Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: **Розробка веб-платформи для збуту сільськогосподарської продукції**

Виконав(ла): студент(ка) 6 курсу, групи СТм-61
спеціальності 126 «Інформаційні системи та технології»
(шифр і назва спеціальності)

_____ Слободяник А.М.
(підпис) (прізвище та ініціали)

Керівник _____ Готович В.А.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Мацюк О.В.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ Шкодзінський О.К.
(підпис) (прізвище та ініціали)

Тернопіль
2020

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
« 22 » грудня 2020 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня **магістр**
(назва освітнього ступеня)

за спеціальністю 126 «Інформаційні системи та технології»
(шифр і назва спеціальності)

студенту Слободянику Анатолію Миколайовичу
(прізвище, ім'я, по батькові)

1. Тема роботи **Розробка веб-платформи для збуту сільськогосподарської продукції**

Керівник роботи **Готович В.А., к.т.н.**
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « *6* » *листопада* 2020 року № *4/7-826* .
2. Термін подання студентом завершеної роботи 22 грудня 2020 року
3. Вихідні дані до роботи Літературні джерела щодо розробки веб-сервісів і розробки програного забезпечення

4. Зміст роботи (перелік питань, які потрібно розробити)
1 Аналітичний огляд існуючих рішень. 2 Проектування програмного продукту. 3 Програмна реалізація. 4 Охорона праці та безпека в надзвичайних ситуаціях

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
Титульний слайд, актуальність теми сільського господарства, слайд із інформацією про об`єкт дослідження, предмет дослідження, метою дослідження,слайд із науковою новизною, слайд про проблеми збуту сільськогосподарської продукції, слайд про вплив карантинних обмежень сферу, слайд про постійних клієнтів які зазнали впливу карантинних обмежень, слайд про вирішення проблеми, слайд про переваги переходу в онлайн формат, слайд із Списком використаних технологій, слайд із скірншотом головної сторінки сервісу «Market Place», слайд з БД, слайд з архітектурою, слайд із висновками

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Дмитроца Л. П., доцент		
Безпека в НС	Клепчик В. М., ст. вик.		

7. Дата видачі завдання 21 вересня 2020 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Ознайомлення з завданням до кваліфікаційної роботи	21.09.20-27.09.20	*Виконано*
2	Підбір наукових джерел щодо проблем збуту сількогосподарської продукції	28.09.20-04.10.20	*Виконано*
3	Переклад та опрацювання наукових джерел	05.10.20-11.10.20	*Виконано*
4	Аналіз предмету дослідження та предметної області	12.10.20-18.10.20	*Виконано*
5	Оформлення розділу «Аналітичний огляд існуючих рішень»	19.10.20-25.10.20	*Виконано*
6	Оформлення розділу «Проектування програмного продукту»	26.10.20-01.11.20	*Виконано*
7	Оформлення розділу «Програмна реалізація»	02.11.20-08.11.20	*Виконано*
8	Виконання завдання до підрозділу «Охорона праці»	09.11.20-15.11.20	*Виконано*
9	Виконання завдання до підрозділу «Безпека в надзвичайних ситуаціях»	16.11.20-22.11.20	*Виконано*
10	Оформлення кваліфікаційної роботи	23.11.20-29.11.20	*Виконано*
11	Нормоконтроль	30.11.20-05.12.20	*Виконано*
12	Перевірка на плагіат	09.12.20	*Виконано*
13	Попередній захист кваліфікаційної роботи	14.12.20	*Виконано*
14	Захист кваліфікаційної роботи	23.10.20	

Студент _____ *Слободяник А.М.*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Готович В.А.*
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка веб-платформи для збуту сільськогосподарської продукції // Кваліфікаційна робота освітнього рівня “Магістр” // Слободяник Анатолій Миколайович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп’ютерно-інформаційних систем і програмної інженерії, кафедра комп’ютерних наук, група СТм-61 // Тернопіль, 2020 // С.– 90 , рис. – 34 ,табл. – 8, додат. – 3 , бібліогр. – 33 .

Ключові слова: РОЗРОБКА, ВЕБ-ПЛАТФОРМА, АРХІТЕКТУРА, ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ, АГРОПРОМИСЛОВИЙ КОМПЛЕКС.

Метою дослідження є створення рішення для вирішення проблем із реалізацією та збуту сільськогосподарської продукції з якими стикається сфера агропромислового комплексу.

Завдання кваліфікаційної роботи включає в себе детальний аналіз існуючих рішень, з врахуванням всіх переваг і недоліків, щоб створити систему, яка б ефективно виконувала поставлені на неї задачі, була максимально зручною і простою для кінцевого користувача.

Також проведено детальний аналіз і обрано найкращі архітектурні рішення для реалізації ідеальної веб-платформи, також обрано стек технологій які дозволять швидко і зручно створювати, вдосконалювати і у майбутньому підтримувати даний веб-сервіс.

Основна ідея нового рішення це розділення оголошень на два види «Куплю» і «Продам». Саме таке рішення дозволить покрити весь ринок оголошень і бути максимально ефективним не лише для фермерів і ферм, а й для людей які мають зв`язок із реалізацією сільськогосподарської продукції, і дозволить вирішити всі або майже всі проблеми із якими вони стикалися і стикаються сьогодні.

Рішення пришвидшить процес реалізації продукції підбором оголошень рекомендацій, які найкраще підходять під конкретне оголошення користувача.

ANNOTATION

Web-platform development for agricultural products sales // Diploma thesis Master degree// Slobodyanyk Anatolii Mykolayovych // Ternopil' Ivan Pul'uj National Technical University, Faculty of Computer Information System and Software Engineering, Department of Computer Science, group STm-61 // Ternopil, 2020 // P.– 90, Tables – 8, Fig. – 34,Annexes. – 3 , References – 33 .

Keywords: DEVELOPMENT, WEB PLATFORM, INFORMATION TECHNOLOGIES, ARGICULTURAL COMPLEX.

The purpose of the research is to create a solution to solve problems with the sale and marketing of agricultural products faced by the argo-industrial complex.

The task includes a detailed analysis of existing web platforms, taking into account all the advantages and disadvantages, to create a system that would effectively perform the tasks and would be a convenient and easy to use for the end user.

Also done a detailed analysis and selected the best architectural solutions for the implementation of an ideal web planform, also selected a stack of technologies that will quickly and easily create, improve and support this web service.

The main idea of this new solution is to separate adverts into two types "Buy" and "Sell". This solution will help to cover the entire market of adverts and be the most effective not only for farmers and farms, but also for people who have a connection with the sale of agricultural products, and will solve all or almost all the problems which they have faced before and during quarantine period.

The solution will speed up the product sales process by selecting adverts recommendations that are the best suited for a particular user ad.

ЗМІСТ

ВСТУП

Актуальність теми роботи. Землеробство, сьогодні ж сільське господарство, протягом тисячоліть було частиною людського життя. Діяльність, яка тривалий час забезпечувала населення світу провізією. Майже кожна країна має сільське господарство, яке по різному розвинуте і з швидкими темпами невпинно розвивається за участі сучасних технологій.

Нова техніка, різні види добрив, види культур, методи, техніки, тисячолітній досвід, усе це зробило сільське господарство саме таким, яким воно є сьогодні.

Але поки залишаються не вирішеними проблеми, які не дозволяють даній діяльності розвиватися з достатньою швидкістю, рівномірно і вчасно поширювати, транспортувати провізію у місця і країни із слабше розвиненим сільським господарством, нагодувавши населення.

Ціни, які коливаються в залежності від врожайності сезону, що зі змінами клімату, ще сильніше проявляється, чи в залежності від того наскільки швидко той чи інший товар було поставлено вчасно, логістика є однією із найголовніших аспектів даної діяльності [3]. Все це є сучасними проблемами сільського господарства.

Із появою коронавірусу, карантинні обмеженнями створили умови які стали випробуваням на стійкість і міцність економіки країн, і сфер агропромислового комплексу також.

Це дозволило описаним проблемам краще проявитися і привернути увагу до моментів які слід було і потрібно вдосконалити зараз. Проблеми примусили учасників ринку боротися за виживання і шукати нові шляхи, і способи збуту та форми співпраці з посередниками [2].

При цьому масовий, і в основному, примусовий вихід в онлайн формат, показав чудові результати. Настільки ефективні, що Кабінет Міністрів України у програмі відновлення економіки окремо виділив і описав створення централізованого онлайн-сервісу, яким будуть користуватися малі і середні ферми для реалізації своєї продукції [1].

На даний момент не існує рішення, яке би вирішило всі проблеми реалізації сільськогосподарської продукції у такий складний період [7]. А рішення які є сьогодні не достатньо пристосовані саме для сільського господарства і не покривають весь ринок оголошень.

Тому розробка веб-платформи, яка дозволить користувачам, а саме фермерам, організаціям, фермам і людям які займаються реалізацією продукції цієї сфери, пришвидшити процес реалізації продукції в аграрній промисловості і уникнути проблем, є актуальною задачею.

Метою дослідження є детальний аналіз процесу реалізації сільськогосподарської продукції, а також аналіз існуючих рішень, які надають можливість користувачу торгівлею. Розробка власного рішення, яке б дало можливість пришвидшити і спростити реалізацію і торгівлю продукцією, з урахуванням всіх переваг і недоліків проаналізований рішень.

Для досягнення цілі кваліфікаційної роботи, необхідно вирішити такі **завдання**:

- дослідити та проаналізувати веб-сервіси, які можуть спростити реалізацію фермерам, порівняти їхні можливості, а також виділити переваги і недоліки систем;
- на основі результатів аналізу розробити специфікацію функціональних та нефункціональних вимог до майбутнього програмного продукту;
- розробити архітектуру програмного продукту та спроектувати структура бази даних;
- проаналізувати і обрати стек технологій і мов програмування для реалізації даного програмного продукту;
- розробити веб-платформу “Market Place” як вирішення проблем фермерів і людей які пов’язані із реалізацією сільськогосподарської продукції, згідно результатів проведених робіт у попередній пунктах.

Об’єкт дослідження - процес збуту сільськогосподарської продукції на ринку.

Науковою новизною роботи є розробка веб-платформи для вирішення задач збуту сільськогосподарської продукції яка, на відміну від існуючих рішень, спрощує роботу користувача за рахунок:

- чіткого поділу товарів та послуг по категоріях;
- забезпечення можливості створення оголошення на купівлю товару;
- усунення численних недоліків, в тому числі тривалого в часі завантаження сторінок;
- спрощення процесу реєстрації для нового користувача платформи.

Предмет досліджень - застосування технологій створення веб-орієнтованого програмного забезпечення для розробки веб-платформи “Market Place”.

Апробація результатів магістерської роботи. Окремі результати роботи представлені на двох наукових конференціях:

1. IX Міжнародна науково-технічна конференція молодих учених та студентів “Актуальні задачі сучасних технологій” на тему “Інформаційна система для вирішення проблем із реалізацією сільськогосподарської продукції”;

2. VIII науково-технічна конференція “Інформаційні моделі, системи та технології” на тему “Веб-платформа для вирішення проблем із реалізацією сільськогосподарської продукції”.

1 АНАЛІТИЧНИЙ ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

1.1 Аналіз існуючих рішень

Переглянувши найпопулярніші веб-платформи оголошень, кожна має свої переваги і недоліки або відсутні функції, які вкрай необхідні.

Одним із найпопулярніших сервісів, де можна створити оголошення є сервіс OLX [4]. Зручний інтерфейс, багато користувачів, але користувачу фермеру, не зразу вдасться зорієнтуватися з категорією де шукати потрібні товари. На головній сторінці відсутній перелік категорій які мають зв’язок з сільським господарством (рис. 1.1).

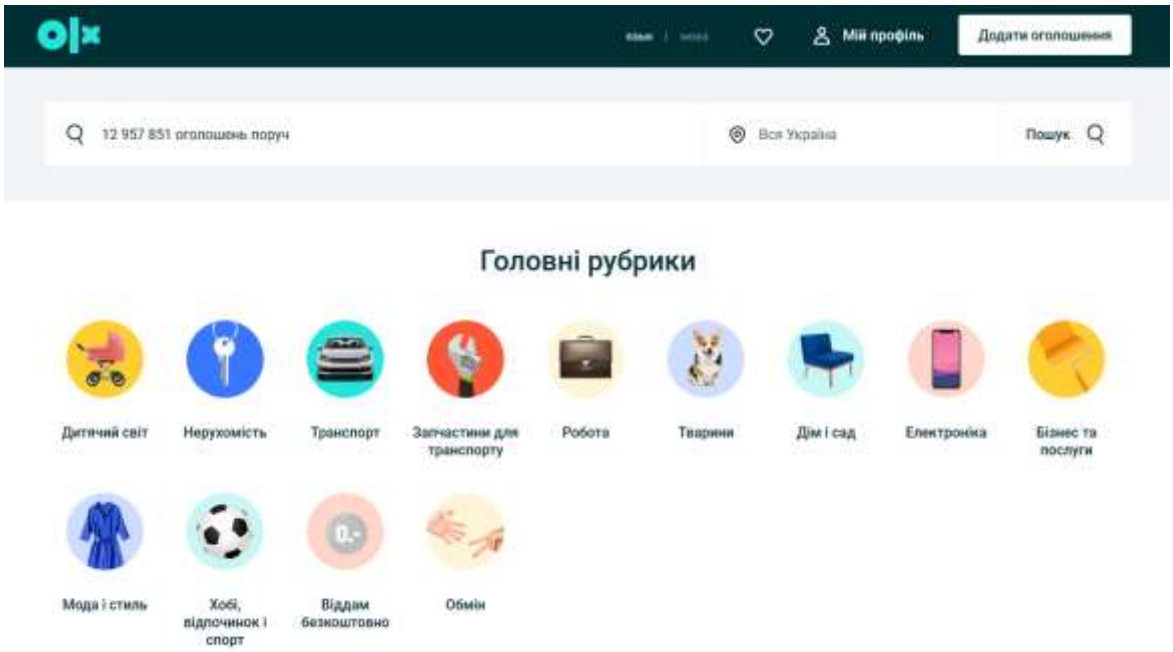

Рисунок 1.1 - Головна сторінка сервісу OLX

Це один з недоліків сервісу. Але після введення назви товару який нас цікавить у полі пошуку вже пропонується здійснити пошук у категорії “Продукти харчування” (рис. 1.2). Тому це не дуже критично, але все ж таки користувач не одразу з орієнтується, що слід робити.

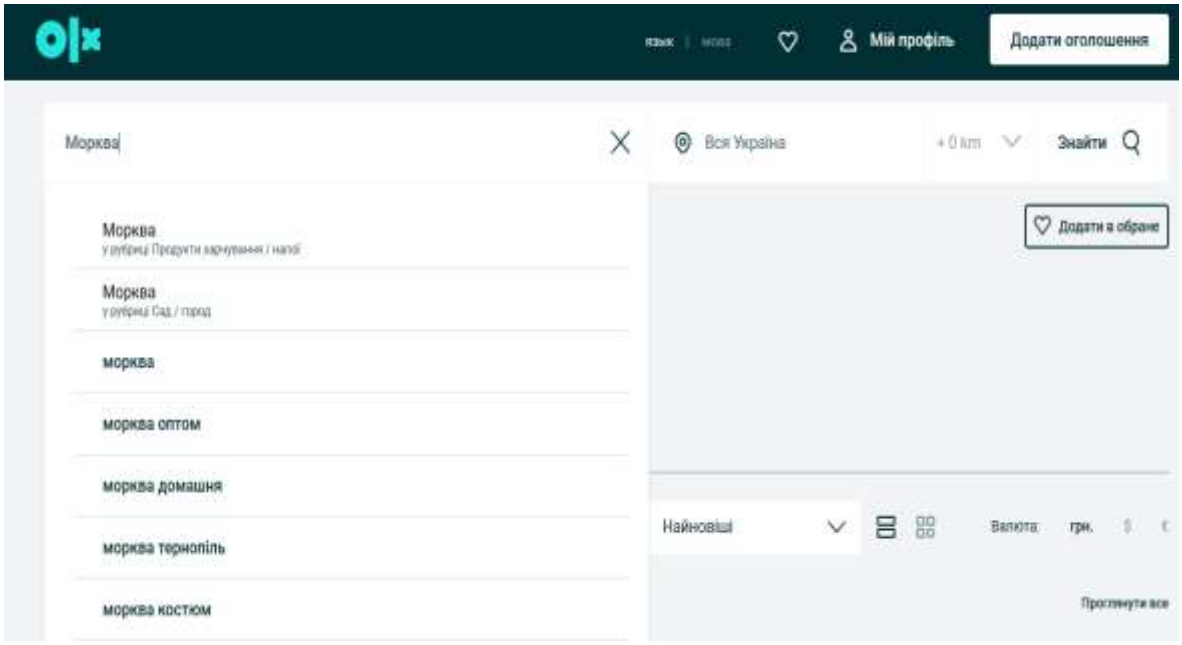

Рисунок 1.2 - Пропозиції пошуку

Вибравши дану категорію ми отримуємо оголошення про продаж товару який нас цікавить (рис. 1.3).

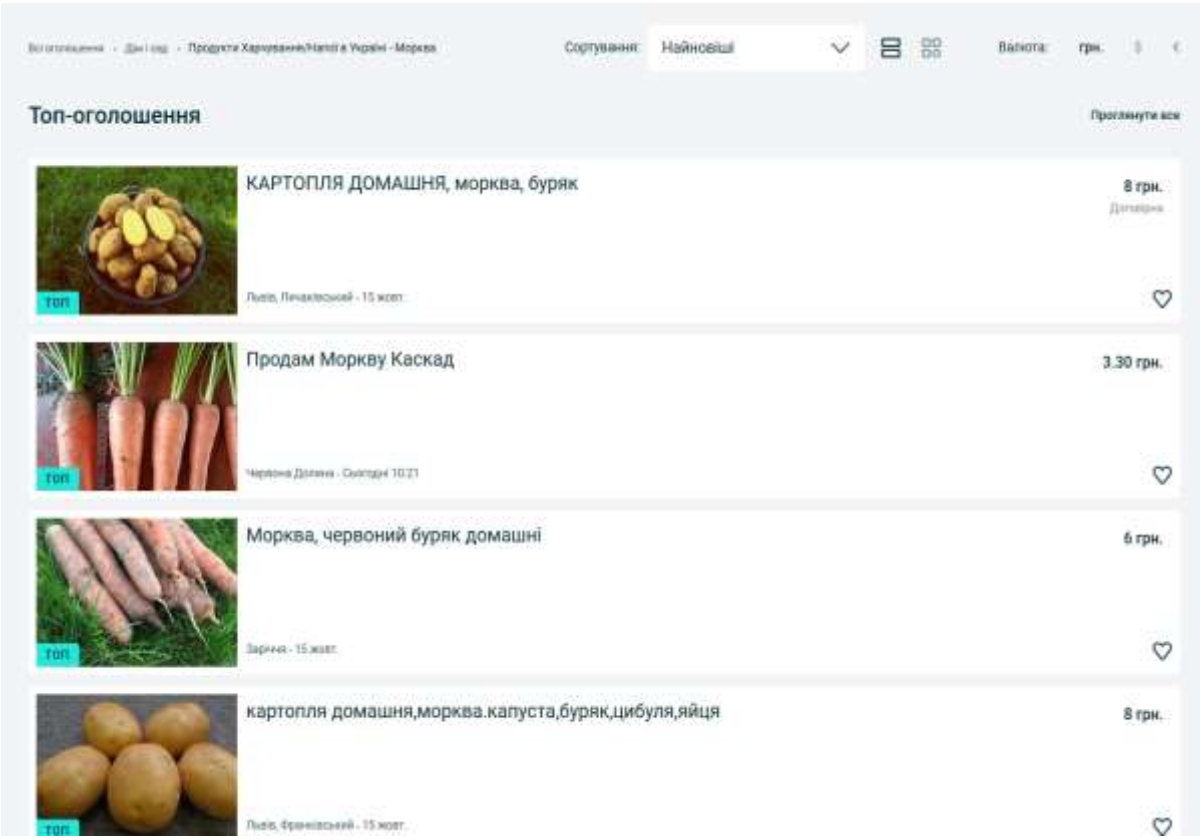

Рисунок 1.3 - Результат пошуку по обраній категорії

Ще одним недоліком є не чітке сортування товарів по категоріях. До прикладу сортування по категорії “Продукти харчування і напої” можна зустріти які не зовсім даній категорії (рис. 1.4 і 1.5).

