МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет
імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри інтелектуальних
інформаційних систем, д.т.н., проф.
_____ Ю.П. Кондратенко
«____»_____2022 року

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

ІНФОРМАЦІЙНА СИСТЕМА МАГАЗИНУ АВТОЗАПЧАСТИН СПОРТИВНИХ АВТОМОБІЛІВ

Спеціальність 122 «Комп'ютерні науки»

122 – БКР – 401.21810301

Виконав студент 4-го курсу, групи 401
_____І. В. Барбулат
«___» червня 2022 р.

Керівник: канд. пед. наук, доцент
_____Н. М. Болюбаш
«___» червня 2022 р.

Миколаїв – 2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет ім. Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

Рівень вищої освіти **бакалавр**
Спеціальність **122 «Комп'ютерні науки»**
(шифр і назва)
Галузь знань **12 «Інформаційні технології»**
(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних
інформаційних систем, д-р техн. наук, проф.
_____ Ю. П. Кондратенко
«____» _____20__ р.

З А В Д А Н Н Я

на виконання кваліфікаційної роботи

Видано студенту групи 401 факультету комп'ютерних наук Барбулату Іллі Володимировичу.

1.Тема кваліфікаційної роботи «Інформаційна система магазину автозапчастин спортивних автомобілів».

Керівник роботи Болюбаш Надія Миколаївна, к. пед. наук, доцент.

Затв. наказом Ректора ЧНУ ім. Петра Могили від «__» ____ 202_ р. № _____

2. Строк представлення кваліфікаційної роботи студентом «__» _____ 2022 р.

3. Вхідні (початкові) дані до роботи: діяльність ринку спортивних автомобілів в Україні, сукупність характеристик спортивних автомобілів.

Очікуваний результат: Інформаційна система магазину автозапчастин спортивних автомобілів із вбудованою системою рекомендацій.

4. Перелік питань, що підлягають розробці (зміст пояснювальної записки):

— аналіз сучасного стану ринку автозапчастин спортивних автомобілів та теоретичних засад підтримки діяльності інтернет-сервісів для їх продажу;

— обґрунтування вибору технологій та засобів розробки інформаційної системи автомагазину;
— розробка та здійснення програмної реалізації інформаційної системи магазину автозапчастин спортивних автомобілів.

5. Перелік графічного матеріалу: презентація, 40 рисунків та 23 додатків.

6. Завдання до спеціальної частини: “Санітарно-гігієнічні вимоги приміщення для роботи з комп’ютерною технікою.”

7. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис
Спеціальна частина з охорони праці	старший викладач, Макарова О.В.	

Керівник роботи __канд. пед. наук, доц. Болюбаш Н. М._____
(наук. ступінь, вчене звання, прізвище та ініціали)

_____
(підпис)

Завдання прийнято до виконання ___Барбулат І. В._____
(прізвище та ініціали)

_____
(підпис)

Дата видачі завдання «_____» _____ 20__ р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема: «Інформаційна система магазину автозапчастин спортивних автомобілів»

№	Найменування роботи	Початок	Закінчення	Примітки
1	Подання заяви на затвердження теми та керівника БКР	26.10.2021	30.10.2021	Виконано
2	Отримання завдання на виконання БКР	25.11.2021	25.11.2021	Виконано
3	Складання календарного плану роботи на весь період виконання БКР	26.112021	10.12.2021	Виконано
4	Огляд літературних джерел, аналіз предметної області, існуючих аналогів у сфері продажу автозапчастин	11.12.2021	31.12.2021	Виконано
5	Створення дизайну, проєктування та програмна реалізація застосунку	1.01.2022	1.04.2022	Виконано
6	Тестування системи	1.04.2022	15.04.2022	Виконано
7	Робота над розділами пояснювальної записки БКР	16.04.2022	15.05.2022	Виконано
8	Розробка спеціальної частини з охорони праці	16.05.2022	22.05.2022	Виконано
9	Проходження переддипломної практики, збір та аналіз матеріалів, остаточне оформлення розділів БКР	26.05.2022	05.06.2022	Виконано
10	Попередній захист БКР	30.05.2022	30.05.2022	Виконано
11	Обговорення отриманих результатів з керівником, доробка та остаточне оформлення БКР	1.06.2022	16.05.2022	Виконано
12	Подання БКР рецензенту	16.05.2022	18.05.2022	Виконано
13	Створення слайдів для захисту та написання доповіді	18.06.2022	20.06.2022	Виконано
14	Подання БКР, її електронної копії та інших документів до захисту	20.06.2022	22.06.2022	Виконано
15	Захист БКР перед ЕК	27.06.2022		Виконано

Розробив студент_____Барбулат І. В._____ _____
(прізвище та ініціали) *(підпис)*

Керівник роботи __канд.пед.н., доцент Болюбаш Н.М. _____
(наук. ступінь, вчене звання, прізвище та ініціали) *(підпис)*

« __ » _____ 202_ р.

АНОТАЦІЯ
до бакалаврської роботи

Тема: «Інформаційна система магазину автозапчастин спортивних автомобілів»

Студент: Барбулат Ілля Володимирович
Керівник: канд.пед.н, доцент Болюбаш Надія Миколаївна

Бакалаврська кваліфікаційна робота присвячена проектуванню, розробці, програмній реалізації та впровадженню інформаційної системи магазину автозапчастин спортивних автомобілів.

Об'єкт дослідження – продаж автозапчастин в мережі Інтернет.

Предмет дослідження – web-орієнтовані програмні засоби для магазину автозапчастин спортивних автомобілів.

Метою роботи є поліпшення продажу автозапчастин шляхом розробки і впровадження інформаційної системи для підтримки діяльності магазину автозапчастин спортивних автомобілів із вбудованою системою рекомендацій.

Бакалаврська кваліфікаційна робота робота складається з фахової частини і спеціальної частини з охорони праці. Пояснювальна записка дипломної роботи складається зі вступу, трьох розділів, висновків та додатків.

У першому розділі здійснено аналіз сучасного ринку автозапчастин спортивних автомобілів в Україні та Інтернет-сервісів для підтримки їх продажу. У другому розділі розглядаються технології та засоби розробки інформаційної системи. У третьому розділі описано проектування та програмну реалізацію інформаційної системи магазину автозапчастин спортивних автомобілів.

У четвертому розділі розкрито питання спеціальної частини з охорони праці.

Дипломна робота містить 87 сторінок, 40 рисунків, 23 джерела, 7 додатків.

ABSTRACT
for bachelor's work

Subject: «Store shop sports car parts information system»

Student: Barbulat Illia Volodymyrovych

Leader: Ph.D., associate professor Bolyubash Nadiya Mikolaivna

Thesis is devoted to the design, development, software implementation and implementation of the information system of the auto parts store of sports cars.

The object of research is the sale of auto parts on the Internet.

The subject of the research is a Web-based software for sports car parts store.

The purpose of the thesis is to improve the sale of auto parts by developing and implementing an information system to support the activities of the auto parts store of sports cars with a built-in system of recommendations.

Thesis consists of a professional part and a special part on labor protection. Explanatory note of the thesis consists of an introduction, three chapters, conclusions and appendix.

The first section analyzes the modern market of auto parts for sports cars in Ukraine and Internet services to support their sales. The second section discusses the means of developing an information system. The third section describes the design and software implementation of the information system of the auto parts store of sports cars.

The fourth section reveals the issue of a special part of labor protection.

Thesis contains 87 pages (without appendix), 40 figures, 23 sources, 7 supplements.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

БД – база даних

ГН – гігієнічні норми

ЕПТ – Електрично – променева трубка

КНО – Коефіцієнт природної освітленості

КТ – комп’ютерна техніка

ОС – операційна система

ПЗ – програмне забезпечення

AOT – Ahead-of-Time

CSS – Cascading Style Sheets

DLL – Dynamic Link Library

FDD – Feature Driven Development

HTML – Hyper Text Markup Language

MVC – Model View Control

ORM – Object-Relational Mapping

Пояснювальна записка

до кваліфікаційної роботи

на тему:

ІНФОРМАЦІЙНА СИСТЕМА МАГАЗИНУ АВТОЗАПЧАСТИН СПОРТИВНИХ АВТОМОБІЛІВ

Спеціальність 122 «Комп’ютерні науки»

122 – БКР – 401.21810301

Виконав студент 4-го курсу, групи 401
_____І. В. Барбулат
«___» червня 2022 р.

Керівник: канд. пед. наук, доцент
_____Н. М. Болюбаш
«___» червня 2022 р.

Миколаїв – 2022

ВСТУП

Актуальність. Інформатизація суспільства на сучасному етапі характеризується розширенням надання послуг через мережу Інтернет у сфері продажу автозапчастин до автомобілів. Набуває поширення взаємодія з покупцями і клієнтами автомагазинів з використанням мережевих сервісів.

В Україні існує багато інтернет-площадок, задіяних на автомобільному ринку країни: OLX, Ria.com,Prom.ua, Rozetka,Aukro. Вони мають широкий набір послу для перегляду, відбору по здійснення покупки необхідних товарів в Інтернет. Однак надання рекомендацій покупцям із врахуванням попередніх історій покупок реалізовано не у повній мірі, а спеціалізовані магазини із продажу автозапчастин спортивних автомобілів відсутні. У цих умовах перспективним напрямком поліпшення задоволення потреб власників спортивних автомобілів є розширення магазинів з продажу автозапчастин до них та впровадження у їх діяльність інформаційної системи для взаємодії з клієнтами та продажу через Інтернет із вбудованими алгоритмами надання рекомендацій.

Це обумовило **мету дослідження**, яка полягає у поліпшенні продажу автозапчастин шляхом розробки і впровадження інформаційної системи для підтримки діяльності магазину автозапчастин спортивних автомобілів із вбудованою системою рекомендацій.

Відповідно до поставленої мети було сформульовано **завдання дослідження**.

1. Здійснити аналіз сучасного стану ринку автозапчастин спортивних автомобілів в Україні та теоретичних засад підтримки діяльності інтернет-сервісів для їх продажу.
2. Обґрунтувати вибір технологій та засобів розробки інформаційної системи автомагазину.
3. Розробити та здійснити програмну реалізацію інформаційної системи магазину автозапчастин спортивних автомобілів.

Об'єкт дослідження – продаж автозапчастин в мережі Інтернет.

Предмет дослідження – web-орієнтовані програмні засоби для магазину автозапчастин спортивних автомобілів.

Методологічною основою дослідження є загальнонаукові та аналітичні методи, які дозволили вивчити предмет та об'єкт дослідження, дослідити напрями та шляхи оптимізації продажу автозапчастин.

Практичне значення отриманих результатів полягає в тому, що використання розробленої інформаційної системи дозволить підвищити якість роботи магазину з продажу автозапчастин спортивних автомобілів.

Структура дипломної роботи. Відповідно до мети, завдань і предмета дослідження, дипломна робота містить основну та спеціальну частини. Основна частина роботи складається із вступу, трьох розділів, висновку, списку використаних джерел та 7 додатків. Загальний обсяг роботи – 87 сторінок, із них основного тексту основної частини – 57 сторінок, спеціальної – 16 сторінок. Кількість використаних джерел – 23.

1 АНАЛІЗ РИНКУ ПРОДАЖ АВТОЗАПЧАСТИН СПОРТИВНИХ АВТОМОБІЛІВ В УКРАЇНІ

1.1 Особливості сучасного стану ринку продаж автозапчастин спортивних автомобілів

Однією з тенденцій сучасного ринку автозапчастин для звичайних та спортивних авто полягає у перенесені торгових майданчиків в інтернет. Усе більше українців віддають перевагу інтернет ресурсам ніж спеціалізованим точкам продажу. Це пов’язано більш низькими цінами, які пропонують онлайн-продавці.

Відсутність фізичних магазинів та штату співробітників дозволяє знизити витрати, що у свою чергу знижує вартість продукту. Теж саме стосується малих підприємств які самі виготовляють деталі, це дозволяє їм сконцентрувати фінансові ресурси на покращення якості та об’ємів готової продукції.

Ще одною причиною, через яку покупці відходять від стандартної моделі покупок запчастин та переходять до онлайн, полягає у їх доступності. Зазвичай у звичайних магазинах асортимент обмежений тільки популярними марками. Якщо у людини не дуже популярна марка авто або автомобіль спортивний, то потрібно буде довго шукати запчастину, довго за нею їхати або довго чекати доставки. Також ніхто не відміняє факту що деталь може бути не оригінальною чи з дефектом.

Аналізуючи статистику запитів у мережі інтернет можна зробити висновок – кількість запитів про покупку запчастин починаючи з 2017 року росте і навіть під час пандемії цей процес не зазнав значних змін. Самий пік припав на лютий 2019 року (рис. 1.1).

Для більш глибокого розуміння стану ринку автозапчастин в Україні буде використано два аналітичні ресурси. Перший з них – Ringostar, який є надійним джерелом інформації та користується попитом серед бізнесменів. Цей ресурс

аналізує відвідувачів сайтів та їх дзвінками. За наявною статистикою у період з листопада 2019 року по липень 2020 року, у перечень зростаючих інтересів покупців потрапили автозапчастини.

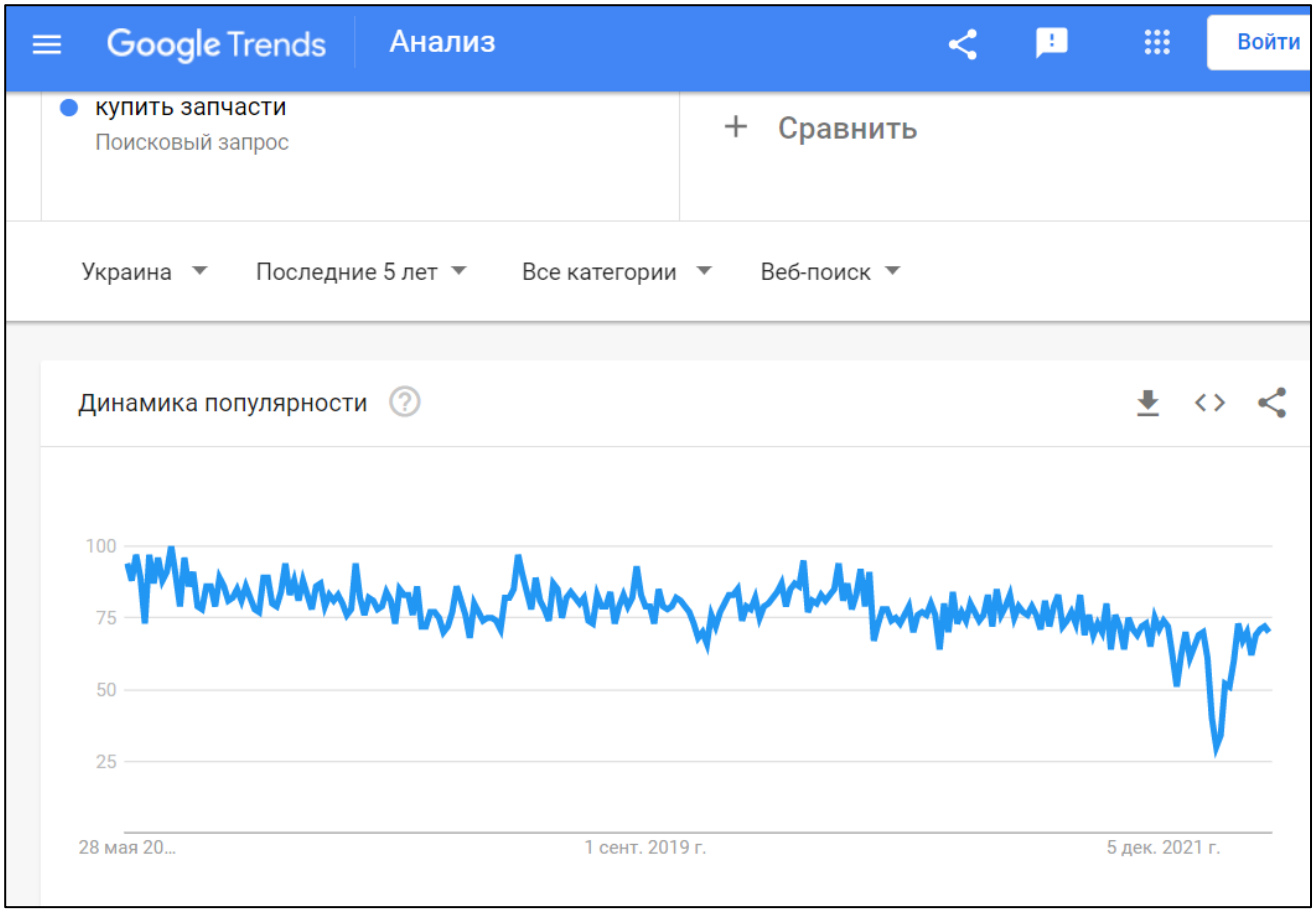


Рисунок 1.1 – Динаміка запитів автозапчастин

Аналіз по областям ми бачимо наступне: по кількість запитів на першому місці знаходиться Полтавська область, а на останньому Львівська область (див. рис. 1.2). Треба враховувати факт, що ці дані можуть змінюватися кожний день.

Необхідно відмітити, що тематика автозапчастин у період глобального карантину не знизилася, а навіть виросла (див. рис. 1.3).

Рисунок 1.2 – Динаміка запитів автозапчастин по областям України

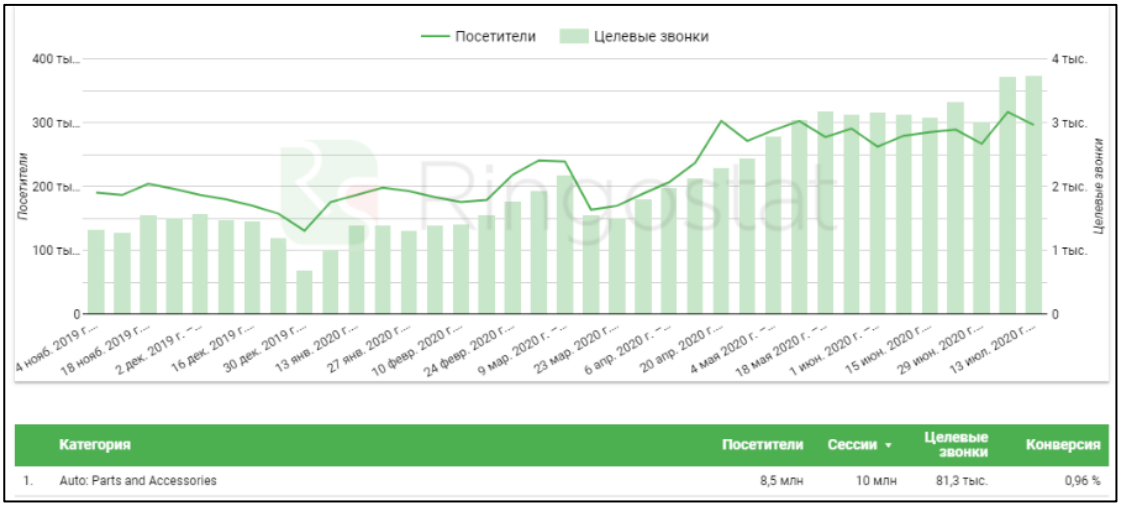


Рисунок 1.3 – Статистика Ringostar

Інший ресурс – SimilarWeb, за допомогою якого продивляються статистику трафіку, через які джерела інформації люди переходять на сайт. У результаті аналізу отримаємо відсоток кожного джерела трафіку (див. рис. 1.4):

1. Органічний трафік з пошукових систем (86.88%).
2. Прямі заходи (11.12%).
3. Рекламні оголошення (0.03%).
4. Трафік з соціальних мереж (1.28%).
5. Переходи з інших сайтів (0.26%).

6. Повідомлення на пошту (0.44%).

