаграму розгортання (рис. 3.8).

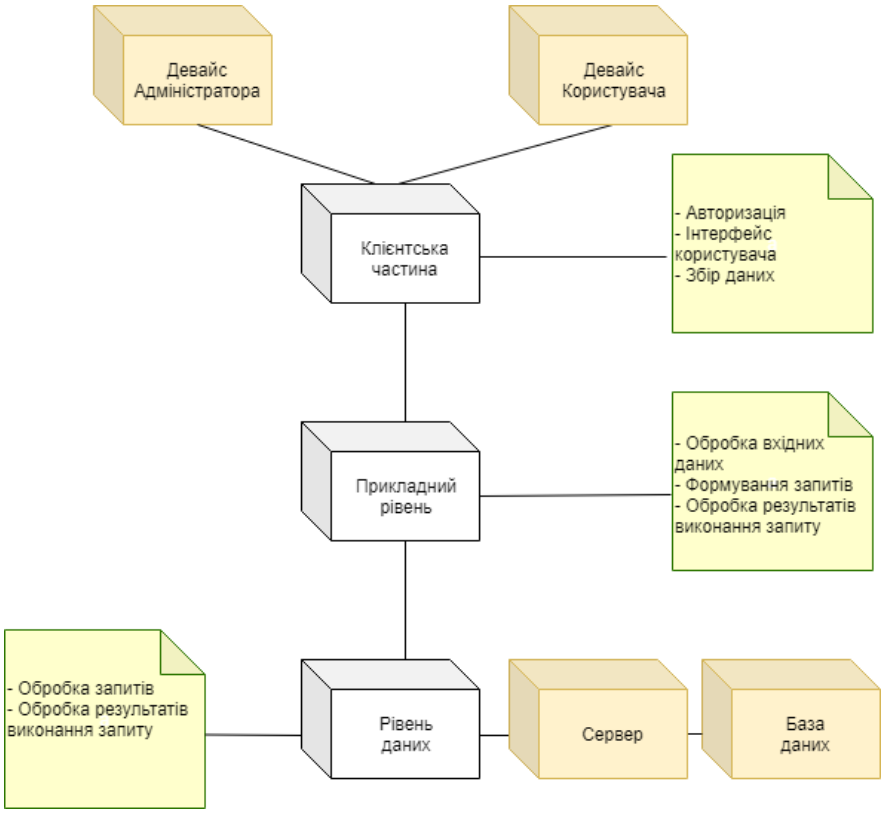

Рис. 3.8. Діаграма розгортання

3.4. Методологія розробки

Добре налагоджене тестування є запорукою працездатного проекту. Тому для реалізації даного веб-додатку була обрана методологія *test-driven development*. *TDD*, *test-driven development* або процес розробки через тестування – це методологія розробки програмного забезпечення, яка ґрунтується на повторенні коротких циклів розробки: спочатку пишеться тест, що покриває бажану зміну, потім пишеться програмний код, який реалізує бажану поведінку системи і

дозволить пройти написаний тест, а потім проводиться рефакторинг написаного коду із постійною перевіркою проходження всіх тестів.

Тестування ПЗ – це процедура, яка дозволяє підтвердити або спростувати працездатність коду та коректність його роботи. При тестуванні додатку передаються вхідні дані і запитується виконання певної команди, після чого проводиться перевірка отриманих результатів на відповідність стандарту, якщо результат відповідає очікуваному – тест вважається пройденим. Ця процедура може бути автоматизована, у цьому випадку перевірка працездатності та правильності роботи програми у порівнянні з ручним тестуванням здійснюється набагато швидше, повноцінніше та фактично частіше.

Методика розробки через тестування полягає в організації автоматичного тестування програм, що розробляються, шляхом написання модульних, інтеграційних і функціональних тестів, що визначають вимоги до коду безпосередньо перед написанням цього самого коду. Спочатку пишеться тест, який перевіряє коректність роботи ще ненаписаного програмного коду.

Цей тест, зрозуміло, не проходить. Після цього розробник пише код, який виконує дії, потрібні для проходження тесту. Після того, як тест успішно пройдено, за необхідністю здійснюється рефакторинг (доробка та переробка) написаного коду, причому в цьому випадку рефакторинг здійснюється вже під контролем проходження тестів, що простіше та надійніше.

Цикл розробки з *TDD*:

- додати тест для нової (ще не реалізованої) функціональності або для відтворення існуючого бага;
- запустити всі тести та переконатися, що новий тест не проходить;
- написати код, який забезпечить проходження тесту;
- запустити тести і переконатися, що всі вони пройшли успішно: проходження нового тесту підтверджує реалізацію нового функціоналу чи виправлення існуючої помилки, а проходження інших дозволяє переконатися, що раніше реалізований функціонал працює, як і раніше, коректно;

– зайнятися рефакторингом та оптимізацією – поліпшення супроводжуваності та швидкодії доцільно здійснювати вже після того, як вдалося досягти перевіреної працездатності;
– перезапустити тести та переконатися, що вони все ще проходять успішно;
– повторити цикл.

Ця методологія дозволяє домогтися створення придатної для автоматичного тестування програми і дуже хорошого покриття коду тестами, оскільки ТЗ перекладається мовою автоматичних тестів, тобто все, що програма повинна робити, перевіряється. Також *TDD* часто спрощує програмну реалізацію: оскільки виключається надмірність – якщо компонент проходить тест, він вважається готовим. Якщо ж існуючі тести проходять, але працює компонент не так, як очікується, це означає, що тести поки не відображають всіх вимог і це привід додати нові тести.

Архітектура програмних продуктів, що розробляються таким чином, зазвичай краще (у додатках, які придатні для автоматичного тестування, зазвичай дуже добре розподіляється відповідальність між компонентами, а складні процедури, що виконуються, декомпозовані на безліч простих). Стабільність роботи програми, розробленого через тестування, також вища за рахунок того, що всі основні функціональні можливості програми покриті тестами та їх працездатність постійно перевіряється. Супроводжуваність проектів, де тестується все або практично все, дуже висока – розробники можуть не боятися вносити зміни до коду, якщо щось не спрацює, то про це повідомлять результати автоматичного тестування.

3.5. Висновки до розділу

Враховуючи особливості функціонування ВД створення онлайн-оголошень з продажу, покупки або обміну товарами та послугами, було зроблено аргументований вибір у бік «Клієнт-серверної» архітектури, відповідно до якої ВД розбито на три рівні: рівень представлення, прикладний рівень та рівень даних. За

допомогою мови *UML* було представлено структуру та архітектуру ВД у вигляді структурних та поведінкових діаграм.

РОЗДІЛ 4
ВИКОРИСТАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОНЛАЙН ПРОДАЖІВ

4.1. Взаємодія користувача з вебдодатком

Для початку розробки свого особистого вебдодатку, потрібно зрозуміти як він буде працювати, щоб правильно визначити його функції та написати їх. На рисунку 2.1 відображено схему роботи вебдодатку.

Рис. 2.1. Взаємодія користувача з вебдодатком

Зрозумівши як працює вебдодаток, можна приступати до його розробки.

4.2. Реалізація серверної частини

Back-end відноситься до розробки на стороні сервера. Він фокусується на базах даних, сценаріях, архітектурі вебсайту. Він містить дії за кадром, які відбуваються під час виконання будь-якої дії на вебсайті. Це може бути вхід в обліковий запис або покупка в інтернет-магазині. Код, написаний розробниками серверної частини, допомагає браузерам спілкуватися з інформацією бази даних.

Рис. 2.2. Структура *Back-end*

Найпоширеніший приклад програмування *Back-end* – це коли ви читаєте статтю в блозі. Шрифти, кольори, дизайн тощо становлять інтерфейс цієї сторінки, тобто *Front-end*. У той час як вміст статті відтворюється з сервера і витягується з бази даних. Це серверна частина програми.

Найпоширеніші мови програмування:

– *Java*

Високорівнева, об’єктно-орієнтована, заснована на класах внутрішня мова, створена, щоб мати якомога менше залежностей реалізації.

Популярні програми : *Spotify, Twitter*.

– *Ruby*

Це загальна, високорівнева та інтерпретована серверна мова.

Популярні програми : *AirBnb, Twitch, Crazy Egg*.

– *Python*

Це високорівнева, об'єктно-орієнтована, інтерпретована серверна мова з динамічною семантикою.

Популярні програми: *Reddit, Quora, Dropbox*.

– *PHP*

Це широко використовувана мова сценаріїв загального призначення з відкритим вихідним кодом, спеціально розроблена для написання вебдодатків

(сценаріїв), що виконуються на вебсервері.

Популярні програми: *Wikipedia, MailChimp, iStockPhoto*

– *JavaScript*

Являє собою структуровану інтерпретовану мову програмування, високорівневий сценарій з динамічним типом і мультипарадигмою..

Популярні програми: *Netflix, Uber*.

– *SQL*

Це специфічна для домену мова, яка використовується в програмуванні та створена для обробки даних у *RDMS*.

Популярні програми: *PostgresSQL, MySQL*

– *Rust*

Це багатопарадигмальна серверна мова, створена для продуктивності та безпеки, особливо для безпечного паралельного використання.

Популярні програми: *One Signal, OVH, 10x Genomics*

– *C/C++*

Це процедурна серверна мова програмування загального призначення, що містить структурне програмування, рекурсію та область видимості лексичних змінних.

Популярні програми: *Adobe Photoshop, Firefox, Bloomberg*

– *NodeJS*

Це середовище виконання *JavaScript* з відкритим кодом.

Популярні програми: *PayPal, eBay, Godaddy*

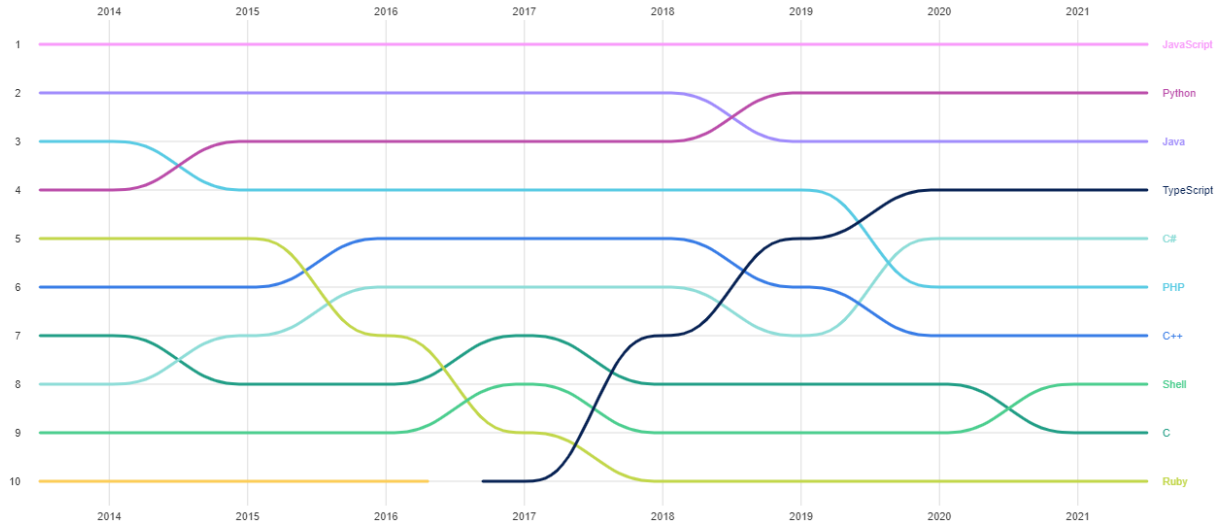

Рис. 2.3. Графік популярності мов програмування

Для реалізації бізнес-логіки та серверної частини ВС використано МП *PHP* [12].

PHP – це широко використовувана мова сценаріїв загального призначення з відкритим вихідним кодом, спеціально розроблена для написання вебдодатків (сценаріїв), що виконуються на вебсервері.

Важливою перевагою *PHP* є те, що *PHP*-скрипти виконуються на стороні сервера. Ви можете навіть сконфігурувати свій сервер таким чином, щоб *HTML*-файли оброблялися процесором *PHP*, так що клієнти навіть не зможуть дізнатися чи отримують вони звичайний *HTML*-файл або результат виконання скрипта.

PHP дозволяє створити якісні вебдодатки за дуже короткі терміни, отримуючи ПП, що легко модифікуються і підтримуються в майбутньому.

Особливості *PHP*:

1. Наявність інтерфейсів до багатьох баз даних.

У *PHP* вбудовані бібліотеки для роботи з *MySQL, PostgreSQL, SQLite, mSQL, Oracle, dbm, Hyperware, Informix, InterBase, Sybase*. завдяки стандарту відкритого інтерфейсу зв'язку з базами даних (англ. *Open Database Connectivity Standard, ODBC*) можна підключатися до всіх баз даних, до яких існує драйвер.

2. Нетрадиційність

Мова *PHP* здаватиметься знайомою програмістам, що працюють в різних областях. Багато конструкцій мови запозичені з *C, Perl*. Код *PHP* дуже схожий на той, який зустрічається в типових програмах мовами *C* або *Pascal*. Це помітно знижує початкові зусилля при вивченні *PHP. PHP* — мова, що поєднує переваги *Perl* та *C* і спеціально спрямована на роботу в Інтернеті, мова з універсальним і зрозумілим синтаксисом. І хоча *PHP* є досить молодою мовою, вона здобула таку популярність серед *web*-програмістів, що в наш час є найпопулярнішою мовою для створення вебзастосунків (скриптів).

3. Наявність сирцевого коду та безкоштовність.

Стратегія *Open Source*, і розповсюдження початкових текстів програм в масах, безсумнівно справили сприятливий вплив на багато проєктів, в першу чергу — *Linux* хоч і успіх проєкту *Apache* сильно підкріпив позиції прихильників *Open Source*. Сказане відноситься і до історії створення *PHP*, оскільки підтримка користувачів зі всього світу виявилася дуже важливим чинником в розвитку проєкту *PHP*. Ухвалення стратегії *Open Source* і безкоштовне розповсюдження початкових текстів *PHP* надало неоціненну послугу користувачам. Окрім цього, користувачі *PHP* в усьому світі є свого роду колективною службою підтримки, і в популярних електронних конференціях можна знайти відповіді, навіть на найскладніші питання.