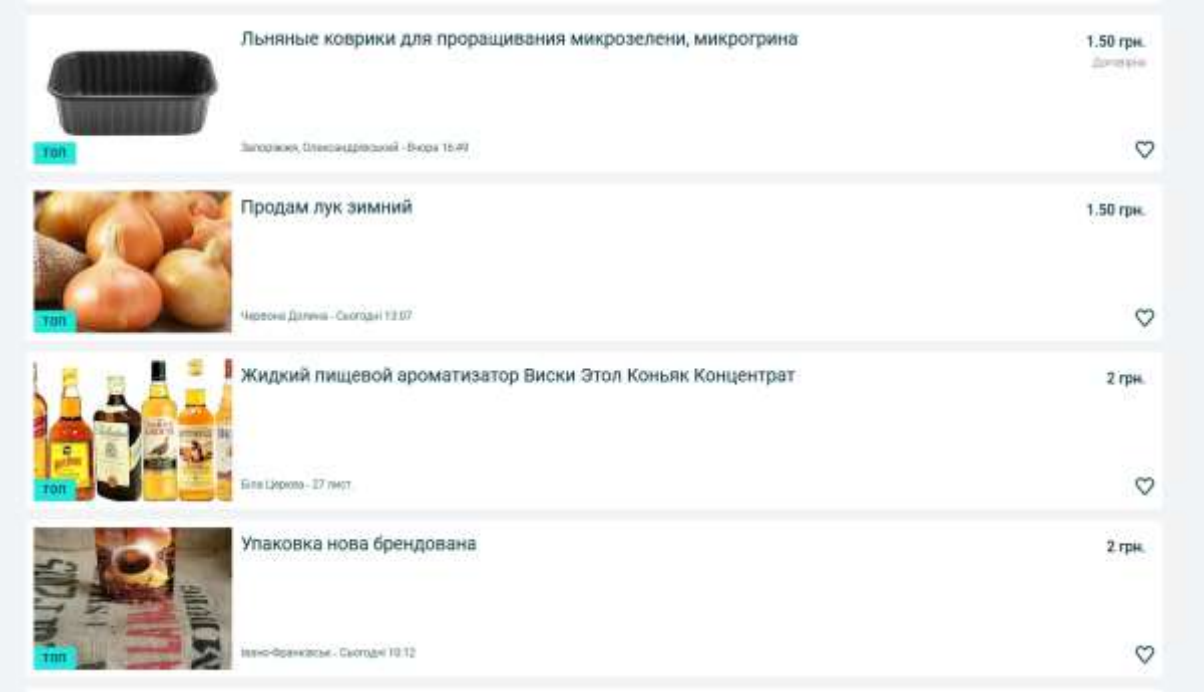

Рисунок 1.4 – Список товарів категорії “Продукти харчування”

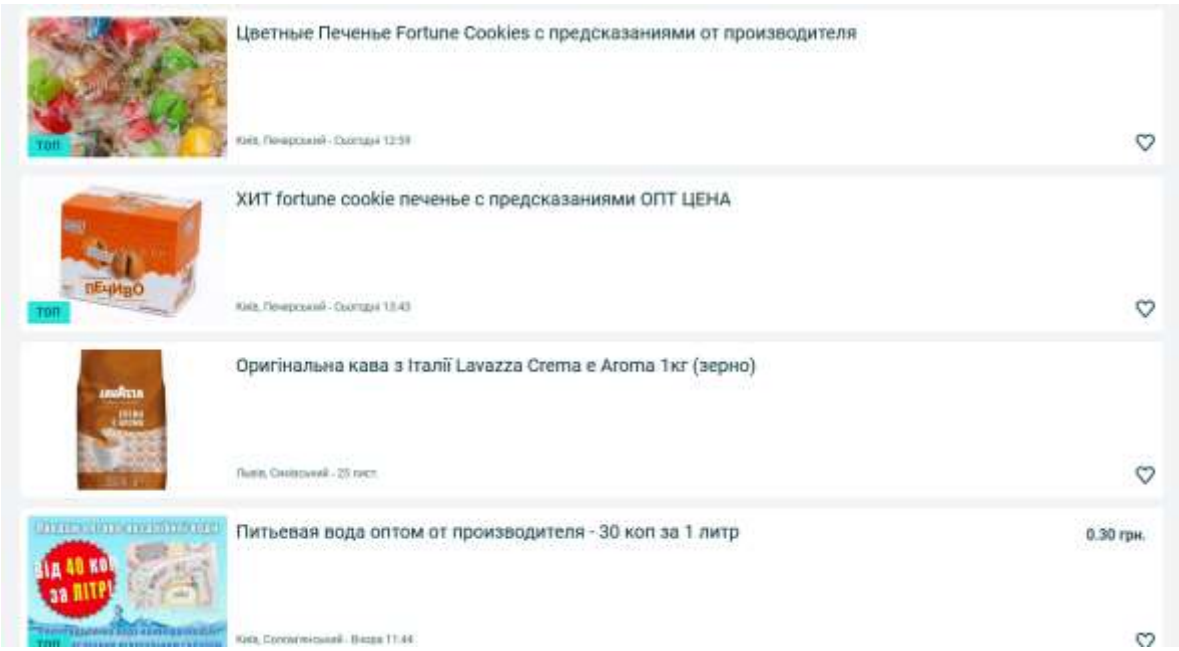

Рисунок 1.5 – Список товарів категорії “Продукти харчування”

Для сервісу оголошень - це дефект, який слід виправляти. Це не припустимо сервісу даного типу. Коли користувач здійснює пошук по категорії, до прикладу “Продукти харчування і напої”, він очікує побачити продукти харчування і напої, а не лляні килимки для вирощування мікрозелені, даний товар повинен бути у категорії “Дім і Сад”.

До того ж, всі оголошення у даному сервісі орієнтовані лише продаж і неможливо створити оголошення на купівлю товару (рис. 1.6), тобто створити оголошення типу “Куплю”.

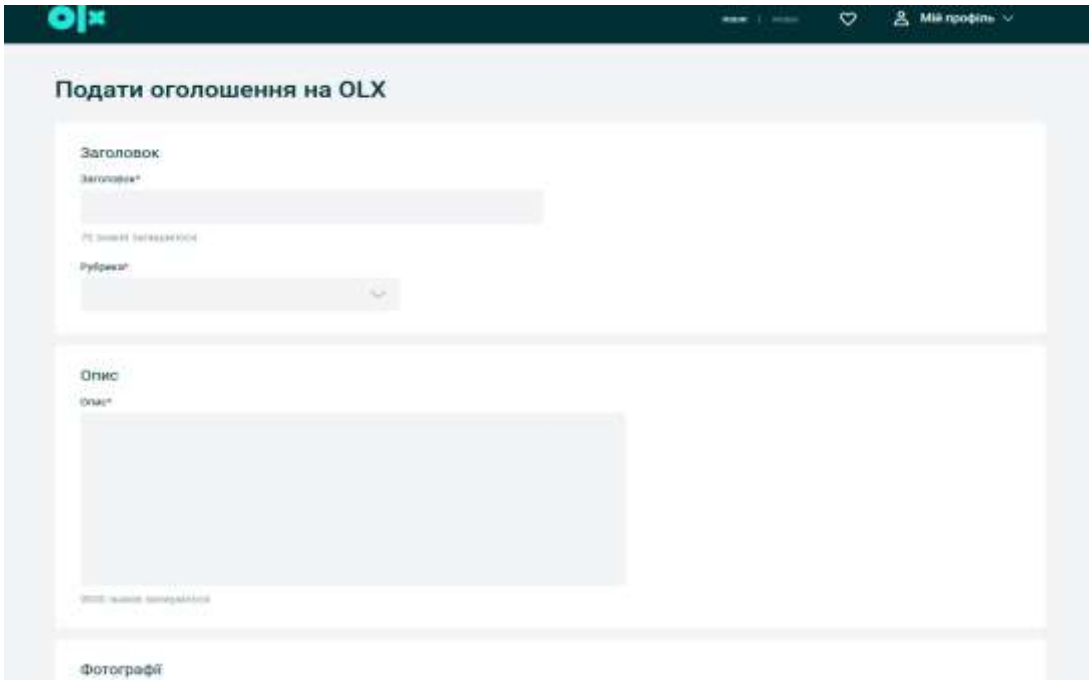

Рисунок 1.6 - Створення оголошення

Як висновок, даний сервіс недостатньо пристосований для реалізації сільського господарства і повного покриття ринку оголошень.

Даний сервіс може створити кращі умови для сільського виправивши дефекти із фільтруванням товарів по категоріях. І розділенням оголошень на типи “Куплю” і “Продам”, це спростить фільтрування оголошень користувачам, і фермерам у тому числі. Сервіс “Бесплатка” [5] теж досить популярний сервіс оголошень. Порівняно з OLX даний сервіс має значно більше категорій на головній сторінці (рис. 1.7).

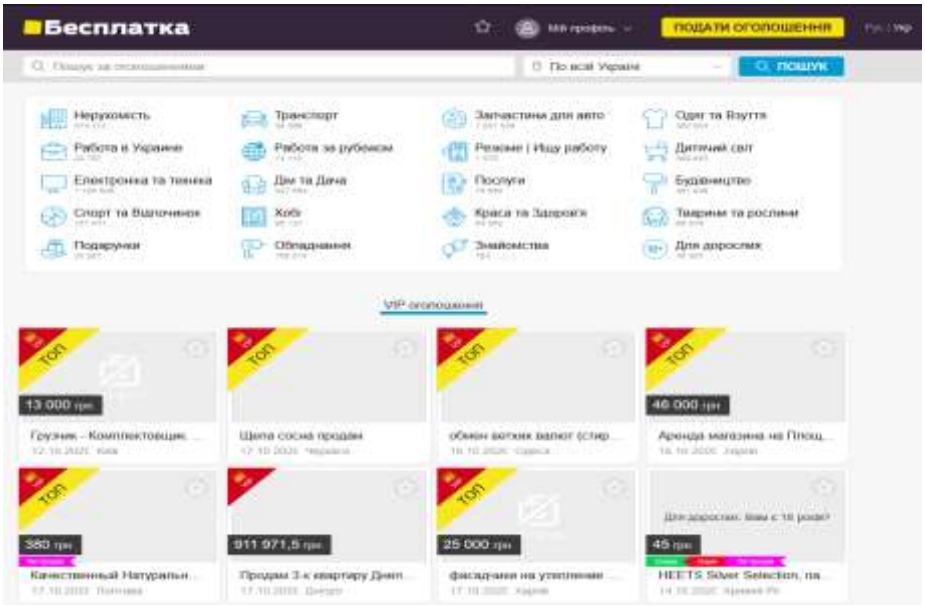

Рисунок 1.7 - Головна сторінка сервісу “Бесплатка”

Сервіс значно краще реалізований для збуту товарів сільського господарства. Але у даному сервісі, так само, як і у сервісі “OLX”, відсутнє розподілення оголошень на “Куплю” і “Продам” типи. Категорія “Куплю” присутнє і стосується лише нерухомості (рис. 1.8).

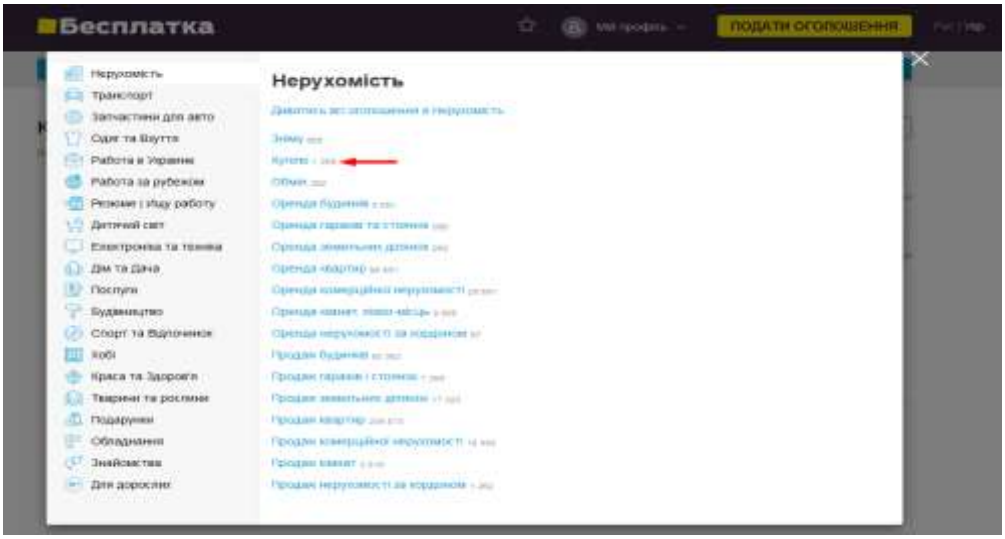

Рисунок 1.8 - Категорія “Куплю”

Веб сервіс Ринок - це стартап проект [6], який лише набирає популярність, і орієнтований лише на сільське господарство, який має майже все необхідне для реалізації сільськогосподарської продукції (рис. 1.9).

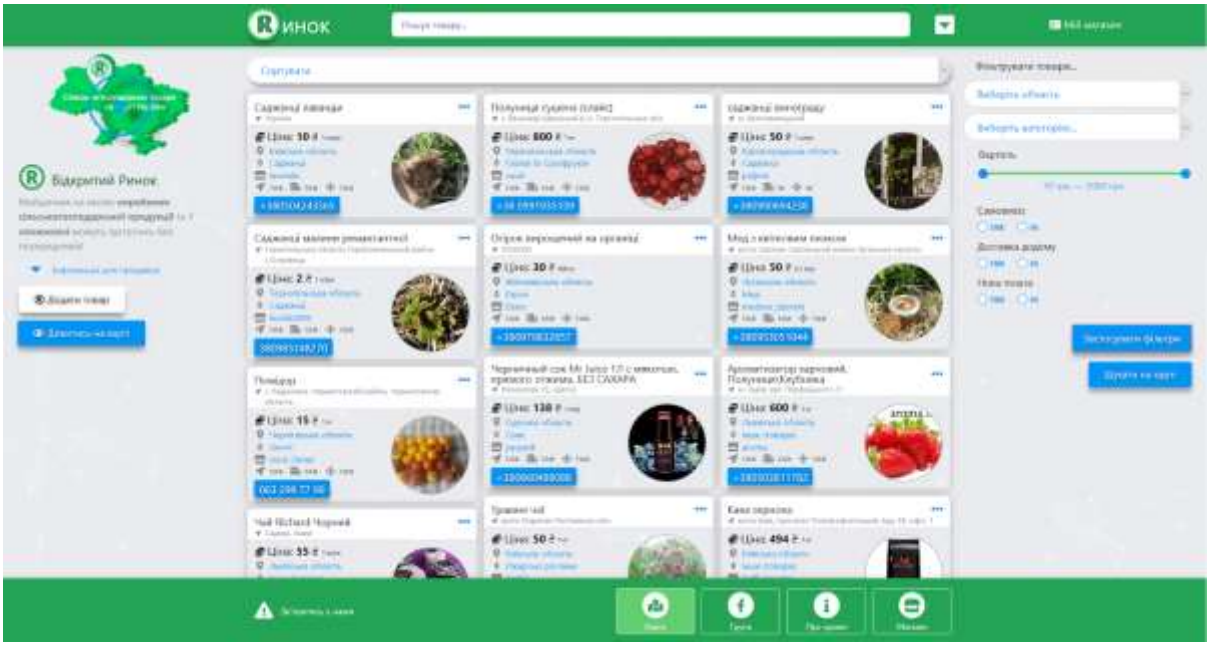

Рисунок 1.9 – Голосна сторінка сервісу “Ринок”

Даний сервіс, як “БЕСПЛАТКА” і “OLX” орієнтовані більше на продаж і відсутня категорія “Куплю” яка, на мою думку є однією із ключових.

Також присутні багато візуальних дефектів, які частково впливають відношення до сервісу, деякі з них можуть спантеличити користувача.

Після декількох спроб зареєструватися у даному сервісі, щоб краще дослідити переваги і недоліки, зареєструватися не вдалося (рис. 1.10 і 1.11).

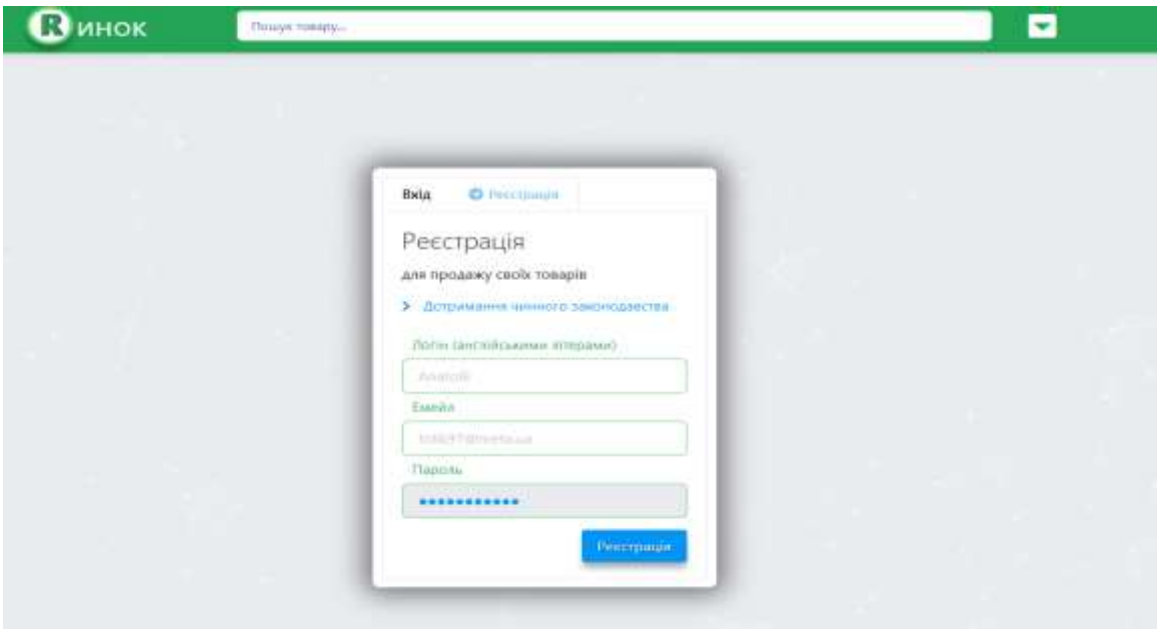

Рисунок 1.10 - Реєстрація у сервісі “Ринок”

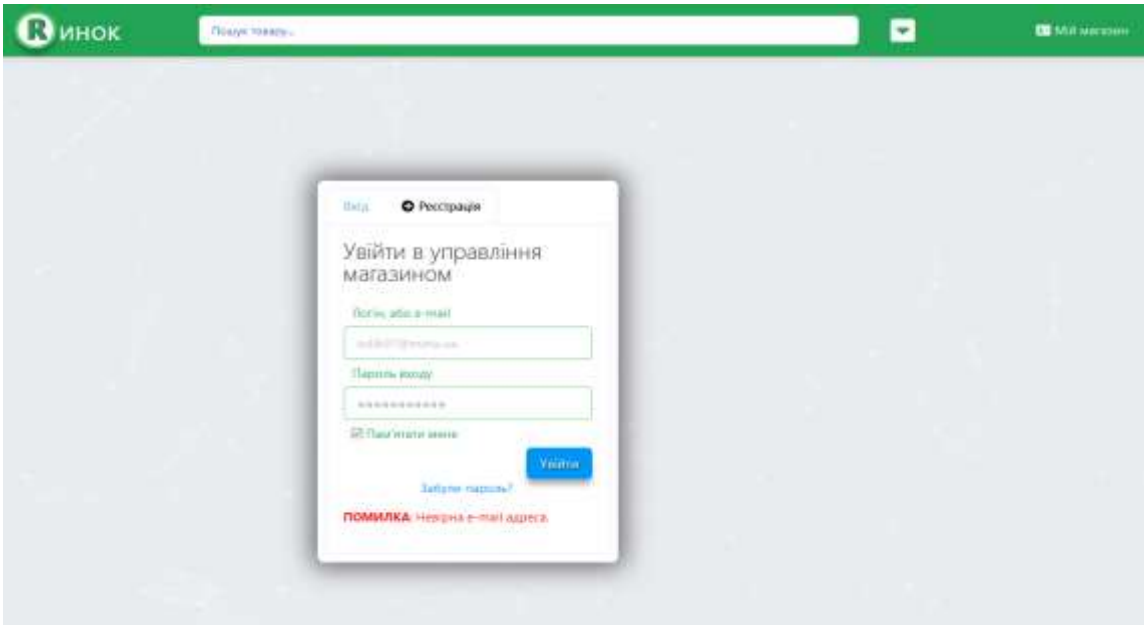

Рисунок 1.11 - Спроба увійти у сервіс після реєстрації

Досить банальна помилка, але якщо сервісом вже користуються люди, такі помилки просто недопустимі. Ще слід зауважити, що користуватися сервісом досить незручно через те що елементи головної веб-сторінки, яку можна відкрити без реєстрації у сервісі, загружаються досить довго.