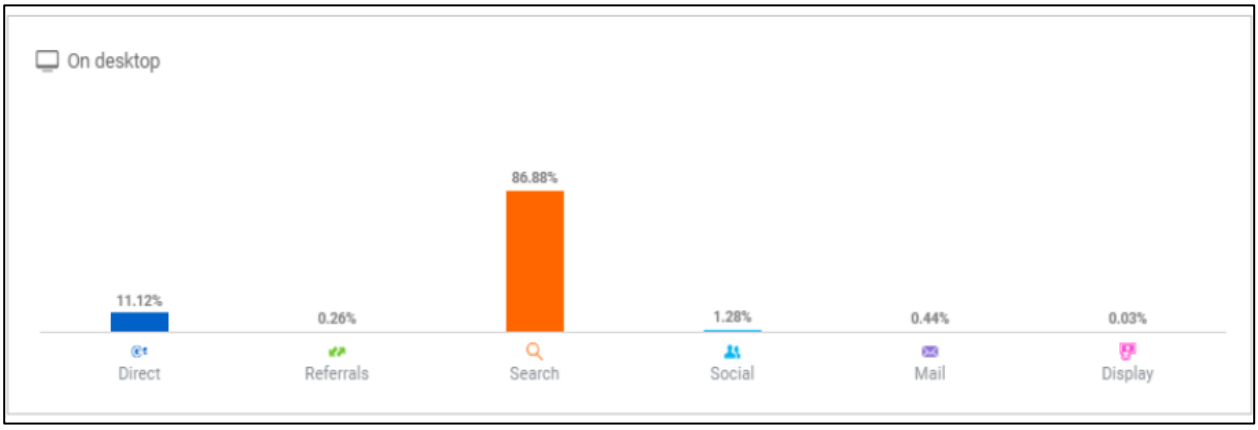


Рисунок 1.4 – Статистика SimilarWeb

1.2 Процес прийняття рішень покупця в інтернеті

Процес прийняття рішення дуже схожий на той, чи є споживач офлайн або онлайн. Але одним з основних відмінностей є торгове середовище та маркетингова комунікація. Згідно традиційної моделі прийняття рішень споживача, рішення про покупку споживача зазвичай починається з усвідомлення необхідності, потім пошуку інформації, альтернативних оцінок, прийняття рішення про покупку.

Що стосується онлайн-спілкування, коли клієнти бачать рекламу на банерах або онлайн-рекламу, то ці рекламні оголошення привертають увагу клієнтів і стимулюють їх зацікавленість до конкретних продуктів. Якщо у них недостатньо інформації, вони будуть шукати онлайн-канали, наприклад, онлайн-каталоги, веб-сайти або пошукові системи. Коли в клієнтів буде достатньо інформації, вони повинні будуть порівняти ці варіанти продуктів або послуг. На етапі пошуку вони можуть шукати огляди продуктів або коментарі клієнтів. Вони дізнаються, який бренд або компанія пропонує їм найкращою відповідність їх очікуванням. На цьому етапі добре організована структура веб-сайту і привабливий дизайн -

важливі речі, щоб переконати споживачів зацікавитися покупкою товарів і послуг. Більш того, характер джерел інформації може впливати на поведінку покупців[1].

Найбільш корисною характеристикою Інтернету є те, що він підтримує етап попередньої покупки, оскільки він допомагає клієнтам порівнювати різні варіанти. На етапі закупівель асортимент продукції, послуги з продажу та якість інформації, мабуть, є найбільш важливим моментом, щоб допомогти споживачам вирішити, який продукт вони повинні вибрати, чи того, що продавець повинен купити. Постіндустріальне поведінка стане більш важливим після покупки в Інтернеті. У споживачів іноді виникають проблеми або проблеми з продуктом, або вони можуть захотіти змінити або повернути продукт, який вони купили. Таким чином, послуги з поверненню та обміну стають більш важливими на цьому етапі.

1.3 Мережеві сервіси підтримки діяльності магазинів продажу автозапчастин

На даний момент на українському ринку спортивних автозапчастин немає спеціалізованих сайтів чи магазинів. Звичайно є магазини, які продають автозапчастини, але вони в більшості сфокусовані на повсякденні авто.

До можливих аналогів можна віднести різноманітні торгові майданчики, такі як: OLX, Ria.com,Prom.ua, Rozetka,Aukro. На цих майданчиках одні люди продають товари, інші їх купляють. Кожен із цих майданчиків надає змогу викласти товар на продаж та чекати поки покупець не зацікавиться товаром та не купить його. Але усі перераховані майданчики займаються продажом різноманітних товарів, починаючи від побутових дрібниць закінчуючи габаритною технікою.

Тому до аналогів з продажу автозапчастин спортивних авто можна долучити такі торгові майданчики:

- сайт оголошень та торговий майданчик OLX;
- торговий майданчик Ria.com(rapchasti.ria.com).

Огляд OLX. На даний момент OLX є найбільшим торговим майданчиком України. Не дивлячись на те, що насправді це сайт об’яв, все ж таки його можна назвати торговим майданчиком, оскільки цей ресурс має функцію доставки. У цьому випадку покупець має змогу одразу сплатити товар, як на торговому майданчику.

На сайті можна опублікувати оголошення будь-який користувач, після SMS-верифікації. Товари можна виставляти, як на продаж, так і на обмін або віддати безкоштовно.

Оголошення можна публікувати безкоштовно до 30 днів, але в обмеженій кількості. У більшості рубрик існує ліміт на публікацію безкоштовних оголошень. В принципі, за цим можна не стежити, оскільки при досягненні або перевищенні ліміту ви побачите повідомлення. Тому для професійних користувачів передбачені платні «пакети розміщень». Вони дають змогу розміщувати більше оголошень. Крім цього, за їх допомогою можна створити "Бізнес-сторінку" з розширеними можливостями. Доставку та оплату товару можна здійснювати на власний розсуд або за домовленістю з покупцем. Проте, з деяких пір, як писали вище, на сайті з'явилася можливість продавати товари з OLX доставкою. У цьому випадку, покупець відразу оплачує товар, але гроші на ваш рахунок надійдуть після того, як покупець отримає товар. Проте, послуга «OLX Доставка» доступна не в усіх рубриках та розділах. І має обмеження за вагою до 30 кг та ціною від 30 до 24 799 грн.

Технічні особливості. Автоматичного переставлення оголошень немає. Необхідно кожне оголошення переставляти у ручному режимі. Для просування оголошення є платні послуги. Система відгуків відсутня. А також немає грошової компенсації для користувачів, які постраждали від шахрайства. Тобто. немає системи «захист споживачів». OLX рекомендує надавати перевагу оголошенням зі значком «olx доставка». Проте існує кнопка скарги. За допомогою якої можна подати скаргу. Адміністрація розглядає кожну скаргу та вживає заходів, відповідно до Правил та Умов сервісу.

Огляд RIA.com. Це безкоштовна дошка оголошень. На сайті понад 2,3 млн. пропозицій та більш ніж 3 млн. відвідувань на місяць. Проте, з недавніх пір, RIA розширило можливості свого сайту, перетворивши його на торговий майданчик. І тепер, окрім оголошень, можна відкривати власний інтернет-магазин на платформі сайту. Найпопулярнішими напрямками цього майданчика є такі категорії: Продаж автомобілів, Продаж нерухомості та Автотовари.

Незважаючи на те, що оголошення можна давати безкоштовно, все ж таки їх кількість обмежена. Усього можна розміщувати до 49 безкоштовних оголошень на місяць. У різних рубриках допускається різна кількість, але загальна кількість не повинна перевищувати 49 оголошень. Оголошення можна розміщувати від 7 до 60 днів. Для розміщення оголошення на сайті надається три варіанти на вибір. Перший варіант безкоштовний, це "Звичайне оголошення". Два інших варіанти платні, це "ТОП оголошення" та "Максимальний ТОП". З їхньою допомогою можна просувати своє оголошення в топ. З деяких пір на сайті з'явився новий функціонал "Автоматична публікація". Ця функція доступна для користувачів, які мають активний платний пакет оголошень або інтернет-магазин.

У випадку з магазинами, так само передбачено три тарифні плани на вибір. Це «Базовий», «Активний» та «Топовий». Кожен з яких можна обрати на 1, 3, 6 та 12 місяців. Ціни залежать як від вибраного пакета, так і від терміну дії. Ціни коливаються від 280 грн. за місяць за базовий пакет до 6375 грн. на рік за топовий пакет. У всіх магазинах є можливість автоматично завантажити всі товари та послуги у великій кількості. Це можна зробити за допомогою імпорту YML, Excel, файлів XML. Є система відгуків. Для просування товарів та послуг у магазині ви отримуєте рівні ТОП та оновлення дат публікацій. Для автозапчастин передбачено платне комплексне просування товару для більш ніж 8 млн. покупців. Для цього розроблено три платні пакети: Старт, Плюс та Преміум. Кожен з яких можна замовити на 1, 3, 6 та 12 місяців.

Проаналізувавши наявні аналоги, можна зробити висновок, що прямих аналогів немає. Бо розглянуті ресурси є торговими майданчиками, які не

сфокусовані тільки на спортивних автозапчастинах. Їх каталог охоплює великий перечень товарів, що не дає їм змоги сконцентруватися на одному виді товарів та врахувати специфіку продажу спортивних автозапчастин.

1.4 Алгоритм Apriori

При аналізі інформації, яка збірається у базі даних магазину автозапчастин для поліпшення маркетингу та взаємодії з потенційними покупцями доцільно застосовувати асоціативні правила. Їх використання дає можливість підвищити рівень задоволення потреб клієнтів автомагазину та рівень доходу магазину за рахунок пропозиції тих товарів, які часто купують разом[5].

Навчання на асоціативних правилах (або скорочено ARL) являє собою, з одного боку, простий, з іншого – досить часто застосований у нашому житті метод пошуку взаємозв'язків (асоціацій) в різних наборах даних. Найчастіше всього цей алгоритм використовують для виявлення типової поведінки покупців [6].

У загальному вигляді ARL можна описати : Хто купив x, також купить y. Основою виступає аналіз транзакцій різних покупців, всередині кожної з яких є власний унікальний itemset з набору items. За допомогою цього алгоритму знаходяться ті самі правила кожної з транзакцій, які потім упорядковуються.

Наприклад в нас є певний датасет (або колекція) $D = d_0 \ldots d_j$, де d – унікальна транзакція –items (наприклад чек з магазину).

Всередині кожногої транзакції d існує набір елементів з деякої множини I (items, це може бути набір товарів), в ідеальному випадку він представлений у бінарному вигляді:

d1 = [{Хліб: 1}, {Вода: 0}, {Кола: 1}, {...}], d2 = [{Хліб: 0}, {Вода: 1}, {Кола: 1}, {...}].

Прийнято кожен itemset описувати через кількість нульових значень, наприклад: [{Хліб: 1}, {Вода: 0}, {Кола: 1}] є 2-itemset.

Таким чином, датасет представлено у виглояді матриці зі значеннями {1,0} (юінарний датасет). Існують і інші види запису – вертикальний датасет, який показує для кожного окремого item вектор транзакцій, де він присутній, і транзакційний датасет: приблизно як у касовому чеку.

Асоціативним правилом називаюь імплікацію $X \to Y$, де X та Y є набори елементів множини I, серед яких немає однакових елементів. Фактивно, це означає, що якщо у транзакції буде присутнім елемент X, то з великою ймовірністю там буде також присутнім і елемент Y.

X називають умовою, а Y наслідком асоціативного правила.

Для знаходження таких правил розраховують їх оцінки [5]. До основних оцінок відносять підтримку та достовірність.

Почнемо з такого поняття як – підтримка (Support).

Підтримка асоціативного правила є відношенням кількості транзакцій, які містять умову та наслідок, до загальної кількоісті транзакцій:

$$S(X \to Y) = P(X \cup Y) \tag{1.1}$$

де $P(X \cup Y)$ – є ймовірністю появи у транзакції умови та наслідку одночасно.

Наступне ключове поняття – достовірність (confidence).

Достовірність асоціативного правила $C(X \to Y)$ визначається як відношення кількості транзакцій, що містять умову і наслідок, до кількості транзакцій, що містять тільки умову. Це показник того, як часто наше правило спрацьовує для всього датасета.

Наведемо приклад: ми хочемо порахувати confidence для правила «хто купує хліб, той купує і підгузки». Для цього спочатку порахуємо, який support у правила «купує хліб», потім порахуємо support у правила «купує хліб і підгузки», і просто поділимо одне на інше. Тобто ми порахуємо в скількох випадках (транзакціях) спрацьовує правило «купив хліб» supp (X), «купив підгузки і хліб»

Не менш важливим є поняття ліфта (lift).

Ліфт асоціативного правила $X \to Y$ є відношенням частоти появи умови в транзакціях, які містять і умову і наслідок, до частоти появи наслідку у цілому:

$$L(X \to Y) = \frac{C(X \to Y)}{P(Y)} \tag{1.2}$$

Простими словами, lift – це відношення «залежності» items до їх «незалежності». Наприклад, ми хочемо зрозуміти залежність покупки хліба і покупки підгузків. Для цього вважаємо support правила «купив хліб і підгузки» і ділимо його на добуток правил «купив хліб» і «купив підгузки». У разі, якщо lift = 1, ми говоримо, що items незалежні і правил спільної покупки тут немає. Якщо ж lift> 1, то величина, на яку lift більше цієї самої одиниці, і покаже нам «силу» правила. Чим більше одиниці, тим краще. Якщо lift <1, то це покаже, що правило підстави $ x_2 $ негативно впливає на правило $ x_1 $. По-іншому lift можна визначити як відношення confidence до expected confidence, тобто відношення достовірності правила, коли обидва (або більше) елемента купуються разом до достовірності правила, коли один з елементів купувався (неважливо, з другим або без).

Під час роботи інтернет-площадо із продажу товарів кількість транзакцій є дуже великою, і для пошуку асоціативних правил доцільно застосовувати алгоритми, які суттєво зменшують простір пошуку. Найбільш поширеним із низ є алгоритм Apriori, який було використано у даній роботі при розробці проєкту.

Пошук асоціативних правил за алгоритмом Apriori передбачає:

1) задання порогових мінімальних значень підтримки та достовірності;
2) пошуку усіх можливих 1, 2-х та більше елементних наборів, які мають підтримку, більшу за задане порогове значення;
3) об’єднання всіх знайдених у попередньому пункті наборів елементів у одну мнижину, та формування усіх можливих асоціативних правил;
4) Розрахунок достовірності знайдених асоціаціативних правил та відкидання тих із них, доствірність яких менша за наперед задане порогову значення.

1.5 Постановка задачі дослідження

Розробка інформаційної системи магазину автозапчастин спортивних автомобілів із використанням алгоритмі надання рекомендацій є актуальною проблемою розвитку мережевих сервісів підтримки його діяльності. Здійснений аналізу стану ринку спортивних автозапчастин дозволив висунути наступні вимоги до інформаційної системи автомагазину.

1. Серед користувачів доцільно виділити покупців та продавців.

2. Користувач має змогу: здійснювати пошук потрібного товару серед наявних у магазині автозапчастин; додавати до спеціального списку потенційно потрібний товар; формувати кошик замовлень та замовляти товар.

3. Продавець магазину має змогу виставляти наявний товар, отримувати статистику по популярності товарів, отримувати замовлення товарів та здійснювати їх продаж.

4. Для роботи інформаційної системи магазину повинна бути спроектована база даних, яка буде містити інформацію про наявні автозапчастини, відвідувачів сайту, сформовані ними замовлення та історії пошуку необхідних деталей.

5. Створення лаконічного та простого користувацького інтерфейсу.

Об’єкт дослідження – продаж автозапчастин в мережі Інтернет.

Предмет дослідження – web-орієнтовані програмні засоби для магазину автозапчастин спортивних автомобілів.

Метою даної роботи є поліпшення продажу автозапчастин шляхом розробки і впровадження інформаційної системи для підтримки діяльності магазину автозапчастин спортивних автомобілів із вбудованою системою рекомендацій.

Для досягнення поставленої мети було поставлено такі **завдання**.

1. Здійснити аналіз сучасного стану ринку автозапчастин спортивних автомобілів в Україні та теоретичних засад підтримки діяльності інтернет-сервісів для їх продажу.
2. Обґрунтувати вибір технологій та засобів розробки інформаційної системи автомагазину.
3. Розробити та здійснити програмну реалізацію інформаційної системи магазину автозапчастин спортивних автомобілів.

Висновки до розділу 1

Здійснений аналіз показав, що сучасний стан інформатизації суспільства супроводжується перенесенням реалізації послуг із продажу товарів у мережу Інтернет. У таких умовах існує потреба у впровадженні інформаційних систем підтримки діяльності магазинів із продажу товарів та автозапчастин зокрема. Наявність таких систем сприяє підвищенню ефективності роботи працівників магазину по роботі з клієнтами, підвищує продаж товару та просування послуг у мережі Інтернет. Виявлено, що сьогодні існує багато кросплатформених сервісів для підтримки діяльності автомагазинів, однак спеціалізовані магазини із продажу автозапчастин спортивних автомобілів відсутні. Такі автозапчастини продають також на торгових майданчиках, які сфокусовані не тільки спортивних автозапчастинах. Це обумовлює необхідність створення спеціалізованої інформаційної системи магазину автозапчастин спортивних автомобілів.

Виявлено, що роботу з клієнтами магазину та підвищення продуктивності роботи магазину можна поліпшити, надаючи рекомендації покупцями, які уже зробили певний вибір для подальших покупок у магазині. Такий механізм можна реалізувати шляхом пошуку асоціативних правил з використанням алгоритму Apriori.

2 ТЕХНОЛОГІЇ ТА ЗАСОБИ РОЗРОБКИ ІНФОРМАЦІЙНОЇ СИСТЕМИ МАГАЗИНУ АВТОЗАПЧАСТИН

2.1 Мова програмування C#

Для реалізації поставленої задачі була обрана мова програмування C# - це мова програмування, яка поєднує об'єктно-орієнтовані та аспектно-орієнтовані концепції. Розроблена у 1998 – 2001 роках групою інженерів під керівництвом Андерса Хейлсберга у компанії Microsoft як основну мову для розробки застосунків платформи .NET. Компілятор C# входить до стандартної установки самої .NET, тому програми на ньому можна створювати і компілювати навіть без інструментальних засобів на кшталт Visual Studio.

C# відноситься до сім'ї мов з C-подібним синтаксисом, їх синтаксис найбільш близький до C++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, навантаження операторів, покажчики на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML. Перейнявши багато від своїх попередників - мов C++, Java, Delphi, Модула і Smalltalk - C#, спираючись на практику їх використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем: так, C# не підтримує множинне наслідування класів (на відміну від C++).

C# розроблявся як мова програмування прикладного рівня для CLR і, як такої, залежить, перш за все, від можливостей CLR. Це стосується насамперед системи типів C#, яка відображає FCL. Присутність або відсутність тих чи інших виразних особливостей мови диктується тим, чи конкретна мовна особливість може бути трансльована у відповідні конструкції CLR. Так, з розвитком CLR від версії 1.1 до 2.0 значно збагатився сам C#; подібної взаємодії слід очікувати і надалі. Однак ця закономірність була порушена з виходом C# 3.0, що є розширення мови, що не спираються на розширення платформи .NET. CLR надає C#, як і всім іншим .NET-орієнтованим мовам, багато можливостей, яких

позбавлені «класичні» мови програмування. Наприклад, складання сміття не реалізована в самому C#, а проводиться CLR для програм, написаних на C# точно так, як це робиться для програм на VB.NET, J# та ін.

2.2 Платформа .NET Core

Розробка .NET Core стала результатом роботи над різноманітними додатками. На кожному пристрої використовуються різні ОС, їх версії, конфігурації, драйвери та інше. Саме тому почалися пошуки способу уніфікувати середовище для запуску додатків. І однією з найуспішніших спроб у цій галузі можна назвати .NET, яка була розроблена понад 20 років тому компанією Microsoft.

Головна ідея її створення полягала в тому, щоб додати спеціальний програмний «прошарок», який розташовувався б між ОС і додатком. Саме на цей прошарок були покладені функції згладжування особливостей конкретної конфігурації ОС, тим самим спрощуючи запуск програми в середовищі. Модель .NET активно застосовується для розробки додатків таких видів:

- веб-API та мікрослужби;
- класичні та мобільні програми;
- web-додатки;
- служби Windows;
- консольні застосунки.

Переваги у використанні платформи полягають в тому, що файли коду і проекту виглядають ідентично і не залежать від типу програми, що розробляється.

Крім ОС Windows, платформа .NET Core також доступна в різних дистрибутивах Linux. Розробники публікують новий випуск практично для всіх дистрибутивів Linux, майже завжди в них є диспетчер пакетів для установки. Тому платформа стає все більш популярною у програмістів і поступово витісняє схожі рішення.

