Хмельницький національний університет
Факультет програмування та комп'ютерних і телекомунікаційних систем
Кафедра комп'ютерних наук та інформаційних технологій

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему ***«Інтернет-магазин одягу з експертною системою підбору товару»***

Галузь знань: *12 – Інформаційні технології*
Шифр і назва галузі знань

Спеціальність: *122 – Комп'ютерні науки*
Шифр і назва спеціальності

Освітня програма: *Комп'ютерні науки*
Назва освітньої програми

Виконав: *студент 4 курсу, група КН-17-1* — Підпис — *Б.В. Семенюк*
Курс, група виконавця; Підпис; Ініціали, прізвище

Керівник: *к.фіз.-мат.н. кафедри КНІТ* — Підпис — *В.Ц. Міхалевський*
Науковий ступінь, посада; Підпис; Ініціали, прізвище

Нормоконтроль: *к.т.н., доцент кафедри КНІТ* — Підпис — *Р.О. Багрій*
Науковий ступінь, посада; Підпис; Ініціали, прізвище

До захисту допускаю:
Зав. кафедри КНІТ, д.т.н., професор — Підпис — *О.В. Бармак*
Підпис; Ініціали, прізвище

09 червня 2021 р.

Хмельницький 2021

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет *програмування та комп'ютерних і телекомунікаційних систем*
Кафедра *комп'ютерних наук та інформаційних технологій*
Освітній ступінь *бакалавр*
Галузь знань *12 – Інформаційні технології*
Спеціальність *122 – Комп'ютерні науки*

ЗАТВЕРДЖУЮ
Завідувач кафедри *комп'ютерних наук та інформаційних технологій*

(підпис)
д.т.н., професор О.В. Бармак
«*08*» *лютого* *2021* року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

1. Тема кваліфікаційної роботи бакалавра: ***«Інтернет-магазин одягу з експертною системою підбору товару»***

2. Завдання видано студентці *Семенюку Богдану Васильовичу*
(прізвище, ім'я, по батькові)

3. Керівник роботи *доцент кафедри КНІТ Міхалевський Віталій Цезарійович*
(посада, прізвище, ім'я, по батькові)

4. Затверджено наказом університету від «*05*» *лютого* *2021* р. № *11*

5. Зміст пояснювальної записки (перелік задач) та вихідні дані:

Мета роботи – розробка web-сайту інтернет-магазину одягу, з реалізацією автоматичного оброблення запитів клієнтів з відповідною бізнес-логікою та консольний сервіс для підбору одягу даного магазину. Створений web-додаток дозволяє спростити та автоматизувати продаж одягу у магазині, також полегшити процес підбору розміру при покупці через інтернет. Розроблений програмний продукт призначений працівників магазину та покупців.

Виконавець: *студент 4 курсу, група КН-17-1* *Б.В. Семенюк*
Курс, група виконавця | Підпис | Ініціали, прізвище

Керівник: *к.фіз.-мат.н. кафедри КНІТ* *В.Ц. Міхалевський*
Науковий ступінь, посада | Підпис | Ініціали, прізвище

Анотація

Тема кваліфікаційної роботи бакалавра: *«Інтернет-магазин одягу з експертною системою підбору товару»*

Автор кваліфікаційної роботи бакалавра: *Семенюк Богдан Васильович*

Керівник кваліфікаційної роботи бакалавра: *доцент кафедри КНІТ, доцент Міхалевський Віталій Цезарійович*

Кваліфікаційна робота бакалавра містить:

Пояснювальна записка				Кількість додатків
Сторінок	Рисунків	Таблиць	Джерел інформації	
59	36	12	23	2

Метою кваліфікаційної роботи бакалавра є створення web-застосування магазин одягу та сервісу для підбору одягу. Для написання програмних продуктів було обрано технологію ASP.NET MVC 5 та ASP.NET Core Web Api, мову програмування C#.

Результатом виконання кваліфікаційної роботи бакалавра є web-сайт інтернет-магазин одягу, з реалізацією автоматичного оброблення запитів клієнтів з відповідною бізнес-логікою та консольний сервіс для підбору одягу даного магазину. Створений web-додаток дозволяє спростити та автоматизувати продаж одягу у магазині, також полегшити процес підбору розміру при покупці через інтернет. Розроблений програмний продукт призначений працівників магазину та покупців.

Виконавець: *студент 4 курсу, група КН-17-1* _____ *Б.В. Семенюк*

Курс, група виконавця — Підпис — Ініціали, прізвище

Зміст

Перелік скорочень

Скорочення, термін, позначення	Пояснення
АІС	Автоматизована інформаційна система
БД	База даних
ІТ	Інформаційні технології
ККП	Комплексний курсовий проект
КН	Комп’ютерні науки
ПП	Програмний продукт
СКБД	Система керування базами даних
ЕК	Електронна комерція
МО	Магазин одягу

Вступ

Важливою та суттєвою перевагою електронної комерції [1] є масштабне пониження затрат на оформлення угоди та її подальше обслуговування. Саме через бізнес-процеси, які мають змогу бути переведеними до електронної бази, мають потенціал зниження витрат на них же, як наслідок у свою чергу призводить до зниження собівартості різних груп товарів чи послуг. Найбільш розповсюдженим прикладом діяльності електронної комерції є інтернет-магазин, який, за звичаєм, представляє із себе веб-застосунок, до складу сутностей та функцій якого входять: вітрина товарів, маніпулювання корзиною, створення та оплата замовлень.

Магазин одягу [2] – магазин, який основним родом своєї діяльності займається та спеціалізується на продажу одягу. Звичайно ж він супроводжується такими сутностями як бренди, сети, колекції, продажі, тощо. Отже, кращого обліку цих речей в магазині, пропонується створити облік в електронному вигляді.

В свою чергу, застосування електронного обліку, несе за собою ряд переваг, одна з яких це – автоматизація бізнес-процесів, що у свою чергу дає можливість прискорити та полегшити роботу з магазином. Дозволить легко регулювати продаж товару (видаляти товар з вітрини, коли той був проданий). Також система дозволить додавати, видаляти та змінювати дані про працівників, клієнтів. Дозволить отримувати потрібну інформацію, натисканням однієї кнопки. Наприклад, можна з легкістю вивести список клієнтів.

Великим недоліком при здійснені покупки в інтернет-магазині одягу є процес підбору одягу. Зазвичай від представляє собою вимірювання власних показників та порівняння з вказаною розмірною сіткою певного товару.

Експертна система підбору розміру одягу повинна спростити цей процес. Тобто покупцю більше не потрібно порівнювати свої дані зі вказаними виробником, а лише ввести свої дані у відповідні поля та отримати відповідні варіанти розміру з відсотком влучності вибору того чи іншого розміру.

1 Характеристика предметної області і постановка задачі

1.1 Аналіз предметної області

Електронна комерція [1] – це сфера діяльності цифрової економіки, що підпорядковує переважну більшість грошових переказів та фінансових транзакції, безпосереднє здійснення яких, відбувається через інформаційні засоби (апаратні та програмні) та бізнес-процеси, пов'язані з проведенням цих транзакцій.

Як правило, електронна комерція дозволяє спростити процес торгівлі або надання послуг виробником для споживача. Такий процес можливий лише завдяки використанню інтернет-технологій, які в свою чергу, значно покращують загальний показник ефективної, лінійної взаємодії зі фінальним споживачем, через це фірми отримують змогу змінити собою роль постачальників, отже і зменшити витрати. Не менш важливим є накопичення інформації про процес продажу і клієнтів та вільний доступ до цих даних, це забезпечує змогу до широкого аналізу усіх ланок роботи бізнесу та можливих прогалин у маркетингу. Що у майбутньому дасть змогу покращити необхідні процеси для покращення прибутку та покращити спроможність конкуренції.

На сьогоднішніх реаліях, обмін різного роду контенту: медіа-контенту (фільми, музика, новини тощо), важливої інформації, освітніх матеріалів, а також використання комерційними компаніями для забезпечення продажу їх продукції, є основним родом використання електронного середовища.

Види електронної комерції:

1. Бізнес до бізнесу (B2B).

Електронна комерція бізнес до бізнесу – це один із видів електронної комерції, де взаємодія відбувається між компаніями. Даний тип комерції, який займається, безпосередньо, відносинами між видами комерційної діяльності. У порівнянні з B2C, має значно більші масштаби діяльності та потенціали росту, нараховується близько 80 % електронної комерції відносяться до типу бізнес до бізнесу.

2. Бізнес до споживача (B2C).

Бізнес до користувача, або ж торгівля між клієнтами та компаніями. Займає друге місце по своїй величині та є першим типом прояву комерції. Вміщає в себе, купівлю клієнтами фізичних речей чи товарів, які несуть інформаційну цінність та звичайно ж продаж цих речей бізнесом.

3. Споживач до споживача (C2C).

Споживач до споживача – тип комерцію при якому торгівля відбувається між споживачами або фізичними особами. Даний вид завдячує ростом електронним ринкам, так званим аукціонам (автомобілів, нерухомості. тощо) та своєрідним платформам перепродажу бувших у вжитку товарів.

4. Мобільна торгівля (m-commerce).

Мобільна комерція (торгівля) – у цьому виді продаж та закупівля послуг ж товарів здійснюється через бездротову технологію, а саме, різного роду гаджетів, мобільних телефонів, планшетів та інших переносних кишенькових засобів доступу до мережі інтернет.

Інтернет-магазин [2] – веб-аналогія звичайним магазинам, яка займає місце діяльності в інтернеті. На платформах даного проходить процес, безпосередньої, продаж продукції кінцевому споживачеві, до цього переліку входить доставка. Варто зазначити, що ці операції та інші які стосуються обробки та збереження особистої інформації користувача відбуваються на стороні ресурсу, тобто інтернет-магазину.

Електронний магазин [3] – сайт, який слугує для обробки різного роду замовлень користувачів, грошових переказів, доставки заказів, підтримка в реальному часі. Великою перевагою веб-додатків магазинів над реальними є вільних доступ з будь-якого місцю та надійність передачі даних завдяки протоколам – https, та іншим засобам безпеки.

Для замовлення товару в інтернет-магазину, клієнту необхідно додати товар до корзини, після чого користувач має можливість продовжити покупки або зробити замовлення (заповнення форми своїми даними).

Відповідно, робота адміністратора інтернет-магазину полягає у додаванні, редагуванні та видаленні товару.

Одяг [4] – результат діяльності людини, який створюється з ціллю захисту та покриття тіла. За звичай виконаний із шкіри, хутра, тканини, тощо та за допомогою різних станків для надання виробу бажаної форми.

Покупець [5] – особа, як фізична, так і юридична, яка має змогу здійснювати оплату грошима та стає власником товару, який купує. В даному випадку покупці мають змогу здійснювати покупки різними способами оплати.

Бренд [6] – торгова марка, яка представляє покупцю унікальну комбінацію цінностей, за яку ж споживач і повинен переплатити. Такий прояв діяльності як бренд та брендовий одяг несе за собою додаткові переваги та привілеї, які свою чергу надають, як правило, унікальні емоції. Та слід розуміти, що бренд і торгова марка це не одне і теж, торгова марка – має високий товарообіг, а бренд – має мати високий прибуток.

Розміри одягу – це так званий ідентифікатор параметрів, згідно до замірів людського тіла та його різних частин, як правило він буває у цифровому або буквеному представленні. Для більшого розуміння покупцем який йому розмір потрібний та універсальності процесу підбору виробники використовують системи визначення розмірів. По замовчування розмір знаходиться на бірці, прикріплені до одягу, та якщо це взуття то маркування розміру наноситься на підошву або устілці.

Мода [7] – засіб проявлення смаку в певній сфері діяльності, є не довготривалою мірою, оскільки корегується різними факторами людської діяльності. В деякому сенсі модою можна назвати зміну зразків і форм одягу, що, як правило, відбувається протягом відносно невеликих проміжків часу. Мода постає способом соціальної ідентифікації чи маркуваня.

Колекція [8] – це класифікований набір однорідних предметів, які несуть історичний, художній чи науковий інтерес.

Колекція в моделюванні одягу – впорядкована серія, яка складається з моделей різного роду призначення, об’єднаних єдністю авторської концепції, матеріалів, образу, кольорового вирішення, форми, базових конструкцій, стильового вирішення. Також колекція може вміщувати різні елементи, як і повні набори комплектів та і одиночні віироби.

Стиль одягу [9] – певний набір елементів (одягу, аксесуарів, взуття тощо) які в сукупності створюють образ, тобто стиль який описує певну особу та її вподобання, стать, вік, рід діяльності, характер. Таким чином дана ознака стилю відрізняє людину за індивідуальними особливостями від інших.

Отож, стиль одягу дуже важливий елемент іміджу як людини (зовнішній вигляд) та і цілих компаній (дресс-код).

Таким чином, продаж одягу є важливим та розповсюдженим процесом в житті людини та суспільства. При цьому використовується багато даних про параметри одягу, покупців, продавців, питання оплати та доставки замовлень. Тому автоматизація процесу продажу одягу є актуальним питанням в IT-технологіях.

Отже, спираючись на вище розглянутий аналіз, поставленої задачі, можна відокремити наступні параметри предметної області:

Параметр	**Опис**
Товар	Продукція магазину, яка є у наявності та знаходиться у продажу, має певну ціну та розмір.
Ціна	Міра вартості певного товару, який є у наявності.
Розмір	Набір параметрів покупця за якими він формує певний розмір, обраного товару.
Бренд	Фірма, яка є виробним певної продукції у магазині.
Адрес доставки	Інформація користувача, важливе поле при формуванні замовлення. Являє собою країну та почтовий індекс.
Логін	Адреса електронної пошти зареєстрованого користувача. Необхідний для вказання особистості.

Пароль	Слугує доказом для підтвердження особистості за вказаним логіном.
Замовлення	Фінальний процес покупки. Формується вибором товару, заповнення форми з певними полями.
Продаж	Ключовий процес інтернет-магазину, приставляє собою продаж, оновлення даних про товар, додавання нового товару та видалення з вітрини проданого.

1.2 Аналіз інформаційного забезпечення предметної області

1.2.1 Аналіз існуючого програмного забезпечення предметної області

На даний момент у предметній області продажу одягу часто використовують можливості інформаційних технологій, що є позитивно. Магазини, як правило, використовують різноманітні технології, такі як ASP.NET, PHP та інші.

На Рисунку 1.1 зображена частина сайту "ZARA" [10], який пропонує можливості перегляду нових колекцій, вибору найближчого магазину та що є в наявності та території України. Також на цьому сайті можна побачити відгуки про товар, дати нових релізів тощо.

Рисунок 1.1 – Головна сторінка сайту «ZARA»

Недоліком цієї системи є те що покупці не мають можливості зробити замовлення обраного товару безпосередньо з сайту. Але вони мають можливість до перегляду найближчого магазину відносно них де є обраний товар. Приклад перегляду асортименту за вказаним фільтром зображено на рисунку 1.2.

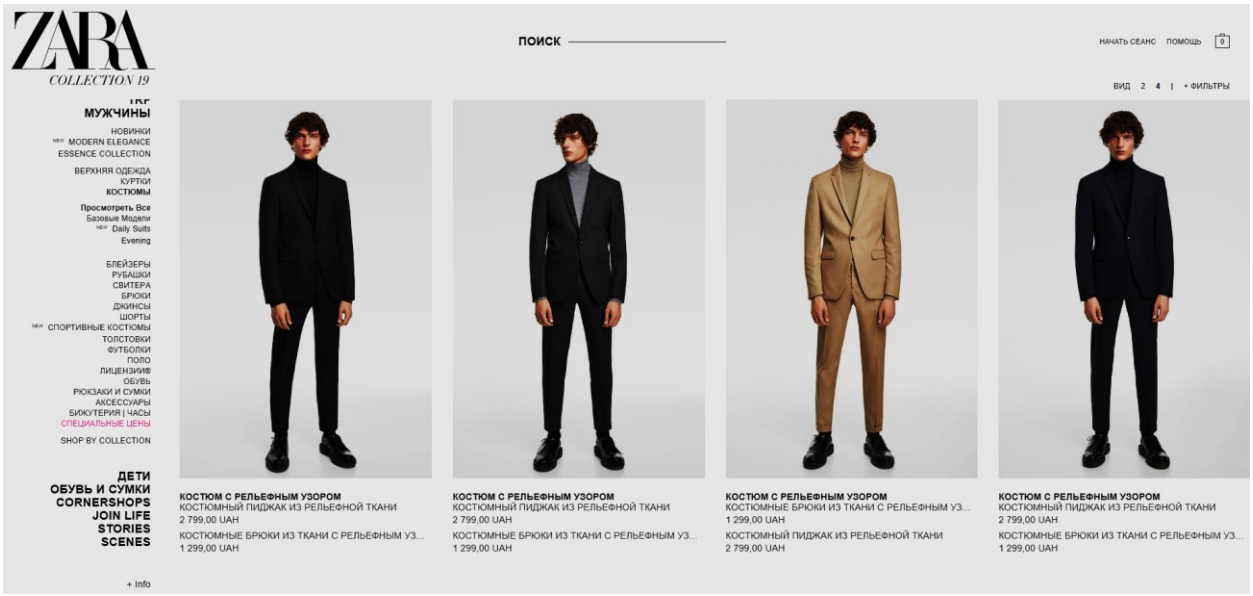


Рисунок 1.2 – Перегляд асортименту за вказаним фільтром на сайті «ZARA»

На Рисунку 1.3 зображений головний сайт магазину «ASOS» [11]. Дизайн сайту дещо відрізняється, та система роботи сайту. На відміну від попереднього тут є можливість замовити товар безпосередньо з сайту, а не купувати в найближчому магазині, що в деякому роді є набагато зручніше для користувача.

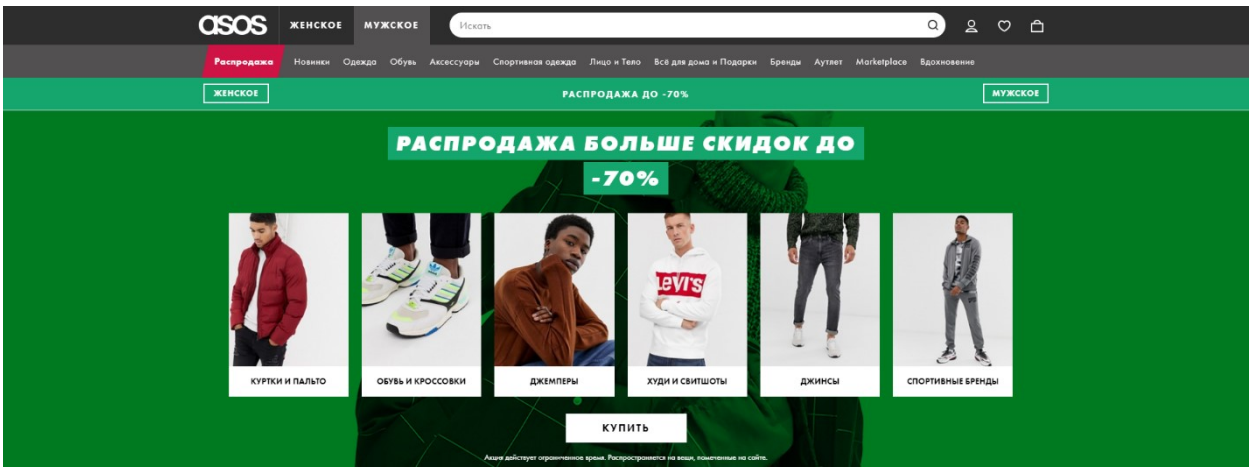


Рисунок 1.2 – Головна сторінка сайту «ASOS»

Перевагою цього виду інтернет магазину є можливістю замовити обраний товар з сайту за вказаною адресою, що набагато зручніше. Повний процес замовлення наведено на Рисунках 1.3 – 1.7.

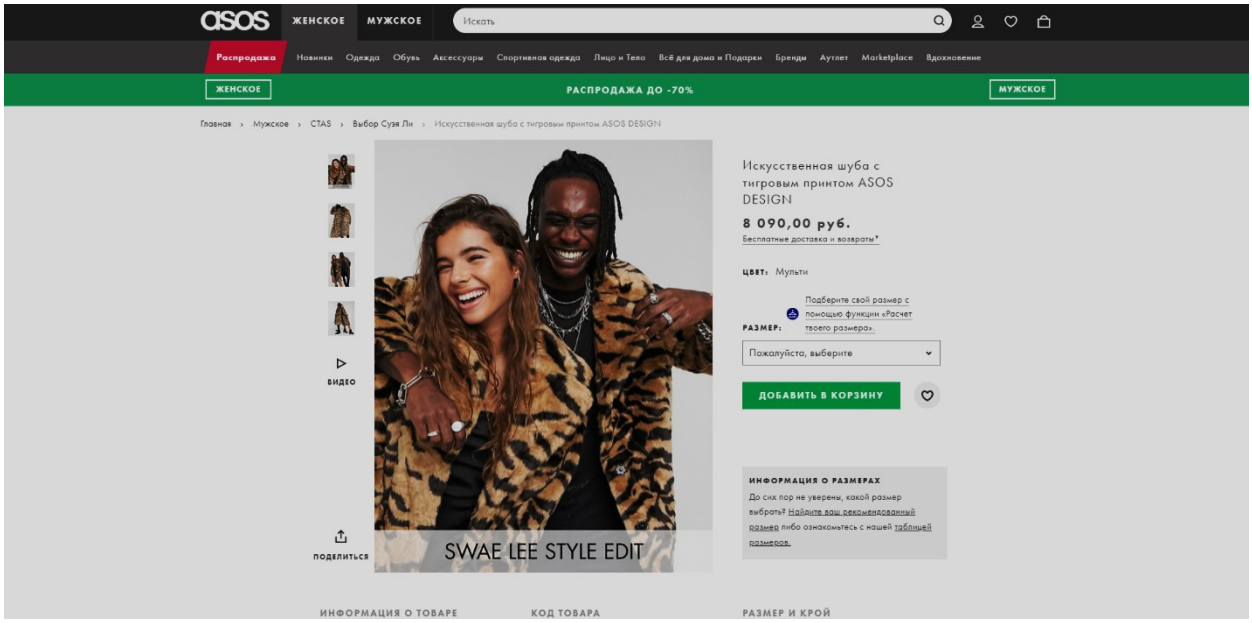


Рисунок 1.3 – Вибір товару та розміру у інтернет-магазині «ASOS»

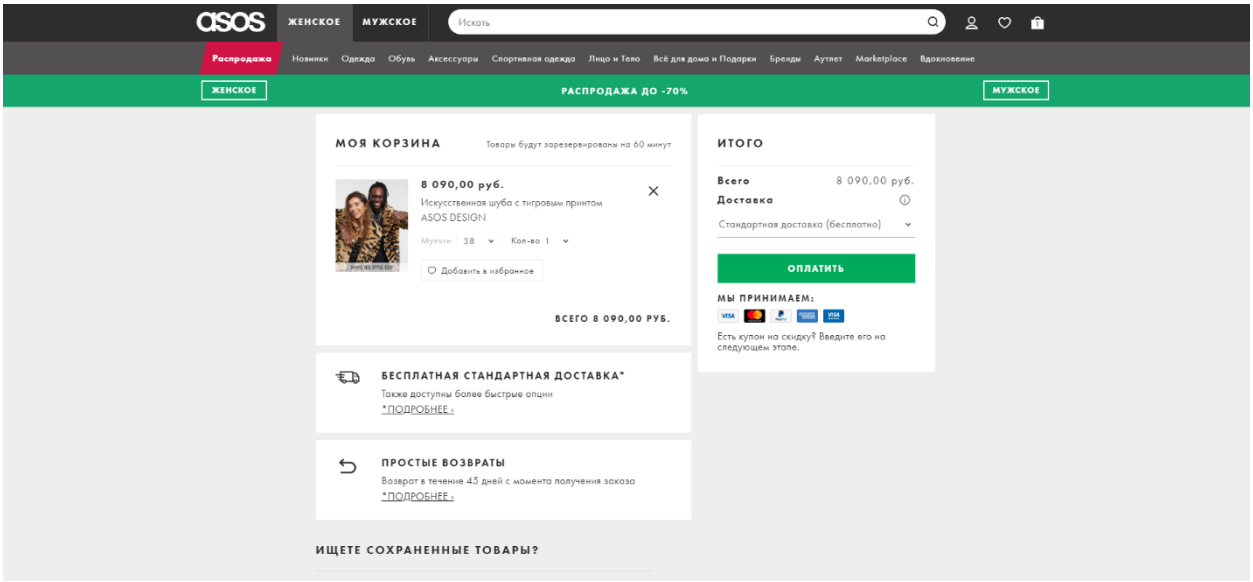


Рисунок 1.4 – Вибір виду доставки та кількості обраного товару у інтернет-магазині «ASOS»

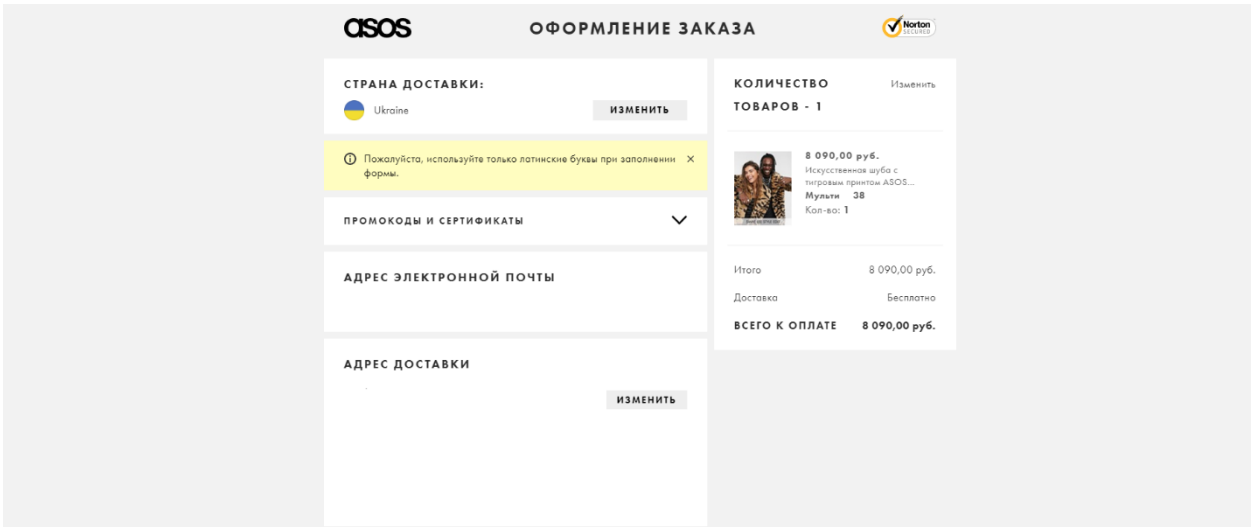


Рисунок 1.5 – Оформлення заказу та введення персональних даних у інтернет-магазині «ASOS»

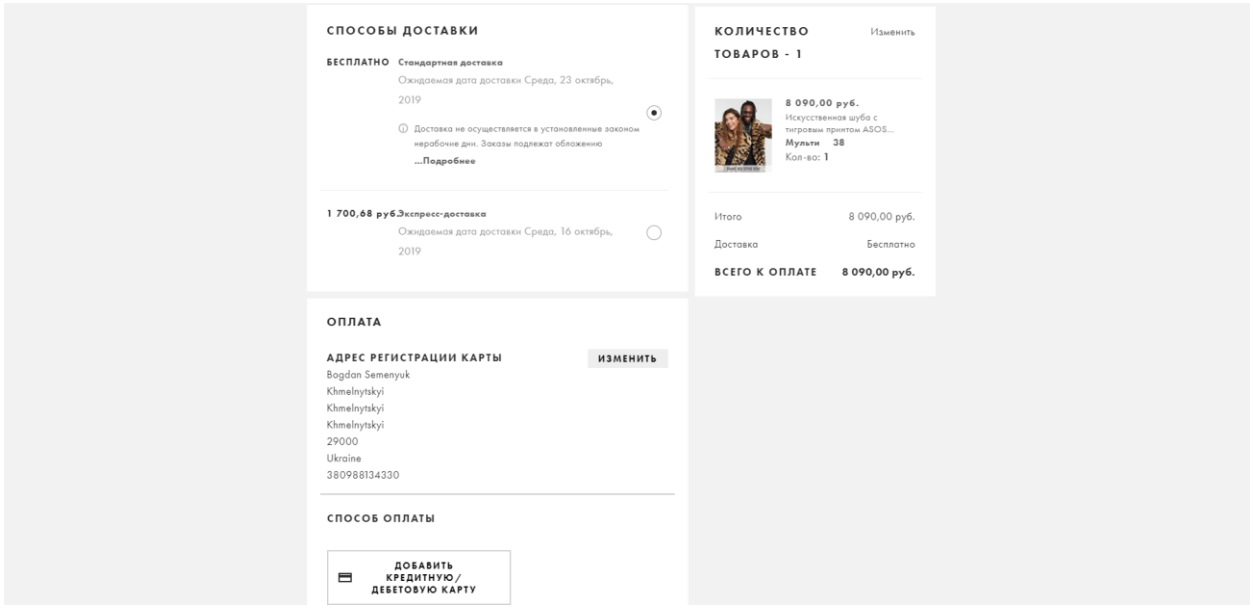


Рисунок 1.6 – Вибір способу доставки та оплати у інтернет-магазині «ASOS»

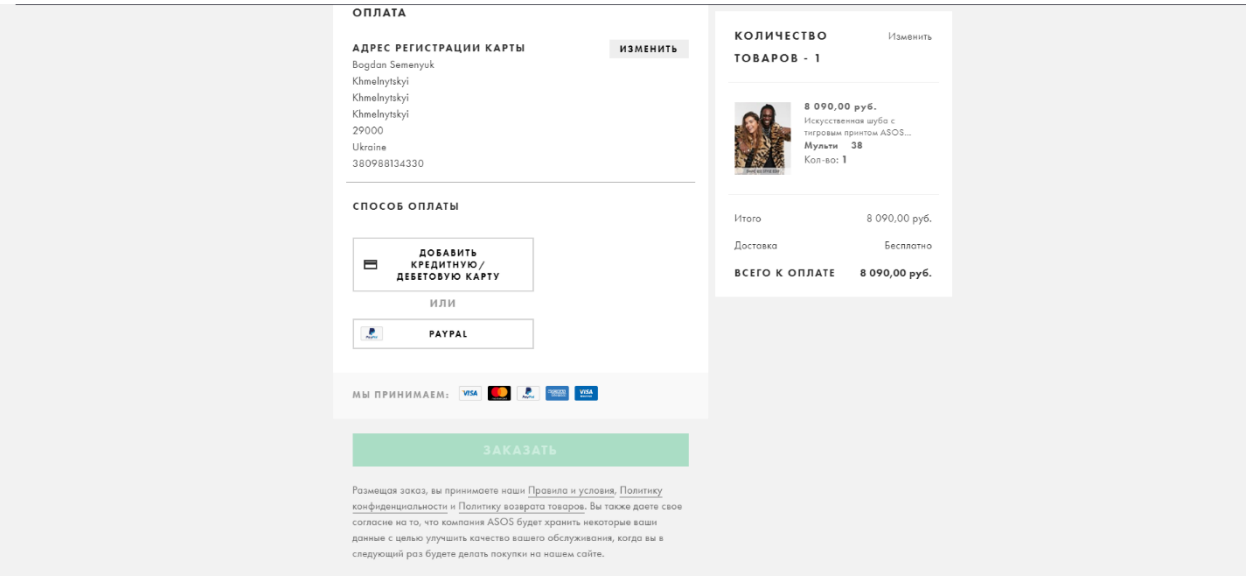


Рисунок 1.7 – Кінцевий етап після введення всіх даних підтвердження замовлення у інтернет-магазині «ASOS»

Отже, наразі можна знайти багато прикладів інтернет-магазинів. В основному, вони надають функціональні можливості клієнтам реєстрації, перегляду товару та робити інші речі, зв’язані зі звичайним походом до магазину, але не мають можливостей приміряти одяг та безпосередньо підібрати розмір.

