

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

«Сайт для замовлення піци»

Виконав студент групи ІПЗ-19-3
ЛИННИК Максим Русланович

Керівник роботи:
КРАВЧЕНКО Світлана Миколаївна

Рецензент:
ЄФРЕМОВ Юрій Миколайович

Житомир – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

В.о. зав. кафедри інженерії
програмного забезпечення
Андрій Морозов
«02» серпня 2023 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ЛИННИК Максим Русланович**

Керівник роботи: **КРАВЧЕНКО Світлана Миколаївна**

Тема роботи: **«Сайт для замовлення піци»**,

затверджена Наказом закладу вищої освіти від **«02» серпня 2023 р. №326с**

Вихідні дані для роботи: розробка та проектування сайту для замовлення піци.

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Кравченко С.М.	02.08.2023р.	02.08.2023р.
2	Кравченко С.М.	02.08.2023р.	02.08.2023р.
3	Кравченко С.М.	02.08.2023р.	02.08.2023р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускової кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Аналізування вимог до програмного продукту	2 серпня 2023 – 5 серпня 2023	Виконано
2	Пошук, огляд та аналіз аналогічних розробок	6 серпня 2023 – 7 серпня 2023	Виконано
3	Формулювання технічного завдання та опрацювання літературних джерел	7 серпня 2023 – 9 серпня 2023	Виконано
4	Проектування структури програми	10 серпня 2023 – 12 серпня 2023	Виконано
5	Написання програмного коду	13 серпня 2023 – 23 серпня 2023	Виконано
6	Оформлення пояснювальної записки	23 серпня 2023 – 28 серпня 2023	Виконано
7	Попередній захист	25 серпня 2023	Виконано

РЕФЕРАТ

Випускна робота бакалавра складається з програмного додатку та пояснювальної записки. Пояснювальна записка до випускної роботи містить 46 сторінок, 21 ілюстрацій та 14 таблиці.

Метою роботи є розробка сайту для замовлення піци, з конструкторами для шаблонів піц, кошиків та замовлень, модулем авторизації користувачів.

В роботі визначено основні завдання на розробку сайту, проаналізовано аналоги розробленого сайту, обґрунтовано вибір фреймворків та методів розробки.

Наведено сценарії роботи програмного додатку, структура та опис бази даних. Алгоритм роботи основних функцій, реалізованих за допомогою мови C#.

КЛЮЧОВІ СЛОВА: БД, MVC, КОНТРОЛЕР, ПРЕДСТАВЛЕННЯ, ЗАПИТ.

					ІПЗ.КР.Б – 121 – 23 – ПЗ			
Зм.	Арк.	№ докум.	Підпис	Дата	Сайт для замовлення піци Пояснювальна записка	Літ.	Арк.	Аркушів
Розроб.	М. Линник						4	
Керівник	С. Кравченко					Житомирська політехніка, група ІПЗ-19-3		
Н.Контр.	Ю Єфремов							
Зав. каф.	А. Морозов							

ABSTRACT

The bachelor's thesis consists of a software application and an explanatory note. The explanatory note to the final thesis contains 46 pages, 21 illustrations and 14 tables.

The purpose of the work is to develop a website for ordering pizza, with designers for pizza templates, baskets and orders, a user authorization module.

The work defines the main tasks for the development of the site, analyzes the analogues of the developed site, substantiates the choice of frameworks and development methods.

The scenarios of the software application, the structure and description of the database are presented. The algorithm of the main functions implemented using the C# language.

KEY WORDS: DB, MVC, CONTROLLER, REPRESENTATION, REQUEST.

					ИЗ.КР.Б – 121 – 23 – ИЗ	
						5

ЗМІСТ

ВСТУП	8
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ ПРОГРАМНОГО ДОДАКТУ	10
1.1. Постановка задачі	10
1.2. Аналіз аналогів програмного продукту	11
1.3. Вибір архітектури веб-сайту	14
1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення	15
Висновки до розділу 1	17
РОЗДІЛ 2. ПРОЕКТУВАННЯ ПРОГРАМНОГО ДОДАТКУ	19
2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи	19
2.2. Розробка бази даних системи	21
2.3. Проектування та реалізація алгоритмів роботи системи	26
2.4 Реалізація програмного додатку для замовлення піци	27
Висновки до розділу 2	33
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ПРОГРАМНИМ ДОДАТКОМ	34
3.1. Порядок встановлення та налаштування системи	34
3.2. Структура інтерфейсу. Інтерфейс та порядок роботи з програмним додатком	34
Висновки до розділу 3	42
ВИСНОВКИ	43
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	45
ДОДАТКИ	47

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

MVC – Model View Controller (з *англ.* Модель Уявлення Контролер)

БД – База даних

СКБД – Система керування базами даних

ВСТУП

Актуальність теми. Сучасне життя людини вкрай насичене різного роду стресами, перепонами та нескінченним потоком речей, які повинні бути виконані вчасно. Кожен день іти на роботу, з роботи, робити домашні справи та навіть займатися волонтерською діяльністю – все це не залишає і хвилини часу на перекус. Адже їжа - основа життєдіяльності людини. Без якісного обіду та вечері ніяка складна робота чи перешкода у житті не буде подолана. Більш того, для нашого здоров'я необхідно правильно і якісно харчуватися, а, як відомо, від здоров'я залежить все в нашому житті. Що ж, як не найсмачніша піца – може задовольнити таку необхідну потребу людини. Не лише потреба в їжі, а й у позитивних емоціях, адже ніщо так не бадьорить і тішить – як трапезування піци у родинному колі.

Втім, як і було правильно помічено - у сучасної людини немає жодної хвилини вільного часу. Тому потреба у зручному способі отримання піци - те, що потрібно всьому людству. А піца, їжа та позитивні емоції в одній особі, - найбільш затребуваний продукт для доставки. Саме тому створення сайту для замовлення піци – актуальне завдання.

Метою випускної роботи є розроблення програмного додатку для замовлення піци.

Встановлена мета обумовлює наступні завдання:

- конструктор шаблонів піц під індивідуальні потреби людини;
- створення індивідуальних кошиків з додатковими продуктами;
- замовлення піци на конкретну дату та час;

Об'єктом дослідження є технології створення зручного сайту для замовлення піци.

Предметом дослідження є використання веб технологій для створення сайту, з можливістю створення замовлень.

В роботі було використано методи веб-програмування та роботи з БД.

Розроблена система може бути впроваджена у будь-які веб-сайти для замовлення піц.

Пояснювальна записка до випускної роботи містить 46 сторінок, 21 ілюстрацій та 14 таблиць, список літературних джерел містить 12 найменування.

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ ПРОГРАМНОГО ДОДАКТУ

1.1. Постановка задачі

Основна задача додатку полягає в створенні зручного сайту для створення замовлень. Перш за все, необхідно дати можливість користувачу створювати замовлення на конкретну дату та час. Далі потрібна можливість створення кошиків з різними додатковими до піци продуктами, а також функцію розрахування ціни. В кінці потрібно надати функціонал редагування інгредієнтів піци для людей з алергією на ті чи інші продукти. Всі ці індивідуально створені речі повинні мати змогу бути використаними повторно.

Для реалізації наявного завдання необхідно створити базу даних і конструктори для шаблонів піц, кошиків та замовлень, а також модуль реєстрації та авторизації. Опис основних етапів:

1. Проектування та розробка БД.
2. Розробка модуля користувачів з можливостями реєстрації, авторизації та наявності адміна.
3. Розробка конструктора піц з можливостями: виключення із піци деяких продуктів або подвоєння їх порції, вибір розмірів та частити від піци, розрахування ціни відповідно до заданих параметрів, збереження шаблону для подальшого використання.
4. Розробка конструктора кошика з можливостями: додавання до кошика наявних додаткових продуктів, вибір шаблону піци користувача, розрахування ціни відповідно до заданих параметрів, збереження шаблону для подальшого використання.
5. Розробка конструктора замовлень з можливостями: вказання дати та часу, адреси та кошика користувача.

Результатом виконання поставленого завдання є програмний додаток для створення замовлень на основі індивідуально створених кошиків та шаблонів піц.

1.2. Аналіз аналогів програмного продукту

Замовлення їжі - це дуже популярна штука, особливо у світлі епідемії, коли виходити надвір, м'яко кажучи, небажано. Втім, у звичайний час це спосіб заощадити купу часу на похід у ресторан, піцерію, або в магазин за продуктами, щоб самому приготувати їжу. На ці зайві рухи немає часу у сучасної людини, а тому тут виявляються дуже доречні сервіси для замовлення їжі додому, зокрема піци. Саме тому в кожній більш-менш пристойній піцерії є свій сайт.

Опишемо деякі з них.

Pizza Nata – сайт з не самим приємним дизайном, без можливості редагування піц, крім розмірів.

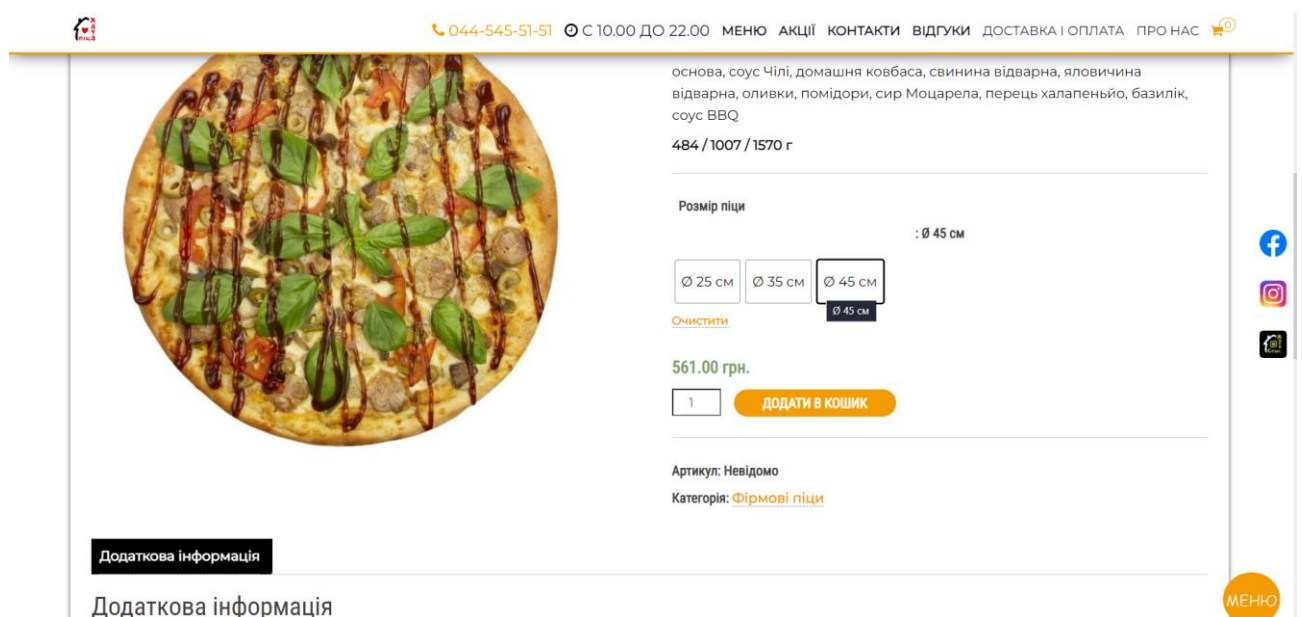


Рис.1.1 Сайт Pizza Nata

Mono pizza – зручний мінімалістичний сайт, без можливості редагувати піци.