Починаючи з 2016 року, було випущено понад 10 версій доповнення. Перші з них називалися .NET Core, проте з 2019 року ця назва була скорочена до .NET.

Архітектура .NET Core базується на архітектурі .NET Framework. Проте є й кілька кардинальних відмінностей: платформа має можливість використання хмарних технологій, функціонал для кросплатформенності та модульності. Також у версії Core відбулося відділення середовища виконання бібліотеки.

Оскільки доповнення є модульним, кожен його компонент оновлюється через окремий менеджер пакетів. Це дозволяє оновлювати кожен модуль окремо. У результаті застосунок може працювати з окремими модулями та не залежати від оновлення всієї платформи.

Моделі розгортання. Однією з основних характеристик .NET є гнучке розгортання. Э можливість встановити платформу як частину програми, так і окремо. У разі використання FDD вдається зменшити пакет розгортання, мінімізувати використання пам'яті та дискового простору. Розгортання програм на платформі здійснюється в наступних режимах.

1. При розміщенні програми, що залежить від платформи, генеруються виконувані та двійкові файли, включаючи ПЗ та його залежності. Кінцевого користувача для запуску програми необхідно додатково запустити середовище виконання. Виконуваний файл залежить від платформи, а двійковий - DLL - є кросплатформенним.

2. Під час тестування автономної програми використовується файл, що містить середовище виконання, бібліотеки, а також саме ПЗ та його залежності. За рахунок цього користувачі можуть відкрити програму на пристрої, на якому не встановлювалося раніше середовище .NET. Автономні програми перебувають у прямому підпорядкуванні біля платформи, але за необхідності їх можна опублікувати як компіляції AOT.

Паралельно можна використовувати різні версії середовища для завантаження програми. Це дозволить запускати програми, що залежать від

платформи. Файли, що виконуються, генеруються для певної платформи, які позначаються спеціальним ідентифікатором середовища.

Кросплатформеність. На даний момент існує безліч різноманітних ОС, тому розробники зацікавлені у тому, щоб їх додатки поширювалися у різних операційних системах. І випадків, коли запустити програму, розроблену для іншої операційної системи, неможливо, стає дедалі більше. У цьому випадку для запуску потрібно «портувати» програму і лише потім намагатися встановити її в поточній ОС. У ряді випадків цього можна досягти простим перекомпілюванням вихідного тексту програми. Однак найчастіше це не допомагає, а це означає, що застосунок доведеться переписувати практично з нуля.

Однак Microsoft врахували цей момент під час розробки .NET Core. По суті платформа стала відігравати роль «шару, що стандартизує», який дозволив домогтися адаптації вихідного коду під середовище ОС. Це означає, що додаток для однієї ОС буде без проблем запускатися в інший. Такий підхід дозволив уникнути портування окремих програм, а портувати лише програмний прошарок. Для досягнення мети Microsoft вирішили портувати один із компонентів моделі .NET, внаслідок чого і з'явилася платформа .NET. Її функціональність практично повністю збігається з вихідною версією і в чомусь навіть стала набагато ширшою.

На сьогоднішній день .NET Core активно застосовується у трьох операційних системах: крім Windows, це ще Linux та macOS. Це дозволяє запускати нові програми у всіх ОС без додаткових змін. Саме ця особливість платформи називається кросплатформенністю.

У рамках крос-сумісності платформа включає і модульну інфраструктуру. Це дозволяє отримати доступ до пакетних даних та функцій. Завдяки цьому значно спрощується процес створення ПЗ, підвищується продуктивність та безпека процесів. Крім цього, модульна інфраструктура допомагає швидше оновлювати платформу, оскільки модулі випускаються та оновлюються окремо.

2.3 Фреймворк ASP.NET Core

ASP.NET Core – це новий загальнодоступний та кросплатформений фреймворк для створення сучасних програм, пов'язаних із підключенням до інтернету, таких як веб-програми, програми для інтернету речей та мобільних серверів [8]. Застосунки ASP.NET Core можуть працювати на .NET Core або на повній платформі .NET Framework. Цей фреймворк був спроектований таким чином, щоб забезпечити оптимізовану платформу розробки для програм, які переміщуються в хмару або локально виконуються. Він складається з модульних компонентів з мінімальним навантаженням, тому зберігається гнучкість при побудові індивідуальних рішень. Існує можливість розробляти та запускати кросплатформні ASP.NET Core додатки на Windows, Mac та Linux. Фреймворк ASP.NET Core доступний на GitHub.

ASP.NET Core має ряд архітектурних змін, які призводять до більш компактної та модульної структури. Фреймворк більше не ґрунтується на файлі System.Web.dll. Він заснований на наборі детальних та добре структурованих пакетів NuGet. Це дозволяє оптимізувати програму за допомогою пакетів NuGet, які вам необхідні. Переваги меншої площі поверхні програми включають: більш суворий захист, знижений рівень обслуговування, покращену продуктивність та зниження витрат у моделі «плати за те, що використовуєш».

2.4 Архітектура Model-View-Controller

Структура архітектури Model-View-Controller MVC поділяє додаток на три основні групи компонентів: модель, вигляд та контролер. Це дозволяє реалізувати принципи розподілу завдань. Відповідно до цієї структури запити користувачів надсилаються до контролера, який відповідає за роботу з моделлю для виконання дій користувачів та отримання результатів запитів. Контролер вибирає вигляди для відображення користувачеві з усіма необхідними даними моделі[10].

Вигляд та контролер залежать від моделі. Проте сама модель не залежить ні від контролера, ні від вигляду. Це одна з ключових переваг патерну mvc. Такий поділ дозволяє створювати та тестувати моделі незалежно від їх візуального представлення.

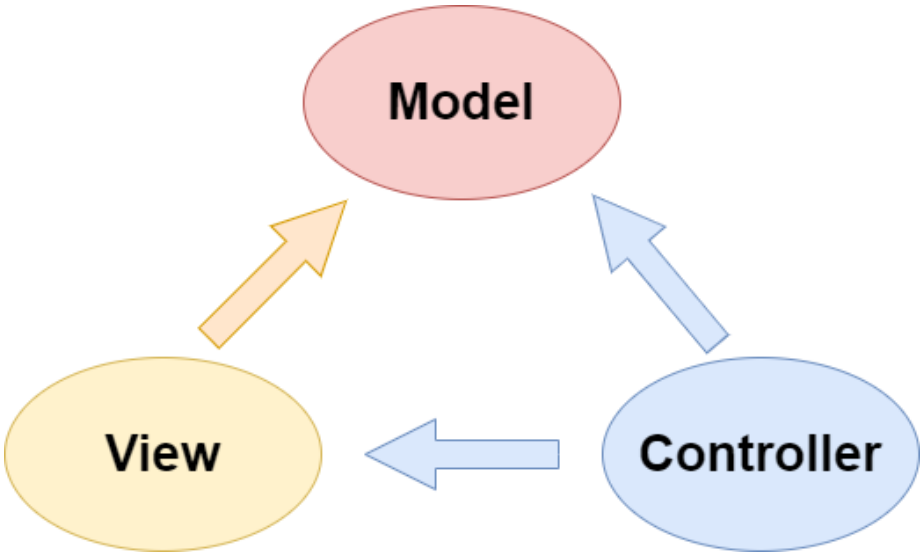


Рисунок 2.1 – Структура патерну MVC

Функції моделі. Модель у застосунку MVC представляє стан програми та бізнес-логіку або операцій, які мають виконуватися. Бізнес-логіка повинна бути включена до складу моделі разом із логікою реалізації для збереження стану програми. Як правило, строго типізовані уявлення використовують типи ViewModel, призначені для зберігання даних, що відображаються в цьому вигляді. Контролер створює та заповнює ці екземпляри ViewModel з моделі.

Функції вигляду. Представлення відповідають за показ інформації через інтерфейс користувача. Вони використовують обробник Razor для впровадження коду .NET у розмітку HTML. Вигляди повинні мати мінімальну логіку, яка має бути пов'язана з представленням вмісту. Якщо є потреба виконувати більшу частину логіки у предствленні для відображення даних зі складної моделі, рекомендується скористатися компонентом представлення, ViewModel або шаблоном представлення, що дозволяє спростити представлення.

Функції контролера. Контролери - це компоненти для керування взаємодією з користувачем, роботи з моделлю та вибору уявлення для відображення. У програмі MVC представлення служить лише для відображення інформації. Обробку введених даних, формування відповіді та взаємодію з користувачем забезпечує контролер. У структурі MVC контролер є початковою відправною точкою і відповідає за вибір робочих типів моделей і представлених уявлень (він контролює, яким чином програма відповідає на конкретний запит).

Тож цей патерн використовують для написання незалежних блоків коду, які можна як завгодно міняти, не торкаючись інших.

Наприклад, щоб можна було переписати спосіб обробки даних, не змінюючи при цьому спосіб відображення. Це дозволяє ефективно працювати декільком програмістам – кожен займається своїм компонентом. При цьому розробнику не потрібно вникати в чужий код і його дії не вплинуть на інші фрагменти програми.

2.5 Фреймворк Entity Framework Core

Entity Framework Core (EF Core) є об'єктно-орієнтованою, розширюваною технологією від компанії Microsoft для доступу до даних. EF Core є ORM-інструментом. Тобто EF Core дозволяє працювати базами даних, але є більш високий рівень абстракції: EF Core дозволяє абстрагуватися від самої бази даних та її таблиць та працювати з даними незалежно від типу сховища. Якщо фізично ми оперуємо таблицями, індексами, первинними та зовнішніми ключами, але на концептуальному рівні, який нам пропонує Entity Framework, ми вже працюємо з об'єктами.

Як технологія доступу до даних Entity Framework Core можна використовувати на різних платформах стека .NET. Це і стандартні платформи типу Windows Forms, консольні програми, WPF, UWP та ASP.NET Core. При

цьому кросплатформова природа EF Core дозволяє використовувати її не тільки на ОС Windows, але і на Linux і Mac OS X.

Центральною концепцією Entity Framework є поняття сутності чи entity. Сутність визначає набір даних, пов'язаних з певним об'єктом. Тому дана технологія передбачає роботу не з таблицями, а з об'єктами та їх колекціями.

Будь-яка сутність, як і будь-який об'єкт із реального світу, має низку властивостей. Наприклад, якщо сутність описує людину, ми можемо виділити такі властивості, як ім'я, прізвище, зростання, вік. Властивості необов'язково представляють прості дані типу int або string, але можуть представляти і більш комплексні типи даних. І в кожної сутності може бути одна або кілька властивостей, які відрізнятимуть цю сутність від інших і будуть унікально визначати цю сутність. Такі властивості називають ключами.

При цьому сутності можуть бути пов'язані асоціативним зв'язком один-до-багатьом, один-до-одному і багато-багатьом, подібно до того, як у реальній базі даних відбувається зв'язок через зовнішні ключі.

Відмінною рисою Entity Framework Core як технології ORM є використання запитів LINQ для вибірки даних з БД. За допомогою LINQ ми можемо створювати різні запити на вибір об'єктів, у тому числі пов'язаних різними асоціативними зв'язками. А Entity Framework при виконанні запиту транслює вирази LINQ у вирази, зрозумілі для конкретної СУБД.

Розглянемо підходи ORM.

Перший – Code First. Він має на увазі, що спочатку пишеться код на C #, а потім з цього коду створюється база даних. Для цього підходу дуже важливо визначити класи моделі або entity, яка буде зберігатися в базі даних, описати її в класах C# у вигляді моделі, і написати клас контексту, який буде працювати з базою даних, що використовується. Підхід Code First найчастіше використовується програмістами C#.

Другий підхід - Database-First- підходить для тих, хто добре знає SQL, але в цьому випадку необов'язково добре знати C #. Насамперед створюється база

даних, потім генерується EDMX-модель бази даних. У цьому XML у файлі .edmx міститься інформація про структуру бази, модель даних та мапінг їх один на одного. Visual Studio є графічний дизайнер, за допомогою якого можна працювати з .edmx

Model-First – третій підхід ORM. Його часто використовують архітектори, тому що при цьому підході можна не знати ні SQL, ні синтаксис C#. В цьому випадку спочатку створюється графічна модель EDMX, у цей час у фоновому режимі створюються класи C# моделі, а потім генерується база даних на основі діаграми EDMX.

Міграції. У процесі розробки цілком ймовірна ситуація, що клас моделі Entity Framework змінився, і доводиться видаляти базу даних, щоб зберігалася відповідність моделі. Але при видаленні бази даних видаляються всі дані з неї.

Щоб зберегти дані при зміні моделі, Entity Framework Core існує функція міграції. Вона дозволяє послідовно застосовувати зміни до бази даних, щоб синхронізувати її з моделлю даних.

У міграції існують операції, які дозволяють видаляти, додавати стовпці та таблиці, зовнішні ключі, змінювати налаштування стовпців, додавати, видаляти та змінювати дані, і так далі. При створенні міграції автоматично створюється клас, де виконуються операції, які необхідні застосування міграції Up() та її повернення метод Down().

2.6 Інтегроване середовище розробки Visual Studio

Visual Studio – це стартовий майданчик для написання, налагодження і складання коду, а також подальшої публікації додатків. Інтегроване середовище розробки (IDE) являє собою багатофункціональну програму, яку можна використовувати для різних аспектів розробки програмного забезпечення. Крім стандартного редактора і відладчика, які існують в більшості середовищ IDE,

Visual Studio включає в себе компілятори, засоби автозавершения коду, графічні конструктори і багато інших функцій для спрощення процесу розробки.

До популярних засобів підвищення продуктивності відносять наступні.

1. Хвилясті лінії. Вони позначають помилки або потенційні проблеми коду прямо під час введення. Ці візуальні підказки дозволяють усувати проблеми негайно і не чекати, поки помилка буде виявлена під час збирання або запуску програми. Якщо навести курсор миші на хвилясту лінію, на екран будуть виведені додаткові відомості про помилку. Крім того, в поле зліва може з'являтися значок лампочки зі швидкими діями щодо усунення помилки (див. рис.2.1).

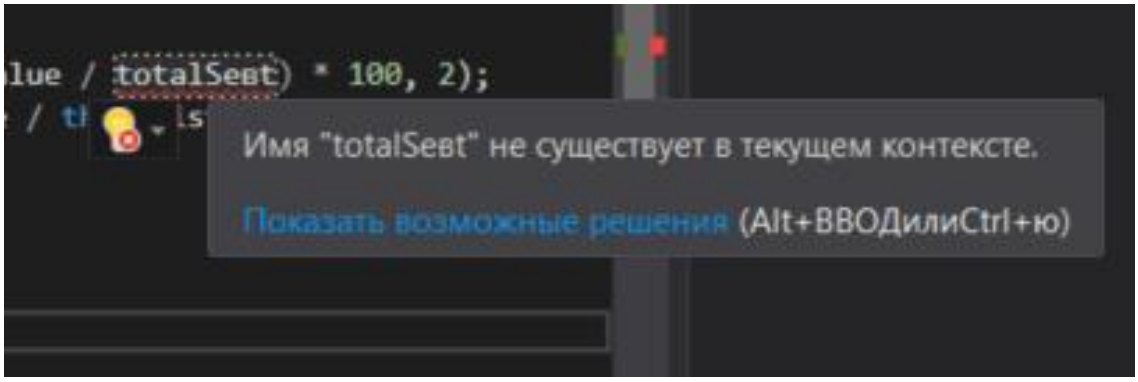


Рисунок 2.2 – Хвилясті лінії

2. Очищення коду. Користувач має можливість одним натисканням кнопки відформатувати код і застосувати до нього виправлення, запропоновані параметрами стилю коду, угодами в файлі EditorConfig і аналізаторами Roslyn. Очищення коду допомагає усунути багато проблем в коді ще до перевірки коду (зараз ця можливість доступна тільки для коду на C #, див. рис.2.2).

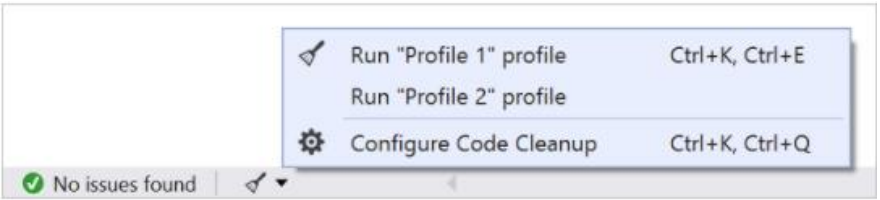


Рисунок 2.3 – Очищення коду

3. Рефакторинг. Рефакторинг включає в себе такі операції, як інтелектуальне перейменування змінних, витягування однієї або декількох рядків коду в новий метод, зміна порядку параметрів методів і багато іншого (див. рис.2.3).

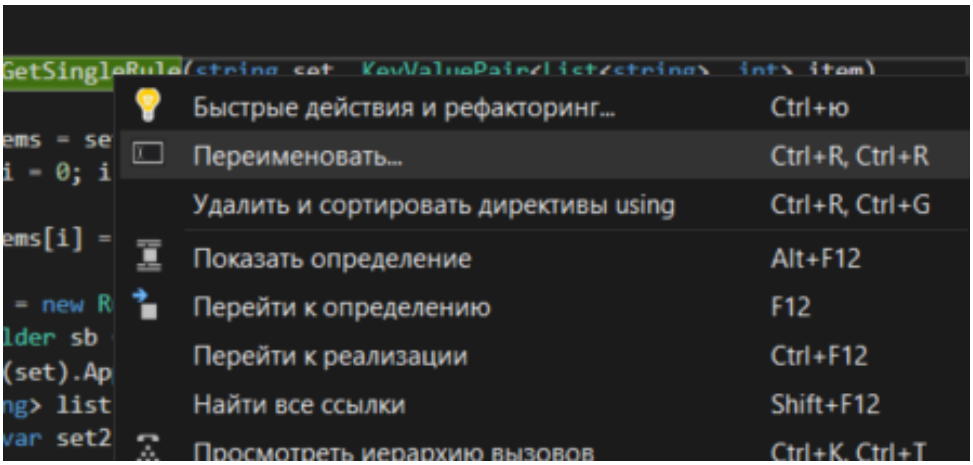


Рисунок 2.4 – Рефакторинг

4. IntelliSense. IntelliSense - це набір функцій, що відображають відомості про код безпосередньо в редакторі і в деяких випадках автоматично створюють невеликі уривки коду. По суті, це базова документація, вбудована в редактор, з якою вам не доводиться шукати інформацію десь ще. Функції IntelliSense залежать від мови. У посібниках з IntelliSense для C #, IntelliSense для Visual C ++, IntelliSense для JavaScript і IntelliSense для Visual Basic (див. рис.2.4).

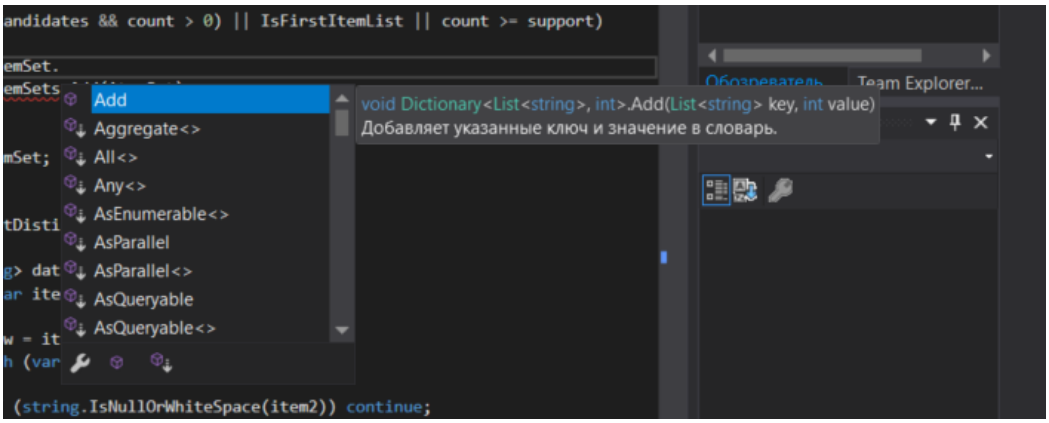


Рисунок 2.5 – IntelliSense

5. Перейти до визначення. З функцією "Перейти до визначення" ви безпосередньо переходите туди, де визначена функція або тип (див. рис.2.5).