4. Ефективність

Ефективність є дуже важливим чинником у програмуванні для середовищ розрахованих на багато користувачів, до яких належить і *web*. Важливою перевагою *PHP* є те, що ця мова належить до інтерпретованих. Це дозволяє обробляти сценарії з достатньо високою швидкістю. За деякими оцінками, більшість *PHP*-сценаріїв (особливо не дуже великих розмірів) обробляються швидше за аналогічні їм програми, написані на *Perl*. Проте хоч би що робили розробники *PHP*, виконавчі файли, отримані за допомогою компіляції, працюватимуть значно швидше — в десятки, а іноді і в сотні разів. Але продуктивність *PHP* достатня для створення цілком серйозних вебзастосунків.

4.3. Середовище розробки *PhpStorm*

PhpStorm являє собою інтелектуальний редактор для *PHP*, *HTML* і *JavaScript* з можливостями аналізу коду на льоту, запобігання помилок у сирцевому коді і автоматизованими засобами рефакторинга для *PHP* і *JavaScript*. Автодоповнення коду в *PhpStorm* підтримує специфікацію *PHP* 5.3/5.4/5.5/5.6/7.0/7.1 (сучасні і традиційні проекти), включаючи генератори, співпрограми, простори імен, замикання, типажі і синтаксис коротких масивів. Присутній повноцінний *SQL*-редактор з можливістю редагування отриманих результатів запитів.

PhpStorm розроблений на основі платформи *IntelliJ IDEA*, написаної на *Java*. Користувачі можуть розширити функціональність середовища розробки за рахунок установки плаґінів, розроблених для платформи *IntelliJ*, або написавши власні плаґіни.

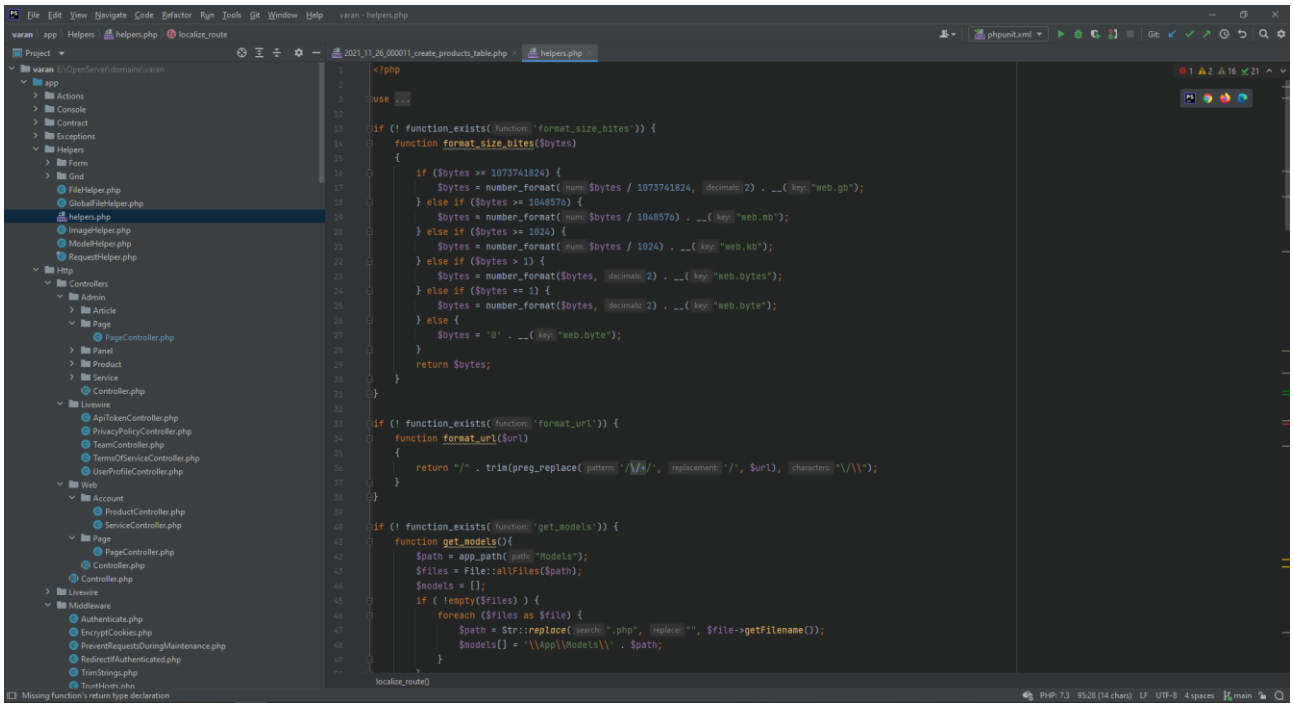

Рис. 3.1. Середовище розробки *PhpStorm*

Основні можливості:

– потужний і функціональний редактор коду з підсвічуванням синтаксису, авто-форматуванням та авто-відступами для мов, що підтримуються;
– проста та потужна навігація в коді;

– допомога при написанні коду, що включає автодоповнення, авто-імпорт, шаблони коду, перевірка на сумісність версії інтерпретатора мови, і багато іншого;
– швидкий перегляд документації для будь-якого елемента прямо у вікні редактора, перегляд зовнішньої документації через браузер, підтримка *docstring* – генерація, підсвічування, автодоповнення та багато іншого;
– велика кількість інспекцій коду;
– потужний рефакторинг коду, який надає широкі можливості для виконання швидких глобальних змін у проекті.

4.4. Контроль версій проекту

Контроль версій, також відомий як управління вихідним кодом, – це практика відстеження змін програмного коду та управління ними. Системи контролю версій – це програмні інструменти, які допомагають командам розробників керувати змінами у вихідному коді з часом. У світлі ускладнення середовищ розробки вони допомагають командам розробників працювати швидше та ефективніше. Системи контролю версій найбільш корисні командам *DevOps*, оскільки допомагають скоротити час розробки та збільшити кількість успішних розгортань.

Програмне забезпечення контролю версій відстежує всі зміни, що вносяться в код, у спеціальній базі даних. При виявленні помилки розробники можуть повернутися назад і порівняти з попередніми версіями коду для виправлення помилок, зводячи до мінімуму проблеми для всіх учасників команди.

Переваги контролю версій.

Повна історія змін кожного файлу протягом тривалого періоду. Це стосується всіх змін, внесених величезною кількістю людей протягом довгих років. Зміною вважається створення та видалення файлів, а також редагування їхнього вмісту.

Розгалуження та злиття. Ця можливість корисна не тільки за одночасної роботи учасників команди: окремі люди також можуть отримати з неї користь і працювати над кількома незалежними напрямками. Створення «гілок» в дозволяє

мати кілька незалежних напрямків розробки, а також виконувати їх злиття, щоб розробники могли перевірити, що зміни, внесені в кожну з гілок, не конфліктують один з одним.

Відстежуваність.

Можливість відслідковувати кожну зміну, внесену в програмне забезпечення, і пов'язувати її з ПЗ для керування проектами та відстеження помилок, а також залишати до кожної зміни коментар з описом мети та призначення зміни може допомогти не лише при аналізі основних причин виникнення помилок, але та при проведенні іншого аналізу. Історія з коментарями під час читання коду допомагає зрозуміти, що цей код робить і чому дія реалізована саме таким чином.

Для дипломного проекту було вирішено використовувати систему контролю версій *Git*.

Git – розподілена система контролю версій, розроблена Лінусом Торвальдсом для роботи над ядром операційної системи *Linux*. Серед великих проектів, у яких використовується *git*, можна назвати ядро *Linux, Qt, Android*. *Git* вільний і розповсюджується під ліцензією *GNU GPL* 2 і також як *Mercurial*, доступний практично на всіх операційних системах. За своїми базовими можливостями *git* схожий на *Mercurial*, але завдяки ряду переваг (висока швидкість роботи, можливість інтеграції з іншими системами, зручний інтерфейс) і дуже активній спільноті, що сформувалася навколо цієї системи, *git* вийшов у лідери ринку розподілених систем контролю версії.

GitHub – сервіс онлайн-хостингу репозиторіїв, що володіє всіма функціями розподіленого контролю версій та функціональністю управління вихідним кодом – все, що підтримує *Git* і навіть більше. Також *GitHub* може похвалитися контролем доступу, багтрекінгом, керуванням завданнями для кожного проекту. *Git*-репозиторій, завантажений на *GitHub*, доступний за допомогою інтерфейсу командного рядка *Git* та *Git*-команд.

GitHub програмного забезпечення онлайн оголошень на рис. 3.15.

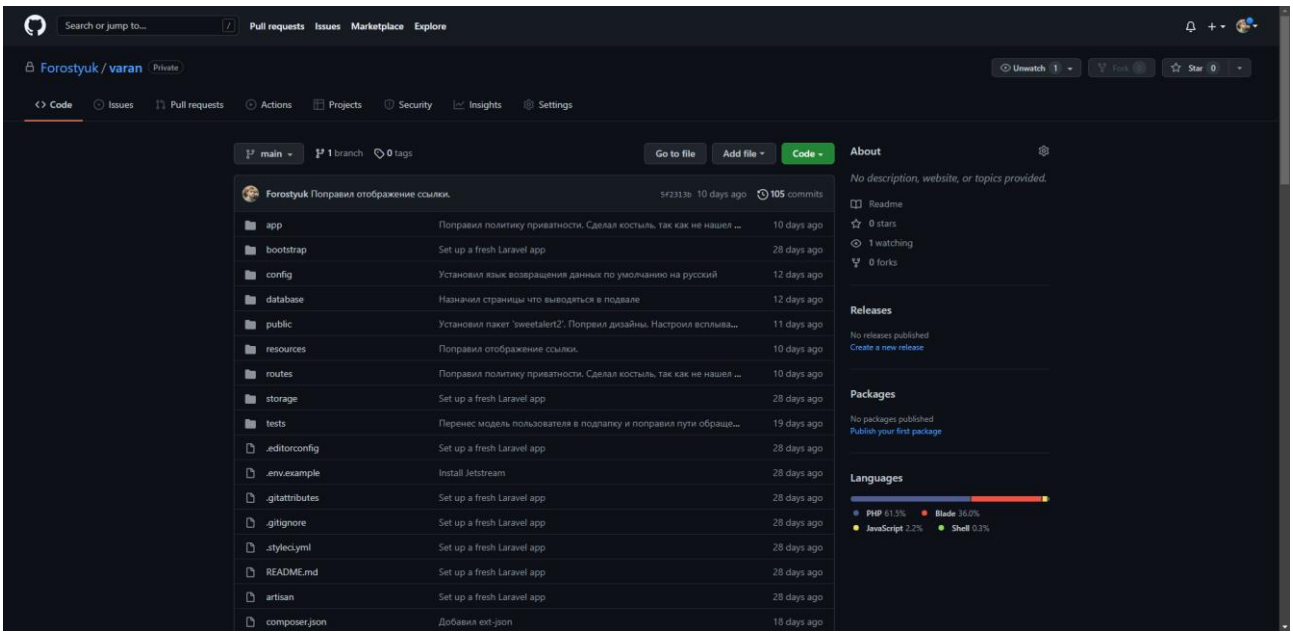

Рис. 3.15. *GitHub* мікросервісу цілей та досягнень

На наведеному зображенні можна побачити, щоб *GitHub* сторінка проекту мікросервісу цілей та досягнень дозволяє передивится: вихідний код проекту, історію змін проекту, заведені баги (*issues*), запити на зміну вихідного коду (*pull requests*), розробників проекту (*contributors*) і т.д.

4.5. Реалізація клієнтської частини

Для розмітки сторінок ВД використано мову гіпертекстової розмітки *HTML* [10].

HTML - стандартна мова розмітки для створення вебсторінок і вебдодатків. З *CSS* і *JS*, вона утворює тріаду основних технологій для *WWW*.

HTML описує структуру вебсторінки семантично для зовнішнього вигляду документа.

HTML впроваджує засоби для:

- створення структурованого документа шляхом позначення структурного складу тексту: заголовки, абзаци, списки, таблиці, цитати та інше;
- отримання інформації через гіперпосилання;
- створення інтерактивних форм;
- включення зображень, звуку, відео, та інших об'єктів до текста.

Для реалізації поведінки клієнтської частини використано МП *JS* [11].

JS - динамічна, об'єктно-орієнтована прототипна МП. Надає можливість на стороні клієнта (пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд вебсторінки.

Основні характеристики

- Архітектура на стороні клієнта. Тут використовується модель «на стороні клієнта», що означає, що вона зменшує навантаження на сервер і працює дуже швидко відповідно до ресурсів клієнта.
- Багатофункціональний дизайн. Його можна використовувати для створення дуже інтригуючих і привабливих інтерфейсів. Він має підтримку кількох зовнішніх бібліотек, які доповнюють функції.
- Підтримує широкий спектр мов. Він підтримує *HTML* і його можна легко каскадувати для керування вмістом. Крім того, його можна запрограмувати на отримання даних з кількох джерел або способів.
- Перевірки безпеки. *JavaScript* вимагає шифрування та належних перевірок безпеки в коді як архітектура «на стороні клієнта». Це вказує на те, що код надходить до клієнта, який може бути легко зламаний, якщо код не зашифрований належним чином.

Мова *JS* використовується для:

- написання сценаріїв вебсторінок для надання їм інтерактивності;
- створення односторінкових вебзастосунків;
- програмування на стороні сервера;
- стаціонарних застосунків;
- мобільних застосунків;
- сценаріїв в прикладному ПЗ;

CSS — ще одна з найкращих мов інтерфейсу. Каскадні таблиці стилів – це в основному інструмент вебдизайну, який використовується для керування макетом вебсторінки.

CSS — це окремий каскадний стиль, який працює в інтерфейсі. *CSS*, по суті, є невеликим файлом, який поєднує задню частину з невеликим файлом.

4.6. Система управління базами даних

Для збереження інформації обрано СУБД *MySQL* [13].

MySQL – це популярна система СУБД, яка дуже часто застосовується в поєднанні з *PHP*.

У реляційній БД дані зберігаються не усім скопом, а в окремих таблицях, завдяки чому досягається виграш у швидкості та гнучкості. Таблиці зв’язуються між собою за допомогою відносин, завдяки чому забезпечується можливість об'єднання при виконанні запитів даних із декількох таблиць. *SQL*, як частину системи *MySQL*, можна охарактеризувати як мову структурованих запитів, яка є найбільш поширеною стандартною мовою, що використовується для доступу до БД.

MySQL складається з двох частин: серверної та клієнтської.