1.2.2 Аналіз сучасних засобів створення програмного забезпечення

Веб-застосунок [12] – лінійний або паралельний застосунок, який у вигляді клієнтської сторони використовує браузер, та у ролі серверу виступає веб-сервер. Браузер слугує, так званим комиілятором, коду для клієнтської сторони, зазвичай це HTML, CSS та JS. Основною ж функцією браузера є інтерпретація від та відправлення даних серверу, цей процес включає в себе валідацію, збереження кукісів, обробку помилок тощо. У свою чергу сервер повинен обробляти, зберігати дані, які надходять від клієнта, також це зв’язок з джерелом даних, зазвичай це база або банк даних. Явною перевагою цього підходу є незалежність клієнтів від конкретної операційної системи, чи пристрою, тому веб-додатки є крос-платформними рішеннями, у порівняні з

мобільними чи десктопними застосунками, де існує велика прив’язка до мови операційної системи.

Іншими словами застосунок, додаток або програма – це набір програмних засобів, підсилених бібліотеками та мовними особливостями реалізації. Який повинен виконувати певний набір поставлених задач, та повертати бажаних результат. Слід розуміти різницю між драйверами, операційними системами, засобами розробки та застосунками. Застосунок слугує для вирішення бажаних проблем користувача, а не підтримка апаратних засобів.

На даний момент для розробки веб застосунків можна вибрати наприклад із 2 існуючих платформ: .NET та PHP.

Microsoft .NET [13] – кінцевий продукт компанії Microsoft, який представляє собою платформу для створення програмних продуктів, як і звичайних додатків, так і складних, ресурсозатратних рішень чи навіть веб-сайтів. Даний фреймворк є деякого роду продовженням підходів та принципів, відлагоджених часом, у технології Java. Однією зі головних ідей реалізації .NET являється сумісність служб, які створені різними мовами програмування, своєрідний аналог можливості у Java.

У 2020 році компанія Microsoft випустила нову версію платформи, .NET 5, яка містить в собі також нову версію C# 9.0. Нова версія це великий крок у розширення кросс-платформної розробки на платформі .NET 5, вона об’єднала в собі .net framework та .net core в єдину, тим самим позбувшись розгалуження напрямків під час розробки.

Нові мовні функції включають записи, встановлення лише для ініціалізації, оператори верхнього рівня, вдосконалення узгодження шаблонів, цільові типи виразів, коваріантні повернення та багато іншого!

Великим акцентом на C # 9.0 є незмінне представлення форм даних, яке в основному представлене новими типами записів та властивостями лише для ініціювання. Зазвичай записи позначаються як одна з найцікавіших нових функцій у версії C # 9.0.

Запис є новим незмінним посилальним типом у C #. Результатом їх присвоєння або зміни значень є створення копії цього об'єкта. Це те саме, що .NET обробляє типи значень і рядки. Ще однією незмінною особливістю випуску є властивості лише для init, точніше, init accessor, варіант встановленого доступу, який можна викликати лише під час ініціалізації об'єкта для певної властивості.

Найбільша відмінність від класів полягає в тому, що типи записів використовують рівність на основі вартості, і вони незмінні за замовчуванням. З їх допомогою створюється весь незмінний об’єкт, а не лише окремі його властивості. Зазвичай для створення незмінних посилальних типів C # змушує писати трохи додаткового коду.

Отже, слід зазначити основні переваги платформи .NET та мови програмування C#:

1. Легко розпочати роботу. C# – це мова високого рівня, тому вона ближча до інших популярних мов програмування, таких як C, C++ та Java, і тому стає простою для вивчення будь-якою людиною.

2. Широко використовується для розробки настільних та веб-додатків, Це одна з найпопулярніших мов, яка використовується в професійних робочих столах. Якщо хтось хоче створити програми Microsoft, їх першим вибором буде C#.

3. Спільнота. Чим більша спільнота, тим вона краща, оскільки нові інструменти та програмне забезпечення розроблятимуться для покращення. C# має велику спільноту, тому розробляються розробки, щоб вона існувала в системі і не вимерла.

4. Розробка ігор. C# широко використовується в розробці ігор і буде продовжувати домінувати. C# інтегрується з Microsoft і, отже, має велику цільову аудиторію. Такі функції C#, як автоматичне збирання сміття, інтерфейси, об’єктно-орієнтовані тощо роблять C # популярною мовою для розробки ігор.

PHP [14] – скриптова мова програмування (попередня назва якої: Personal Home Page Tools), вона була написана для рендеру HTML-сторінок на стороні сервера та клієнта. Hypertext Preprocessor – це найпоширеніша мова серверного управління, що застосовується у сфері веб-додатків (включно зі

дотнет, джава, перл, пайтон, рубі). Значною перевагою PHP є те що вона підтримується великою кількістю провайдерів. Також, слід розуміти, що PHP – це, на сам перед, проект відкритого типу програмного оснащення.

На платформі PHP, на відміну від .NET в якості бази даних необхідно використовувати MySQL замість MS SQL Server у .NET.

Дві вище розглянуті платформи для розробки веб-застосунків мають свої як і переваги так і недоліки, але обидві є хорошим вибором і в даному випадку, опираючись на персональні вподобання, було обрано платформу .NET та технології ASP MVC 5 для реалізації постановленого завдання. Перевагою даної технології є особливість кешування веб-сторінки, що пришвидшує виконання повторного запиту до web-сайту.

1.3 Постановка задачі

Метою бакалаврської роботи є розробка інтернет-магазину одягу з експертною системою підбору одягу на платформі .NET та технології ASP MVC 5, Core Web Api. В інтернет-магазині необхідно передбачити наявність двох категорій користувачів – адміністратора та клієнта.

Для *адміністратора інтернет-магазина* мають бути доступні наступні функції:

– робота з одягом (перегляд, додавання, редагування, видалення);

– робота з клієнтами (перегляд);

– робота з допоміжними даними інтернет-магазину (магазинами, працівниками, розмірами, брендами, колекціями тощо);

– робота з системою підбору одягу;

– робота з контентом;

– робота з замовленнями (перегляд, зміна статусів).

Для *клієнта інтернет-магазину* мають бути доступні наступні функції:

– робота з одягом (перегляд, додавання у корзину);

– робота з корзиною (перегляд, редагування);

– робота з списком вподобань (перегляд, редагування);

– робота з замовленнями (перегляд статусу);

– робота з допоміжними даними (дані профілю клієнта, контактні дані інтернет-магазину для підтримки).

Експертна система – кінцевий вигляд даного продукту передбачає розділений сервісний додаток, принципово незалежна бекграунд працівник, який є незалежним від основного функціонального процесу інтернет-торгівлі. Головною задачею, якого є прямий підбір розміру користувача за попередньо введеним параметрам, замірам особи.

2 Проектування структури інформаційної системи

2.1 Аналіз та автоматизація обробки інформаційних потоків

Згідно розподілу функцій між клієнтом та адміністратором інтернет-магазину, кожному з них відповідають різні бізнес-процеси. Зокрема, адміністратор має проводити роботу з товаром, клієнтами, магазином та допоміжними даними. В той час як клієнт здійснює роботу з покупками (дії з вітриною, корзиною та сайтом).

Відповідно, бізнес-процеси у роботі інтернет-магазину, які підлягають автоматизації, наступні (рисунок 2.1):

– Бізнес-процес «Робота адміністратора з клієнтами» - призначений для роботи з даними покупців (перегляд списку клієнтів, редагування, видалення);

– Бізнес-процес «Робота адміністратора з товаром» - призначений для роботи з даними товару (перегляд списку товару, редагування, видалення, додавання);

– Бізнес-процес «Робота адміністратора з магазином» - призначений для роботи з даними магазину (перегляд замовлень, додавання, редагування та видалення товару, перегляд статистики, зміна та перегляд усіх даних);

– Бізнес-процес «Робота клієнта з магазином» - призначений для роботи клієнта з магазином (створення замовлення та перегляд статусу, перегляд каталогу, додавання у корзину та редагування, перегляд особистого кабінету).

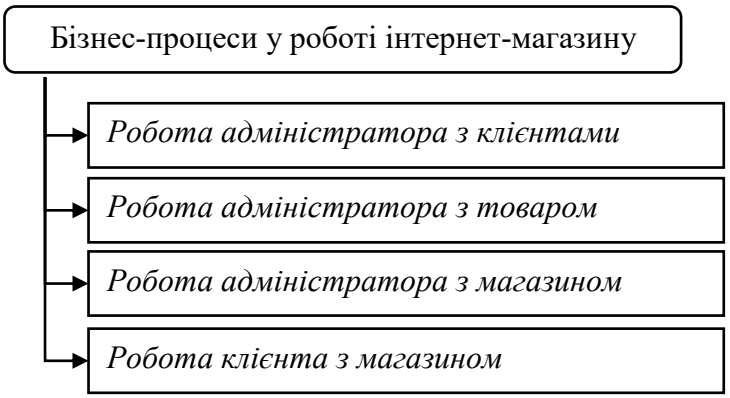


Рисунок 2.1 – Бізнес-процеси у роботі інтернет-магазину

Процес роботи адміністратора з сайтом включає в себе підтвердження особистості адміністратора, при успішній авторизації він потрапляє до панелі адміністратора де має можливість вибору даних з якими він бажає працювати (даними клієнтів, товару, замовлень, тощо). Відповідно після вибору даних для роботи адміністратор має вибір функцій роботи з ними (перегляд, редагування, видалення, додавання), тобто має повні права доступу до всіх можливостей роботи з сайтом (рисунок 2.2).

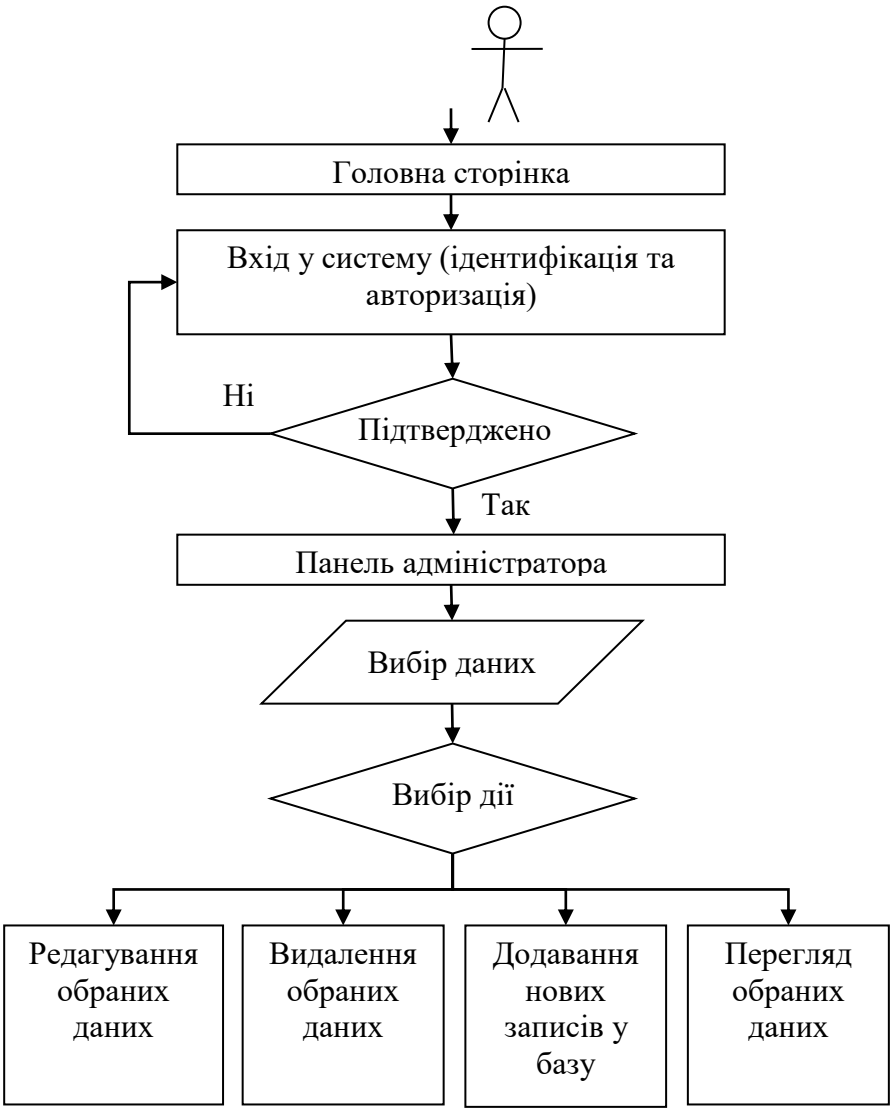


Рисунок 2.2 – Діаграма активності роботи адміністратора з сайтом

Процес роботи клієнта з сайтом передбачено декілька можливостей. Перше це вхід у систему, якщо користувачу не вдається увійти, тоді йому пропонується спробувати ще раз або пройти реєстрацію. Після успішної авторизацію користувач потрапляє до головної сторінки. Далі в нього є перелік

можливостей це перегляд товару, робота з корзиною, перегляд то редагування особистих даних та робота з додатковими даними (перегляд стану замовлення, інформація про магазин, підтримка), як це наведено на рисунку 2.3.

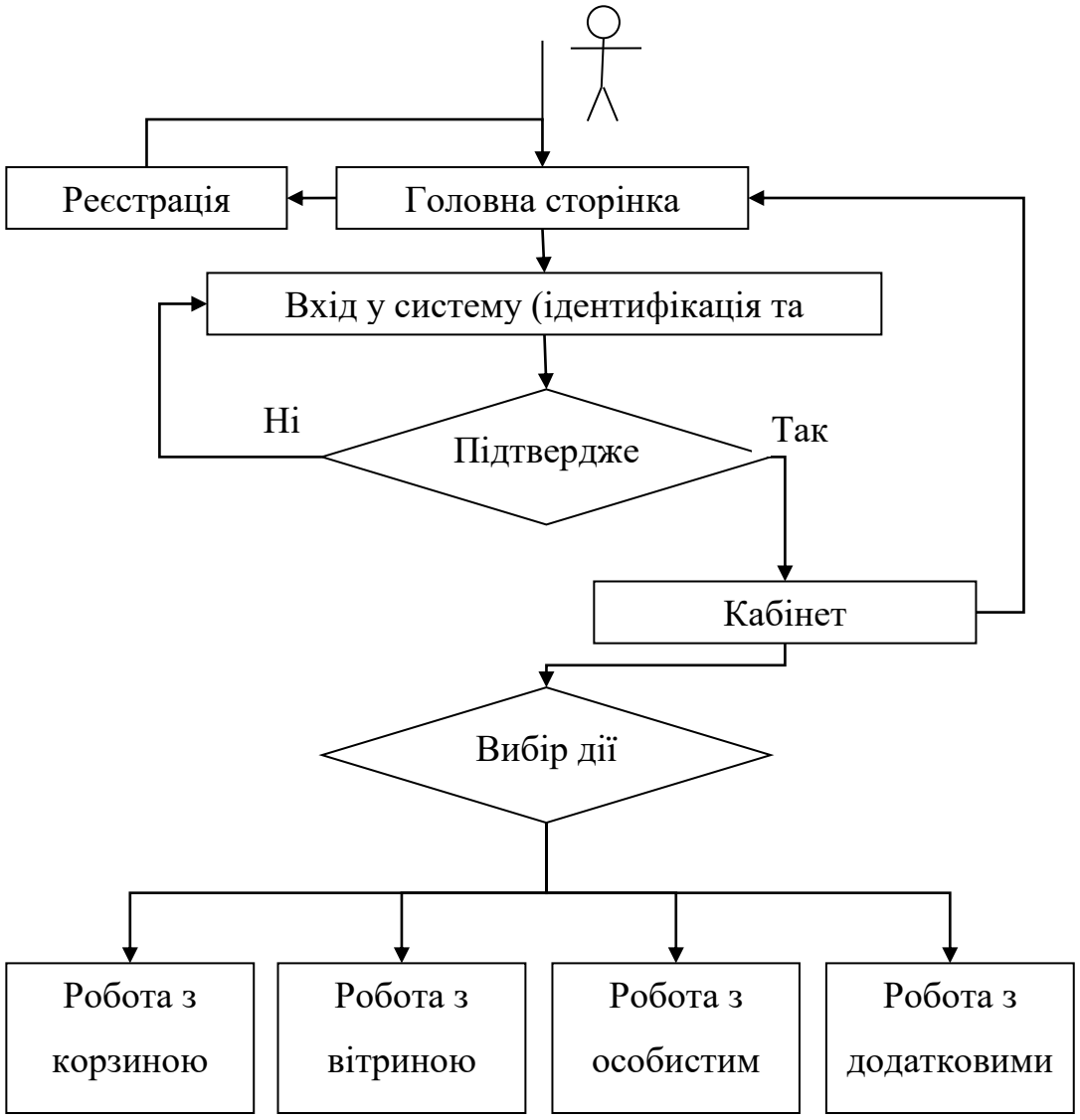


Рисунок 2.3 – Діаграма активності роботи клієнта з сайтом

Отже, для різних груп користувачів має бути передбачений різний функціонал. Також необхідна ідентифікація та авторизація під час входу в систему для перевірки прав доступу.

При виконанні наведених функцій адміністратора та клієнта інтернет-магазину внесені дані мають бути збережені в відповідній базі даних.

2.2 Розробка структури інформаційної системи

Для розробки структури бази даних сайту інтернет-магазину важливо встановити головні сутності предметної області, включаючи властивості. На рисунку 2.4 зображена даталогічна модель бази даних інтернет-магазину, яка передбачає такі таблиці: англ: Orders (замовлення), OrderOnes (одне замовлення), Clothes (одяг), Stores (магазини), TypeBuys (тип покупки), Clients (клієнти), Photos (фотографії), Sizes (розміри), Brands (бренди), Materials (матеріали), TypeClothes (тип одягу).

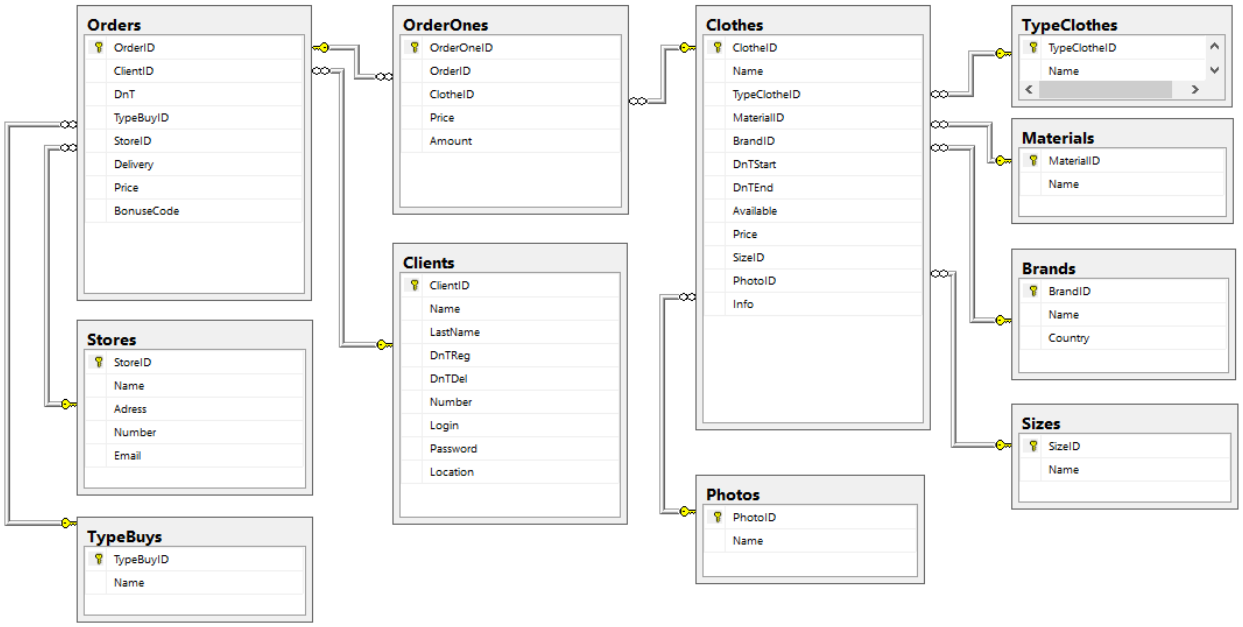


Рисунок 2.4 – Даталогічна модель бази даних інтернет-магазину

Кожна з таблиць у базі даних слугує для опису даних окремої сутності. Властивості сутності – табличний атрибут. Наприклад, сутність «Клієнт» (рисунок 2.5) має властивості: прізвище, ім'я, дата та час реєстрації та видалення з бази, номер телефону, логін, пароль, адреса.

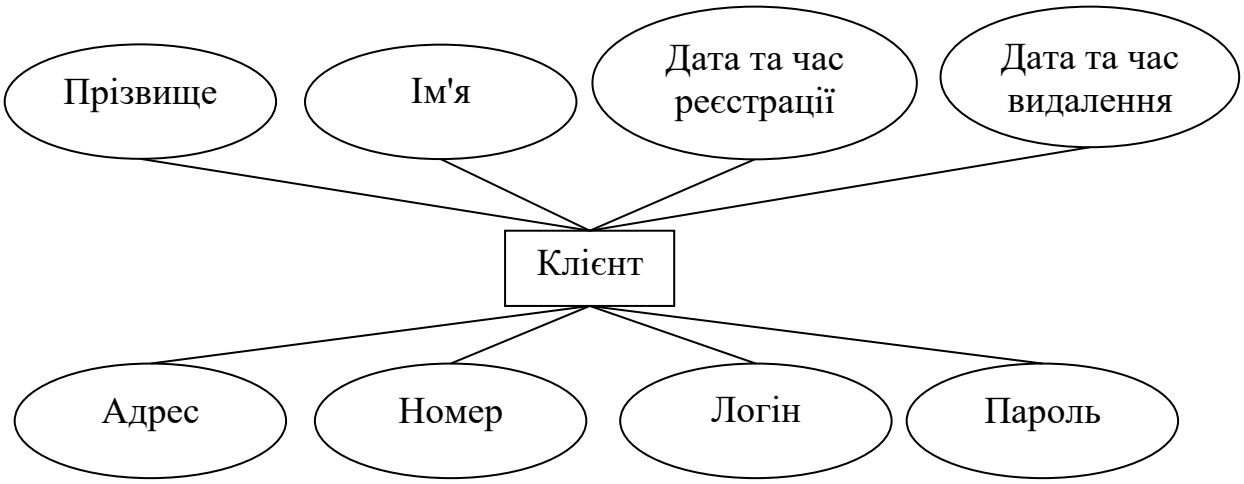


Рисунок 2.5 - Властивості сутності "Клієнт"

Таблиця «Orders» призначена для збереження даних про замовлення й має наступні атрибути: англ: OrderID, ClientID, DnT, TypeBuyID, StoreID, Delivery, Price, BonuseCode (таблиця 2.1).

Таблиця 2.1 – Атрибути таблиця «Orders»

№ п/п	**Назва**	**Тип даних**	**Опис**
1	OrderID	INT	Унікальний ідентифікатор лічильник
2	ClientID	INT	Вторинний ключ
3	DnT	NVARCHAR	Дата та час замовлення
4	TypeBuyID	INT	Вторинний ключ
5	StoreID	INT	Вторинний ключ
6	Delivery	NVARCHAR	Адреса доставки
7	Price	NVARCHAR	Ціна
8	BonuseCode	NVARCHAR	Бонус-код

Таблиця «OrderOnes» призначена для збереження даних про одне замовлення й має наступні атрибути: OrderOneID, OrderID, ClotheID, Price, Amount (таблиця 2.2).

Таблиця 2.2 – Атрибути таблиця «OrderOnes»

№ п/п	**Назва**	**Тип даних**	**Опис**
1	OrderOneID	INT	Унікальний ідентифікатор лічильник

2	OrderID	INT	Вторинний ключ
3	ClotheID	INT	Вторинний ключ
4	Price	NVARCHAR	Ціна
5	Amount	NVARCHAR	Кількість

Таблиця «Clothes» призначена для збереження даних про одяг й має наступні атрибути: OrderOneID, OrderID, ClotheID, Price, Amount (таблиця 2.3).