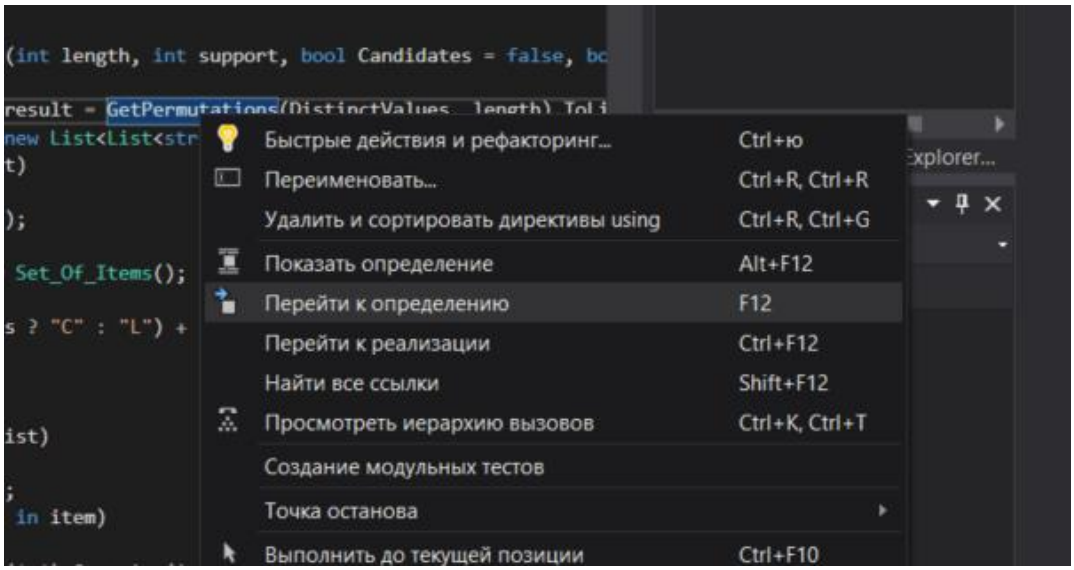


Рисунок 2.6 – Перехід до визначення

2.7 Razor pages

Razor Pages[9] — це кращий спосіб створення програм на основі сторінок або форм ASP.NET Core. Типова програма MVC із строго типізованим поданням буде використовувати контролер для виконання однієї або декількох дій. Контролер буде взаємодіяти з доменом або моделлю даних та створювати екземпляр класу моделі уявлення. Потім цей клас моделі уявлення передається уявленню, пов'язаному з цією дією. При використанні цього підходу в поєднанні зі структурою папок програм MVC за промовчанням для додавання нової сторінки в програму потрібно змінити контролер в одній папці, представлення у вкладеній підпапці в іншій папці і модель представлення ще в одній папці.

Сторінки Razor поєднують дію і модель представлення в один клас і пов'язують цей клас з представленням (названою сторінкою Razor). Всі сторінки

Razor розміщуються в папці Views в корені проекту ASP.NET Core. Сторінки Razor використовують угоду про маршрутизацію, засновану на їхньому імені та розміщенні в цій папці. Обробники ведуть себе так само, як методи дії, але мають у своєму імені HTTP-дієслово, яке вони обробляють. Їм також не обов'язково повертатися, оскільки за умовчанням передбачається, що вони повертають сторінку, з якою вони пов'язані.

Також треба зазначити, що Razor для відображення даних використовує HTML та CSS. HTML відображає мову гіпертекстової розмітки. "Мова розмітки" означає, що HTML використовує теги для ідентифікації різних типів контенту та цілей, які кожен переслідує на веб-сторінці. CSS – це каскадні таблиці стилів. Ця мова розмітки визначає, як HTML-елементи веб-сайту мають відображатись на інтерфейсі сторінки.

У той час як HTML є основною структурою сайту, CSS - це те, що дає сайту стиль. Кольори, цікаві шрифти та фонові зображення – це заслуга CSS. Ця мова впливає на весь настрій веб-сторінки, що робить його неймовірно потужним інструментом та важливою навичкою для веб-розробників. Він також дозволяє веб-сайтам адаптуватися до різних розмірів екрана та типів пристроїв.

Висновки до розділу 2

Для розробки системи було обрано мову програмування C#, платформу .NET Core, яка має можливість використання хмарних технологій, функціонал для кросплатформенності і модульності. Також було використано кросплатформений фреймворк ASP.NET Core в комбінації з патерном MVC, який у свою чергу для виведення даних використовує Razor pages у комбінації з HTML та CSS. Для роботи з БД використовувся фреймворк Entity Framework Core, який використовує підходи ORM та підтримує міграції, що дозволяє легко маніпулювати БД без втрати даних. У якості інтегрованого середовища розробки було обрано Visual Studio.

3 ПРОЄКТУВАННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ АВТОМАГАЗИНУ СПОРТИВНИХ АВТОМОБІЛІВ

3.1 Проєктування та створення бази даних

Невід’ємною частиною кожної інформаційної системи є база даних, яка містить у собі усі дані про систему.

Розроблена бд містить у собі наступні таблиці:

- User, містить дані про користувачів та має наступні поля:Id, Login, Password, RoleID;
- Cart, містить дані про кошик покупців та має наступні поля:Id, UserID, ProductID, Count, Status;
- Image, містить дані про фотографію продукту та має наступні поля:Id, ProductID, Name, Path;
- Product, містить дані про товар та має наступні поля:Id, UserID, CategoryID, Name, ,About, CarBrand, Price;
- Category, містить дані про категорії товарів та має наступні поля:Id, Name;
- Liked, містить дані про товари які вподобали користувачі та має наступні поля:Id, UserID, ProductID, Status;
- Roles, містить дані про ролі які мають користувачі та має наступні поля: Id, Name.

Після проектування БД треба перейти до її створення.

При створені бази даних буде використовуватися один із методів ORM, а саме підхід code-first. Суть цього підходу полягає у наступному: кожна таблиця є сутністю, для якої створюється окрема модель (клас C#).

У моделі прописуються усі поля які зазначені у таблиці, наприклад для таблиці User модель має наступний вигляд:

```
public class User
  {  public int Id { get; set; }
     public string Login { get; set; }
     public string Password { get; set; }
     public string Name { get; set; }
     public string Surname { get; set; }
     public int? RoleID { get; set; }
     public Role Role { get; set; }
     public List<Liked> Likeds { get; set; }
     public List<Product> Products { get; set; }
     public List<Cart> Carts { get; set; }
  }
```

А для таблиці Role:

```
     public class Role
    {public int Id { get; set; }
     public string Name { get; set; }
     public List<User> Users { get; set; }
     public Role()
     {
        Users = new List<User>();
     }
    }
```

Коли прописані моделі для кожної з таблиць, створюється контекст бази даних, який підключає бд через Entity Framework Core. Контекст бази даних являє собою клас, похідний від DbContext. Цей клас містить одну або кілька властивостей типу DbSet<T>, де T – тип об’єкта, що зберігається в базі даних.

Також треба звернути на важливий метод цього класу OnModelCreating(). У цьому методі зазвичай прописуються конфігурації бд; початкові дані, які будуть

додані до бд при її ініціалізації. У нашому випадку ми додаємо туди ролі для користувачів та створюємо адміністратора(рис.3.1).

Наступник кроком є підключення до проекту бази даних. Спочатку пропишемо рядок підключення у файлі appsettings.json (рис. 3.2).

```
public class AppDbContext : DbContext
    {
        public AppDbContext(DbContextOptions<AppDbContext> options)
            :base(options)
        { Database.EnsureCreated(); }

        public DbSet<Cart> Carts { get; set; }
        public DbSet<Category> Categories { get; set; }
        public DbSet<Image> Images { get; set; }
        public DbSet<Liked> Likeds { get; set; }
        public DbSet<Product> Products { get; set; }
        public DbSet<Role> Roles { get; set; }
        public DbSet<User> Users { get; set; }

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            string sellerRoleName = "seller";
            string userRoleName = "user";
            string userRoleAdmin = "admin";

            string adminLogin = "admin";
            string adminPassword = "123";
            //adding roles
            Role sellerRole = new Role { Id = 1, Name = sellerRoleName };
            Role userRole = new Role { Id = 2, Name = userRoleName };
            User adminUser = new User { Id = 1, Login = adminLogin, Password = adminPassword };

            base.OnModelCreating(modelBuilder);
            modelBuilder.Entity<Role>().HasData(new Role[] { sellerRole, userRole });
            modelBuilder.Entity<User>().HasData(new User[] { adminUser });
        }
```

Рисунок 3.1 – Клас DbContext

```
"ConnectionStrings": {
    "DefaultConnection": "Server=DESKTOP-S7GB93D\\SQLEXPRESS;Database=data;Trusted_Connection=True;MultipleActiveResultSets=true"
    }
```

Рисунок 3.2 – Рядок підключення БД

Потім у головному файлі застосунку Program.cs зазначимо сервіс для використання контексту бази даних (рис.3.3).

Фінальним кроком є впровадження міграції. Щоб застосувати міграції треба у Visual Studio перейти до консолі та ввести команду «Add-Migranion назва міграції». Створиться файл міграції у якому генерується код бази даних для її подальшого впровадження. Після чого треба застосувати команду «update-database» і у нас створиться БД із зазначеними таблицями.

```
builder.Services.AddDbContext<AppDbContext>(options =>
        options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")));
```

Рисунок 3.3 – Додавання сервісу контексту БД

Превагою такого підходу до створення БД є те, що у будь-який момент програміст може змінити вміст таблиць без втрати даних. І у випадку втрати файлу з базою даних можна використати міграції для її відновлення, але без повернення її даних.

3.2 Створення користувачів

Наступною важливою частиною є користувачі, а якщо точніше їх розділення на покупців та продавців. Бо існує функціонал який призначений тільки для продавців: виставлення товарів на продаж, перегляд статистики та тільки для покупців: замовлення товарів, формування кошику замовлень. З цією метою існують два поняття – автентифікація та авторизація.

Автентифікація – процес перевірки облікових даних користувача(логін, пароль). Якщо дані введені користувачем співпадають із записами, що зберігаються у базі даних, користувачу надається доступ. У випадку неправильності введених даних користувач не має змоги зайти на ресурс. Але це відноситься до користувачів, які раніше пройшли реєстрацію, в інакшому випадку відвідувач має змогу зареєструватися у системі.

Авторизація відбувається після того, як користувач успішно автентифікується у системі системою. Процес авторизації визначає, чи людина, що пройшла перевірку, доступ до певних ресурсів: інформації, файлів, бази даних. Фактори автентифікації, необхідні для авторизації, можуть відрізнятися залежно від рівня безпеки.

Після уточнення понять автентифікації та авторизації перейдемо до їх реалізації.

У БД є таблиця Users, яка містить у собі поля які описують користувача. Але нас цікавить поле RoleID, яке пов’язано з таблицею Role. Таблиця Role містить поле Id та назву ролі. Дані таблиці пов'язані ставленням одним-багатьом, тобто один користувач може мати лише одну роль, а до однієї ролі можуть належати кілька користувачів (додаток А).

Використання ролей допоможе розділити користувачів та функції які вони виконують, іншими слова реалізація авторизації. Наприклад продавець має можливість додавати товар та редагувати його, а покупець ні, тому перед методом який призначений продавцю ми прописуємо атрибут, який має наступний вигляд: [Authorize(Roles = “seller”)]. Тут ми вказуємо що тільки користувач з певною роллю має право використовувати метод.

Для реєстрації користувачів та їх входу до системи створимо окремі допоміжні моделі (додаток Б) та на основі цих моделей створюються вигляди (додаток В). У моделі ми вказуємо які дані повинен ввести користувач, щоб зареєструватися або увійти до системи.

Щоб користувачі мали змогу реєстрації та входу до системи, у головному файлі додамо сервіс автентифікації, який використовує файли кукі та вказує шлях для вигляду реєстрації та входу до системи (рис.3.4). Вигляд форми для реєстрації та для входу наведені у додатку Є

```
builder.Services.AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme)
                .AddCookie(options =>
                {
                    options.LoginPath = new Microsoft.AspNetCore.Http.PathString("/Account/Login");
                    options.AccessDeniedPath = new Microsoft.AspNetCore.Http.PathString("/Account/Login");
                });
```

Рисунок 3.4 – Додавання сервісу автентифікації

Метод для реєстрації покупця та продавця майже однакові, їх різниця у тому, яка роль надається (рис.3.5).

```
 [HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> RegisterUser(RegisterModel model)
{
    if (ModelState.IsValid)
    {
        User user = await _context.Users.FirstOrDefaultAsync(u => u.Email == model.Email);
        if (user == null)
        {

            user = new User { Login = model.Login, Password = model.Password };
            Role userRole = await _context.Roles.FirstOrDefaultAsync(r => r.Name == "user");
            if (userRole != null)
                user.Role = userRole;

            _context.Users.Add(user);
            await _context.SaveChangesAsync();

            await Authenticate(user);

            return RedirectToAction("Index", "Home");
        }
        else
            ModelState.AddModelError("", " Account already exists ");
    }
       return View(model);
```

Рисунок 3.5 – Метод реєстрації користувача

У методі реєстрації користувача ми спочатку перевіряємо чи є такий користувач, якщо такий є – виводиться відповідне повідомлення. В іншому випадку створюється новий користувач, та надається відповідна роль (рис.3.6).

У цьому методі входу до системи ми перевіряємо чи існує користувач з введеними даними, якщо такого не знайдено – виводиться повідомлення помилки. В іншому випадку ми аутентифіковуємо користувача, надаючи йому відповідну роль.

У додатку Г наведено код контролера кошика покупці, в додатку Д – код класу для фільтрації по категоріям. Код класу для відправки листа по замовленню міститься у додатку Е.

```
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Login(LoginModel model)
{
    if (ModelState.IsValid)
    {
        User user = await _context.Users
            .Include(u => u.Role)
            .FirstOrDefaultAsync(u => u.Login == model.Login && u.Password == model.Password);
        if (user != null)
        {
            await Authenticate(user);

            return RedirectToAction("Index", "Home");
        }
        ModelState.AddModelError("", " Incorrect username and/or password ");
    }
    return View(model);
}
private async Task Authenticate(User user)
{
    // create one claim
    var claims = new List<Claim>
            {
                new Claim(ClaimsIdentity.DefaultNameClaimType, user.Login),
                new Claim(ClaimsIdentity.DefaultRoleClaimType, user.Role?.Name)
            };
    // create a ClaimsIdentity object
    ClaimsIdentity id = new ClaimsIdentity(claims, "ApplicationCookie", ClaimsIdentity.DefaultNameClaimType,
        ClaimsIdentity.DefaultRoleClaimType);
    // setting authentication cookies
    await HttpContext.SignInAsync(CookieAuthenticationDefaults.AuthenticationScheme, new ClaimsPrincipal(id));
      }
```

Рисунок 3.6 – Метод для входу до системи

3.3 Реалізація алгоритму Apriori

Для формування статистики покупок був реалізований алгоритм Apriori, які аналізує транзакції користувачів і у результаті шляхом знаходження асоціативних правил дозволяє передбачати, який товар іще потрібен покупцю, якщо він уже визначився з деякими покупками.

За роботу алгоритму відповідають чотири класи, кожен із яких має свій функціонал.

Класи Set_Of_Items та Rule мають у собі прописані методи get і set. За допомогою цих класів ми маємо змогу працювати з даними та працювати з асоціативними правилами (рис.3.7).

Наступний клас – Output. У цьому класі прописані три методи для виводу інформації. Перші два методи створені для виводу даних у таблицю, розділяючи їх комою. Третій метод має просту функцію – виводить знак відсотку біля

потрібних даних, у нашому випадку це значення оцінок підтримки (support) та достовірності (confidence) (рис.3.8).

```
public  class Set_Of_Items : Dictionary<List<string>, int>
    {
        public string Label { get; set; }
        public int Support { get; set; }
          }


public class Rule
    {
        public string Label { get; set; }
        public double Confidance { get; set; }
        public double Support { get; set; }
          }
```

Рисунок 3.7 – Класи Set_Of_Items та Rule

```
public static class Output
  {
      public static string Display(this List<string> list, string separator = ", ")
      {
          if (list.Count == 0)
              return string.Empty;
          StringBuilder sb = new StringBuilder();
          sb.Append(list[0]);
          for (int i = 1; i < list.Count; i++)
          {
              sb.Append(string.Format("{0}{1}", separator, list[i]));
          }
          return sb.ToString();
      }

      public static string Display(this List<string> list, string exclude, string separator = ", ")
      {
          List<string> dump = new List<string>();
          foreach (var item in list)
          {
              if (item == exclude) continue;
              dump.Add(item.ToString());
          }
          return dump.Display();
      }

      public static string Percent(this object item)
      {
          return item.ToString() + " %";
      }
        }
```

Рисунок 3.8 – Клас Output

Головним класом є Apriori_Algo. Тут описані головні методи алгоритму. Першим методом є SetDistinctValues. Цей метод аналізує вхідні дані та додає їх до колекції, для подальшої роботи з нею (рис.3.9).

```
public void SetDistinctValues(List<string> values)
        {
            List<string> data = new List<string>();
            foreach (var item in values)
            {
                var row = item.Split(' ');
                foreach (var item2 in row)
                {
                    if (string.IsNullOrWhiteSpace(item2)) continue;
                    if (!data.Contains(item2))
                        data.Add(item2);
                }
            }
            DistinctValues = new List<string>();
            DistinctValues.AddRange(data.OrderBy(a => a).ToList());
        }
```

Рисунок 3.9 – Метод SetDistinctValues

Далі починається робота з асоціативними правилами. Для початку нам треба підрахувати значення підтримки та довіри. Підраховується кількість одного продукту та загальна кількість продуктів, на основі цих даних робляться потрібні розрахунки (рис.3.10).

```
private Rule GetSingleRule(string set, KeyValuePair<List<string>, int> item)
        {
            var setItems = set.Split(',');
            for (int i = 0; i < setItems.Count(); i++)
            {
                setItems[i] = setItems[i].Trim();
            }
            Rule rule = new Rule();
            StringBuilder sb = new StringBuilder();
            sb.Append(set).Append(" => ");
            List<string> list = new List<string>();
            foreach (var set2 in item.Key)
            {
                if (setItems.Contains(set2)) continue;
                list.Add(set2);
            }
            sb.Append(list.Display());
            rule.Label = sb.ToString();
            int totalSet = 0;
            foreach (var first in ItemSets)
            {
                var myItem = first.Keys.Where(a => a.Display() == set);
                if (myItem.Count() > 0)
                {
                    first.TryGetValue(myItem.FirstOrDefault(), out totalSet);
                    break;
                }
            }
            rule.Confidance = Math.Round(((double)item.Value / totalSet) * 100, 2);
            rule.Support = Math.Round(((double)item.Value / this.list.Count) * 100, 2);
            return rule;
        }
```

Рисунок 3.10 – Метод отримання асоціативного правила

На основі отриманих результатів будуються готові асоціативні правила (рис.3.11).

```
public List<Rule> GetRules(Set_Of_Items items)
        {
            List<Rule> rules = new List<Rule>();
            foreach (var item in items)
            {
                foreach (var set in item.Key)
                {
                    rules.Add(GetSingleRule(set, item));
                    if (item.Key.Count > 2)
                        rules.Add(GetSingleRule(item.Key.Display(exclude: set), item));
                }
            }

            return rules.OrderByDescending(a => a.Support).ThenByDescending(a => a.Confidance).ToList();
              }
```

Рисунок 3.11 – Метод отримання асоціативних правил

Останній метод слугує для сортування отриманих раніше результатів, це зроблено для коректного виводу даних (рис.3.12).

```
public Set_Of_Items GetItemSet(int length, int support, bool Candidates = false, bool IsFirstItemList = false)

        {
            List<IEnumerable<string>> result = GetPermutations(DistinctValues, length).ToList();
            List<List<string>> data = new List<List<string>>();
            foreach (var item in result)
            {
                data.Add(item.ToList());
            }
            Set_Of_Items itemSet = new Set_Of_Items();
            itemSet.Support = support;
            itemSet.Label = (Candidates ? "C" : "L") + length.ToString();
            foreach (var item in data)
            {
                int count = 0;
                foreach (var word in list)
                {
                    bool found = false;
                    foreach (var item2 in item)
                    {
                        if (word.Split(' ').Contains(item2))
                            found = true;
                        else
                        {
                            found = false;
                            break;
                        }

                    }
                    if (found)
                        count++;
                }
                if ((Candidates && count > 0) || IsFirstItemList || count >= support)
                {
                    itemSet.Add(item, count);
                    ItemSets.Add(itemSet);
                }
            }
            return itemSet;
        }
```

Рисунок 3.12 – Метод Сортування та виводу правил

3.4 Функціонал продавця

Перш за все продавець може переглядати усі наявні товари (рис.3.13).