Сервер *MySQL* постійно працює на комп’ютері. Клієнтські програми (наприклад, скрипти *PHP*) надсилають серверу *MySQL SQL*-запити через механізм сокетів (тобто, за допомогою програмних засобів), сервер їх обробляє і запам’ятовує результат. Тобто скрипт (клієнт), вказує на те, яку інформацію він хоче отримати від сервера БД. Після чого сервер БД посилає відповідь (результат) клієнту (скрипту).

Механізм використання сокетів передбачає технологію клієнт-сервера, а це означає, що в системі повинна бути запущена спеціальна програма - *MySQL*-сервер, який приймає і обробляє запити від програм. Так як вся робота відбувається в дійсності на одній машині, накладні розходи по роботі з мережевими засобами незначні (установка і підтримка з'єднань з *MySQL*-сервером обходиться досить дешево).

4.7. Фреймворк

За для того, щоб якомога ефективніше та зручніше вести розробку нашого вебдодатку, потрібно обрати фреймворк відповідно до обраної мови програмування.

Вебфреймворк (*WF*) або фреймворк вебдодатків (*WAF*) — це програмна платформа, яка розроблена для підтримки розробки вебдодатків, включаючи вебсервіси, вебресурси та веб *API*. Вебфреймворки забезпечують стандартний спосіб створення та розгортання вебдодатків у всесвітній мережі. Вебфреймворки мають на меті автоматизувати накладні витрати, пов'язані з звичайними діями, що виконуються в веброзробці. Наприклад, багато вебфреймворків надають бібліотеки для доступу до бази даних, шаблонів та керування сеансами, і вони часто сприяють повторному використанню коду. Хоча вони часто націлені на розробку динамічних вебсайтів, вони також застосовні до статичних вебсайтів.

Розробка додатків або програмного забезпечення є складним процесом, який складається з різноманітних кроків, таких як тестування, написання коду, розробка додатків та багато іншого.

Щоб спростити ці процедури для інженерів-програмістів, ідеальним є використання фреймворків програмування. Таким чином, фреймворки не тільки тримають у порядку ваші процедури розробки, але й забезпечують відповідну базову структурну підтримку вашому кодуванню.

Ось основні причини використовувати фреймворк під час програмування:

1. Скорочує час розробки
2. Основною причиною використання фреймворку є його властивість долати тривалість розробки. Це, безумовно, займає менше часу та енергії кодерів, забезпечуючи підтримку помилок, сеансів та обробки даних.
3. З фреймворком розробникам також не потрібно піклуватися про автентичну логіку та очищення даних, оскільки вона виконує ці функції.
4. Робить процес розробки більш організованим

5. Фреймворк також належним чином організовує функції програми. Розробникам не доведеться турбуватися про перевпорядкування окремих файлів, оскільки фреймворк працює з файлами інтерфейсу та вебкаталогами автономно, використовуючи бізнес-логіку.
6. Захищає код
7. Фреймворки також пропонують чудову безпеку коду для розробників. Завдяки хорошому фреймворку програмістам не доведеться турбуватися про будь-які кібератаки на сценарій програми. Зокрема, фреймворки забезпечують велику безпеку під час розробки вебдодатків.
8. Підтримка спільноти
9. Як і мови програмування, більшість фреймворків також мають відкритий код і мають власні великі спільноти. Ці спільноти пропонують рішення та ресурси, щоб допомогти колегам або початківцям програмістам.
10. *CRUD*

    C - Create, R - read, U - update, D - delete.
11. Фреймворки також мають чотири ключові функції *CRUD*. Тут ви з великою легкістю створюєте, читаєте, оновлюєте та видаляєте дані. Бібліотеки *Framework* також полегшують адміністрування сеансів, підхід до баз даних, керування файлами *cookie* та *HTML*-сторінками, налаштування шаблонів тощо.
12. Підвищує продуктивність
13. Фреймворк сприяє швидкому створенню прототипів, розгортанню додатків і повторному використанню коду. Ось чому це підвищує швидкість розробки вашого програмного забезпечення. Підвищена продуктивність також є головною перевагою швидкої розробки додатків, якою користуються кодери під час використання фреймворків.
14. Застосовується для командної роботи
15. Іншою причиною використання фреймворку є його застосовність для командної роботи. В основному, якщо ви працюєте з великими командами розробників віддалено, то фреймворки запропонують вам багато переваг.

16. Аналогічно, ваші експерти з баз даних можуть керувати завданнями, пов’язаними з даними, а досвідчені розробники можуть створювати стійкі плагіни, бібліотеки тощо.

Для покращення функціоналу ВД, збільшення можливостей до розширювання, більш простого функціоналу імпортування додатку на інші пристрої, а також збільшення рівня безпеки використання даних на серверній частині було обрано *PHP* фреймворк *Laravel*[5]

Рис. 2.4. Логотип *Laravel*

Laravel – популярний фреймворк для самих різноманітних проектів.

Фреймворк *Laravel* призначений для розробки ВД відповідно до шаблону *Model-View-Controller (MVC)*.

Однією з особливостей *Laravel* є модульна система упакування з виділеним менеджером залежностей *Composer*[21], утиліти, які допомагають в розгортанні додатків і технічного обслуговування.

Фреймворк *Laravel* має велику екосистему з негайним розгортанням, маршрутизацією, шаблонизацією, *ORM*, запитами до БД і лістингом.

Ключові особливості *Laravel*:

1. Управління залежностями

Лідерство в залежності – одна з найважливіших характеристик *Laravel*, ключовим аспектом вивчення сучасних інтернет-додатків є розуміння функціональності контейнера послуг (*IoC*). Найсильніший інструмент управління залежностями класів в *Laravel* - це *IoC* (*Control Invert*) чи службовий контейнер. *Dependency Injection* - це засіб видалення та введення жорстко запрограмованих курсів за допомогою інструмента, подібного до композитора.

2. Модульність

Модульність – це ступінь поділу та рекомбінації частин вебдодатку. Ви можете розділити логіку компанії на різні компоненти, які працюють разом для функціонування вебпрограми. Він призначений для модульного використання, сам *Laravel* є також набором деталей. Ви можете легко створювати та розробляти масштабні корпоративні програми, використовуючи модульну структуру. Він пропонує дуже простий посібник зі створення в *Laravel* модулів або пакетів.

3. Аутентифікація

Аутентифікація є компонентом будь-якої сучасної вебпрограми. Для написання аутентифікації в іншому середовищі, такому як *Codeigniter*, може знадобитися багато часу. Він пропонує блокову аутентифікацію, яка дозволяє створити повнофункціональну схему аутентифікації за допомогою простої команди. Він також надає зручні документи для вашої власної автентифікації.

4. Кешування

Кешування - це метод інформації, що зберігається на сайті тимчасового зберігання, який може бути швидко отриманий при необхідності Кешування в основному використовується для підвищення ефективності програми. Практично вся інформація від перспективи до шляхів кешується в *Laravel*. Це скорочує час обробки, допомагаючи підвищити ефективність.

5. Маршрутизація

Це просто для розуміння маршрутизації в *Laravel* та аналогічно рейковій рамі *Ruby*. Маршрутизація *Laravel* може бути використана для простого створення спокійної програми. Ви можете групувати, називати, фільтрувати та зв'язувати інформацію про вашу модель з шляхами. Маршрути *Laravel* можуть бути дуже гнучкими та керованими для створення зручних для пошукових систем *URL*.

6. Безпека

Laravel пропонує інтуїтивно зрозумілий спосіб створення безпечних вебпрограм. Замість звичайних текстових паролів усі паролі зберігаються у вигляді хеша. Для хешування паролів він використовує *BCrytp*. Він забезпечує безпеку

атаки з використанням *SQL*-ін'єкцій, а також екранує всі записи користувача, запобігаючи впровадженню тегів скрипта.

7. Міграційна система

Як і Ruby on Rails, він надає систему міграції для створення бази даних. Замість використання *SQL*, ви можете використовувати *PHP* для написання мігрантів, які створюють структуру вашої бази даних. Ви можете використовувати ці міграції для створення баз, таблиць та індексів. Натомість ви можете виконати нову міграцію, якщо ви хочете змінити стовпець таблиці, вам не потрібно повторювати створення таблиці знову.

8. Artisan

Artisan – це ім'я інструменту командного рядка *Laravel*. Він включає десятки вбудованих команд, які можна використовувати для виконання завдань з інтерфейсом командного рядка. У процесі розробки цей інструмент дозволяє уникнути повторюваних завдань.

9. Побудовник запитів до баз даних

Автор бази даних запитів *Laravel* пропонує простий спосіб створення запитів до бази даних. Він включає безліч допоміжних функцій, які ви можете використовувати для фільтрації ваших даних. Складні запити можуть легко реалізуватися за допомогою посилань *Laravel*. Синтаксис розробника запитів *Laravel* полегшує розуміння і зручність написання запитів до бази даних.

10. Шаблон двигуна

Blade – це шаблонний двигун *Laravel*. *Blade* пропонує кілька допоміжних функцій для форматування ваших даних у виставах. *Blade* також використовує шаблон для створення складних макетів. Розширення файлу всіх *Blade* шаблонів.

11. Eloquent OPM

В основі *Eloquent ORM* від *Laravel* лежить підтримка багатьох механізмів баз даних. *MySQL* та *SQLite* працюють відмінно. Він надає всі промовисті функції з повною документацією.

12. Інші контролери

Інші контролери *Laravel* дозволяють вам відокремити логіку запитів *GET* або *POST*. Також можна створювати контролери ресурсів, які легко використовувати для генерації *CRUD*. Потім можна підключити контролер ресурсу до шляху, щоб автоматично обслуговувати всі шляхи *CRUD*.

4.8. Прототип клієнтської частини вебсервісу

Головна сторінка вебсервісу

При завантажені сторінки ВД, користувач одразу потрапляє на головну сторінку (рис. 4.1), де відразу відображаються популярні категорії, нові товари, тощо.

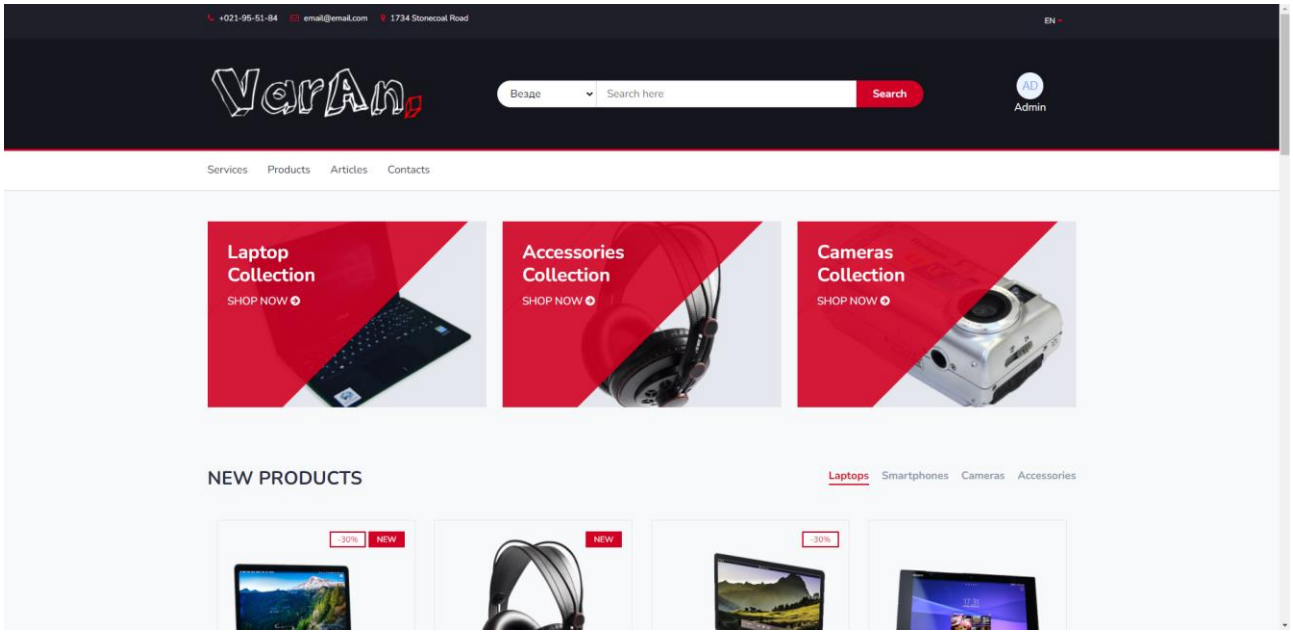

Рис. 4.1. Головна сторінка

Реєстрація

При бажанні самому додавати товари та послуги користувач може сам зареєструватися.

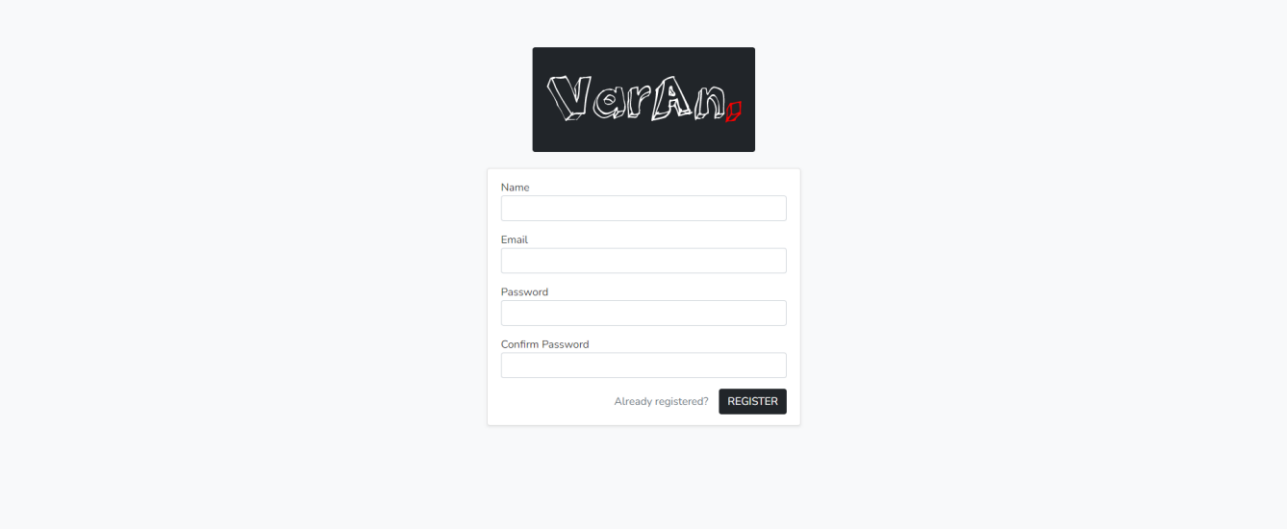

Рис. 4.2. Сторінка реєстрації

Редагування інформації про користувача

Після реєстрації користувач зможе змінити всю інформацію про себе (рис. 4.2).