Дослідивши найпопулярніші сервіси оголошень які існують сьогодні, можна стверджувати, що продаж і реалізація сільськогосподарської продукції - це досить складний процес, який потребує багато часу і сервіси, які сьогодні існують не достатньо пристосовані для такої цілі. В будь якому випадку, це чудово, що розвиток не стоїть на місці, шукаються шляхи і нові методи вирішення проблем які присутні сьогодні у будь якій сфері.

1.2 Технічні вимоги до рішення

1.2.1 Найменування та область застосування

Темою кваліфікаційної роботи “Інформаційна система для вирішення проблем із збуту сільськогосподарської продукції” тобто створення рішення, де люди, ферми і організації які займаються реалізацією і торгівлею

сільськогосподарською продукції зможуть з комфортом і без лишніх витрат її здійснювати.

Впровадження проекту дозволить:

- зменшити час, що витрачається на пошук клієнтів і покупців товару;
- зменшити час на створення оголошень у сервісах, які недостатньо пристосовані під це;
- розділити реалізацію сільськогосподарської продукції на “Куплю” і “Продам” категорії, що значно спростить процес торгівлі і покриє весь ринок оголошень;
- пришвидшить процес реалізації продукції
- підбір оголошень рекомендацій які найкраще підходять під оголошення користувача
- можливість фільтрувати оголошення.

1.2.2 Призначення розробки

Розробка даної веб-платформи ведеться з метою створити конкурентоспроможну систему, яка дозволяється покрити весь ринок оголошень на сьогоднішній день і спростити реалізацію, і торгівлю сільськогосподарської продукції фермам, людям і організаціям.

1.2.3 Вимоги до програмного забезпечення

Часові обмеження

Часові обмеження на роботу програми мають бути описані, виходячи з характеру задачі, що розв’язується. Наприклад, вимоги до часових характеристик в інформаційно-пошукових системах можуть становити до 10 сек. на відповідь на запит; в системах керування виробничим процесом - до 1 сек., а в програмах, що експлуатуються раз на місяць, жорсткі вимоги можуть не ставитися.

Вимоги до надійності

З метою забезпечення надійного функціонування програми повинні бути передбачені:

- ієрархічна структура комплексу;
- блоки, які контролюють усунення збійних ситуацій в роботі системи (наприклад блок контролю наявності необхідних баз даних);
- контроль вводу даних, який забезпечує стійкість програми до помилок користувача (наприклад, контроль по діапазону значень даних). Зверніть увагу, що повинні бути вказані конкретні методи контролю щодо даних, які обробляються;
- можливості, які забезпечують збереження та використання даних під час оновлення роботи після аварійного переривання;
- обробка виняткових ситуацій (наприклад, не знайдено запис);
- захист від несанкціонованого доступу до інформаційної бази (введення пароля, наприклад , під час видалення запису);
- видача чітких та корисних повідомлень про місцезнаходження помилки та її характер.

Умови експлуатації

У цьому підрозділі необхідно описати експлуатаційні характеристики програмного комплексу:

- перелік функціональних можливостей комплексу та різновиди їх реалізації;
- режим взаємодії користувача з програмним комплексом;
- режими роботи програми;
- можливість видачі вихідної інформації за бажанням користувача як на екран, так і на друкуючий пристрій;
- вимоги до складу та параметрів технічних засобів для ефективної роботи програми;

- вимоги до інформаційної та програмної сумісності (тип , версія ОС, необхідне програмне середовище).

1.2.4 Вимоги до програмної документації

В результаті розробки проекту необхідно сформувати пакет документації, який повинен включати як онлайн документацію або файл допомоги та паперову документацію.

Вимоги до програмної документації:

- кожний програмний модуль або процедурний блок повинен мати початковий блок коментарів, який містить призначення, склад вхідних та вихідних даних, обмеження та умови використання, дату введення в дію;
- коментарі повинні бути стислими, чіткими;
- кожний модуль повинен мати одну точку входу та одну точку виходу.

1.2.5 Специфікація вимог до програмного продукту

Для візуального зображення можливостей програмного забезпечення використовується діаграми варіантів використання системи (рис. 1.12).

Рисунок 1.12 - Діаграма варіантів використання

Опис варіантів використання поданий у таблицях 1.1 - 1.5.

Функція “Реєстрація в системі” вимагає від користувача системи пройти процедуру реєстрації на сайті, заповнивши поля реєстраційної форми. Процес реєстрації включає в себе заповнення обов’язкових полів форми, таких як: “Ім’я”, “Прізвище”,”Логін”, “Область”,”Місто/Село”, “Мобільний номер”, “E-mail” і “Пароль”.

Дана функція є однією із найвідповідальніших, оскільки користувачу слід вірно ввести коректні дані про себе чи про свою компанію. Дані згодом будуть використовуватися системою для зображення відповідної інформації в оголошеннях користувача. Це дозволить іншим користувачам контактувати з даним користувачем для подальших операцій і можливо майбутньої співпраці за межами системи.

Таблиця 1.1 - Варіант використання “Реєстрація в системі”

Контекст використання	**Реєстрація в системі**
Дійові особи	Користувач
Передумова	-
Тригер	Відкрита сторінка реєстрації в системі
Сценарій	1. Перехід на сторінку реєстрації 2. Заповнення полів форми 3. Підтвердження
Пост-умова	-

Графічний образ сторінки з формою реєстрації зображений на рисунку 1.13. Зіркою (*) позначено обов'язкові поля.

Рисунок 1.13 – Графічний образ реєстраційної форми

Функція “Авторизація у системі”. Для того щоб зайти у свій аккаунт в сервісі, користувачу слід заповнити форму авторизації, а саме два її поля: “Логін/Email”, “Пароль”. Після цього авторизований користувач буде мати доступ до функцій: “Перегляд оголошень”, “Створення нового оголошення”, “Редагування власних оголошень”.

На сторінці авторизації знаходиться форма введення авторизаційних даних, яка складається з двох полів, ім’я користувача або email, пароль і кнопка підтвердження.

Таблиця 1.2 - Варіант використання “Авторизація користувача”

Контекст використання	**Авторизація в системі**
Дійові особи	Користувач
Передумова	Реєстрація в системі
Тригер	Відкрита сторінка авторизації
Сценарій	1. Перехід на сторінку авторизації 2. Заповнення полів форми

Продовження таблиці 1.2

	3. Підтвердження
Пост-умова	Авторизований користувач

Графічний образ сторінки з формою авторизації зображений на рисунку 1.14.

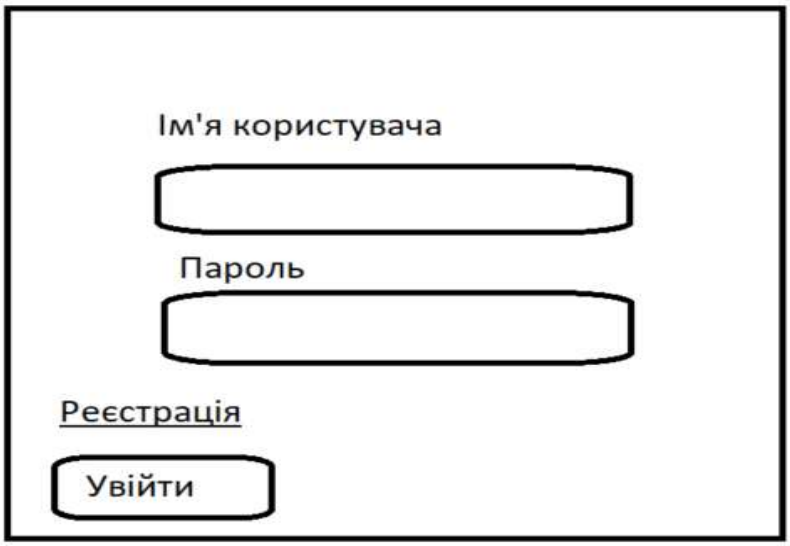

Рисунок 1.14 – Графічний образ сторінки авторизації

Функція “Перегляд оголошень” дає можливість користувачу переглядати перелік оголошень в системі. На сторінці буде розміщено список оголошень, і фільтри для зручної взаємодії із списком.

Таблиця 1.3 - Варіант використання “Перегляд оголошень”

Контекст використання	**Перегляд оголошень**
Дійові особи	Користувач
Передумова	Авторизація в системі
Тригер	Відкрита головна сторінка веб платформи

Продовження таблиці 1.3

Сценарій	1. Перехід на головну сторінку веб платформи 2. Можливість фільтрувати список оголошень
Пост-умова	-

Графічний образ сторінки зі списком оголошень зображено на рисунку 1.15.

Рисунок 1.15 – Графічний образ сторінки зі списком оголошень

Функція “Створення нового оголошення” дає можливість користувачу створити своє оголошення у системі, з допомогою яких спростить свою реалізацію сільськогосподарської продукції.

Таблиця 1.4 - Варіант використання “Створення нового оголошення”

Контекст використання	**Створення нового оголошення**
Дійові особи	Користувач
Передумова	Авторизація в системі

Продовження таблиці 1.4

Тригер	Відкрита сторінка створення нового оголошення у системі
	1. Можливість описати товар який користувач хоче реалізувати, тобто продати вказавши тип оголошення як “Продам” 2. Можливість описати товар який цікавить користувача, тобто купити вказавши тип оголошення як “Куплю” 3. Можливість вказати детальний опис товару 4. Можливість вказати ціну товару 5. Можливість вказати кількість товару 6. Можливість вказати до якої категорії товарів відноситься товар 7. Можливість завантажити фото товару
Пост-умова	-

Графічний образ сторінки створення нового оголошення зображено на рисунку 1.16.

Рисунок 1.16 – Графічний образ сторінки створення нового оголошення

Функція “Редагувати власні оголошення” дає можливість користувачу редагувати створені ним оголошення, оновлюючи інформацію і дані описані в оголошенні про товар. Дана функція дозволить користувачу підтримувати актуальність оголошення і оновляти, до прикладу, кількість товару, що доступна до продажу, або навпаки, з`явилася потреба у більшій кількості товару.

Таблиця 1.5 - Варіант використання “Редагування власного оголошення”

Контекст використання	**Редагування власного оголошення**
Дійові особи	Користувач
Передумова	Створене оголошення
Тригер	Перехід на сторінку для редагування створеного користувачем оголошення
Сценарій	1. Можливість змінити опис товару в оголошенні 2. Можливість змінити ціну товару описаного в оголошенні 3. Можливість змінити тип оголошення 4. Можливість змінити категорію оголошення 5. Можливість змінити назву оголошення 6. Можливість змінити кількість товару
Пост-умова	-

Графічний образ редагування оголошення зображено на рисунку 1.17

Рисунок 1.17 – Графічний образ сторінки редагування оголошення

1.3 Висновок до першого розділу

У даному розділі було проаналізовано предметну область для продукту. досліджено структура на напрями діяльності користувача у системі. Також були проаналізовані усі бізнес процеси які відбуваються у даній системі користувачем. Для кращого розуміння розроблюваного продукту було порівняно три сервіси які надають можливість створювати оголошення: “OLX”, “Бесплатка” і “Вільний ринок”. Після порівняння функціональності систем було визначено і проаналізовано всі переваги і недоліки систем, які потрібно врахувати в процесі розробки веб-платформи “Market Place”. Було детально проаналізовано і описано варіанти використання системи, стадії та етапи розробки і порядок контролю та прийому.

2 ПРОЕКТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ

Для проектування і розробки веб-платформи перш за все, слід обрати яка архітектура найкраще підійде в даному випадку, проаналізувавши найбільш популярні, і які стосуються саме цього напрямку роботи.

2.1 Огляд і аналіз архітектурних шаблонів

Архітектурних шаблонів [30] на сьогоднішній день існує десятки і сотні, кожен з яких має свої переваги і недоліки, маючи спільні риси з іншими. Але найпопулярнішими є:

- мікросервісна архітектура;
- клієнт-серверна архітектура;
- модель-вид-контролер архітектура.

Перша з них, мікросервісна архітектура [31], ідея якої є розділення всієї системи на окремі, самостійні сервіси, які можуть незалежно виконувати свою функцію і при потребі, спілкується з іншими сервісами. Кількість сервісів може сильно коливатися в залежності від програмного продукту який розробляється і його вимог, до прикладу це може бути від два чи три сервісу, до десятків або можливо і сотень сервісів.

До переваг даної архітектури відносяться:

- досить високий рівень незалежності від інших сервісів, кожен сервіс може бути розгорнутим на окремому фізичному сервері;
- незалежна розробка кожного сервісу, дозволяється працювати лише над індивідуальною функціональністю одного сервісу або ж мережі декількох сервісів, в залежності від продукту;
- спрощення заміни реалізації певних функцій чи алгоритмів у сервісах;
- стійкість цілої системи при виході з ладу якогось із сервісів;

- ефективне використання ресурсів, і при необхідності деякі сервіси можуть мати свою базу даних;
- значно менше коду потрібно для реалізації порівняно із монолітом, що дозволяє швидше залучати нових розробників до команди;
- можливість реалізувати кожен окремий сервіс зовсім іншими технологіями і мовами програмування, що є дуже гнучко.

Але як і кожна архітектура, дана не є виключенням щодо недоліків:

- найбільше уваги приділяється реалізації інфаструктури сервісів, забезпечення сервісів, і в залежності від кількості сервісів, масштабу продукту, команді потрібні розробники які відповідають за налаштування серверів і мереж між ними;
- зростають ризики конфліктів між розробниками через різні погляди , думки, бажання і досвід;
- зростає кількість дубльованого коду, який може бути майже у кожному сервісі, що є не дуже ефективним, особливо коли будуть якісь дефекти;
- зростає складність тестування і відкладки сервісів і взаємозв`язків між ними;
- також ускладнюється забезпечення безпеки, тому що кожний сервіс може знаходитись на іншому фізичному сервісі і мережі, як результат ускладнюється робота команди і збільшується час на забезпечення безпеки;
- не завжди можна забезпечити постійний і стабільний зв`язок між сервісами і фізичними серверами.

Модель-вид-контролер архітектура [32] більш популярна у технологіях .NET. Де є ASP.NET MVC і ASP.NET Core MVC. У обох багато спільного, в тому числі і архітектурна ідея, яка полягає у тому, що розділити програмний продукт на три частини:

- модель, що відповідає за структуру, збереження, обробку і операції над даними;

- вид, це графічна частина програмного продукту, що може бути веб інтерфейсом або ж інтерфейсом десктопних програм, основна ідея полягає у представленні даних користувачу;
- контролер, елемент програми, що керує результатами обчислень моделі над даними і представленню цих даних користувачу використовуючи вид.

Переваги даної архітектури:

- при правильному підході і реалізації, можна перевикористати модель для декількох видів, що значно спрощує роботу на програмним продуктом;
- значно спрощує підтримку продукту у майбутньому;
- новим членам команди простіше зорієнтуватися у продукті;
- можливе перевикористання видів, у різних типів графічних інтерфейсів;
- значно простіше розробникам робити відкладку коду у пошуках дефектів;
- в цілому, єдине концепція системи;

З недоліків це:

- потрібність у достатній кількості ресурсів, для забезпечення стабільної роботи, оскільки системи досить складна і містить практично три незалежні елементи, і вся взаємодія між ними відбувається через контролер, який повинен все контролювати, генерувати, будувати, регулювати, валідувати всі дані і операції щодо виду і моделі;
- ускладнений процес додання нової функціональності у систему, оскільки при доданні слід також реалізувати дану функціональність у всіх трьох частинах системи, доприкладу, якщо додається якась таблиця, це повинний бути і вид, де користувач зможе побачити дані, контролер, який буде все валідувати і контролювати, також і модель, яка буде всі дані калькулювати при потребі і зберігати у базі даних.

Ідея клієнт-серверної архітектури полягає у тому, щоб розділити програмний продукт або програмний комплекс на умовні дві частини які пов`язані між собою третьою частиною, роль якої в основному відіграє мережа.

Архітектура клієнт-сервер передбачає основні елементи:

- сервер, або сервери, що обробляють отримані дані і після певних маніпуляцій і їх обробки видають інформацію;
- клієнти, користувачі, що звертаються до сервера, в залежності від запиту можуть отримувати результат із сервера;
- мережа, яка забезпечує обмін даними між клієнтом та серверами.

Збереження та обробка даних здійснюється на стороні сервера, відображення і надсилання запитів до сервера виконується на боці клієнта.

На рисунку 2.1 зображено трирівнева схема архітектури веб-сервісу.

Рисунок 2.1 - Схема клієнт-серверної архітектури

Як висновок, можна сказати, що аналізувати архітектури, враховувати, передбачувати і перш за все знати всі переваги і недоліки дуже корисно, і пригодиться не лише реалізації програмного продукту “Market Place”, але і у розробках в майбутньому.

2.2 Опис і обґрунтування вибору технологій

Оскільки буде розроблятися веб-платформа, буде доцільно використати найпопулярніших у даній сфері технології , а саме ASP.NET Core MVC з версією .NET Core 3.1. ASP.NET Core - це технологія від Microsoft, призначена для створення різноманітних веб-додатків, від невеликих веб-сайтів до великих веб-порталів та веб-служб. Фреймворк є повністю відкритий. Усі вихідні файли для фреймворку доступні на GitHub [8]. ASP.NET Core може працювати поверх кроссплатформенної платформи .NET Core, яка може бути розгорнута в основних популярних операційних системах: Windows, Mac OS[15], Linux. Тобто серверна частина “Market Place” може бути розташована на сервері з будь якою операційному системою.

Завдяки модульності фреймворку всі необхідні компоненти веб-програми можна завантажити як окремі модулі через менеджер пакетів Nuget. ASP.NET Core включає структуру MVC, яка інтегрує функціональність MVC, веб-API та веб-сторінки. У попередніх версіях платформи ці технології реалізовувались окремо і тому містили багато дублікатів функціональних можливостей. Тепер вони об’єднані в одну модель програмування ASP.NET Core MVC. На додаток до поєднання вищезазначених технологій в одну модель, до MVC було додано ряд додаткових функцій. Однією з таких особливостей є помічник тегів, який дозволяє більш легко поєднувати синтаксис html з кодом C #.

ASP.NET Core є розширюваним. Фреймворк побудований з набору відносно незалежних компонентів. І ми можемо або використовувати вбудовану реалізацію цих компонентів, або розширити їх, використовуючи механізм успадкування, або навіть створити та використовувати наші власні компоненти зі своєю функціональністю.

Переваги вибору даної технології наступні:

- Новий легкий та модульний конвеєр запитів HTTP;
- Використання платформи .NET Core та її функціональних можливостей;

- Поширення пакетів платформ через NuGet;
- Вбудована підтримка створення та споживання пакетів NuGet;
- Єдиний стек веб-розробки, що поєднує веб-інтерфейс та веб-API;
- Конфігурація для спрощеного використання в хмарних сервісах;
- Вбудована підтримка введення залежностей;
- Розширюваність;
- Крос-платформа: можливість розробляти та розгортати програми ASP.NET на Windows, Mac та Linux.

Для роботи із базою даних було обрано Entity Framework Core, оскільки використовуючи саме цю технологію, можна створити мережу об’єктів (сутностей) з усіма зв'язками які потрібні, EF на основі цих сутностей створить базу даних у MS SQL.

2.3 Опис системи контролю версій

Основною частиною процесу розробки програмного продукту є система контролю версіями.

Система контролю версіями - це програмний інструмент, система, що записує всі зміни які відбуваються із файлами протягом часу, це можуть бути одиниці інформації, код і скрипти програмних продуктів різних масштабів, веб сайтів, веб-платформ, текстових документів, тощо.

Дозволяє зберігати попередні версії файлів і завантажувати їх при потребі. Вона зберігає повну інформацію про файл і всі його версії, а також повну структуру проекту на всіх стадіях розробки.

Версійність програмного продукту важливий елемент процесу розробки, дозволяє стабільно і поступово розробляти продукт і публікувати його користувачам у використання.

Найпопулярніші системи контролю версій зображено у таблиці 2.1

Таблиця 2.1 - Топ найпопулярніших систем контролю версій

Назва	Рік заснування
GIT [18]	2005
GITHUB [17]	2008
SVN (Subversion) [19]	2000
Directual [20]	2016
Mercurial [21]	2005
Team Foundation Server [22]	2005

Місце зберігання даних файлів називають репозиторієм. Всередині кожного з репозиторіїв можуть бути створені паралельні лінії розробки — гілки. Для реалізації “Market Place” було обрано GITHUB. Дана система дозволяє створювати публічні і приватні репозиторії. Вона базується на системі GIT. Сервіс безкоштовний для проектів з відкритим кодом.