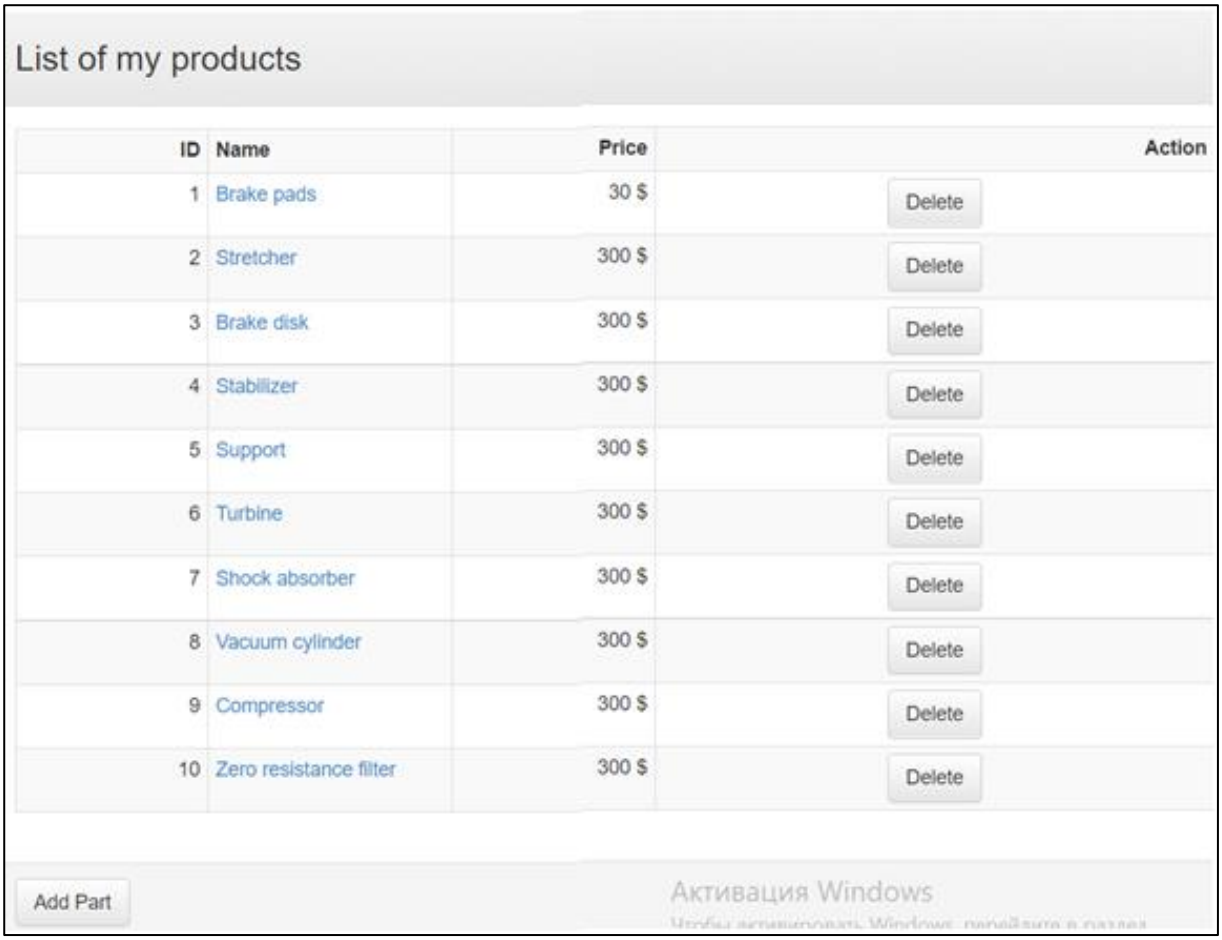


Рисунок 3.13 – Список товарів продавця

Також продавець може додавати, видаляти і редагувати свої товари. Форми додавання та редагування однакові(див. рис.3.14).

Окремої уваги заслуговує додавання картинки для товару. Для її відображення у вигляді ми створюємо елемент div, у якому вказуємо тип файлу та його положення (рис.3.15).

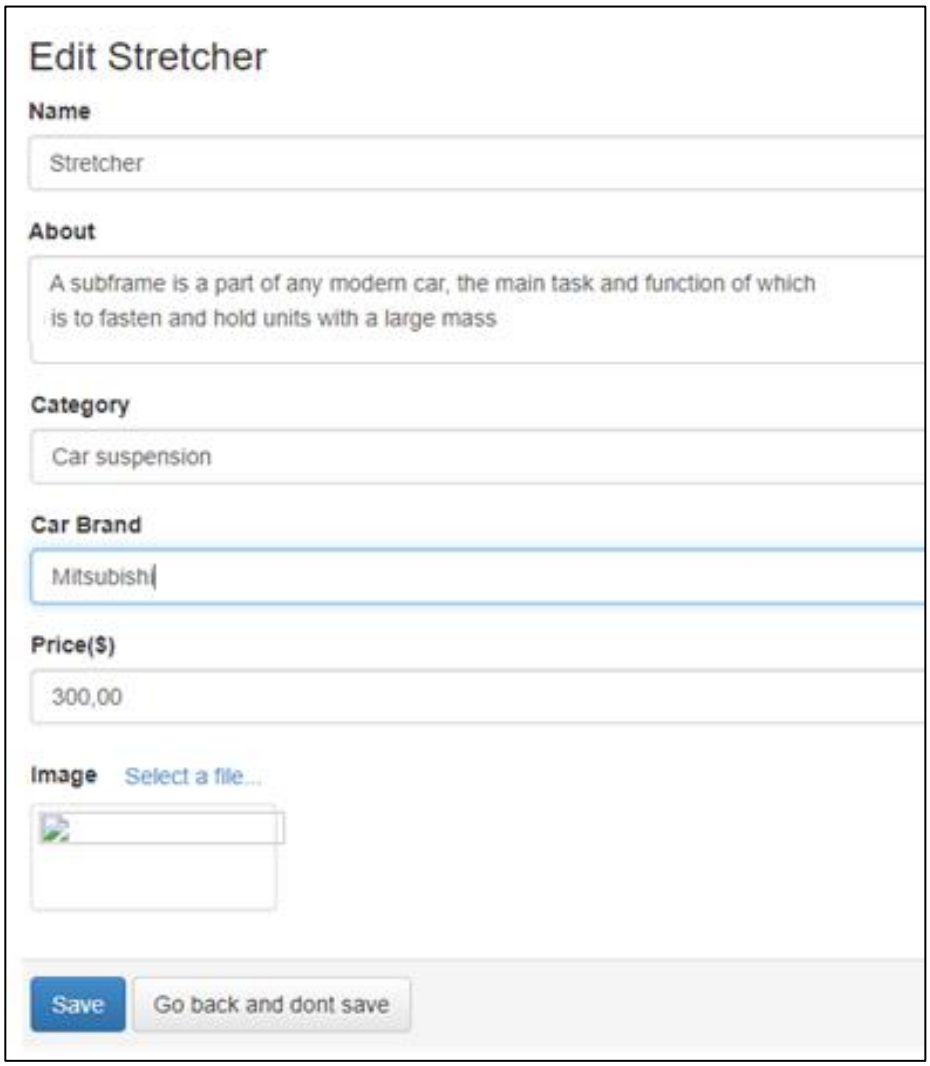


Рисунок 3.14 – Форма редагування товару

```
<div style="position:relative;">
                <label>Image</label>
                <a class='btn' href='javascript:;'>
                    Select a file...
                    <input type="file" name="Image" size="40"
                           style="position:absolute;z-index:2;top:0;
                                  left:0;filter: alpha(opacity=0); opacity:0;
                                  background-color:transparent;color:transparent;"
                           onchange='$("#upload-file-info").html($(this).val());'>
                </a>
                <span class='label label-info' id="upload-file-info"></span>
                  </div>
```

Рисунок 3.15 – Додавання картинки: тип файлу та його положення

У контролері пропишемо метод який зберігає файл на сервері у каталозі wwwroot, у якому ми створимо папку Files. Для отримання повного шляху до каталогу wwwroot використовується властивість WebRootPath об'єкта IWebHostEnvironment. Для копіювання файлу в папку Files створюється потік FileStream, який записується файл за допомогою методу CopyToAsync (рис.3.16).

```
[HttpPost]
        public async Task<IActionResult> AddFile(IFormFile uploadedFile)
        {
            if (uploadedFile != null)
            {
                // path to Files folder
                string path = "/Files/" + uploadedFile.FileName;
                // save the file to the Files folder in the wwwroot directory
using (var fileStream = new FileStream(_appEnvironment.WebRootPath + path, FileMode.Create))
                {
                    await uploadedFile.CopyToAsync(fileStream);
                }
                FileModel file = new FileModel { Name = uploadedFile.FileName, Path = path };
                _context.Files.Add(file);
                _context.SaveChanges();
            }

            return View();
              }
```

Рисунок 3.16 – Додавання картинки: збереження файлу

Найважливіше для продавця – це перегляд статистики, яка формується на основі замовлень продавців та її аналізу шляхом знаходження асоціацій, розхрахунку їх підтримки та достовірності, виявлення асоціативних правил з використанням алгоритму Apriory.

Ця статистика допомагає продавцю оптимізувати список своїх товарів для підвищення продажів. Треба зазначити, що для продавця є доступними для перегляду асоціативні правила які зустрічаються найчастіше – відібрані за алгоритмом Apriori (рис.3.17).

Адміністратор може переглядати усі знайдені асоціації.

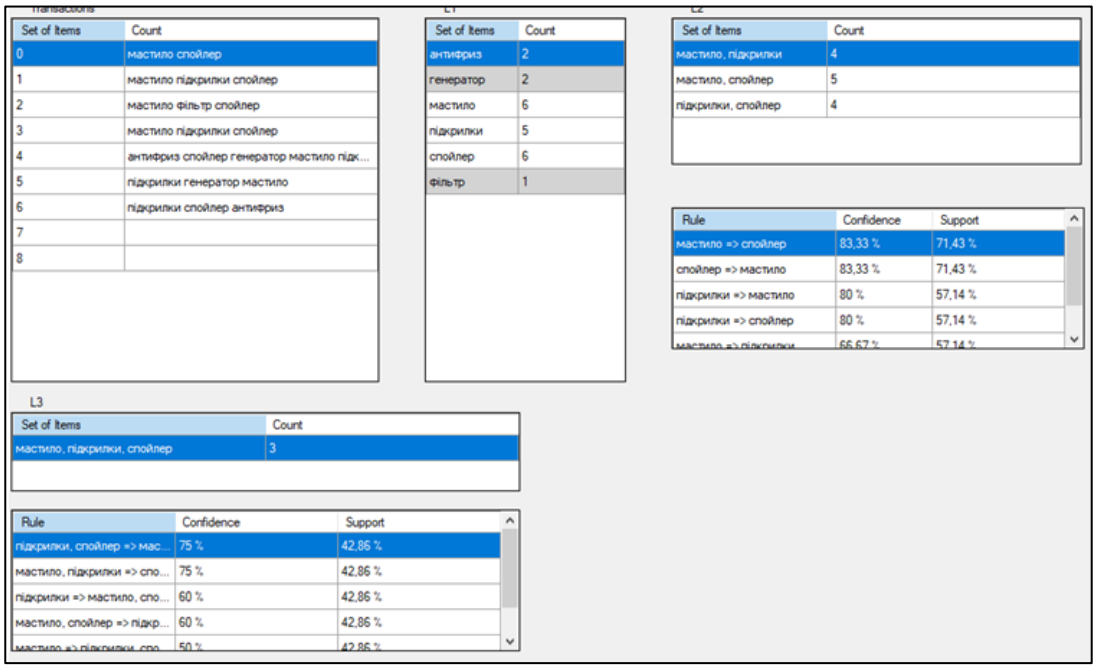


Рисунок 3.17 – Виявлені асоціативні правила

3.4 Функціонал покупця

На головній сторінці (рис.3.18) покупець бачить список товарів. Зліва є таблиця з категоріями, за якими покупець фільтрує товари. Також кожен товар можна додати до кошика (рис.3.20) або до списку товарів які зацікавили покупця (рис.3.19).

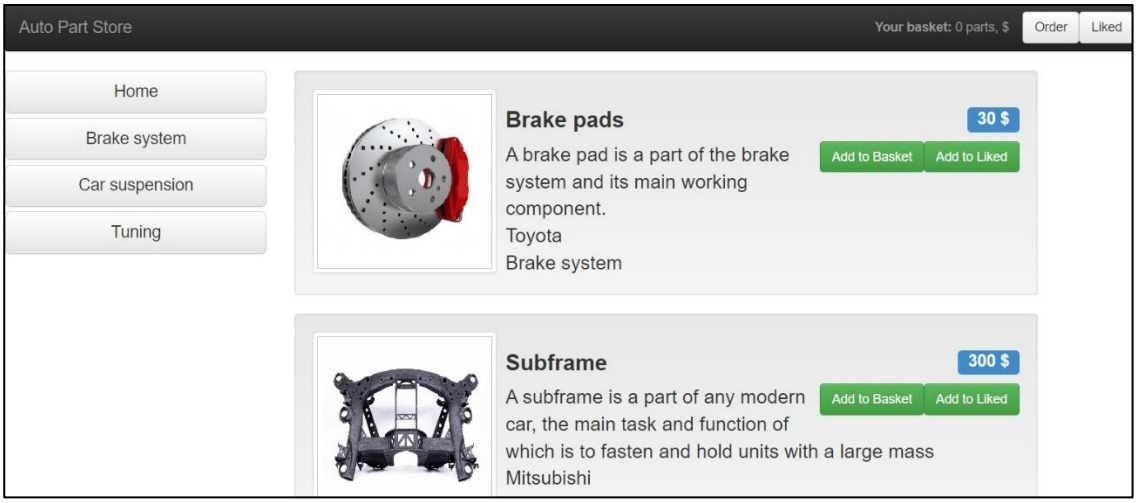


Рисунок 3.18 – Головна сторінка

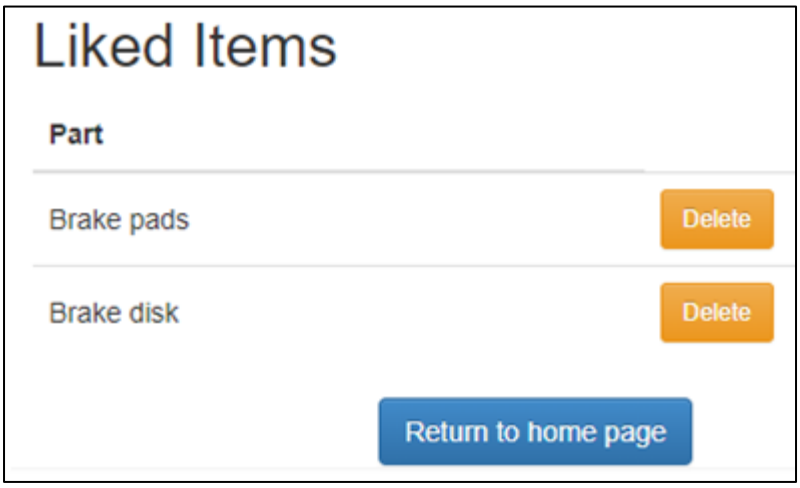


Рисунок 3.19 – Список товарів які зацікавили покупця

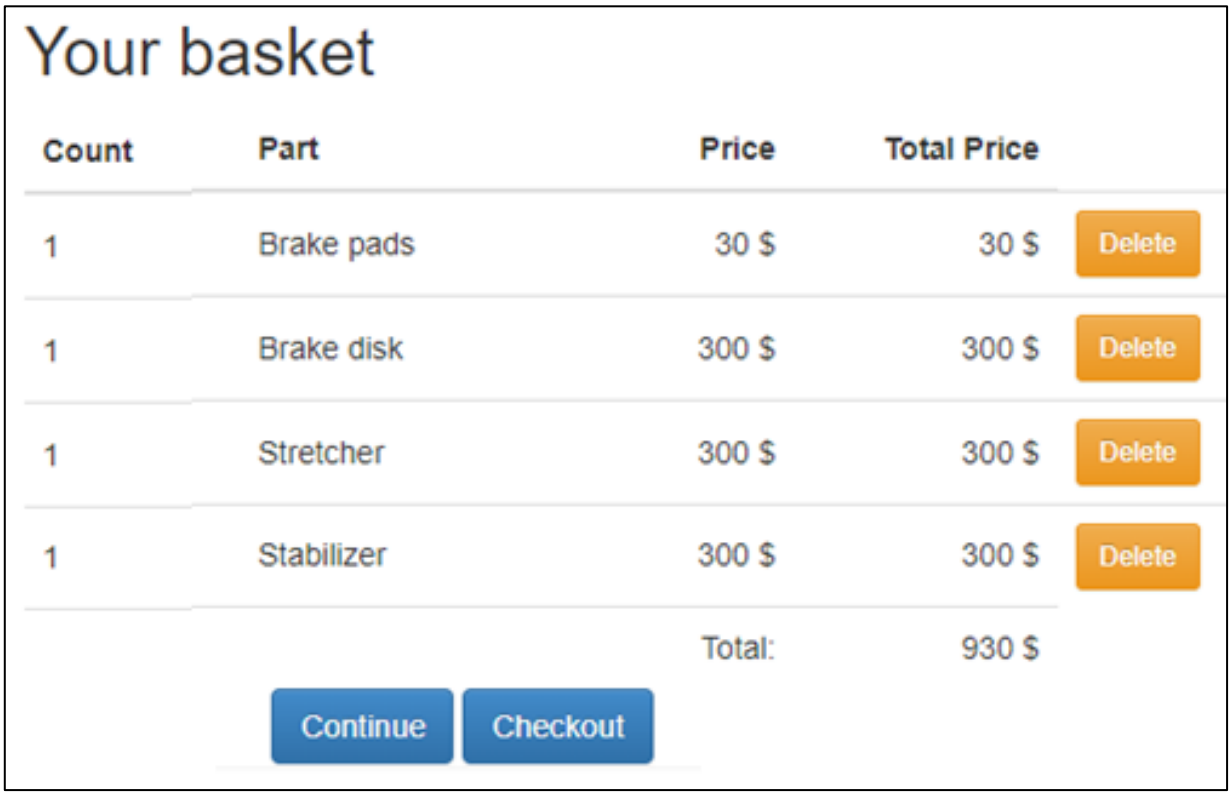


Рисунок 3.20 – Кошик покупця

Покупець має можливість фільтрувати товари по категоріям. Для реалізації цього функціоналу був створений метод та вигляд, які знаходять усі категорії товарів та фільтрують їх (рис.3.21, рис.3.22).

```
public PartialViewResult Menu(string types = null)
        {
            ViewBag.Selected_Type = types;

            IEnumerable<string> type = repository.Product
              .Select(part => part.CategoryID)
              .Distinct()
              .OrderBy(x => x);
            return PartialView(type);


        }
```

Рисунок 3.21 – Метод фільтрації по категоріям

```
@model IEnumerable<string>

@Html.ActionLink("Home","List", "Pproduct", null,
    new { @class = "btn btn-block btn-default btn-lg" })

@foreach (var link in Model)
{
    @Html.RouteLink(link, new
{
    controller = "Product",
    action = "List",
    types = link,
    type = link,
    page = 1
}, new
{
    @class = "btn btn-block btn-default btn-lg"
    + (link == ViewBag.Selected_Type ? " btn-primary" : "")
})
}
```

Рисунок 3.22 – Вигляд фільтрації по категоріям

У результаті покупець на головній сторінці бачить меню, за допомогою якого відбуваеться фільтрація по категоріям Brake system та Tuning (рис.3.23, рис.3.24). Для здійснення фільтрації необхідно натиснути відповідні кнопки на панелі у лівій частині головного вікна. А якщо покупець хоче повернутися на головну сторінку для цього є кнопка Home.