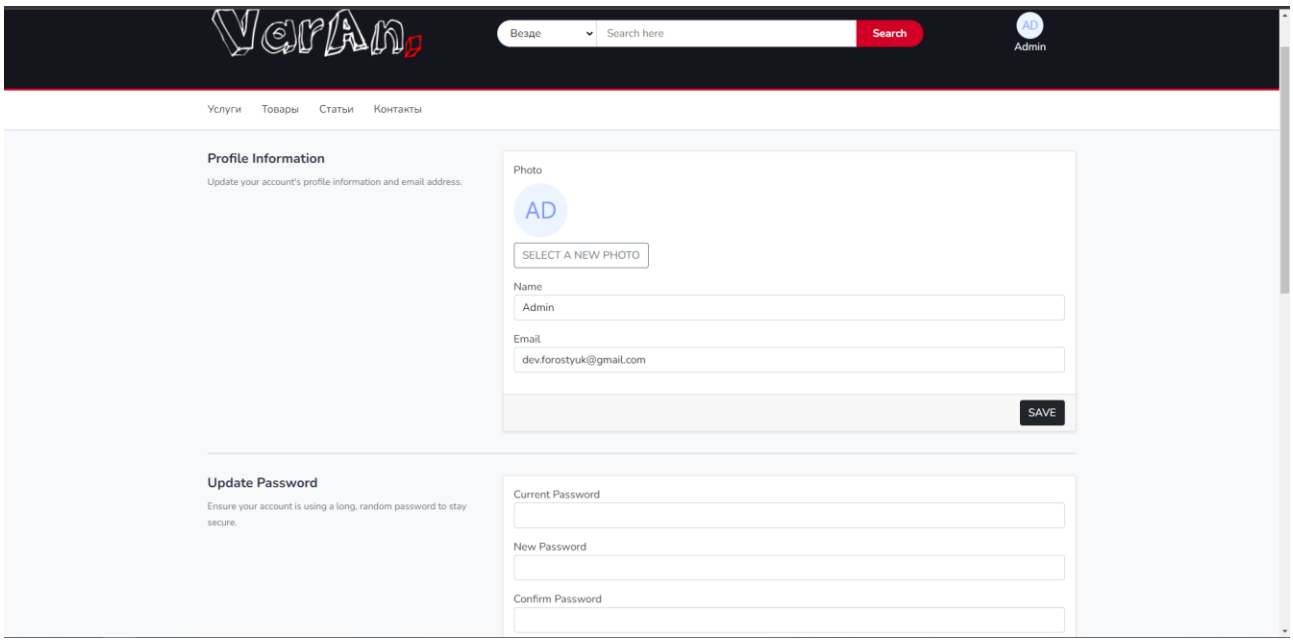

Рис. 4.2. Редагування інформації про користувача

Користувач переглянути останні сесії, а також при бажанні видалити свій аккаунт (рис. 4.3).

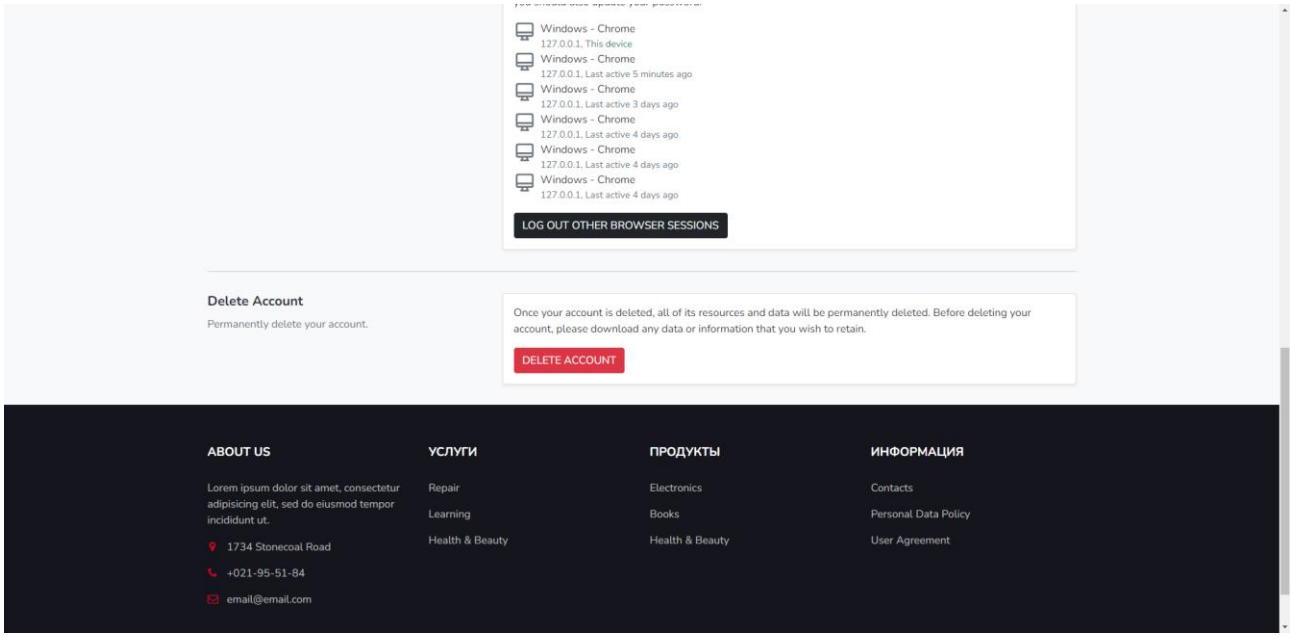

Рис. 4.3. Застосування фільтру

Для скасування фільтру необхідно зняти помітки на обраних критеріях та знови натиснути кнопку «Застосувати фільтр».

Товари та послуги

Користувач має можливість перейти в меню додавання, редагування, або видалення товарів чи послуг з будь якої точки ВД, скориставшись цим меню(рис. 4.4).

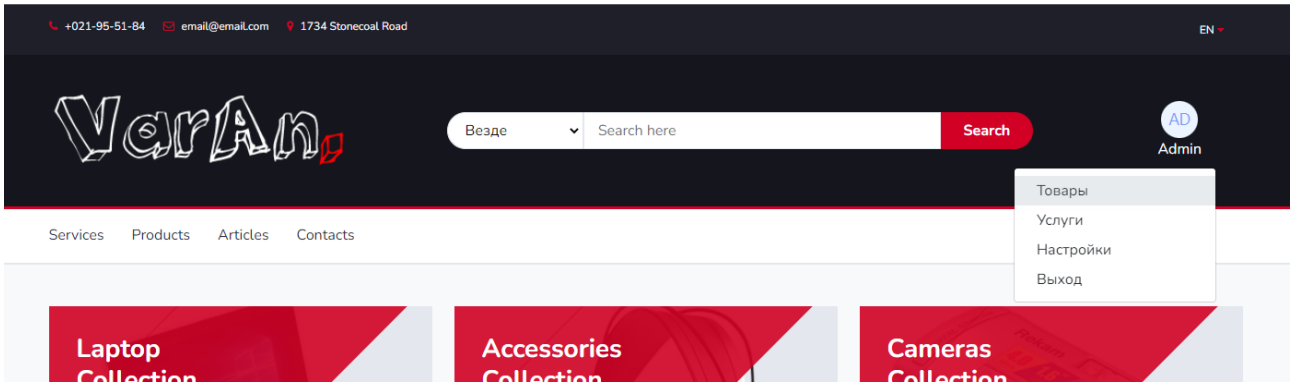

Рис. 4.4. Швидкий доступ до інформації

Перегляд списку товарів

Користувач має можливість переглянути весь список товарів(рис. 4.5), що він додав, а також видалити будь який товар.

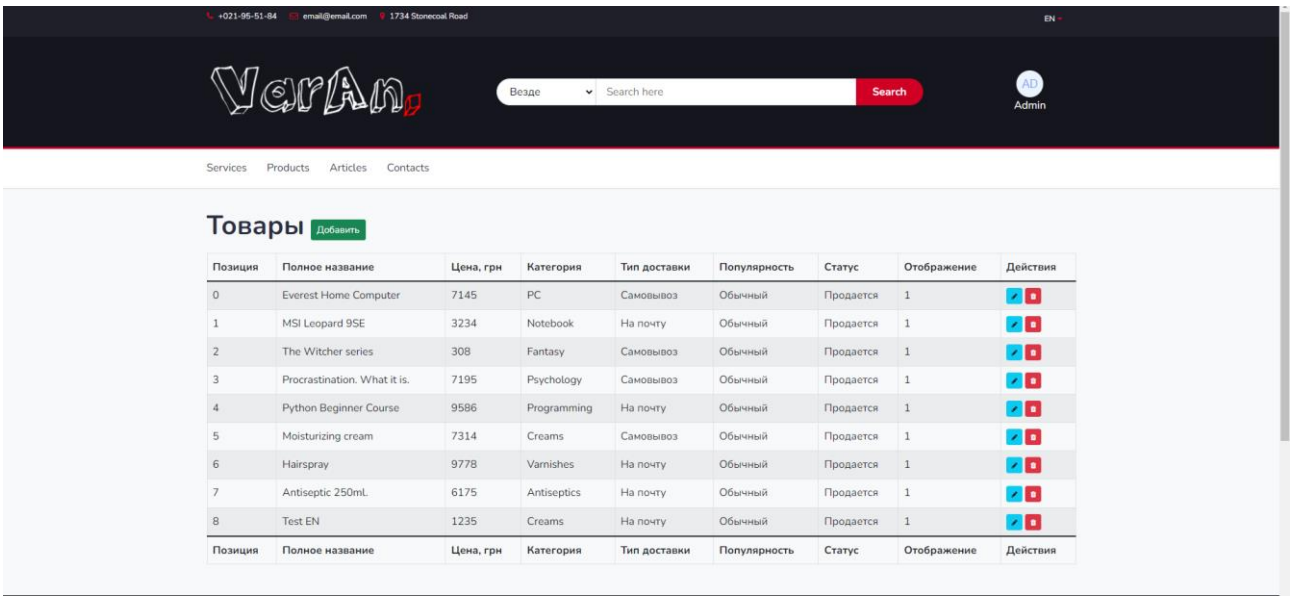

Рис. 4.5. Список товарів

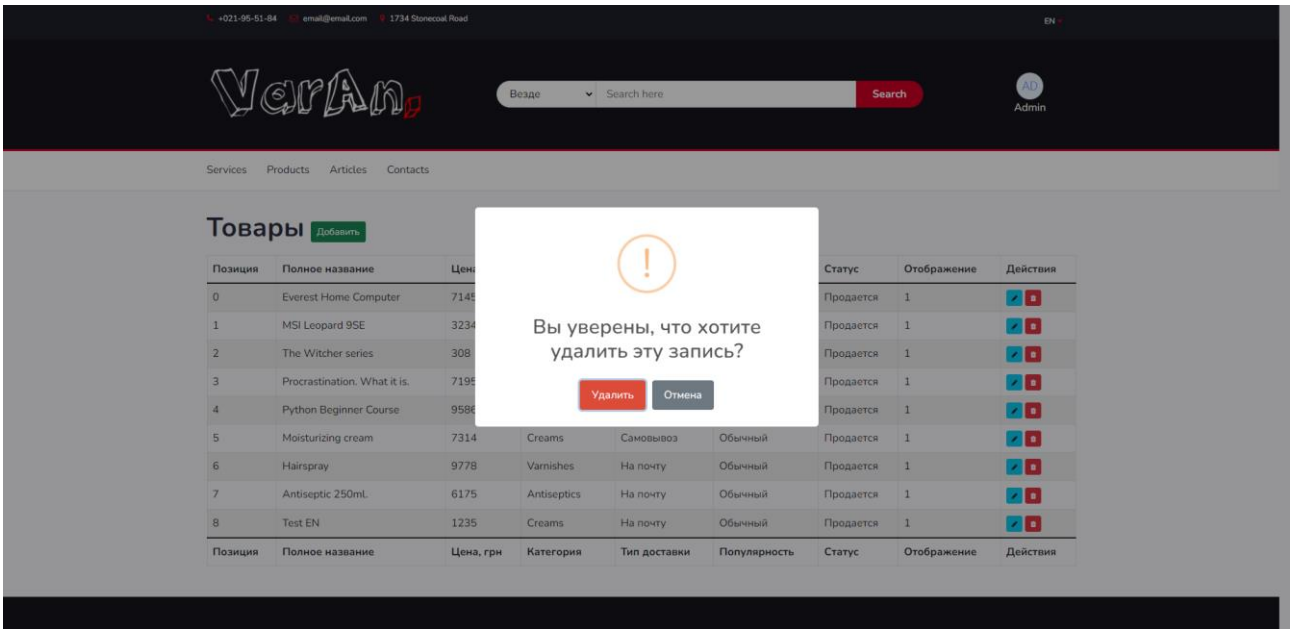

Рис. 4.6. Підтвердження видалення

Додавання товару

Користувач має змогу додати товар в будь яку доступну категорію, а також на всіх доступних у ВД мовах, та в будь який час. Обов’язкові поля позначаються червоною зіркою (рис. 4.7-4.8).