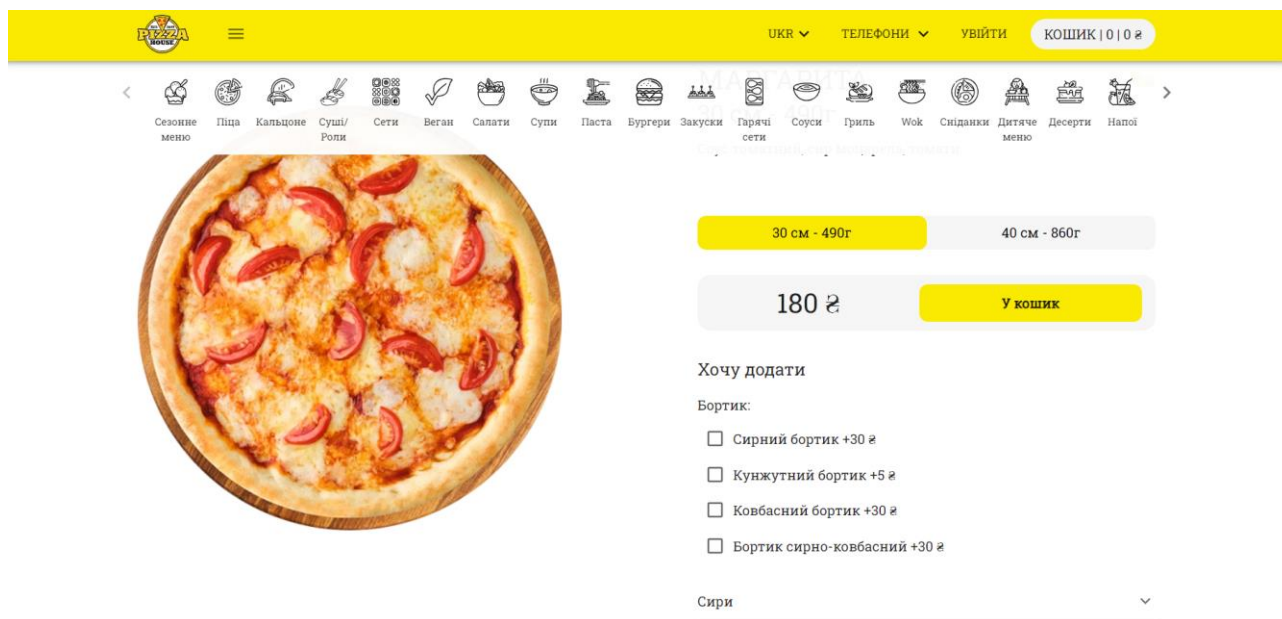


Рис.1.4 Сайт Pizza house

Pizza 33 – сайт з надмірним дизайном та з можливістю редагування піци.

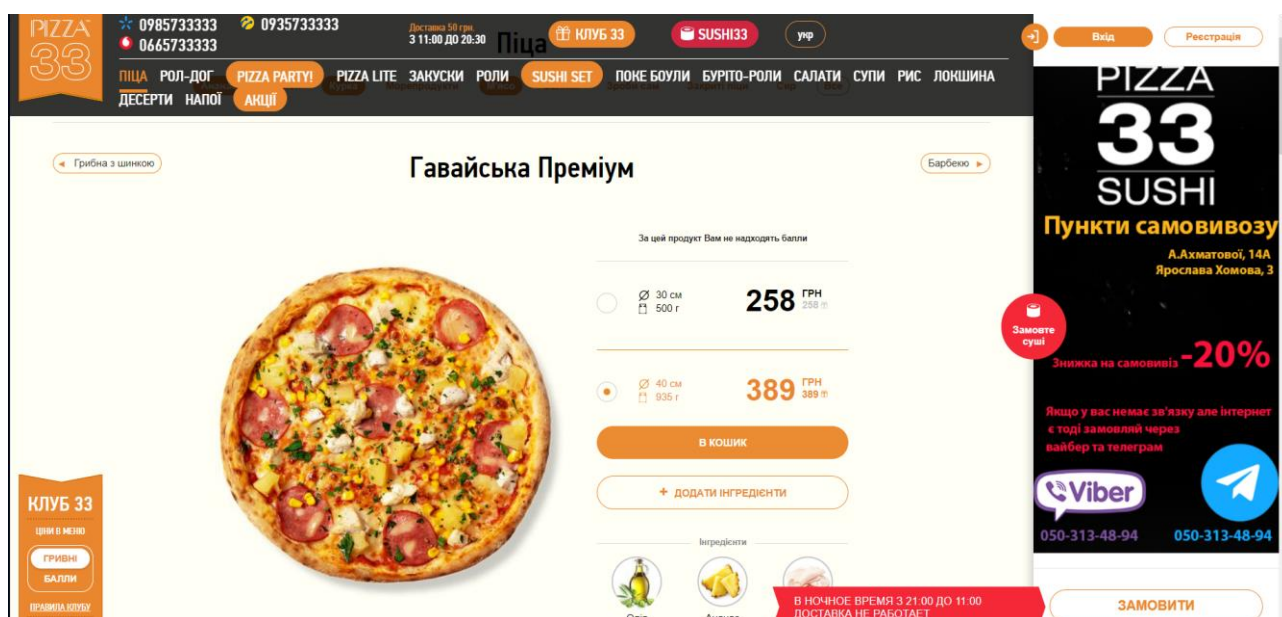


Рис.1.5 Сайт Pizza 33

Таблиця 1.1.

Можливості аналогів

Сайт	Можливість редагування піци	Реєстрація	Збереження кошика
Pizza Hata	-	+	-
Mono pizza	-	+	-
Tarantino family	-	+	-
Pizza house	+	+	-

Pizza 33	-	+	-
----------	---	---	---

Після аналізу аналогів програмного забезпечення для замовлення піци можна зробити висновки, що всі програми мають можливість зареєструватися, замовити піцу, інколи вибрати розмір, додати її до кошику та зробити замовлення по деякій адресі..

Особливостями сервісів є зосередження уваги на конструкторі піци та кошика, зручний розрахунок ціни з урахуванням змін у шаблонах, модуль реєстрації.

1.3. Вибір архітектури веб-сайту

Для реалізації сайту для замовлення піци було використано патерн MVC.

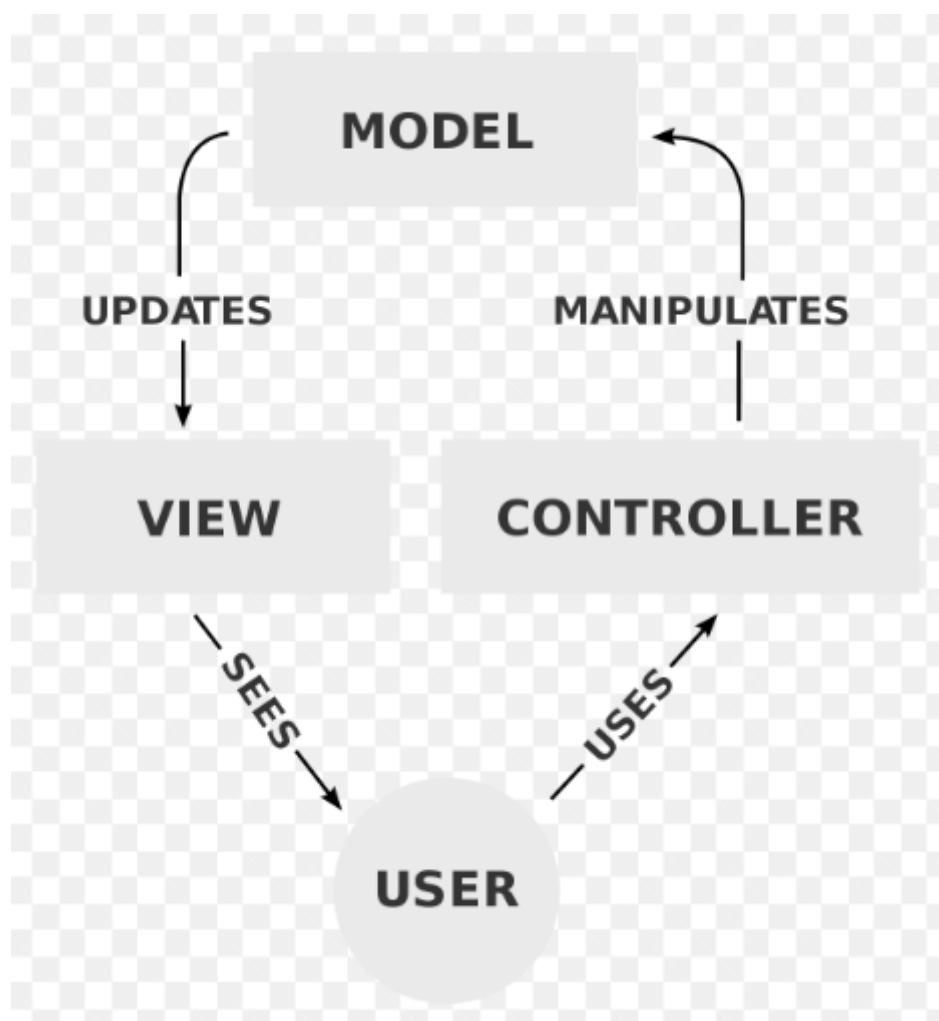


Рис.1.6. Архітектура MVC

MVC — схема поділу даних та логіки на три окремі компоненти: модель, подання та контролер — розробка та модифікація яких може здійснюватися незалежно.