Дана система була обрана через те, що є максимально простою у використанні, і дозволяє легко ділитись і поширювати код. Зробивши репозиторій публічним, всі охочі можуть глянути реалізацію проекту, і провести аналіз коду на всім дефектів, і вдосконалення певних процесів у коді.

2.4 Аналіз і обґрунтування вибору середовища програмування

Одним із важливих моментів процесу розробки програмного забезпечення є вибір середовища програмування, тобто IDE.

IDE (Integrated Development Environment) – інтегроване середовище програмування. Є різні види середовища програмування, одні більш простіші, такі як текстовий редактор із додатковою функціональністю для розробки, підсвітка синтаксису різноманітних мов програмування, інтерфейс який можна налаштувати під себе, і розширені можливості навігації по коду. Але такий тип

не для свій мов підійде, і другий тип середовищ програмування, теж містять текстовий редактор, але разом із цим, ще мають багату функціональність для розробки, компілятор, функції відкладки коду, автоматизація певних процесів, тестування, візуальні редактори для десктопних рішень.

Оскільки веб-платформа буде розроблятися .NET технологіями тому доцільно буде проаналізувати саме ті середовища програмування які найбільше орієнтовані на роботу саме із цими технологіями.

Однією із найпопулярніших є “Microsoft Visual Studio” [16], логотип зображено на рисунку 2.2. Середовище програмування від компанії Microsoft, всі версії якої можуть створювати програмні продукти різних типів, веб-сервіси, десктопні програми, мобільні додатки ігри, та інші. Дане IDE є чудовим рішенням і для студента, і для досвідченого розробника.

Мови та технології які підтримуються: Ajax, ASP.NET, DHTML, JavaScript, JScript, Visual Basic, Visual C#, C++, Visual F#, XAML та інші.

Рисунок 2.2 – Логотип Microsoft Visual Studio

“Microsoft Visual Studio” має багато особливостей:

- IntelliSense – це технологія автодоповнення коду, дозволяє швидко писати код пропонуючи розробнику перелік методів які найкраще підходять у поточній стрічці коду (рис. 2.3), пропонує найкращі варіанти назвати змінні та інші елементи, пропонує розробнику вирішення проблем із назвами методів, змінних тощо, дозволяти одночасно редагувати змінні у декількох місцях;

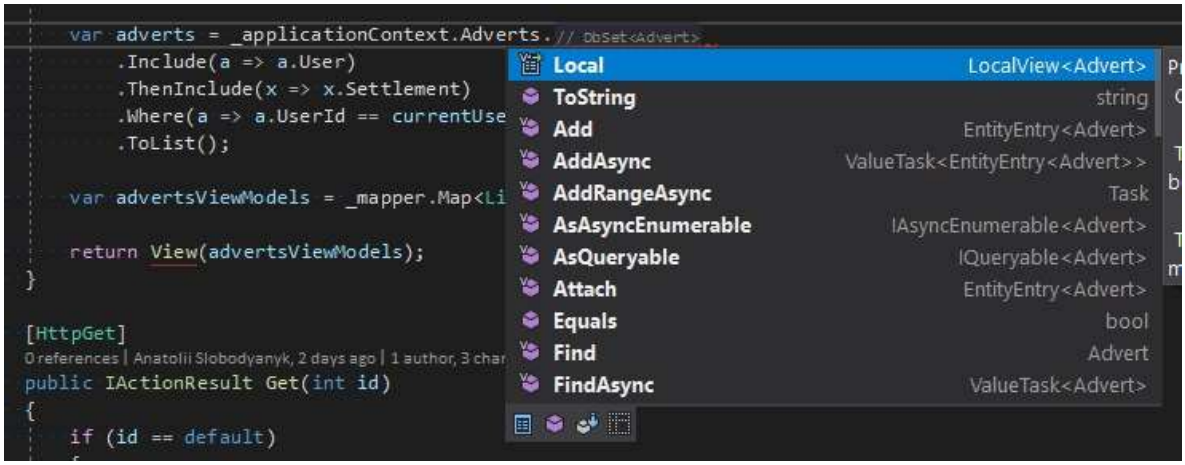

Рисунок 2.3 – Приклад роботи IntelliSense

- Величезна кількість бібліотек і доповнень до середовища, які так чи інакше спрощують процес розробки програмного забезпечення, користуючись вбудованим менеджером доповнень це робити значно простіше, також сайт “Visual Studio Marketplace” [28] із доповненнями з їх детальними описами і залежностями;
- Можливість швидко налаштувати панель інструментів і саме середовище в цілому;
- Вбудований аналізатор продуктивності програмного продукту із можливістю відкладки;
- Можливість роботи у різних вікнах, тобто можна окремі файли відкрити як окремі вікна, просто взявши вікно редатора файлу і перетягувши його за межі вікна середовища, згодом окремі вікна при потребі можна об`єднювати в одне окреме від IDE і працювати на декількох моніторах;
- Вбудований Microsoft SQL Server який дозволяє працювати із базами даних без додаткових програм;
- Вбудований IIS Express, технологія, що дозволяє розгортати веб-серверні програмні продукти з допомогою IIS Http Server, і що в свою чергу дозволяє повноцінно користуватися веб-сервером і тестувати локально на комп`ютері;

- Існує версія Visual Studio Community, яка є безкоштовна, і для студентів і для професійного використання, але з трішки меншим списком інструментів.

Visual Studio найбільш орієнтована на операційну систему Windows, хоча існує ще версія під MacOS,але там обмежений список інструментів, і для розробки програмного забезпечення має використовуватися лише .NET Core, багатоплатформенна технологія від компанії Microsoft для розробки ПЗ.

Але дана IDE має досить серйозний недолік, це потреба у великій кількості ресурсів, тобто для роботи із даним середовищем слід мати досить потужний комп`ютер чи ноутбук.

Ще одною популярною для .NET технологій є багатоплатформенна IDE “Rider” [29] від компанії “JetBrains”, яка досить швидко розвивається і намагається скласти конкуренцію Visual Studio. Логотип “Rider” зображений на рисунку 2.4.

Рисунок 2.4 – Логотип середовища програмування “Rider”

З особливостей “Rider” є:

- Вбудована технологія “Resharper” яка є аналогом IntelliSense з Visual Studio, так ж слід зауважити , що “Resharper” також поширюється окремо від “Rider” як платне доповнення до Visual Studio із безкоштовним 30-ти денним випробувальним періодом;
- Теж присутня можливість налаштовування панелі інструментів;

- Більш широка можливість налаштувати інтерфейс середовища в цілому, можна додати картинки на фон коду і середовища, тощо. Що в певній мірі є перевагою на Visual Studio;
- Можливість студенту отримати безкоштовну ліцензію із використанням середовища програмування не в комерційних цілях, що дозволяє урізноманітнити вибір IDE для початківців.

Серед всіх недоліків, основним даної IDE є те, що розробка можлива лише з використанням .NET Core. Visual Studio у свою чергу може працювати з .NET Framework. Це дуже важливо, беручи до уваги те, що .NET Core появився не так, давно, і багато ПЗ написано і розробляється саме на .NET Framework. Ще одним недоліком є те, що середовище платне для професійного використання.

Із врахуванням всіх переваг і недоліків, як інструментарій для розробки буде використано останній випуск середовища програмування від компанії Microsoft, Visual Studio 2019 тому що це повноцінне середовище програмування і є підтримка MS SQL Server, що дуже важливо.

2.5 Проектування структури бази даних

У розробці бази даних була вибрана реляційна модель даних. Вона найкраще підходить для вирішення задачі, тому що вона має ряд переваг:

- програми не залежні від даних;
- простота розробки та моделювання;
- наявність можливості керування даними з допомогою операцій на множинами даних;
- можливості різних застосувань;
- забезпечення методологічного підходу;
- оптимізація доступу до бази даних;
- забезпечення користувача різними рівнями доступу.

Діаграма бази даних зображена на рисунку 2.5

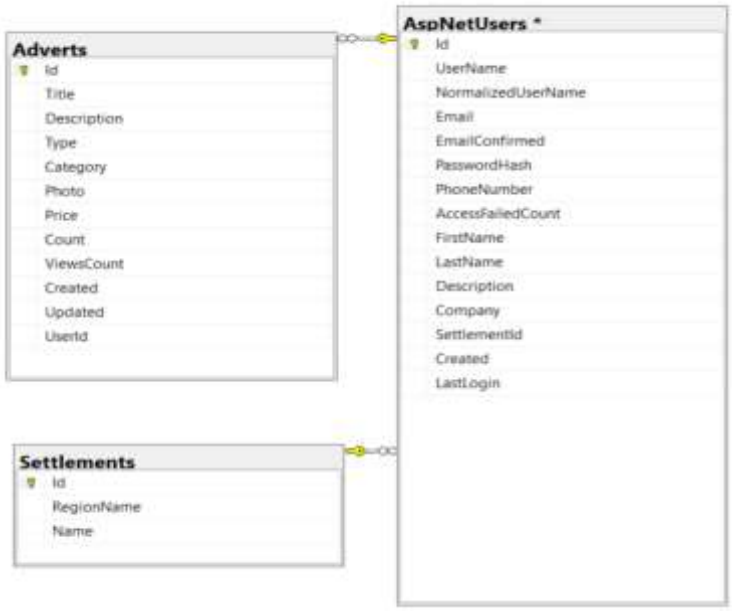

Рисунок 2.5 - Діаграма бази даних

Таблиця AspNetUsers створена для того щоб зберігати дані користувачів і складається із:

1. Id - унікальний ідентифікатор користувача. Тип даних integer.
2. UserName - логін користувача. Тип даних string.
3. NormalizedUserName - нормалізований вид UserName. Додано для того щоб уникнути ситуацій із використанням UserName з різним регістром символів. Тип даних string.
4. Email - email користувача. Тип даних string.
5. NormalizedEmail - нормалізований email користувача.Тип даних string.
6. EmailConfirmed - інформація чи email користувача підтверджений. Тип даних boolean.
7. PasswordHash - хеш пароля користувача. Тип даних string.
8. PhoneNumber - мобільний номер користувача. Тип даних string.
9. AccessFailedCount - кількість невдалих спроб отримати доступ до ресурсу/сторінки. Тип даних integer.
10. FirstName - ім’я користувача. Тип даних string.

11. LastName - прізвище користувача. Тип даних string.
12. Description - опис користувача. Тип даних string.
13. Company - компанія користувача. Тип даних string.
14. SettlementId - зовнішній ключ для реалізації зв’язку між користувачем (User) і населеним пунктом (Settlement). Тип даних integer.
15. Created - дата реєстрації (створення користувача) Тип даних date.
16. LastLogin - дата останнього входу користувача у систему. Тип даних date.

Таблиця Settlements створення для зберігання даних про населені пункти де мешкають користувачі і складається:

1. Id - унікальний ідентифікатор населеного пункту користувача. Тип даних integer.
2. Name - назва населеного пункту користувача. Тип даних string.
3. RegionName - назва області де розташований населений пункт користувача. Тип даних string.

Таблиця Adverts створена для зберігання дані оголошень і складається із:

1. Id - унікальний ідентифікатор населеного пункту користувача. Тип даних integer.
2. Title - назва оголошення. Тип даних string.
3. Description - опис оголошення з деталями. Тип даних string.
4. Type - тип оголошення (“Куплю”, “Продам”). Тип даних integer.
5. Category - категорія товару для якого створюється оголошення. Тип даних integer.
6. Photo - фото товару. Тип даних string.
7. Price - ціна. Тип даних float.
8. Count - кількість товару. Тип даних integer.
9. ViewsCount - кількість переглядів оголошення. Тип даних integer.
10. Created - дата створення. Тип даних date.
11. Updated - дата оновлення інформації оголошення. Тип даних date.

12. UserId - унікальний ідентифікатор користувача. Доданий для реалізації зв'язку між користувачем (User) і оголошенням (Advert). Тип даних integer.

Саме така структура бази даних забезпечить потрібний взаємозв`язок між сутностями у програмі, і дозволить правильно і раціонально організувати архітектуру яка була вибрана.

2.6 Висновок до другого розділу

У даному розділі було розроблено і спроектовано архітектуру веб-додатку яка найкраще підходила для реалізації програмного продукту і оскільки система оперує даними які слід зберігати було ще спроектовано структуру бази даних. Оскільки база даних є реляційному, було створено діаграму зв’язків між таблицями.

Описано і обґрунтовано вибір технологій з допомогою яких буде реалізовано програмний продукт. Також обґрунтовано вибір, налаштування і підключення системи контролю версій. Вибрано і інстальовано середовище програмування з допомогою якого буде розроблятися програмний продукт.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

В процесі розробки веб-платформи “Market Place” будуть використовуватися наступні технології і мови програмування:

C# - це об’єктно-орієнтована мова програмування з допомогою якої можна реалізуватися різні додатки від десктопних програм до веб-порталів і веб-сервісів які обслуговують мільйони користувачів щодня.

ASP.NET Core - це технологія від Microsoft, призначена для створення різноманітних веб-додатків, від невеликих веб-сайтів до великих веб-порталів та веб-служб.

.NET Core - це кроссплатформенна модульна платформа для розробки програмного забезпечення з відкритим кодом.

JavaScript - динамічна і об’єктно-орієнтована прототипна мова програмування. Найчастіше використовується для реалізації взаємодії користувача із веб сторінками, асинхронно обмінюватися даними із сервером, змінювати структуру сторінки і зовнішній вигляд.

Microsoft SQL Server - це система управління базами даних, яка розробляється корпорацією Microsoft. Як сервер даних виконує основну функцію по збереженню та наданню даних у відповідь на запити інших застосунків.

SQL - це декларативна мова програмування для взаємодії користувачі із базами даних, що застосовується для формування запитів, оновлення і керування реляційними базами даних, створення схеми бази даних та їх модифікацій, системи контролю доступу до бази даних [11].

EntityFramework Core - це об’єктно-орієнтована технологія від компанії Microsoft для доступу до даних. Він являється ORM-інструментом (object-relational mapping - відображення даних в якості реальних об’єктів). Тобто дозволяє взаємодіяти з базою даних як об’єктами і абстрагуватися від самої бази даних і таблиць, і працювати з будь якою реляційною базою даних.

CSS (Cascading Style Sheets) - це спеціальна мова стилізації веб сторінок.

HTML (HyperText Markup Language) - мова розмітки веб сторінки, з допомогою якої можна розміщуватися елементи на веб сторінці.

Bootstrap - відкритий набір інструментів для створення сайтів і веб-сервісів, вміщює в собі HTML і CSS шаблони для оформлення різних елементів веб сторінки [12].

jQuery – JavaScript-бібліотека з відкритим кодом. Використовується для швидкого орієнтування і навігації по веб сторінці, створенню анімацій, обробки подій, тощо [13].

AutoMapper – бібліотека яка використовуються для швидкого конвертування класи одного типу у класи іншого типу, присвоюючи значення властивостей першого, властивостям другого. По замовчуванню підставляються властивості з одинаковими назвами, але це можна налаштувати у профільних класах, у яких вказується назви яких властивостей до яких слід присвоювати [14].

3.1. Обґрунтування вибору архітектурного шаблону

Проаналізувавши найпопулярніші архітектурні ідеї і шаблони, врахувавши всі переваги та недоліки, найкращим рішенням буде реалізувати архітектуру програмного продукту “Market Place”, комбінація модель-вид-контролер і клієнт-серверної архітектур буде ідеальним рішенням.

Переваги даного рішення є у тому, що з модель-вид-контролер буде запозичено ідею роздільності проекту на три умовні частини, буде значно спрощено підтримку і відкладку сервісу, сервіс зможе мати не лише веб інтерфейс, як це є у модель-вид-контролер, а й можливість створення окремих клієнтський елементів архітектури, до прикладу мобільних додатків, або десктопне рішення, а це якості клієнт-серверної архітектури.

Графічне представлення архітектури зображено на рисунку 3.1.

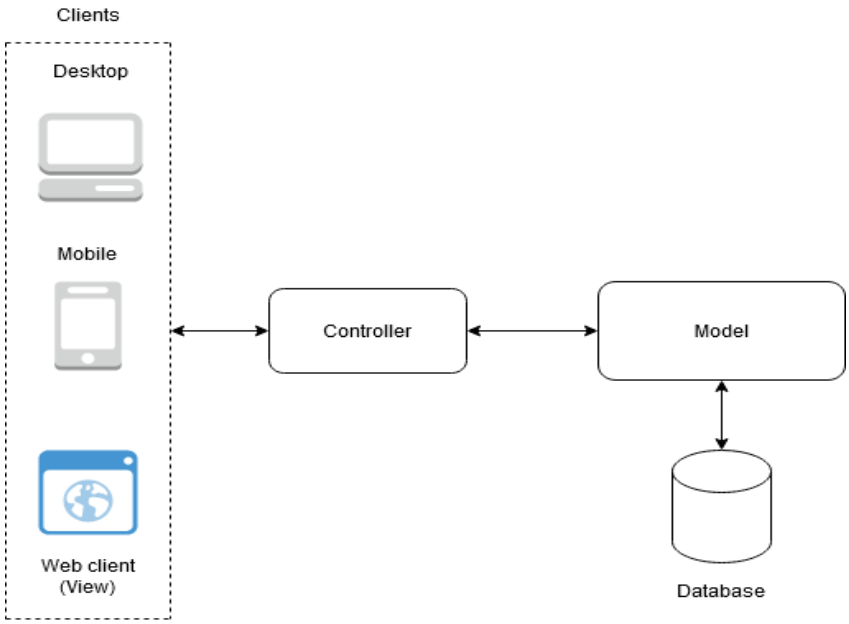

Рисунок 3.1 – Графічне зображення архітектури

Дане рішення дозволить збільшувати кількість клієнтських частин програмного продукту, що в свою чергу дозволить користувачам активніше і частіше користуватися улюбленим сервісом, і розширяти функціональність.

Також слід врахувати, що поки не має потреби у використанні архітектурних рішень, які більш орієнтовані для потужних і масивних веб-платформ, тощо. При правильній реалізації поточної структури, у майбутньому не виникне складнощів, розділити на мікросервіси, чи знайти і реалізувати саме те рішення, яке найкраще підходить у тій ситуації з врахуванням кількості користувачів і кількості запитів у секунду.

3.2 Опис інтерфейсу користувача

Важливість дизайну інтерфейсу користувача [33] складно переоцінити, щоб правильно і ефективно реалізувати веб клієнт, слід дотримуватися певних правил:

- перш за слід врахувати до якої сфери відноситься ідея і проблеми які вирішує програмний продукт, щоб підібрати правильну палітру кольорів, картинок і звісно сам логотип сервісу, який повинен бути унікальним,

користувач дивлячись на нього міг без сумнівів вказати з цим пов`язаний продукт;

- інтерфейс повинен бути максимально простим і зрозумілим для кінцевого користувача;
- зображення на сайті повинне мати коректне розширення, користувач міг розгледіти, що там зображено і залишав хороше враження про сайт.

Тому дотримуючись перелічених правил, і беручи до уваги те, що сервіс орієнтований на сільське господарство, буде доцільно використати у якості фону сервісу зображення із відповідним змістом. Відповідно із логотипом реалізація таж сама.

Логотип сервісу “Market Place” зображено на рисунку 3.2.

Рисунок 3.2 – Логотип сервісу “Market Place”

Реалізація сторінки входу у систему зображена на рисунку 3.3.

Рисунок 3.3 – Сторінка входу у сервіс “Market Place”

3.3 Програмна реалізація проекту

Застосувавши стек технологій які були обрані для реалізації даного продукту і використовуючи вибране середовище програмування створюється проект із назвою “Market Place” (рис. 3.4).

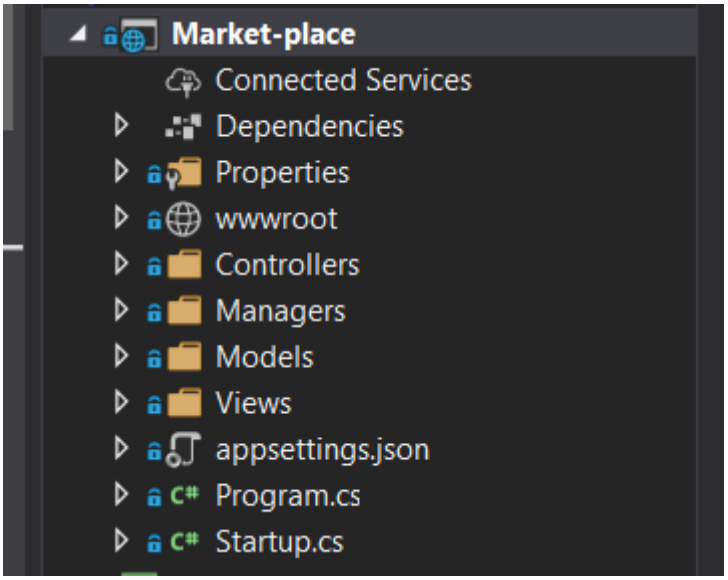

Рисунок 3.4 - Структура проекту “Market Place”

Файли проекту розподілені по директоріям. Properties містить файли у яких прописані налаштування щодо використання IIS Express як сервер. В WWWROOT знаходяться всі JavaScript скрипти, CSS файли зі стилями, картинки і іконки. Controllers містить файли з точками входу, тобто шляхи куди клієнти системи можуть надсилати запити і отримувати відповідь.