Таблиця 2.3 – Атрибути таблиця «Clothes»

№ п/п	Назва	Тип даних	Опис
1	ClotheID	INT	Унікальний ідентифікатор лічильник
2	Name	NVARCHAR	Вторинний ключ
3	TypeClotheID	INT	Вторинний ключ
4	MaterialID	INT	Вторинний ключ
5	BrandID	INT	Вторинний ключ
6	DnTStart	NVARCHAR	Дата та час початку продажу
7	DnTEnd	NVARCHAR	Дата та час кінця продажу
8	Available	NVARCHAR	Наявність у магазині
9	Price	NVARCHAR	Ціна
10	SizeID	INT	Вторинний ключ
11	PhotoID	INT	Вторинний ключ
12	Info	NVARCHAR	Додаткова інцормація

Таблиця «Stores» призначена для збереження даних про магазин й має наступні атрибути: StoreID, Name, Adress, Number, Email (таблиця 2.4).

Таблиця 2.4 – Атрибути таблиця «Stores»

№ п/п	Назва	Тип даних	Опис
1	StoreID	INT	Унікальний ідентифікатор лічильник
2	Name	NVARCHAR	Назва
3	Adress	NVARCHAR	Адреса
4	Number	NVARCHAR	Контактний телефон
5	Email	NVARCHAR	Електронна пошта

Таблиця «TypeBuys» призначена для збереження даних про тип покупки й має наступні атрибути: TypeClotheID, Name (таблиця 2.5).

Таблиця 2.5 – Атрибути таблиця «TypeBuys»

№ п/п	Назва	Тип даних	Опис
1	TypeClotheID	INT	Унікальний ідентифікатор, лічильник
2	Name	NVARCHAR	Назва

Таблиця «Clients» призначена для збереження даних про клієнтів й має наступні атрибути: ClientID, Name, LastName, DnTReg, DnTDel, Number, Login, Password, Location (таблиця 2.6).

Таблиця 2.6 – Атрибути таблиця «Clients»

№ п/п	Назва	Тип даних	Опис
1	ClientID	INT	Унікальний ідентифікатор лічильник
2	Name	NVARCHAR	Ім’я
3	LastName	NVARCHAR	Фамілія
4	DnTReg	NVARCHAR	Дата та час реєстрації
5	DnTDel	NVARCHAR	Дата та час видалення
6	Number	NVARCHAR	Номер телефону
7	Login	NVARCHAR	Логін
8	Password	NVARCHAR	Пароль
9	Location	NVARCHAR	Локація

Таблиця «Photos» призначена для збереження даних про фото й має наступні атрибути: PhotoID, Name (таблиця 2.7).

Таблиця 2.7 – Атрибути таблиця «Photos»

№ п/п	Назва	Тип даних	Опис
1	PhotoID	INT	Унікальний ідентифікатор лічильник
2	Name	NVARCHAR	Назва

Таблиця «Sizes» призначена для збереження даних про розміри й має наступні атрибути: SizeID, Name (таблиця 2.8).

Таблиця 2.8 – Атрибути таблиця «Sizes»

№ п/п	Назва	Тип даних	Опис
1	SizeID	INT	Унікальний ідентифікатор лічильник
2	Name	NVARCHAR	Назва

Таблиця «Brands» призначена для збереження даних про бренди й має наступні атрибути: BrandID, Name, Country (таблиця 2.9).

Таблиця 2.9 – Атрибути таблиця «Brands»

№ п/п	Назва	Тип даних	Опис
1	BrandID	INT	Унікальний ідентифікатор лічильник
2	Name	NVARCHAR	Назва
3	Country	NVARCHAR	Країна виробника

Таблиця «Materials» призначена для збереження даних про матеріали й має наступні атрибути: MaterialID, Name (таблиця 2.10).

Таблиця 2.10 – Атрибути таблиця «Materials»

№ п/п	Назва	Тип даних	Опис
1	MaterialID	INT	Унікальний ідентифікатор лічильник
2	Name	NVARCHAR	Назва

Таблиця «TypeClothes» призначена для збереження даних про тип одягу й має наступні атрибути: MaterialID, Name (таблиця 2.11).

Таблиця 2.11 – Атрибути таблиця «TypeClothes»

№ п/п	Назва	Тип даних	Опис
1	TypeClotheID	INT	Унікальний ідентифікатор лічильник
2	Name	NVARCHAR	Назва

Отже, для реалізації поставленої задачі, інтернет-магазину, є доцільною наведена вище база даних.

Для даної інформаційної системи передбачено наступний перелік бізнес процесів.

Таблиця 2.12 – Бізнес-процеси

Номер бізнес-процесу	Назва бізнес-процесу
1_А_з_к	Робота адміністратора з клієнтами
2_А_з_т	Робота адміністратора з товаром
3_А_з_м	Робота адміністратора з магазином
4_К_з_м	Робота клієнта з магазином

Детальний опис наведений нижче:

1. Бізнес-процес 1_А_з_к (Робота адміністратора з клієнтами). Процес передбачає взаємодію адміністратора з таблицею клієнти (Clients), тобто юзер з перною роллю то типу адміністратора чи модератора (доступом до адмін панелі), може переглянути список клієнтів сайту. Може виконуватися лише функція зчитування (read).

2. Бізнес-процес 1_А_з_т (Робота адміністратора з товаром). Процес передбачає взаємодію адміністратора з певним рядом таблиць (Clothes, Photos, Sizes, Brands, Materials, TypeClothes). В роботі з даними таблицями адміністратор має доступ до повного набору функцій (create, read, update, delete). Даний процес представляє собою примітивний алгоритм роботи з товаром. Для прикладу процес додавання нового товару на вітрину розділяється на декілька етапів. Спочатку потрібно переконатися в наявності даних зв’язаних таблицях (Photos, Sizes, Brands, Materials, TypeClothes), типу якщо з’являється одяг нового бренду то відповідно необхідно додати новий бренд до таблиці брендів. Коли ж всі дані додані та перевірені на коректність, необхідно додати сутність самого продукту (одягу). Результатом цього процесу є відображення ново доданого товару на сторінці «вітрина» інтернет-магалину.

3. Бізнес-процес 1_А_з_м (Робота адміністратора з магазином). Процес передбачає роботу користувача за таблицями (Orders, OrderOnes, Stores). В даному випадку надана функція часткового редагування для цих таблиць. Тобто юзер може лише редагувати певні поля цих таблиць. Так для прикладу адміністратор має змогу відхилити замовлення власноруч, але не може видалити. Тут використовується підхід (soft delete) коли записи не видаляться а лише проставляються у полі «IsDeleted» значення «True». Це необхідно для збору статистики.

4. Бізнес-процес 1_К_з_м (Робота клієнта з магазином). Даний процес відповідає за здійснення покупок клієнтом. Передбачено лише додавання нових записів у таблиці (Orders, OrderOnes). Тобто користувач має змогу додати товар у корзину, після чого формується замовлення, тут додається новий запис у таблицю замовлень та зв'язані записи у корзині з цим замовленням отримують ключ на нього, після цього корзина стає пустою. Звісно ж клієнт має змогу переглянути свої замовлення в особистому кабінеті.

Для розробки структури управління даними систему підбору розміру буде використано два об'єкта (data transfer object), а саме моделі: для відправки запиту (PersonDto) та відповідно для отримання відповіді (PersonSizeDto).

Модель персони включає в себе набір даних користувача (рисунок 2.6), так звана форма яку потрібно заповнити, щоб отримати результат розрахунку.

```
0 references
public class PersonDto
{
    0 references
    public int Height { get; set; }
    0 references
    public int Weight { get; set; }
    0 references
    public int ChestGirth { get; set; }
    0 references
    public int WaistCircum { get; set; }
    0 references
    public int HipGirth { get; set; }
    0 references
    public int LegLength { get; set; }
    0 references
    public int TypeClotheId { get; set; }
    0 references
    public int GenderId { get; set; }
}
```

Рисунок 2.6 – Об’єкт для передачі даних PersonDto

Отож, даний набір даних це не більше як сутність персони з представлення певного спектру її властивостей, у даному випадку це:

– ріст;
– вага;
– обхват грудної клітки;
– обхват талії;
– обхват стегон;
– довжина ноги від поясу;
– тип одяг;
– стать.

У відповідь після відправлення вище наведеного переліку параметрів, система відправляє список із найбільш вірогідних розмірів, для певного товару, зі відсотковою оцінкою доцільність кожного із списку. Вигляд моделі для відповіді наведено на рисунку 2.7.

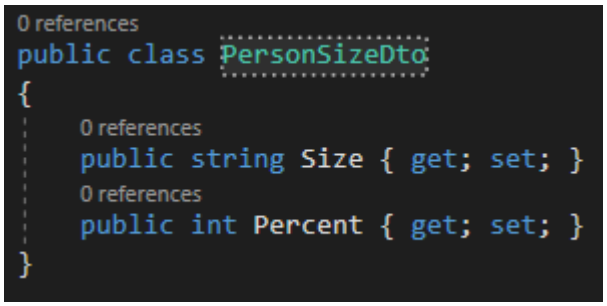


Рисунок 2.7 – Об’єкт для передачі даних PersonSizeDto

Загалом, така реалізація структури підбору розмірів відповідає поставленим нормам, але дуже важливим фактором, який впливатиме на результати, буде виставлення коректних коефіцієнтів та діапазонів.

2.3 Вибір засобів розробки інформаційної системи

2.3.1 Опис мови програмування

Мова програмування C# (сі шарп) [15] – високорівнева, об'єктно-орієнтована мова управління програмними та апаратними, на платформі .NET. Перевагою якої є безпечна система типізації (яка дозволяє відловити багато помилок на етапі компіляції) та поліморфна інтерфейсна архітектура. Розробники: Андерсон Гейлсберг, Скотт Вілтамут та Пітер Гольде, при фірмі Microsoft.

Синтаксис C# по своїй структурі нагадує Java та C++. Слід зазначити що мова в своєму арсеналі має: підтримку поліморфізму, строгу статичну типізацію, перевантаження усіх видів операторів, делегати, події, атрибути, властивості, поля, винятки на різних етапах, коментарі тощо. Звичайно, багато что було унаслідувано від попередників таких як: Smalltalk, Java, C, C++. Такий підхід унаслідування та врахування помилок попередників дав розробникам мови C# гарну базу для аналізу відомих рішень та підвищив ефективність фінального продукту. Результатом стала лаконічна мова зі простим ситаксисом, хоча напочатку були прогалини та недоліки, їх було усунуто в наступних версіях.

Отже для задачі, що розглядається, головною мовою програмування буде C#.

2.3.2 Вибір фреймворку

.NET Core [16] – це нова версія .NET Framework, яка є безкоштовною платформою загального призначення з відкритим кодом, що підтримується корпорацією Майкрософт. Це крос-платформний фреймворк, який працює на операційних системах Windows, macOS та Linux.

.NET Core Framework можна використовувати для створення різних типів додатків, таких як мобільні, настільні, веб, хмарні, IoT, машинне навчання, мікросервіси, ігри тощо.

.NET Core написано з нуля, щоб зробити його модульним, легким, швидким і крос-платформним фреймворком. Він включає основні функції, необхідні для запуску базової програми .NET Core. Інші функції пропонуються як пакети NuGet, які ви можете додати його у свою програму за потреби. Таким чином, програма .NET Core прискорює продуктивність, зменшує обсяг пам'яті та стає простою в обслуговуванні.

У .NET Framework є деякі обмеження. Наприклад, він працює лише на платформі Windows. Крім того, вам потрібно використовувати різні API .NET для різних пристроїв Windows, таких як робочий стіл Windows, магазин Windows, Windows Phone та веб-додатки. На додаток до цього .NET Framework – це загальномашинна система. Будь-які внесені до нього зміни впливають на всі додатки, які залежать від нього. Дізнайтеся більше про мотивацію .NET Core тут.

Web API [17] представляє інший спосіб побудови програми ASP.NET дещо відмінний від ASP.NET MVC. Web API представляє собою веб-службу, яка може взаємодіяти з різними додатками. При цьому додаток може бути веб-додатком ASP.NET, або може бути мобільним або звичайним десктопних додатком.

Також треба відзначити, що платформа Web API 2 не є частиною фреймворка ASP.NET MVC і може бути задіяна як в зв'язці з MVC, так і в поєднанні з Web Forms. Тому в Web API є своя система версій. Так, перша версія з'явилася з .net 4.5. А разом з .NET 4.5.1 і MVC 5 вийшла Web API 2.0.

ASP.NET MVC Framework [18] – це програмне забезпечення з відкритим кодом від Microsoft. Її фреймворк веб-розробки поєднує в собі особливості архітектури MVC (Model-View-Controller), найсучасніші ідеї та методики з розробки Agile та найкращі частини існуючої платформи ASP.NET. Цей посібник надає повну картину фреймворку MVC та навчає, як створити програму за допомогою цього інструменту.

ASP.NET зовні багато в чому зберігає схожість із старішою технологією ASP, що дозволяє розробникам відносно легко перейти на ASP.NET. У той же час внутрішній устрій ASP.NET істотно відрізняється від ASP, оскільки вона

заснована на платформі. NET і, отже, використовує всі нові можливості, що надаються цією платформою.

Переваги ASP.NET:

– ASP.NET має перевагу у швидкості в порівнянні з іншими технологіями, заснованими на скриптах (PHP, тощо);

– Розширюваний набір елементів управління і бібліотек класів дозволяє швидше розробляти застосунки;

– ASP.NET спирається на багатомовні можливості .NET, що дозволяє писати код сторінок на C#, VB, C/C++ тощо;

– Поділ візуальної частини та бізнес-логіки;

– Розширювана модель обробки запитів.

Модель–вигляд–контролер [19] (MVC) – архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення.

Модель: зберігає та керує даними.

Часто база даних, у нашому швидкому прикладі ми будемо використовувати локальну веб-пам’ять у браузері для ілюстрації концепції.

Вигляд – графічний інтерфейс користувача, це візуальне представлення даних – як діаграма, діаграма, таблиця, форма.

Представлення містить всю функціональність, яка безпосередньо взаємодіє з користувачем, як-от натискання кнопки або подія введення.

Контролер – Мізки програми. Контролер з'єднує модель і вигляд. Контролер перетворює вхідні дані з подання на вимоги щодо отримання чи оновлення даних у моделі.

Контролер отримує вхідні дані від виду, використовує логіку для переведення вхідних даних на попит на модель, модель захоплює дані, контролер передає дані з моделі назад у подання, щоб користувач міг побачити їх на гарному дисплеї.

Для вирішення, вище поставленої задачі, було обрано фреймворк ASP.NET MVC 5 та Web API.

2.3.3 Вибір системи керування базою даних

Microsoft SQL Server [20] – це система управління реляційними базами даних (СУБД), розроблена корпорацією Майкрософт. Він в першу чергу розроблений і розроблений, щоб конкурувати з базами даних MySQL та Oracle. SQL Server підтримує ANSI SQL, який є стандартною мовою SQL (мова структурованих запитів). Однак SQL Server постачається із власною реалізацією мови SQL, T-SQL (Transact-SQL).

Для даної задачі, обрано систему керування базою даних SQL Server.

2.3.4 Опис технології доступу до даних

ADO.NET Entity Framework [22] (EF) - об'єктно-орієнтована технологія доступу до даних, є об'єктно-реляційним картографічним рішенням (ORM) для Microsoft .NET Framework. Забезпечує можливість взаємодії з об'єктами як через LINQ у вигляді LINQ до сутностей, так і за допомогою Entity SQL. Для полегшення побудови веб-рішень використовуються як сервіси передачі даних ADO.NET (Astoria), так і комбінація Фонду комунікацій Windows та Фонду презентації Windows, що дозволяє створювати багаторівневі програми, застосовуючи один із MVC, Шаблони дизайну MVP або MVVM.

Отже, було обрано технологію доступу до даних Entity Framework для розробки інтернет-магазину.

3 Програмна реалізація

3.1 Структура і функціональне призначення модулів системи, їх взаємозв'язок

Розробка проекту буде проводитись у програмі Visual Studio 2019. Проект з перевірко дійсності, реєстрація та авторизація буде реалізована за допомогою Identity, що значно спрощує процес розробки, цього буде достатньо для невеликого інтернет-магазину.

Використання моделі, представлення та контролера допомагає отримати добре структурований проект. В даному випадку буде використано, три котролера: AdminController, AccountController, ScaseController. Адмін-контролер – для функцій управління магазином які передбачені лише для адміністратора, акаунт-контролер – для реєстрації та авторизації користувачі та надання їм відповідний прав доступу, вітрина-контролер – призначений для відображення певних даних про товар, роботи з корзиною, створення замовлень. Відповідно до цих контролерів будуть реалізовані відповідні представлення для роботи з даними. Моделі реалізовані для відповідних таблиць з бази даних. Вигляд класової структури проекту на рисунку 3.1.

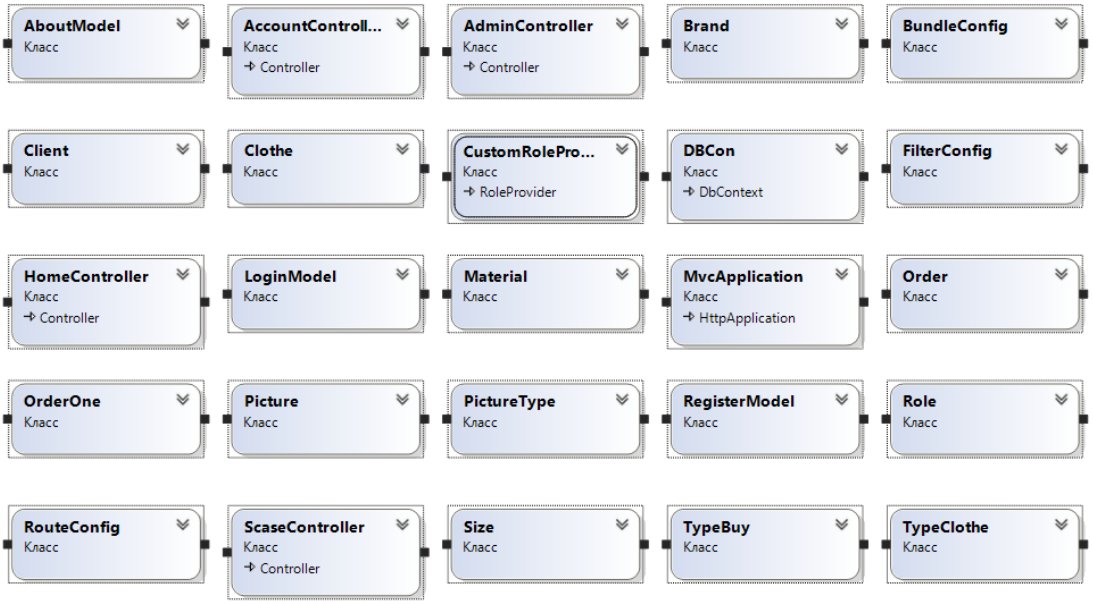


Рисунок 3.1 – Діаграма класів для інтернет-магазину

3.2 Розробка програмних модулів

3.2.1 Користувачі

Для користувачів інтернет-магазину необхідно проводити реєстрацію та авторизацію, відповідно необхідні коректні перевірки : чи є даний логін вільний та чи правильно підтверджений пароль, при реєстрації; чи існує користувач з таким логіном та чи правильно був введений пароль відповідно про авторизації в систему. Результат виконання програмного коду який відповідає реєстрацію користувача можна побачити на рисунках 3.2-3:

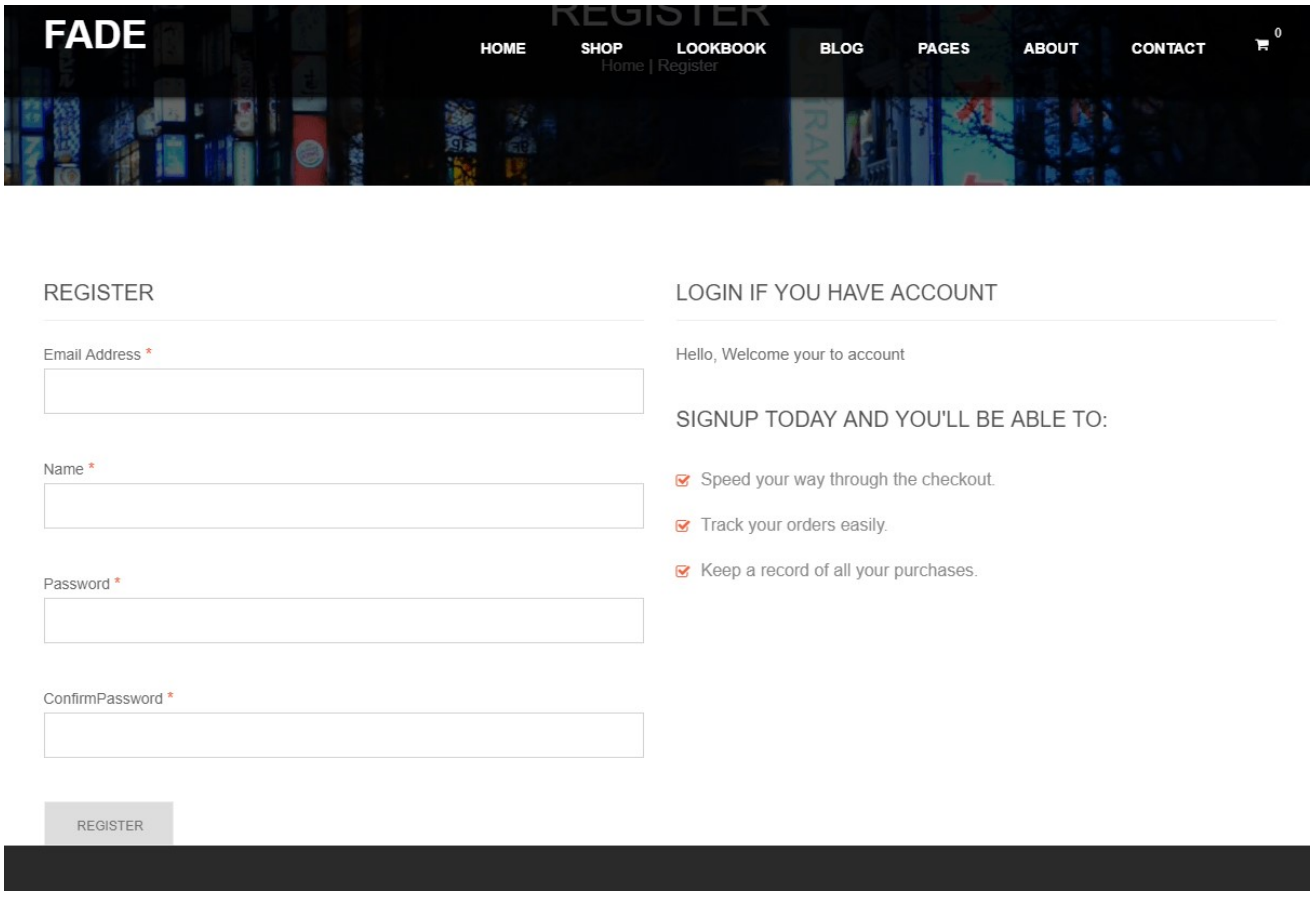


Рисунок 3.2 – Форма для реєстрації користувача.

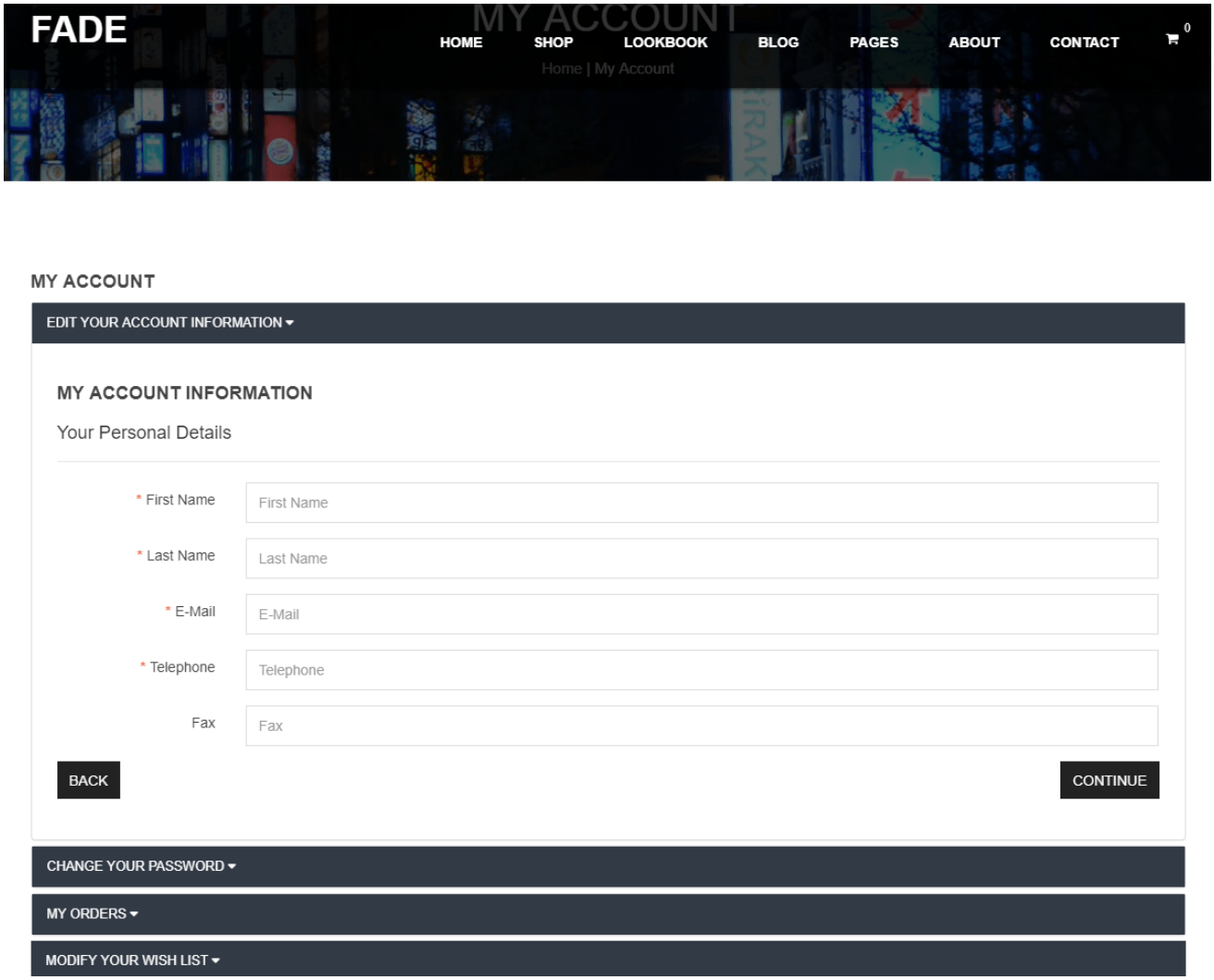


Рисунок 3.3 – Персональний кабінет користувача.

Процес авторизації продемонстрований на рисунках 3.4-5.