1. Модель

Модель працює з даними та методами роботи з ними: запити до бази даних та валідація. Вона не повинна працювати з візуалізацією даних або взаємодіяти з користувачем, просто повинна надавати доступ до даних та управління ними. Модель може мати кілька різних уявлень, які використовуватимуть її методи.

2. Представлення

Представлення відповідає за отримання необхідних даних із моделі та не обробляє введені дані користувача. Дизайн, форми та інше візуальне взаємодія з користувачем знаходиться тут.

3. Контролер

Контролер забезпечує зв'язок між користувачем та системою, контролюючи та спрямовуючи дані від користувача до системи та навпаки.

В підсумку, можна сказати, що даний архітектурний шаблон чудово підходить для розробки сайту для замовлення піци, особливо для зручної роботи з логікою БД та представленням даних користувачу.

1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Для програмного продукту були використані: мова програмування C#, мова програмування JavaScript та бібліотека jQuery, мова програмування SQL, мови програмування HTML та CSS, фреймворк Entity Framework, фреймворк ASP.NET MVC Framework, інтегроване середовище розробки Microsoft Visual Studio, для роботи з даними був використаний SQL Server Management Studio.

C# – об'єктно-орієнтована мова програмування загального призначення. C# відноситься до мов з C-подібним синтаксисом, їх синтаксис найбільш близький до C++ і Java. Мова має статичну типізацію, підтримує поліморфізм, навантаження операторів (у тому числі операторів явного та неявного приведення типу), делегати, атрибути, події, змінні, властивості, узагальнені

типи та методи, ітератори, анонімні функції з підтримкою замикань, LINQ, винятки, коментарі у форматі XML. бібліотек.

JavaScript – мультипарадигменна мова програмування. Підтримує об'єктно-орієнтований, імперативний та функціональний стилі. Основні архітектурні риси: динамічна типізація, слабка типізація, автоматичне керування пам'яттю, прототипне програмування, функції як об'єкти першого класу.

jQuery – набір функцій JavaScript, що фокусується на взаємодії JavaScript та HTML. Бібліотека jQuery допомагає легко отримувати доступ до будь-якого елемента DOM, звертатися до атрибутів та вмісту елементів DOM, маніпулювати ними. Також бібліотека jQuery надає зручний API для роботи з AJAX.

SQL – декларативна мова програмування, що використовується для створення, модифікації та управління даними в реляційній базі даних, що керується відповідною системою управління базами даних. Є насамперед інформаційно-логічною мовою, призначеною для опису, зміни та вилучення даних, що зберігаються в реляційних базах даних.

HTML — стандартизована мова гіпертекстової розмітки документів для перегляду веб-сторінок у браузері. Мова надає засоби для створення заголовків, абзаців, списків, посилань, цитат та інших елементів. У HTML можна вбудувати програмний код мовою програмування JavaScript для керування поведінкою та змістом веб-сторінок. Також включення CSS в HTML дозволяє задавати зовнішній вигляд та макет сторінки.

CSS — формальна мова декорування та опису зовнішнього вигляду веб-сторінки, написаного за допомогою мови розмітки HTML. Мова використовується творцями веб-сторінок для завдання кольорів, шрифтів, стилів, розташування окремих блоків та інших аспектів представлення цих веб-сторінок.

Entity Framework – об'єктно-орієнтована технологія доступу до даних. Фреймворк дозволяє працювати з даними у формі специфічних для домену об'єктів та властивостей (наприклад, з клієнтами та їх адресами) без

необхідності враховувати формат базових таблиць та стовпців бази даних, де зберігаються ці дані.

ASP.NET MVC Framework — фреймворк для створення веб-додатків, що базується на взаємодії трьох компонентів: контролера, моделі та представлення. Контролер приймає запити, обробляє введення користувача, взаємодіє з моделлю і представленням, і повертає користувачеві результат обробки запиту. Модель представляє шар, що описує логіку організації даних у додатку. Представлення отримує дані з контролера і генерує елементи інтерфейсу користувача для відображення інформації.

Microsoft Visual Studio - лінійка продуктів компанії Microsoft, що включають інтегроване середовище розробки (IDE) програмного забезпечення та низку інших інструментів. Дані продукти дозволяють розробляти як консольні програми, так і ігри та програми з графічним інтерфейсом, у тому числі з підтримкою технології Windows Forms, UWP а також веб-сайти, веб-програми, веб-служби. Visual Studio включає редактор вихідного коду з підтримкою технології IntelliSense і можливістю найпростішого рефакторингу коду. Вбудований налагоджувач може працювати як відладчик рівня вихідного коду, так і відладчик машинного рівня. Інші вбудовані інструменти включають редактор форм для спрощення створення графічного інтерфейсу програми, веб-редактор, дизайнер класів і дизайнер схеми бази даних. Visual Studio дозволяє створювати та підключати сторонні доповнення (плагіни) для розширення функціональності практично на кожному рівні.

SQL Server Management Studio — утиліта для Microsoft SQL Server 2005 та пізніших версій для конфігурування, керування та адміністрування всіх компонентів Microsoft SQL Server. Утиліта включає скриптовий редактор та графічну програму, яка працює з об'єктами та налаштуваннями сервера. Microsoft SQL Server - система управління реляційними базами даних, розроблена корпорацією Microsoft. Основна мова запитів - Transact-SQL.

Висновки до розділу 1

Проведений аналіз предметної області визначив основні особливості та необхідний функціонал для програмного продукту, а саме:

					ІІЗ.КР.Б – 121 – 23 – ІІЗ	
						17

- реєстрація та авторизація користувачів;
- конструктори піц, кошика та замовлення;
- збереження створених шаблонів для подальшого використання;

Аналіз аналогів показав, що основними функціями є підбір індивідуальних параметрів для замовлення різної глибини опрацювання.

Архітектура для розробки програмного додатку:

- Мова програмування C#;
- Мова програмування JavaScript та бібліотека jQuery;
- Мови програмування HTML та CSS;
- Фреймворк Entity Framework;
- Фреймворк ASP.NET MVC Framework;

А також інтегроване середовище розробки Microsoft Visual Studio та СУБД SQL Server Management Studio.

.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ПРОГРАМНОГО ДОДАТКУ

2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи

Програмний додаток для зручного замовлення індивідуальних шаблонів піц та кошиків.

Вимоги користувачів

Незареєстровані користувачі мають доступ до наступних функцій:

1. Зареєструватися або авторизуватися.
2. Зайти на сторінку каталогу без можливості створення шаблону.

Зареєстровані користувачі мають доступ до наступних функцій:

1. Вийти з системи.
2. Зайти на сторінку каталогу та створити шаблон піци.
3. Створити кошик.
4. Створити замовлення.
5. Видалити власні шаблони піц, кошики та замовлення.
6. При наявності повноважень адміна – переглядати та видаляти шаблони піц, кошики та замовлення інших користувачів.

Функціональні вимоги:

1. Авторизація та реєстрація користувачів у системі.
2. Перегляд каталогу з подальшим створенням індивідуального шаблону піци, розрахунок його ціни та збереження в системі.
3. Доступ до створення індивідуального кошику, розрахунок ціни та збереження його в системі.
4. Можливість створення замовлення на конкретну дату та час.
5. Видалення створених кошиків, шаблонів піц та замовлень.

Нефункціональні вимоги:

1. Навчання роботи з системою:

- час, потрібний для навчання з додатком не більше 10 хвилин;
- час відповіді системи для звичайних запитів не повинен перевищувати 1 секунд, а для більш складних запитів – 3 секунди;
- інтерфейс представлення повинен бути інтуїтивно зрозумілим та мінімалістичним для користувача та не вимагати від нього додаткової підготовки;
- середній час безперервної роботи – 30 робочих днів;
- максимальна норма помилок та дефектів в роботі додатку – 1 помилок на 2000 запитів.

2. Продуктивність

Система повинна підтримувати 200 користувачів одночасно працюючих із запитами.

3. Можливість експлуатації:

- система повинна мати можливість збільшувати потужності;
- оновлення версій повинно здійснюватися примусово і в автоматичному режимі.

Для реалізації програмного продукту використовувалась мова C# та Фреймворк ASP.NET MVC Framework.

У програмному продукту наявні класи кожен з яких відповідає таблиці з БД, ось деякі з них:

- User – це клас користувача, де зберігаються дані про нього, а також за допомогою унікального номера присвоюються шаблони піц, кошики та замовлення. Клас має такі методи: GetUserData – отримання даних користувача по номеру, GetAuthUserId – отримання номера авторизованого користувача, GetAuthUserName – отримання імені авторизованого користувача, GetUserNames – отримання даних усіх користувачів.
- Sample_pizza – це клас шаблонів піц, за допомогою якого зберігається унікальний номер та реалізоване збереження параметрів замовлення.

Клас має такі методи: GetIdSample – вибір номера останнього шаблону, GetSamples – отримання усіх шаблонів користувача, GetSamplesPriceById – отримання ціни шаблону за номером, GetIdSamplePizza – отримання номеру піци по якій був створений шаблон.

- Baket – це клас кошиків для збереження даних та зв'язків з параметрами індивідуального кошика. Клас має такі методи: GetAllBaket – отримання усіх кошиків користувача по номеру, GetBaketPrice – отримання ціни кошика по номеру, GetIdBaket – отримання номеру останнього кошику.
- Order – це клас замовлень для збереження даних. Клас має метод: GetAllOrder – отримання усіх замовлень користувача по номеру.

Таким чином програмний додаток має весь потрібний функціонал для взаємодії з таблицями БД, тільки видалення було реалізовано окремо.

2.2. Розробка бази даних системи

База даних складається із 11 таблиць та має назву Pizza.

Таблиця 2.1

Інформація про таблиці бази даних Pizza

Назва таблиці	Призначення
Addition	Додаткові товари кошика
Baket	Кошик
Baket_add	Зв'язок кошиків та додаткових товарів
Buffer	Зв'язок кошиків та шаблонів піц
Ingredient	Інгредієнти піц
Order	Замовлення
Pizza	Піци
Pizza_ingr	Зв'язок піц та інгредієнтів
Sample_ingr	Зв'язок шаблонів піц та інгредієнтів
Sample_pizza	Шаблони піц
User	Користувач

Структура бази даних наведена на рис. 2.3.