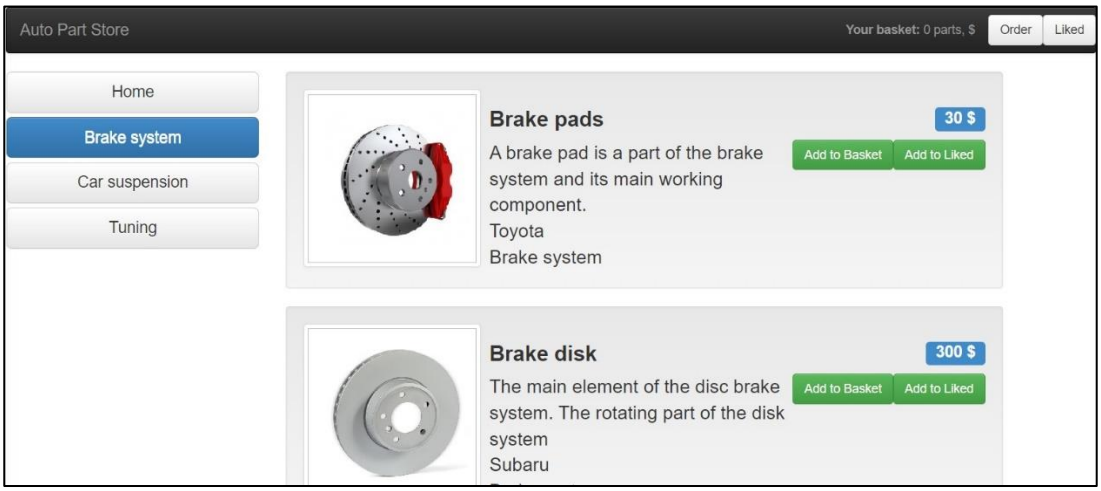


Рисунок 3.23 – Фільтрація за категорію Brake system

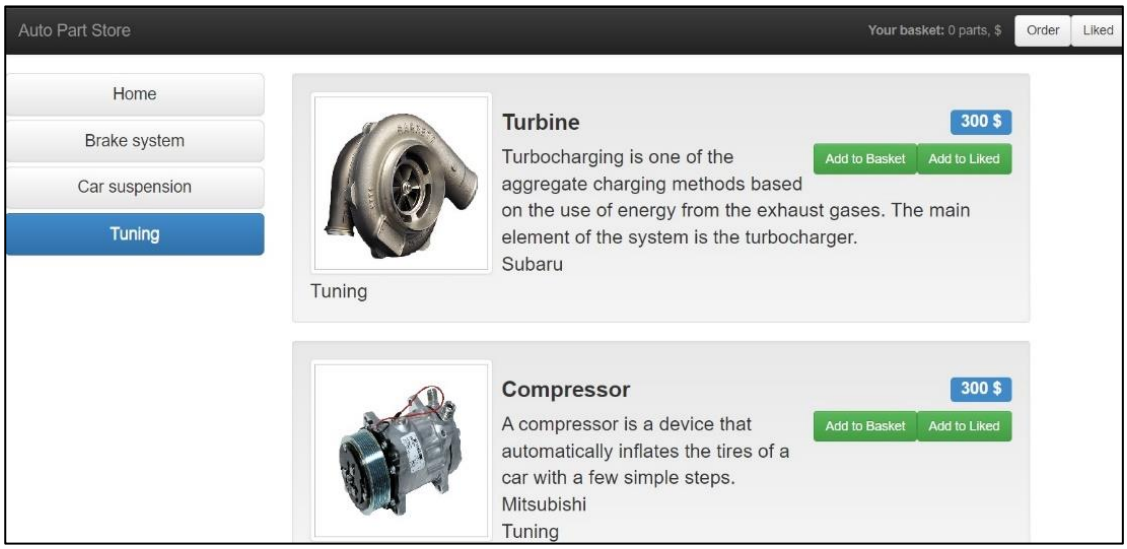


Рисунок 3.24 – Фільтрація за категорію Tuning

Після додавання товару до кошика покупець може продовжити перегляд товарів або оформити замовлення.

При оформленні замовлення покупець заповнює форму для покупки, після заповнення форми вміст замовлення та дані для доставки замовлення відправляються продавцю на електронну пошту (рис.3.25, рис. 3.26).

New order checked

List of items
1 x Brake pads (total: 30,00 ₽2 x Brake disk (total: 600,00 ₽1 x Compressor (total: 300,00 ₽Total cost: 930,00 ₽----- Delivery Vova kobera
221653653532
0512
Nikolaev
Ukraine

C.O.D.: noPayment: no

Рисунок 3.25 – Лист продавцю про замовлення

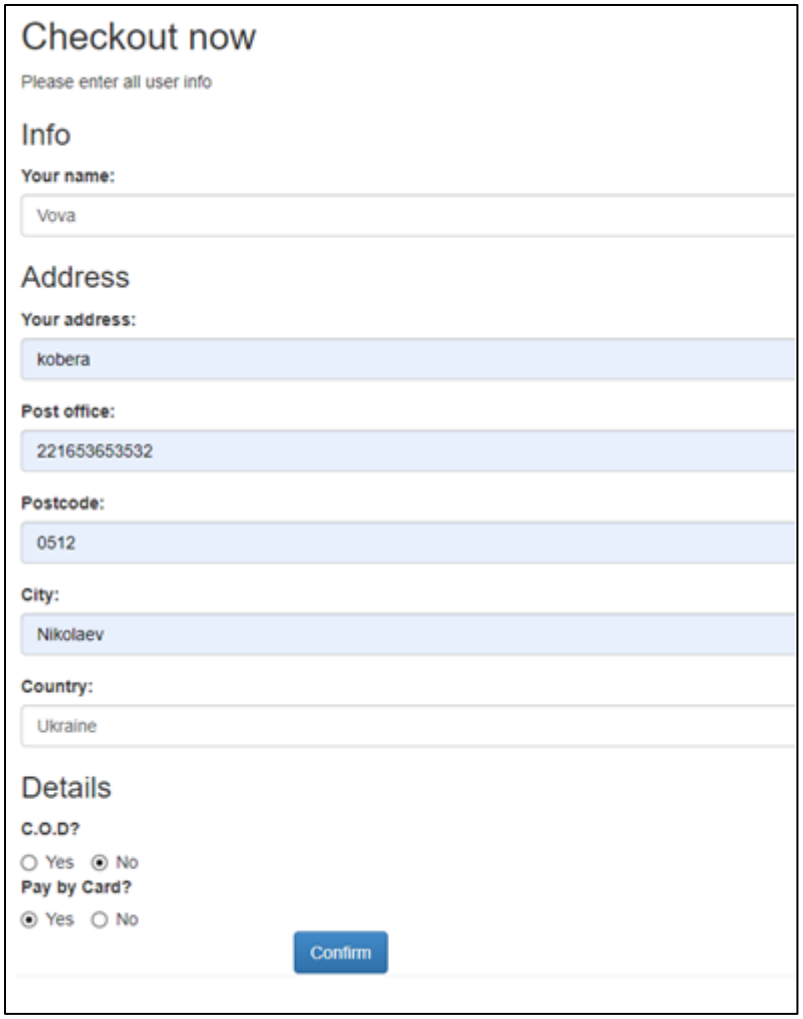


Рисунок 3.26 – Форма для замовлення

На основі сформованих асоціативних правил покупець при перегляді товару має можливість бачить рекомендовані товари (рис.3.27).

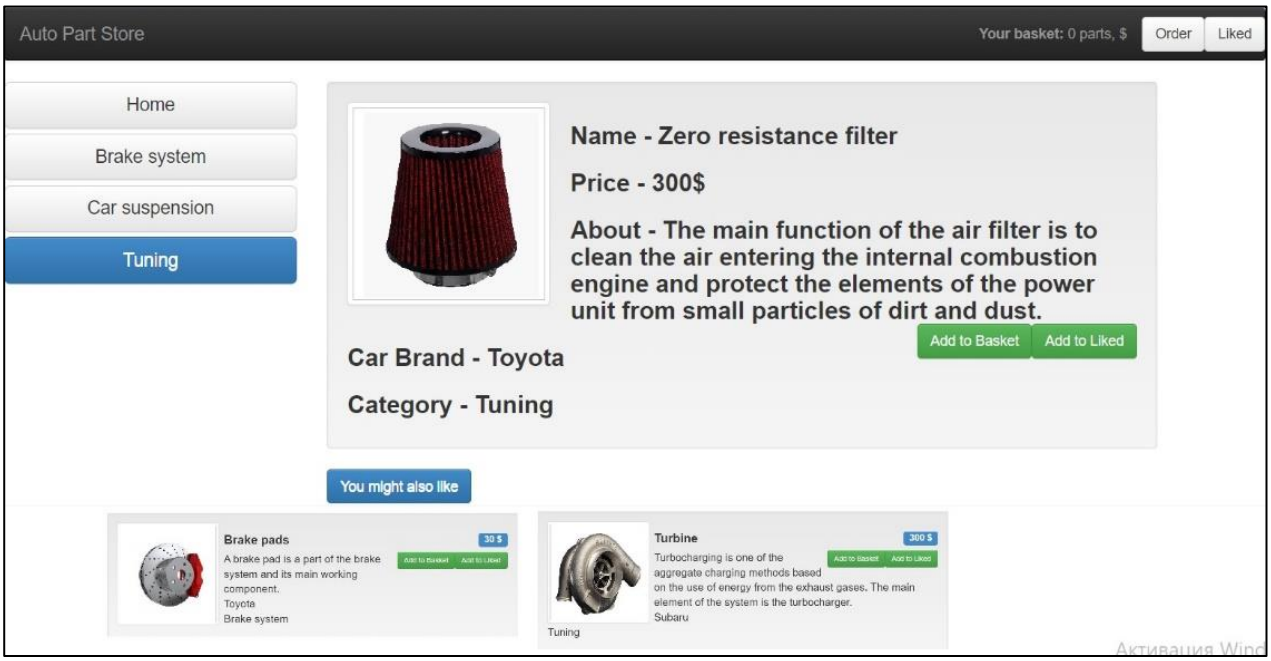


Рисунок 3.27 – Рекомендації на основі асоціативних правил

3.5 Функціонал адміністратора

Як зазначалося раніше, при створені міграції ми за замовчуванням створили користувача з роллю адміністратора, який може переглядати повний перелік асоціативних правил, сформований на основі замовлень покупців. Є можливість переглянути список усіх тразакцій та у наступній таблиці усі товари і кількість їх замовлень (рис.3.28).

Виводяться також знайдені асоціації, на основі яких формуюються асоціативні правила та відображаються розраховані для них підтримка й достовірність. Адміністратор має змогу змінювати порогові значення підтримки та достовірності, пересуваючи маркер повзунків у правій нижній частині вікна (рис.3.28, рис.3.29). Установлені адміністратором порогові значення для оцінок асоціативних правил впливають на формування рекомендацій, які буде отримувати покупець при відвідування магазину та виборі певних товарів і послуг.

Transactions

Set of Items	Count
0	Brake_pads Zero_resistance_filter
1	Brake_pads Turbine Zero_resistance...
2	Brake_pads Vacuum_cylinder Zero_r...
3	Brake_pads Turbine Zero_resistance...
4	Compressor Zero_resistance_filter Br...
5	Turbine Brake_disk Brake_pads
6	Turbine Zero_resistance_filter Compr...

Table1

Set of Items	Count
Brake_disk	2
Brake_pads	6
Compressor	2
Turbine	5
Vacuum_cylinder	1
Zero_resistance_filter	6

Рисунок 3.28 – Список транзакцій, перелік товарів та кількості їх замовлень

Table2

Set of Items	Count
Brake_disk, Brake_pads	2
Brake_disk, Turbine	2
Brake_pads, Turbine	4
Brake_pads, Zero_resistance_filter	5

Rule	Confidence	Support
Brake_pads => Zero_resistance_...	83,33 %	71,43 %
Zero_resistance_filter => Brake_...	83,33 %	71,43 %
Turbine => Brake_pads	80 %	57,14 %
Turbine => Zero_resistance_filter	80 %	57,14 %
Brake_pads => Turbine	66,67 %	57,14 %
Zero_resistance_filter => Turbine	66,67 %	57,14 %

Table3

Set of Items	Count
Brake_disk, Brake_pads, Turbine	2
Brake_pads, Turbine, Zero_resistanc...	3
Compressor, Turbine, Zero_resistanc...	2

Rule	Confidence	Support
Turbine, Zero_resistance_filter =...	75 %	42,86 %
Brake_pads, Turbine => Zero_re...	75 %	42,86 %
Turbine => Brake_pads, Zero_re...	60 %	42,86 %
Brake_pads, Zero_resistance_filt...	60 %	42,86 %
Brake_pads => Turbine, Zero_re...	50 %	42,86 %
Zero_resistance_filter => Brake_...	50 %	42,86 %

Confidence Support

Рисунок 3.29 – Асоціативні правила для двоелементних наборів даних

Table2

Set of Items	Count
Brake_pads, Turbine	4
Brake_pads, Zero_resistance_filter	5
Turbine, Zero_resistance_filter	4

Rule	Confidence	Support
Brake_pads => Zero_resistance_fi...	83,33 %	71,43 %
Zero_resistance_filter => Brake_p...	83,33 %	71,43 %
Turbine => Brake_pads	80 %	57,14 %
Turbine => Zero_resistance_filter	80 %	57,14 %
Brake_pads => Turbine	66,67 %	57,14 %
Zero_resistance_filter => Turbine	66,67 %	57,14 %

Table3

Set of Items	Count
Brake_pads, Turbine, Zero_resistanc...	3

Rule	Confidence	Support
Turbine, Zero_resistance_filter => ...	75 %	42,86 %
Brake_pads, Turbine => Zero_resi...	75 %	42,86 %
Turbine => Brake_pads, Zero_resi...	60 %	42,86 %
Brake_pads, Zero_resistance_filte...	60 %	42,86 %
Brake_pads => Turbine, Zero_resi...	50 %	42,86 %
Zero_resistance_filter => Brake_p...	50 %	42,86 %

Confidence Support

Рисунок 3.30 – Асоціативні правила для трьохелементних наборів даних

Висновки до розділу 3

У даному розділі було описано розробку та програмну реалізацію інформаційної магазину автозапчастин спортивних автомобілів, яка дозволяє покупцям переглядати товари та сортувати їх за категоріями, формувати список товарів які зацікавили покупця, формувати кошик та оформлення замовлення із використанням рекомендацій, сформованих з використанням алгоритму Apriori. Продавець має можливість додавати нові товари, переглядати усі товари та редагувати їх, також отримувати статистику по продажам товарів, виявляти асоціативні правила для оптимізації та покращення продажу.

Спеціальний розділ

ОХОРОНА ПРАЦІ

до кваліфікаційної роботи

на тему:

ІНФОРМАЦІЙНА СИСТЕМА МАЗАГИНУ АВТОЗАПЧАСТИН СПОРТИВНИХ АВТОМОБІЛІВ

Спеціальність 122 «Комп’ютерні науки»

122 – БКР – 401.21810301

Виконав студент 4-го курсу, групи 401

_____ *І. В. Барбулат*

(підпис, ініціали та прізвище)

«___» червня 2022 р.

Консультант ст. викладач

_____*О. В. Макарова*

(підпис, ініціали та прізвище)

«___» червня 2022 р.

Миколаїв – 2022

4 ОХОРОНА ПРАЦІ: САНІТАРНО-ГІГІЄНІЧНІ ВИМОГИ ПРИМІЩЕННЯ ДЛЯ РОБОТИ З КОМП'ЮТЕРНОЮ ТЕХНІКОЮ

У наші дні все більше підприємств почитають використовувати комп'ютери для роботи. Використання комп'ютерної техніки полегшує різні процеси на підприємствах. Але при всіх перевагах комп'ютерів, він може нести негативні наслідки для працівників, які працюють за такою технікою. Люди які постійно працюють за комп'ютером зазвичай мають проблеми із зором, із спиною, а також страждають від електромагнітного випромінювання. Але іншою важливою проблемою є дотримання санітарно – гігієнічних вимог приміщення для роботи з комп'ютерною технікою. Для нормальної роботи слід дотримуватися умов мікроклімату, освітлення, вібрації. Також треба захистити користувача від впливу іонізуючих та неіонізуючих електромагнітних полів та впливу від випромінювання моніторів.

Метою спеціального розділу з охорони праці є створення безпечних і здорових умов праці на робочому місці. Відповідно до поставленої мети було сформульовано **завдання дослідження**:

1. Навести основні вимоги до мікроклімату.
2. Розглянути вимоги до освітлення приміщень, робочих місць та зроблений розрахунок коефіцієнту природного освітлення у приміщенні.
3. Проаналізувати вимоги до вібрації.
4. Дослідити захист користувачів від впливу іонізуючих та неіонізуючих електромагнітних полів та випромінювання моніторів.

4.1 Вимоги до мікроклімату

Якість повітря в приміщенні впливає на самопочуття людини. У приміщеннях з комп'ютерною технікою, внаслідок того що третина

споживаної нею енергії розсіюється у вигляді тепла, формуються специфічні умови мікроклімату а саме, підвищується температура до 26-27 °С, відносна вологість падає до 40%. Вміст двоокису вуглецю збільшується внаслідок знаходження великої кількості користувачів в приміщенні, водночас повітря деіонізується, а це збільшує кількість позитивних іонів і несприятливо впливає на працездатність, самопочуття, сприйняття нового матеріалу.

Деіонізація повітря пояснюється притягненням від'ємних іонів до екрану монітора, який має позитивний потенціал і відштовхуванням позитивних іонів (це стосується моніторів з електронно-вакуумними трубками). Медичними дослідженнями встановлено, що на життєдіяльність людини впливає не кількість іонів повітря, а співвідношення між позитивно і негативно зарядженими іонами. Згідно норм в приміщеннях з комп'ютерною технікою оптимальним вважається вміст 3000-5000 від'ємних іонів кисню в І см3 повітря, а позитивних 1500-3000 іонів. Концентрація від'ємних іонів біля монітору повинна бути не меншою 600 іон/см3.

При роботі комп'ютерної техніки в повітря виділяються шкідливі для організму людини речовини. Новий комп'ютер створює в приміщенні специфічний запах. Корпус монітору, нагріваючись при нормальній роботі до 50-55 °С, починає виділяти в повітря пари трифенілфосфата. Але нагрівається не лише монітор, але і блок живлення, і процесор, і материнська плата і відеокарта. А всі вони містять смоли та органічні речовини з фтором, хлором і фосфором, які при нагріванні можуть виділятися в повітря [14].

Озон утворюється внаслідок впливу електричних зарядів, які виникають у лазерних принтерах, на кисень повітря. І хоча нові лазерні принтери здійснюють фільтрацію озону, проблема існує, з часом фільтр псується і його необхідно вчасно замінити. Озон сильно подразнює слизисту оболонку носа, очей і горла та може призвести до ракових захворювань як канцерогенна речовина.

У виробничих приміщеннях на робочих місцях з ПК мають забезпечуватись оптимальні значення параметрів мікроклімату: температури, відносної вологості й швидкості руху повітря (ДСанПіН 3.3.2.007-98, ДСН 3.3.6.042-99) (табл. 4.1).

Таблиця 4.1. Норми мікроклімату для приміщень з КТ

Пора року	Категорія робіт	Температура повітря, C^0, не більше	Відносна вологість повітря, %	Швидкість руху повітря, м/с
Холодна	легка – 1а	22 – 24	40 – 60	0,1
	легка – 1б	21 – 23	40 – 60	0,1
Тепла	легка – 1а	23 – 25	40 – 60	0,1
	легка – 1б	22 – 24	40 – 60	0,2

Рівні позитивних і негативних іонів у повітрі приміщень з ПК мають відповідати санітарно-гігієнічним нормам ГН 2152-80 (табл. 4.2).