Подано код точки яка на запит із ідентифікатором оголошення перенаправить користувача на сторінку із даним оголошенням.

У лістингу 3.1 наведено код метода контролера який повертає дані оголошення по його ідентифікатору.

Лістинг 3.1 – Метод контролера на витяг даних оголошення

```
[HttpGet]
public IActionResult Get(int id)
```

```
{
      if (id == default)
{
      return RedirectToAction("Index", "Home");
}

var advert = _applicationContext.Adverts
      .Include(a => a.User)
      .ThenInclude(x => x.Settlement)
      .FirstOrDefault(a => a.Id == id);

var advertsViewModel = _mapper.Map<AdvertViewModel>(advert);

var adverts = _applicationContext.Adverts
      .Include(a => a.User)
      .ThenInclude(x => x.Settlement)
      .Where(a => a.Type != advert.Type
      && a.Category == advert.Category)
      .OrderBy(x => x.Price)
      .ToList();

var advertsViewModels = _mapper.Map<List<AdvertViewModel>>(adverts);
      advertsViewModel.Recommandations = advertsViewModels;
      return View("Index", advertsViewModel);
}
```

Атрибутом [HttpGet] вказується, що дана точка сервера лише повертає дані, в даному випадку оголошення.

Далі слід перевірити чи ідентифікатор не дорівнює нулю, якщо умова здійсниться то користувача буде перенаправлено на головну сторінку.

Якщо умова не здійсниться, формується запит до бази даних, і використанням технології .NET LINQ [10]. Задача даного запиту знайти у базі даних оголошення з цим ідентифікатором, тим самим знайти користувача який це оголошення створив і підвантажити населений пункт користувача, щоб згодом на сторінці користувачу показати ці дані, але перед цим слід ще

сформувати модель-об’єкт для веб сторінки, тому використовуючи бібліотеку AutoMapper, це можна зробити дуже швидку.

Далі формується запит, задача якого витягти з бази даних оголошення протилежного типу, тобто якщо оголошення по ідентифікатору мало тип “Купити”, то для рекомендацій будуть підгружатись оголошення з типом “Продам”, врахувавши спільну категорію і відфільтрувавши результат по зростаючій ціні, теж формуємо дану колекцію об'єктів у модель-об’єкти для веб сторінки. Як результат, користувач побачить на екрані оголошення із рекомендаціями (рис. 3.5).

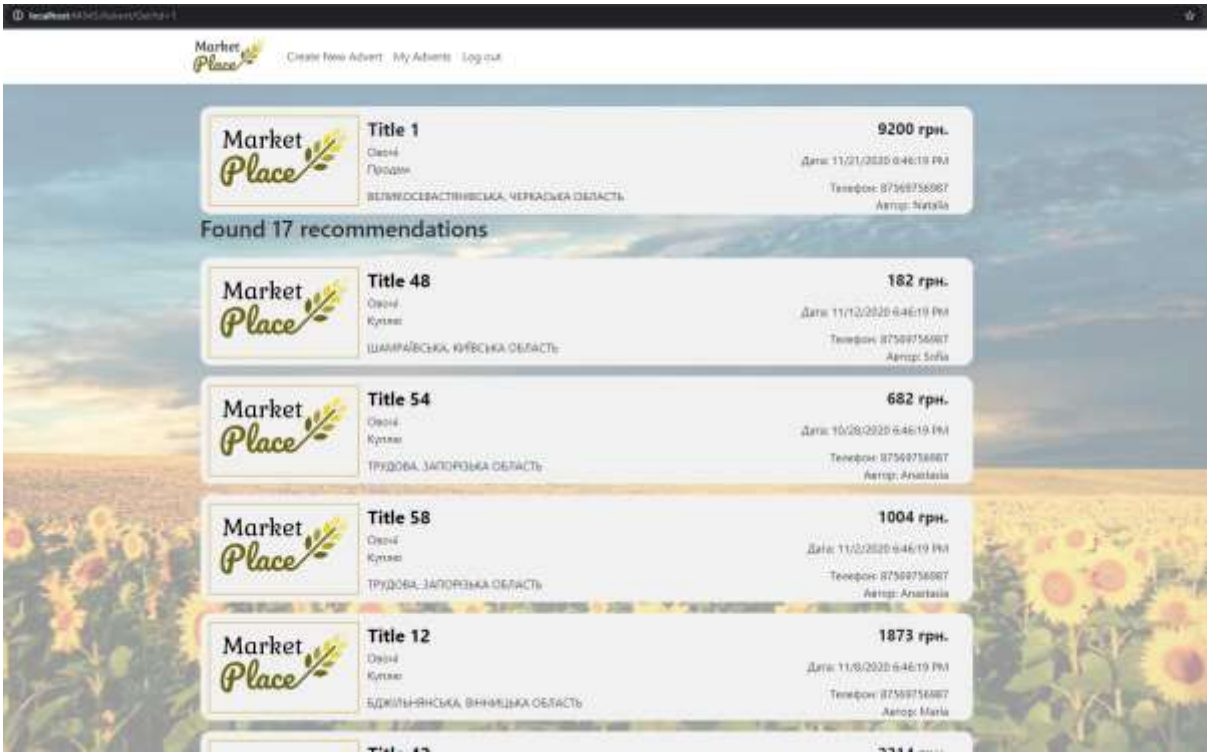

Рисунок 3.5 - Оголошення з рекомендаціями

Для створення нового оголошення буде використовуватися інша точка. Коли користувач натисне кнопку створити після заповнення всі обов’язкових полів (рис. 3.6).

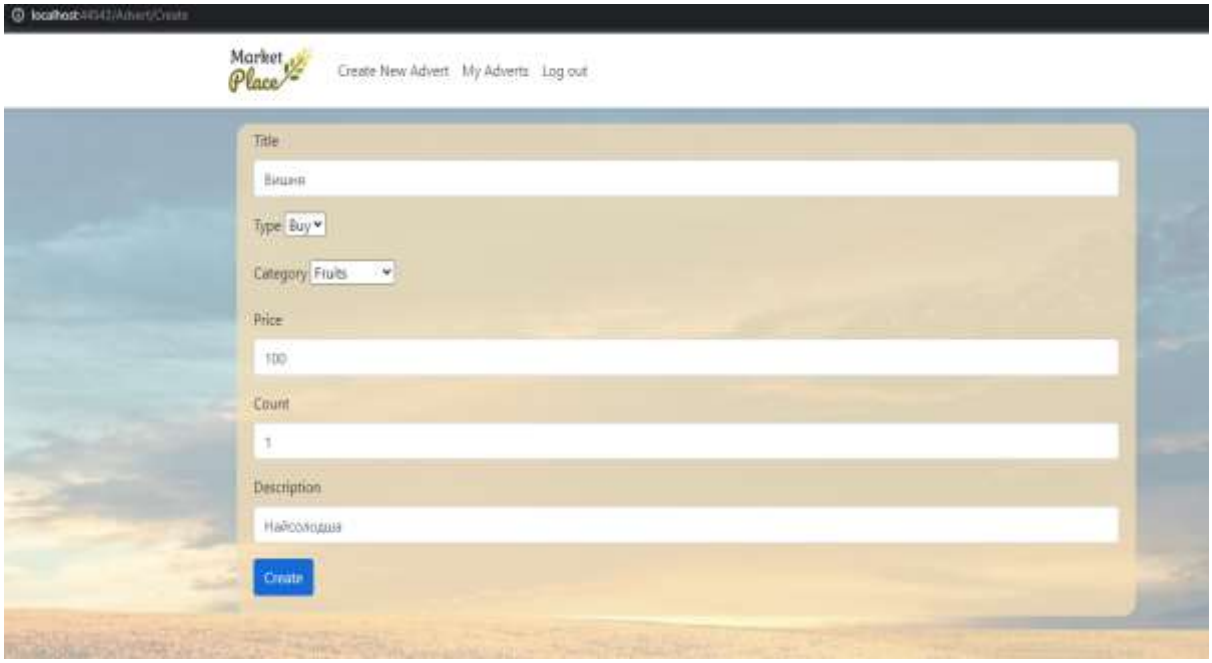

Рисунок 3.6 - Сторінка створення нового оголошення

У лістингу 3.2 наведено код метода контролера на створення нового оголошення.

Лістинг 3.2 – Метод контролера для створення нового оголошення

```
[HttpPost]
public ActionResult Create(CreateAdvertViewModel createAdvertViewModel)
{
var nameIdentifier = User.FindFirst(ClaimTypes.NameIdentifier).Value;
var currentUserId = int.Parse(nameIdentifier);
createAdvertViewModel.UserID = currentUserId;
var advert = _mapper.Map<Advert>(createAdvertViewModel);
var id = _advertManager.Create(advert);

return RedirectToAction("Get", new { id });
}
```

Дана точка отримує модель-об’єкт веб сторінки із заповненими полями (рис 3.7).

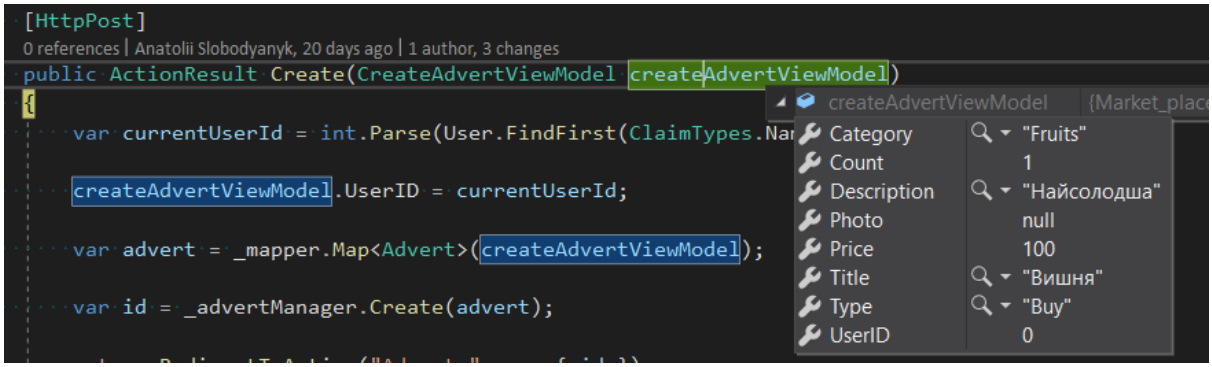

Рисунок 3.7 - Значення полів моделі-об’єкта веб сторінки

Але тут поки що невідомо який користувач створив дане оголошення. Тому у даному методі витягуючи із запиту дані користувача і ідентифікатор теж, ми можемо вказати його в оголошені, що робиться у наступних двох стрічках. Далі модель-об’єкт трансформується у об’єкт-сутність, на основі якої створена таблиця оголошень у базі даних. Після всіх маніпуляцій, об’єкт advert передається методу Create менеджера оголошень, як містить усю логіку для цієї дії.

У лістингу 3.3 наведено код метода менеджера оголошень для створення нового оголошення.

Лістинг 3.3 – Код метода менеджера оголошень для створення оголошення

```
public int Create(Advert advert)
{
var result = 0;

advert.Created = DateTime.Now;
advert.Updated = DateTime.Now;

var newAdvert = _applicationContext.Adverts.Add(advert);

result = _applicationContext.SaveChanges();

return result == 0 ? result : newAdvert.Entity.Id;
}
```

У даному методі перед тим як створити оголошення, вказуються значення полів об’єкта оголошення, такі як Created і Updated. Дані поля вказують коли було створено дане оголошення, потім об’єкт передається методам EntityFramework і зміни зберігаються. Після створення перевірки, перевіряється результат і якщо сталась якась помилка, оголошення не буде створене і користувач буде повідомлений про помилку.

Якщо створення оголошення пройде успішно, користувач буде перенаправлений на сторінку перегляду вибраного оголошення із рекомендаціями (рис. 3.8).

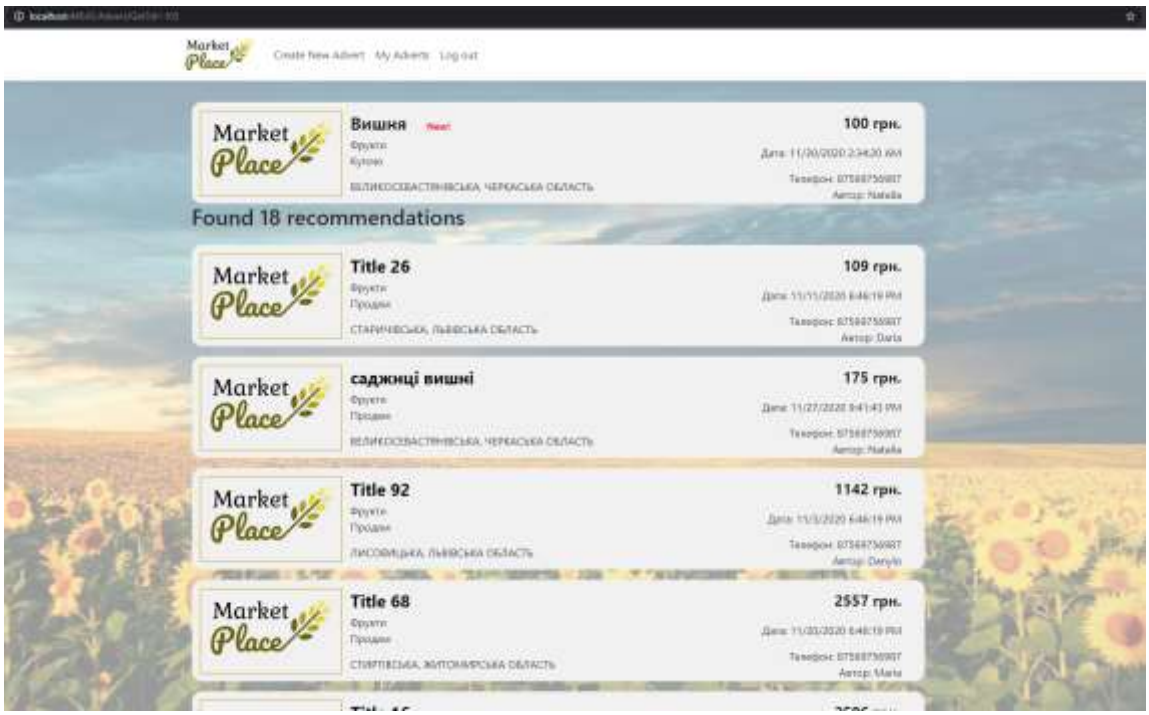

Рисунок 3.8 - Перегляд оголошення із рекомендаціями

Проект Market Place Data (MP.Data) відповідає за базу даних, сутності на основі яких будувалися таблиці з допомогою EntityFramework, моделі-об’єкти, для налаштування сервісу Identity Server [9], який використовується для роботи з користувачами, і всіма процесами, пов’язаними із ними, до прикладу: реєстрація, авторизація, тощо (рис. 3.9).

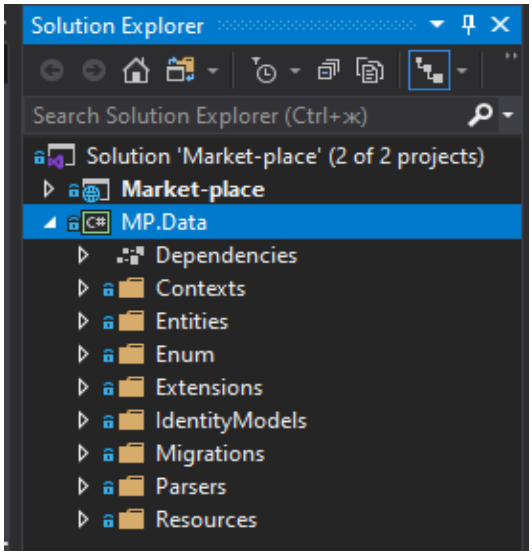

Рисунок 3.9 - Структура проекту MP.Data

Основними директоріями у даному проекті є Contexts, яка містить клас, з допомогою якого відбувається взаємодія із базою даних з використанням елементів EntityFramework. Entities містить всі сутності на основі яких будувались таблиці у базі даних. IdentityModels містить класи користувачів і ролей, які використовуються Identity Server для всіх процесів із користувачами (рис. 3.10).

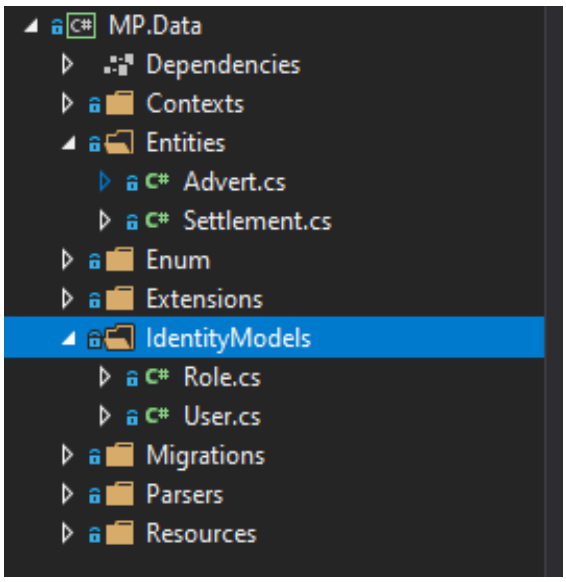

Рисунок 3.10 - Вміст Entities і IdentityModels директорій

Resources і Parsers містять файли і класи з методами, які дозволяють заповнити базу даних населеними пунктами із файлу формату JSON (лістинг. 3.4) при старті продукту у місце де буде його розгорнуто.

Лістинг 3.4 – Текст файлу у вигляді JSON формату


```
{
      "FirstLevel": "0100000000",
      "SecondLevel": 0,
      "ThirdLevel": 0,
      "ForthLevel": 0,
      "Category": null,
      "Name": "АВТОНОМНА РЕСПУБЛІКА КРИМ/М.СІМФЕРОПОЛЬ"
},
{
      "FirstLevel": "0100000000",
      "SecondLevel": "0110000000",
      "ThirdLevel": 0,
      "ForthLevel": 0,
      "Category": null,
      "Name": "МІСТА АВТОНОМНОЇ РЕСПУБЛІКИ КРИМ"
},
{
      "FirstLevel": "0100000000",
      "SecondLevel": "0110100000",
      "ThirdLevel": 0,
      "ForthLevel": 0,
      "Category": null,
      "Name": "СІМФЕРОПОЛЬ"
},
{
      "FirstLevel": "0100000000",
      "SecondLevel": "0110100000",
      "ThirdLevel": "0110130000",
      "ForthLevel": 0,
      "Category": null,
      "Name": "РАЙОНИ М.СІМФЕРОПОЛЯ"
```

```
},
{
      "FirstLevel": "0100000000",
      "SecondLevel": "0110100000",
      "ThirdLevel": "0110136300",
      "ForthLevel": 0,
      "Category": "P",
      "Name": "ЗАЛІЗНИЧНИЙ"
},
{
      "FirstLevel": "0100000000",
      "SecondLevel": "0110100000",
      "ThirdLevel": "0110136600",
      "ForthLevel": 0,
      "Category": "P",
      "Name": "КИЇВСЬКИЙ"
},
```


Даний файл містить назви всіх населених пунктів України різних категорій, міста, села, селища, і села міського типу, загалом їх біля 40 тисяч. Виконуючи певні маніпуляції, назви вносяться у базу даних.

3.4 Структура класів і схема їх взаємодії

В процесі реалізації і написання текстів програмного продукту, вибудовується ієрархія файлів, класів та методів і схема залежностей між ними. Кожен із них як один дрібний механізм який виконує свою маленьку роль, але беручи все разом отримуємо величезний і складний механізм, який працює і виконує своє призначення.

На рисунку 3.11 зображена діаграма зв`язків між класами яка сгенерована середовищем програмування Visual Studio.