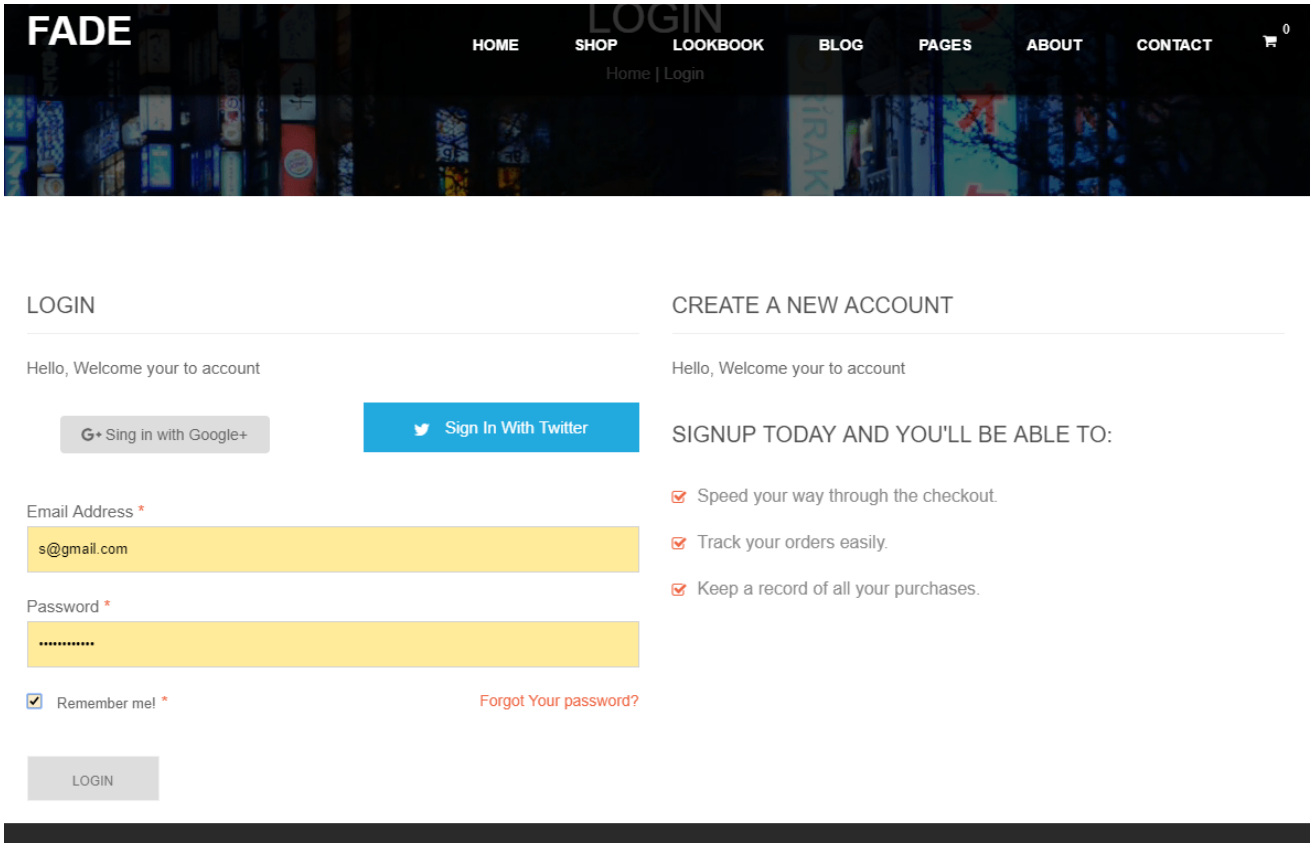


Рисунок 3.4 – Форма для авторизації користувача

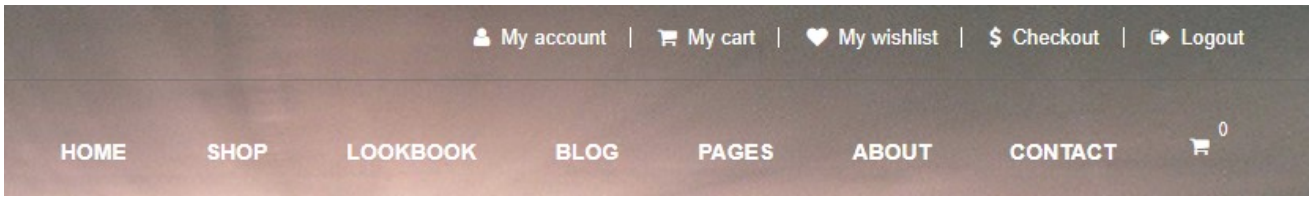


Рисунок 3.5 – Вигляд зміненої шапки після успішної авторизації

Також важливим є розділення ролей користувачів адміністратора, покупця та гостя. Для гостя буде передбачена можливість перегляду асортименту та загальний сторінок, які не потребують авторизації. Приклад на рисунку 3.6.

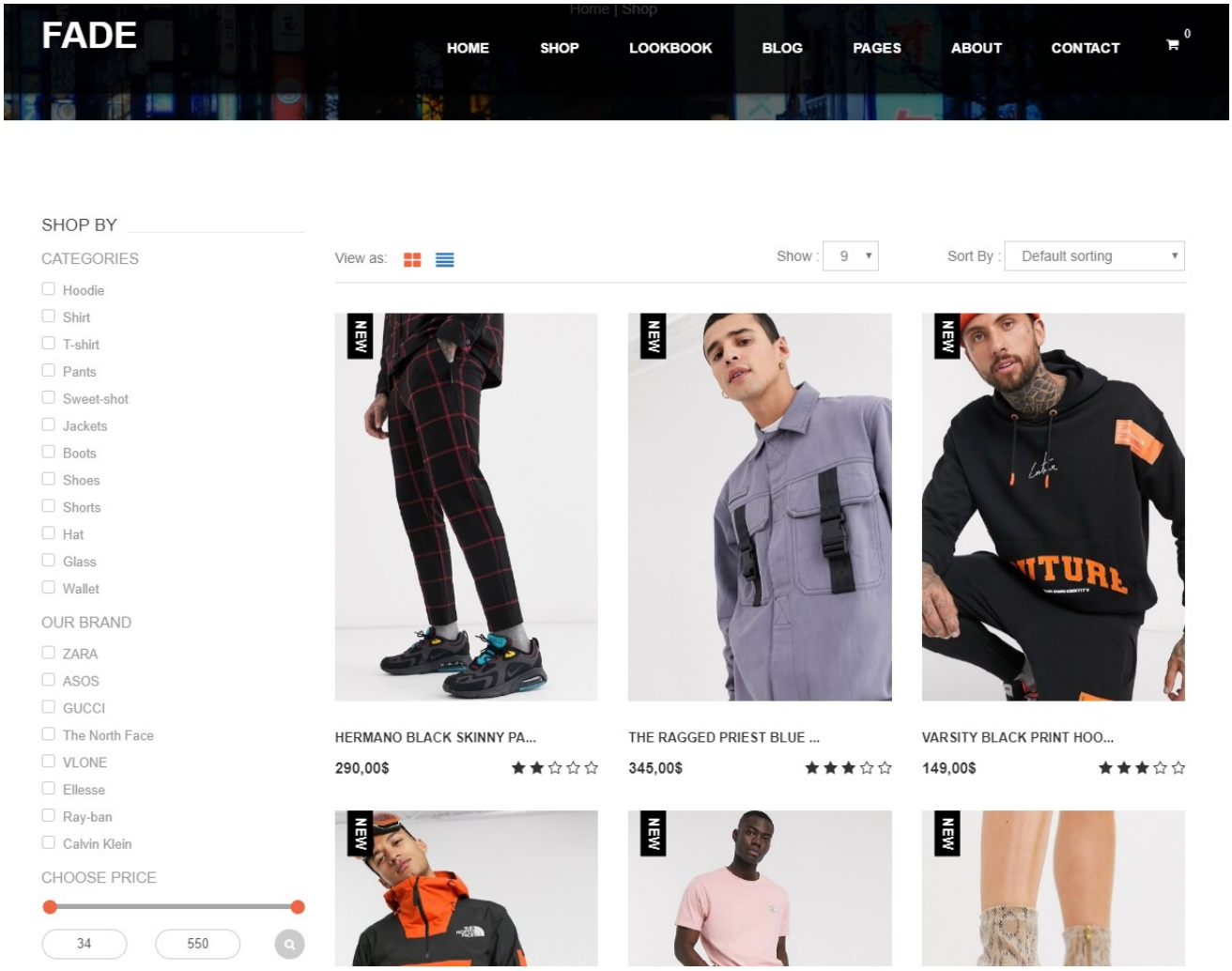


Рисунок 3.6 – Сторінка на які показано весь товар

В свою чергу у покупця буде можливість додавати товар у корзину та створення замовлення. Результати роботи пост и гет версій функції створення корзини на рисунках 3.7-8.

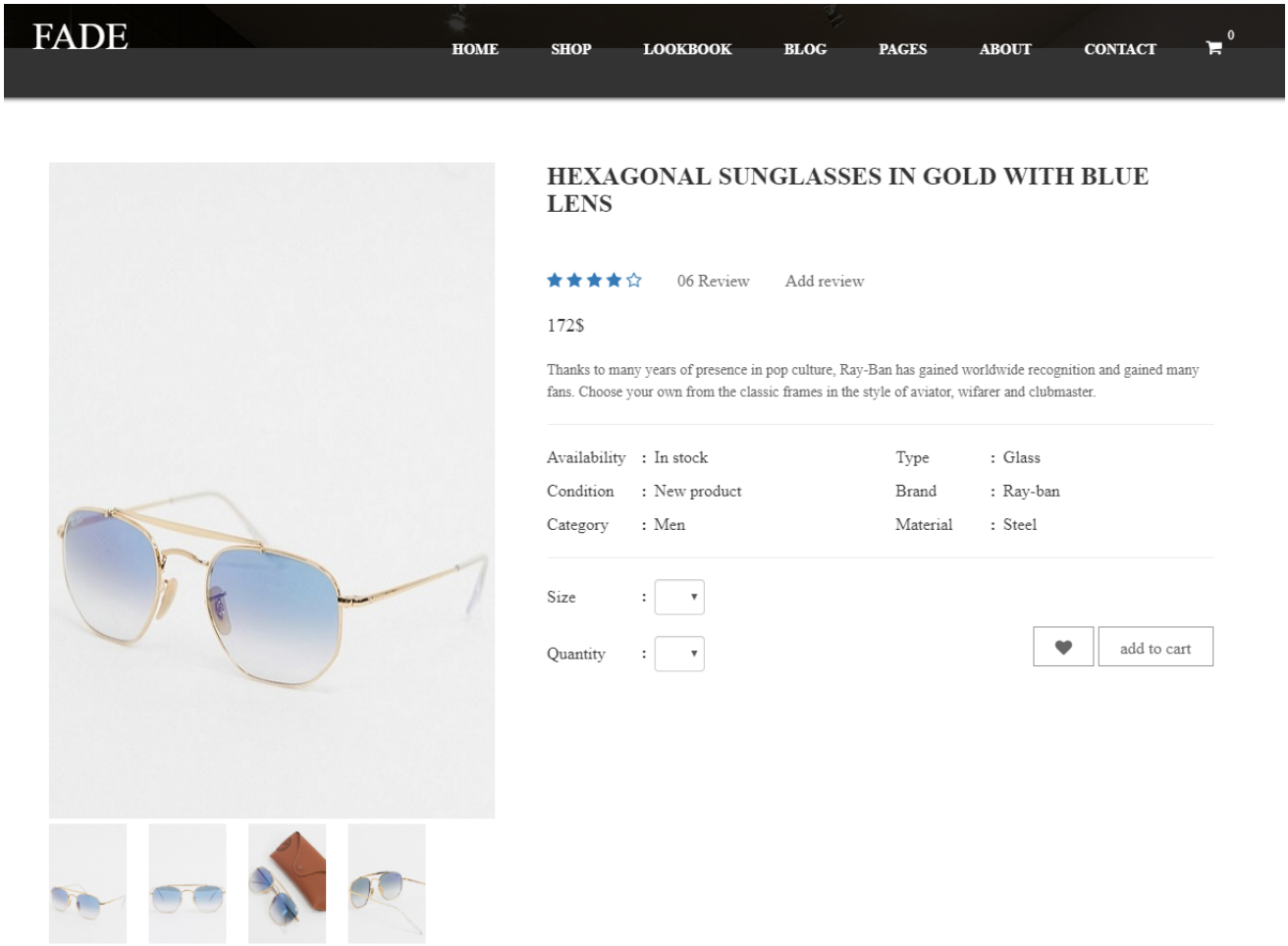


Рисунок 3.7 – Детальний перегляд товару (створення корзини)

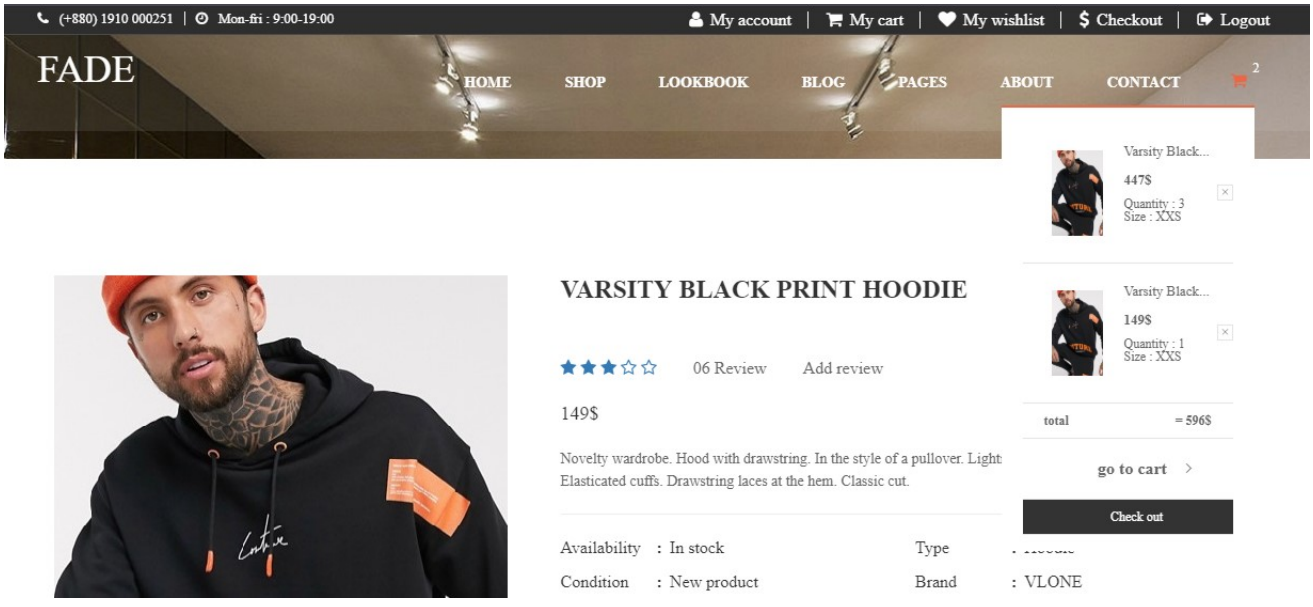


Рисунок 3.7 – Вигляд корзини після додавання товару у неї

Для адміністратора передбачений особливий доступ до функцій управління інтернет-магазином таки як: перегляд, видалення та редагування даних з певних таблиць бази даних. Приклад роботи адміністратора з сайтом зображено на рисунках 3.8-13.

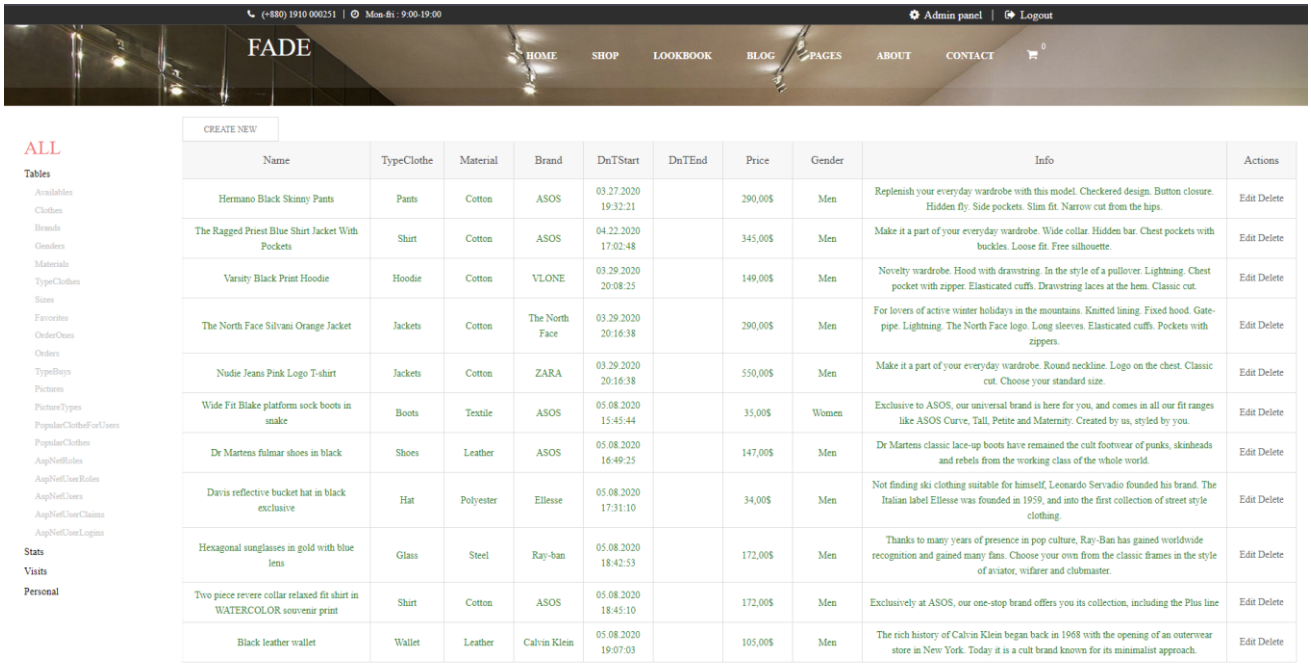


Рисунок 3.8 – Вигляд панелі адміністратора з обраним тегом для відображення даних з певної таблиці

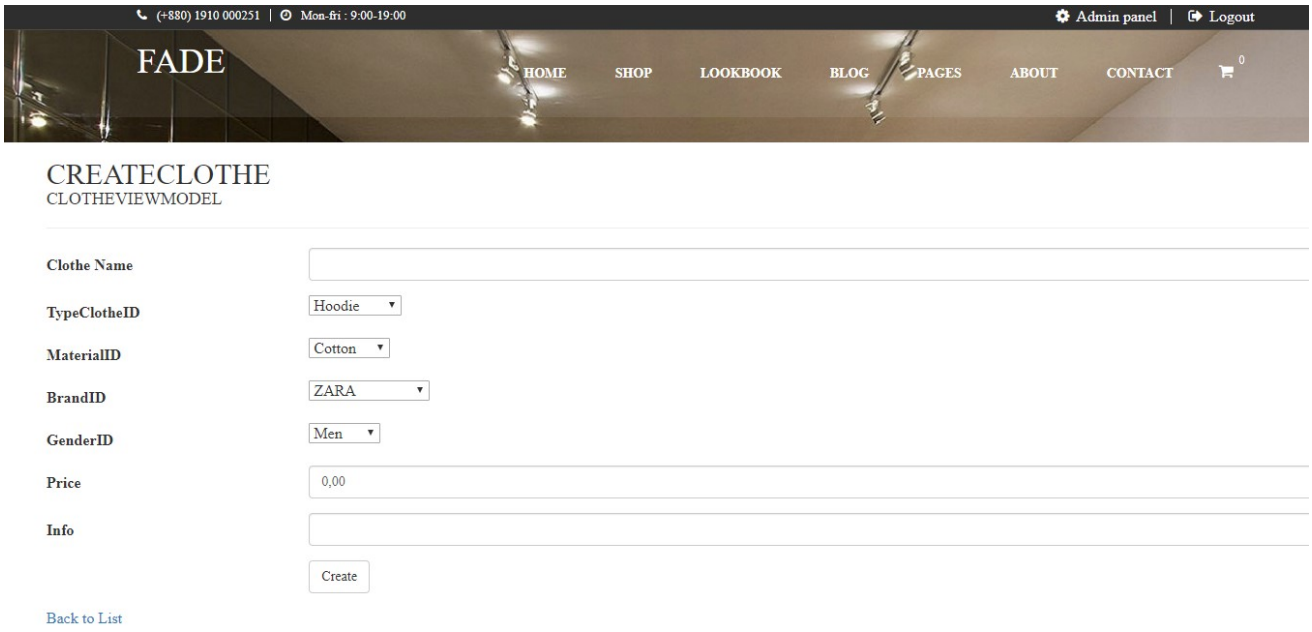


Рисунок 3.9 – Вигляд форми для додавання записів до певної таблиці

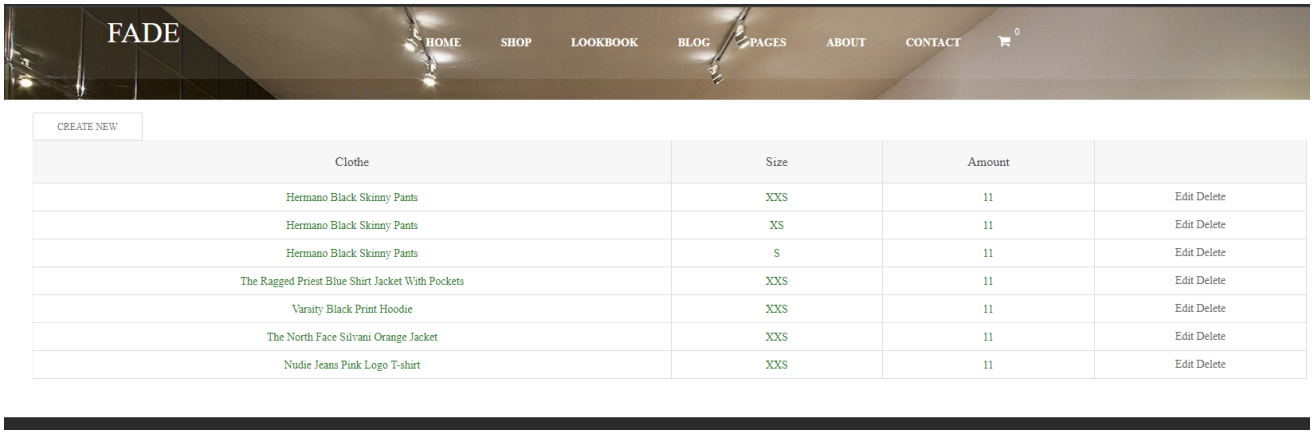


Рисунок 3.10 – Результат додавання нових записів до обраної таблиці

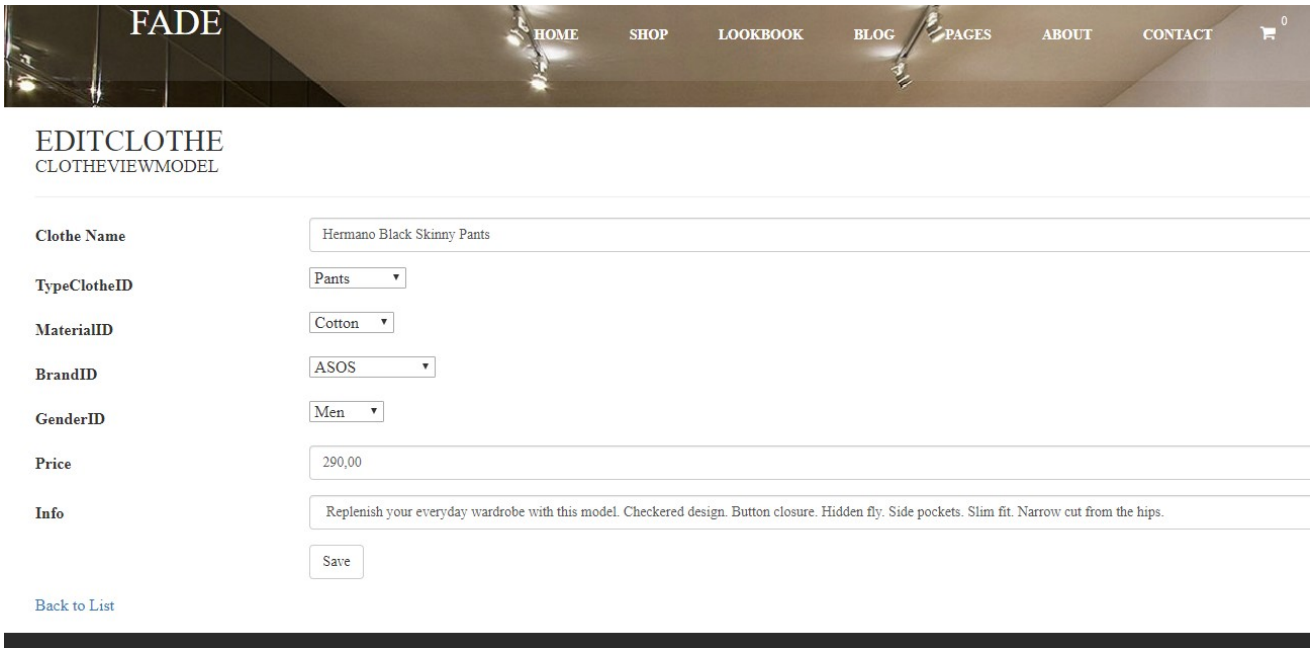


Рисунок 3.11 – Вигляд форми для редагування певних записів в обраній таблиці

За даною аналогією проводиться взаємодія адміністратора з базою даних (таблицями та записами у них), що є зручною практикою для управління інтернет-магазином.

3.2.2 Вітрина

Даний модуль відповідає за сторінки сайту на який відображена загальна інформація про магазин, сторінки для роботи з товаром (перегляд, детальна інформація, додавання товару у корзину, редагування корзини, створення замовлень). Також підключення різних виглядів сайту для різних груп користувачів, та показ деяких елементів (шапка, футер, панель навігації) на різних сторінках інтернет-магазину.

Головна сторінка сайту показана на рисунку 3.12.

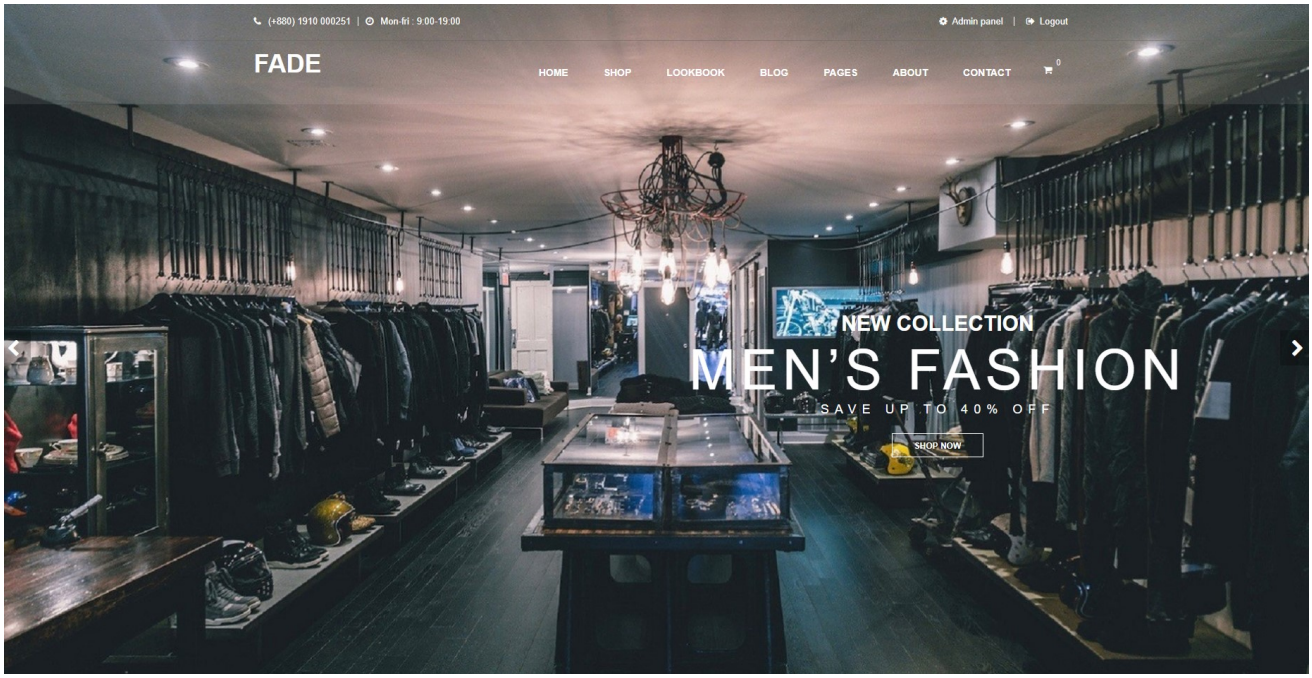


Рисунок 3.12 – Вигляд головної сторінки сайту

Сторінка з товаром на якій показано товар та навігацій панель в якій передбачено сортування товару за категоріями та пошук товару. Результат сортування товару на рисунку 3.13.