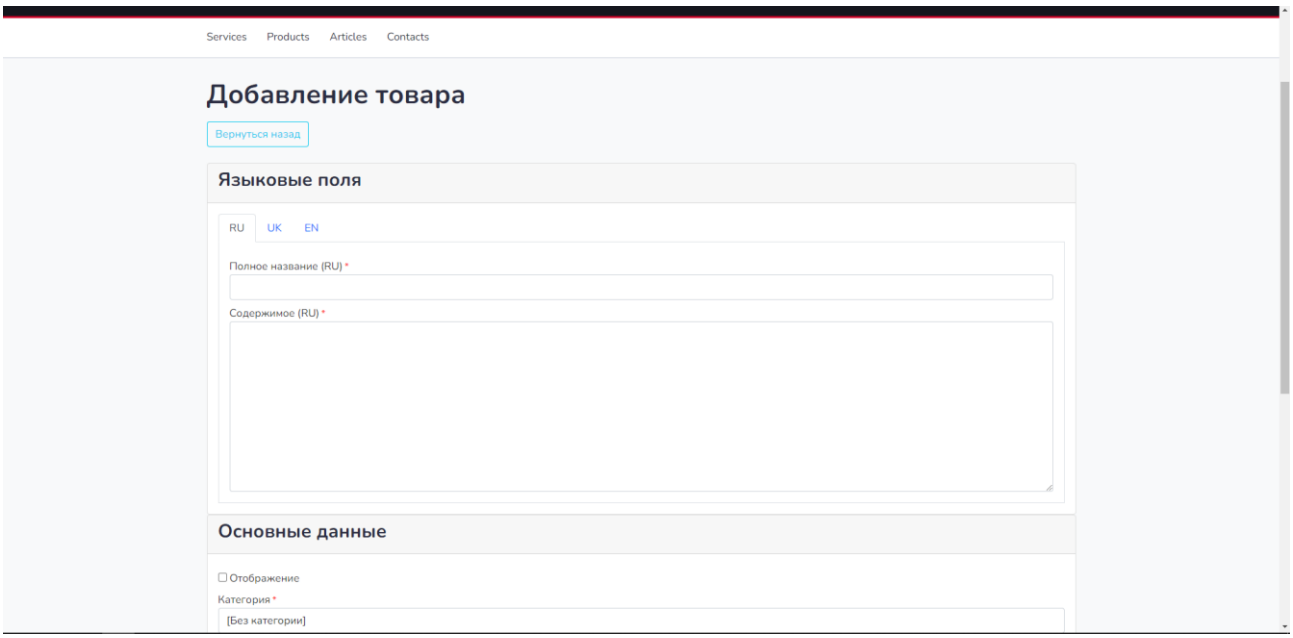

Рис. 4.7. Форма додавання довару, частина 1.

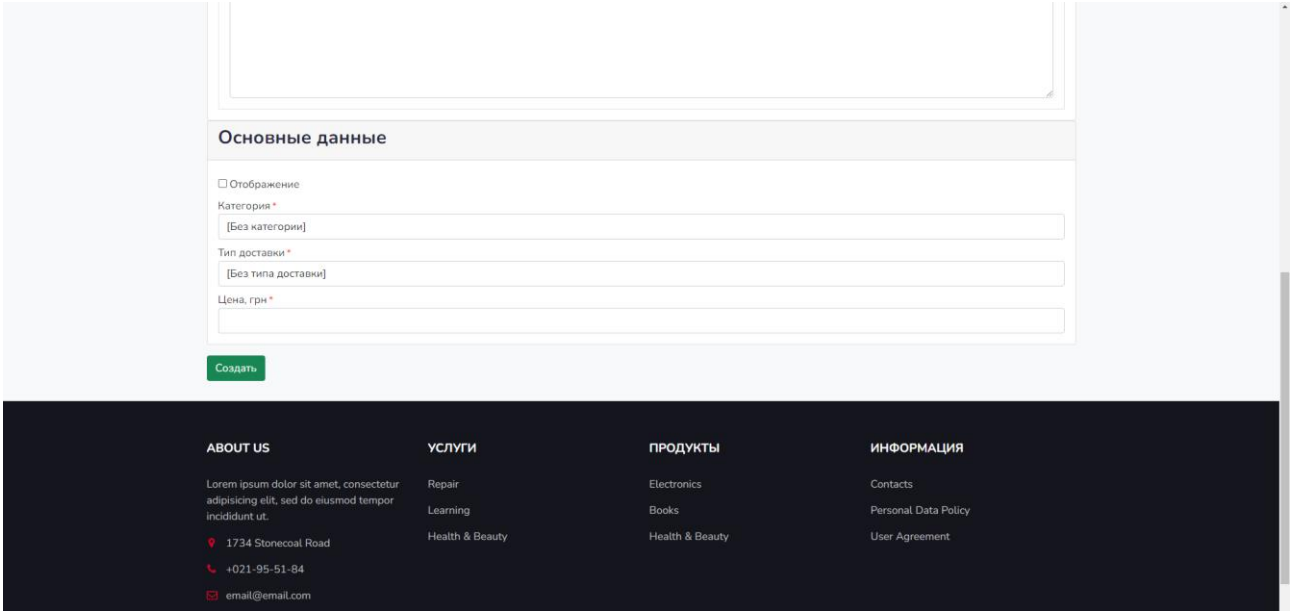

Рис. 4.8. Форма додавання довару, частина 2

Інформація про сервіс

Для отримання інформації про сервіс користувачу необхідно перейти на вкладинку «Про сервіс». На даній вкладинці міститься міститься коротка інформація про сам ВС оголошень, контакти для отримання довідок, а також карта з розташуванням (рис. 4.5).

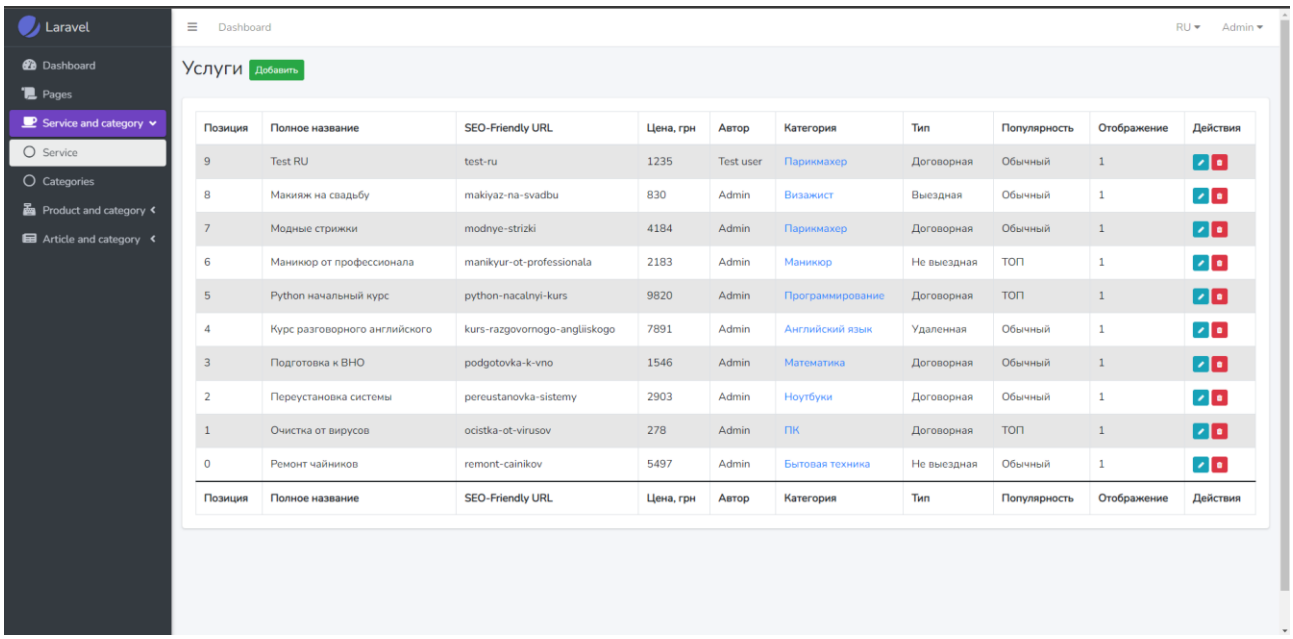

Рис. 4.5. Вкладинка «Про сервіс»

Форма скарг та пропозицій

Щоб залишити повідомлення адміністраторам ВС, користувачеві необхідно натиснути на іконку конверта в правому нижньому куті, після чого відкриється форма для заповнення (рис. 4.6).

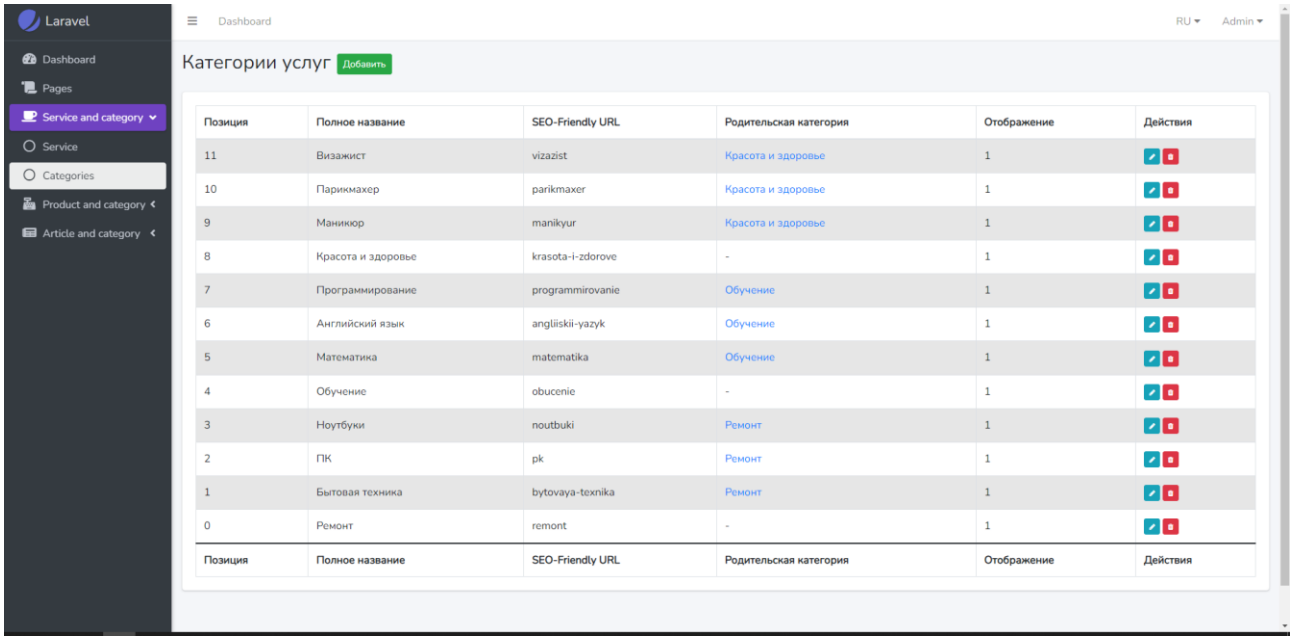

Рис. 4.6. Форма скарг та пропозицій

4.9. Прототип панелі адміністратора

Авторизація

Для роботи із сервісом з правами адміністратора користувачу виділяється персональний логін та пароль, які йому необхідно ввести на сторінці авторизації до панелі адміністратора(рис. 4.7).

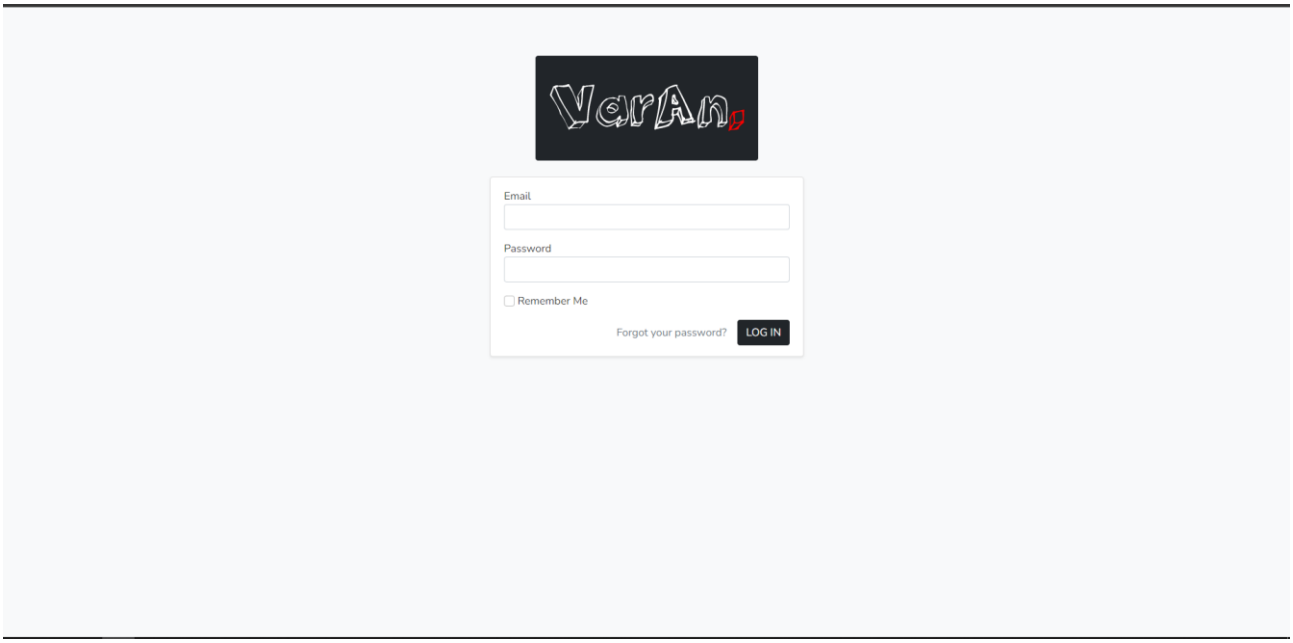

Рис. 4.7. Форма авторизації адміністратора

Панель адміністратора

Після введення відповідних логіна та пароля користувач потрапляє до панелі адміністратора(рис. 4.8), а у разі ведення хибних даних залишається на формі авторизації.