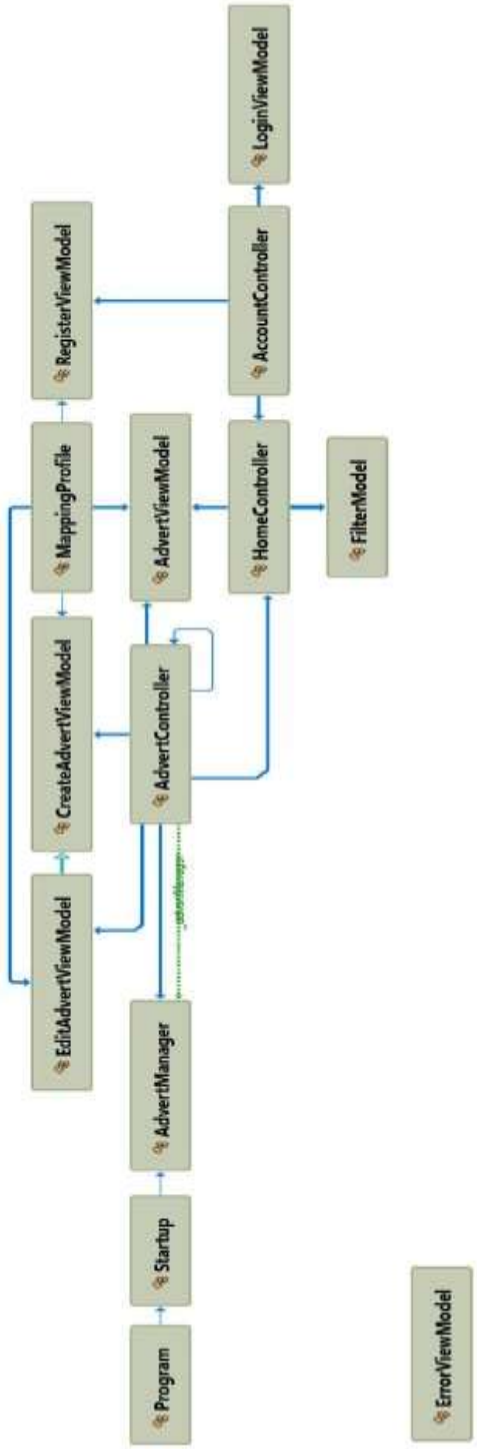

Рисунок 3.11 – Діаграма зв`язків між класами

Першим класом при старті програми є метод Main у клас Program. З допомогою внутрішніх реалізацій і бібліотек .NET Core, а саме класу Host із Microsoft.AspNetCore.Hosting, запускається сервер із налаштуваннями які прописані у класі Startup. Код класу Startup можна переглянути у додатку А. У даному класі описані всі налаштування програми, вказуються і налаштовуються

внутрішні класи та сервіси які виконують різні ролі. У методі ConfigServices, здійснюється налаштування і підключення сервісів, дана технологія називається Dependency injection (ін`єкція залежностей), існує багато бібліотек і рішень які також дозволяють це реалізувати, але .NET Core уже має вбудований інструмент, який розташований у Microsoft .Extensions.DependencyInjection.

Є три види життєвого циклу сервісу у Dependency Injection:

1. Transient – перед кожним зверненням до сервісу буде створюватися нових екземпляр класу даного сервісу із усіма його методами і властивостями;

2. Scoped – дозволяє не перестворювати екземпляри класу сервісу в межах обробки одного запиту;

3. Singleton – екземпляр класу сервісу буде створений при першому звернені до нього і більше ніколи не буде перестворюватися поки не буде перерваний робочий цикл програми.

Застосовується це досить просто, достатньо лише додати інтерфейс який реалізує даний клас через який буде взаємодія, або саму назву класу як вхідний параметр конструктора іншого класу (див. рис. 3.12), також є можливість реалізувати Dependency Injection через ін`єкцію передаючи залежність як вхідний параметр у метод класу або через ексесори (методи get і set) будь якої властивості класу, але останні два способи зустрічаються дуже ріко і застосовуються лише у рідкісних випадках.

Найпопулярніший і легкий у реалізації це через вхідний параметр конструктора. До прикладу ін`єкція сервісу AdvertManager у AdvertController клас контролер який відповідає за всю взаємодію із видами у сервісі оголошень, що відповідає за роботу із оголошеннями і містить всю логіку і функції для реалізації даних процесів.

Після ін`єкції, користуючись екземпляром AdvertManager, у кожному action (метод у контролері) можна працювати із об`єктами оголошенн. Створювати нові, редагувати існуючі або й видаляти ті які вже не потрібні користувачу, на кожну операцію є свій метод у контролері і відповідний метод

у AdvertManager із всією необхідною логікою, а саме ін`єкцією сервісів для роботи із базою даних, конвертування об’єктів тощо.

У лістингу 3.5 наведено приклад коду ін`єкції екземпляру класу менеджера оголошень AdvertManager.

Лістинг 3.5 – Приклад коду ін`єкції залежностей

```
public class AdvertController : Controller
{
	private AdvertManager _advertManager;
	private readonly ILogger<AdvertController> _logger;
	private readonly ApplicationContext _applicationContext;
	private readonly IMapper _mapper;

	public AdvertController(AdvertManager advertManager,
		ILogger<AdvertController> logger,
		ApplicationContext applicationContext,
		IMapper mapper)
		{
			_advertManager = advertManager;
			_logger = logger;
			_applicationContext = applicationContext;
			_mapper = mapper;
		}
```

Серед сервісів які налаштовуються у класі Startup є ApplicationContext, який відповідає за роботу із базою даних через EntityFramework Core, клас додається як сервіс, щоб згодом у будь якому класі де це потрібно і слід робити, можна було це застосувати;

Також було налаштовано сервіс Identity. Даний сервіс відповідає за всю роботу із користувача у системі. Авторизація, реєстрація, пароль, зміна паролю, усе це робить він. Для налаштування потрібні два класи User і Role, які будуть відповідати користувчені і його ролі відповідно. Дані класи мають

наслідуватися від IdentityUser і IdetityRole. Дані класи уже мають всі необхідні властивості, до прикладу ім`я, мобільний номер тощо, для користувача і назва ролі, нормалізована назва ролі. При необхідності це можна розшири, тому і застосовується наслідування. Потрібні властивості додаються у класі які були створені, а саме User і Role.

До налаштувань Identity можна віднести конфігурації IdentityOptions. Тут вказуються всі налашування які потрібні у програмі, налаштування паролю користувача, тобто якої складності він має бути, мінімальна довжина, наявність спеціальних символів, тощо. Налаштовується ще блокування при великій кількості невдалих спроб увійти у систему. І чи можна користувачам системи використовувати один email для декількох акаунтів у системі, використання спеціальних символів у логіні користувача, тощо.

Налаштовується як сервіс AutoMapper, приклад ін`єкції даного сервісу також зображений на рисунку 3.12. Дана бібліотека дозволяє швидко і зручно налаштувати конвертування одного об’єкта класу в інший. Для цього слід вказати деталі даного конвертування у класі який наслідує Profile клас бібліотеки із всім необхідним для цього.

У лістингу 3.6 наведено клас із налаштуваннями мапингу типів класів.

Лістинг 3.6 – Клас із налаштуваннями мапингу типів класів

```
using AutoMapper;
using MP.Data.Entities;
using MP.Data.IdentityModels;

namespace Market_place.Models
{
      public class MappingProfile : Profile
      {
      public MappingProfile()
      {
      CreateMap<RegisterViewModel, User>();
```

```
        CreateMap<Advert, AdvertViewModel>()
              .ForMember(dest => dest.Type,
                    opt => opt.MapFrom(src => src.Type))
              .ForMember(dest => dest.Category,
                    opt => opt.MapFrom(src => src.Category))
              .ForMember(dest => dest.UserCompany,
                    opt => opt.MapFrom(src => src.User.Company))
              .ForMember(dest => dest.UserCity,
                    opt => opt.MapFrom(src => src.User.City + ", "))
              .ForMember(dest => dest.UserRegion,
                    opt => opt.MapFrom(src => src.User.Region))
              .ForMember(dest => dest.UserName,
                    opt => opt.MapFrom(src => $"{src.User.FirstName}
              {src.User.LastName}"));
        CreateMap<CreateAdvertViewModel, Advert>();

        CreateMap<Advert, EditAdvertViewModel>();i
              CreateMap<EditAdvertViewModel, Advert>();
        }
}
```

Також у класі Startup налаштовуються шляхи до сторінок видів при автентифікації користувача у системі. Тобто яку сторінку слід показувати користувачу коли він ще не у системі, і якщо у нього немає доступу до системи.

Усі класи у системі можна умовно розділити на групи. Класи першої групи відповідають і налаштовують середовище у якій будуть працювати інші дві групи. У другій групі знаходяться поведінкові класи, тобто класи сервіси, класи контролери тощо. Вони відповідають за логіку системи і здійснюють всі свої процеси на останній групі класів, які відіграють роль моделей, тобто класи екземпляри який є сутностями, до прикладу сутність користувача це екземпляр класу User із всіма даними саме цього користувача, і вже в залежності від необхідності здійснюються операції над цими даними і властивостями.

Моделі тобто останню групу також можна розділити умовно на дві групи. Перша група це сутності на рівні сервісів і бази даних, у них значно більше даних і властивостей, а ніж у іншої група, яка у свою чергу відіграє роль моделей видів, які застосовуються у ASP.NET MVC, в якості об’єктів транспортування даних і властивостей до контролерів і відображення їх на сторінках видів. Об’єкти цих двох умовних груп конвертуються між собою, щоб розділити логіку відображення даних і логіку роботи над даними у сервісах і у базі даних.

3.5 Висновок до третього розділу

У даному розділі було сплановано і розроблено архітектуру програмного продукту і можливістю розширення у майбутньому використовуючи вибранні середовища програмування і системи контролю версій з врахуванням всі переваг і недоліків.

Було спроектовано графічний інтерфейс системи, щоб кінцевим користувачам було просто і зручно з орієнтуватися і користуватися веб інтерфейсом в цілому.

Використовуючи обрані технології для реалізації продукту, було створено систему яка спроможна швидко обробляти дії користувача у системі тим самим спрощуючи і вдосконалюючи сам процес реалізації і збуту сільськогосподарської продукції користувача.

В процесі реалізації було набуто багато знань і вмінь, які знайдуть своє застосування у майбутньому, тобто вміння аналізувати і реалізовувати архітектурні рішення, проектування графічного інтерфейсу користувача, реалізація функціональності сервісу сучасними технологіями.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

На всіх підприємствах створюються безпечні та здорові умови праці, встановлюються правові засади регулювання відносин у галузі охорони праці між роботодавцями та працівниками, а також створюються умови праці, що відповідають вимогам збереження життя і здоров'я працівників у процесі трудової діяльності.

Забезпечення здорових і безпечних умов праці покладається на адміністрацію підприємства. Адміністрація зобов'язана впроваджувати сучасні засоби техніки безпеки, попереджуючі виробничий травматизм, і забезпечувати санітарно-гігієнічні умови, що запобігають виникненню професійних захворювань працівників.

Метою охорони праці є науковий аналіз умов праці, технологічних процесів, апаратури та обладнання з точки зору можливості виникнення появи небезпечних факторів, виділення шкідливих виробничих речовин. На основі такого аналізу визначаються небезпечні ділянки виробництва, можливі аварійні ситуації і розробляються заходи щодо їх усунення або обмеження наслідків.

4.1 Аналіз санітарно-гігієнічних умов на робочому місці програміста: мікроклімат, освітленість робочої поверхні, шуми, випромінювання

Із розвитком інформаційних систем і технологій все більше людей стають офісними працівниками. Причому у порівнянні з представниками інших галузей, кількість швидко росте.

В результаті цього, працедавці повинні дбати про створення безпечних, відповідних і здорових умов праці, що передбачено Конституцією України (ч. 4 ст. 43), ст. 153 Кодексу законів про працю України, ст. 6 та ч. 1 ст. 13 Закону України “Про охорону праці” [23].

Але на практиці цих правил не всі дотримуються. На сьогоднішній день, не є рідкістю робота у офісах які розташовані у підвальних, тісних, малооосвітленийй і не провітрюваних, підвальних приміщеннях.

При систематичному впливі виробничих факторів, які не відповідають нормативним показникам, в результаті цього зростає рівень захворюваності органів зору, слуху, руху та нервової системи.

Таким чином, вивчення всіх умов праці на робочому місці є необхідним для запобігання впливу небезпечних та шкідливих факторів на організм.

Основними із санітарно-гігієнічних вимог до умов праці в офісних приміщеннях є розташування і розмір робочого місця. Площа одного робочого місця повинна складати не менше як 6,0 м2, а об'єм не менший за 20 м3. Робочі місця працівників повинні бути розташовані подальше від стін із вікнами, а саме не менше ніж 1 м і 1,4 м від звичайної стіни. Заборонено облаштовувати робочі місця у підвальних приміщеннях будинків. Приміщення повинні бути облаштовані не полімерними матеріалами, оскільки вони виділяють шкідливі хімічні речовини у повітря. Поверхня підлоги повинно бути матовою і рівною, неслизькою і з антистатичними властивостями [27].

Також особливу роль відіграє психологічний комфорт у приміщенні, на це впливає палітра кольорів. До того ж, правильно підібрані кольори можуть покращити продуктивність працівників.

Світлі, пастельні тони кольорів, до прикладу, світлі відтінки зеленого, синього, сірого кольорів, можлива їх комбінація. Яскраві кольори, такі як червоний, жовтогарячий, навіюють втому, роздратування і агресію.

Вологе прибирання є обов'язковим для будь якого офісного приміщення, для того щоб не допустити запилення підлоги та меблів.

Конструкція столу та крісла повинна бути максимально зручною із можливістю регулювання для підтримання правильної роботі пози та забезпечити оптимальне розміщення обладнання: дисплеїв, клавіатури, ноутбука, мишки, тощо.

Приміщення повинні бути облаштовані системами опалення і кондиціонування повітря щоб забезпечити оптимальний мікроклімат [25]:

- температура повітря повинна бути 22-25 С;
- вологість повітря 40-60%;
- швидкість руху повітря не більше 0,1 м/с.

При недотриманні вказаних показників мікроклімату в офісних приміщеннях, робочий день повинен бути скорочений мінімум на 10%.

Найважливішим серед вимог до умов праці є освітлення приміщення. Відомо, що досить тривала робота з комп'ютером і документами при недостатній освітленості призведе до значного перенапруження зору, що дуже шкідливо для здоров'я. КПО, тобто коефіцієнт природного освітлення має бути не нижчим ніж 1,5%. Для регулювання природного світла рекомендується використовувати жалюзі.

Робоче місце з персональним комп'ютером чи ноутбуком повинне бути розташоване так, що уникнути прямих сонячних променів в очі. Рівень освітленості на робочому місці повинна бути 300-500 лк.

Випромінювання присутнє у кожній із галузей виробництва: промисловість, сільське господарство, медицина, атомна енергетика (ядерні реактори). Ризик випромінювання присутній і при роботі з рентгенівськими установками, з радіоактивними елементами, зварних процесів, під час роботи на комп'ютері тощо.

Випромінювання можна поділити на: іонізуюче, ультрафіолетове, електромагнітне, лазерне. Іонізуюче випромінювання це такий тип випромінювання, при взаємодії із середовищем якого спричиняється виникнення електричних зарядів різних знаків. Проходячи через організм людини, воно викликає загибель клітин і порушує функції центральної нервової системи, що, згодом, викличе порушення функції заліз внутрішньої секреції і змін у судинній системі. В результаті цих змін порушується нормальне функціонування біохімічних процесів та обмін речовин, що призводить до променевої хвороби і раку.

Іонізуюче випромінювання при взаємодії з шкірою викликає опіки , сухість і випадання волосся, при взаємодії з очима викликає катаракту

Від іонізуючого випромінювання забезпечити захист можна такими методами та засобами:

- ізолювавши і оточивши джерела випромінювань з допомогою спеціальних огорож, екранів і камер;
- використовуючи дистанційне керування;
- використовуючи засоби індивідуального захисту;
- застосовуючи прилади сигналізації і контролю якості повітря тощо;
- зменшивши час перебування працівників у радіаційно небезпечній зоні.

Електромагнітне випромінювання — це процес взаємодії електричного і магнітного полів що утворюють електромагнітне поле, що випромінює прискорені заряджені частинки. Його джерелами є телевізійні та радіолокаційні станції, мобільного станції радіозв’язку, високовольтні мережі, комп’ютерна техніка тощо [26].

Вплив електромагнітних полів на організм залежить від частоти коливань, напруженості та інтенсивності електромагнітного поля і тривалості його впливу.

Через підвищений рівень електромагнітних випромінювань страждає передусім нервова і серцево-судинна системи, виникають головні болі і перевтома, знижується точність рухів і порушується сон, з'являються зміни тиску крові, гіпотонія.

Від електромагнітного випромінювання слід теж захищатись і це можна зробити з допомогою таких методів:

- дистанційно контролювати і керувати процеси в екранованому приміщенні. Захисні екрани послаблюють напруженість електромагнітного поля. Виготовляють екрани у вигляді металевої сітки, розміщеної між екранованим простором та джерелом електромагнітного поля;

- організаційними заходами, а саме проведенням дозиметричного контролю, медичних оглядів, додаткових відпусток, скороченням робочого часу;
- застосуванням засобів індивідуального захисту (окулярів, масок, шоломів, рукавиць, костюмів, спеціального взуття, спецодягу, тощо).

Дослідження свідчать, що при роботі за комп'ютером важливу роль відіграють небезпечні і шкідливі чинники, які поділяються на психофізіологічні та фізичні.

Фізичні чинники це:

- підвищення напруги в електричному полі;
- збільшений рівень випромінювання електромагнітного поля;
- підвищений рівень статичної електрики;
- швидко зростаючий рівень іонізації повітря.

Психофізіологічні чинники це:

- статичні і динамічні перевантаження;
- розумове перевантаження;
- перенапруження зору при роботі з моніторами.

За даними Міжнародної організації праці (МОП) в персоналу який обслуговує техніку із дисплеями погіршується зір, проявляються м'язові болі, психічні та нервові розлади, депресія, стрес, паніка, дезорієнтація, зменшується точність рухів, розлади вестибулярного апарату, захворювання серцево-судинної системи, новоутворення, рак. Наслідки залежать від великої кількості різноманітних факторів серед них це: тривалість роботи з дисплеєм, інтенсивність роботи, тощо.

Важливо також звернути увагу на те, що автоматизм та одноманітність роботи людини викликають і порушують його самопочуття, активна ж розумова діяльність згладжує дискомфорт у роботі з відео терміналом.

Професійний ризик працівників, які обслуговують комп'ютери, пов'язаний із можливим опроміненням. Катодне, ультрафіолетове, інфрачервоне, а також радіочастотне випромінювання екрана також становить

небезпеку. Але з цього приводу у її серйозності думки спеціалістів і лікарів розходяться. Одні вважають, що випромінювання від дисплея, до приклад від звичайного телевізора, не перевищує допустимих норм. Інші наполягають, що шкода від дисплейної техніки порівняно з телевізором значно більша ніж здається. Останнє твердження пояснюється близькістю екрана та тривалістю роботи з ним.

В Україні діють розроблені нормативні документи, що регламентують і контролюють роботи з візуальними дисплейними терміналами (ДНАОП 0,00-1,31-99). Дані документи затверджені наказом Держнаглядохоронпраці від 10.02.1999 р. № 2 за умови дотримання Державних санітарних правил і норм роботи з візуальними дисплейними терміналами електронно-обчислювальних машин 3.3.2.-007-98 [24].

Найефективнішим вирішенням даних питань є широке і швидке поширення атестацій робочих місць, пов’язаних із візуальними дисплейними терміналами й електронно-обчислювальною технікою.

При цьому повинні дотримуватися на такі вимоги:

- обмеження часу перебування службовців і робітників біля дисплею;
- видачу дозволів на довільні перерви в роботі;
- обмеження контролю за обсягом оброблюваної оператором інформації;
- впровадження бригадного методу організації праці;
- створення умов для участі працівників в інших видах діяльності.

Це поки що найреальніша можливість захистити працівника, що працює біля комп’ютера, від професійного захворювання.

4.2. Організація робочого місця користувача відеодисплейним терміналом

З розвитком технологій, збільшується кількість робочих місць які обладнані комп’ютерами, ноутбуками, моніторами, проекторами, тощо. Але

мало хто знає, як правильно організувати робоче місце так, що не погіршити, а покращити продуктивність, не завдати шкоди здоров’ю працівника.

В середньому робочий день офісного працівника складає 7-8 годин (як передбачено нормами Кодексу законів про працю України) при п’яти або й шестиденному робочому тижні, можна зробити висновок, що більшу частину цього часу працівник проводить із комп'ютером.

У законодавстві України є перелік нормативно-правових актів, які так чи інакше описують і регулюють дане, і досить широке питання. Обов’язком роботодавця є забезпечити працівника комфортним, безпечним і правильно організованим робочим місцем для здійснення роботи, такі умови передбачено частиною 2 ст. 2 та ч. 1 ст. 21 КЗпП, а також ст. 13 Закону України «Про охорону праці». Даний закон визначає основні положення щодо реалізації конституційного права працівників на охорону їх життя і здоров'я у процесі трудової діяльності, на належні, безпечні і здорові умови праці, регулює за участю відповідних органів державної влади відносини між роботодавцем і працівником з питань безпеки, гігієни праці та виробничого середовища і встановлює єдиний порядок організації охорони праці в Україні.

Більшість актів описують правила і норми щодо конструкцій електронно-обчислювальної техніки, особливостей облаштування приміщень та інших вимог.

Користувачі ПК повинні бути забезпечені правильно організованим робочим місцем з урахуванням всіх відповідних вимог. Конструкції всіх елементів робочого місця та їх розташування на робочому місці повинні відповідати ергономічним вимогам з врахуванням особливостей діяльності.

Конструкція робочого місця має забезпечувати і підтримувати оптимальну робочу позу працівника. Місця слід розташовувати так, щоб природне світло падало збоку, переважно зліва.

Конструкція столу має відповідати вимогам ергономіки і забезпечувати правильне і зручне розташування девайсів на поверхні столу.

Висота поверхні столу з ПК має регулюватися в межах 680-800мм, ширина - 600-1400мм і глибина - 800-1000мм. Також робочий стіл повинен мати простір для ніг заввишки не менше а ніж 600мм і завширшки не менша ніж 500мм, глибина повинна бути не менше ніж 450мм.

Монітор повинен бути розташований на оптимальній відстані від очей користувача, яка становить 500-700мм. Розташування екрана має забезпечувати зручність зорового контакту у вертикальній площині під кутом 30° (див. рис. 4.1).

Робоче місце яке споряджене відеотерміналом, повинне дотримуватись всіх вимог, що описані законодавством України, перелік вимог описаний у таблиці 4.1 та таблиці 4.2 [26].

Рисунок 4.1 - Ергономіка робочого місця

Таблиця 4.1 - Вимоги до відеотерміналів

Найменування параметра	**Значення параметра**
Яскравість знака (яскравість фону), кд/кв. м	від 35 до 120
Зовнішня освітленість екрана, лк	від 100 до 250
Контраст (для монохромних зображень)	від 3:1 до 1,5:1
Нерівномірність яскравості в робочій зоні екрана	не більше 1,7:1

Таблиця 4.2 - Відхилення форми робочої зони екрана від прямокутності

Найменування параметра	Значення параметра
по горизонталі та вертикалі	не більше 2 %
по діагоналі	не більше 4 % відношення суми коротких сторін до суми довгих
Різниця довжин рядків або стовпчиків	не більше 2 % середнього значення
Розмір мінімального елемента зображення (пікселя) для монохромних зображень, мм	0,3
Допустима тимчасова нестабільність зображення (мигання)	не повинна бути зафіксована у 90 відсотків спостерігачів
Відбивна властивість, дзеркальне та змішане відображення (відблиск), %, (допускається виконання вимог при застосуванні приекранного фільтра)	не більше 1
Відношення ширини знака до його висоти для великих літер	від 0,7 до 0,9
Мінливість розміру знака	не більше 5 % висоти
Ширина лінії контуру знака	0,15 - 0,1 висоти знака

Висновок, облаштування робочого місця відеодисплейним терміном, набагато важливіший процес ніж це здається. Дотримування цих простих правил, зробить людей здоровішими, особливо у час карантинних обмежень.

4.3 Висновок до четвертого розділу

Здоров’я людини це найважливіший атрибут його життя, за яким слід правильно піклуватися, дотримуватись правил, які написані спеціалістами, дослідниками, лікарями.

При роботі з комп`ютерами, слід спостерігати, і тримати в порядку робоче місце, і немає різниці де це робоче місце, в офісі чи вдома, воно повинно бути правильно налаштованим, і дотримуватися всіх правил гігієни.

Бути здоровим, це право кожної людини і ніхто не повинен його обмежувати, якщо роботодавець порушує правила і закон, працівник неповинен від цього страждати і захищати себе, і оточуючих, бо закон на його стороні.

Дотримуюсь елементарних правил, можна налаштувати правильні процесі у компаніях, які зроблять потрібні умови праці, що будуть максимально продуктивними і дозволять розвивати не лише компанії, але і країну в цілому.

ВИСНОВОК

У даній кваліфікаційній роботі було проаналізовано предметну область для продукту. Досліджено структури на напрями діяльності користувача у системі. Також були проаналізовані усі бізнес процеси які відбуваються у даній системі користувачем. Для кращого розуміння розроблюваного продукту було порівняно три сервіси які надають можливість створювати оголошення: “OLX”, “Бесплатка” і “Вільний ринок”. Після порівняння функціональності систем було визначено і проаналізовано всі переваги і недоліки систем, які потрібно врахувати в процесі розробки веб-платформи “Market Place”. Було детально проаналізовано і описано варіанти використання системи, стадії та етапи розробки і порядок контролю та прийому.

Також було розроблено і спроектовано архітектуру створюваного веб-додатку яка найкраще підходить для реалізації програмного продукту і оскільки система оперує даними які слід зберігати, було ще спроектовано структуру бази даних. І оскільки база даних є реляційному, було створено діаграму зв’язків між таблицями. Було описано і обґрунтовано вибір технологій з допомогою яких буде реалізовано програмний продукт. Також обґрунтовано вибір, налаштування і підключення системи контролю версій. Вибрано і інстальовано середовище програмування з допомогою якого буде розроблятися програмний продукт

Як результат було успішно розроблено веб-платформу з допомогою якої користувачі, основна частина з яких це фермери, компанії, ферми, які займаються торгівлею і реалізацією сільськогосподарської продукції.

Даний проект створювався з метою вирішення проблем з якими стикалися і сьогодні стикаються люди у цій сфері.

Список використаних джерел

1. Матеріали для обговорення. Програми стимулювання економіки для подолання наслідків COVID-19: "Економічне відновлення". [Електронний ресурс]. – 2020 – Режим доступу: URL: https://www.kmu.gov.ua/news/programa-stimulyuvannya-ekonomiki-dlya-podolannya-naslidkiv-covid-19-ekonomichne-vidnovlennya. – Дата доступу: 22.11.2020.
2. Вплив COVID-19 та карантинних обмежень на економіку України. [Електронний ресурс]. – 2020 – Режим доступу: URL: https://surl.li/gtnp.– Дата доступу 22.11.2020.
3. Цінові гойдалки: що і як подорожчає у квітні. [Електронний ресурс]. – 2016 – Режим доступу: URL: https://enovosty.com/uk/news_economy-ukr/full/514-cenovye-kacheli-chto-i-kak-podorozhaet-v-aprele. – Дата доступу: 22.11.2020.
4. Сервіс OLX [Електронний ресурс]. – 2020 – Режим доступу: URL: https://www.olx.ua/uk/ – Дата доступу: 22.11.2020.
5. Сервіс БЕСПЛАТКА [Електронний ресурс]. – 2020 – Режим доступу: URL: https://besplatka.ua/uk – Дата доступу: 22.11.2020.
6. Вільний Ринок. [Електронний ресурс]. – 2020 – Режим доступу: URL:https://www.prostir.ua/?news=vidkrytyj-rynok-startuje-onlajn-projekt-za-pidtrymky-amu. – Дата доступу: 22.11.2020.
7. Врожайний карантин: як фермери пережили період обмежень. [Електронний ресурс]. – 2020 – Режим доступу: URL: https://mind.ua/publications/20211682-vrozhajnij-karantin-yak-fermeri-perezhili-period-obmezhen. – Дата доступу: 22.11.2020.
8. .NET Core GitHub репозиторій [Електронний ресурс]. – 2017 – Режим доступу: URL: https://github.com/dotnet/aspnetcore – Дата доступу: 23.11.2020.
9. .NET Identity [Електронний ресурс]. – 2017 – Режим доступу: URL: https://docs.microsoft.com/ru-ru/aspnet/core/security/authentication/identity?view=aspnetcore-5.0&tabs=visual-studio – Дата доступу: 23.11.2020.

10. .NET LINQ – 2016 – Режим доступу: URL: https://docs.microsoft.com/ru-ru/dotnet/csharp/linq/ – Дата доступу: 23.11.2020.

11. SQL – 2020 – Режим доступу: URL: https://uk.wikipedia.org/wiki/SQL – Дата доступу: 23.11.2020.

12. Офіційний сайт Bootstrap – 2020 – Режим доступу: URL: https://getbootstrap.com/ – Дата доступу: 23.11.2020.

13. Офіційний сайт jQuery – 2020 – Режим доступу: URL: https://jquery.com/ – Дата доступу: 23.11.2020.

14. Офіційний сайт AutoMapper – 2020 – Режим доступу: URL: https://automapper.org/ – Дата доступу: 23.11.2020.

15. Інформація про операційну систему Mac OS [Електронний ресурс]. – 2020 – Режим доступу: URL: https://uk.wikipedia.org/wiki/Mac_OS – Дата доступу: 23.11.2020.

16. Офіційний сайт Visual Studio [Електронний ресурс]. – 2019 – Режим доступу: URL: https://visualstudio.microsoft.com/vs/ – Дата доступу: 23.11.2020.

17. Головна сторінка сервісу GitHub [Електронний ресурс]. – 2020 – Режим доступу: URL: https://github.com/ – Дата доступу: 23.11.2020.

18. Документація системи GIT [Електронний ресурс]. – 2020 – Режим доступу: URL: https://git-scm.com/ – Дата доступу: 23.11.2020.

19. Головна сторінка сервісу Subversion [Електронний ресурс]. – 2018 – Режим доступу: URL: http://subversion.apache.org/ – Дата доступу: 23.11.2020.

20. Головна сторінка сервісу Directual [Електронний ресурс]. – 2020 – Режим доступу: URL: https://www.directual.com/ – Дата доступу: 23.11.2020.

21. Головна сторінка сервісу Mercurial [Електронний ресурс]. – 2019 – Режим доступу: URL: https://www.mercurial-scm.org/ – Дата доступу: 23.11.2020.

22. Головна сторінка сервісу TFS [Електронний ресурс]. – 2020 – Режим доступу: URL: https://azure.microsoft.com/en-us/services/devops/server/ – Дата доступу: 23.11.2020.

23. Про затвердження загальних вимог стосовно забезпечення роботодавцями охорони праці працівників [Електронний ресурс]. – 2020 –

Режим доступу: URL: https://zakon.rada.gov.ua/laws/show/2694-12#Text – Дата доступу: 25.11.2020.

24. Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями [Електронний ресурс]. – 2018 – Режим доступу: URL: https://zakon.rada.gov.ua/laws/show/z0508-18#Text – Дата доступу: 25.11.2020.

25. Гігієнічні вимоги до організації роботи з візуальними дисплейними терміналами електронно-обчислювальних машин [Електронний ресурс]. – 2017 – Режим доступ: URL: https://www.buh24.com.ua/gigiyenichni-vimogi-do-organizatsiyi-roboti-z-vizualnimi-displeynimi-terminalami-elektronno-obchislyuvalnih-mashin/ – Дата доступу: 25.11.2020.

26. Аналіз і профілактика профзахворювань та виробничого травматизму [Електронний ресурс]. – 2020 – Режим доступу: URL: https://studopedia.com.ua/1_134487_analIz-I-profIlaktika-profzahvoryuvan-ta-virobnichogo-travmatizmu.html – Дата доступу: 26.11.2020.

27. Робота в офісі: основні санітарно-гігієнічні вимоги [Електронний ресурс]. – 2018 – Режим доступу: URL: https://oppb.com.ua/news/robota-v-ofisi-osnovni-sanitarno-gigiyenichni-vymogy – Дата доступу: 26.11.2020.

28. Microsoft Visual Studio Marketplace [Електронний ресурс] – 2020 – Режим доступу до ресурсу: URL: https://marketplace.visualstudio.com/ – Дата доступу: 26.11.2020.

29. Офіційний сайт середовища програмування “Rider” [Електронний ресурс] – 2020 – Режим доступу до ресурсу: URL: https://www.jetbrains.com/rider/ – Дата доступу: 26.11.2020.

30. Архітектурні шаблони [Електронний ресурс] – 2020 – Режим доступу до ресурсу: URL: https://uk.wikipedia.org/wiki/Архітектурні_шаблони_програмного_забезпечення – Дата доступу: 26.11.2020.

31. Мікросервісна архітектура [Електронний ресурс] – 2020 – Режим доступу до ресурсу: URL: https://uk.wikipedia.org/wiki/Мікросервіси – Дата доступу: 26.11.2020.

32. Модель-вид-контроллер архітектура [Электронный ресурс] – 2015 – Режим доступу до ресурсу: URL: https://habr.com/ru/post/249263/ – Дата доступу: 26.11.2020.

33. Основні поняття інтерфейсів користувача та засоби їх проектування [Електронний ресурс] – 2020 – Режим доступу до ресурсу: URL: http://elar.khnu.km.ua/jspui/bitstream/123456789/1415/2/Rozdil1.pdf – Дата доступу: 26.11.2020.

ДОДАТКИ

Додаток А

Програмний код розробленого програмного продукту «Market Place»

Лістинг А.1 – Програмний код із файлу Program.cs

```
public class Program
   {
      public static void Main(string[] args)
      {
         CreateHostBuilder(args).Build().Run();
      }

      public static IHostBuilder CreateHostBuilder(string[] args) =>
         Host.CreateDefaultBuilder(args)
            .ConfigureWebHostDefaults(webBuilder =>
            {
               webBuilder.UseStartup<Startup>();
            });
   }
```

Лістинг А.2 – Програмний код із файлу Startup.cs

```
public class Startup
   {
      public Startup(IConfiguration configuration)
      {
         Configuration = configuration;
      }

      public IConfiguration Configuration { get; }

      // This method gets called by the runtime. Use this method to add services to the container.
      public void ConfigureServices(IServiceCollection services)
      {
         services.AddDbContext<ApplicationContext>(options =>
            options.UseSqlServer(Configuration.GetConnectionString("DefaultConnection")));

         services.AddIdentity<User, Role>()
            .AddEntityFrameworkStores<ApplicationContext>();

         services.AddAutoMapper(typeof(Startup));

         services.AddTransient<AdvertManager>();

         services.Configure<IdentityOptions>(options =>
         {
            // Password settings.
            options.Password.RequireDigit = true;
            options.Password.RequireLowercase = true;
            options.Password.RequireNonAlphanumeric = true;
            options.Password.RequireUppercase = true;
            options.Password.RequiredLength = 6;
```

```
            options.Password.RequiredUniqueChars = 1;

            // Lockout settings.
            options.Lockout.DefaultLockoutTimeSpan = TimeSpan.FromMinutes(5);
            options.Lockout.MaxFailedAccessAttempts = 5;
            options.Lockout.AllowedForNewUsers = true;

            // User settings.
            options.User.RequireUniqueEmail = true;
        });

        services.AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme)
            .AddCookie(options =>
            {
                options.LoginPath = new Microsoft.AspNetCore.Http.PathString("/Account/Login");
            })
            .AddOAuthValidation();

        services.ConfigureApplicationCookie(options =>
        {
            // Cookie settings
            options.Cookie.HttpOnly = true;
            options.ExpireTimeSpan = TimeSpan.FromMinutes(5);

            options.LoginPath = "/Account/Login";
            options.AccessDeniedPath = "/Account/AccessDenied";
            options.SlidingExpiration = true;
        });

        services.AddMvc();

        services.AddControllersWithViews().AddRazorRuntimeCompilation();
    }

    // This method gets called by the runtime. Use this method to configure the HTTP request
pipeline.
    public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
    {
        if (env.IsDevelopment())
        {
            app.UseDeveloperExceptionPage();
        }
        else
        {
            app.UseExceptionHandler("/Shared/Error");
        }

        app.UseStaticFiles();

        app.UseRouting();

        app.UseAuthentication();
        app.UseAuthorization();
```

```
        app.UseEndpoints(endpoints =>
        {
           endpoints.MapControllerRoute(
              name: "default",
              pattern: "{controller=Home}/{action=Index}");
        });
        }
}
```

Лістинг А.3 – Програмний код із файлу Register.cshtml

```
@model RegisterViewModel

@{
   Layout = "_LayoutNoMenu";
}

<div class="login-bg">
   <div class="container-fluid">
      <div class="row">
         <div class="col-md-12 wrapper"></div>
      </div>
      <div class="row">
         <div class="form center">
            <div class="main-label">Registration</div>
            @using (Html.BeginForm("Register", "Account", FormMethod.Post, new { @class = ""
}))
            {
               @Html.AntiForgeryToken()
               @Html.ValidationSummary(true, "", new { @class = "text-danger" })
               <div class="row">
                  <div class="col-lg-6">
                     <div class="form-group">
                        @Html.LabelFor(m => m.FirstName, "First name")
                        @Html.TextBoxFor(m => m.FirstName, new { @class = "form-control",
placeholder = "First name" })
                        @Html.ValidationMessageFor(m => m.FirstName, "", new { @class = "text-
danger" })
                     </div>
                     <div class="form-group">
                        @Html.LabelFor(m => m.LastName, "Last name")
                        @Html.TextBoxFor(m => m.LastName, new { @class = "form-control",
placeholder = "Last name" })
                        @Html.ValidationMessageFor(m => m.LastName, "", new { @class = "text-
danger" })
                     </div>
                     <div class="form-group">
                        @Html.LabelFor(m => m.UserName, "Login")
                        @Html.TextBoxFor(m => m.UserName, new { @class = "form-control",
placeholder = "Login" })
```

```
                    @Html.ValidationMessageFor(m => m.UserName, "", new { @class = "text-
danger" })
                </div>
                <div class="form-group">
                    @Html.LabelFor(m => m.Region, "Region")
                    @Html.DropDownListFor(m => m.Region, ViewData["Regions"] as
SelectList, new { @class = "form-control", placeholder = "Region" })
                    @Html.ValidationMessageFor(m => m.Region, "", new { @class = "text-
danger" })
                </div>
                <div class="form-group">
                    @Html.LabelFor(m => m.City, "City/Village")
                    @Html.TextBoxFor(m => m.City, new { @class = "form-control", placeholder
= "City/Village" })
                    @Html.ValidationMessageFor(m => m.City, "", new { @class = "text-danger"
})
                </div>
            </div>
            <div class="col-lg-6">
                <div class="form-group">
                    @Html.LabelFor(m => m.PhoneNumber, "Phone Number")
                    @Html.TextBoxFor(m => m.PhoneNumber, new { @class = "form-control",
placeholder = "Phone Number" })
                    @Html.ValidationMessageFor(m => m.PhoneNumber, "", new { @class =
"text-danger" })
                </div>
                <div class="form-group">
                    @Html.LabelFor(m => m.Company, "Company")
                    @Html.TextBoxFor(m => m.Company, new { @class = "form-control",
placeholder = "Company" })
                    @Html.ValidationMessageFor(m => m.Company, "", new { @class = "text-
danger" })
                </div>
                <div class="form-group">
                    @Html.LabelFor(m => m.Email, "Email")
                    @Html.TextBoxFor(m => m.Email, new { @class = "form-control",
placeholder = "Email" })
                    @Html.ValidationMessageFor(m => m.Email, "", new { @class = "text-
danger" })
                </div>
                <div class="form-group">
                    @Html.LabelFor(m => m.Password, "Password")
                    @Html.PasswordFor(m => m.Password, new { @class = "form-control",
placeholder = "Password" })
                    @Html.ValidationMessageFor(m => m.Password, "", new { @class = "text-
danger" })
                </div>
                <div class="form-group">
                    @Html.LabelFor(m => m.PasswordConfirm, "Confirm password")
                    @Html.PasswordFor(m => m.PasswordConfirm, new { @class = "form-
control", placeholder = "Confirm password" })
                    @Html.ValidationMessageFor(m => m.PasswordConfirm, "", new { @class =
"text-danger" })
```

```
                </div>
            </div>
            <div class="col-lg-12 form-group">
                <input type="submit" value="Registrate" class="btn btn-primary" />
            </div>
        </div>
    }
    </div>
  </div>
 </div>
</div>
```

Лістинг А.4 – Програмний код із файлу Login.cshtml

```
@model LoginViewModel

@{
   Layout = "_LayoutNoMenu";
}

<div class="login-bg">
   <div class="container-fluid">
      <div class="row">
         <div class="col-md-12 wrapper"></div>
      </div>
      <div class="col-md-4 form center">
         <div class="logo">
            <img class="center" src="~/images/logo.png"/>
         </div>
         @using (Html.BeginForm("Login", "Account"))
         {
            @Html.ValidationSummary(true, "", new { @class = "text-danger" })
            <div class="form-group">
               @Html.LabelFor(m => m.UserName, "Login")
               @Html.TextBoxFor(m => m.UserName, new { @class = "form-control", placeholder =
"Login" })
               @Html.ValidationMessageFor(m => m.UserName, "", new { @class = "text-danger"
})
            </div>
            <div class="form-group">
               @Html.LabelFor(m => m.Password, "Password")
               @Html.PasswordFor(m => m.Password, new { @class = "form-control", placeholder =
"Password" })
               @Html.ValidationMessageFor(m => m.Password, "", new { @class = "text-danger" })
            </div>
            <div class="form-group">
               <a href="@Url.Action("Register", "Account")">Registration</a>
            </div>
            <div class="form-group">
               <input type="submit" value="Log In" class="btn btn-primary" />
            </div>
         }
```

```
      </div>
   </div>
</div>
```

Лістинг А.5 – Програмний код із файлу Create.cshtml

```
@model CreateAdvertViewModel

@{
   Layout = "_Layout";
}

<div class="row">
   <div class="container-fluid">
      <div class="form center">
         <div class="col-md-12">
            @using (Html.BeginForm("Create", "Advert"))
            {
               @Html.ValidationSummary(true, "", new { @class = "text-danger" })
               <div class="form-group">
                  @Html.LabelFor(m => m.Title, "Title")
                  @Html.TextBoxFor(m => m.Title, new { @class = "form-control", placeholder =
"Title" })
                  @Html.ValidationMessageFor(m => m.Title, "", new { @class = "text-danger" })
               </div>
               <div class="form-group">
                  @Html.LabelFor(m => m.Type, "Title")
                  @Html.DropDownListFor(m => m.Type, ViewData["Types"] as SelectList)
                  @Html.ValidationMessageFor(m => m.Type, "", new { @class = "text-danger" })
               </div>
               <div class="form-group">
                  @Html.LabelFor(m => m.Category, "Category")
                  @Html.DropDownListFor(m => m.Category, ViewData["Categories"] as SelectList)
                  @Html.ValidationMessageFor(m => m.Category, "", new { @class = "text-danger"
})
               </div>
               <div class="form-group">
                  @Html.LabelFor(m => m.Price, "Price")
                  @Html.TextBoxFor(m => m.Price, new { @class = "form-control", placeholder =
"Price" })
                  @Html.ValidationMessageFor(m => m.Price, "", new { @class = "text-danger" })
               </div>
               <div class="form-group">
                  @Html.LabelFor(m => m.Count, "Count")
                  @Html.TextBoxFor(m => m.Count, new { @class = "form-control", placeholder =
"Count" })
                  @Html.ValidationMessageFor(m => m.Count, "", new { @class = "text-danger" })
               </div>
               <div class="form-group">
                  @Html.LabelFor(m => m.Description, "Description")
                  @Html.TextBoxFor(m => m.Description, new { @class = "form-control",
placeholder = "Description" })
```

```
                @Html.ValidationMessageFor(m => m.Description, "", new { @class = "text-
danger" })
            </div>

            <div class="form-group">
                <input type="submit" value="Create" class="btn btn-primary" />
            </div>
        }
    </div>
  </div>
 </div>
</div>
```

Лістинг А.6 – Програмний код із файлу Edit.cshtml

```
@model EditAdvertViewModel

@{
  Layout = "_Layout";
}

<div class="row">
  <div class="container-fluid">
    <div class="form center">
      <div class="col-md-12">
        @using (Html.BeginForm("Edit", "Advert"))
        {
          @Html.HiddenFor(m => m.ID)
          @Html.HiddenFor(m => m.UserID)
          @Html.ValidationSummary(true, "", new { @class = "text-danger" })
          <div class="form-group">
            @Html.LabelFor(m => m.Title, "Title")
            @Html.TextBoxFor(m => m.Title, new { @class = "form-control", placeholder =
"Title" })
            @Html.ValidationMessageFor(m => m.Title, "", new { @class = "text-danger" })
          </div>
          <div class="form-group">
            @Html.LabelFor(m => m.Type, "Title")
            @Html.DropDownListFor(m => m.Type, ViewData["Types"] as SelectList)
            @Html.ValidationMessageFor(m => m.Type, "", new { @class = "text-danger" })
          </div>
          <div class="form-group">
            @Html.LabelFor(m => m.Category, "Category")
            @Html.DropDownListFor(m => m.Category, ViewData["Categories"] as SelectList)
            @Html.ValidationMessageFor(m => m.Category, "", new { @class = "text-danger"
})
          </div>
          <div class="form-group">
            @Html.LabelFor(m => m.Price, "Price")
            @Html.TextBoxFor(m => m.Price, new { @class = "form-control", placeholder =
"Price" })
            @Html.ValidationMessageFor(m => m.Price, "", new { @class = "text-danger" })
```

```
                </div>
                <div class="form-group">
                    @Html.LabelFor(m => m.Count, "Count")
                    @Html.TextBoxFor(m => m.Count, new { @class = "form-control", placeholder =
"Count" })
                    @Html.ValidationMessageFor(m => m.Count, "", new { @class = "text-danger" })
                </div>
                <div class="form-group">
                    @Html.LabelFor(m => m.Description, "Description")
                    @Html.TextBoxFor(m => m.Description, new { @class = "form-control",
placeholder = "Description" })
                    @Html.ValidationMessageFor(m => m.Description, "", new { @class = "text-
danger" })
                </div>

                <div class="form-group">
                    <input type="submit" value="Save" class="btn btn-primary" />
                </div>
            }
        </div>
    </div>
</div>
```

Лістинг А.7 – Програмний код із файлу AdvertManager.cshtml

```
using System;
using MP.Data.Contexts;
using MP.Data.Entities;
using System.Linq;

namespace Market_place.Managers
{
    public class AdvertManager
    {
        private readonly ApplicationContext _applicationContext;

        public AdvertManager(ApplicationContext applicationContext)
        {
            _applicationContext = applicationContext;
        }

        public int Create(Advert advert)
        {
            var result = 0;

            var dbAdvert = _applicationContext.Adverts.SingleOrDefault(a => a.ID == advert.ID);
            if (dbAdvert != null)
            {
                return result;
            }
```

```
            dbAdvert.Created = DateTime.Now;
            dbAdvert.Updated = DateTime.Now;

            var newAdvert = _applicationContext.Adverts.Add(dbAdvert);

            result = _applicationContext.SaveChanges();

            return result == 0 ? result : newAdvert.Entity.ID;
        }

        public void Edit(Advert advert)
        {
            var advertDb = _applicationContext.Adverts.SingleOrDefault(a => a.ID == advert.ID);
            if (advertDb == null)
            {
                return;
            }

            advertDb.Category = advert.Category;
            advertDb.Count = advert.Count;
            advertDb.Description = advert.Description;
            advertDb.Photo = advert.Photo;
            advertDb.Price = advert.Price;
            advertDb.Title = advert.Title;
            advertDb.Type = advert.Type;
            advertDb.Updated = DateTime.Now;

            _applicationContext.Update(advertDb);

            _applicationContext.SaveChanges();
        }
    }
}
```

Лістинг А.8 – Програмний код із файлу _SharedAdvert.cshtml

```
@model AdvertViewModel
@{
    var link = ViewData["Link"];
}
<div class="col-lg-12 advert-container">
    <div class="advert">
        <div class="advert-photo"><img class="img-center" src="~/images/logo.png" />        </div>
        <div class="advert-info">
            <table class="advert-table">
                <tbody>
                    <tr>
                        <td onclick="location.href = '@link'">
                            <div class="advert-title">@Model.Title</div>
                            @if (DateTime.Now.Subtract(Model.Created).Hours < 1)
                            {
                                <div class="new-advert">NEW!</div>
```

```
                    }
                </td>
                <td>
                    <div class="to-right advert-price">@Model.Price грн.</div>
                </td>
            </tr>
            <tr>
                <td>
                    <div>@Model.Category</div>
                    <div>@Model.Type</div>
                </td>
                <td><div class="to-right">Date: @Model.Created</div><br /></td>
            </tr>
            <tr>
                <td>
                    <div class="advert-city">@Model.UserCity</div>
                    <div class="advert-region">@Model.UserRegion region</div>
                </td>
                <td>
                    @if (!string.IsNullOrEmpty(Model.UserPhoneNumber))
                    {
                        <div class="to-right">Phone: @Model.UserPhoneNumber</div><br />
                    }
                    <div class="to-right">Created by:
@Model.UserName@(!string.IsNullOrEmpty(Model.UserCompany) ?
$@""" {Model.UserCompany}""" : "")</div>
                </td>
            </tr>
        </tbody>
    </table>
</div>
</div>
</div>
```

Додаток Б

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний технічний університет імені Івана Пулюя (Україна)
Національна академія наук України
Університет імені П'єра і Марії Кюрі (Франція)
Маріборский університет (Словения)
Технічний університет у Кошице (Словаччина)
Вільнюський технічний університет ім. Гедимінаса (Литва)
Шяуляйська державна колегія (Литва)
Жешувський політехнічний університет ім. Лукасевича (Польща)
Білоруський національний технічний університет (Республіка Білорусь)
Міжнародний університет цивільної авіації (Марокко)
Національний університет біоресурсів і природокористування України (Україна)
Наукове товариство ім. Шевченка
ГО «Асоціація випускників Тернопільського національного технічного університету імені Івана Пулюя»

АКТУАЛЬНІ ЗАДАЧІ СУЧАСНИХ ТЕХНОЛОГІЙ

Збірник

тез доповідей

Том ІІ

ІХ Міжнародної науково-технічної конференції молодих учених та студентів

25-26 листопада 2020 року

УКРАЇНА
ТЕРНОПІЛЬ – 2020

УДК 004:02
А. М. Слободяник
Тернопільського національного технічного університету імені Івана Пулюя, Україна

ІНФОРМАЦІЙНА СИСТЕМА ДЛЯ ВИРІШЕННЯ ПРОБЛЕМ ІЗ РЕАЛІЗАЦІЄЮ СІЛЬСЬКОГОСПОДАРСЬКОЇ ПРОДУКЦІЇ

A.M. Slobodyanyk

INFORMATION SYSTEM FOR SOLVING PROBLEMS WITH SALE OF AGRICULTURAL PRODUCTS

Із появою коронавірусу, карантинні обмеження стали випробуванням на стійкість і міцність для всієї економіки країни, і сфери агропромислового комплексу.

Це дозволило проблемам краще проявитися і привернути увагу до моментів які слід було і потрібно зараз вдосконалити. Порушення логістики, паніка серед покупців, стрибки цін - усе це примусило учасників ринку боротися за виживання і шукати нові шляхи, і способи збуту та форми співпраці з посередниками [2].

При цьому масовий, і в основному, примусовий вихід в онлайн формат, показав чудові результати. Настільки ефективні, що Кабінет Міністрів України у програмі відновлення економіки окремо виділив і описав створення централізованого онлайн-сервісу, яким будуть користуватися малі і середні ферми для реалізації своєї продукції [1]. Тому як один із способів щоб вирішити або спростити вирішення проблем пропонується веб-платформа Market Place.

Протягом карантину фермери і аграрії стикнулися із вимушеними змінами у роботі, проблемами і викликами.

Основними з яких є це:

- Цінові "гойдалки" - основною причиною коливання цін є психологічний фактор покупців і впевненість багатьох українців у тому, що деякі фрукти та овочі є ліками від коронавірусу [3]. Хлібопекарі утримували ціни на свою продукцію і беручи до уваги те, що за поточний сезон ціни на зерно зросли майже на 30%, борошно подорожчало на 21-24% залежно від сорту і це 39% від собівартості хлібобулочних виробів, і ніхто з виробників не підвищує ціни [5];

- Зміна логістичних процесів є однією з найголовніших проблем, спричинених протиепідемічними заходами, стали обмеження транспортних сполучень і перевезень у більшості країн. Товари надходять нерівномірно, тому трапляються коливання цін. Сповільнення руху міжнародних транспортних перевезень у деяких місцях призводило до несвоєчасного або не в повному обсязі постачання імпортних продуктів. Вимоги карантину спровокували і підвищений попит на кадри у галузі вантажоперевезень. Оскільки водій, який віз фуру продукції до Європи, після повернення мав пройти 14-денне обсервування та не міг вийти у черговий рейс, тому перевізник мав тимчасово знайти йому заміну іншою кандидатурою [5];

- Основні побоювання фермерів. Нестандартні умови, спричинені карантином, змусили аграріїв щодня приймати нові рішення на всіх ланках діяльності: продажі, виробництво, продажі, переробка, логістика тощо. Фактично фермери перебувають у максимально непередбачуваних умовах. На першому місці у них ризики пов'язані із невиконанням своїх зобов'язань партнерами: як клієнтами, так і постачальниками. Важливо методично розраховувати і передбачати ризики, при цьому забезпечити якість і безпечність продукції, яка випускається, і звісно безпеку працівників [5];

- Фінансові втрати через закриття продовольчих ринків. Закриття продовольчих ринків більшою мірою вплинуло на малих підприємців, продукція яких не змогла

потрапити на полиці магазинів і супермаркетів. "В Agricom Group до карантину середні продажі продукції на ринках складали 10% – 20%, а після закриття впали до 0%. Проте, ці обсяги компенсувалися збільшенням замовлень від національних і регіональних мереж магазинів і супермаркетів" - розповів виконавчий директор компанії "Agricom Group" Петро Мельник журналісту Mind.ua [5].

Під час карантину, з'явилось декілька стартапів онлайн-платформ які якраз і створювалися, щоб допомогти аграріям вирішити більшість вище описаних проблем. Одним із них є "Відкритий ринок" реалізований ГО «Освітньо-аналітичний центр розвитку громад» у партнерстві з Асоціацією міст України [4]. Вони створили онлайн-платформу де виробники та споживачі сільськогосподарської продукції можуть зустрітися без посередників та допомоги один одному у цей складний період карантину.

Створення онлайн-платформи є ефективним рішенням, але воно має бути максимально зручним і простим для кінцевого користувача, також повинно максимально вирішувати всі або майже всі його проблеми. Економити час і бути максимально ефективним. Тому онлайн платформа повинна містити оголошення двох типів, на купівлю і на продаж, саме такою є платформа Market Place. Це дозволить користувачам розділити всі оголошення у системі, створювати і шукати те, що потрібно. Даний сервіс буде обслуговувати не лише користувачів які щось продають, але і користувачів які у пошуках потрібного їм товару чи продукту. Найпопулярніші сервіси даної функціональності не маю, оскільки більш орієнтовані лише на більшу частину ринку, але не весь ринок, а саме на створення оголошень на продаж.

До прикладу, фермер, зареєструвавшись у Market Place сервісі, зможе створити оголошення на продаж картоплі чи моркви яку вирощує, і згодом обрати серед тих хто відповів на оголошення, найкращого покупця котрий запропонував найвигіднішу ціну. Також може створити оголошення на купівлю, і йому як фермеру потрібні добрива, створивши оголошення на купівлю добрив, це зекономить багато часу не шукаючи людей, номерів телефонів і оголошень у газетах не тративши свій дорогоцінний час і зайнятися своїми справами пов'язаними з фермою, або з сім'єю, що теж дуже важливо. Пройде певний період часу і зайшовши у веб-сервіс зможе обрати серед запропонованих ті добрива, які потрібно і по найкращій ціні.

Це чудове рішення яке вирішить проблеми фермерів і людей які теж займаються реалізацією сільськогосподарської продукції. Також онлайн-платформи, що були створені, і Market Place у тому числі, допоможуть швидко оговтатись після карантину фермам і підприємствам, і допоможуть в рази покращити реалізація у майбутньому.

Література

1.Матеріали для обговорення Програми стимулювання економіки для подолання наслідків COVID-19: "Економічне відновлення". // [Електронний ресурс]. - Режим доступу: https://www.kmu.gov.ua/news/programa-stimulyuvannya-ekonomiki-dlya-podolannya-naslidkiv-covid-19-ekonomichne-vidnovlennya

2.Вплив COVID-19 та карантинних обмежень на економіку України. // [Електронний ресурс]. - Режим доступу: https://surl.li/gtnp

3.Цінові гойдалки: що і як подорожчає у квітні. // [Електронний ресурс]. - Режим доступу: https://enovosty.com/uk/news_economy-ukr/full/514-cenovye-kacheli-chto-i-kak-podorozhaet-v-aprele

4.Вільний Ринок. // [Електронний ресурс]. - Режим доступу:https://www.prostir.ua/?news=vidkrytyj-rynok-startuje-onlajn-projekt-za-pidtrymky-amu

5. Врожайний карантин: як фермери пережили період обмежень. // [Електронний ресурс] - Режим доступу: https://mind.ua/publications/20211682-vrozhajnij-karantin-yak-fermeri-perezhili-period-obmezhen

Бз

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ІВАНА ПУЛЮЯ

М А Т Е Р І А Л И

VIII НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

«ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ»

9–10 грудня 2020 року

ТЕРНОПІЛЬ
2020

УДК 004:02
А.М. Слободяник
(Тернопільський національний технічний університет імені Івана Пулюя)

ВЕБ-ПЛАТФОРМА ДЛЯ ВИРІШЕННЯ ПРОБЛЕМ ІЗ РЕАЛІЗАЦІЄЮ СІЛЬСЬКОГОСПОДАРСЬКОЇ ПРОДУКЦІЇ

UDC 004:02
A. Slobodyanyk

WEB PLATFORM FOR SOLVING PROBLEMS WITH SALE OF AGRICULTURAL PRODUCTS

Із появою коронавірусу, карантинні обмеження стали випробуванням на стійкість і міцність для всієї економіки країни, і сфери агропромислового комплексу в тому числі.

Це дозволило проблемам краще проявитися і це привернуло увагу до моментів які слід було вдосконалити: порушення логістики, паніка серед покупців і стрибки цін [1].

Тому як один із способів щоб вирішити або спростити вирішення цих проблем пропонується веб-платформа Market Place.

Market Place – це місце де люди і організації сфери агропромислового комплексу зможуть публікувати оголошення, тим самим значно вдосконалять і прискорять процес реалізації сільськогосподарської продукції. У даному веб-сервісі, на відміну від сучасних рішень, основною ідеєю є розділити оголошення на категорії: “Куплю” і “Продам”. Саме таке розділення дозволить покрити весь ринок торгівлі і оголошень. І значно прискорить сам процес взаємодії компаній, людей і організацій у даній сфері.

До прикладу, фермер, зареєструвавшись у Market Place сервісі, зможе створити оголошення на продаж картоплі чи моркви яку вирощує, і згодом обрати серед тих хто відповів на оголошення, найкращого покупця котрий запропонував найвигіднішу ціну. Також може створити оголошення на купівлю, і йому як фермеру потрібні добрива, створивши “Куплю” оголошення для добрив, це зекономить багато часу не шукаючи людей, номерів телефонів і оголошень у газетах не тративши свій дорогоцінний час і зайнятися своїми справами пов’язаними з фермою, або з сім’єю, що теж дуже важливо. Через певний період часу і зайшовши у веб-сервіс зможе обрати серед запропонованих ті добрива, які потрібно і по найкращій ціні.

Для створення даної веб-платформи буде використовування стек сучасних і на сьогоднішній день найпопулярніших технологій:

1. ASP.NET Core MVC із .NET Core 3.1 (2019 р.) [2] із використанням Razor сторінок [3]. Це вільне та відкрите програмне забезпечення для реалізації веб застосунків, розроблена компанією Microsoft;
2. HTML, CSS, JavaScript і бібліотеки JQuery, Bootstrap [4]. Даний перелік технологій із Razor сторінками буде забезпечувати реалізацію UI (User Interface), тобто зовнішній вигляд веб-сторінок сервісу.

Література.

1. Вплив COVID-19 та карантинних обмежень на економіку України. // [Електронний ресурс]. - Режим доступу: https://surl.li/gtnp.
2. Overview of ASP.NET Core MVC // [Електронний ресурс]. - Режим доступу: https://docs.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-5.0.
3. Introduction to Razor Pages in ASP.NET Core // [Електронний ресурс]. - Режим доступу:https://docs.microsoft.com/en-us/aspnet/core/razor-pages/?view=aspnetcore-5.0&tabs=visual-studio.
4. Bootstrap // [Електронний ресурс] – Режим доступу: https://getbootstrap.com.