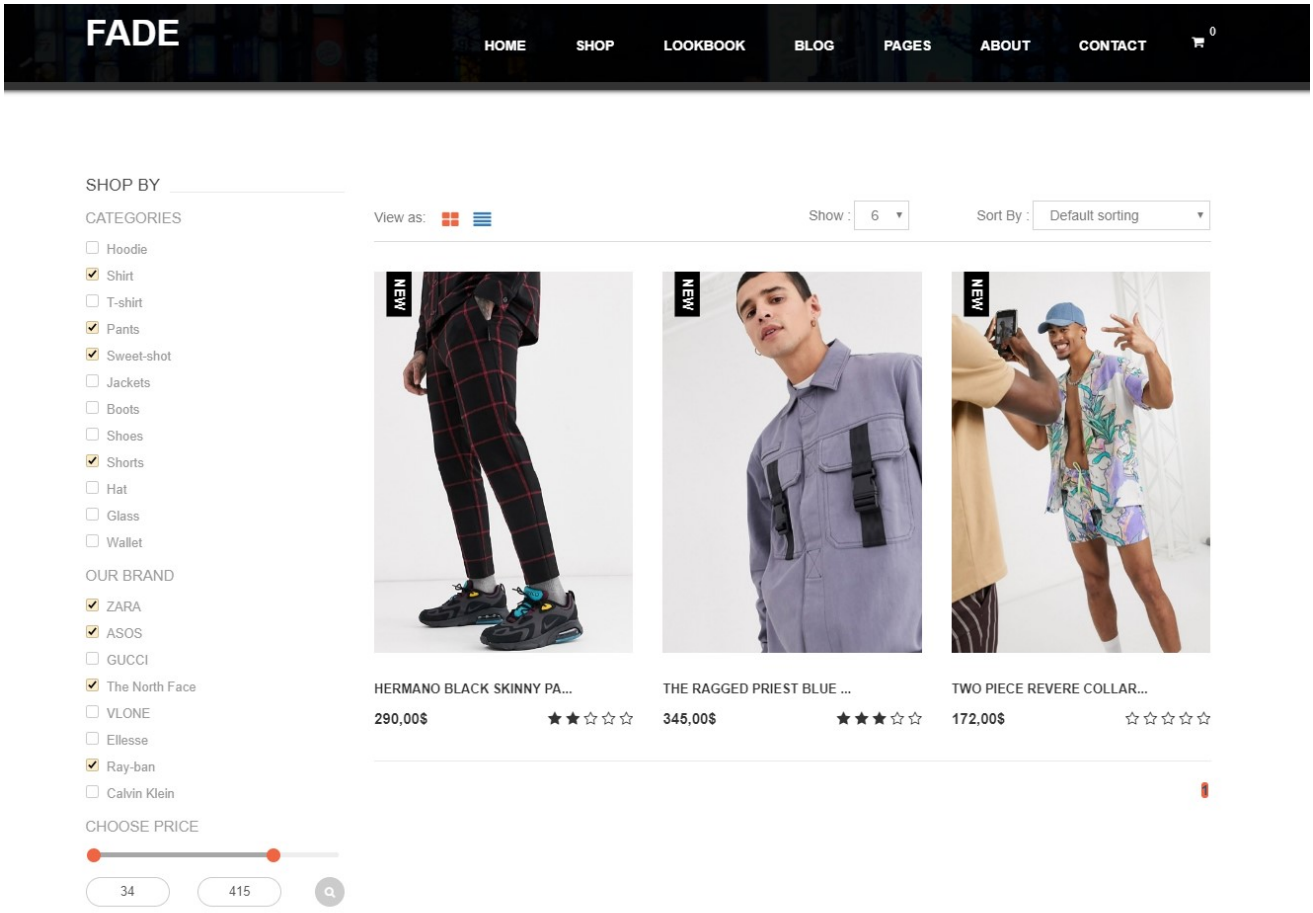


Рисунок 3.13 – Результат сортування товару за вказаним фільтром

Метод який відповідає за надання додаткової інформації та фото, подальше додання товару у корзину з певного розміру та кількості на рисунку 3.14.

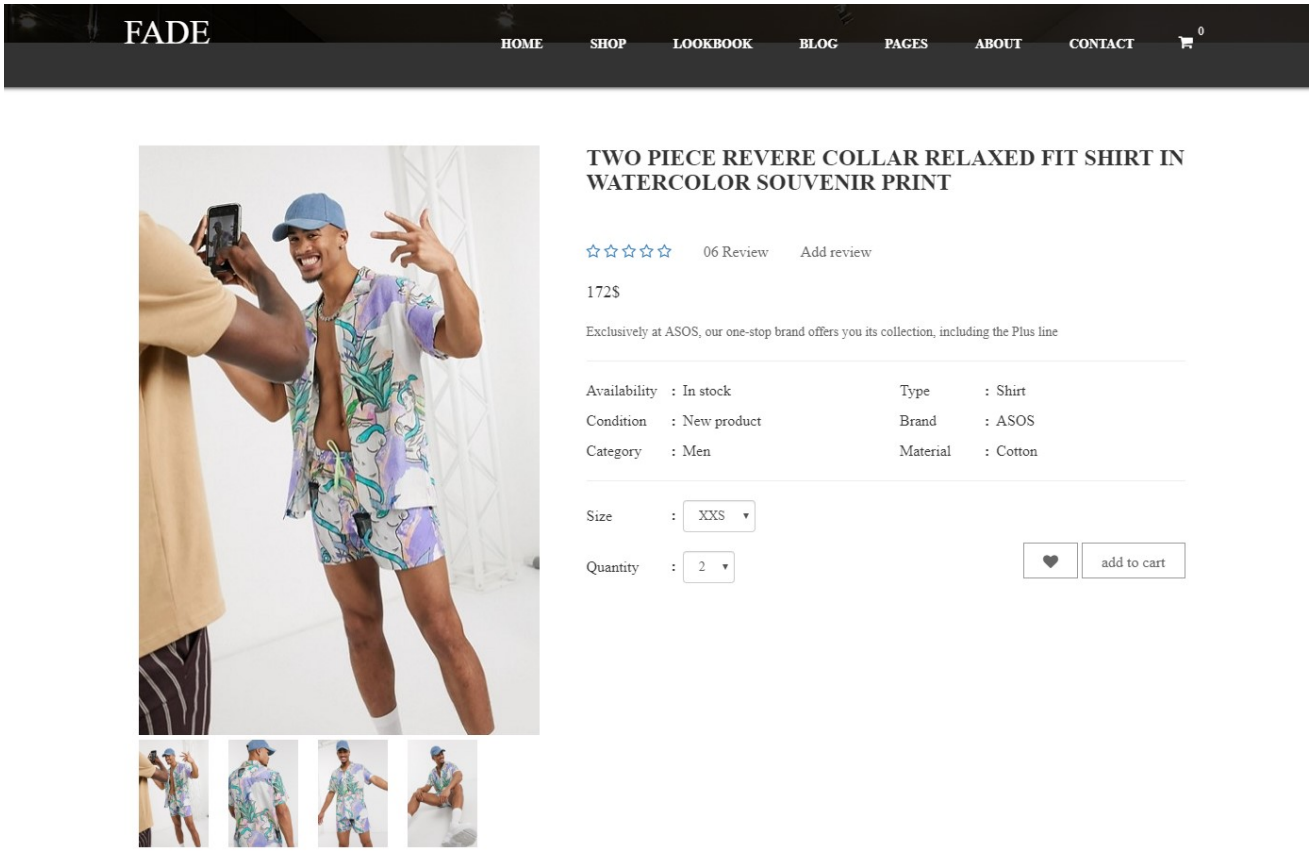


Рисунок 3.14 – Сторінка з додатковою інформацією про товар та вибором параметрів для додавання у корзину

Останнім елементом є корзина та створення замовлень. Результат додавання товару у корзину на рисунку 3.15.

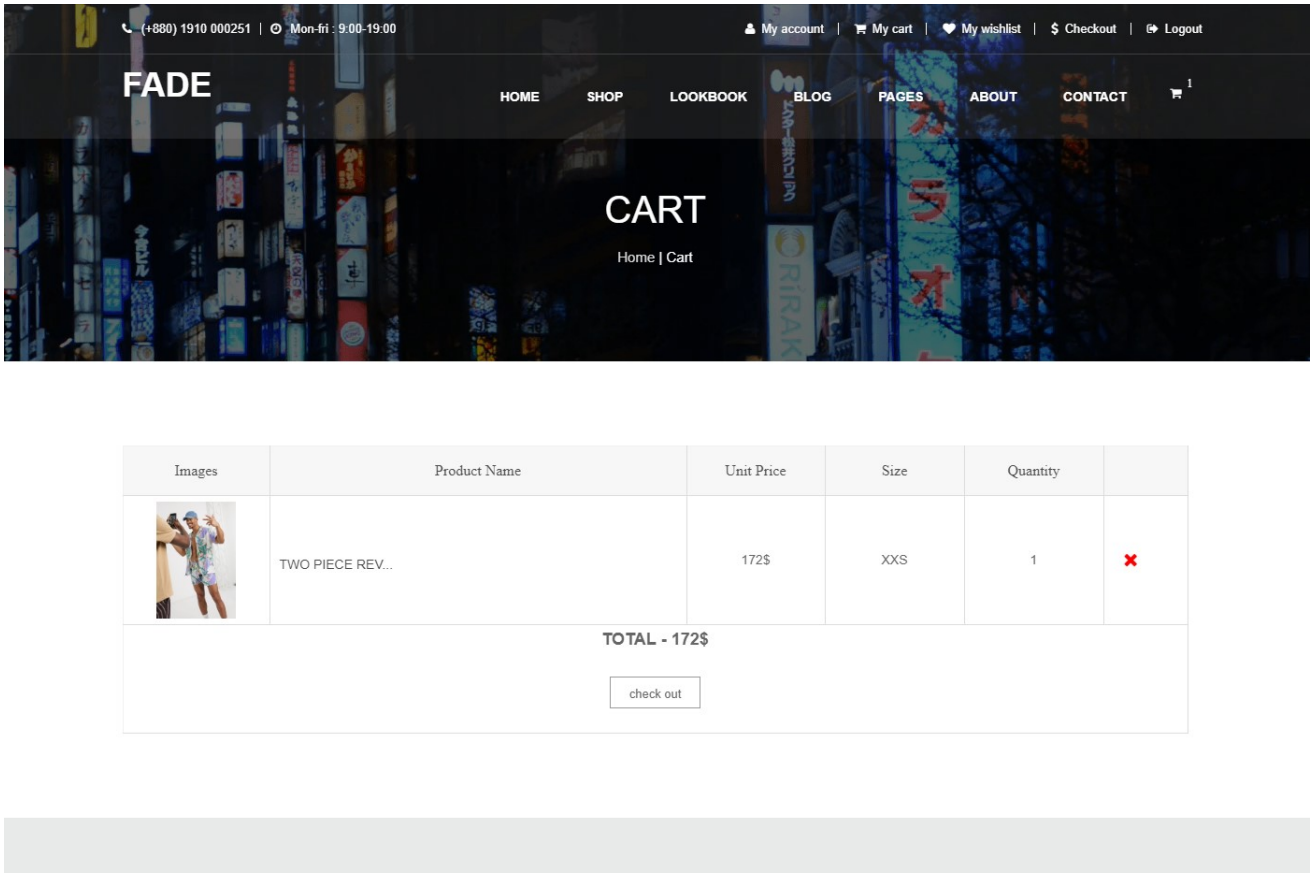


Рисунок 3.16 – Корзина після додавання товару у неї

Процес створення замовлення передбачає собою введення покупцем відповідних даних для заповнення форми та створення замовлення. Дана реалізація продемонстрована на рисунках 3.17-18.

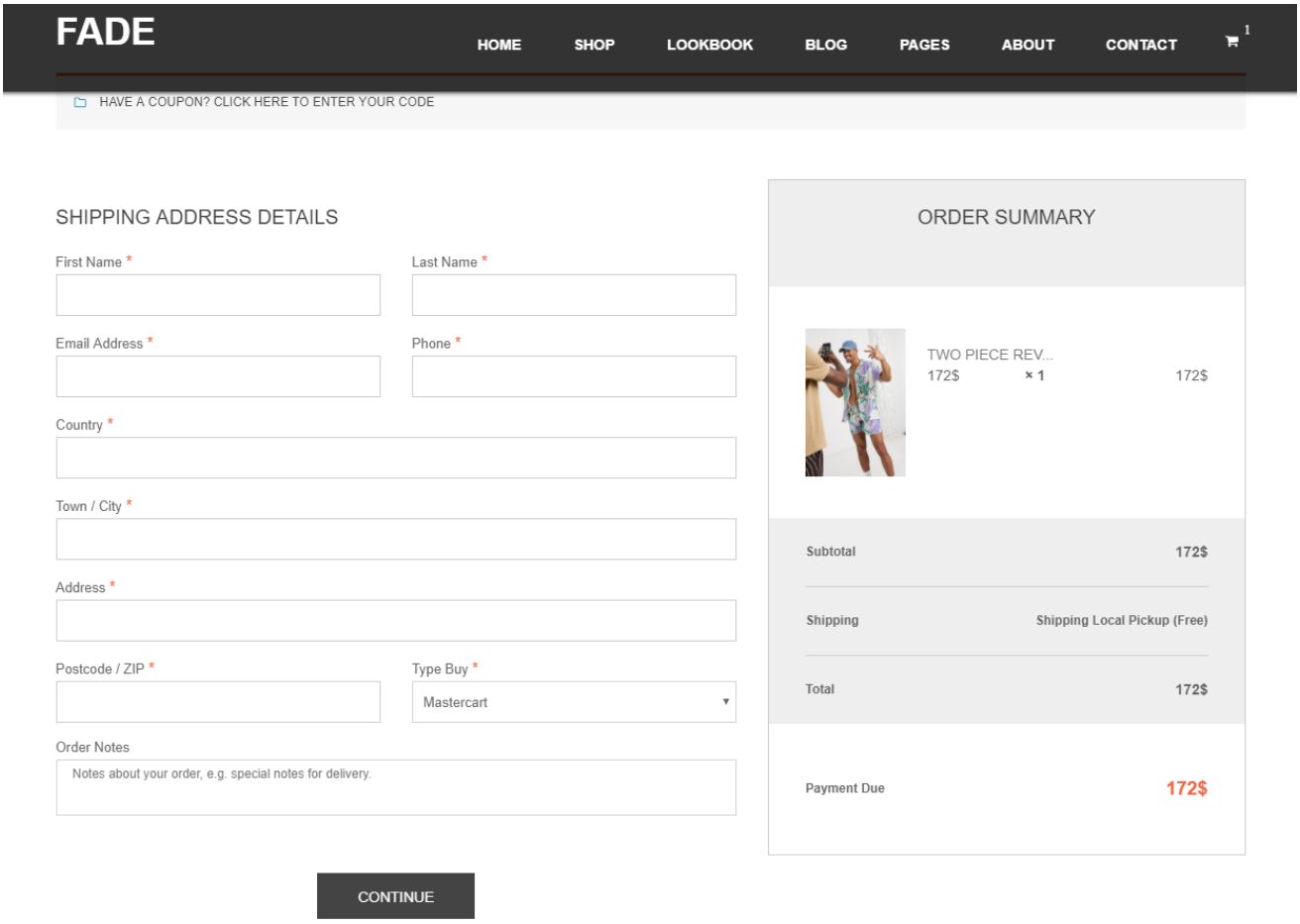


Рисунок 3.17 – Заповнення полів форми для створення замовлення

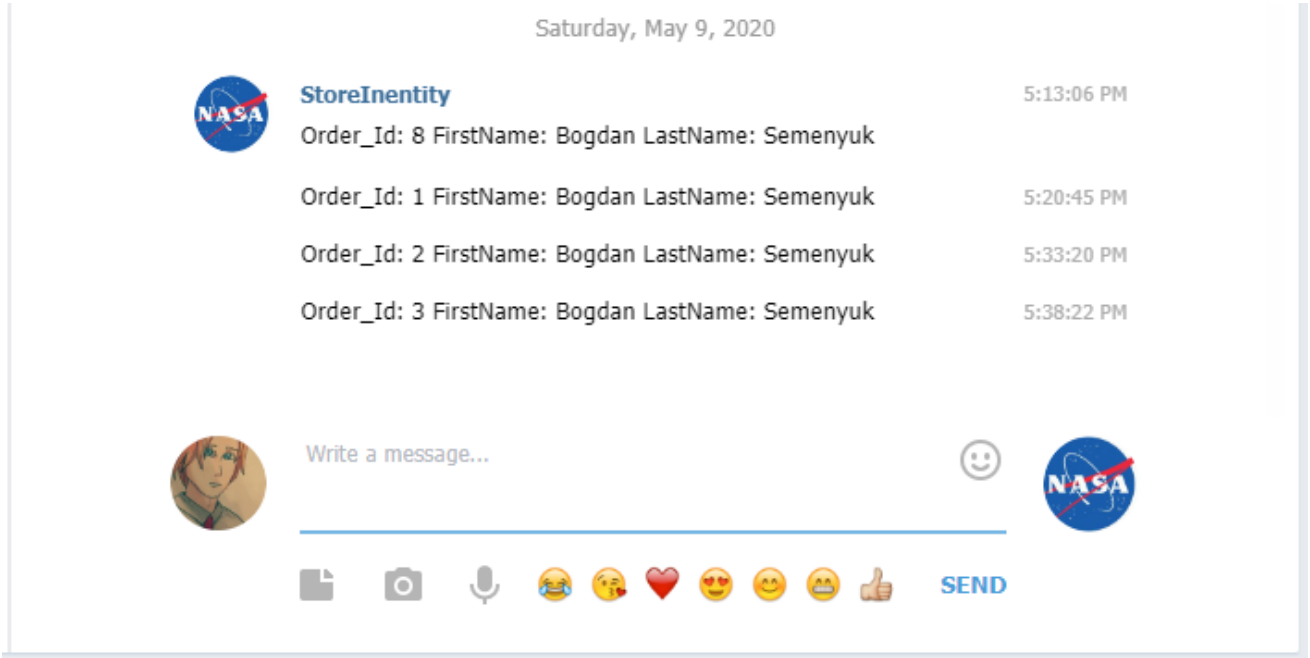


Рисунок 3.18 – Повідомлення від бота адміністратору в телеграмі про нове замовлення

3.2.3 Система підбору розміру

Розробка даної системи являє собою – відкритий сервіс (open api service), доступ до якого буду здійснюватися завдяки протоколу https, що є перевагою, оскільки даний додаток отримає підтримку косс-платформності та взаємодії з різного роду застосуваннями. Головним є збереження структури та обробки даних на клієнтській стороні, тому що засобом передачі слугує простий json-об’єкт. Для розробки буде використовуватися Visual Studio 2019, тип проекту Web Api .Net Core Application. Даний проект має наступну файлову структуру (рисунок 3.19).

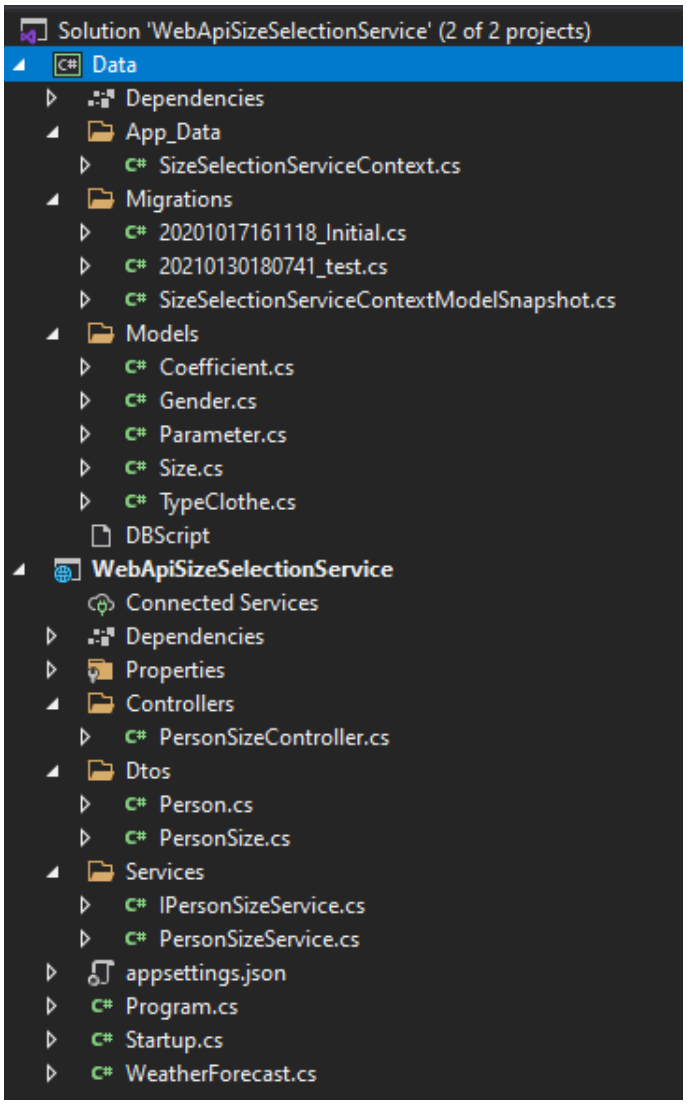


Рисунок 3.19 – Вигляд файлової структури проекту

Обрана структура – це простий та зрозуміли підхід для розділення представлення даних та функціональних частин на різні проекти. Що підвищує

читадельність коду та простору у розширенні. Оскільки тип проекту api application, то основними функціональними одиницями є контролери, моделі та у даному випадку сервіси. Сервіси – це один із видів структуризації управління та доступом до даних, певний сервіс типізується конкретної сутність та надає доступ до зчитування, створення, редагування та видалення даних, також може включати в себе допоміжні методи обробки інформації, як і було реалізовано. Основною особливість сервісної архітектури є інтерфейс та реалізатор, така реалізація дає змогу використати контейнери такі як (Dependency injection, Aurofac, Ninject). Отож, даний застосунок має в своєму арсеналі сервіс PersonSizeService, він вміщує в собі логіку за розрахування та обробку розмірів. В свою чергу для отримання даних слугує апі контроллер, приймає дані через метод пост і якщо дані валідні відсилає відповідь з прогнозами розмірів. Результат розрахунків наведено на рисунку 3.20.

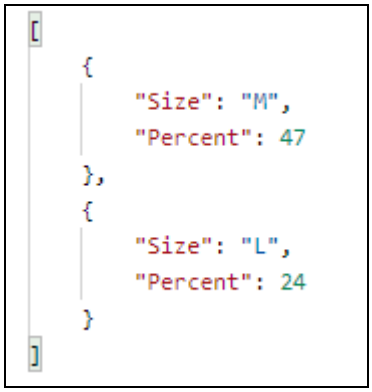


Рисунок 3.20 – Вигляд файлової структури проекту

Результат, як можна спостерігати на рисунку 3.23 вдалий але потребує графічного представлення, для зручності та зрозумілості користувача. Для розробки інтерфейсу простим та швидким підходом буде використати прості html сторінки в поєднанні з скриптами та асинхронними запитами. Вигляд – це дещо вдосконалена форма у вигляді повзунків, шо має інтуїтивний дизайн для користувача (рисунок 3.21).

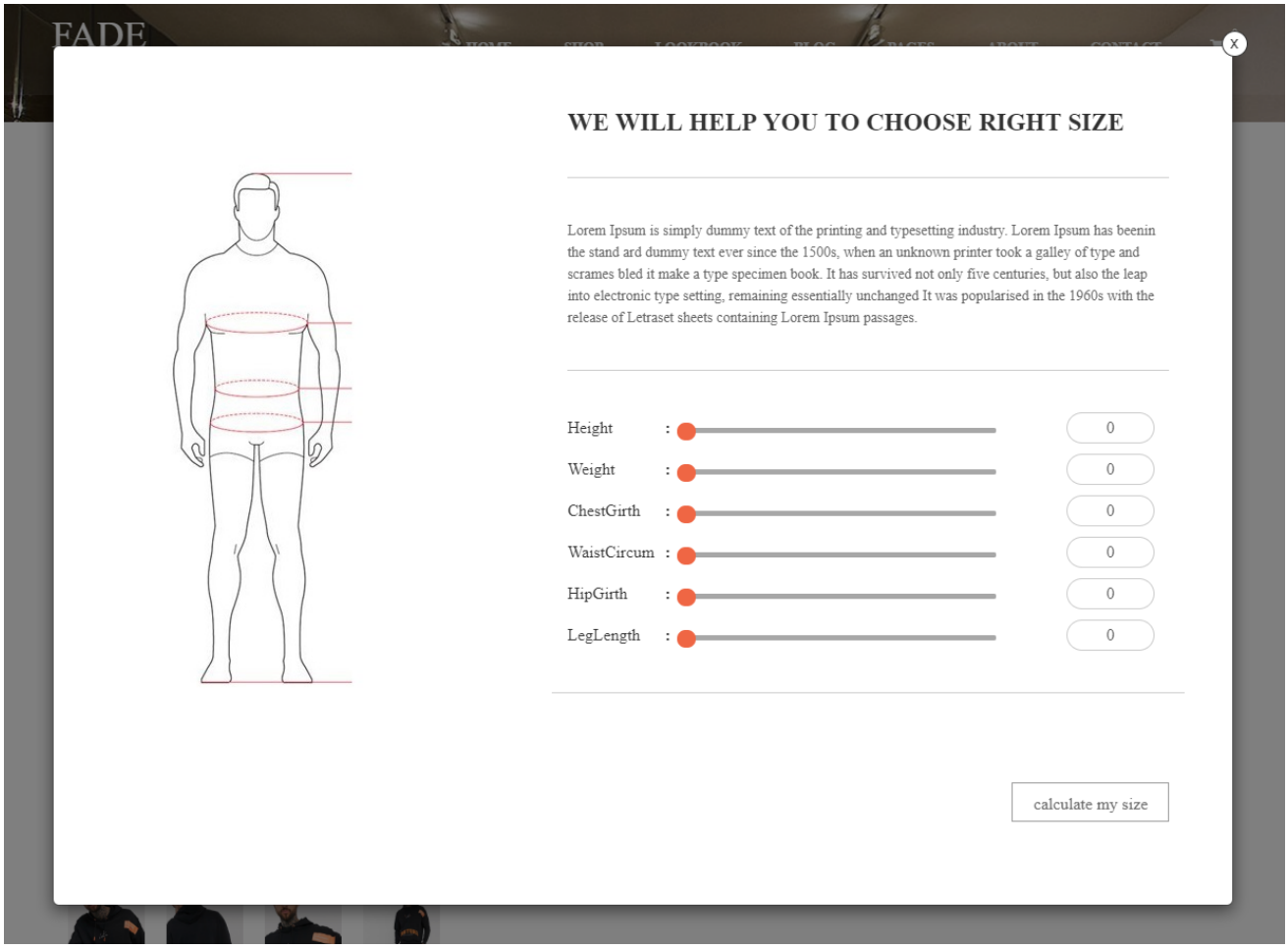


Рисунок 3.21 – Графічний інтерфейс системи підбору розмірів

Тепер є можливість заповнити дані форми відповідно до особистих параметрів обраного типу одягу. Наглядний приклад заповнення на рисунку 3.22.