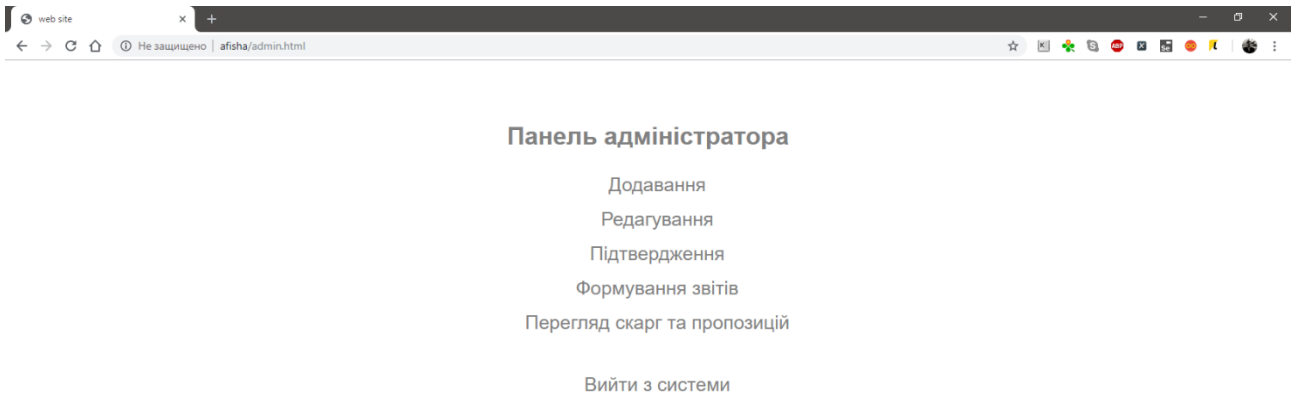

Рис. 4.8. Панель адміністратора

Для попередження небажаних змін та некоректних дій збоку третіх осіб, адміністратор має можливість вийти з системи, після чого відбудеться перенаправлення на форму авторизації.

Адміністратору ВС доступні наступні функції:

– Додавання нового оголошення;
– Редагування існуючих оголошень;
– Підтвердження заявок;
– Формування звітів;
– Перегляд скарг та пропозицій.

Додавання оголошення

Додавання нової події відбувається шляхом заповнення форми (рис. 4.9) з усіма обов’язковими полями. Якщо не заповнено хоч одне з полів, при підтверджені додавання оголошення, адміністратора буде сповіщено про не заповнення одного з полів відповідним повідомленням, у іншому випадку буде сповіщено про успішне додавання оголошення.

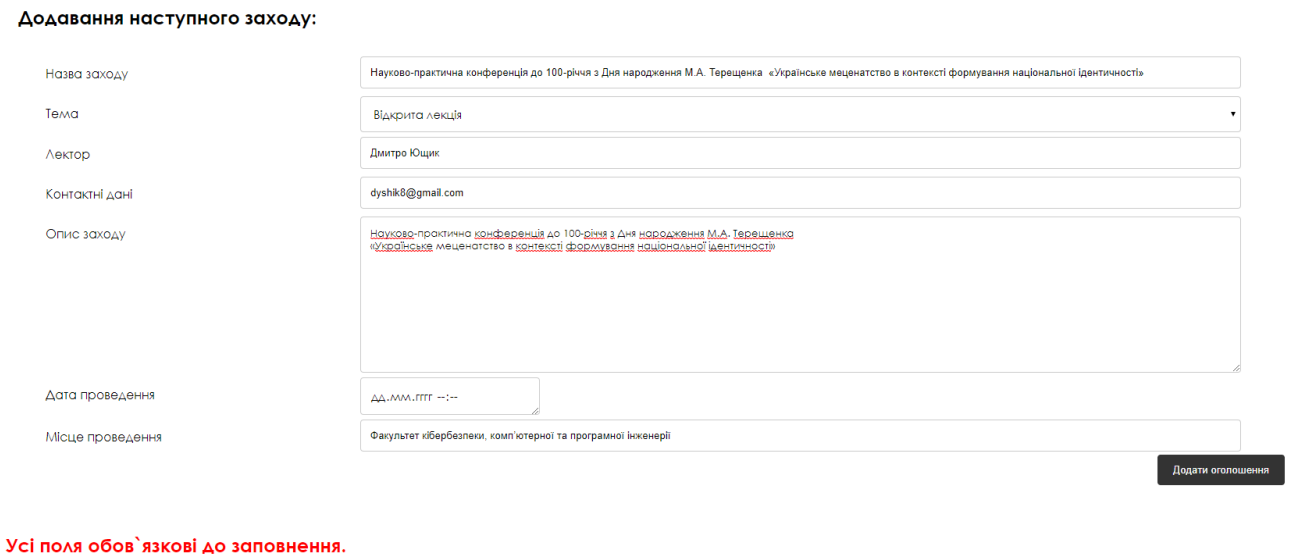

Рис. 4.9. Помилка не заповнення полів

Редагування оголошень

Для редагування існуючого оголошення необхідно обрати бажану подію зі списку подій(рис 4.10) та натиснути клавішу «Редагувати захід».

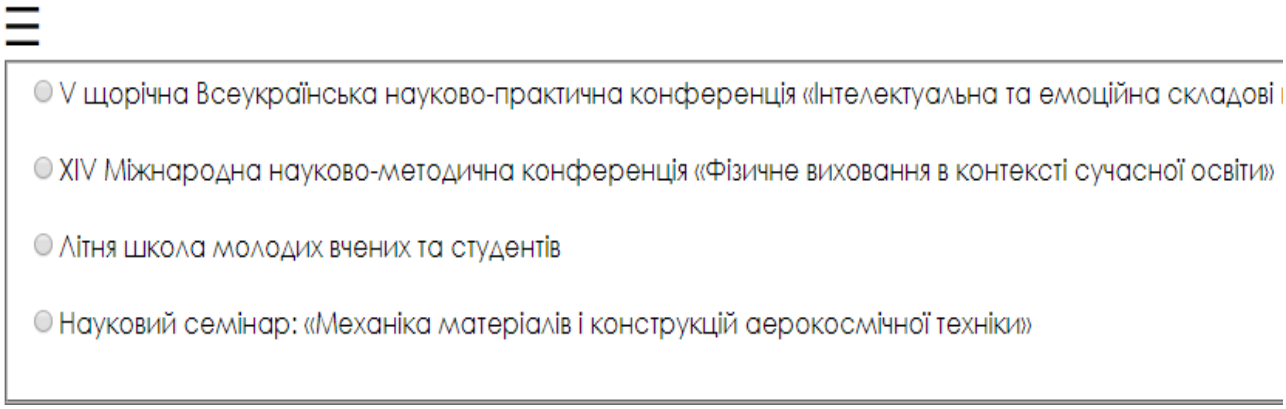

Рис. 4.10. Вибір подій для редагування

Обравши бажану подію, відбувається перехід на сторінку, де буде відображена уся інформація про подію (рис. 4.11), яку, у разі внесення коректив, можна редагувати. Після редагування оголошення інформація про подію оновлюється та одразу стає доступною для усіх користувачів.

Окрім редагування оголошення є можливість його видалення, якщо з деяких причин захід буде скасовано.

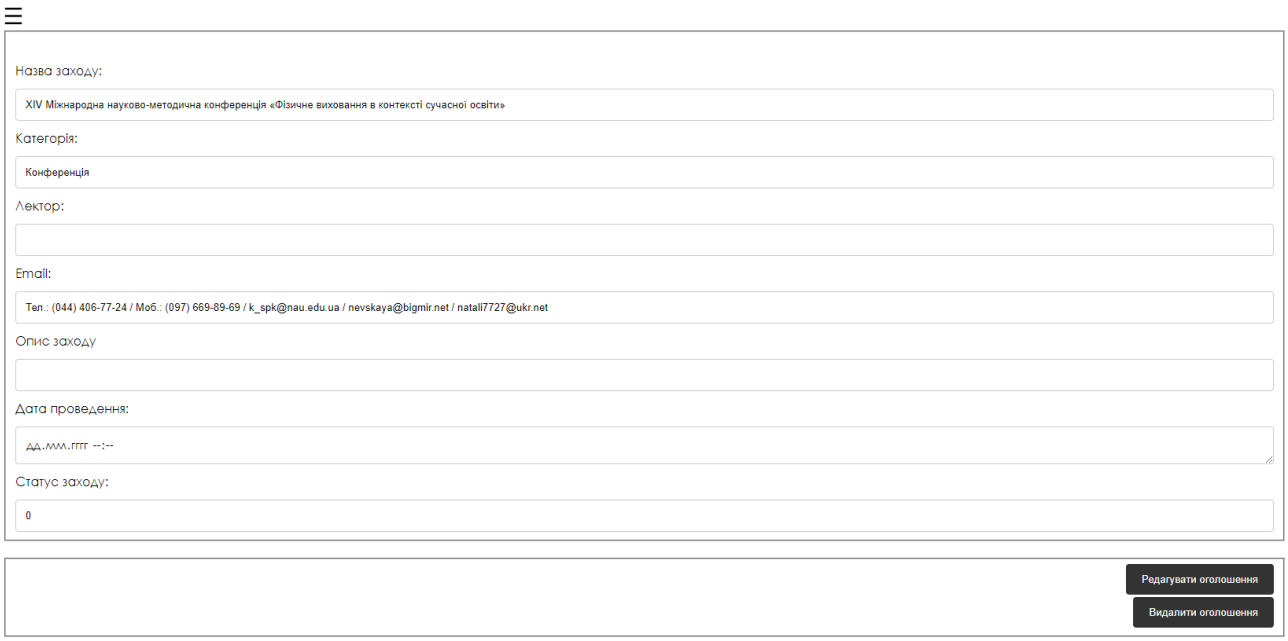

Рис 4.11. Форма редагування оголошення

Підтвердження заявки

Після подачі заявки лектором, вона стає доступна адміністратору для підтвердження або відхилення. Необхідно обрати одну з поданих заявок для відображення інформації про запланований захід (рис. 4.12). Якщо заявка влаштовує адміністрацію університету повністю, натиснути «підтвердити заявку». Якщо заявка частково влаштовує, то можливо внести відповідні корективи, після чого підтвердити заявку. Якщо подана заявка не влаштовує університет, то натиснути «відхилити заявку».

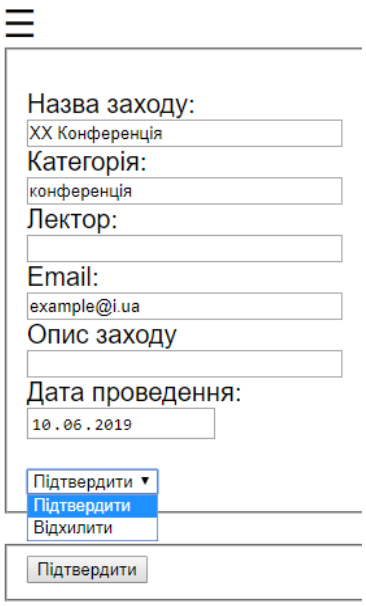

Рис. 4.12. Підтвердження заявки

Формування звітів

Для ведення статистики проведених заходів та подання звітів у ВС передбачено автоматичне формування *Excel*-файлу, із зазначенням необхідних полів(рис 4.13). Це значно полегшує процес обробки інформації.

	A	B	C	D
1	**Назва**	**Тип заходу**	**Лектор**	**Дата проведення**
2	V щорічна Всеукраїнська науково-практична конференція «Інтелектуальна та емоційна складові навчання іноземних мов: новітні тенденції і завдання для вищої школи»	Конференція		07.06.2019
3	XIV Міжнародна науково-методична конференція «Фізичне виховання в контексті сучасної освіти»	Конференція		14.06.2019
4	Літня школа молодих вчених та студентів	Конференція		18.06.2019
5	Науковий семінар: «Механіка матеріалів і конструкцій аерокосмічної техніки»	Семінар		24.06.2019
6	XIX міжнародна науково-технічна конференція АС ПГП «Промислова гідравліка і пневматика» (міжвузівська конференція)	Конференція		17.09.2019
7	Науково-методичний семінар «Інтегровані технології та системи для виробництва та освіти»	Семінар		19.09.2019
8	Науковий семінар: «Механіка матеріалів і конструкцій аерокосмічної техніки»	Семінар		30.09.2019

Рис. 4.13. Приклад формування звіту

Перегляд скарг та пропозицій

Щоб враховувати побажання користувачів, передбачено форму скарг та пропозицій, де можливо за власним бажанням поділитися враженнями роботи з ВС, або висловити свої пропозицій для покращення роботи(рис. 4.14).

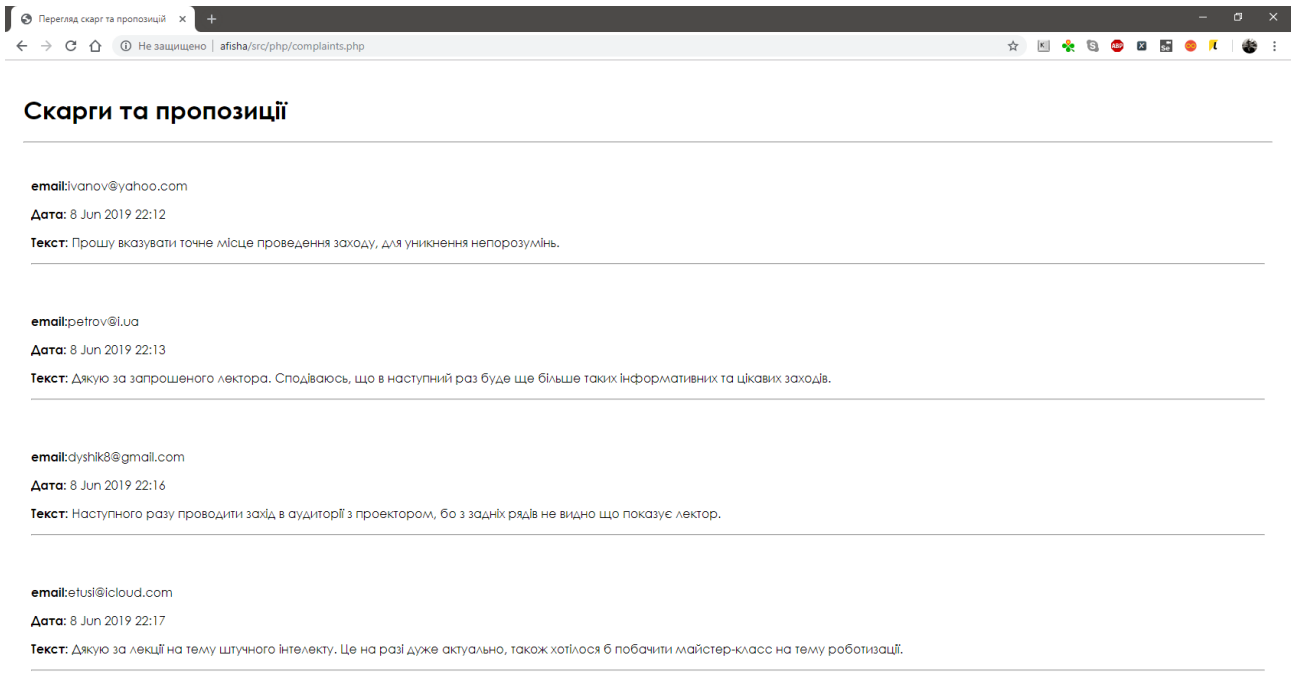

Рис. 4.14. Перегляд скарг та пропозицій

4.10. Висновки до розділу

В даному розділі було проаналізовано та аргументовано обрані технології та засоби, за допомогою яких буде розроблятися прототип ВС онлайн-оголошень. Продемонстровано можливості роботи ВС для різних типів користувачів, таких яка: «Користувач» та «Адміністратор.» Описано інструкції щодо правильної роботи з ВС.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було розроблено ПЗ для створення онлайн-оголошень з продажу, покупки або обміну товарами та послугами. Даний сервіс вирізняється високою ергономічністю, ефективністю та спрощенням процедури пошуку та додавання товарів та послуг різних категорій.