Таблиця 4.2. Рівні іонізації повітря приміщень з КТ

Рівні іонізації повітря	Кількість іонів в 1 $см^3$ повітря	
	n+	n-
Мінімально необхідні	400	600
Оптимальні	1500 – 3000	3000 – 5000
Максимально допустимі	50000	50000

У приміщеннях з КТ має бути забезпечений 3-кратний обмін повітря за годину. Для забезпечення постійних параметрів мікроклімату (температури, вологості, швидкості руху і чистоти повітря) в приміщеннях можуть бути встановлені побутові кондиціонери.

4.2 Вимоги до освітлення приміщень та робочих місць

Приміщення з КТ повинні мати природне та штучне освітлення. При незадовільному освітленні зменшується продуктивність праці користувачів КТ[9]. Система освітлення повинна відповідати таким вимогам:

– освітленість на робочому місці повинна відповідати характеру зорової роботи, який визначається об’єктом розрізнення - найменшим розміром об’єкту, що розглядається на моніторі персонального комп’ютера;

– необхідно забезпечити достатньо рівномірне розподілення яскравості на робочій поверхні монітора, а також в межах навколишнього простору;

– на робочій поверхні повинні бути відсутні різкі тіні;

– в полі зору не повинно бути відблисків;

– величина освітленості повинна бути постійною під час роботи;

– оптимальної спрямованості світлового потоку і необхідного складу світла.

Природне освітлення в приміщеннях з КТ повинно відповідати вимогам ДСанПіН 3.3.2.007-98 та ДБН В 2.5.28-2006. Природне освітлення має здійснюватись через світлові прорізи, орієнтовані переважно на північ або північний схід і забезпечувати коефіцієнт природної освітленості (КПО) не нижче 1,5 %. Для захисту від прямих сонячних променів, які створюють прямі та відбиті відблиски на поверхні екранів і клавіатури, повинні бути передбачені сонцезахисні

пристрої, вікна повинні мати жалюзі або штори. Задовільне природне освітлення легше створити в невеликих приміщеннях на 5-8 робочих місць.

Штучне освітлення в приміщеннях з робочими місцями, обладнаними ПК, має здійснюватись системою загального рівномірного освітлення.

Значення освітленості на поверхні робочого столу в зоні розміщення документів має становити 300-500 лк. Якщо ці значення освітленості неможливо забезпечити системою загального освітлення, допускається використовувати місцеве освітлення. При цьому світильники місцевого освітлення слід встановлювати таким чином, щоб не створювати відблисків на поверхні екрана, а освітленість екрана має не перевищувати 300 лк[21].

Як джерела світла в штучному освітленні мають застосовуватись переважно люмінісцентні лампи типу ЛБ. Допускається застосування ламп розжарювання у світильниках місцевого освітлення.

Система загального освітлення має становити суцільні або преривчасті лінії світильників, розташовані збоку від робочих місць (переважно ліворуч), паралельно лінії зору працюючих. Для загального освітлення допускається використання світильників таких класів світлорозподілу:

– прямого світла - П;

– переважно відбитого світла - В.

Для загального освітлення можна застосовувати світильники серії ЛПО 36 із дзеркальними гратами, укомплектовані високочастотними пускорегулювальними апаратами (ВЧ ПРА) лише в модифікації

«Кососвітло». Застосовувати світильники без розсіювачів та крануючих гратів забороняється.

Яскравість світильників загального освітлення в зоні кутів випромінювання від 50° до 90° з вертикаллю в повздовжній та поперечній площинах повинна становити не більше ніж 200 кд/м2, захисний кут світильників - не менш, ніж 40°. Показник осліплености для джерел загального штучного освітлення у виробничих приміщеннях з ПК не повинен перевищувати 20, а показник дискомфорту - не більше 40.

У приміщеннях з КТ необхідно передбачити обмеження прямих відблисків від джерел природного та штучного освітлення та обмеження відбитих відблисків на робочих поверхнях (екран, стіл, клавіатура). Яскравість світлових поверхонь (вікна, джерела штучного освітлення тощо), що розташовані в полі зору, не повинна перевищувати 200 кд/м2. Яскравість відблисків на екрані ПК не повинна перевищувати 40 кд/м2, а яскравість стелі при застосуванні системи відбитого освітлення не повинна перевищувати 200 кд/м2. Захистом від прямих відблисків повинно бути зменшення яскравості видимої частини джерел світла, шляхом застосування спеціальних розсіювачів, відбивачів та інших світлозахисних

пристроїв, а також правильне розміщення робочих місць відносно джерел світла; від відбитих відблисків - правильне розміщення предметів, використання матових поверхонь предметів у приміщенні.

Необхідно обмежити нерівномірність розподілу яскравості в полі зору працюючих з ПК. Співвідношення яскравостей робочих поверхонь не повинно перевищувати 3:1, а співвідношення яскравостей робочих поверхонь та поверхонь стін, обладнання тощо -5:1.

Коефіцієнт запасу (Кз) для освітлювальних установок загального освітлення приймається рівним 1,4.

Величина коефіцієнта пульсації освітленості не повинна перевищувати 5%, що забезпечується застосуванням газорозрядних ламп у світильниках загального та місцевого освітлення з високочастотними пускорегулюючими апаратами (ВЧ ПРА) для світильників будь-яких типів. Якщо немає світильників з ВЧ ПРА, то лампи багатолампових світильників або світильники загального освітлення, розташовані поруч, слід вмикати на різні фази трифазної мережі.

Для забезпечення нормованих значень освітленості у приміщеннях з ПК слід чистити шибки і світильники не менше двох разів на рік і вчасно замінювати лампи, що перегоріли.

Нормованим параметром природного освітлення являється коефіцієнт природного освітлення (КПО), який визначається за формулою:

$$\text{КПО} = \left(\frac{\text{E}_{\text{внутр}}}{\text{E}_{\text{зовн}}}\right) * 100\%, \tag{4.1}$$

де:

$\text{E}_{\text{внутр}}$ — освітленість, що створюється в деякій точці приміщення світлом неба;

$\text{E}_{\text{зовн}}$ — зовнішня горизонтальна освітленість, створювана повністю відкритим склепінням.

В залежності від розряду виконуваних зорових робіт КПО має різні норми. Робота оператора ПК відноситься до робіт середньої точності (IУ розряд зорових робіт, мінімальний розмір об'єкту розрізнення складає 0,5 - 1,0мм), для яких при використанні бокового освітлення КПО=1,5%.

Для штучного освітлення нормованим параметром виступає $E_{мін}$ - мінімальний рівень освітленості, та $К_п$ - коефіцієнт пульсації світлового потоку, який не повинний бути більшим ніж 20%. У табл. 4.3 наведені Норми штучного та природного освітлення виробничих приміщень.

Таблиця 4.3 Норми штучного та природного освітлення виробничих приміщень

Характе-ристика зорових робіт	Наймен-ший розмір об'єкта розпіз-навання, мм	Розряд зорової роботи	Підрозряд зорової роботи	Штучне освітлення	Природне освітлення
				Освітленість, лк	КПО, %
				загальне освітлення	бокове освітлення
Середньої точності	0,5-1	IV	а б в г	300 200 200 200	1,5
Малої точності	1-5	V	а б в г	300 200 200 200	1,0
Груба	Більше 5	VI	–	200	1,0

Мінімальна освітленість встановлюється в залежності від розряду виконуваних зорових робіт. Для IУ розряда зорових робіт вона складає 300-500 лк.

На практиці використовують метод відносної площі світлових прорізів для розрахунку природного освітлення. Відносна площа світлових прорізів – це відношення площі вікон до площі підлоги приміщення, що освітлюється:

$$\alpha = \frac{S_{вікн}}{S_{підл}} * 100\%, \quad (4.2)$$

де α - відносна площа світлових прорізів;

$S_{вікн}$ - сумарна площа вікон у приміщенні, м^2;

$S_{підл}$ - площа підлоги у цьому ж приміщенні, м^2.

Норми штучного та природного освітлення виробничих приміщень за методом відносної площі світлових прорізів наведені у табл. 4.4.

Розрахуємо, якого розряду зорову роботу можна виконувати у приміщенні користувача ПК з однобічним природним освітленням. площа приміщення – 60 м^2; 3 вікна розміром 2×1,5 м кожне.

Таблиця 4.4 Рекомендовані значення відносної площі світлових прорізів α для виробничих приміщень

Розряд зорової роботи	Вид робіт за ступенем точності	α,%
II	Дуже високої точності	16-20
III	Високої точності	14-16
IV	Середньої точності	12-14
V	Малої точності	10-12
VI	Грубі	8-10

Спочатку визначаємо сумарну площу вікон (світлових прорізів):

$$S_{вік} = 3 \cdot 2 \cdot 1{,}5 = 9 \text{ м}^2.$$

Визначаємо відносну площу світлових прорізів за формулою (4.2):

$$\alpha = \frac{S_{\text{вікн}}}{S_{\text{підл}}} * 100\% = \frac{9}{60} * 100\% = 15\%$$

За даними табл. 4.2 визначаємо, що у приміщенні можна запланувати виконання зорової роботи високої точності, що відповідає III розряду зорової роботи, що є задовільними умовами для роботи за ПК.

4.3 Вимоги до вібрації

Під час виконання робіт з ПК у виробничих приміщеннях значення характеристик вібрації на робочих місцях мають не перевищувати допустимі відповідно до ДСанПіН 3.3.2.007-98, ДСН 3.3.6-039-99[11] (табл. 4.5).

Таблиця 4.5. Санітарні норми вібрації категорії 3 технологічної типу "в"

Середньо-геометричні частини смуг, Гц	Допустимі значення по осях X,Y,Z							
	віброприскорення				віброшвидкості			
	м/с^2		ДБ		м/сх 10^{-2}		ДБ	
	1/3 окт	1/1 окт	1/3 окт	1/1 окт	1/3 окт	1/1 окт	1/3 окт	1/1 окт
1,6	0,0125		32		0,13		88	
2,0	0,0112	0,02	31	36	0,089	0,18	85	91
2,5	0,01		30		0,063		82	
3,15	0,009		29		0,0445		79	
4,0	0,008	0,014	28	33	0,032	0,063	76	82
5,0	0,008		28		0,025		74	
6,3	0,008		28		0,02		72	
8,0	0,008	0,014	28	33	0,016	0,032	70	76
10,0	0,01		30		0,016		70	
12,5	0,0125		32		0,016		70	
16,0	0,016	0,028	34	39	0,016	0,028	70	75
20,0	0,0196		36		0,016		70	
25,0	0,025		38		0,016		70	
31,5	0,0315	0,056	40	45	0,016	0,028	70	75
40,0	0,04		42		0,016		70	
50,0	0,05		44		0,016		70	
63,0	0,063	0,112	46	51	0,016	0,028	70	75
80,0	0,08		48		0,016		70	
Кориговані в еквівалентні значення та їх рівні		0,014		33		0,028		75

4.4 Захист користувачів від впливу іонізуючих та неіонізуючих електромагнітних полів та випромінювання моніторів

Джерелом електромагнітних випромінювань є монітори на основі ЕПТ, системні блоки, принтер, клавіатура, численні з’єднувальні кабелі.

Монітори, сконструйовані на основі електронно-променевої трубки, є джерелами електростатичного поля, м'якого рентгенівського, ультрафіолетового, інфрачервоного, видимого, низькочастотного, наднизькочастотного і високочастотного електромагнітного випромінювання (ЕМВ).

Рентгенівське випромінювання виникає в результаті зіткнення пучка електронів із внутрішньою поверхнею екрана ЕПТ. Для ефективного поглинання рентгенівського випромінювання скляне покриття містить свинець. Зазвичай скло кінескопа непрозоре для рентгенівського випромінювання, при значенні прискорюючої анодної напруги менше 25 кВ енергія рентгенівського випромінювання майже повністю поглинається склом екрана.

У нормально працюючого монітора з ЕПТ рівні рентгенівського

випромінювання не перевищують рівень звичайного фонового випромінювання, набагато нижче допустимого рівня. Із збільшенням відстані інтенсивність випромінювання зменшується в геометричній прогресії.

Джерелом електростатичного поля є додатній потенціал, який подається на внутрішню поверхню екрана для прискорення електронного променя. Значення прискорюючої анодної напруги для кольорових моніторів може досягати 18 кВ. Із зовнішньої сторони до екрана притягаються негативні частинки з повітря, що при нормальній вологості мають певну провідність. Якщо зовнішня поверхня екрана заземлена, тоді негативний заряд на ній зменшує напруженість електростатичного поля на 0-50% для сухого повітря і більше ніж на 50% для вологого.

Екрани, які випускалися останнім часом, виготовлялися з провідних матеріалів, тому електростатичний потенціал на їх поверхні практично нульовий.

Рідкокристалічні монітори формують зображення методом, який принципово різниться від ЕПТ. Тому проблем рентгенівського

випромінювання і статичного заряду на поверхні екрану у них просто не існує.

Джерелами ЕМВ є блоки живлення від мережі (частота 50 Гц), система кадрової розгортки (5Гц -2 кГц), система рядкової розгортки (2-400 кГц), блок модуляції променя ЕПТ (5-10 мГц). Електромагнітне поле має електричну (Е) і магнітну (Н) складові, причому взаємозв'язок їх достатньо складний. Оцінка складових електричного і магнітного поля здійснюється окремо.

Електромагнітні випромінювання наднизької частоти не викликають

іонізацію, а, відповідно, і мутації. їх дія на живі клітини мало вивчена. Всі роботи, що доводять шкоду електромагнітного поля наднизької частоти, спираються на епідеміологічні дані. Це означає, що тут можуть бути неточності, не виключені впливи інших чинників. Конкретного, ушкоджувального механізму дії електромагнітного поля наднизької частоти не встановлено.

Негативний вплив електромагнітного поля низьких частот на людину

виявляється наступним чином. Випромінювання низької частоти в першу чергу негативно впливає на центральну нервову систему, викликаючи головний біль, запаморочення голови, нудоту, депресію, безсоння, відсутність апетиту, виникнення синдрому стресу.

Низькочастотне електромагнітне поле може бути причиною шкірних

захворювань (висипка, себороїдна екзема, рожевий лишай і ін.), хвороб серцево- судинної системи і кишково-шлункового тракту; воно впливає на білі кров'яні тільця, що призводить до виникнення пухлин, у тому числі і злоякісних. Електростатичне поле великої напруженості здатне змінювати і переривати клітинний розвиток, а також викликати катаракту з наступним помутнінням кришталика.

Електромагнітні поля наднизької частоти очевидно не представляють загрози для здоров'я людини, проте, внаслідок того, що їх дія мало вивчена,

рекомендується зменшити або звести їх до мінімуму. Попри безліч досліджень, експерти не можуть прийти до однієї думки про наявність і масштаби шкоди електромагнітного випромінювання. Хоча багато користувачів ПК скаржаться на симптоми, які важко пояснити за рахунок інших робочих чинників. Тому, до одержання нових знань в цій області, вирішено обмежувати рівні випромінювань настільки, наскільки це можливо.

Рівні електромагнітних випромінювань моніторів, що вважаються безпечними для здоров'я, регламентуються нормами MPR II 1990:10 Шведського національного комітету з вимірів та випробовувань, що вважаються базовими, і більш жорсткими нормами ТСО'91, ТСО'92, ТСО'95, ТСО'99 та ТСО'ОЗ Шведської конфедерації профспілок.

Українські нормативні документи НПАОП 0.00-1.31-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин» та ДСанПіН 3.3.2.007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними термінаналами електронно-обчислювальних машин», повністю збігаються у частині рівнів ЕМВ з вимогами MPR II.

Напруженості електромагнітного поля з електричною та магнітною складовою в діапазонах частот 5 Гц-2 кГц та 2 кГц - 400 кГц, а також величина електростатичного потенціалу, згідно вимог стандартів MPR II, ТСО'91, ТСО'92, ТСО'95, ТСО'99, ТСО'ОЗ, ТСО'05, ТСО'Об та ДСанПіН 3.3.2.007-98 не повинні перевищувати значень, які наведено в таблиці 4.6.

Відповідно до ДСанПіН 3.3.2.007-98 потужність дози рентгенівського випромінювання на відстані 5 см від екрану та інших поверхонь ПК не повинна перевищувати 100 мкР/год; а інтенсивність ультрафіолетового випромінювання на відстані 0,3 м від екрану не повинна перевищувати в діапазоні довжин хвиль 400-320 нм -2 Вт/м2, 320-280 нм -0,002 Вт/ $м^2$; а ультрафіолетового випромінювання в діапазоні 280-200 нм - не повинно бути.

Для моніторів, які не відповідали нормам, в 80-90-х роках минулого століття рекомендувалося встановити захисний фільтр для екрана, що

послаблює змінне електричне й електростатичне поля. При виготовленні фільтрів використовувалися скло, пластмаса, сітки. Фільтри на базі цих матеріалів послаблюють електричне поле до 1% від початкового рівня (послаблення електричного поля на частоті 100Гц - 90% для сітчастих,77% - для скляних, 1% - рідинних електролітичних. На частоті 20кГц - 99% - 97% для скляних та пластмасових, 6% - для рідинних електролітичних). Крім того, використання фільтрів створював психологічний комфорт. Важливо було, щоб фільтр мав антивідблискові властивості.

Таблиця 4.6. Вимоги національних та міжнародних стандартів на рівні випромінювань

Стандарт	Напруженість змінного електричного поля для діапазонів, В/м		Напруженість змінного магнітного поля для діапазонів, нТл		Електро-статичний потенціал, В
	5Гц-2кГц	2кГц-400кГ ц	5Гц-2кГц	2кГц-400кГ ц	
MPR II	<25	<2,5	<250	<25	<500
TCO ‘92, TCO ‘95 , TCO ‘99 , TCO‘03,TCO ‘05,TCO‘06	<10	<1,0	<200	<25	<500
ДСанПіН 3.3.2.007-98	<25	<2,5	<250	<25	<500

Важливо було, щоб фільтр мав антивідблискові властивості.

При використанні фільтрів виникало чимало проблем. Результати досліджень шести захисних екранів різних фірм показали, що рівень електричного поля зменшувався майже на 95%, проте жоден екран не забезпечував достатнього зменшення інтенсивності слабких магнітних полів. Якщо блокуються електричні поля в діапазоні низьких частот та дуже низьких частот, тоді не має жодного впливу на магнітні поля цих частот. Таке спостерігалося і в інших діапазонах частот. Встановлення фільтрів на екранах ЕПТ дозволяло зменшити електричну складову ЕМП безпосередньо біля екрану, але внаслідок

перерозподілу поля це призводило до збільшення його на відстанях більше 1-1,5 м від екрану по осі ЕПТ і зміни співвідношення полів по боках від нього. Тобто, жоден захисний екран, що пропонувався, не поглинав повністю обидва типи випромінювань - електричні і магнітні.

Одним з найкращих фільтрів для виконання всіх п'яти вимог директиви ЕС 90/270/ЕЕС до моніторів було використання екранного фільтру Polaroid з коловим поляризатором (СР-фільтр), який складається з двох активних компонентів (шарів) -лінійного поляризатора і оптичного компенсатора (табл. 4.7).

Таблиця 4.7. Вимоги станда зту MPR до значень параметрів фільтрів екранів

Вид випромінювання	Залишкове значення після поглинання
Ренгенівське випромінювання	< (70-40%) в залежності від величини випромінення
Електростатичний потенціал	< (10%+ 100 В)
Електричне поле (напруженість) 0,005 - 2 кГц 2 - 400 кГц	< (10%+ 1,5 В/м) <(10%+ 0,1 В/м)
Магнітне поле (індукція) 0,005 - 2 кГц 2-400 кГц	< (10% + 30 нТ) <(10% + 1,5 нТ)

СР-фільтр Polaroid поглинає ультрафіолетове випромінювання, збільшує різкість символів, зменшує блимання екрану та відзначається порівняно високою ціною. Для здійснення колективного захисту, якщо сусідні робочі місця потрапляли у зону впливу поля (на відстані 1,2-2,5 м від дисплея) - встановлювали захисне покриття задньої і бічних стінок, монтували спеціальні екрануючі панелі, на задню і бічні поверхні монітора, встановлювали перегородки між різними користувачами.

Висновки до розділу 4

Під час виконання спеціальної частини з охорони праці було досліджено санітарно – гігієнічні умови при роботі з комп'ютерною технікою, а саме:

1. Наведені основні вимоги до мікроклімату.