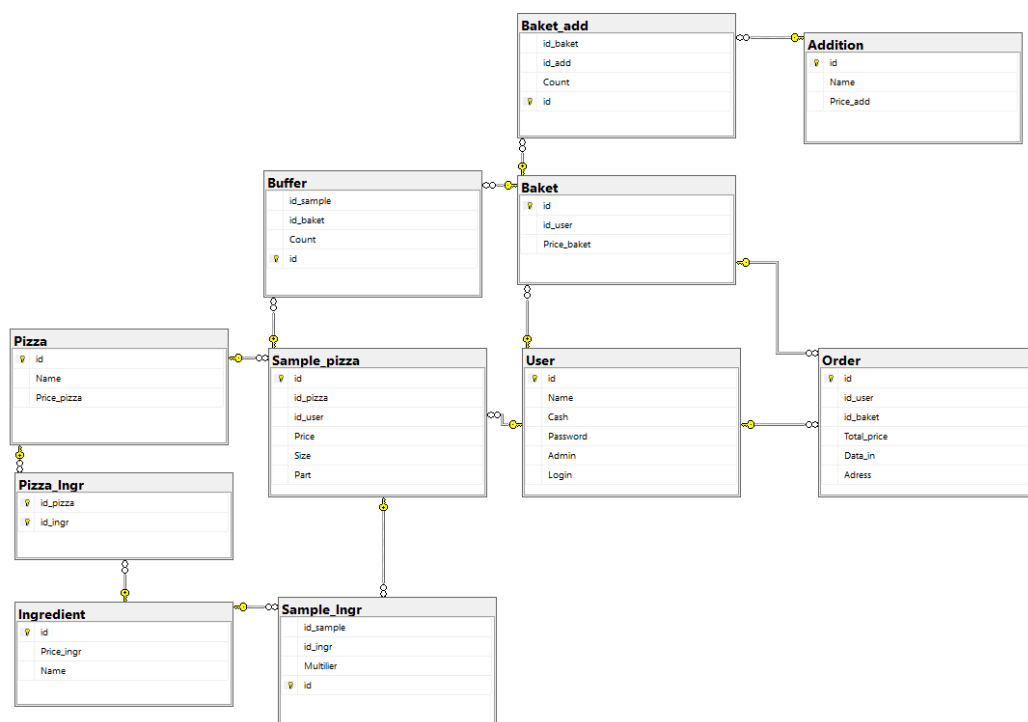


Рис.2.1. Структурна схема БД

Далі наведемо опис основних таблиць бази даних.

Таблиця «Addition» призначена для вибору додаткових товарів до кошику. Структура таблиці наведена нижче:

Таблиця 2.2

Опис полів таблиці «Addition»

Назва	Тип даних	ПК	Опис поля
id	Int	+	Номер
Name	varchar(50)	-	Назва
Price_add	int	-	Ціна

Таблиця «Baket» призначена для вибору кошика. Структура таблиці наведена нижче:

Таблиця 2.3

Опис полів таблиці «Baket»

Назва	Тип даних	ПК	Опис поля
id	int	+	Номер
id_user	int	-	Номер користувача
Price_baket	float	-	Ціна кошика

Таблиця «Baket_add» призначена для зв'язку багато до багатьох таблиць «Baket» та «Addition». Структура таблиці наведена нижче:

Таблиця 2.4

Опис полів таблиці «Baket_add»

Назва	Тип даних	ПК	Опис поля
id_baket	int	-	Номер кошика
id_add	int	-	Номер додаткового товару
Count	int	-	Кількість товару
id	int	+	Номер

Таблиця «Buffer» призначена для зв'язку багато до багатьох таблиць «Baket» та «Sample_pizza». Структура таблиці наведена нижче:

Таблиця 2.5

Опис полів таблиці «Buffer»

Назва	Тип даних	ПК	Опис поля
id_sample	int	-	Номер шаблону піци
id_baket	int	-	Номер кошику
Count	int	-	Кількість піц
id	int	+	Номер

Таблиця «Ingredient» призначена для вибору інгредієнтів піц. Структура таблиці наведена нижче:

Таблиця 2.6

Опис полів таблиці «Ingredient»

Назва	Тип даних	ПК	Опис поля
id	int	+	Номер
Price_ingr	int	-	Ціна
Name	varchar(50)	-	Назва

Таблиця «Order» призначена для вибору замовлень. Структура таблиці наведена нижче:

Таблиця 2.7

Опис полів таблиці «Order»

Назва	Тип даних	ПК	Опис поля
id	int	+	Номер
id_user	int	-	Номер користувача
id_baket	int	-	Номер кошика
Total_price	float	-	Ціна
Data_in	datetime	-	Дата на коли замовлення
Adress	varchar(50)	-	Адреса

Таблиця «Pizza» призначена для вибору піци. Структура таблиці наведена нижче:

Таблиця 2.8

Опис полів таблиці «Pizza»

Назва	Тип даних	ПК	Опис поля
id	int	+	Номер
Name	varchar(50)	-	Назва
Price_pizza	int	-	Ціна

Таблиця «Pizza_ingr» призначена для зв'язку багато до багатьох таблиць «Pizza» та «Ingredient». Структура таблиці наведена нижче:

Таблиця 2.9

Опис полів таблиці «Pizza_ingr»

Назва	Тип даних	ПК	Опис поля
id_pizza	int	+	Номер піци
id_ingr	int	+	Номер інгредієнта

Таблиця «Sample_ingr» призначена для зв'язку багато до багатьох таблиць «Sample_pizza» та «Ingredient». Структура таблиці наведена нижче:

Таблиця 2.10

Опис полів таблиці «Sample_ingr»

Назва	Тип даних	ПК	Опис поля
-------	-----------	----	-----------

id_sample	int	-	Номер кошика
id_ingr	int	-	Номер інгредієнта
Multilier	int	-	Кількість інгредієнта
id	int	+	Номер

Таблиця «Sample_pizza» призначена для шаблонів піц. Структура таблиці наведена нижче:

Таблиця 2.11

Опис полів таблиці «Sample_pizza»

Назва	Тип даних	ПК	Опис поля
id	int	+	Номер
id_pizza	int	-	Номер піци
id_user	int	-	Номер користувача
Price	float	-	Ціна
Size	int	-	Розмір
Part	int	-	Частина

Таблиця «User» призначена для користувачів. Структура таблиці наведена нижче:

Таблиця 2.12

Опис полів таблиці «User»

Назва	Тип даних	ПК	Опис поля
id	int	+	Номер
Name	varchar(50)	-	Ім'я
Cash	int	-	Гроші
Password	int	-	Пароль
Admin	int	-	Адмін
Login	varchar(50)	-	Імейл

Отже реалізована база даних у відповідності до вимог реляційної моделі забезпечує збереження та доступ до даних для сайту замовлення піц. База даних складається з 11 таблиць. Основними є Baket, Addition, Ingredient, Order, Pizza, Sample_pizza, User. Зв'язками багато до багатьох є Baket_add, Buffer, Pizza_ingr, Sample_ingr.

2.3. Проектування та реалізація алгоритмів роботи системи

Основними модулями додатку є конструктори і модуль роботи з користувачем та його даними.

Конструкторів є 3:

1. Конструктор шаблонів піц.

Для того, щоб створити шаблон піци користувач повинен авторизуватись, вибрати на сторінці каталогу піцу та натиснути кнопку «Шаблон». Після чого відкриється сторінка піци та потрібно буде вибрати індивідуальні параметри для піци, а також розмір та частину. Можливо вирахувати ціну шаблону за допомогою відповідної кнопки. Після натиснення кнопки «Зберегти шаблон піци» - шаблон буде збережено.

2. Конструктор кошиків.

Для того, щоб створити кошик – потрібно зайти на сторінку кошику та вибрати індивідуальні параметри, а саме додаткові товари та кількість піц. Також потрібно вибрати шаблон піци, тобто він повинен уже бути створений. Після чого можливо вирахувати ціну за кошик натисненням відповідної кнопки. Після натиснення кнопки «Зберегти кошик» - кошик буде збережено.

3. Конструктор замовлень.

Для того, щоб створити замовлення – потрібно зайти на сторінку замовлення та вибрати кошик(він повинен бути створений), ввести адресу та вибрати необхідну дату з часом. Після натиснення кнопки «Створити замовлення» - замовлення буде збережено.

Модуль роботи з користувачем та його даними – це насамперед авторизація, реєстрація та видалення створених шаблонів піц, кошиків та замовлень. Для реєстрації потрібно зайти на відповідну сторінку та ввести ім'я, пароль та ім'я. Для авторизації достатньо ввести пароль та ім'я, при умові, що користувач вже зареєстрований. Для видалення шаблонів піц, кошиків та замовлень потрібно зайти на сторінку профіля та вибрати номер об'єкту, що потрібно видалити, та натиснути на кнопку «Видалити». Якщо

користувач має повноваження адміна – йому доступна відповідна сторінка та можливість видалення замовлень, шаблонів піц та кошиків інших користувачів.

Таким чином було розглянуто основні алгоритми сайту для замовлення піци.

2.4 Реалізація програмного додатку для замовлення піци

Для роботи програмного додатку було написано модель DB яка керує запитам до БД, а для видалення було використано DbContext. Нижче буде наведений лістинг коду деяких важливих функцій.

Підключення до БД в моделі DB:

```
public static IDbConnection Connection
{
    get
    {
        return new SqlConnection("Server=DESKTOP-
23FRT8U\\SQLEXPRESS;Database=Pizza;Trusted_connection=True;TrustServerCertificate =
True;MultipleActiveResultSets=true");
    }
}
```

Запит до БД для отримання інгредієнтів піци:

```
public static List<Ingredient> GetIngr(int id)
{
    var buf_id = Connection.Query<Pizza_Ingr>($"SELECT [id_ingr] FROM
[Pizza_Ingr] WHERE [id_pizza] =('{id}')").ToList();
    List<Ingredient> result = new List<Ingredient>();
    for (int i = 0; i < buf_id.Count; i++)
    {
        result.Add(Connection.Query<Ingredient>($"SELECT * FROM [Ingredient]
WHERE [id] =('{buf_id[i].id_ingr}')").ToList()[0]);
    }
    return result;
}
```

Реєстрація користувача:

```
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Register(RegisterViewModel model)
{
    if (ModelState.IsValid)
    {
        User user = await db.User.FirstOrDefaultAsync(u => u.Login ==
model.Login);
        if (user == null)
        {
            db.User.Add(new User { Login = model.Login, Password =
model.Password, Admin = 0, Cash = model.Cash, Name = model.Name });
            await db.SaveChangesAsync();
        }
    }
}
```

```

        await Authenticate(model.Login);

        return RedirectToAction("Index", "Home");
    }
    else
        ModelState.AddModelError("", "Некорректные логин и(или) пароль");
    }
    return RedirectToAction("Index", "Home");
}

```

Секція скриптів на сторінці з піцою:

```

@section scripts
{
    <script type="text/javascript">
        $("#submit").click(function () {
            let ids = [ @jsonString ]
            let arr = []
            for (let i = 0; i < @ingr.Count; i++) {
                arr.push($("##{ids[0][i]}").val())
            }
            var size = $("#Size").val()
            var part = $("#Part").val()
            var id = @item.id;
            var b = JSON.stringify(arr)
            var date = { size: size, part: part, b: b, id: id }
            $.post("/Pizza/Summa", date, function (data) {
                $("#dialog").html(data);
                $("#modal").modal('show');
            })
        })

    </script>
    <script>
        $("#save").click(function () {
            let ids = [ @jsonString ]
            let arr = []
            for (let i = 0; i < @ingr.Count; i++) {
                arr.push($("##{ids[0][i]}").val())
            }
            var size = $("#Size").val()
            var part = $("#Part").val()
            var id = @item.id;
            var b = JSON.stringify(arr)
            var date = { size: size, part: part, b: b, id: id }
            $.post("/Pizza/Save", date)
        })
    </script>
}

```

Розрахунок ціни шаблону піци:

```

[HttpPost]
public ActionResult Summa(string size, string part, string b, string id)
{
    var pizza = GetPizza(int.Parse(id));
    double total_price = pizza[0].Price_pizza;
    var ssize = int.Parse(size);
    if(ssize == 1)
    {
        total_price = total_price / 2;
    }
    else if(ssize == 3)
    {
        total_price = total_price * 2;
    }
}

```

```

    }
    var ppart = int.Parse(part);
    if(ppart == 3)
    {
        total_price = total_price * 0.75;
    }
    else if(ppart == 2)
    {
        total_price = total_price * 0.5;
    }
    else if (ppart == 1)
    {
        total_price = total_price * 0.25;
    }
    var ingr = GetIngr(int.Parse(id));
    List<int> ingrn = new List<int>();
    Regex regex = new Regex(@"\d*");
    MatchCollection matches = regex.Matches(b);
    if (matches.Count > 0)
    {
        foreach (Match match in matches)
            if (!(match.Value == "")) { ingrn.Add(int.Parse(match.Value)); }
    }

    for (int i = 0; i < ingrn.Count; i++)
    {
        total_price = total_price - ingr[i].Price_ingr;
        var j = ingr[i].Price_ingr * ingrn[i];
        total_price = total_price + j;
    }
    return PartialView(total_price);
}

```

Дизайн сторінки з каталогом:

```

<h2>Каталог нашої смачної піци</h2>
@using System.Text.RegularExpressions;

@{var i = 0;}
<section class="text-center mb-4">
    @{var item = @Model;}
    @{for (i = 0; i < @Model.Count; i++)
    {
        <div class="row wow fadeIn">
            <div class="col-lg-3 col-md-6 mb-4">
                
                Номер: @item[i].id
                Назва: @item[i].Name
                Ціна: @item[i].Price_pizza
                @{ if (User.Identity.IsAuthenticated)
                    {<a asp-controller="Pizza" asp-action="Pizza" asp-route-
value="@item[i].id">Шаблон</a>}
                    else
                    {<strong>Для замовлення зареєструйтесь</strong>}
                }
            }
        </div>
        @{i++;}
        @{if (i == Model.Count) { break; } }
        <div class="col-lg-3 col-md-6 mb-4">
            
            Номер: @item[i].id
            Назва: @item[i].Name
            Ціна: @item[i].Price_pizza
            @{ if (User.Identity.IsAuthenticated)

```

```

        {<a asp-controller="Pizza" asp-action="Pizza" asp-route-
value="@item[i].id">Шаблон</a>}
        else
        {<strong>Для замовлення зареєструйтесь</strong>}
    }
</div>
@{i++;}
@if (i == Model.Count) { break; }
<div class="col-lg-3 col-md-6 mb-4">
    
    Номер: @item[i].id
    Назва: @item[i].Name
    Ціна: @item[i].Price_pizza
    @{ if (User.Identity.IsAuthenticated)
        {<a asp-controller="Pizza" asp-action="Pizza" asp-route-
value="@item[i].id">Шаблон</a>}
        else
        {<strong>Для замовлення зареєструйтесь</strong>}
    }
</div>
@{i++;}
@if (i == Model.Count) { break; }
<div class="col-lg-3 col-md-6 mb-4">
    
    Номер: @item[i].id
    Назва: @item[i].Name
    Ціна: @item[i].Price_pizza
    @{ if (User.Identity.IsAuthenticated)
        {<a asp-controller="Pizza" asp-action="Pizza" asp-route-
value="@item[i].id">Шаблон</a>}
        else
        {<strong>Для замовлення зареєструйтесь</strong>}
    }
</div>
</div>;
}
</section>

```

Підключення контекстів та функції авторизації з реєстрацією:

```

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

    public IConfiguration Configuration { get; }

    // This method gets called by the runtime. Use this method to add services to the
    container.
    public void ConfigureServices(IServiceCollection services)
    {
        string connection = Configuration.GetConnectionString("DefaultConnection");
        services.AddDbContext<UserContext>(options => options.UseSqlServer(connection));
        services.AddDbContext<Sample_pizzaContext>(options =>
options.UseSqlServer(connection));
        services.AddDbContext<Sample_IngrContext>(options =>
options.UseSqlServer(connection));
        services.AddDbContext<BaketContext>(options =>
options.UseSqlServer(connection));
        services.AddDbContext<BufferContext>(options =>
options.UseSqlServer(connection));
    }
}

```

```

        services.AddDbContext<Baket_addContext>(options =>
options.UseSqlServer(connection));
        services.AddDbContext<OrderContext>(options =>
options.UseSqlServer(connection));
        services.AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme)
            .AddCookie(options =>
            {
                options.LoginPath = new
Microsoft.AspNetCore.Http.PathString("/User/Authorization");
            });

        services.AddControllersWithViews();
    }

    // This method gets called by the runtime. Use this method to configure the HTTP
    request pipeline.
    public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
    {
        if (env.IsDevelopment())
        {
            app.UseDeveloperExceptionPage();
        }
        else
        {
            app.UseExceptionHandler("/Home/Error");
            // The default HSTS value is 30 days. You may want to change this for
            production scenarios, see https://aka.ms/aspnetcore-hsts.
            app.UseHsts();
        }
        app.UseHttpsRedirection();
        app.UseStaticFiles();

        app.UseRouting();

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

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

```

В структурі проекту наявна спеціалізована папка `img` для зберігання картинок піц. Це було виконано для зручності використання цих файлів на сайті або будь-де.

Структура наведена нижче:

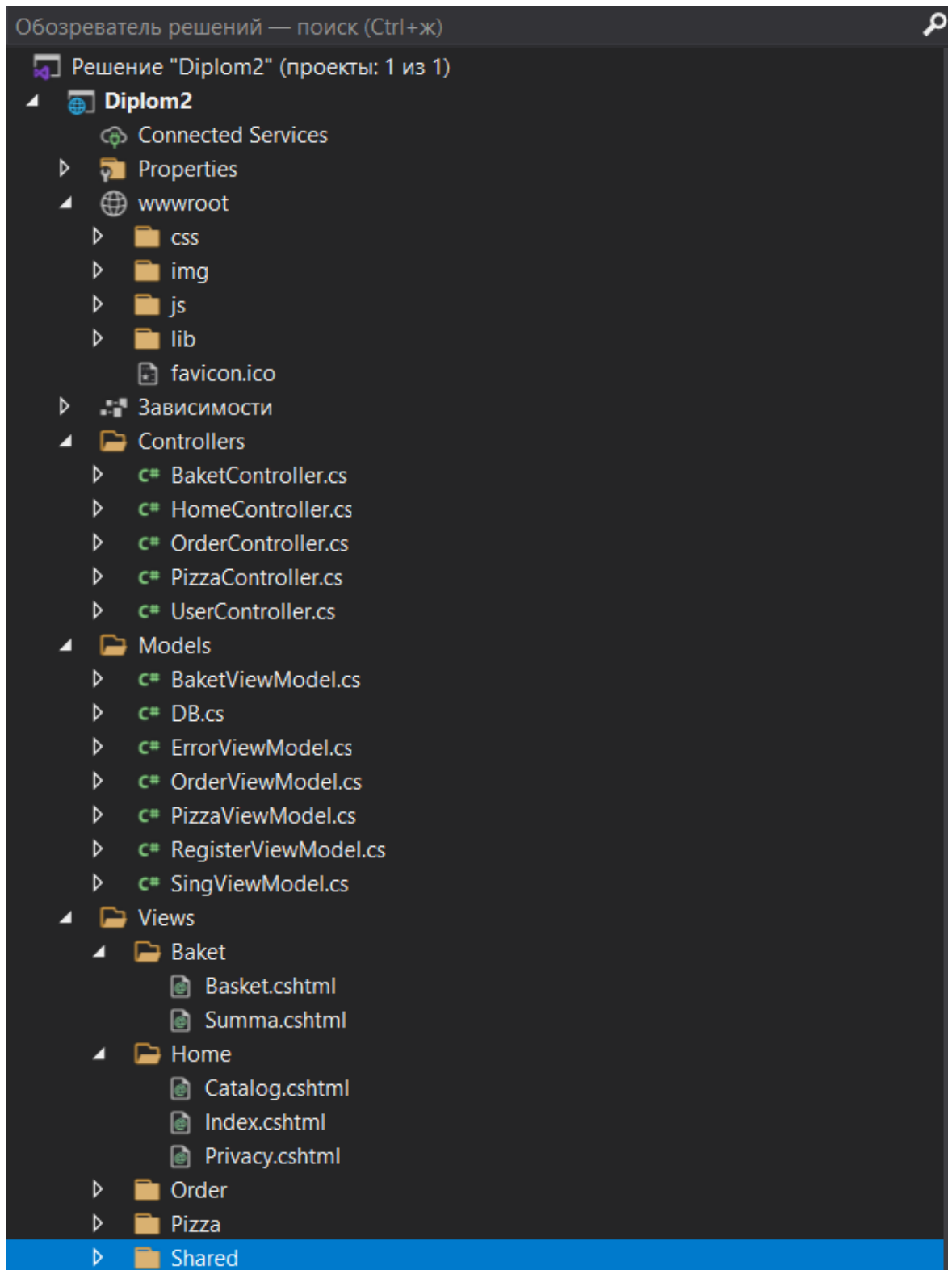


Рис.2.2 Структура проекту

Було розглянуто основні складні функції та методи програмного додатку.

Висновки до розділу 2

Програмний додаток для замовлення піци надає можливості підбору індивідуальних параметрів піц та кошиків, та їх збереження для подальшого використання користувачем

БД складається з 11 таблиць. Основними є Basket, Addition, Ingredient, Order, Pizza, Sample_pizza, User. Зв'язками багато до багатьох є Basket_add, Buffer, Pizza_ingr, Sample_ingr.

Було розглянуто основні алгоритми процесів сайту для замовлення піци на прикладі головних модулів – конструкторів та модуля користувача.

Були приведені основні складні ділянки коду та структура проекту.

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ПРОГРАМНИМ ДОДАТКОМ

3.1. Порядок встановлення та налаштування системи

Сайт для замовлення піци – це веб-сайт створений на C#, тому потрібно перенести на хостинг і він буде працювати. БД також потрібно перенести на сервер.

Зважаючи на те, що сайт для своєї роботи не потребує значної кількості апаратних ресурсів, конфігурація комп'ютера, що необхідна буде залежати головним чином від вибору операційної системи та серверу.

Наприклад в системі Windows 10 для роботи програми буде достатньою наступна конфігурація:

64-розрядний (x64) процесор з тактовою частотою 2.3 ГГц або вище.

2 ГБ ОЗУ.

20 ГБ вільного місця на жорсткому диску.

Сайт не потребує спеціалізованого апаратного забезпечення.

3.2. Структура інтерфейсу. Інтерфейс та порядок роботи з програмним додатком

Після завантаження головної сторінки сайту користувач вибирає потрібні йому пункти меню. В залежності від доступу там наявні різні опції для вибору, найменше опцій у не зареєстрованого користувача, найбільше у адміна. Також на головній сторінці присутнє вітання з ім'ям користувача.

Ласкаво просимо Владислав

Піца (італ. pizza) — традиційна італійська страва, спочатку у вигляді круглого дріжджового коржика, що випікається з покладеною зверху начинкою з томатного соусу, сиру та часто інших інгредієнтів, таких як м'ясо, овочі, гриби та інші продукти. Невелику піцу іноді називають піцеттою. Кухар, який спеціалізується на приготуванні піци, – піцайоло.



© 2023 - ООО Pizza

Рис.3.1. Головна сторінка

ООО Pizza Головна Авторизуватися\Вийти Каталог Зареєструватися

Рис.3.2. Меню не авторизованого користувача

ООО Pizza Головна Авторизуватися\Вийти Каталог Кошик Замовлення Профіль

Рис.3.3. Меню авторизованого користувача

Для створення шаблону піци потрібно перейти на сторінку каталогу та бути авторизованим. Для цього потрібно авторизуватися або зареєструватися після чого користувач автоматично буде авторизований. На сторінці каталогу потрібно натиснути кнопку під зображенням піци, щоб створити шаблон саме тієї піци. Цієї кнопки не буде у не авторизованого користувача.

Login

Password

Name

Card

Реєстрація

Рис.3.4. Реєстрація

Вхід на сайт

Введіть Email

Введіть пароль

Увійти

Рис.3.5. Авторизація

Каталог нашої смачної піци



Номер: 3 Назва: Дей Ціна: 300
[Шаблон](#)



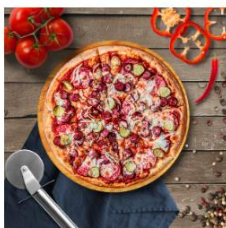
Номер: 4 Назва: М'ясний Вайт Ціна: 321 [Шаблон](#)



Номер: 5 Назва: Пепероні Ціна: 323 [Шаблон](#)



Номер: 6 Назва: Чікен Чіз Ціна: 360 [Шаблон](#)



Номер: 7 Назва: Мисливська Ціна: 400 [Шаблон](#)

Рис.3.6. Каталог авторизованого користувача

ООО Pizza Головна Авторизуватися\Вийти Каталог Зареєструватися

Каталог нашої смачної піци



Номер: 3 Назва: Дей Ціна: 300
Для замовлення зареєструйтесь



Номер: 4 Назва: М'ясний Вайт Ціна: 321 **Для замовлення зареєструйтесь**



Номер: 5 Назва: Пепероні Ціна: 323 **Для замовлення зареєструйтесь**



Номер: 6 Назва: Чікен Чіз Ціна: 360 **Для замовлення зареєструйтесь**



Номер: 7 Назва: Мисливська Ціна: 400 **Для замовлення зареєструйтесь**

Рис.3.7. Каталог не авторизованого користувача

Після натискання кнопки «Шаблон» користувач вибирає необхідні йому параметри шаблону піци та має можливість розрахувати ціну, яка буде

виведена за допомогою модального вікна. Для збереження шаблону потрібно натиснути відповідну кнопку.

ООО Pizza Головна Авторизуватися\Вийти Каталог Кошик Замовлення Профіль Адмін

Дей, Ціна: 300



Номер: 1, Назва: Томатний соус,
Ціна за стандартну порцію: 20,

Номер: 2, Назва: Соус чілі, Ціна за
стандартну порцію: 22,

Номер: 3, Назва: Сир, Ціна за
стандартну порцію: 23,

Номер: 4, Назва: Курка, Ціна за
стандартну порцію: 50,

Номер: 5, Назва: Мисливські
ковбаси, Ціна за стандартну
порцію: 40,

Рис.3.8. Сторінка створення шаблону піци

Розрахована ціна

X

Ціна:
280

Рис.3.9. Розрахована ціна шаблону

Після збереження хоча б одного шаблону піци користувачу можливо створити кошик. Користувач вибирає необхідні параметри та кількість піци, після чого може розрахувати ціну та зберегти кошик за допомогою відповідних кнопок.

Кошик

Номер: 1, Назва: Серветки, Ціна за
одну штуку: 10,



Номер: 2, Назва: Кола, Ціна за
одну штуку: 30,



Номер: 3, Назва: Вода, Ціна за
одну штуку: 40,



Номер: 4, Назва: Перець, Ціна за
одну штуку: 20,



280 , Дей ▾

Кількість:

Розрахувати ціну

Зберегти кошик

Рис.3.10. Сторінка створення кошику

Після створення кошику наявна можливість створити замовлення. Користувач вибирає йому потрібну дату та число, вписує адресу та вибирає кошик, який зберіг до цього.

▼

Адреса:

Дата:

дд.мм.гггг

Час:

--:--

Створити замовлення

Рис.3.11. Сторінка створення замовлення

Після створення об’єктів користувачем, вони можуть бути переглянуті на сторінці профілю користувача та там же видалені за потреби. Також на сторінці наявні ім’я користувача та його логін.

Профіль

Владислав

mail2@gmail.com

Замовлення

Номер: 8,

Адреса: Житомир,

Ціна: 620,

Дата: 31.08.2023 06:59:00,

Номер кошика: 6

8

Видалити

Кошики

Номер: 6,

Ціна: 620,

6

Видалити

Шаблони піц

Номер: 17,

Ціна: 280,

Назва піци: Дей

17

Видалити

Рис.3.12. Сторінка профілю користувача

Також наявна сторінка адміна з можливістю видалення об’єктів інших користувачів, якщо такі повноваження наявні у користувача.

					ІІЗ.КР.Б – 121 – 23 – ІІЗ	
						40

Профіль

Володимир

mail@gmail.com

Замовлення

Видалити

Кошики

Видалити

Шаблони піц

Видалити

Профіль

Владислав

mail2@gmail.com

Замовлення

Номер: 8,

Адреса: Житомир,

Ціна: 620,

Дата: 31.08.2023 06:59:00,

Номер кошика: 6

Кошики

Номер: 6,

Ціна: 620,

6

Видалити

Шаблони піц

Номер: 17,

Ціна: 280,

Назва піци: Дей

17

Видалити

8

Видалити

Рис.3.13. Сторінка адміна

Таблиця 3.1

Відповідність представлень сторінкам сайту

Сторінка сайту	Представлення
Головна	Index
Кошик	Basket
Каталог	Catalog
Замовлення	Order
Сторінка створення шаблону піци	Pizza
Авторизація	Authorization
Реєстрація	Register
Профіль	UsersItems

Висновки до розділу 3

В даному розділі було визначено, що розроблений програмний додаток не потребує спеціалізованого апаратного забезпечення.

На реалізованому сайті для замовлення піци можуть перебувати неавторизований користувач, авторизований користувач та авторизований користувач з повноваженнями адміна. Для неавторизованого користувача доступно тільки переглядання каталогу та можливість зареєструватися. Для авторизованого доступний весь функціонал, а саме створення шаблону піц, кошиків та замовлень, видалення вже створених об'єктів. Для користувача с повноваженнями адміна доступна можливість видаляти об'єкти створені іншими користувачами.

.

ВИСНОВКИ

Під час виконання випускної роботи бакалавра було спроектовано та реалізовано сайт для замовлення піци.

1. Проведено аналіз аналогічних розробок Pizza Hata, Mono pizza, Tarantino family, Pizza house, Pizza 33. Визначено що особливостями сервісів є зосередження уваги на конструкторі піци та кошика. Також визначений базовий функціонал – модуль авторизації та замовити піцу з різною кількістю параметрів.
2. Для реалізації біли використані: мова програмування C#, мова програмування JavaScript та бібліотека jQuery, мова програмування SQL, мови програмування HTML та CSS, фреймворк Entity Framework, фреймворк ASP.NET MVC Framework, інтегроване середовище розробки Microsoft Visual Studio, для роботи з даними був використаний SQL Server Management Studio.
3. Були проаналізовані функціональні та нефункціональні вимоги до програмного додатку.
4. Спроектовано базу даних для збереження інформації. База даних складається з 11 таблиць. Основними є Baket, Addition, Ingredient, Order, Pizza, Sample_pizza, User. Зв'язками багато до багатьох є Baket_add, Buffer, Pizza_ingr, Sample_ingr.
5. Розроблено алгоритми функціонування системи, визначено порядок взаємодії класів під час виконання програмного коду та реалізовано додаток.
6. Реалізовано інтерфейс сайту з різними варіаціями представлень для авторизованого та не авторизованого користувача, а також з повноваженнями адміна. Це зроблено за допомогою компонентів View: Index, Basket, Catalog, Order, Pizza, Authorization, Register, UsersItems, AdminItems.

Реалізований програмний додаток готовий до використання в будь-якій піцерії для зручності створення замовлень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. SQL Server Data Type Mappings [Електронний ресурс] – Режим доступу до ресурсу: <https://learn.microsoft.com/en-us/dotnet/framework/data/adonet/sql-server-data-type-mappings>.
2. How to serialize and deserialize (marshal and unmarshal) JSON in .NET [Електронний ресурс] – Режим доступу до ресурсу: <https://learn.microsoft.com/en-us/dotnet/standard/serialization/system-text-json/how-to?pivots=dotnet-6-0>.
3. Use Bootstrap’s JavaScript modal plugin [Електронний ресурс] – Режим доступу до ресурсу: <https://getbootstrap.com/docs/5.0/components/modal/>.
4. Introduction to the task-based asynchronous pattern (TAP) [Електронний ресурс] – Режим доступу до ресурсу: <https://www.csharptutorial.net/csharp-concurrency/csharp-task/>.
5. Learn ASP.NET Core [Електронний ресурс] – Режим доступу до ресурсу: <https://www.tutorialsteacher.com/core>
6. ASP.NET Core Tutorial [Електронний ресурс] – Режим доступу до ресурсу: https://www.tutorialspoint.com/asp.net_core/index.htm
7. 3 Common Problems with ClaimsIdentity and ClaimsPrincipal in ASP.NET Core [Електронний ресурс] – Режим доступу до ресурсу: <https://www.benday.com/2021/08/13/3-common-problems-with-claimsidentity-and-claimsprincipal-in-asp-net-core/>.
8. SQL INSERT INTO Statement [Електронний ресурс] – Режим доступу до ресурсу: https://www.w3schools.com/sql/sql_insert.asp
9. Overview of ASP.NET Core authentication [Електронний ресурс] – Режим доступу до ресурсу: <https://learn.microsoft.com/en-us/aspnet/core/security/authentication/?view=aspnetcore-7.0>
10. DbContext Class [Електронний ресурс] – Режим доступу до ресурсу: <https://learn.microsoft.com/en-us/dotnet/api/microsoft.entityframeworkcore.dbcontext?view=efcore-7.0>

11.Entity Framework Core: DbContext [Электронный ресурс] – Режим доступа до ресурсу: <https://www.entityframeworktutorial.net/efcore/entity-framework-core-dbcontext.aspx>

12.Dapper Vs. Entity Framework [Электронный ресурс] – Режим доступа до ресурсу: <https://www.linkedin.com/pulse/dapper-vs-entity-framework-nadim-attar>

ДОДАТКИ

					ПЗ.КР.Б – 121 – 23 – ПЗ	
						47

Лістинг програми:

UserController.cs

```

using System.Collections.Generic;
using System.Threading.Tasks;
using Microsoft.EntityFrameworkCore;
using Microsoft.AspNetCore.Mvc;
using System.Security.Claims;
using static Diplom2.Models.DB;
using Diplom2.Models;
using Microsoft.AspNetCore.Authentication;
using Microsoft.AspNetCore.Authentication.Cookies;
using Microsoft.Extensions.Configuration;
using System.Data;
using Microsoft.Data.SqlClient;
using System.Linq;
using System;

namespace AuthApp.Controllers
{
    public class UserController : Controller
    {
        private OrderContext dbO;
        private UserContext db;
        private BasketContext dbB;
        private BufferContext dbBuf;
        private Basket_addContext dbAdd;
        private Sample_pizzaContext dbS;
        private Sample_IngrContext dbI;

        public UserController(UserContext context, OrderContext contextO, BasketContext
contextB, BufferContext contextBuf, Basket_addContext contextAdd, Sample_pizzaContext
contexts, Sample_IngrContext contextI)
        {
            db = context;
            dbO = contextO;
            dbB = contextB;
            dbBuf = contextBuf;
            dbAdd = contextAdd;
            dbS = contexts;
            dbI = contextI;
        }

        [HttpGet]
        public IActionResult Authorization()
        {
            return View();
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Authorization(SingViewModel model)
        {
            if (ModelState.IsValid)
            {
                User user = await db.User.FirstOrDefaultAsync(u => u.Login == model.Login &&
u.Password == model.Password);
                if (user != null)
                {
                    await Authenticate(model.Login);

                    return RedirectToAction("Index", "Home");
                }
            }
        }
    }
}

```

```

        }
        ModelState.AddModelError("", "Некорректные логин и(или) пароль");
    }
    return RedirectToAction("Index", "Home");
}
[HttpGet]
public IActionResult Register()
{
    return View();
}
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Register(RegisterViewModel model)
{
    if (ModelState.IsValid)
    {
        User user = await db.User.FirstOrDefaultAsync(u => u.Login == model.Login);
        if (user == null)
        {
            db.User.Add(new User { Login = model.Login, Password = model.Password,
Admin = 0, Cash = model.Cash, Name = model.Name });
            await db.SaveChangesAsync();

            await Authenticate(model.Login);

            return RedirectToAction("Index", "Home");
        }
        else
        {
            ModelState.AddModelError("", "Некорректные логин и(или) пароль");
        }
        return RedirectToAction("Index", "Home");
    }
}

public IActionResult UsersItems()
{
    var model = GetAuthUserId(User.Identity.Name);
    return View(model);
}

public IActionResult AdminItems()
{
    return View();
}

[HttpPost]
public async Task<IActionResult> DeletOrder(string id)
{
    try
    {
        var idd = int.Parse(id);
        Order order = db0.Order.Single(a => a.id == idd);
        db0.Remove(order);
        await db0.SaveChangesAsync();
    }
    catch (Exception e)
    {
        Console.WriteLine("Exception thrown {0}", e.Message);
    }
    return RedirectToAction("UsersItems", "User");
}

[HttpPost]
public async Task<IActionResult> DeletBaket(string id)
{
    var idd = int.Parse(id);

```

```

        idd);

        Basket basket = dbB.Basket.Single(a => a.id == idd);
        var buffer = dbBuf.Buffer.Where(b => EF.Property<int>(b, "id_basket") == idd);
        var bak_add = dbAdd.Basket_add.Where(b => EF.Property<int>(b, "id_basket") ==

        foreach (var baffle in buffer)
        {
            dbBuf.Remove(baffle);
        }
        foreach (var add in bak_add)
        {
            dbAdd.Remove(add);
        }
        dbB.Remove(basket);
        await dbAdd.SaveChangesAsync();
        await dbBuf.SaveChangesAsync();
        await dbB.SaveChangesAsync();
        return RedirectToAction("UsersItems", "User");
    }

    [HttpPost]
    public async Task<IActionResult> DeletSample(string id)
    {
        try
        {
            var idd = int.Parse(id);
            Sample_pizza sample = dbS.Sample_pizza.Single(a => a.id == idd);
            var ingr = dbI.Sample_Ingr.Where(b => EF.Property<int>(b, "id_sample") ==

            foreach (var ing in ingr)
            {
                dbI.Remove(ing);
            }
            dbS.Remove(sample);
            await dbI.SaveChangesAsync();
            await dbS.SaveChangesAsync();
        }
        catch (Exception e)
        {
            Console.WriteLine("Exception thrown {0}", e.Message);
        }
        return RedirectToAction("UsersItems", "User");
    }

    private async Task Authenticate(string email)
    {
        var claims = new List<Claim>
        {
            new Claim(ClaimsIdentity.DefaultNameClaimType, email)

        };
        ClaimsIdentity id = new ClaimsIdentity(claims, "ApplicationCookie",
ClaimsIdentity.DefaultNameClaimType, ClaimsIdentity.DefaultRoleClaimType);
        await HttpContext.SignInAsync(CookieAuthenticationDefaults.AuthenticationScheme,
new ClaimsPrincipal(id));
    }

    public async Task<IActionResult> Logout()
    {
        await
HttpContext.SignOutAsync(CookieAuthenticationDefaults.AuthenticationScheme);
        return RedirectToAction("Authorization", "User");
    }
}
}

```

Модель DB:

```
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Collections;
using System.Linq;
using System.Threading.Tasks;
using System.Data;
using Dapper;
using Microsoft.Data.SqlClient;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using FluentNHibernate;
using System.Data.Entity.ModelConfiguration;

namespace Diplom2.Models
{
    public class DB : Microsoft.EntityFrameworkCore.DbContext
    {
        public class UserContext : Microsoft.EntityFrameworkCore.DbContext
        {
            public Microsoft.EntityFrameworkCore.DbSet<User> User { get; set; }
            public UserContext(DbContextOptions<UserContext> options)
                : base(options)
            {
                Database.EnsureCreated();
            }
        }
        public class BaketContext : Microsoft.EntityFrameworkCore.DbContext
        {
            public Microsoft.EntityFrameworkCore.DbSet<Baket> Baket { get; set; }
            public BaketContext(DbContextOptions<BaketContext> options)
                : base(options)
            {
                Database.EnsureCreated();
            }
        }
        public class BufferContext : Microsoft.EntityFrameworkCore.DbContext
        {
            public Microsoft.EntityFrameworkCore.DbSet<Buffer> Buffer { get; set; }
            public BufferContext(DbContextOptions<BufferContext> options)
                : base(options)
            {
                Database.EnsureCreated();
            }
        }
        public class Baket_addContext : Microsoft.EntityFrameworkCore.DbContext
        {
            public Microsoft.EntityFrameworkCore.DbSet<Baket_add> Baket_add { get; set; }
            public Baket_addContext(DbContextOptions<Baket_addContext> options)
                : base(options)
            {
                Database.EnsureCreated();
            }
        }
        public class Sample_pizzaContext : Microsoft.EntityFrameworkCore.DbContext
        {
            public Microsoft.EntityFrameworkCore.DbSet<Sample_pizza> Sample_pizza { get;
set; }
            public Sample_pizzaContext(DbContextOptions<Sample_pizzaContext> options)
                : base(options)
            {

```

```

        Database.EnsureCreated();
    }
}

public class Sample_IngrContext : Microsoft.EntityFrameworkCore.DbContext
{
    public Microsoft.EntityFrameworkCore.DbSet<Sample_Ingr> Sample_Ingr { get; set; }

    public Sample_IngrContext(DbContextOptions<Sample_IngrContext> options)
        : base(options)
    {
        Database.EnsureCreated();
    }
}

public class OrderContext : Microsoft.EntityFrameworkCore.DbContext
{
    public Microsoft.EntityFrameworkCore.DbSet<Order> Order { get; set; }
    public OrderContext(DbContextOptions<OrderContext> options)
        : base(options)
    {
        Database.EnsureCreated();
    }
}

public static IDbConnection Connection
{
    get
    {
        return new SqlConnection("Server=DESKTOP-
23FRT8U\\SQLEXPRESS;Database=Pizza;Trusted_connection=True;TrustServerCertificate =
True;MultipleActiveResultSets=true");
    }
}

public static List<User> GetUserNames()
{
    var result = Connection.Query<User>("SELECT * FROM [User]").ToList();
    return result;
}

public static List<Pizza> GetPizzaNames()
{
    var result = Connection.Query<Pizza>("SELECT * FROM [Pizza]").ToList();
    return result;
}

public static List<Pizza> GetPizza(int id)
{
    var result = Connection.Query<Pizza>($"SELECT * FROM [Pizza] WHERE [id]
=('{id}')").ToList();
    return result;
}

public static List<Ingredient> GetIngr(int id)
{
    var buf_id = Connection.Query<Pizza_Ingr>($"SELECT [id_ingr] FROM [Pizza_Ingr]
WHERE [id_pizza] =('{id}')").ToList();
    List<Ingredient> result = new List<Ingredient>();
    for (int i = 0; i < buf_id.Count; i++)
    {
        result.Add(Connection.Query<Ingredient>($"SELECT * FROM [Ingredient] WHERE
[id] =('{buf_id[i].id_ingr}')").ToList()[0]);
    }
}

```

```

        return result;
    }

    public static int GetIdSample()
    {
        var all = Connection.Query<Sample_pizza>("SELECT * FROM [Sample_pizza]").ToList();
        var result = all[all.Count - 1].id;
        return result;
    }

    public static int GetIdBaket()
    {
        var all = Connection.Query<Baket>("SELECT * FROM [Baket]").ToList();
        var result = all[all.Count - 1].id;
        return result;
    }

    public static double GetBaketPrice(int id)
    {
        var result = Connection.Query<Baket>($"SELECT [Price_baket] FROM [Baket] WHERE [id] = ({id})").ToList()[0].Price_baket;
        return result;
    }

    public static string GetAuthUserName(string email)
    {
        var result = Connection.Query<User>($"SELECT [Name] FROM [User] WHERE [Login] = ({email})").ToList()[0].Name;
        return result;
    }

    public static int GetAuthUserId(string email)
    {
        var result = Connection.Query<User>($"SELECT [id] FROM [User] WHERE [Login] = ({email})").ToList()[0].id;
        return result;
    }

    public static List<Addition> GetAdd()
    {
        var result = Connection.Query<Addition>("SELECT * FROM [Addition]").ToList();
        return result;
    }

    public static List<Sample_pizza> GetSamples(int id)
    {
        var result = Connection.Query<Sample_pizza>($"SELECT * FROM [Sample_pizza] WHERE [id_user] = ({id})").ToList();
        return result;
    }

    public static List<Baket> GetBakets(int id)
    {
        var result = Connection.Query<Baket>($"SELECT * FROM [Baket] WHERE [id_user] = ({id})").ToList();
        return result;
    }

    public static double GetSamplesPriceById(int id)
    {
        var result = Connection.Query<Sample_pizza>($"SELECT [Price] FROM [Sample_pizza] WHERE [id] = ({id})").ToList()[0].Price;
        return result;
    }
}

```

```

        public static string GetPizzaName(int id)
        {
            var result = Connection.Query<Pizza>($"SELECT [Name] FROM [Pizza] WHERE [id]
=('{id}'))").ToList()[0].Name;
            return result;
        }

        public static int GetIdSamplePizza(int id)
        {
            var result = Connection.Query<Sample_pizza>($"SELECT [id_pizza] FROM
[Sample_pizza] WHERE [id] =('{id}'))").ToList()[0].id_pizza;
            return result;
        }

        public static int GetBufferIdSample(int id)
        {
            var result = Connection.Query<Buffer>($"SELECT [id_sample] FROM [Buffer] WHERE
[id_baket] =('{id}'))").ToList()[0].id_sample;
            return result;
        }

        public static User GetUserData(int id)
        {
            var result = Connection.Query<User>($"SELECT * FROM [User] WHERE [id]
=('{id}'))").ToList()[0];
            return result;
        }

        public static List<Order> GetAllOrder(int id)
        {
            var result = Connection.Query<Order>($"SELECT * FROM [Order] WHERE [id_user]
=('{id}'))").ToList();
            return result;
        }

        public static List<Baket> GetAllBaket(int id)
        {
            var result = Connection.Query<Baket>($"SELECT * FROM [Baket] WHERE [id_user]
=('{id}'))").ToList();
            return result;
        }

        public static List<Sample_pizza> GetAllSample(int id)
        {
            var result = Connection.Query<Sample_pizza>($"SELECT * FROM [Sample_pizza] WHERE
[id_user] =('{id}'))").ToList();
            return result;
        }

        public class User
        {
            public int id { get; set; }
            public string Name { get; set; }
            public int Cash { get; set; }
            public int Password { get; set; }
            public int Admin { get; set; }
            public string Login { get; set; }
        }

        public class Order
        {
            public int id { get; set; }
            public int id_user { get; set; }
            public int id_baket { get; set; }
            public double Total_price { get; set; }
            public DateTime Data_in { get; set; }
            public string Adress { get; set; }
        }

```

```

public class Basket
{
    public int id { get; set; }
    public int id_user { get; set; }
    public double Price_basket { get; set; }
}

public class Pizza
{
    public int id { get; set; }
    public string Name { get; set; }
    public int Price_pizza { get; set; }
}

public class Sample_pizza
{
    public int id { get; set; }
    public int id_pizza { get; set; }
    public int id_user { get; set; }
    public double Price { get; set; }
    public int Size { get; set; }
    public int Part { get; set; }
}

public class Buffer
{
    public int id_sample { get; set; }
    public int id_basket { get; set; }
    public int Count { get; set; }
    public int id { get; set; }
}

public class Pizza_Ingr
{
    public int id_pizza { get; set; }
    public int id_ingr { get; set; }
}

public class Ingredient
{
    public int id { get; set; }
    public string Name { get; set; }
    public int Price_ingr { get; set; }
}

public class Sample_Ingr
{
    public int id { get; set; }
    public int id_sample { get; set; }
    public int id_ingr { get; set; }
    public int Multilier { get; set; }
}

public class Basket_add
{
    public int id_basket { get; set; }
    public int id_add { get; set; }
    public int Count { get; set; }
    public int id { get; set; }
}

public class Addition
{
    public int id { get; set; }
    public string Name { get; set; }
    public int Price_add { get; set; }
}

```

```

    }
}

Представлення Pizza:
@using System.Text.RegularExpressions;
@using System.Text.Json;
@using System.Text.Json.Serialization;
@{var item = Diplom2.Models.DB.GetPizza(@Model[0])[0];
    var ingr = Diplom2.Models.DB.GetIngr(item.id);
    int[] ids = new int[ingr.Count];
    for (int i = 0; i < ingr.Count; i++)
    {
        ids[i] = ingr[i].id;
    }
    string jsonString = JsonSerializer.Serialize(ids);
}
<div class="text-center">
    <h1 class="display-4">@item.Name, Ціна: @item.Price_pizza</h1>
</div>

<section class="text-center mb-4">
    <div class="row wow fadeIn">
        <div class="col-lg-3 col-md-6 mb-4">
            @for (int i = 0; i < ingr.Count; i++)
            {
                <div>
                    Номер: @ingr[i].id,
                    Назва: @ingr[i].Name,
                    Ціна за стандартну порцію: @ingr[i].Price_ingr,
                    <div class="slidecontainer">
                        <input type="range" min="0" max="2" value="1" class="slider"
id="@ingr[i].id">
                    </div>
                </div>
            }
        </div>
        <div>
            Розмір:
            <input type="range" min="1" max="3" value="2" class="slider" id="Size">
        </div>
        <div>
            Частина:
            <input type="range" min="1" max="4" value="4" class="slider" id="Part">
        </div>
    </div>
</div>
<div>
    <input type="button" id="submit" value="Розрахувати ціну" />
</div>
<div>
    <input id="save" type="button" value="Зберегти шаблон піци" />
</div>
<div id="modal" class="modal" tabindex="-1">
    <div id="dialog" class="modal-dialog">
    </div>
</div>
@section scripts
{
    <script type="text/javascript">
        $("#submit").click(function () {
            let ids = [ @jsonString ]
            let arr = []

```

```

        for (let i = 0; i < @ingr.Count; i++) {
            arr.push(`${ids[0][i]}`).val())
        }
        var size = $('#Size').val()
        var part = $('#Part').val()
        var id = @item.id;
        var b = JSON.stringify(arr)
        var date = { size: size, part: part, b: b, id: id }
        $.post("/Pizza/Summa", date, function (data) {
            $('#dialog').html(data);
            $('#modal').modal('show');
        })
    })

</script>
<script>
$("#save").click(function () {
    let ids = [ @jsonString ]
    let arr = []
    for (let i = 0; i < @ingr.Count; i++) {
        arr.push(`${ids[0][i]}`).val())
    }
    var size = $('#Size').val()
    var part = $('#Part').val()
    var id = @item.id;
    var b = JSON.stringify(arr)
    var date = { size: size, part: part, b: b, id: id }
    $.post("/Pizza/Save", date)
})</script>
}
</section>

```