Також було досліджено принципи роботи вебдодатків, а також перенесено основну логіку оголошень на функціонал вебдодатку. Визначено основні труднощі, що виникають у користувачів при використанні сервісів онлайн-оголошень. Проаналізовано існуючі програмні рішення, визначено наскільки вони відповідають потребам користувачів, порівняно їх функціонал, виділені переваги та недоліки.

В рамках кваліфікаційної роботи було проведено дослідження процесу створення та пошуку онлайн оголошень для товарів та послуг. Також було проведено порівняння з існуючими рішеннями, на підставі порівняльного аналізу були введені критеріїв для оцінки:

- Функціональність
- Ергономічність інтерфейсу
- Вартість
- Розширення
- Підтримка користувача

На етапі проектування програмного забезпечення було виділено різні модулі при аналізі існуючих рішень. Також була спроектована архітектура та алгоритми програмного забезпечення, які відповідають вимогам функціонування онлайн-оголошень, та якісно реалізовано поставлені задачі. Також була спроектована архітектура взаємодії між різними рівнями вебдодатку та продумана взаємодія між різними елементами програмного забезпечення.

На етапі розробки системи була реалізована архітектура, що дозволить безпечно взаємодіяти між різними шарами програмного забезпечення, а також

розроблено всі елементи системи, у відповідності до етапу проектування. Розроблено вебдодаток, за допомогою якого надано можливість створювати онлайн-оголошення для товарів та послуг. Забезпечено зручний та привабливий інтерфейс для взаємодії користувача з функціоналом вебдодатку. Реалізовано функціонал, який вигідно відрізняється серед існуючих програмних рішень.

В ході виконання кваліфікаційної роботи отримано програмне забезпечення для створення онлайн-оголошень з продажу, покупки або обміну товарами та послугами. Завдяки широкому та потужному функціоналу, зручному інтерфейсу користувачам набагато простіше та зрозуміліше стало додавати, редагувати, видаляти та знаходити онлайн-оголошення для товарів та послуг. За допомогою розробленого програмного забезпечення відбувається взаємодія з базою даних в якій зберігається вся інформація. Також створено механізм відновлення даних, що в разі втрати якихось даних, допоможе повернути останній актуальний стан бази даних. Реалізовано підтримку кроссбраузерності та кроссплатформеності, що дозволить використовувати повний функціонал програмного забезпечення на різних пристроях.

Основні результати кваліфікаційної роботи:

1. Досліджено принципи роботи вебдодатків, а також перенесено основну логіку оголошень на функціонал вебдодатку.
2. Визначено які труднощі виникають в користувачів при використанні сервісів онлайн-оголошень.
3. Розроблено архітектуру та алгоритми програмного забезпечення, які відповідають вимогам функціонування онлайн-оголошень, та якісно реалізують поставлені задачі.
4. Розроблено ПЗ для створення онлайн-оголошень з продажу, покупки або обміну товарами та послугами.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Інформація з сайту «*yiiframework.com*» - Режим доступу: *https://www.yiiframework.com/doc/guide/1.1/uk/topics.webservice*.
2. Інформація з сайту «*capterra.com*» — Режим доступу: *https://www.capterra.com/event-management-software/*.
3. Інформація з сайту «*kanbanchi.com*» - Режим доступу: *https*://*www.kanbanchi.com*/
4. Інформація з сайту «*Laravel.com*» - Режим доступу: *https*://*laravel.com*/.
5. Інформація з сайту «*studopedia.su*» — Режим доступу: *https*://*studopedia.su*/18_23238_*arhitektura-informatsionnih-sistem-lokalnaya-klient-server-dvuh-i-trehurovnevaya-arhitektura.html*
6. Граді Буч, Джеймс Рамбо, Івар Якобсон Введення в *UML* від творців мови, Москва, 2010, 496 с.
7. Алекс Янг, Майк Кантелон, *Node.js* у дії, 2018, 432 с.
8. Інформація з сайту «*php.su*» — Режим доступу: *http*://*www.php.su*/*php*/?*php*
9. Інформація з сайту «*php.su*» — Режим доступу: *http*://*www.php.su*/*mysql*/?*info*
10. Семмі П'юрівал Основи розробки веб-додатку. Пітер, 2015. 272 с.
11. Дженніфер Роббінс *HTML5*. Кишеньковий довідник. Вільямс, 2015. 192 с.
12. *David A. Crowder Building a Web Site*. *For Dummies*, 2010. 360 *p.*
13. Девід Скляр, Адам Трахтенберг *PHP* Рецепти програмування. 3-тє вид. Пітер, 2015. 784 с.
14. Бейлі Лінн, Моррісон Майкл Вивчаємо *PHP* та *MySQL*. Ексмо, 2010. 800 с.
15. Девід Фленаган *JavaScript*. Детальний посібник. 6-те вид. Символ-Плюс, 2012. 1080 с.

16. Державний стандарт України. Інформація та документація. Бібліографічне посилання. Загальні положеннята правила складання. ДСТУ 8302:2015.

17. Державний стандарт України. Документація, звіти в сфері науки і техніки. Структура і правила оформлення. ДСТУ 3008-95.

18. Положення про дипломні роботи (проекти) випускників Національного Авіаційного Університету Київ, 2006. 72 с.

19. Інформація з сайту «*getcomposer.org*» — Режим доступу: *https://getcomposer.org/*

20. Інформація з сайту «*laravel.com*» - Режим доступу: *https://laravel.com/*

21. Інформація з сайту «*laravel.su*» - Режим доступу: *https://laravel.su/*

22. Джош Локхарт, *PHP*: Правильний шлях - Режим доступу: *https://getjump.github.io/ru-php-the-right-way/*

23. *Steve Prettyman, Learn PHP* 7: *Object Oriented Modular Programming using HTML*5, *CSS*3, *JavaScript, XML, JSON, and MySQL*, 2015. 316 с.

24. Інформація з сайту «*packagist.org*» - Режим доступу: *https://packagist.org/packages/mcamara/laravel-localization.*

25. Інформація з сайту «*github.com*» - Режим доступу: *https://github.com/spatie/laravel-translatable*

26. Інформація з сайту «*github.com*» - Режим доступу: *https://github.com/codezero-be/laravel-unique-translation*

27. Інформація з сайту «*github.com*» - Режим доступу: *https://github.com/spatie/laravel-sitemap*

28. Інформація з сайту «*habr.com*» - Режим доступу: *https://habr.com/ru/post/*38730/

29. Інформація з сайту «*nodejs.org*» - Режим доступу: *https://nodejs.org/uk/docs/*

30. Інформація з сайту «*getbootstrap.com*» - Режим доступу: *https://getbootstrap.com/docs/*5.1/*getting-started/introduction/*

ДОДАТОК А

Лістинг кращих рішень в коді

Інструкція до встановлення і лістинг коду.

На платформі де буде встановлено вебдодаток має бути доступ до консолі розробника, та мають бути встановлено менеджер пакетів для мови *PHP* «*composer*» і менеджер пакетів для мови *JS* – «*npm*».

Встановлення *Laravel*:

composer create-project laravel/laravel project-name

cd project-name

Після встановлення і входу до папки з проектом потрібно встановити пароль і назву таблиці в базі даних та виконати такі команди:

composer require mcamara/laravel-localization

php artisan vendor:publish --provider="Mcamara\LaravelLocalization\LaravelLocalizationServiceProvider"

composer require spatie/laravel-translatable

php artisan vendor:publish --provider="Spatie\Translatable\TranslatableServiceProvider"

composer require codezero/laravel-unique-translation

composer require spatie/laravel-sitemap

php artisan vendor:publish --provider="Spatie\Sitemap\SitemapServiceProvider" --tag=config

composer require spatie/laravel-sitemap

php artisan migrate

Далі приведу до прикладу мої приклади реалізації принципу *DRY* (з англ. *Don’t Repeat Youself*).

Я зробив розділення файлів, де зберігаються можливі шляхи на сайті, з розділенням на адмін панель та загальнодоступну частину, і конфігурація зроблена так, що власник ресурсу сам керує за яким шляхом зайти в адміністративну панель.

```
class RouteServiceProvider extends ServiceProvider
{
    public static $HOME = '/dashboard';
    public static $ADMIN_HOME;

    protected $namespace = 'App\\Http\\Controllers';

    public function __construct($app)
    {
        parent::__construct($app);
        $this::$ADMIN_HOME = config("app.admin_segment");
    }

    public function boot()
    {
        $this->configureRateLimiting();
        parent::boot();
    }

    public function map()
    {
```

```
    $this->mapFortifyRoutes();
    $this->mapJetstreamRoutes();
    $this->mapAdminRoutes();
    $this->mapWebRoutes();
}

protected function mapWebRoutes()
{
    Route::prefix(LaravelLocalization::setLocale())
        ->middleware('web')
        ->namespace($this->namespace)
        ->as("web.")
        ->group(base_path('routes/web.php'));
}

protected function mapAdminRoutes()
{
    Route::prefix(LaravelLocalization::setLocale() . '/' . $this::$ADMIN_HOME)
        ->middleware('admin')
        ->namespace($this->namespace)
        ->as("admin.")
        ->group(base_path('routes/admin.php'));
}

protected function mapFortifyRoutes()
{
    Route::prefix(LaravelLocalization::setLocale())
        ->middleware(config('fortify.middleware', ['web', "admin"]))
        ->group(base_path('routes/fortify.php'));
}
```

```
    protected function mapJetstreamRoutes()
    {
        Route::prefix(LaravelLocalization::setLocale())
            ->middleware(config('jetstream.middleware', ['web', "admin"]))
            ->group(base_path('routes/jetstream.php'));
    }

    protected function configureRateLimiting()
    {
        RateLimiter::for($this::$ADMIN_HOME, function (Request $request) {
            return Limit::perMinute(60)->by(optional($request->user())->id ?: $request->ip());
        });
    }
}
```

Ось моя реалізація можливих шляхів в адміністративній панелі:

```
Route::group(['middleware' => 'auth', 'namespace' => RequestHelper::adminNamespace()], function () {
    Route::get('/', 'Panel\PanelController@home')->name("home");

    Route::resource('page', 'Page\PageController', ['except' => 'show']);

    Route::resource('service', 'Service\ServiceController', ['except' => 'show']);
    Route::resource('service-category', 'Service\ServiceCategoryController', ['except' => 'show']);

    Route::resource('product', 'Product\ProductController', ['except' => 'show']);
```

```
    Route::resource('product-category', 'Product\ProductCategoryController', ['except' => 'show']);

    Route::resource('article', 'Article\ArticleController', ['except' => 'show']);
    Route::resource('article-category', 'Article\ArticleCategoryController', ['except' => 'show']);
});
```

А також реалізація шляхів у загальнодоступній частині сайту:

```
Route::group(['namespace' => RequestHelper::webNamespace()], function () {
    Route::group(['middleware' => ['auth', 'verified']], function (){
        Route::resource('user/product', "Account\ProductController", ['except' => 'show']);
        Route::resource('user/service', "Account\ServiceController", ['except' => 'show']);
    });

    Route::get('/', 'Page\PageController@show')->name("home");
    Route::get('404', 'Page\PageController@error')->name("404");
    Route::get('{first?}/{second?}/{third?}/{fourth?}/{fifth?}/{sixth?}',
'Page\PageController@show')->name("show");
});
```

Оскільки в більшості контроллерів *Laravel* код повторюється, то код в мене відповідає цьому прикладу, тільки зі зміною деяких назв, тому що взагалі обійтись одним контроллером не можливо:

```
class PageController extends Controller
{
    public function __construct(PageContract $repository)
    {
        $this->repository = $repository;
    }
```

```
    public function renderAbleTitles(): array
    {
        return [
            'index' => __("labels.title_pages"),
            'create' => __("labels.title_add", ["name" => __("labels.pages")]),
            'edit' => __("labels.title_edit", ["name" => __("labels.page")]),
        ];
    }

    public function index()
    {
        return $this->setModels($this->repository->listPages())->render();
    }

    public function create()
    {
        return $this->setModel($this->repository->newPage())->render();
    }

    public function store()
    {
        return $this->repository->fillAndSave($this->repository->newPage()) ? $this->redirectAfterSuccessfulAction() : $this->backWithError();
    }

    public function edit($id)
    {
        return $this->setModel($this->repository->findPageById($id))->render();
    }
```

```
    public function update($id)
    {
        return $this->repository->fillAndSave($this->repository->findPageById($id)) ? $this->redirectAfterSuccessfulAction() : $this->backWithError();
    }

    public function destroy($id)
    {
        $result = $this->repository->deletePage($id);

        if (is_array($result)) {
            return $this->backWithError($result["error"]);
        } else {
            return $this->redirectAfterSuccessfulAction();
        }
    }
}
```

Також я вирішив додати батьківський класс *Model.php*, і в ньому перевизначати певні функції та структури з базового набору *Laravel*:

```
abstract class Model extends BaseModel
{
    protected static string $listOptionKeyAttribute = 'id';
    protected static string $listOptionValueAttribute = 'title';
    protected static string $titleKey = 'title';
    public $asterisked = [];

    public function __construct(array $attributes = [])
    {
```

```
    parent::__construct($attributes);

    $this->setDefaultValues();
}

public function titleValue()
{
    return $this->{$this::$titleKey};
}

private function setDefaultValues(): void
{
    if (method_exists($this, 'setDefaultLocale')) {
        $this->setDefaultLocale();
    }

    if (method_exists($this, 'setDefaultNumber')) {
        $this->setDefaultNumber();
    }

    if (method_exists($this, 'setDefaultSort')) {
        $this->setDefaultSort();
    }

    if (method_exists($this, 'setDefaultSortInNav')) {
        $this->setDefaultSortInNav();
    }

    if (method_exists($this, 'setDefaultColumnInNav')) {
        $this->setDefaultColumnInNav();
```

```
    }

    if (method_exists($this, 'setDefaultPrice')) {
        $this->setDefaultPrice();
    }

    if (method_exists($this, 'setDefaultPrice2')) {
        $this->setDefaultPrice2();
    }
}

public function setCorrectDefaultValues(): void
{
    foreach ($this->attributes as $k => $v) {
        if ($v === '<p><br></p>') {
            $this->setAttribute($k, null);
        }
    }
}

public static function boot()
{
    parent::boot();

    self::creating(function($model) {
        $model->beforeSave(true);
    });

    self::created(function($model) {
        $model->afterSave(true);
```

```
        });

        self::updating(function($model) {
            $model->beforeSave(false);
        });

        self::updated(function($model) {
            $model->afterSave(false);
        });

        self::deleting(function($model) {
            $model->beforeDelete();
        });

        self::deleted(function($model) {
            $model->afterDelete();
        });
    }

    protected function beforeSave(bool $create): void
    {
        $this->setDefaultValues();

        if (method_exists($this, 'generateSlug')) {
            $this->generateSlug();
        }

        if (method_exists($this, 'setPosition')) {
            if ($create) {
                $this->setPosition();
```

```
        }
    }
}

protected function afterSave(bool $create): void
{
}

protected function beforeDelete(): void
{
    if (method_exists($this, 'deleteImage')) {
        $this->deleteImage();
    }

    if (method_exists($this, 'deleteBottomImage')) {
        $this->deleteBottomImage();
    }

    if (method_exists($this, 'deleteFile')) {
        $this->deleteFile();
    }

    if (method_exists($this, 'deleteFormFile')) {
        $this->deleteFormFile();
    }

    if (method_exists($this, 'deleteCvFile')) {
        $this->deleteCvFile();
    }
}
```

```
    protected function afterDelete(): void
    {
        if (method_exists($this, 'repositioning')) {
            $this->repositioning();
        }
    }

    public function indexRoute(array $parameters = [], $absolute = true): ?string
    {
        return $this->route($this->routeToModel(), 'index', $parameters, $absolute);
    }

    public function createRoute(array $parameters = [], $absolute = true): ?string
    {
        return $this->route($this->routeToModel(), 'create', $parameters, $absolute);
    }

    public function storeRoute(array $parameters = [], $absolute = true): ?string
    {
        return $this->route($this->routeToModel(), 'store', $parameters, $absolute);
    }

    public function showRoute(array $parameters = [], $absolute = true): ?string
    {
        return $this->route($this->routeToModel(), 'show', [$this->primaryValue()] +
$parameters, $absolute);
    }

    public function editRoute(array $parameters = [], $absolute = true): ?string
```

```
    {
        return $this->route($this->routeToModel(), 'edit', [$this->primaryValue()] + $parameters, $absolute);
    }

    public function updateRoute(array $parameters = [], $absolute = true): ?string
    {
        return $this->route($this->routeToModel(), 'update', [$this->primaryValue()] + $parameters, $absolute);
    }

    public function destroyRoute(array $parameters = [], $absolute = true): ?string
    {
        return $this->route($this->routeToModel(), 'destroy', [$this->primaryValue()] + $parameters, $absolute);
    }

    public function destroyImageRoute($absolute = true): ?string
    {
        return route('admin.' . $this->modelName().'.image.destroy', [], $absolute);
    }

    public function destroyFileRoute($absolute = true): ?string
    {
        return route('admin.' . $this->modelName() . '.file.destroy', [], $absolute);
    }

    public function route($name, string $action, $parameters = [], $absolute = true): ?string
    {
```

```
        return Route::has("$name.$action") ? route("$name.$action", $parameters, $absolute) : null;
    }

    public function routeToModel(): string
    {
        return RequestHelper::pathName() . '.' . $this->modelName();
    }

    public function modelName(): string
    {
        $name = class_basename(get_called_class());
        $name = Str::snake($name, '-');

        return Str::lower($name);
    }

    public function primaryValue()
    {
        return $this->{$this->primaryKey};
    }

    public function validate(): array
    {
        return Validator::make(request()->all(), $this->rules(), [], $this->attributeLabels())->validate();
    }

    public function isAsterisked(string $key): bool
    {
```

```
    return in_array($key, $this->asterisked);
}

public function rules(): array
{
    return [];
}

public function attributePlaceholders(): array
{
    return [];
}

public function attributeHints(): array
{
    return [];
}

public function gridAttributes(): array
{
    return [];
}

public function gridValues(): array
{
    return [];
}

public function formMainFields(): array
{
```

```
    return [];
}

public function formLocaleFields(): array
{
    return [];
}

public function getAttributeLabel(string $name): string
{
    return $this->attributeLabels()[$name] ?? $name;
}

public function attributeLabels(): array
{
    return $this->retrieveHarvestedAttributeLabels($this->primaryKey);
}

public function getAttributePlaceholder(string $name): string
{
    return $this->attributePlaceholders()[$name] ?? '';
}

public function getAttributeHint(string $name): string
{
    return $this->attributeHints()[$name] ?? '';
}

public function formImageFields(): array
{
```

```
        return method_exists($this, 'setFormImageFields') ? $this->setFormImageFields() :
[];
    }

    public function formVideoFields(): array
    {
        return method_exists($this, 'setFormVideoFields') ? $this->setFormVideoFields() :
[];
    }

    public function formFileFields(): array
    {
        return method_exists($this, 'setFormFileFields') ? $this->setFormFileFields() : [];
    }

    public function formMetaFields(): array
    {
        return method_exists($this, 'setFormMetaFields') ? $this->setFormMetaFields() :
[];
    }

    private function harvestedData(): array
    {
        return [
            'rules' => [
                'title' => 'required|array',
                'title.*' => 'required|string|max:100',
                'content' => 'nullable|array',
                'content.*' => 'nullable|string',
                'active' => 'nullable|boolean',
```

```
            'in_top_menu' => 'nullable|boolean',
            'in_bottom_menu' => 'nullable|boolean',
            'parent_id' => 'nullable|integer',
            'category_id' => 'required|integer',
            '_category_type_ids' => 'nullable|array',
            '_category_type_ids.*' => 'nullable|integer',
            'status' => 'required|alpha_dash',
            'type' => 'required|alpha_dash',
            'layout' => 'required|alpha_dash',
            'type_delivery' => 'required|alpha_dash',
            'legal_status' => 'nullable|alpha_dash',
            'price' => 'nullable|integer|min:0',
            'publish' => 'nullable|date',
            'slug' => 'nullable|array',
            'slug.*' => ["nullable", "string", "max:100", UniqueTranslationRule::for($this->getTable(), "slug")->ignore($this->id)],
            'meta_title' => 'nullable|array',
            'meta_title.*' => 'nullable|string|max:70',
            'meta_description' => 'nullable|array',
            'meta_description.*' => 'nullable|string|max:180',
            'meta_keywords' => 'nullable|array',
            'meta_keywords.*' => 'nullable|string|max:255',
        ],
        'attributeLabels' => [
            'position' => 'Позиция',
            'author_id' => 'Автор',
            'title' => 'Полное название',
            'content' => 'Содержимое',
            'active' => 'Отображение',
            'in_top_menu' => 'Выводить в верхнем меню',
```

```
    'in_bottom_menu' => 'Выводить в нижнем меню',
    'parent_id' => 'Родительская категория',
    '_category_type_ids' => 'Типы категорий',
    'category_id' => 'Категория',
    'status' => 'Статус',
    'legal_status' => 'Правовой статус',
    'price' => 'Цена, грн',
    'popularity' => 'Популярность',
    'type_delivery' => 'Тип доставки',
    'layout' => 'Шаблон',
    'type' => 'Тип',
    'publish' => 'Дата публикации',
    'slug' => 'SEO-Friendly URL',
    'meta_title' => 'Meta Title',
    'meta_description' => 'Meta Description',
    'meta_keywords' => 'Meta Keywords',
],
'attributePlaceholders' => [
    'parent_id' => '[Без родительской категории]',
    '_category_type_ids' => '[Без типов категории]',
    'category_id' => '[Без категории]',
    'status' => '[Без статуса]',
    'layout' => '[Без шаблона]',
    'type_delivery' => '[Без типа доставки]',
    'slug' => 'Оставьте пустым для автоматического заполнения',
],
'attributeHints' => [
    'image' => 'Изображение',
    '_image' => 'Изображение',
],
```

```
        ];
    }

    private function retrieveHarvestedElement(string $element, $keys): array
    {
        $retrieved = [];

        $keys = (array)$keys;

        foreach ($this->harvestedData()[$element] as $key => $value) {
            if (in_array($key, $keys)) {
                $retrieved[$key] = $value;
            }
        }

        return $retrieved;
    }

    protected function retrieveHarvestedRules($keys): array
    {
        return $this->retrieveHarvestedElement('rules', $keys);
    }

    protected function retrieveHarvestedAttributeLabels($keys): array
    {
        return $this->retrieveHarvestedElement('attributeLabels', $keys);
    }

    protected function retrieveHarvestedAttributePlaceholders($keys): array
    {
```

```
        return $this->retrieveHarvestedElement('attributePlaceholders', $keys);
    }

    protected function retrieveHarvestedAttributeHints($keys): array
    {
        return $this->retrieveHarvestedElement('attributeHints', $keys);
    }

    public static function getList($where = null): array
    {
        return static::orderBy(static::$listOptionValueAttribute)
            ->where($where)
            ->pluck(static::$listOptionValueAttribute, static::$listOptionKeyAttribute)
            ->all();
    }

    public function isUnDeletable(): bool
    {
        return false;
    }

    public function isUnEditable(): bool
    {
        return false;
    }
}
```

Завдяки такому рішенню я маю змогу гнучко налаштовувати поля та їх виведення в список адміністративної панелі та користувацької частини сайту:

class Service extends Model

```
{
    use HasFactory, HasTranslations, HasPosition, HasLocale, HasTimestamps,
HasEnum, HasSlug;

    public $fillable = [
        'slug',
        'author_id',
        'category_id',
        'type',
        'title',
        'content',
        'price',
        'popularity',
        'position',
        'active',
    ];

    public $asterisked = [
        'category_id',
        'type',
        'title',
        'content',
        'price',
    ];

    public $translatable = ['slug', 'title', 'content'];

    public function rules(): array
    {
        return $this->retrieveHarvestedRules([
```

```
            'slug',
            'slug.*',
            'category_id',
            'title',
            'title.*',
            'content',
            'content.*',
            'type',
            'popularity',
            'price',
            'active',
        ]);
    }

    public function attributeLabels(): array
    {
        return $this->retrieveHarvestedAttributeLabels([
            'slug',
            'position',
            'author_id',
            'category_id',
            'title',
            'content',
            'type',
            'popularity',
            'price',
            'active',
        ]);
    }
```

```
public function attributePlaceholders(): array
{
    return $this->retrieveHarvestedAttributePlaceholders([
        'category_id',
        'type',
        'popularity',
        'slug',
    ]);
}

public function gridAttributes(): array
{
    return [
        AttributeHelper::get("position"),
        AttributeHelper::get('title'),
        AttributeHelper::get('slug'),
        AttributeHelper::get('price'),
        AttributeHelper::relation('author_id'),
        AttributeHelper::relation('category_id'),
        AttributeHelper::get('type'),
        AttributeHelper::get('popularity'),
        AttributeHelper::get('active'),
        AttributeHelper::actions(),
    ];
}

public function gridValues(): array
{
    return [
        ValueHelper::get($this->position),
```

```
            ValueHelper::get($this->title),
            ValueHelper::get($this->slug),
            ValueHelper::get($this->price),
            ValueHelper::relation($this->author),
            ValueHelper::relation($this->category),
            ValueHelper::get($this->type, null, true),
            ValueHelper::get($this->popularity, null, true),
            ValueHelper::get($this->active),
            ValueHelper::actions($this),
        ];
    }

    public function formMainFields(): array
    {
        return [
            FieldHelper::checkbox('active'),
            FieldHelper::select('category_id', ServiceCategory::getList()),
            FieldHelper::select('type', self::getEnum("type")),
            FieldHelper::select('popularity', self::getEnum("popularity")),
            FieldHelper::textInput("price", null, null, null, false, ["class" => "price-mask"]),
        ];
    }

    public function formLocaleFields(): array
    {
        return [
            FieldHelper::textInput('slug'),
            FieldHelper::textInput('title'),
            FieldHelper::editor('content'),
        ];
```

```
}

public function gridAccountAttributes(): array
{
    return [
        AttributeHelper::get("position"),
        AttributeHelper::get('title'),
        AttributeHelper::get('price'),
        AttributeHelper::relation('category_id'),
        AttributeHelper::get('type'),
        AttributeHelper::get('popularity'),
        AttributeHelper::get('active'),
        AttributeHelper::actions(),
    ];
}

public function gridAccountValues(): array
{
    return [
        ValueHelper::get($this->position),
        ValueHelper::get($this->title),
        ValueHelper::get($this->price),
        ValueHelper::relation($this->category),
        ValueHelper::get($this->type, null, true),
        ValueHelper::get($this->popularity, null, true),
        ValueHelper::get($this->active),
        ValueHelper::actions($this),
    ];
}
```

```
    public function formAccountMainFields(): array
    {
        return [
            FieldHelper::checkbox('active'),
            FieldHelper::select('category_id', ServiceCategory::getList()),
            FieldHelper::select('type', self::getEnum("type")),
            FieldHelper::textInput("price", null, null, null, false, ["class" => "price-mask"]),
        ];
    }

    public function formAccountLocaleFields(): array
    {
        return [
            FieldHelper::textInput('title'),
            FieldHelper::editor('content'),
        ];
    }

    public function category()
    {
        return $this->belongsTo(ServiceCategory::class, "category_id", "id")-
>orderBy('position');
    }

    public function author()
    {
        return $this->belongsTo(User::class, "author_id", "id");
    }
}
```

В цьому додатку було представлено мої найкращі рішення по зменшенню обсягу коду, спрощення подальшого супроводу даного проекту, і закладено можливість глобального розширення можливостей вебдодатку.