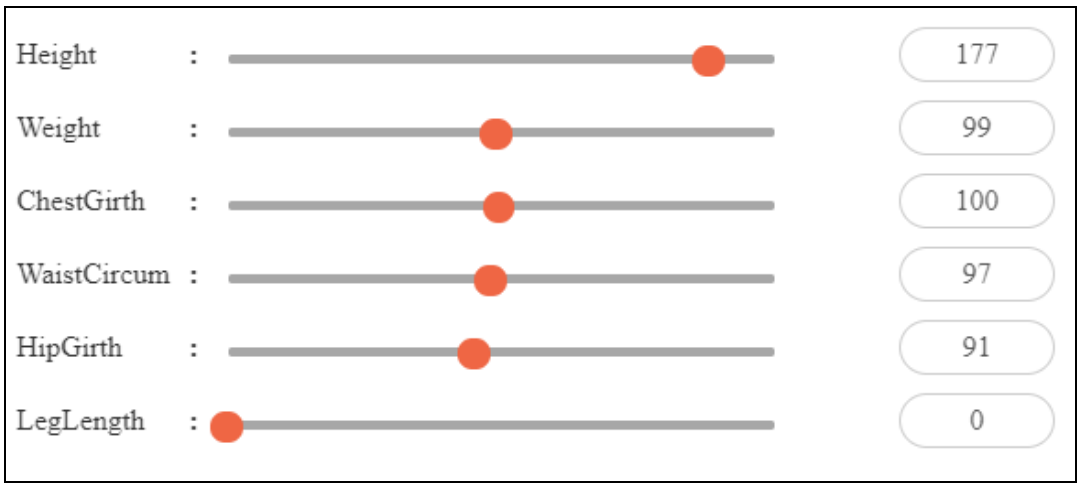


Рисунок 3.22 – Приклад заповнення форми особистими даними

Форма готова, отже, можна відправляти дані на розрахування розміру, після цього юзер отримає відповідь з результатами підбору. Їх може бути два:

успішний (рисунок 2.23), коли виявлено вдалий набір розмірів, або ж невдалий (рисунок 2.24), коли збігів не знайдено.

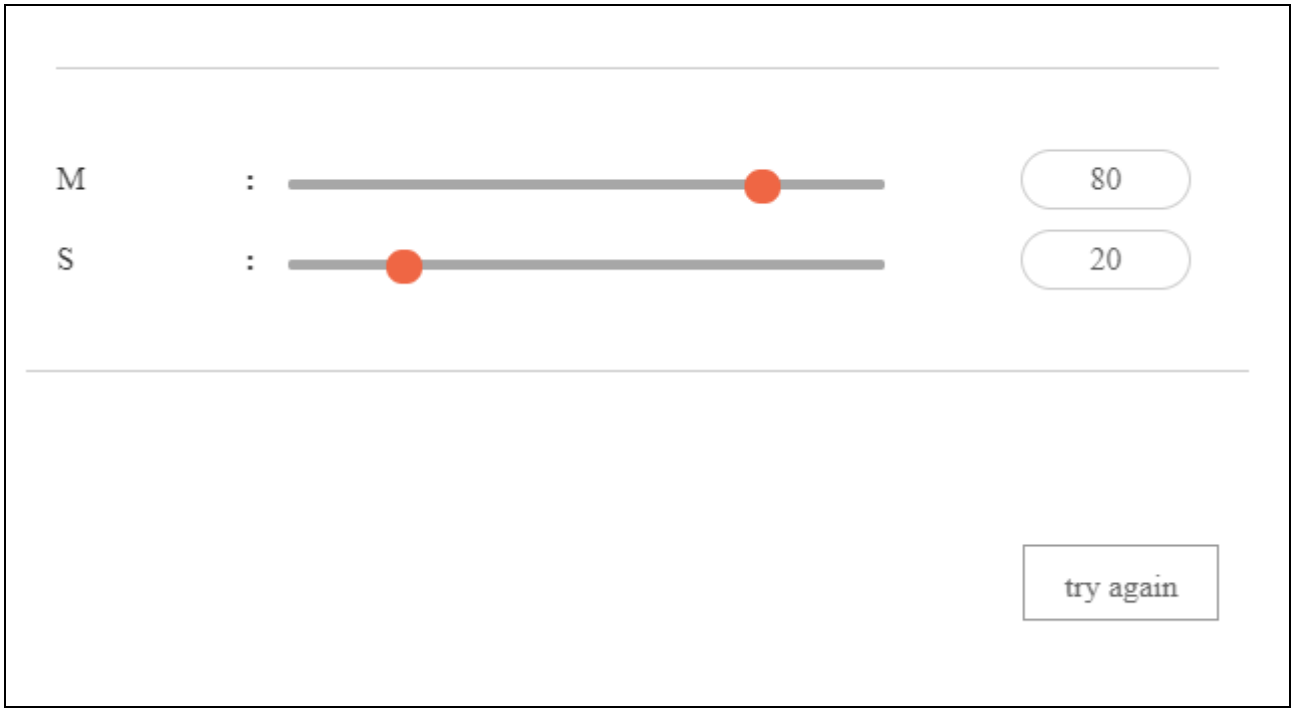


Рисунок 3.23 – Приклад успішної відповіді від сервісу

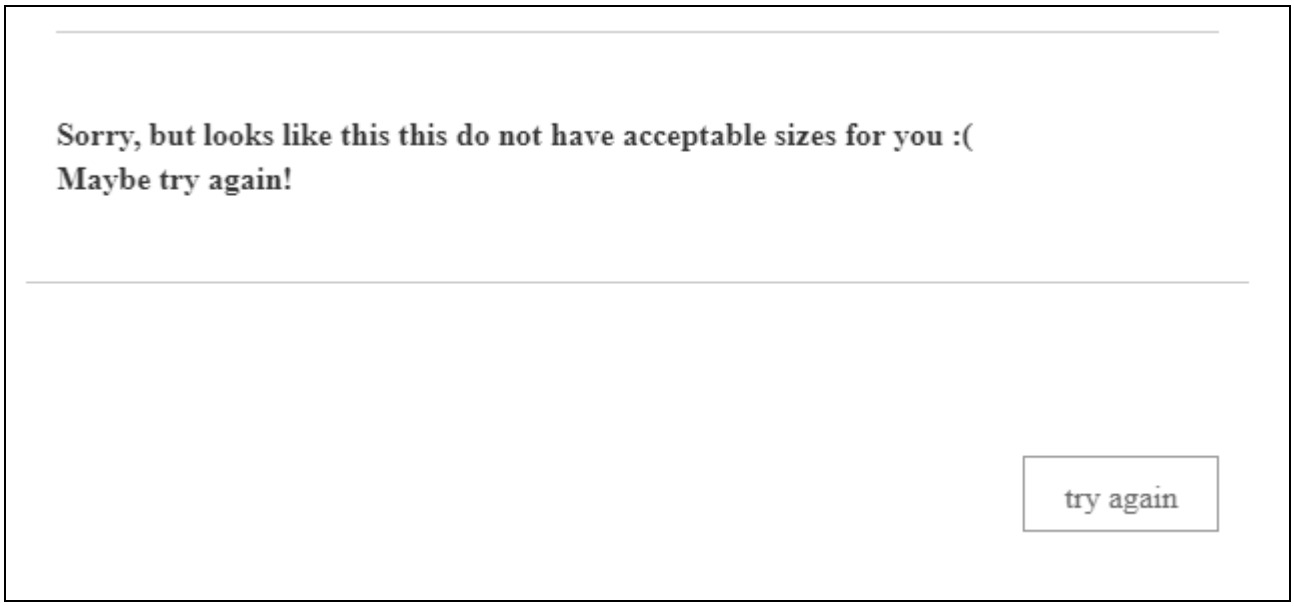


Рисунок 3.24 – Приклад поганої відповіді від сервісу

Останнє, це перевірка експертної системи на працездатність, тобто якщо вона активна і працездатна то юзер має можливість до взаємодії, в іншому випадку слід вивести відповідне повідомлення про недоступність даної процедури підбору, приклад на рисунку 3.25.

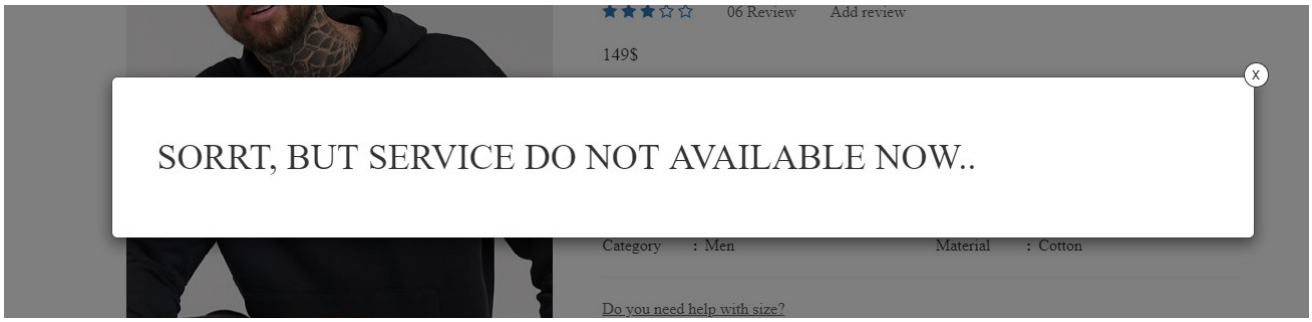


Рисунок 3.25 – Приклад заглушки про те що сервіс тимчасово недоступний

3.3 Інструкція користувача

Для користування даним інтернет-магазином необхідно провести процес реєстрації (рисунок 3.26).

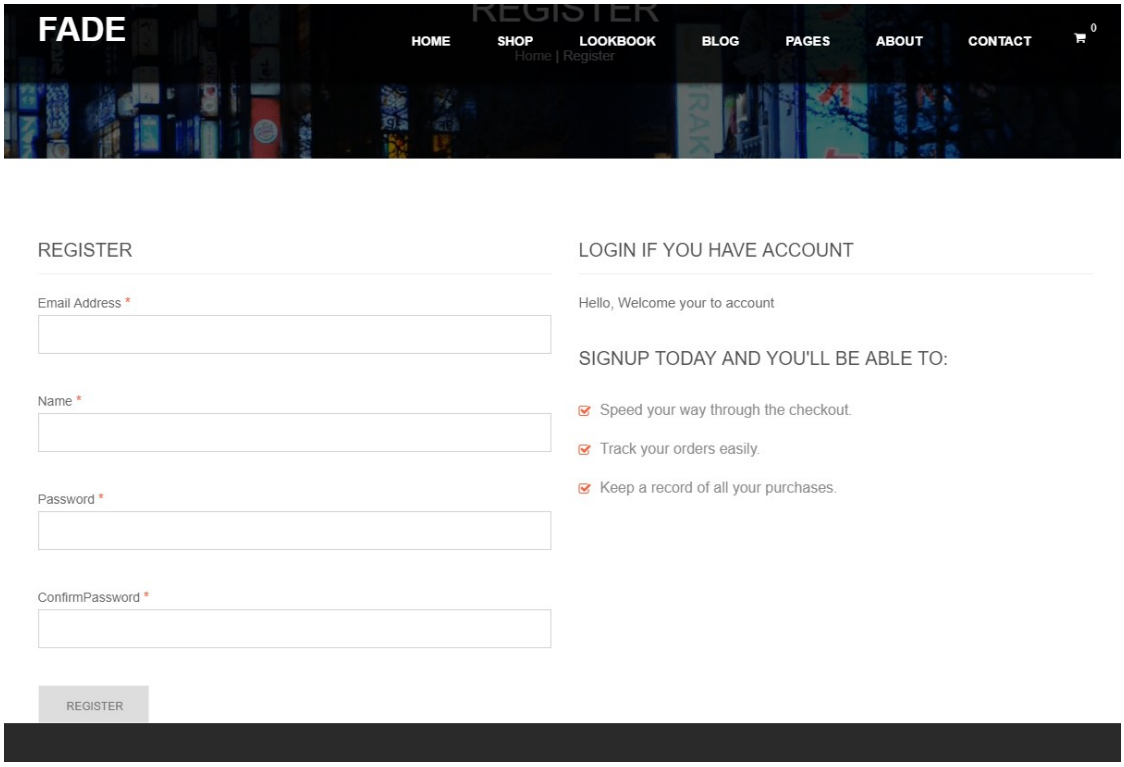


Рисунок 3.26 – Форма для реєстрації нового користувача.

Подальша робота користувача з інтернет-магазин передбачає собою процес покупки (рисунки 3.27-30).

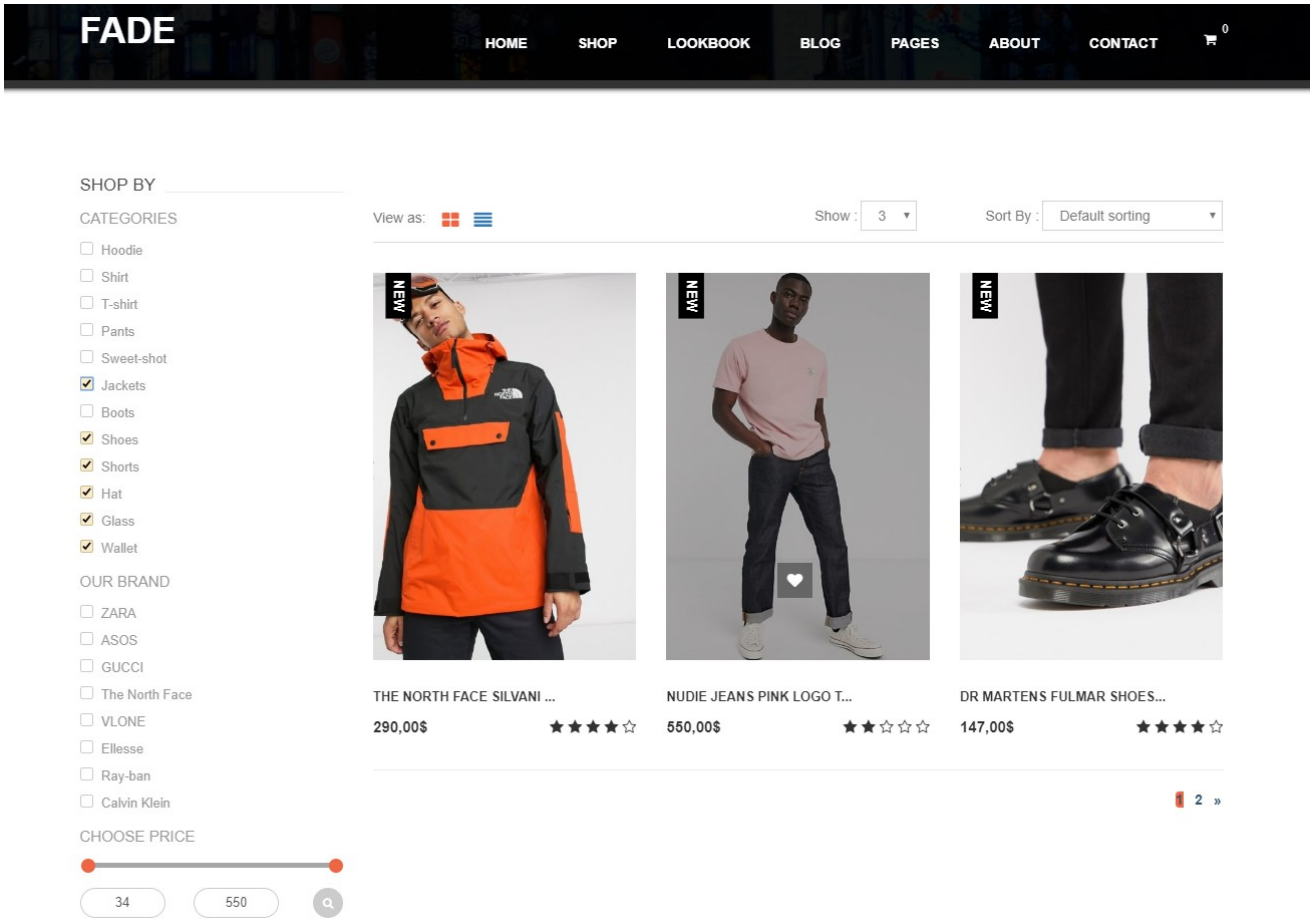


Рисунок 3.27 – Сторінка з товаром магазину.

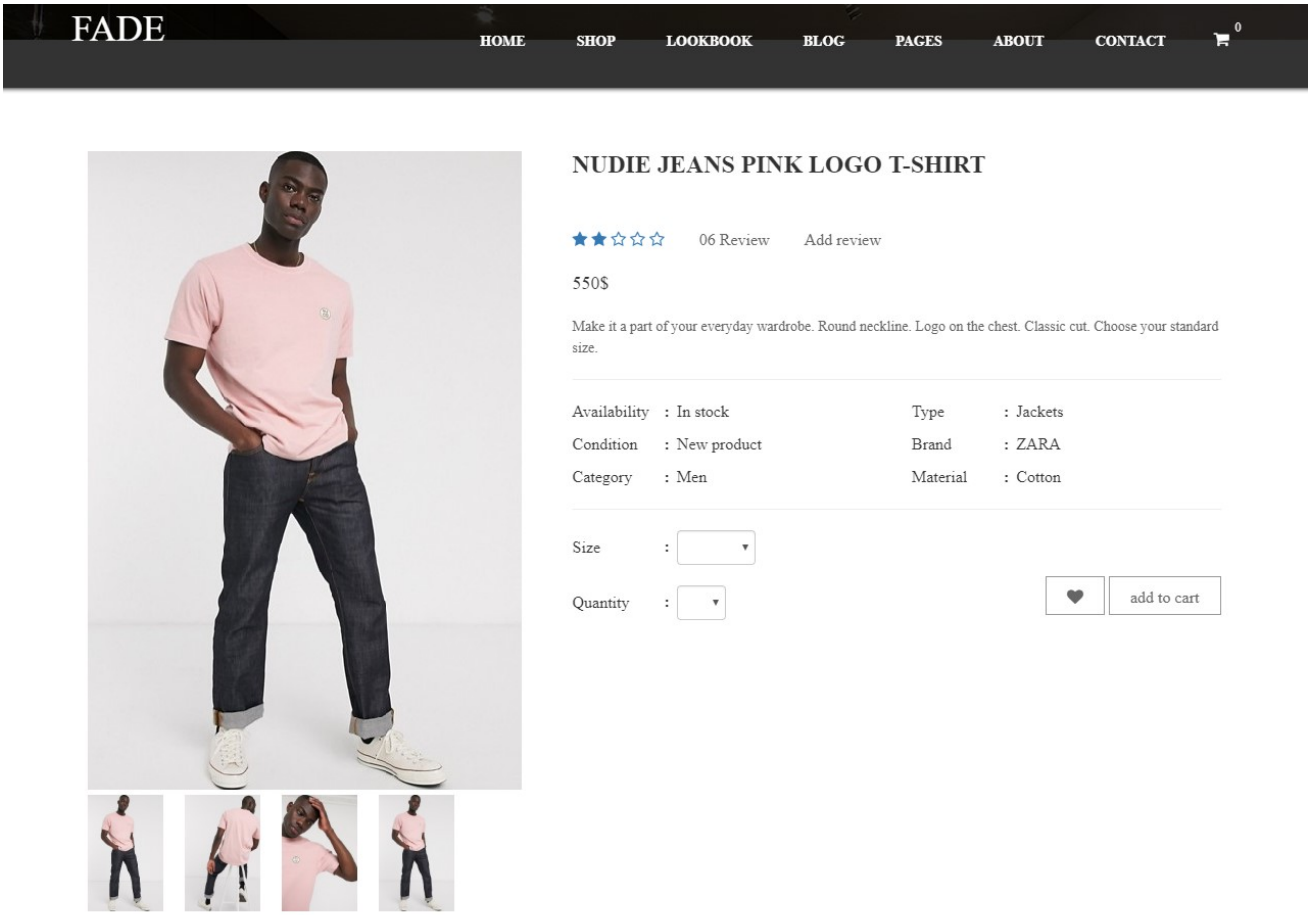


Рисунок 3.28 – Сторінка з детальною інформацією про товар.

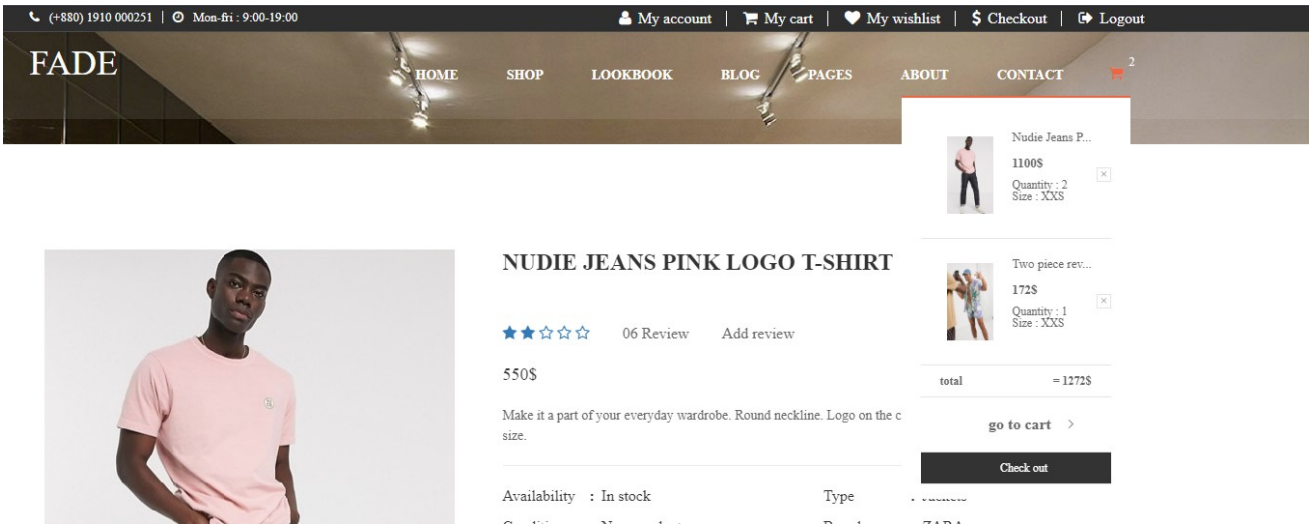


Рисунок 3.29 – Вигляд корзини яка містить одну одиницю товару.

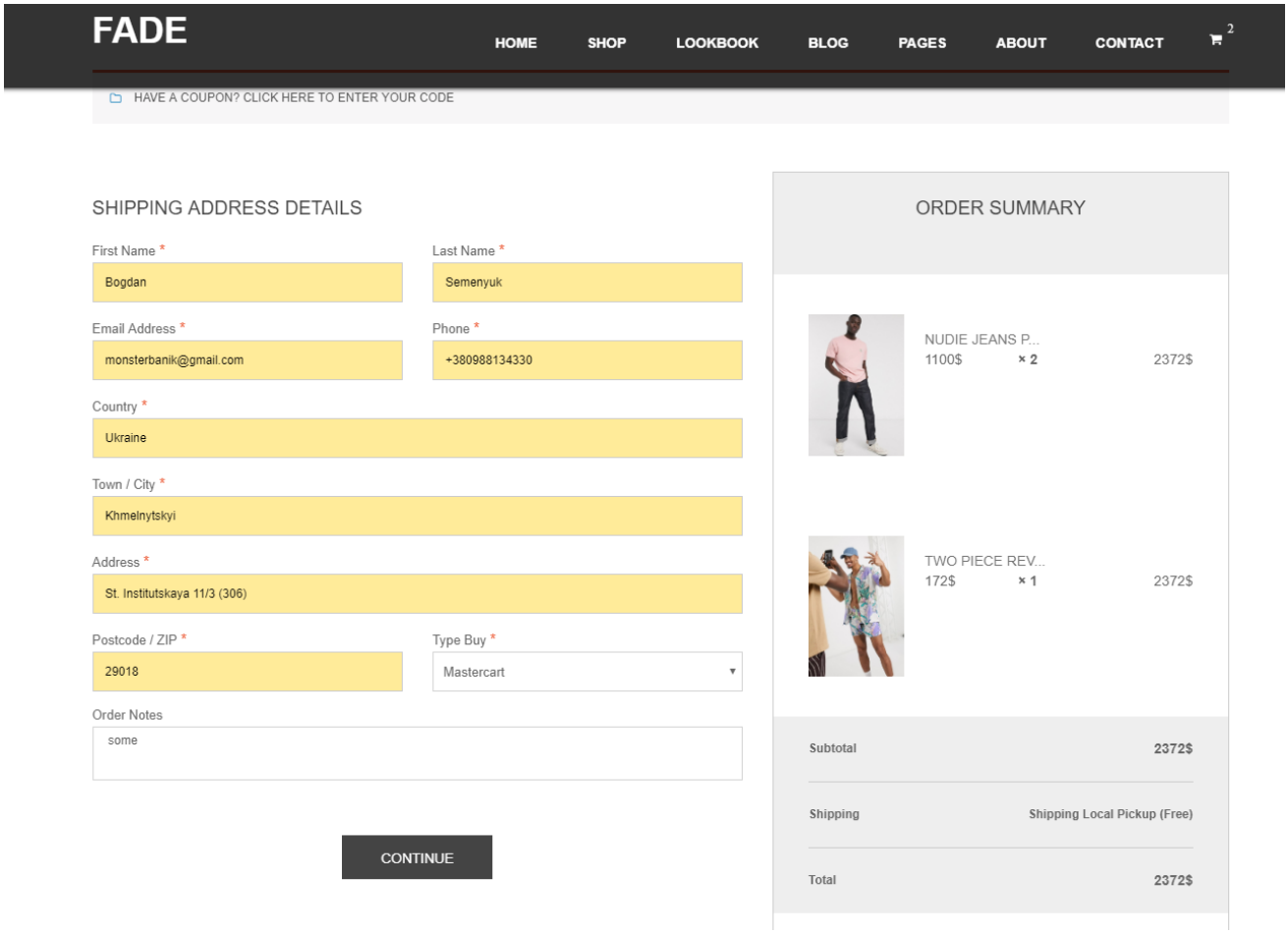


Рисунок 3.30 – Форма для оформлення замовлення.

Вище продемонстрований процес покупки є основною частиною роботи користувача з сайтом, але також можна ще дізнаватися новини про товар і магазин з головної сторінки, що планується реалізувати в подальшій розробці.

3.4 Тестування

Для даного застосунку було створено ряд тестів для перевірки цілісності даних, які повертають методи контролерів.

Приклад тесту для панелі адміністратора (вивід інформації з певної таблиці) зображено на рисунках 3.31-32.

```
[TestMethod]
ссылок: 0
public void Can_Get_ListClothes()
{
    AdminController controller = new AdminController();
    // Act
    Task<ActionResult> result = controller.ListClothes() as Task<ActionResult>;
    // Assert
    Assert.IsNotNull(result);
}

[TestMethod]
ссылок: 0
public void Can_Get_ListBrands()
{
    AdminController controller = new AdminController();
    // Act
    Task<ActionResult> result = controller.ListBrands() as Task<ActionResult>;
    // Assert
    Assert.IsNotNull(result);
}
```

Рисунок 3.31 – Тести для перевірки результату роботи методів адмін-контролера.

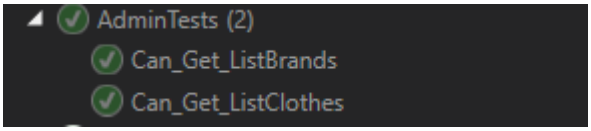


Рисунок 3.32 – Повідомлення про успішне проходження тестів.

Також для роботи користувача з вітриною було створено декілька тестів. Перевірка роботи методу, який відповідає повертає товар з різними фільтрами та розбиває на сторінки (рисунки 3.33-34).

```
[TestMethod]
ссылок: 0
public void Can_Get_ClotheFilter()
{
    ScaseController controller = new ScaseController();
    // Act
    Task<ActionResult> result = controller.ClotheFilter(2, 1, 400.0m, 100.0m, 1, "1", "2", null) as Task<ActionResult>;
    // Assert
    Assert.IsNotNull(result);
}
```

Рисунок 3.33 – Тест методу який повертає колекцію товару.

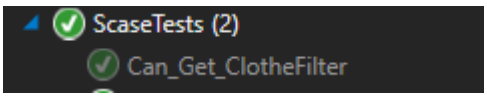


Рисунок 3.34 – Повідомлення про успішне проходження тесту методу вітрина-контролера.

3.5 Вимоги до апаратних та програмних засобів

Вимоги до програмних засобів:

- підтримка браузером HTML, Flash, JavaScript.

Вимоги до апаратних засобів:

- кількість ядер процесора – два і більше;
- частота процесору – 1.7 (Гц) і більше;
- простір у фізичному накопичувачі, не менше 20 (Гб);
- оперативна пам’ять – 8 (Гб) чи більше;
- MS SQL Server – версія не менше 2018 року;
- Windows Server – версія не менше 2012 року.

Висновки

В результаті виконання даного КРБ було отримано web-застосунок інтернет-магазин одягу, розроблений на базі таких програмних засобів: мова C#, ASP .NET MVC 5 і API, MS SQL Server та Entity Framework 6. Перелік усіх поставлених вимог до роботи, було виконано, а саме: було створено розділений веб-додаток магазину, базу даних, процедури бізнес-логіки для різних груп користувачів, авторизація та аутентифікація, автоматизований процес продажу товару.

Також сповна було реалізовано експерту систему підбору розмірів для певного одягу, дана система була реалізована у вигляді розділеного сервісу за допомогою технології Web API.

В перспективі, якщо для даного сайту можна придбати хостинг, варто додати підтримку (адміністрацією сайту для користувачів, у вигляді гарячої телефонної лінії) та налагодити обслуговування (процесу додавання продукції для продажу) для реальних задач.

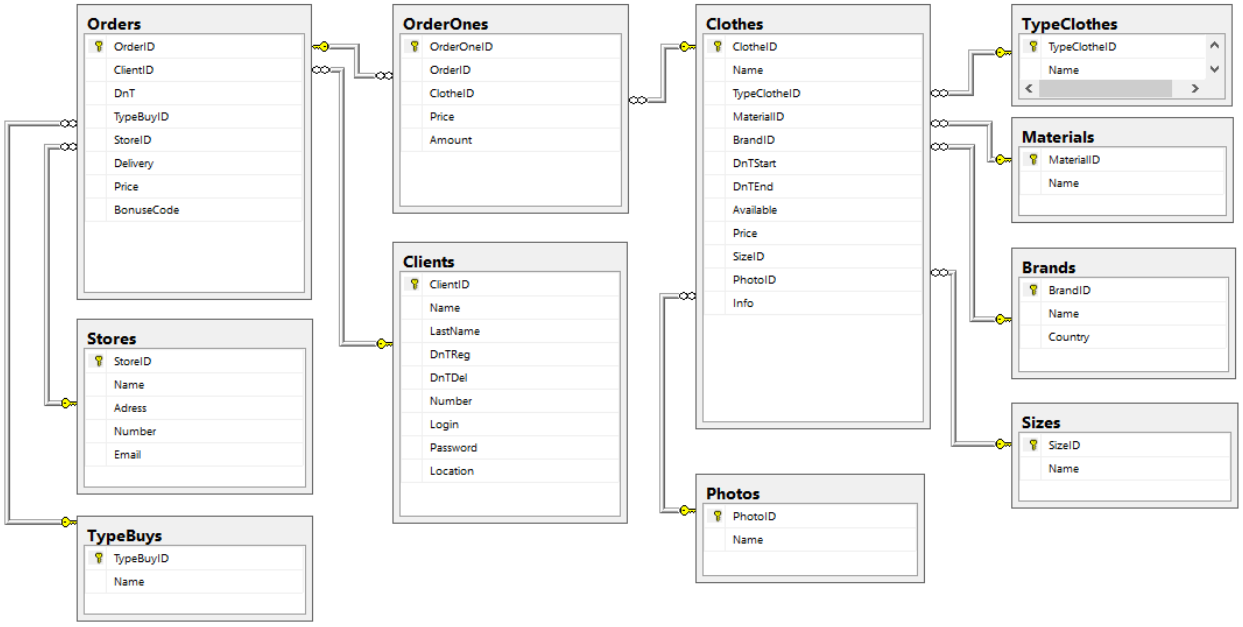
Перелік посилань

1. Wikipedia.com. URL: https://uk.wikipedia.org/wiki/Електронна_комерція
2. Wikipedia.com. URL: https://uk.wikipedia.org/wiki/Інтернет_магазин
3. Wikipedia.com. URL: https://uk.wikipedia.org/wiki/Електронний_магазин
4. Wikipedia.com. URL: https://uk.wikipedia.org/wiki/Одяг
5. Wikipedia.com. URL: https://uk.wikipedia.org/wiki/Покупець
6. Wikipedia.com. URL: https://uk.wikipedia.org/wiki/Бренд
7. Wikipedia.com. URL: https://uk.wikipedia.org/wiki/Мода
8. Wikipedia.com. URL: https://uk.wikipedia.org/wiki/Колекція
9. Wikipedia.com. URL: https://uk.wikipedia.org/wiki/Стиль одягу
10. ZARA.com. URL: https://www.zara.com/ua/ru
11. ASOS.com. URL: https://www.asos.com
12. Wikipedia.com. URL: https://uk.wikipedia.org/wiki/Веб-застосунок
13. Microsoft. URL: https:// www.microsoft.com/net
14. Wikipedia.com. URL: https://uk.wikipedia.org/wiki/php
15. Wikipedia.com. URL: https://uk.wikipedia.org/wiki/c#
16. METANIT. URL: https://metanit.com/sharp/aspnet5/1.1234234.php
17. METANIT. URL: https://metanit.com/sharp/aspnet_webapi/1.1.php
18. METANIT. URL: https://metanit.com/sharp/mvc5
19. Wikipedia.com. URL: https://uk.wikipedia.org/wiki/mvc
20. Wikipedia.com. URL: https://uk.wikipedia.org/wiki/sql_server
21. Wikipedia.com. URL: https://uk.wikipedia.org/wiki/mysql
22. Wikipedia.com. URL: https://uk.wikipedia.org/wiki/entity_framework
23. Wikipedia.com. URL: https://uk.wikipedia.org/wiki/Microsoft_Visual_Studio

ДОДАТКИ

Додаток А

Структура бази даних інтернет-магазину



Додаток Б

Програмні коди

Лістинг AccountController.cs:

```
using Srore.Models;
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Security;
using System.Data.Entity;
using System.Net;

namespace Srore.Controllers
{
    public class AccountController : Controller
    {
        DBCon db = new DBCon();
        public ActionResult Login()
        {
            return View();
        }


        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Login(LoginModel model)
        {
            if (ModelState.IsValid)
            {
                // поиск пользователя в бд
                Client client = null;
                using (DBCon db = new DBCon())
                {
                    client = db.Clients.FirstOrDefault(u =>
u.Email == model.Email && u.Password == model.Password);

                }
                if (client != null && client.RoleID == 2)
                {

FormsAuthentication.SetAuthCookie(model.Email, true);
                    return RedirectToAction("Main", "Scase");
                }
                else if (client != null && client.RoleID == 1)
                {

FormsAuthentication.SetAuthCookie(model.Email, true);
                    return RedirectToAction("List", "Admin");
                }
                else
                {
                    ModelState.AddModelError("", "There is no
user with this username and password");
                }
            }

            return View(model);
        }

        public ActionResult Register()
        {
            return View();
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Register(RegisterModel model)
        {
            if (ModelState.IsValid)
            {
                Client client = null;
                using (DBCon db = new DBCon())
                {
                    client = db.Clients.FirstOrDefault(u =>
u.Email == model.Email);
                }
                if (client == null)
                {
                    // создаем нового пользователя
                    using (DBCon db = new DBCon())
                    {
                        db.Clients.Add(new Client { Name =
model.Name, LastName = model.LastName, Age = model.Age, Number
= model.Number, Location = model.Location, DnTReg =
DateTime.Now.ToString("MM/dd/yyyy HH:mm:ss"), DnTDel = null,
Email = model.Email, Password = model.Password, RoleID = 2 });
                        db.SaveChanges();

                        client = db.Clients.Where(u => u.Email
== model.Email && u.Password ==
model.Password).FirstOrDefault();
                    }
                    // если пользователь удачно добавлен в бд
                    if (client != null)
                    {

FormsAuthentication.SetAuthCookie(model.Name, true);
                        return RedirectToAction("Main",
"Scase");
                    }
                }
                else
                {
                    ModelState.AddModelError("", "User with
this login already exists");
                }
            }

            return View(model);
        }
        public ActionResult Logoff()
        {
            FormsAuthentication.SignOut();
            return RedirectToAction("Main", "Scase");
        }

        [Authorize]
        public ActionResult About()
        {
            string mail = User.Identity.Name;
            var selectedUsers = from client in db.Clients
                                where client.Email == mail
                                select client;
              return View(selectedUsers.ToList());
        }

        [HttpGet]
        [Authorize]
        public ActionResult Edit(int? id)
        {
            if (id == null)
            {
                return new
HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Client client = db.Clients.Find(id);
            if (client == null)
            {
                return HttpNotFound();
            }
            return View(client);
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit([Bind(Include =
"ClientID,Name,LastName,Age,Number,Location,DnTReg,DnTDel,Emai
l,Password,RoleID")] Client client)
        {
            if (ModelState.IsValid)
            {
                db.Entry(client).State = EntityState.Modified;
                db.SaveChanges();
                return RedirectToAction("About");
            }
            return View(client);
        }
    }
}
```

Лістинг AdminContoller.cs:

```
using Srore.Models;
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Data.Entity;

using System.Net;
using System.IO;

namespace Srore.Controllers
{
    public class AdminController : Controller
    {
        // GET: Admin
        DBCon db = new DBCon();

        [Authorize(Roles = "admin")]
        public ActionResult Index()
        {
            return View();
        }

        [Authorize(Roles = "admin")]
        public ActionResult List()
        {
            var    clothes    =    db.Clothes.Include(p    =>
p.TypeClothe);
            clothes.ToList();
            clothes = db.Clothes.Include(p => p.Material);
            clothes.ToList();
            clothes = db.Clothes.Include(p => p.Brand);
            clothes.ToList();

            return View(clothes.ToList());
        }


        [HttpGet]
        [Authorize(Roles = "admin")]
        public ActionResult Edit(int? id)
        {
            if (id == null)
            {
                return                                          new
HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Clothe clothe = db.Clothes.Find(id);

            ViewBag.TC    =    new    SelectList(db.TypeClothes,
"TypeClotheID", "Name");
            ViewBag.M    =    new    SelectList(db.Materials,
"MaterialID", "Name");
            ViewBag.B  =  new  SelectList(db.Brands,  "BrandID",
"Name");

            if (clothe == null)
            {
                return HttpNotFound();
            }
            return View(clothe);
        }

        [HttpPost]
        [Authorize(Roles = "admin")]
        [ValidateAntiForgeryToken]
        public      ActionResult      Edit([Bind(Include      =
"ClotheID,Name,TypeClotheID,MaterialID,BrandID,DnTStart,DnTEnd
,Available,Price,Info")] Clothe clothe)
        {
            if (ModelState.IsValid)
            {
                db.Entry(clothe).State = EntityState.Modified;
                db.SaveChanges();
                return RedirectToAction("List");
            }
            return View(clothe);
        }

        [HttpGet]
        [Authorize(Roles = "admin")]
        public ActionResult Delete(int? id)
        {
            if (id == null)
            {
                return                                          new
HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Clothe clothe = db.Clothes.Find(id);
            if (clothe == null)
            {
                return HttpNotFound();
            }
            return View(clothe);
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Delete(int id)
        {
            Clothe clothe = db.Clothes.Find(id);
            db.Clothes.Remove(clothe);
            db.SaveChanges();
            return RedirectToAction("List");
        }

        [HttpGet]
        [Authorize(Roles = "admin")]
        public ActionResult Create()
        {

            ViewBag.TC    =    new    SelectList(db.TypeClothes,
"TypeClotheID", "Name");
            ViewBag.M    =    new    SelectList(db.Materials,
"MaterialID", "Name");
            ViewBag.B  =  new  SelectList(db.Brands,  "BrandID",
"Name");

            //SelectList clothe = new SelectList(db.Clothes);
            //ViewBag.Clothes = clothe;
            return View();
        }

        [HttpPost]
        public ActionResult Create(Clothe clothe)
        {
            clothe.DnTStart                                 =
DateTime.Now.ToString("MM/dd/yyyy HH:mm:ss");
            db.Clothes.Add(clothe);
            db.SaveChanges();
            return RedirectToAction("List");
        }
        //Picture

        [Authorize(Roles = "admin")]
        public ActionResult ListPicture()
        {

            var pic = db.Pictures.Include(p => p.Clothe);
            pic.ToList();
            pic = db.Pictures.Include(p => p.PictureType);
            pic.ToList();
            return View(pic.ToList());
        }

        [Authorize(Roles = "admin")]
        public ActionResult CreatePicture()
        {
            SelectList          clothes          =          new
SelectList(db.Clothes,"ClotheID","Name");
            ViewBag.Clothes = clothes;
            SelectList       picturetypes       =       new
SelectList(db.PictureTypes, "PictureTypeID", "Name");
            ViewBag.Picturetypes = picturetypes;

            return View();
        }

        [HttpPost]
        [Authorize(Roles = "admin")]
        public    ActionResult    CreatePicture(Picture    pic,
HttpPostedFileBase uploadImage)
        {
            if (ModelState.IsValid && uploadImage != null)
            {
                byte[] imageData = null;
                // считываем переданный файл в массив байтов
                using     (var     binaryReader     =     new
BinaryReader(uploadImage.InputStream))
                {
                    imageData                                   =
binaryReader.ReadBytes(uploadImage.ContentLength);
                }
                // установка массива байтов
                pic.Image = imageData;

                db.Pictures.Add(pic);
                db.SaveChanges();

                return RedirectToAction("ListPicture");
            }
            return View(pic);
        }
        [HttpGet]
        [Authorize(Roles = "admin")]
        public ActionResult EditPicture(int? id)
        {
            if (id == null)
            {
                return                                          new
HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Picture picture = db.Pictures.Find(id);
            if (picture == null)
            {
                return HttpNotFound();
            }
            return View(picture);
        }

        [HttpPost]
        [Authorize(Roles = "admin")]
        [ValidateAntiForgeryToken]
        public    ActionResult    EditPicture([Bind(Include    =
"Id,Name,Image,ClotheID")] Picture picture)
        {
            if (ModelState.IsValid)
            {
                db.Entry(picture).State                         =
EntityState.Modified;
                db.SaveChanges();
                return RedirectToAction("ListPicture");
            }
            return View(picture);
        }

        [HttpGet]
        [Authorize(Roles = "admin")]
        public ActionResult DeletePicture(int? id)
        {
            if (id == null)
            {
                return                                          new
HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
```

```
        Picture picture = db.Pictures.Find(id);
            if (picture == null)
            {
                return HttpNotFound();
            }
            return View(picture);
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult DeletePicture(int id)
        {
            Picture picture = db.Pictures.Find(id);
            db.Pictures.Remove(picture);
            db.SaveChanges();
            return RedirectToAction("ListPicture");
        }
        [Authorize(Roles = "admin")]
        public ActionResult ListOrder()
        {

            var ord = db.Orders.Include(p => p.Client); ;
            ord.ToList();
            ord = db.Orders.Include(p => p.TypeBuy);
            ord.ToList();
            return View(ord.ToList());
        }

        [Authorize(Roles = "admin")]
        public ActionResult EditOrder(int? id)
        {
            SelectList typebuy = new SelectList(db.TypeBuys,
"TypeBuyID", "Name");
            ViewBag.TypeBuy = typebuy;


            Order order = db.Orders.Find(id);
            if (order == null)
            {
                return HttpNotFound();
            }
            ViewBag.DnT = order.DnT;
            ViewBag.TotalSum = order.TotalSum;

            return View(order);
        }

        [HttpPost]
        [Authorize(Roles = "admin")]
        [ValidateAntiForgeryToken]
        public ActionResult EditOrder([Bind(Include =
"OrderID,ClientID,DnT,TypeBuyID,Delivery,TotalSum")] Order
order)
        {
            if (ModelState.IsValid)
            {
                db.Entry(order).State = EntityState.Modified;
                db.SaveChanges();
                return RedirectToAction("ListOrder");
            }
            return View(order);
        }

        //TypeClothes
        [Authorize(Roles = "admin")]
        public ActionResult ListTypeClothe()
        {
            var typeclothe = db.TypeClothes;

            return View(typeclothe.ToList());
        }

        [HttpGet]
        [Authorize(Roles = "admin")]
        public ActionResult EditTypeClothe(int? id)
        {
            if (id == null)
            {
                return new
HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            TypeClothe typeclothe = db.TypeClothes.Find(id);
            if (typeclothe == null)
            {
                return HttpNotFound();
            }
            return View(typeclothe);
        }

        [HttpPost]
        [Authorize(Roles = "admin")]
        [ValidateAntiForgeryToken]
        public ActionResult EditTypeClothe([Bind(Include =
"TypeClotheID,Name")] TypeClothe typeclothe)
        {
            if (ModelState.IsValid)
            {
                db.Entry(typeclothe).State =
EntityState.Modified;
                db.SaveChanges();
                return RedirectToAction("ListTypeClothe");
            }
            return View(typeclothe);
        }

        [HttpGet]
        [Authorize(Roles = "admin")]
        public ActionResult DeleteTypeClothe(int? id)
        {
            if (id == null)
            {
                return new
HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            TypeClothe typeclothe = db.TypeClothes.Find(id);
            if (typeclothe == null)
            {
                return HttpNotFound();
            }
            return View(typeclothe);
        }

        [HttpPost]
        [Authorize(Roles = "admin")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteTypeClothe(int id)
        {
            TypeClothe typeclothe = db.TypeClothes.Find(id);
            db.TypeClothes.Remove(typeclothe);
            db.SaveChanges();
            return RedirectToAction("ListTypeClothe");
        }

        [HttpGet]
        [Authorize(Roles = "admin")]
        public ActionResult CreateTypeClothe()
        {
            return View();
        }

        [HttpPost]
        [Authorize(Roles = "admin")]
        public ActionResult CreateTypeClothe(TypeClothe
typeclothe)
        {
            db.TypeClothes.Add(typeclothe);
            db.SaveChanges();
            return RedirectToAction("ListTypeClothe");
        }
        //TypeBuys
        [Authorize(Roles = "admin")]
        public ActionResult ListTypeBuy()
        {
            var typebuy = db.TypeBuys;

            return View(typebuy.ToList());
        }

        [HttpGet]
        [Authorize(Roles = "admin")]
        public ActionResult EditTypeBuy(int? id)
        {
            if (id == null)
            {
                return new
HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            TypeBuy typebuy = db.TypeBuys.Find(id);
            if (typebuy == null)
            {
                return HttpNotFound();
            }
            return View(typebuy);
        }

        [HttpPost]
        [Authorize(Roles = "admin")]
        [ValidateAntiForgeryToken]
        public ActionResult EditTypeBuy([Bind(Include =
"TypeBuyID,Name")] TypeBuy typebuy)
        {
            if (ModelState.IsValid)
            {
                db.Entry(typebuy).State =
EntityState.Modified;
                db.SaveChanges();
                return RedirectToAction("ListTypeBuy");
            }
            return View(typebuy);
        }

        [HttpGet]
        [Authorize(Roles = "admin")]
        public ActionResult DeleteTypeBuy(int? id)
        {
            if (id == null)
            {
                return new
HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            TypeBuy typebuy = db.TypeBuys.Find(id);
            if (typebuy == null)
            {
                return HttpNotFound();
            }
            return View(typebuy);
        }

        [HttpPost]
        [Authorize(Roles = "admin")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteTypeBuy(int id)
        {
            TypeBuy typebuy = db.TypeBuys.Find(id);
            db.TypeBuys.Remove(typebuy);
            db.SaveChanges();
            return RedirectToAction("ListTypeBuy");
        }

        [HttpGet]
        [Authorize(Roles = "admin")]
        public ActionResult CreateTypeBuy()
        {
```

```
            return View();
        }

        [HttpPost]
        [Authorize(Roles = "admin")]
        public ActionResult CreateTypeBuy(TypeBuy typebuy)
        {
            db.TypeBuys.Add(typebuy);
            db.SaveChanges();
            return RedirectToAction("ListTypeBuy");
        }

        //Brand
        [Authorize(Roles = "admin")]
        public ActionResult ListBrand()
        {
            var brand = db.Brands;

            return View(brand.ToList());
        }

        [HttpGet]
        [Authorize(Roles = "admin")]
        public ActionResult EditBrand(int? id)
        {
            if (id == null)
            {
                return                                  new
HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Brand brand = db.Brands.Find(id);
            if (brand == null)
            {
                return HttpNotFound();
            }
            return View(brand);
        }

        [HttpPost]
        [Authorize(Roles = "admin")]
        [ValidateAntiForgeryToken]
        public    ActionResult    EditBrand([Bind(Include    =
"BrandID,Name")] Brand brand)
        {
            if (ModelState.IsValid)
            {
                db.Entry(brand).State = EntityState.Modified;
                db.SaveChanges();
                return RedirectToAction("ListBrand");
            }
            return View(brand);
        }

        [HttpGet]
        [Authorize(Roles = "admin")]
        public ActionResult DeleteBrand(int? id)
        {
            if (id == null)
            {
                return                                  new
HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Brand brand = db.Brands.Find(id);
            if (brand == null)
            {
                return HttpNotFound();
            }
            return View(brand);
        }

        [HttpPost]
        [Authorize(Roles = "admin")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteBrand(int id)
        {
            Brand brand = db.Brands.Find(id);
            db.Brands.Remove(brand);
            db.SaveChanges();
            return RedirectToAction("ListBrand");
        }

        [HttpGet]
        [Authorize(Roles = "admin")]
        public ActionResult CreateBrand()
        {
            return View();
        }

        [HttpPost]
        [Authorize(Roles = "admin")]
        public ActionResult CreateBrand(Brand brand)
        {
            db.Brands.Add(brand);
            db.SaveChanges();
            return RedirectToAction("ListBrand");
        }
    }
}
```

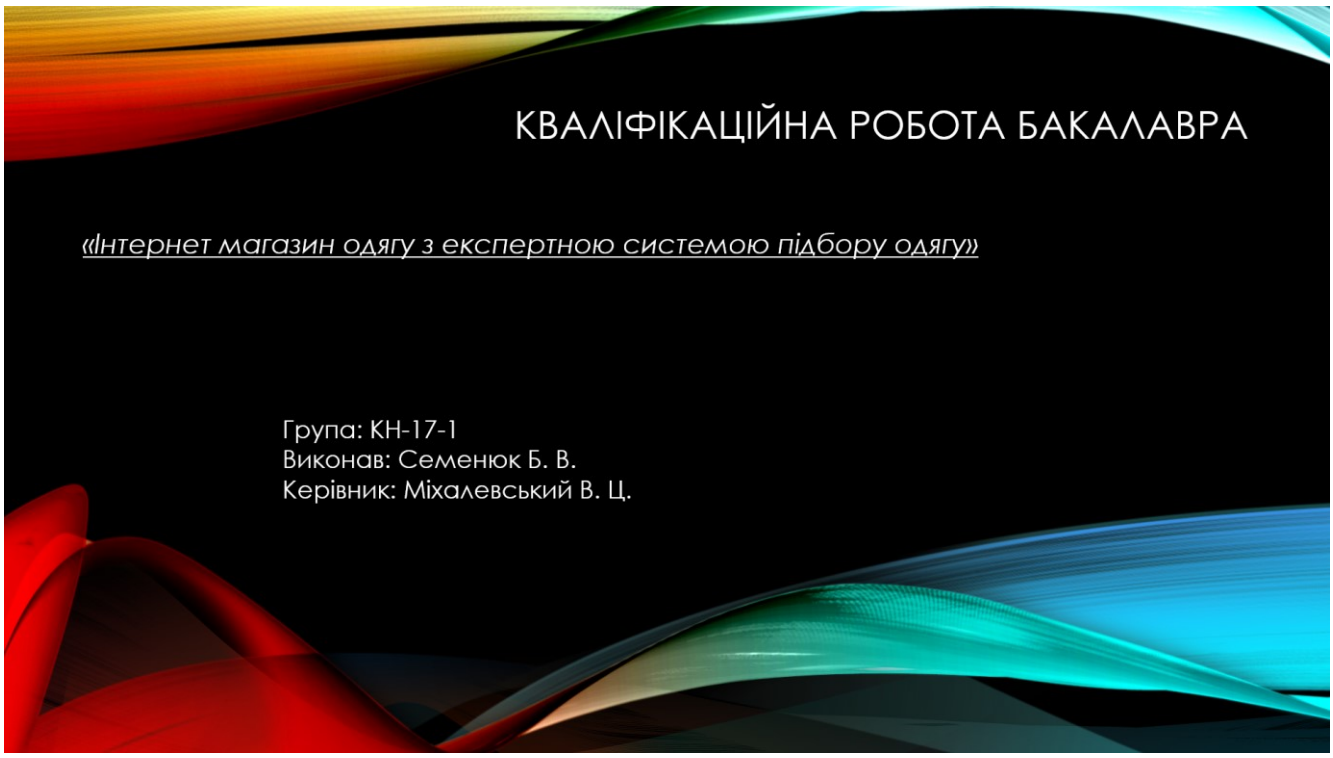
Лістинг ScaseController.cs:

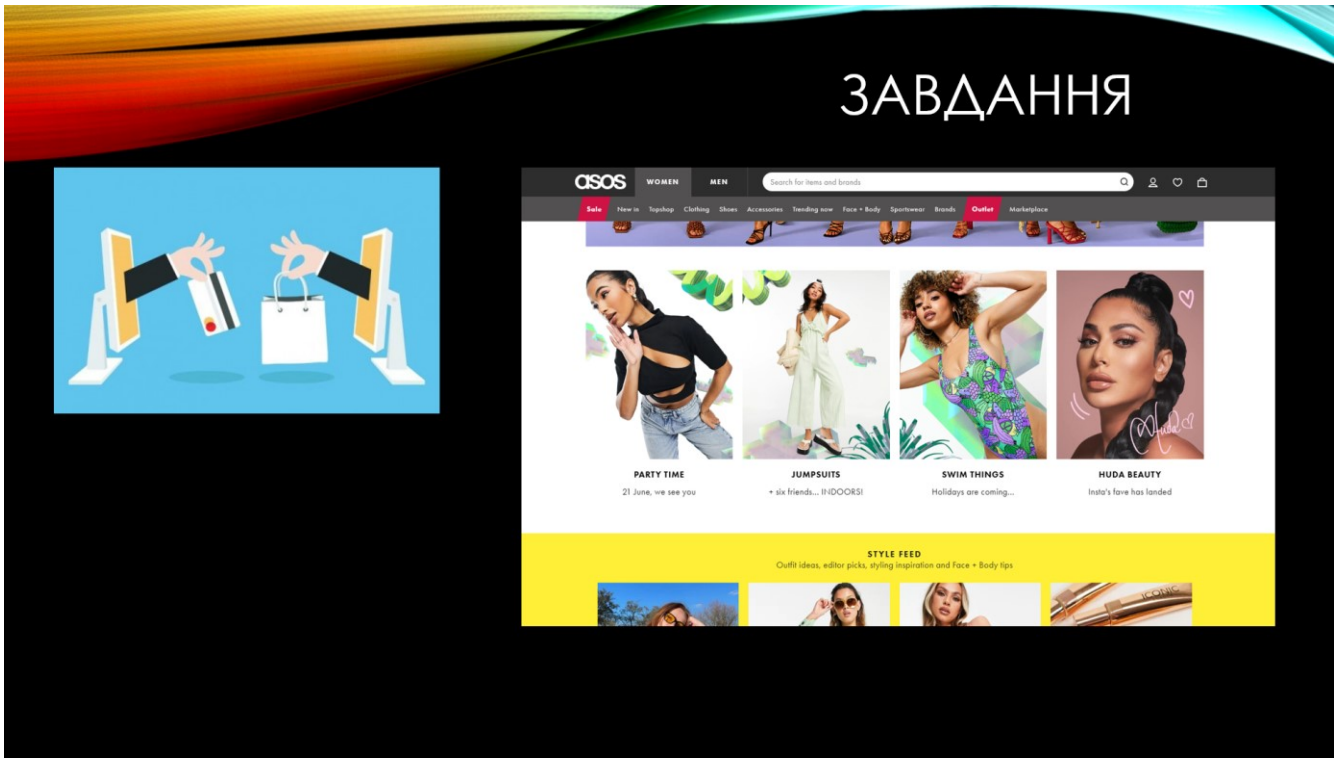
```
using Srore.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Data.Entity;

using System.Net;

namespace Srore.Controllers
{
    public class ScaseController : Controller
    {
        // GET: Scase
        public ActionResult Index()
        {
            return View();
        }
        public ActionResult Main()
        {
            return View();
        }
        DBCon db = new DBCon();
        string picName = "main";


        public ActionResult Filter(string tofilter)
        {
            ViewBag.filter = tofilter;
            var    typemain    =    from    picturetypes    in
db.PictureTypes
                           where picturetypes.Name == picName
                           select picturetypes.PictureTypeID;
            typemain.ToList();

            var    selectedPictures    =    from    pictures    in
db.Pictures
                                   where
pictures.PictureTypeID    ==    (typemain.FirstOrDefault())    &&
pictures.Clothe.TypeClothe.Name == tofilter
                                   select pictures;

            var pic = selectedPictures.Include(p => p.Clothe);
            pic.ToList();
            pic      =      selectedPictures.Include(p      =>
p.Clothe.Brand);
            pic.ToList();
            foreach (Picture ordero in pic)
            {
                ordero.Clothe.Name                              =
ordero.Clothe.Name.Substring(0, 17) + "..."; ;
            }

            return View(pic.ToList());

        }
        public ActionResult List()
        {

            var    typemain    =    from    picturetypes    in
db.PictureTypes
                                   where  picturetypes.Name  ==
picName
                                   select
picturetypes.PictureTypeID;
            typemain.ToList();

            var    selectedPictures    =    from    pictures    in
db.Pictures
                                   where
pictures.PictureTypeID == (typemain.FirstOrDefault())
                                   select pictures;

            var pic = selectedPictures.Include(p => p.Clothe);
            pic.ToList();
            pic      =      selectedPictures.Include(p      =>
p.Clothe.Brand);
            pic.ToList();
            foreach (Picture ordero in pic)
            {
                ordero.Clothe.Name                              =
ordero.Clothe.Name.Substring(0, 17) + "..."; ;
            }

            return View(pic.ToList());

        }

        [Authorize]
        public ActionResult ListBucket()
        {

            //ViewBag.Pictures = from pictures in db.Pictures
            //                   where pictures.Name ==
picName
            //                   select pictures;

            var    orderones    =    db.OrderOnes.Include(p    =>
p.Clothe);
            orderones.ToList();
            orderones = db.OrderOnes.Include(p => p.Size);
            orderones.ToList();

            var selected = from oo in orderones
                           where oo.OrderID == null
                           select oo;


            return View(selected.ToList());
        }

        [HttpGet]
        [Authorize]
        public ActionResult AddBucket(int? id)
        {
            string mail = User.Identity.Name;
            int a = 1;
            if (id == null)
            {
                return                                      new
HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Clothe      clothe      =      db.Clothes.Find(id);
```

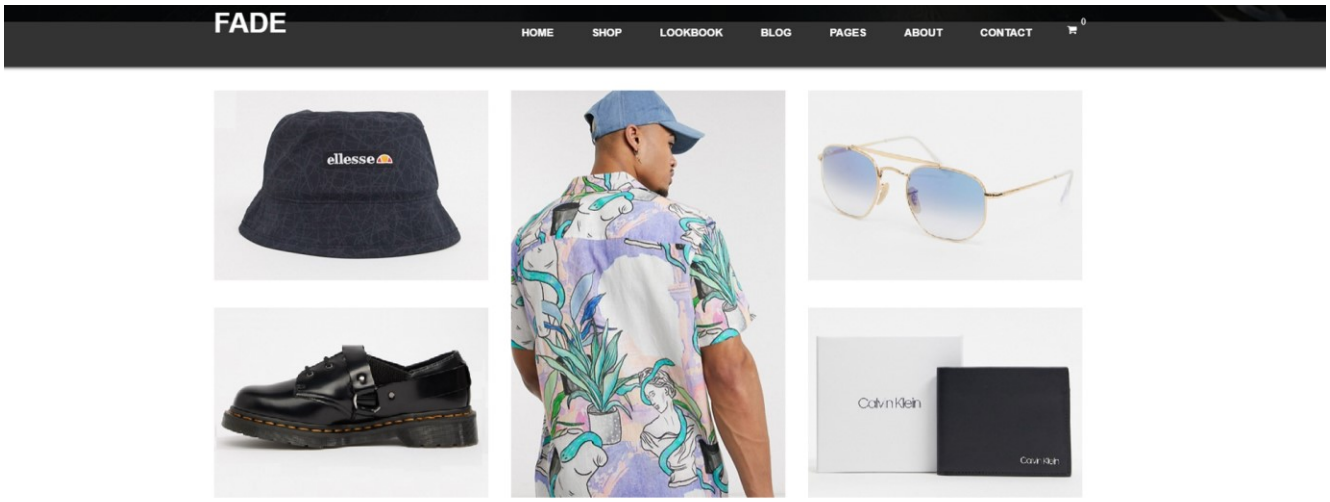
```
            if (clothe == null)
            {
                return HttpNotFound();
            }

            OrderOne orderone = db.OrderOnes.FirstOrDefault(u
=> u.ClientEmail == mail && u.ClotheID == clothe.ClotheID);
            double price = Convert.ToDouble(clothe.Price);
            if (orderone == null)
            {
                db.OrderOnes.Add(new OrderOne { OrderID =
null, ClotheID = clothe.ClotheID, Price = clothe.Price, Amount
= 1, ClientEmail = mail, SizeID = 5 });
                db.SaveChanges();
                //db.OrderOnes.Remove(orderone);
                //db.SaveChanges();
            }
            else if (orderone.SizeID != 5)
            {
                a = orderone.Amount;
                ++a;
                price = a * price;
                db.OrderOnes.Add(new OrderOne { OrderID =
null, ClotheID = clothe.ClotheID, Price =
Convert.ToString(price), Amount = 1, ClientEmail = mail,
SizeID = 5 });
                db.SaveChanges();


            }
            else
            {
                a = orderone.Amount;
                ++a;
                price = a * price;
                db.OrderOnes.Add(new OrderOne { OrderID =
null, ClotheID = clothe.ClotheID, Price =
Convert.ToString(price), Amount = a, ClientEmail = mail,
SizeID = 5 });
                db.SaveChanges();
                db.OrderOnes.Remove(orderone);
                db.SaveChanges();

            }

            return RedirectToAction("List", "Scase");
        }

        [HttpGet]

        public ActionResult CreateBucket(int? id)
        {
            IEnumerable<int> amount = new List<int> { 1, 2, 3,
4, 5 };

            var ty = from clo in db.Clothes
                     where clo.ClotheID == id
                     select clo;
            ty.ToList();

            ViewBag.Cl = new SelectList(ty, "ClotheID",
"Name");


            ViewBag.Am = new SelectList(amount);

            Clothe clothe = db.Clothes.Find(id);
            if (clothe == null)
            {
                return HttpNotFound();
            }

            ViewBag.Clothe = clothe.Name;

            ViewBag.Info = clothe.Info;

            string sp = clothe.Price;
            ViewBag.Price = sp;

            ViewBag.ID = id;


            SelectList size = new SelectList(db.Sizes,
"SizeID", "Name");
            ViewBag.Sizes = size;



            return View();




        }

        [HttpPost]
        [Authorize]
        public ActionResult CreateBucket(OrderOne orderone)
        {

            Clothe clothe =
db.Clothes.Find(orderone.ClotheID);
            if (clothe == null)
            {
                return HttpNotFound();
            }

            orderone.ClientEmail = User.Identity.Name;
            double prisesum = Convert.ToDouble(clothe.Price)*
orderone.Amount;
            orderone.Price =Convert.ToString(prisesum)+",00";
            db.OrderOnes.Add(orderone);
            db.SaveChanges();
            AmountBucket();
            return RedirectToAction("List");
        }
        public ActionResult ListDetails(int? id)
        {


            var selectedPictures = from pictures in
db.Pictures
                                   where pictures.ClotheID ==
id
                                   select pictures;
            //Clothe clothe = db.Clothes.Find(id);
            //string str = clothe.Name;
            //ViewBag.Clothe = str;
            //ViewBag.ID = id;




            return PartialView(selectedPictures.ToList());
        }

        [HttpGet]
        [Authorize]
        public ActionResult Delete(int? id)
        {
            OrderOne orderone = db.OrderOnes.Find(id);
            db.OrderOnes.Remove(orderone);
            db.SaveChanges();
            return RedirectToAction("ListBucket");
        }

        [HttpGet]
        [Authorize]

        public ActionResult EditPM(int? id,int? op)
        {

            string mail = User.Identity.Name;
            int a;
            if (id == null && op == null)
            {
                return                                    new
HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            OrderOne orderone = db.OrderOnes.Find(id);

            if (orderone == null)
            {
                return HttpNotFound();
            }
            else
            {
                double                price                 =
Convert.ToDouble(orderone.Price);
                a = orderone.Amount;
                if(op == 1 && a < 5)
                {
                    price = price / a;
                    ++a;
                    price = a * price;
                    IEnumerable<OrderOne>     orderones     =
db.OrderOnes
                       .Where(c => c.OrderOneID == id)
                       .AsEnumerable()
                       .Select(c => {
                           c.Price = Convert.ToString(price) +
",00";
                           c.Amount = a;
                           return c;
                       });
                    foreach (OrderOne ordero in orderones) {
db.Entry(ordero).State = EntityState.Modified; }
                    db.SaveChanges();
                }

                else if (op == 0 && a != 1)
                {
                    price = price / a;
                    --a;
                    price = a * price;
                    IEnumerable<OrderOne>     orderones     =
db.OrderOnes
                       .Where(c => c.OrderOneID == id)
                       .AsEnumerable()
                       .Select(c => {
                           c.Price = Convert.ToString(price) +
",00";
                           c.Amount = a;
                           return c;
                       });
                    foreach (OrderOne ordero in orderones) {
db.Entry(ordero).State = EntityState.Modified; }
                    db.SaveChanges();
                }
            }
            return RedirectToAction("ListBucket", "Scase");
        }
        [Authorize]
        public ActionResult EditPMS(int? id, int? op)
        {
            int a;
            if (id == null && op == null){ return new
HttpStatusCodeResult(HttpStatusCode.BadRequest); }

            OrderOne orderone = db.OrderOnes.Find(id);
            if (orderone == null){ return HttpNotFound(); }
            else
            {
                a = orderone.SizeID;
                if (op == 1 && a < 8)
```

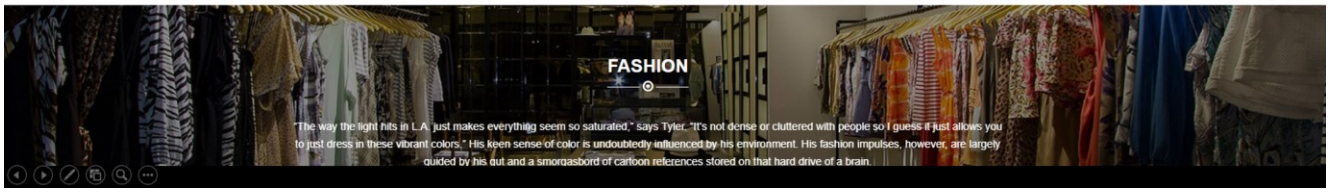
```
                {
                    ++a;
                    IEnumerable<OrderOne>      orderones      =
db.OrderOnes
                        .Where(c => c.OrderOneID == id)
                        .AsEnumerable()
                        .Select(c => {
                            c.SizeID = a;
                            return c;
                        });
                    foreach      (OrderOne      ordero      in
orderones){db.Entry(ordero).State = EntityState.Modified;}
                    db.SaveChanges();
                }
                else if (op == 0 && a != 1)
                {
                    --a;
                    IEnumerable<OrderOne>      orderones      =
db.OrderOnes
                         .Where(c => c.OrderOneID == id)
                         .AsEnumerable()
                         .Select(c => {
                             c.SizeID = a;
                             return c;
                         });
                    foreach (OrderOne ordero in orderones) {
db.Entry(ordero).State = EntityState.Modified; }
                    db.SaveChanges();
                }
            }
            return RedirectToAction("ListBucket", "Scase");
        }

        [Authorize]
        public ActionResult CreateOrder()
        {
            string mail = User.Identity.Name;
            var selected = from clo in db.Clients
                     where clo.Email == mail
                     select clo;
            selected.ToList();

            SelectList    cl    =    new    SelectList(selected,
"ClientID", "Name");

            ViewBag.Client = cl;


            ViewBag.TypeBuy   =   new   SelectList(db.TypeBuys,
"TypeBuyID", "Name");


            var soro = from o in db.OrderOnes
                       where o.ClientEmail == mail
                       select o.Clothe.Name;
            soro.ToList();

            ViewBag.Clothes = new SelectList(soro);


            IEnumerable<OrderOne>  orderones  =  from  oo  in
db.OrderOnes                                    where
oo.ClientEmail == mail && oo.OrderID == null
                                                select oo;

            double total = 0;
            foreach (OrderOne ordero in orderones)
            {
                total += Convert.ToDouble(ordero.Price);

            }

            ViewBag.TotalSum  = Convert.ToString(total+",00");


            return View();
        }

        [HttpPost]
        [Authorize]
        public ActionResult CreateOrder(Order order)
        {

            string mail = User.Identity.Name;

            IEnumerable<OrderOne> orderones = db.OrderOnes
                .Where(c => c.ClientEmail == mail)
                .AsEnumerable()
                .Select(c => {
                    c.OrderID = order.OrderID;
                    return c;
                });
            double total = 0;
            foreach (OrderOne ordero in orderones)
            {
                total += Convert.ToDouble(ordero.Price);
                db.Entry(ordero).State = EntityState.Modified;
            }



            order.DnT    =    DateTime.Now.ToString("MM/dd/yyyy
HH:mm:ss");
            order.TotalSum = Convert.ToString(total + ",00");
            db.Orders.Add(order);
            db.SaveChanges();

            return RedirectToAction("List");

        }

        [Authorize]
        public ActionResult ListOrderDetails()
        {

            string mail = User.Identity.Name;
            var    selectedPictures    =    from    pictures    in
db.OrderOnes
                                    where   pictures.ClientEmail
== mail && pictures.OrderID == null
                                    select pictures;

            var pic = selectedPictures.Include(p => p.Clothe);
            pic.ToList();

            pic = selectedPictures.Include(p => p.Size);
            //var    clothes    =    db.Clothes.Include(p    =>
p.TypeClothe);
            //clothes.ToList();
            //clothes = db.Clothes.Include(p => p.Material);
            //clothes.ToList();
            //clothes = db.Clothes.Include(p => p.Brand);
            //clothes.ToList();


            return PartialView(pic.ToList());
        }

        [HttpPost]

        public ActionResult Search(string strsearch)
        {
            ViewBag.search = strsearch;
            var    typemain    =    from    picturetypes    in
db.PictureTypes
                           where picturetypes.Name == picName
                           select picturetypes.PictureTypeID;
            typemain.ToList();

            var    selectedPictures    =    from    pictures    in
db.Pictures
                                    where
pictures.PictureTypeID == (typemain.FirstOrDefault())
                                    select pictures;

            selectedPictures.ToList();

            selectedPictures.Include(p => p.Clothe);

            var    searchPictures    =    from    spictures    in
selectedPictures
                                  where
spictures.Clothe.Name.Contains(strsearch)
                                  select spictures;

            var pic = searchPictures.Include(p => p.Clothe);
            pic.ToList();


            foreach (Picture ordero in pic)
            {
                ordero.Clothe.Name                                =
ordero.Clothe.Name.Substring(0, 17) + "..."; ;
            }
            pic = pic.Include(p => p.Clothe.Brand);
            return View(pic.ToList());
        }

        [HttpPost]
        [Authorize]
        public ActionResult AmountBucket()
        {
            int res = 0;
            string mail = User.Identity.Name;

            IEnumerable<OrderOne>  orderones  =  from  oo  in
db.OrderOnes
                                               where
oo.ClientEmail == mail
                                               select oo;

            IEnumerable<OrderOne>   select   =   from   os   in
orderones
                                          where os.OrderID ==
null
                                          select os;

            foreach (OrderOne ordero in select)
            {
                res += ordero.Amount;

            }

            return View(ViewBag.Amount = res);
        }
    }
}
```

ПРЕЗЕНТАЦІЯ



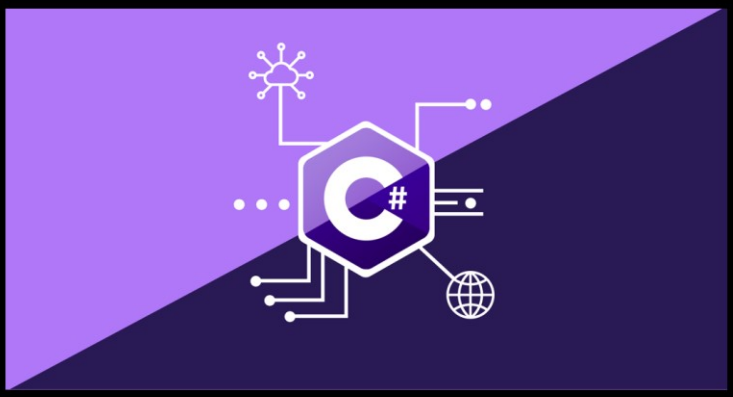


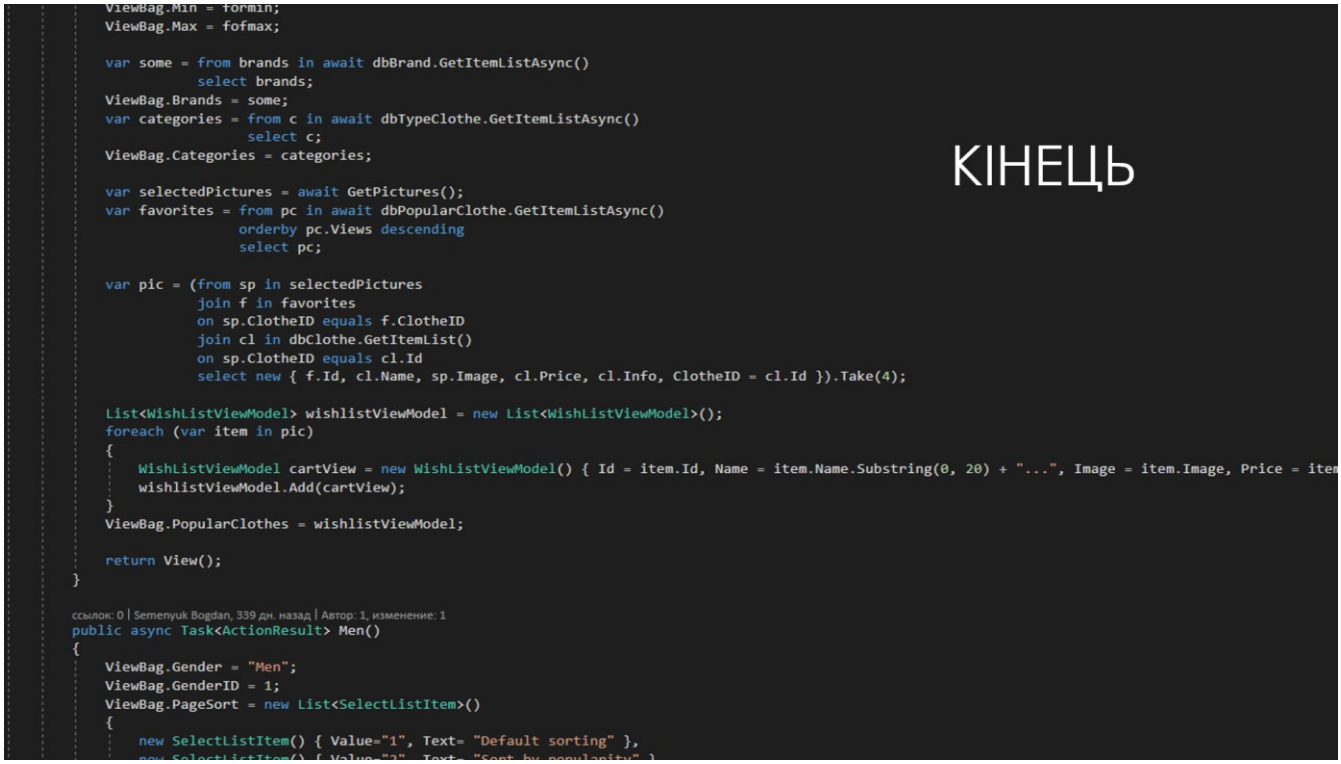




ТЕХНОЛОГІЇ

Framework: ASP.NET MVC 5
Language: C# 7
ORM: Entity Framework 6
Server: MS SQL 2016
Front: html, css, js





```
        ViewBag.Min = formin;
        ViewBag.Max = fofmax;

        var some = from brands in await dbBrand.GetItemListAsync()
                   select brands;
        ViewBag.Brands = some;
        var categories = from c in await dbTypeClothe.GetItemListAsync()
                         select c;
        ViewBag.Categories = categories;

        var selectedPictures = await GetPictures();
        var favorites = from pc in await dbPopularClothe.GetItemListAsync()
                        orderby pc.Views descending
                        select pc;

        var pic = (from sp in selectedPictures
                   join f in favorites
                   on sp.ClotheID equals f.ClotheID
                   join cl in dbClothe.GetItemList()
                   on sp.ClotheID equals cl.Id
                   select new { f.Id, cl.Name, sp.Image, cl.Price, cl.Info, ClotheID = cl.Id }).Take(4);

        List<WishListViewModel> wishlistViewModel = new List<WishListViewModel>();
        foreach (var item in pic)
        {
            WishListViewModel cartView = new WishListViewModel() { Id = item.Id, Name = item.Name.Substring(0, 20) + "...", Image = item.Image, Price = item
            wishlistViewModel.Add(cartView);
        }
        ViewBag.PopularClothes = wishlistViewModel;

        return View();
    }

    ссылок: 0 | Semenyuk Bogdan, 339 дн. назад | Автор: 1, изменение: 1
    public async Task<ActionResult> Men()
    {
        ViewBag.Gender = "Men";
        ViewBag.GenderID = 1;
        ViewBag.PageSort = new List<SelectListItem>()
        {
            new SelectListItem() { Value="1", Text= "Default sorting" },
```

КІНЕЦЬ

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ГОЛОВІ ЕКЗАМЕНАЦІЙНОЇ КОМІСІЇ

Направляється студент Семенюк Б. В. на захист дипломного проекту (роботи)
(прізвище, ініціали)

за спеціальністю 122 - Комп'ютерні науки

На тему: Інтернет-магазин одягу з експертною системою підбору товару

Дипломний проект (робота), рецензія і довідка про перевірку на плагіат додаються.

Декан факультету _____ САВЕНКО О.С.
(підпис) (прізвище та ініціали)

ДОВІДКА УСПІШНОСТІ

Семенюк Б. В. за період навчання на факультеті програмування та компютерних і телекомунікаційних систем з 2017 по 2021 роки. повністю виконав навчальний план спеціальності з такими розподілом оцінок за:
національною шкалою: відмінно 62,50 %, добре 25,00 %, задовільно 12,50 %.
шкалою ЄКТС: A 54,55 %, B 12,73 %, C 18,18 %, D 7,27 %, E 7,27 %.

Методист факультету _____ _____
(підпис) (прізвище та ініціали)

ВИСНОВОК КЕРІВНИКА ДИПЛОМНОГО ПРОЕКТУ (РОБОТИ) ТА ОБГРУНТУВАННЯ ОЦІНКИ

Студент Семенюк Б.В. обсяг робіт по виконанню кваліфікаційної роботи бакалавра виконав повністю. Тема роботи актуальна, розроблена модель та алгоритми оптимізації обміну товарів, інформаційна система відповідає сучасним вимогам, КРБ виконана згідно методичних вимог. Зауваження, що виникали на попередньому захисті, не виявлено.

Оцінка дипломного проекту (роботи) відмінно

Керівник дипломного прокту (роботи) _____ Міхалевич В.Ю.
(підпис) (прізвище та ініціали)

"8" 06 2021 р.

ВИСНОВОК КАФЕДРИ ПРО ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ)

Дипломний проект (роботу) розглянуто. Студент Семенюк Б. В. допускається до захисту цього

Завідувач кафедри КНІТ
(назва)

_____ Бармак О.В. "9" 06 2021 р.
(підпис, прізвище, ініціали)

Tue Jun 08 11:38:34 EEST 2021, Петровський Сергій Степанович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальное совпадение с одним документом 18.0%

Словари проверки: en_US, ru_RU, ua_UA. **Ошибок в документах: 13%**

	Документ		Суммарное совпадение по Базе Данных	
ID: 92668 Название: Інтернет-магазин одягу з експертною системою підбору одягу Добавлено в БД: 2021-06-08 Авторы: Б.В. Семенюк Руководители: В.Ц. Міхалевський Консультанты: Опоненты:	Символы	Лексемы	Символы	Лексемы
	44684	398	15134 (34%)	149 (37%)

Источник плагиата

ID	Описание	Наличие плагиата в документе	
		Символы	Лексемы
90055	Название: ЗВІТ з професійної практики Добавлено в БД: 2021-05-10 Авторы: Семенюк Б. В. Руководители: Скрипник Т.К. Консультанты: Опоненты:	8048 (18.0%)	75 (19.0%)

UNICHECK

Ім'я користувача:
Кафедра КН

ID перевірки:
1008243663

Дата перевірки:
09.06.2021 12:26:28 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
09.06.2021 12:28:22 EEST

ID користувача:
100005671

Назва документа: **Bakalavr_Semenyuk_v_08_06_21 02 Lite**

Кількість сторінок: **57** Кількість слів: **6884** Кількість символів: **50115** Розмір файлу: **11.94 MB** ID файлу: **1008316197**

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

4.81%
Схожість

Найбільша схожість: **2.28%** з джерелом з Бібліотеки (ID файлу: **1008302248**)

3.43% Джерела з Інтернету	117	Сторінка 59
3.24% Джерела з Бібліотеки	44	Сторінка 59

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Підозріле форматування	20 сторінок

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНИХ НАУК ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: *Інтернет-магазин одягу з експертною системою підбору товару*

Автор: Семенюк Б.В., студент групи КН-17-1

Спеціальність: 122 – Комп'ютерні науки

Освітня програма: освітньо-професійна

Науковий керівник: к.фіз.-мат.н., доцент кафедри КНІТ Міхалевський В.Ц.

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	-
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	-
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	-

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) запозичення розміщені в розділах аналізу існуючих аналогів та прототипів, які не описують безпосередньо авторське дослідження і не стосуються результатів роботи;
2) усі запозичення фрагментарні;
3) до запозичень входять фрагменти програмного коду, що на мають авторства і містять поширені конструкції;
4) серед запозичень знаходяться загальновідомі терміни, скорочення та визначення.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості, складає 4,1% і адресується до 19 першоджерел, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ В. Ц. Міхалевський

Гарант ОП _____ О. В. Мазурець

Завідувач кафедри КНІТ _____ О. В. Бармак

Хмельницький національний університет

Кафедра комп'ютерних наук та інформаційних технологій

РЕЦЕНЗІЯ

на кваліфікаційну роботу бакалавра

Студента: *групи КН-17-1 Семенюка Богдана Васильовича*

За темою: *Інтернет-магазину одягу з експертою ситемою підбору товару*

1. Актуальність і значення теми: *розробрений web-додаток дозволяє спростити та автоматизувати продаж одягу у магазині та полегшити процес підбору розміру при покупці через інтернет.*

2. Оцінка запропонованих моделей, підходів, алгоритмів, інформаційної складової та засобів розробки: *web-застосунок продажу товару, згідно усіх поставлених бізнес-процесів, та аглоритм підбору одягу, працють вірно та витримали перевірку на стресостійкісь.*

3. Оцінка розробленої інформаційної системи, її практична цінність та економічна доцільність: *розроблена «Інтернет-магазин одягу з експертною системою підбору товару» застосовується для автоматизації процесу продажу одягу через інтернет.*

4. Загальний висновок та оцінка: *вимоги поставленої задачі виконані в повному обсязі, інтернет-магазим одягу з екпертна система підбору розміру, працює вірно.*

Робота заслуговує на оцінку «*відмінно*»

Рецензент *к.ф.-м.н., доц. Ярецька Н.О.* 08.06.2024