2. Розглянуті вимоги до освітлення приміщень, робочих місць.
3. Проаналізовані вимоги до вібрації.
4. Дослідження методів захисту користувачів від впливу іонізуючих та неіонізуючих електромагнітних полів та випромінювання моніторів.

Також був проведений розрахунок коефіцієнту природного освітлення у приміщенні. У результаті розрахунків у приміщенні можна запланувати виконання зорової роботи високої точності, що відповідає III розряду зорової роботи, що є задовільними умовами для роботи за комп’ютерною технікою.

Вимоги, які прописані у нормативних документах є актуальними на сьогоднішній день, бо збільшується кількість працівників які працюють за комп’ютерами.

Вимоги та нормативи, щодо охорони праці мають дотримуватися на підприємствах під час трудової діяльності людини. Дотримання поставлених вимог до працівників та власників підприємств дозволить мінімізувати шкідливі наслідки, які мають вплив на здоров’я людини.

ВИСНОВКИ

В умовах іформатизації суспільства існує необхідність розвитку інформаційних систем для підтримки продаж у мережі Інтернет автозапчастин для спортивних автомобілів, оскільки це поліпшує ведення бізнесу та задовольняє попити покупців, які мають спортивні авто.

Проведене дослідження дозволяє зробити наступні висновки.

Здійснений аналіз показав, що сучасний стан інформатизації суспільства супроводжується перенесенням реалізації послуг із продажу товарів у мережу Інтернет. У таких умовах існує потреба у впровадженні інформаційних систем підтримки діяльності магазинів із продажу товарів та автозапчастин зокрема. Наявність таких систем сприяє підвищенню ефективності роботи працівників магазину по роботі з клієнтами, підвищує продаж товару та просування послуг у мережі Інтернет. Виявлено, що сьогодні існує багато кросплатформених сервісів для підтримки діяльності автомагазинів, однак спеціалізовані магазини із продажу автозапчастин спортивних автомобілів відсутні. Такі автозапчастини продають також на торгових майданчиках, які сфокусовані не тільки спортивних автозапчастинах. Це обумовлює необхідність створення спеціалізованої інформаційної системи магазину автозапчастин спортивних автомобілів.

Установлено, що роботу з клієнтами магазину та підвищення продуктивності роботи магазину можна поліпшити, надаючи рекомендації покупцями, які уже зробили певний вибір для подальших покупок у магазині. Такий механізм можна реалізувати шляхом пошуку асоціативних правил з використанням алгоритму Apriori.

Серед інструментальних засобів розробки інформаційної системи для розробки системи було обрано мову програмування C#; платформу .NET Core, яка має можливість використання хмарних технологій, функціонал для кросплатформенності і модульності. Також було використано кросплатформений фреймворк ASP.NET Core в комбінації з патерном MVC, який у свою чергу для

виведення даних використовує Razor pages у комбінації з HTML та CSS. Для роботою з бд використовувся фреймворк Entity Framework Core, який використовує підходи ORM та підтримує міграції, що дозволяє легко маніпулювати бд без втрати даних. У якості інтегрованого середовища розробки було обрано Visual Studio.

У результаті проведеного дослідження було здійснено розробку та програмну реалізацію інформаційної системи магазину автозапчастин спортивних автомобілів, яка дозволяє покупцям переглядати товари та сортувати їх за категоріями, формувати список товарів які зацікавили покупця, формувати кошик та оформлення замовлення із використанням рекомендацій, сформованих із використанням алгоритму Apriori. Продавець має можливість додавати нові товари, переглядати усі товари та редагувати їх та також отримувати статистику по продажам товарів, виявляючи асоціативні правила для оптимізації та покращення продажу.

У спеціальному розділі було проаналізовано санітарно-гігієнічні вимоги приміщення для роботи з комп’ютерною технікою.

Поставлені завдання виконано повністю, однак є питання, які потребують подальшої розробки: налаштування більш глибокого динамічного аналізу зібраної інформації, прогнозування підвищення продажів у різні сезони року або від запланованих спортивних змагань, оскільки ці фактори впливають на попит певної категорії товарів серед покупців.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. OTT Streaming Market Global Briefing 2020-30: Covid 19 Growth And Change. *The Business Research Company*, 2020. 18 с.

2. Introduction to recommender systems : вебсайт. URL: https://towardsdatascience.com/introduction-to-recommender-systems6c66cf15ada (дата звернення: 10.03.2022).

3. Schafer J. Ben. Recommender Systems in E-Commerce / J. Ben Schafer, Joseph Konstan, John Riedl // GroupLens Research Project Book. – Minneapolis, Minnesota, USA: University of Minnesota, 2011. 158-166 с.

4. Aggarwal C.. Recommender Systems: The Textbook. Springer, 2016. 518 с.

5. Sarwar B. Item-Based Collaborative Filtering Recommendation Algorithms / Sarwar B., Karypis G., Konstan J., and Riedl J. 2001. 285-289 с.

6. Топ-10 data mining-алгоритмів простою мовою : вебсайт. URL: https://habr.com/ru/company/iticapital/blog/262155 (дата звернення: 10.02.2022).

7. Методи пошуку асоціативних правил : вебсайт. URL: https://intuit.ru/studies/courses/6/6/lecture/186 (дата звернення: 18.03.2022).

8. Загальна інформація ASP.NET Core MVC : вебсайт. URL: https://docs.microsoft.com/ru-ru/aspnet/core/mvc/overview?view=aspnetcore-6.0 (дата звернення: 28.02.2022)

9. Вступ до Razor Pages в ASP.NET Core : вебсайт. URL: https://docs.microsoft.com/ru-ru/aspnet/core/razor-pages/?view=aspnetcore-6.0&tabs=visual-studio (дата звернення: 12.03.2022).

10. Лок Э. ASP.NET Core в действии, СПб: ДМК-Пресс, 2021, 906 с.

11. Фримен А. Professional ASP.NET MVC 5 с примерами на C# для профессионалов. М: Вильямс, 2018, 736 с.

12. Скит Дж. C# для профессионалов. Тонкости программирования. М: Вильямс, 2018, 608 с.

13. Фримен А. Pro ASP.NET Core 6: Develop Cloud-Ready Web Applications Using MVC, Blazor, and Razor Pages. Publisher: APress, 2022. 1253 p.

14. Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями. URL: http://zakon3.rada.gov.ua/laws/show/z0508-18 (дата звернення: 13.05.2022).

15. ДСТУ 2293-99 Охорона праці. Терміни та визначення основних понять. Київ, 1999.

16. Москальова В. М. Основи охорони праці. Підручник. - Київ: ВД Професіонал, 2005. 666 с.

17. Гандзюк М. П., Желібо Е. П., Халімовський М. О. Основи охорони праці / За ред.. Гандзюка М. П. - К.: Каравела 2003. 405 с.

18. Ткачук К. Н., Халімовський М. О., Зацарний В.В., та інші. Основи охорони праці: Підручник. -К.: Основа, 2006. 444 с.

19. Жидецький В.Ц. Основи охорои праці: Підручник. - К.: Основа, 2002. 320 с.

20. Державні санітарні правила і норми роботи з ВДТ ЕОМ ДСанПІН 3.3.2.007-98: вебсайт. URL: http://mozdocs.kiev.ua/view.php?id=2445 (дата звернення: 13.04.2022).

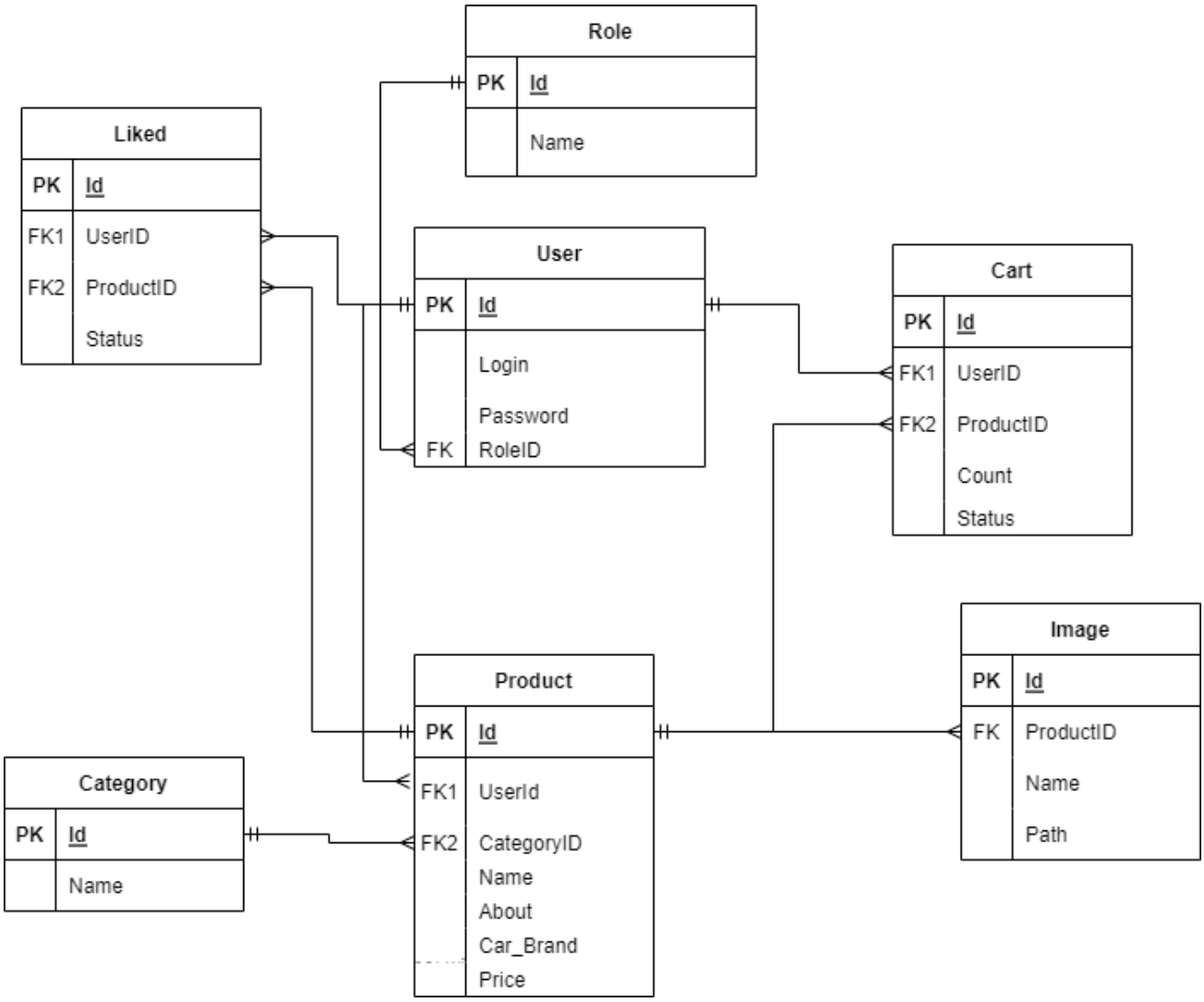
21. СНиП II-4-79. Природне і штучне освітлення: вебсайт URL: https://dnaop.com/html/45036/doc- СНиП_II-4-79 (дата звернення: 13.05.2022). .

22. ДСН 3.3.6.042-99. Санітарні норми мікроклімату виробничих приміщень. : вебсайт URL: https://dnaop.com/html/34094/doc-ДСН_3.3.6.042-99 (дата звернення: 23.05.2022).

23. СН 3044-84. Санітарні норми вібрації робочих місць: вебсайт URL: https://dnaop.com/html/43248/doc -ДНАОП_3044-84 (дата звернення: 10.05.2022).

ДОДАТОК А

Структура бази даних



ДОДАТОК Б

Допоміжні моделі для роботи з продавцем

Модель реєстрації:

```
public class RegisterModel
  {
    [Required(ErrorMessage = "Fill the line")]
    public string Login { get; set; }

    [Required(ErrorMessage = " Fill the line ")]
    [DataType(DataType.Password)]
    public string Password { get; set; }

    [DataType(DataType.Password)]
    [Compare("Password", ErrorMessage = " Put correct password")]
    public string ConfirmPassword { get; set; }
  }
```

Модель входу до системи:

```
public class LoginModel
  {
    [Required(ErrorMessage = " Fill the line ")]
    public string Login { get; set; }

    [Required(ErrorMessage = " Fill the line ")]
    [DataType(DataType.Password)]
    public string Password { get; set; }
  }
```

ДОДАТОК В

Лістинг коду виглядів для роботи з продавцем

Вигляд реєстрації:

```
<form asp-action="Register" asp-controller="Account" asp-anti-forgery="true">
   <div class="validation" asp-validation-summary="ModelOnly"></div>
   <div>
      <div>
         <label asp-for="Login">Enter your Login</label><br />
         <input type="text" asp-for="Email" />
         <span asp-validation-for="Email" />
      </div>
      <div>
         <label asp-for="Password"> Enter your Password </label><br />
         <input asp-for="Password" />
         <span asp-validation-for="Password" />
      </div>
      <div>
         <label asp-for="ConfirmPassword"> Enter password again</label><br />
         <input asp-for="ConfirmPassword" />
         <span asp-validation-for="ConfirmPassword" />
      </div>
      <div>
         <input type="submit" value="Register" />
      </div>
   </div>
```

Вигляд входу:

```
<a asp-action="Register" asp-controller="Account">Register</a>

<form asp-action="Login" asp-controller="Account" asp-anti-forgery="true">
   <div class="validation" asp-validation-summary="ModelOnly"></div>
   <div>
      <div class="form-group">
         <label asp-for="Login"> Enter your Login </label>
         <input type="text" asp-for="Email" />
         <span asp-validation-for="Email" />
      </div>
      <div class="form-group">
         <label asp-for="Password"> Enter your Password </label>
         <input asp-for="Password" />
         <span asp-validation-for="Password" />
      </div>
      <div class="form-group">
         <input type="submit" value="Enter" class="btn" />
      </div>
   </div>
```

ДОДАТОК Г

Контролер кошика покупця

```
public class ShoppingBasketController : Controller
  {
    private IPartRepository repository;
    private IOrder order;

    public ViewResult Checkout()
    {
      return View(new ShippingDetails());
    }

    [HttpPost]
    public ViewResult Checkout(ShoppingBasket basket, ShippingDetails details)
    {
      if(basket.Lines.Count() == 0)
      {
        ModelState.AddModelError("", "Your shopping basket is empty");
      }

      if(ModelState.IsValid)
      {
        order.ProcessOrder(basket, details);
        basket.Clear();
        return View("Completed");
      }
      else
      {
        return View(details);
      }
    }

    public ViewResult Index(ShoppingBasket shoppingBasket, string return_Url)
    {
      return View(new ShoppingBasketIndexViewModel
      {
        Basket = shoppingBasket,
        ReturnUrl = return_Url
      });
    }

    public ShoppingBasketController(IPartRepository repo, IOrder processor)
    {
      repository = repo;
      order = processor;
    }
```

```
        public RedirectToRouteResult AddToBacket(ShoppingBasket shoppingBasket,int part_id, string
return_Url)
        {
            Part part = repository.Parts
                .FirstOrDefault(p => p.Part_Id == part_id);

            if(part != null)
            {
                shoppingBasket.Add_Item(part, 1);
            }
            return RedirectToAction("Index", new { return_Url });
        }

        public RedirectToRouteResult RemoveFromBasket(ShoppingBasket shoppingBasket, int
part_id, string return_Url)
        {
            Part part = repository.Parts
                .FirstOrDefault(p => p.Part_Id == part_id);

            if( part != null)
            {
                shoppingBasket.Remove_Item(part);
            }
            return RedirectToAction("Index", new { return_Url });
        }

         public PartialViewResult Summary(ShoppingBasket shoppingBasket)
        {
            return PartialView(shoppingBasket);
        }

    }
```

ДОДАТОК Д

Клас для фільтрації по категоріям

```
public static class PagingHelper
  {
    public static MvcHtmlString Page_Links(this HtmlHelper html, PaginInfo paginInfo,
        Func<int,string> page_Url)
    {
        StringBuilder rez = new StringBuilder();
        for(int i = 1; i<= paginInfo.Total_Pages; i++)
        {
            TagBuilder tag = new TagBuilder("a");
            tag.MergeAttribute("href", page_Url(i));
            tag.InnerHtml = i.ToString();
            if(i == paginInfo.Current_Page)
            {
                tag.AddCssClass("selected");
                tag.AddCssClass("btn-primary");
            }
            tag.AddCssClass("btn btn-default");
            rez.Append(tag.ToString());
        }
        return MvcHtmlString.Create(rez.ToString());
    }
```

ДОДАТОК Е

Клас для відправки листа про замовлення

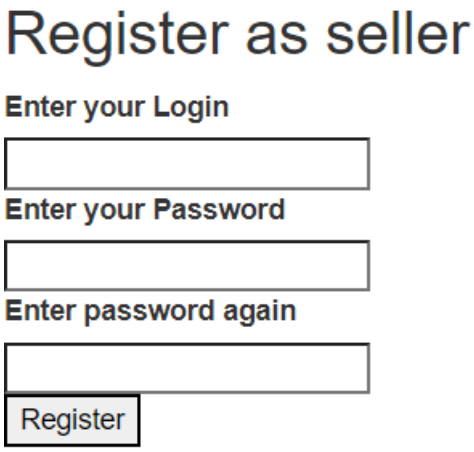
```
public class EmailSettings
  {
    public string MailToAddress = "ilgyha228@gmail.com";
    public string MailFromAddress = "store@example.com";
    public bool UseSsl = true;
    public string Username = "MySmtpUsername";
    public string Password = "MySmtpPassword";
    public string ServerName = "smtp.example.com";
    public int ServerPort = 587;
    public bool WriteAsFile = true;
    public string FileLocation = @"D:\email";
  }

 public  class EmailOrder : IOrder
  {
    private EmailSettings emailSettings;

    public EmailOrder(EmailSettings settings)
    {
      emailSettings = settings;
    }

    public void ProcessOrder(ShoppingBasket shoppingBasket, ShippingDetails shippingDetails)
    {
      using (var smtpClient = new SmtpClient())
      {
        smtpClient.EnableSsl = emailSettings.UseSsl;
        smtpClient.Host = emailSettings.ServerName;
        smtpClient.Port = emailSettings.ServerPort;
        smtpClient.UseDefaultCredentials = false;
        smtpClient.Credentials =
          new NetworkCredential(emailSettings.Username, emailSettings.Password);

        if(emailSettings.WriteAsFile)
        {
          smtpClient.DeliveryMethod = SmtpDeliveryMethod
            .SpecifiedPickupDirectory;
          smtpClient.PickupDirectoryLocation =
            emailSettings.FileLocation;
          smtpClient.EnableSsl = false;
        }

        StringBuilder body = new StringBuilder()
          .AppendLine("New order checked")
          .AppendLine("-----")
```

```
            .AppendLine("List of items");

        foreach(var line in shoppingBasket.Lines)
        {
            var subtotal = line.Part.Price * line.Quantity;
            body.AppendFormat("{0} x {1} (total: {2:c}", line.Quantity,
                line.Part.Name, subtotal);
        }

        body.AppendFormat("Total cost: {0:c}", shoppingBasket.Total_Value())
            .AppendLine("-----")
            .AppendLine("Delivery")
            .AppendLine(shippingDetails.Name)
            .AppendLine(shippingDetails.Line1)
            .AppendLine(shippingDetails.Line2 ?? "")
             .AppendLine(shippingDetails.Line3 ?? "")
             .AppendLine(shippingDetails.City)
             .AppendLine(shippingDetails.Country)
             .AppendLine("-----")
             .AppendFormat("C.O.D.: {0}", shippingDetails.Cash_on_delivery ? "yes" : "no")
             .AppendFormat("Payment: {0}", shippingDetails.Payment ? "yes" : "no");

        MailMessage mailMessage = new MailMessage(
            emailSettings.MailFromAddress,
            emailSettings.MailToAddress,
            "Order shipped",
            body.ToString());

        if(emailSettings.WriteAsFile)
        {
            mailMessage.BodyEncoding = Encoding.UTF8;
        }
        smtpClient.Send(mailMessage);
    }
}
                                                    }
```

ДОДАТОК Є

Форми для реєстрації та входу

Форма для реєстрації:



Форма для входу: