Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
(повне найменування вищого навчального закладу)
Факультет компп'ютерно-інформаційних систем і програмної інженерії
(назва факультету)
Програмної інженерії
(повна назва кафедри)

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проекту (роботи)

магістр
(освітній ступінь (освітньо-кваліфікаційний рівень))

на тему: **Розробка програмної системи керування контентом з використанням мови програмування C# та технологій ASP.Net WebAPI**

Виконав: студент (ка) 6 курсу, групи СПм-62
спеціальності (напряму підготовки) 121 Інженерія Програмного забезпечення
(шифр і назва спеціальності (напряму підготовки))

(підпис) (прізвище та ініціали)

Керівник (підпис) Кінах Я.І. (прізвище та ініціали)

Нормоконтроль (підпис) (прізвище та ініціали)

Рецензент (підпис) (прізвище та ініціали)

м. Тернопіль – 201 9_

АНОТАЦІЯ

Дипломна робота на тему «розробка програмної системи керування контентом на основі використання мови C#по технології ASP.NET WEBAPI» Сороцької Наталі Тарасівни – Тернопільський національний технічний університет імені Івана Пулюя, Факультет комп'ютерно-інформаційних систем і програмної інженерії, Кафедра програмної інженерії, група СПм–62 // Тернопіль, 2019.

С. – 115, рис. – 43, табл. – 4, слайдів. – 12, додат. – 3.

Метою дипломної роботи є дослідження, проектування та розробка Web-сервісу керування рецептами та проектування нейронної мережі для створення рецептів за допомогою машинного навчання. Методи та програмні засоби, використані при виконанні розробки системи: мова програмування C# та її бібліотеки, середовище розробки Visual Studio, середовище розробки бази даних SQLMeneger.

Опрацьована вхідна та результуюча інформація, досліджено технології створення динамічних web-сервісів, описана структура наявної інформаційної бази, блок-схеми та кодові взірці основних функцій, проведена розробка програмного забезпеченя, вивчено питання безпеки охорони праці.

Ключові слова: WEB-СЕРВІС, НЕЙРОННА МЕРЕЖА, ПРОГРАМНА СИСТЕМА, МАШИННЕ НАВЧАННЯ.

ABSTRACT

Diploma work on theme “development of software content management system based on the use of C # language on ASP.NET WEBAPI technology” by Natalia Sorotskaya - Ivan Puliuyi Ternopil National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Software Engineering, SPm-62 group // Ternopil, 2019.

C. - 115, fig. - 43, tab. - 4 slides. - 12, add. - 3.

Diploma work designed to make research, design and develop a web-based recipe management service and design a neural network for creating recipes through machine learning. Methods and software used in the development of the system: C # programming language and its libraries, Visual Studio development environment, SQLMeneger database development environment.

Input and output information processed, dynamic web services creation technologies investigated, structure of available information base, block diagrams and code models of basic functions described, software development conducted, labor safety issues examined.

Keywords: WEB-SERVICE, NEURAL NETWORK, SOFTWARE, MACHINE TRAINING.

ПЕРЕЛІК ОСНОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

ІС - Інформаційна система.

API – (Application Programming Interface) інтерфейс прикладного програмування.

EF – Entity Framework.

REST – (Representational state transfer) — передача репрезентативного стану.

HTTP – (HyperText Transfer Protocol) — протокол передачі гіпертекстау

UML – (Unified Modeling Language) — уніфікована мова моделювання

ML – (Mashine Learning) - машинне навчаня

CLI – (Common Language Infrastructure) –ехнічний стандарт, розроблений корпорацією Microsoft та стандартизований ISO і ECMA

SDK – (software Development Kit) –набір із засобів розробки, утиліт і документації, який дозволяє програмістам створювати прикладні програми за визначеною технологією або для певної платформи

ВСТУП

Сучасний світ важко собі уявити без комп'ютерів. Інформаційні технології стали невід'ємною складовою будь-якої сфери нашого життя. Особливу роль в них відіграє мережа Internet. Це найбільш розвинена у світі інформаційна система, за допомогою якої здійснюється обмін інформацією між мільйонами її користувачів.

Поява web-технологій відіграла важливу роль у формуванні сьогоденного світу інформаційних технологій. Сучасні web-ресурси надають їх користувачам широкий спектр можливостей, пов'язаних з інформацією. Головним завданням будь-якого інтернет-сервісу є інформаційне забезпечення своїх користувачів.

Завдяки стрімкому розвитку інтернету сучасне суспільство змінилося до невпізнання. Люди отримали нові можливості в спілкуванні, трудовій діяльності, проведенні дозвілля. Кордони, що існували раніше, стираються. Зараз люди, що проживають на різних континентах, без будь-яких проблем можуть спілкуватися в мережі і обмінюватися інформаційними ресурсами. І головний подарунок інтернету - це необмежені можливості доступу до колосальних масивів інформації. За допомогою звичайних пошукових систем можна практично миттєво отримати будь-яку потрібну інформацію.

Та з появою нейронних мереж, наші можливості моментально зросли в рази. Можливості пошуку, навчання та створення нових речей невідомих раніше людині, тепер не новинка.

Метою даної магістерської роботи є проектування та розробка web-сервісу керування контентом.

Результатом даної магістерської роботи буде створено web-сервіс, спроектований за архітектурним шаблоном REST [10] та проектована частина подальшої розробки сервісу, яка покращить його можливості за допомогою використання нейронних мереж.

Для досягнення цієї мети необхідно:

- дослідити проблему та шляхи її вирішення;
- провести аналіз предметної області;
- розглянути принципи роботи web-сервісу;
- розглянути принципи розробки та проектування web-сервісу;
- ознайомитись із можливими способами взаємодії програмних клієнтів з сервером через програмний інтерфейс;
- скласти вимоги до функціоналу web-сервісу;
- спроектувати програмний інтерфейс сервісу для взаємодії з зовнішніми системами (API);
- описати структуру web-сервісу;
- описати модель бази даних;
- спроектувати модель бази даних;
- спроектувати структуру сервісу;
- реалізувати базу даних.
- реалізувати структуру сервісу;
- дослідити нейронні мережі;
- провести аналіз основних її задач;
- спроектувати основні функції нейронної мережі;

1.ПРОЕКТУВАННЯ, РОЗРОБКА ТА ТЕСТУВАННЯ ВЕБ СЕРВІСУ

1.1Проблематика та аналіз предметної області

В наш час, новітні технології стрімко розвиваються. Щодня на різноманітну тематику з'являються нові продукти, сайти, проекти, додатки. Будь-хто, за допомогою натискання декількох клавіш, може знайти будь-яку йому потрібну інформацію в інтернеті.

Сфера кулінарії також не відстає в цьому плані. У всесвітній мережі є безліч різноманітних сайтів, зайшовши на які можна побачити різноманітні рецепти, які може приготувати будь-хто: чи то школяр, чи доросла людина.

Проте не кожен сайт може надати саме ту інформацію, яку ви шукаєте. Ви можете знайти сторінки з дорогими вишуканими салатами чи холодними закусками, та не завшди у вашому холодильнику є потрібні інгредієнти. Саме через цю проблему було вирішено розробити сервіс та базу даних рецептів, які за своїми інгрендієнтами та способом приготування будуть доступні кожному.

Коли розробник має якусь хорошу ідею для реалізації, наприклад сайт з рецептами, який буде простим у користуванні і разом з тим матиме доволі велику базу рецептів, або ж додаток на мобільний пристрій, який дозволить готувати страву, не відходячі від плити, то перед ним рано чи пізно постане проблема з наповненням бази даних сайту чи додатку необхідною інформацією, причому у великих обсягах. Так само потрібно буде спроектувати базу даних для рецептів та інгредієнтів, забезпечити їх класифікацію, дозволити зберігати зворотні відгуки від кінцевих користувачів. Для вирішення цієї проблеми може бути декілька способів, наприклад: складання бази даних рецептів та інгредієнтів власноручно, або ж пошук по існуючим сайтам рецептів. Так само розробник зможе написати парсер сайтів, який буде наповнювати базу даних рецептів в автоматичному режимі. Найкращим виходом в цій ситуації є використання сайтів рецептів з існуючими

програмними інтерфейсами для доступу до потрібної розробнику інформації, за допомогою якого його сайт або мобільна аплікація зможе здійснювати пошук рецептів та інградієнтів у вже існуючій базі даних.

Так само перед розробником системи може бути інша проблема – міграція сайту з однієї платформи на іншу, у випадку якщо початкова платформа не в змозі забезпечити доволі просте розширення функціоналу, редизайн, або ж розробка додаткового програмного продукту, який повинен використовувати ту ж саму базу даних. У випадку, якщо відбувається міграція сайту з однієї платформи на іншу - це призводить до переписування серверного коду сайту під цільову платформу. Або ж у випадку написання мобільного додатку виникає потреба забезпечити функціонал для взаємодії з системою що призводить до дублювання вже інсуючого функціоналу. Шляхом вирішення цієї проблеми може бути web-сервіс, який інкапсулює в собі всю необхідну бізнес логіку для створення, збереження, читання, пошуку, редагування та видалення інформації. Це дозволить безболісно проводити зміну дизайну сайту, оскільки бізнес логіка буде відокремлена від користувацької браузерної частини. Так само це дозволить мобільній аплікації в повній мірі використовувати той самий функціонал, що і сайту, уникаючи його дублювання.

Web-сервіс, який буде спроектований під час виконання даного дипломного проекту надаватиме стороннім сервісам змогу використовувати свій функціонал шляхом взаємодії через програмний інтерфейс. Це позбавляє розробників необхідності створювати власну базу даних, піклуватись про її цілісність та внутрішню будову. Web-сервіс також забезпечує використання спільної бази знань для усіх клієнтів, що дозволяє уникнути проблем з пошуком та дублюванням інформації.

Для роботи web-сервісу необхідна база даних, яка дозволить зберігати дані, отримані від сторонніх систем і які будуть в подальшому оброблятись цими системами.

Також необхідно спроектувати програмний інтерфейс, завдяки якому буде відбуватись взаємодія між web-сервісом та сторонніми системами. Для проектуваня програмного інтерфейсу необхідно визначити функціонал,який буде надаватись цим web-сервісом.

1.1.2 Аналіз існуючих інформаційних систем

Провівши пошук ресурсів, які б могли надати у вільний доступ свій API, не було знайдено жодного такого ресурсу. В зв’язку з цим було прийнято рішення розробки web-сервісу, який може служити джерелом інформації для інших програмних інформаційних систем.

Для забезпечення максимально якісного функціоналу даного сервісу необхідно ретельно провести аналіз предметної області.

В якості джерел інформації про предметну область було обрано наступні ресурси:

Cooking channel – американський кулінарний канал, присвячений пристрастним любителям їжі[11];

Рецепстер – пошукова система кулінарних рецептів, яка проводить пошук по іншим сайтам[12];

Jamie Oliver – сайт кулінарних рецептів англійського кухара, ресторатора, телеведучого, популяризатора домашньої кухні та здорового харчування Джеймса Тревора «Джеймі» Олівера[13].

Оглянемо кулінарний канал cookingchanneltv.com. На головній сторінці можна побачити головне меню, сезонні страви, телепрограму та топ популярних страв (рисунок 1.1.2.1).

Рисунок 1.1.2.1- Головна сторінка кулінарного каналу

У головному меню знаходиться розділ “рецепти”, у якому можна знайти безліч популярних сезонних страв та страв різних країн світу. Так само можна знайти потрібний рецепт завдяки пошуку (рисунок 1.1.2.2).

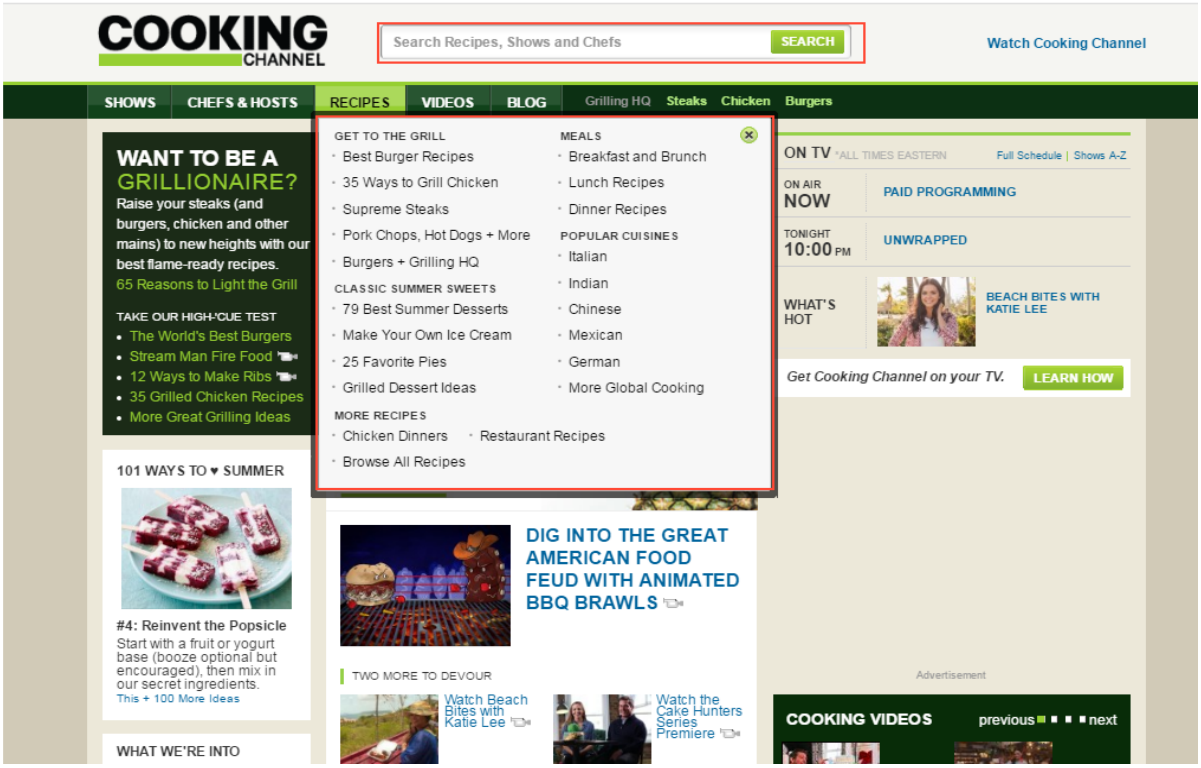

Рисунок 1.1.2.2 – Пошук рецептів

Даний канал є зрозумілим та легким у використанні. Він надає можливість легкого пошуку рецептів, перегляду каналу, та відео приготування рецептів.

Недоліком даного порталу є виключно англійська мова. Також мінусом даного каналу є велика кількість різних компонентів на головній сторінці, через що користувач може запутатись у навігації.

Recipester.ru – це новий сервіс, суть якого полягає в агрегації рецептів з різних кулінарних сайтів. На головній сторінці – пошукова стрічка, в яку можна ввести назву страви або перелік наявних у вас продуктів. Можливо звузити пошук тільки за назвою, за інгредієнтами або по тексту рецепта (рисунок1.1.2.3).

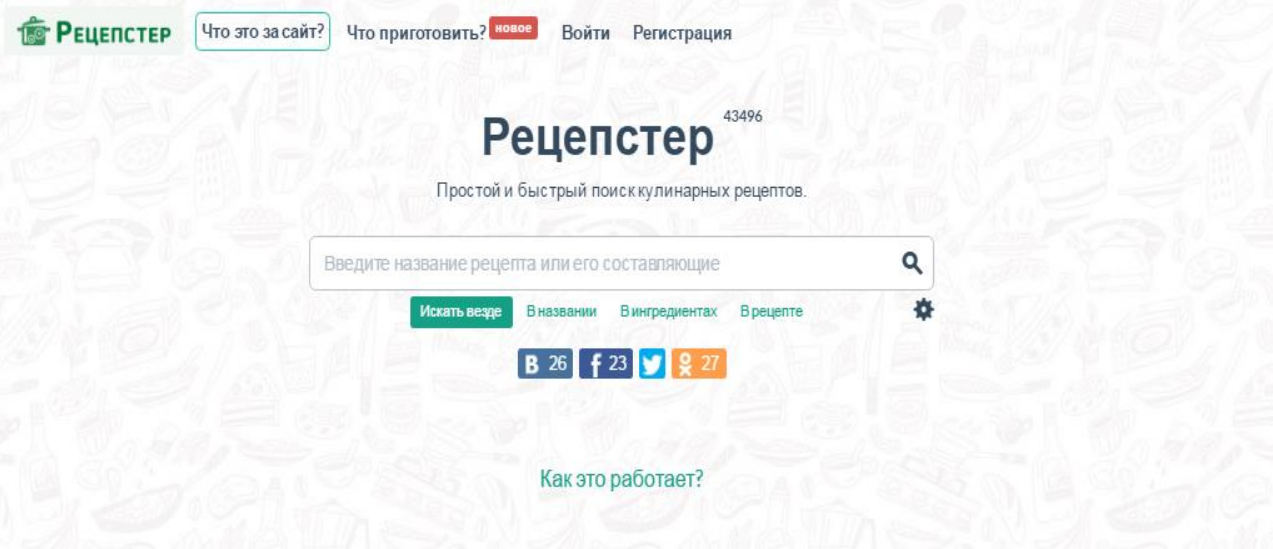

Рисунок 1.1.2.3 – Головна сторінка Recipester.ru

В цьому сервісі наявний розділ “Що приготувати ?” . Необхідно вибрати сніданок, обід або вечерю і отримуєте три варіанти на вибір. Для зміни варіантів необхідно оновити сторінку.

Також в куточку кожного зображення зазначено, з якого воно сайту. Клік по картинці або заголовку відразу перекине вас на оригінальний сайт з рецептом (рисунок 1.1.2.4).

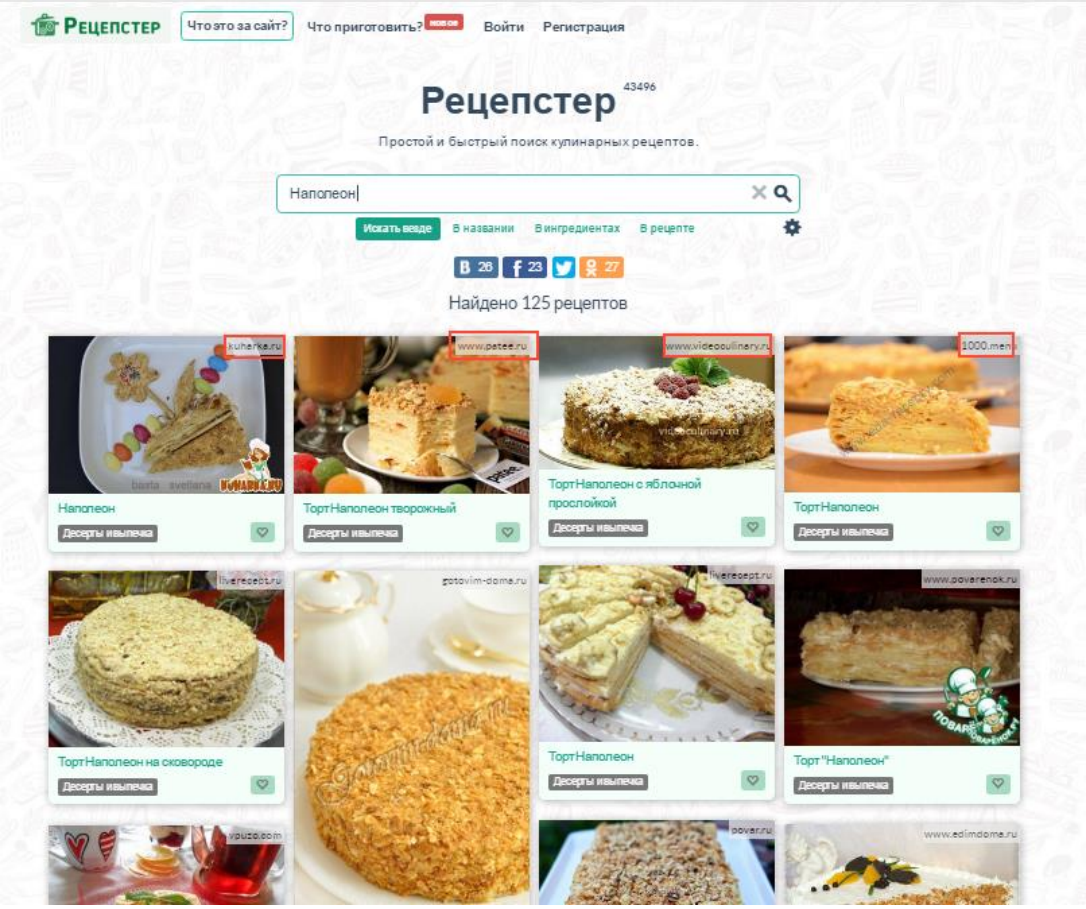

Рисунок 1.1.2.4 – Відображення знайдених рецептів.

Недоліком даного сервісу є невпорядковане розташування рецептів на сторінці, а також при введенні інгредієнту українською мовою вам не завжди коректно відобразиться результат, адже сервіс виключно російський.

jamieoliver.com - це сайт відомого англійського кухара, на якому ви знайдте безліч рецептів, які є простими та легкими у приготуванні. Зручний інтерфейс та легка навігація допоможуть у пошуку улюбленої страви (рисунок 1.1.2.5). Завдяки цьому сайту ви знайдете безліч страв, які сподобаються всій вашій сім'ї, щодня ви зможете радувати їх все новими вишуканими стравами.

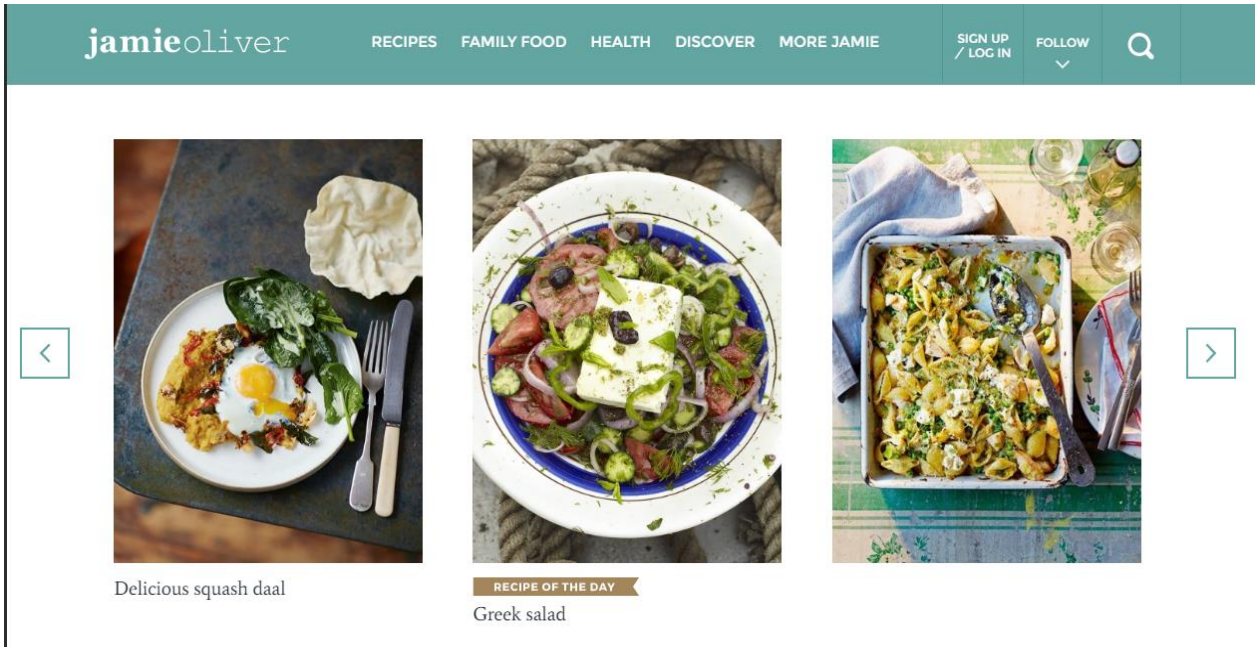

Рисунок 1.1.2.5 – Головна сторінка сайту jamieoliver.com

Серед рецептів маєте змогу знайти безліч страв, які є корисними для здоров’я, страви різних країн світу, вегетеріанських страв, а також здійснити пошук по інгредієнтах. На ньому можна знайти рецепти які легко та швидко приготувати. Наприклад рецепти, які готуються за 15 чи 30 хв. Також переглянути різні рубрики або скористуватись для пошуку зручним меню у верхній панелі сайту (рисунок 1.1.2.6).

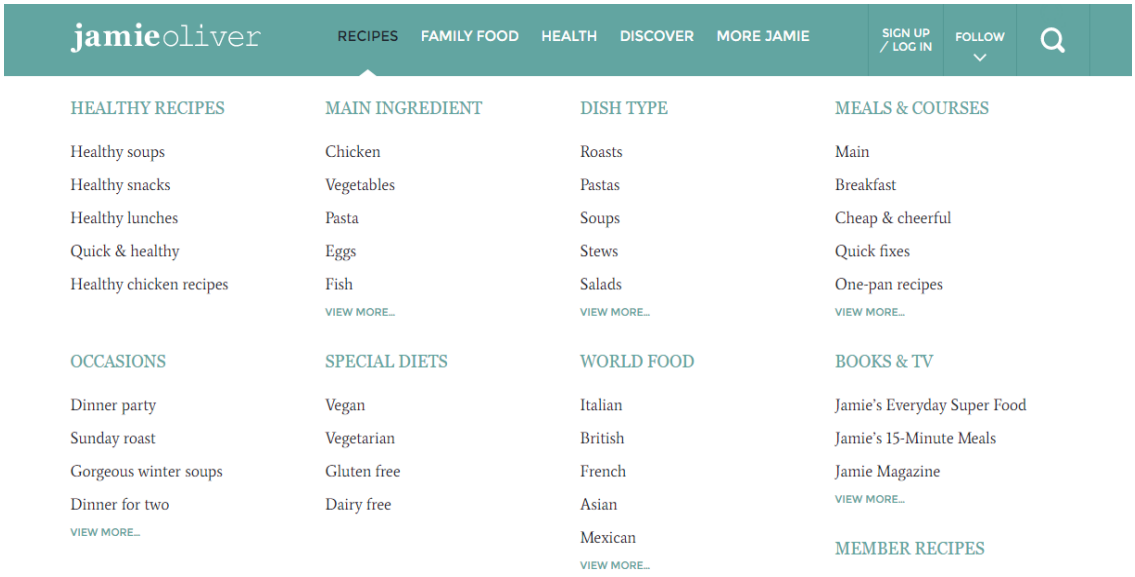

Рисунок 1.1.2.6– Вибір рецептів.

Даний сайт є надзвичайно легким та зрозумілим у користуванні, але також є виключно на англійській мові, тому мовний бар'єр може завадити користувачу знайти необхідну інформацію.

В даному дипломному проекті так само буде розроблено сайт як приклад стороннього додатку, який використовує цей web-сервіс. Створений сайт надасть неявний доступ до web-сервісу, дозволяючи отримувати рецепти з бази знань. Так само сайт надасть змогу користувачу створювати власні рецепти, додавати нові інгредієнти тощо.

1.1.3Постановка задачі

Постановка завдання - важливий етап створення сайту або web-сервісу, на цьому етапі визначаються основні параметри і задається напрямок розробки. Результати постановки задачі цілком відчутні і конкретні, на підставі цих даних відбувається створення технічного завдання.

Постановка завдання є логічним завершенням збору і аналізу інформації про продукт, який належить розробляти.

При розробці web-сервісу кулінарних рецептів буде визначено певний обсяг робіт, а саме:

- аналіз предметної області;
- опис необхідного функціоналу;
- опис вимог до системи;
- опис програмного інтерфейсу системи;
- створення моделей бази даних;
- визначення кількості ресурсів, необхідних для реалізації системи;
- реалізація бізнес логіки сервісу та його взаємодії з базою даних;
- реалізація програмного інтерфейсу для забезпечення взаємодії сервіса з зовнішніми програмними системами;
- розробка тестової клієнської частини;

1.1.4. Аналіз предметної області.

При проведенні аналізу предметної області було виділено ряд сутностей, які є необхідними для корекного функціонування системи та встановлено логічні зв’язки між ними.

Під час аналізу було виділено наступні сутності: рецепт, інградієнт.

Рецепт - це інструкція з приготування кулінарного виробу. Він містить інформацію про необхідні інградієнти, їх кількість та подає інструкцію по поєднанню та обробці. Кулінарні рецепти описують механічну і теплову обробку інгредієнтів, способи сервірування готових страв.

Інградієнт - компонент, який використовується для приготування конкретної страви.

В ході аналізу предметної області між рецептом та інгредієнтом було встановлено наступну залежність, зображену на риснку 1.1.4.1

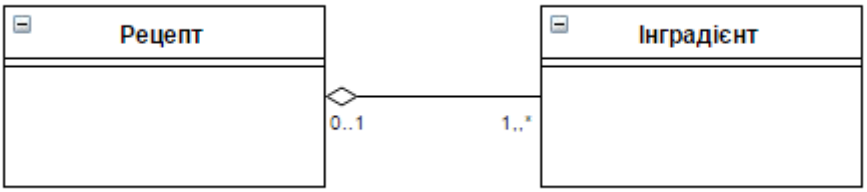

Рисунок 1.1.4.1 – Залежність між сутностями “Рецепт” та “Інредінт”.

Оскільки рецепт містить інгредієнти в певній кількості, було визначено наступну сутність: “Порція”.

Порція - кількість інградієнту, необхідного для приготування конкретної страви.

Залежність між сутностями зображено на рисунку 1.1.4.2

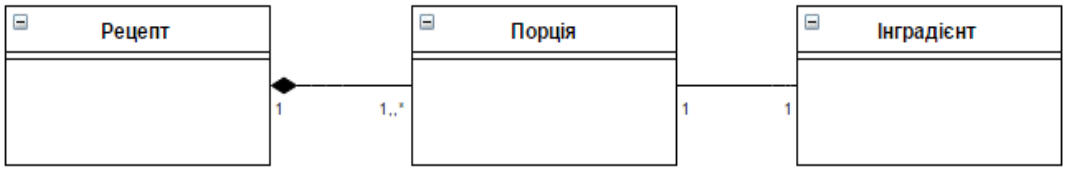

Рисунок 1.1.4.2- Залежність сутності “Порція” із сутностями “Рецепт” та “Інгредієнт”

Порція містить додаткове поле, яке відображає кількість інгредієнту, необхідну для створення однієї порції страви і відноситься до сутності “Інгредієнту” як 1-1, а до сутностіі “Рецепт” як *-1 .

З метою ідентифікації власників рецептів було прийнято рішення проведення додаткової штучної сутності “Користувач”. В результаті з’являється новий зв’язок між сутностями “Користувач” та “Рецепт”, як 1-*.

З метою обмеження доступу користувачів до певного функціоналу було створено додаткові ролі “Адміністратор” та “ Користувач” .

1.1.5 Опис функціоналу.

В результаті детального аналізу вимог до функціоналу системи було виокремлено наступний функціонал:

- додавання рецептів;
- редагування рецептів;
- видалення рецептів;
- читання та пошук рецептів та інгредієнтів;
- додавання інгредієнтів.

Оскільки в ході аналіз предметної області було виокремлено додаткову сутність користувач необхідно забезпечити наступний функціонал:

- реєстрація;
- авторизація.

Також для більшого розширення функціоналу було виокремлений додатковий функціонал:

- створнення коментарів;
- редагування коментарів;
- видалення коментарів;
- додавання рецептів у список улюблених.

1.2. Проектування інформаційної системи

1.2.1 Проектування функціональної структури системи.

Реалізація клієнт-серверної архітектури згідно принципів архітектурного стилю REST дозволяє відокремити клієнтську частину додатку від серверної частини. Слабка зв’язаність серверної і клієнтської частини додатку дає змогу створювати різноманітні клієнтські частини. Наприклад, сайти, мобільні додатки тощо. Також цей підхід дає змогу масштабувати серверну частину шляхом розміщення її на декількох машинах.

Основними перевагами REST є:

- надійність (за рахунок відсутності необхідності зберігати інформацію, яка може бути втрачена);
- масштабування;
- легкість внесення змін;
- продуктивність;
- прозорість системи взаємодій;
- портативність компонентів;
- простота інтерфейсів;
- можливість еволюціонувати, пристосовуватись до нових потреб.

1.2.2 Перелік основних вимог до функціоналу web-сервісу.

Даний web-сервіс повинен надавати можливість створення, читання, редагування та видалення рецептів, додавати до кожного з рецептів певний набір інгредієнтів, переглядати вже існуючі рецепти, додавати до списку «улюблених», переглядати створенні рецепти, а також додавати нові інгредієнти.

1.2.3 Проектування структури web-сервісу.

Необхідно розділити весь функціонал на певні логічні частини згідно предметної області. Це здійснюється з метою спрощення розуміння структури сервісу розробниками, які в майбутньому працюватимуть з даним сервісом.

Згідно предметної області було виділено наступні сутності:

- користувачі сервісу;
- рецепти, які є розміщені цими користувачами;
- інгредієнти, з яких складаються рецепти.

Відповідно до сутностей було виділено три основні модулі:

- модуль інгредієнтів;
- модуль рецептів;
- модуль користувачів.

1.2.4 Проектування бази даних

Проаналізувавши предметну область було виявлено наступні основні сутності:

- інгредієнти.
- рецепти;
- користувачі.

Сутність “Інгредієнт” повинна мати наступні поля

Таблиця 1.2.4.1- Поля сутності “Інгредієнт”

Назва	Для введення назви інгредієнту
Опис	Для введення опису де можна використати цей інгредієнт, або вказати ймовірну його заміну

З метою ідентифікації інгредієнту, зокрема для його видалення і т.д. було введено штучний ідентифікатор IngredientId.

Діаграма моделі “Ігредієнт” для бази даних буде мати вигляд. приведений на рисунку 1.2.4.1

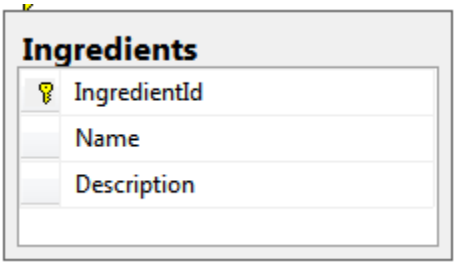

Рисунок 1.2.4.1 - Сутність “Інгредієнт”

Сутність “Рецепт” повиненна мати наступні поля:

Таблиця 1.2.4.2- Поля сутності “Рецепт”

Назва	Для введення назви рецепту
Порція	Для внесення певної кількості інгредієнту (в одиницях виміру)
Опис	Для детальної інформації про спосіб приготування страви
Дата створення	Для пошуку в хронологічній послідовності

З метою ідентифікації рецепту, зокрема для його редагування, видалення, і пошуку та зберігання, було введено штучний ідентифікатор RecipeId.

Оскільки було введено сутність “Користувач”, призначену для ідентифікації власника було додано ключ UserId.

Отже, діаграма моделі “Рецепт” для бази даних буде мати вигляд, приведений на рисунку 1.2.4.2

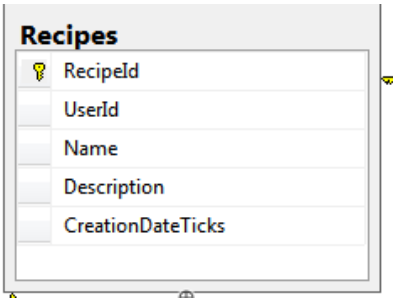

Рисунок 1.2.4.2 - Сутність “Рецепт”

Також було виявлено сутність “Порції”, яка має приналежність до сутності “Інгредієнт” та “Рецепт” і також містить додаткову інформацію про кількість інгредієнту в рецепті.

Сутність “Порція” повинна мати поле “Вага” для задання кількості інгредієнту в одиниці виміру для однієї порції.

З метою ідентифікації інгредієнту створено ідентифікатор IngredientId. Також створено штучні ключі RecipeId та IngredientId для зв’язування порції із рецептом та інгредієнтом.

Діаграма моделі “Порція” для бази даних буде мати вигляд, приведений на рисунку 1.2.4.3

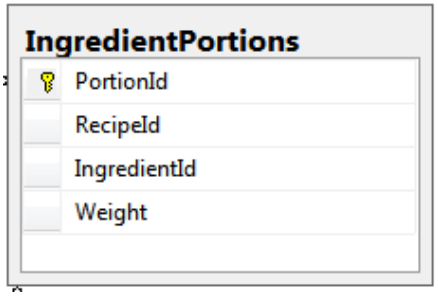

Рисунок 1.2.4.3 - Сутність “ Порція ”

Оскільки було виокремлено сутність “Користувач” з метою ідентифікації власників рецептів, вона повинна містити наступні поля:

Таблиця 1.2.4.3- Поля сутності “Користувач”

Ім’я користувача	Для введення логіну користувача
Пошта	Для підтвердження реєстрації користувача
Пароль	Для захисту аканта користувача

Також вирішено внести додаткові поля, які міститимуть інформацію про номер користувача та двофакторну авторизацію.

З метою ідентифікації користувача створено штучний ідентифікатор UserId. Діаграма моделі “Користувач” для бази даних буде мати вигляд, приведений на рисунку 1.2.4.4

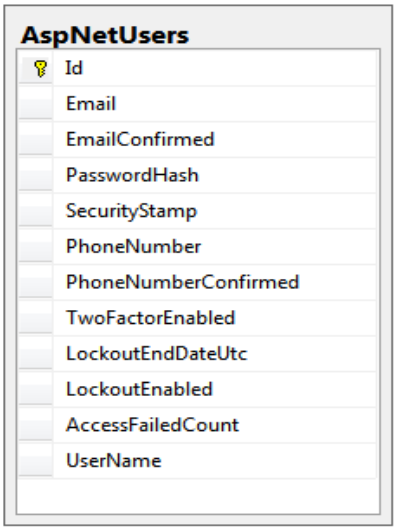

Рисунок 1.2.4.4 - Сутність “Користувач”

Сутність “Користувач” відноситься до сутності “Рецепт” як один до багатьох. Це дозволяє користувачу мати довільну кількість рецептів. Ця залежність реалізована через наявність ключа UserId, який знаходиться в сутності “Рецепт”, та вказує на користувача, який створив цей рецепт.

На рисунку 1.2.4.5 зображено загальну схему бази даних даного web-сервісу

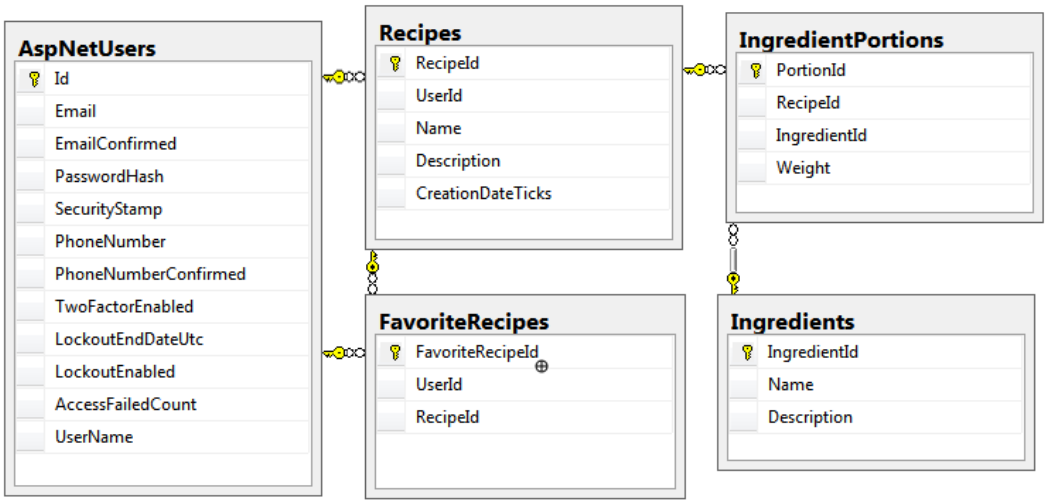

Рисунок 1.2.4.5- Схема бази даних

1.2.5 Проектування програмного інтерфейсу web-сервісу

Згідно принципу REST всі функції взаємодії з даними (створення, читання, редагування та видалення інформації) доступні за допомогою відповідних HTTP методів: GET, POST, PUT, DELETE.

Для кожного з модулів (модуль користувачів, модуль рецептів, модуль інгредієнтів) було складено наступні шаблони UR-шляхів для методів GET, POST, PUT, DELETE.

Таблиця 1.2.4.3- URL для взаємодії із web-сервісом

Метод	Url приклад	Дія	Дані запиту	Дані відповіді
GET	/api/recipes	Get all recipes		JSON
GET	/api/recipes?take=m&scip=n	Get m recipes and ignore n recipes	take, skip	JSON
GET	/api/recipes/{recipeId}	Get recipe	recipeId	JSON
GET	/api/users/{userId}/recipes	Get all recipes from user	userid	JSON

GET	/api/users/{userId}/recipes?take=m&skip=n	Get m recipes from user, ignore n recipes	userId, take, skip	JSON
GET	/api/recipes/{recipeId}/feedbacks	Get all feedbacks from recipe	recipeId	JSON
GET	/api/recipes/{recipeId}/feedbacks?take=m&skip=n	Get m feedbacks from recipe, ignore n feedbacks	recipeId, take, skip	JSON
GET	/api/ingredients	Get all ingredients		JSON
GET	/api/ingredients/{indredientId}	Get ingredient	ingredientId	JSON
GET	/api/users	Get all users		JSON
GET	/api/users/{userId}	Get user	userId	JSON
POST	/api/recipes	Create recipe	JSON	JSON
POST	/api/ingredients	Create ingredient	JSON	JSON
POST	/api/recipes/{recipeId}/feedbacks	Create feedback	JSON	JSON
POST	/api/users	Create user	JSON	JSON
PUT	/api/recipes/{recipeId}	Edit recipe	recipeId, JSON	JSON
PUT	/api/ingredients/{ingredientId}	Edit ingredient	ingredientId, JSON	JSON
PUT	/api/recipes/{recipeId}/feedbacks/{feedbackId}	Edit feedback	recipeId, feedbackId, JSON	JSON
PUT	/api/users/{userId}	Edit user	userId, JSON	JSON
DELETE	/api/recipes/{recipeId}	Delete recipe	recipeId	
DELETE	/api/users/{userId}	Delete user	userId	
DELETE	/api/ingredients/{ingredientId}	Delete ingredient	ingredientId	
DELETE	/api/recipes/{recipeId}/feedbacks/{feedbackId}	Delete feedback	feedbackId	

Рисунок1.2.5.1- Блок- схема запиту на редагування рецепту

1.2.6 Проектування програмного інтерфейсу клієнтської частини

З метою тестування роботи сервісу та подальшого спрощення взаємодії з сервісом клієнтських частин, побудованих з використанням JavaScript, було прийнято рішення реалізувати JavaScript сервіси, які здатні взаємодіяти з сервісом через REST інтерфейс. Даний підхід позбавить розробників клієнтських систем від необхідності прямої роботи з методами сервісу шляхом використання URL.

Проектування програмного інтерфейсу клієнтської частини виконане з використанням підходу ООП. Весь функціонал сервісу поділений на логічні модулі (сервіси), які представлені об’єктами JavaScript. Ці сервіси відповідають за передачу даних з клієнтської частини на серверну та в зворотньому напрямку.

Використання сервісів дозволяє приховати спосіб взаємодії клієнтської частини з серверною.

Згідно інтерфейсу web-сервісу було спроектовано наступні клієнтські сервіси: RecipeService, IngradientService, RegisterService, AuthorizeService.

RecipeService має наступні методи:

- GetAllRecipesAsync – отримувати всі рецепти;
- GetRecipesAsync – отримувати певну кількість рецептів;
- GetRecipeAsync – отримувати вибраний рецепт;
- GetAllFavoriteRecipesAsync – отримувати всі улюблені рецепти;
- AddToFavoriteAsync – додавати рецепт до улюблених;
- RemoveFromFavoriteAsync – видалити рецепт з улюблених;
- GetFavoriteRecipesAsync – отримати певну кількість рецептів;
- GetUsersRecipeAsync – отримати рецепти які належать поточному користуввачу;
- CreateRecipeAsync – створювати рецепт;
- EditRecipeAsync – редагувати рецепт;
- DeleteRecipeAsync – видаляти рецепти.

IngradientService має наступні методи:

- CreateIngredientAsync – створювати інгреієнту;
- GetIngredientsAsync – отримувати інгредієнти ;
- EditIngredientAsync – редагувати інгредієнту;
- DeleteIngredientAsync – видаляти інгредієнту.

RegisterService має наступні методи:

- RegisterAsync – реєструвати користувача.

AuthorizeService має наступні методи:

- GetTokenAsync – отримувати токену доступу;
- AutorizeAsync – авторизувати користувача;

- ValidateTokenAsync – валідувати токен;
- IsAuthorizedAsync – перевіряти авторизацію в системі;
- AuthorizeRequest – дозволяє авторизувати будь-який запит, який відправляється на сервер шляхом підписання його токеном доступу.

Кожен з методів вище наведених сервісів приймає параметр handler, який являє собою функцію з наступною сигнатурою: function (args). Args – це в свою чергу об’єкт, який містить поля result, data. Result набуває значень true або false в залежності від того, чи був надісланий запит оброблений сервером успішно, чи ні. Handler призначений для виклику з отриманими з сервера параметрами після того як запит буде оброблений web-сервером.

1.2.7 Проектування користувацького інтерфейсу клієнської частини

Сайт скаладається з сторінок “Рецепти”, “Улюбленні рецепти”, “Мої рецепти” та “Панель адміністратора”.Для незареєстрованого користувача доступна лише сторінка “Рецепти”.

Для зареєстрованого користувача доступні всі наявні сторінки.

Ввійти на сайт чи зареєструватись користувач може за допомогою кнопки “Log in” у верхньому правому кутку сторінки, вказавши логін та пароль для авторизації, або ж логін, пароль та e-mail для реєстрації.

Для вже авторизованого користувача наявна кнопка “Log out”.

На сторінці “Рецети” відображені всі існуючі рецепти. Якщо користувач не авторизований, він має можливість бачити всі рецепти з описом та інгредієнтами.

Авторизований користувач додатково бачить кнопки “Додати до улюблених” або ж “Видалити з улюблених”, наявні для кожного рецепту в залежності від того, чи рецепт вже доданий в улюблені, чи ні.

Сторінка “Улюблені рецепти” доступна лише авторизованим користувачам. Користувач бачить всі улюблені рецепти, для кожного з цих рецептів наявна кнопка “Видалити з улюблених”.

На сторінці “Мої рецепти” доступні всі рецепти, створенні поточним користувачем, з можливістю їх видалення за допомогою кнопки “Видалити рецепт”, наявної в кожному рецепті. Також наявна форма для створення рецептів, яка активується при натисканні кнопки “Створити рецепт” та деактивується при натисканні кнопки “Вихід”. На формі створення рецептів існують поля для назви та опису рецепту, а також елементи, призначені для вибору інгредієнтів та створення порцій з обраних інгредієнтів.

Сторінка “Панель адміністратора” містить форму, яка дає можливість створювати новий інгредієнт.

1.3 Реалізація та тестування

1.3.1 Реалізація функціоналу

Для реалізацію функціоналу даного web-сервісу прийнято рішення використати мову програмування C#.

C# – це об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET. Синтаксис C# близький до C++ і Java. Мова має строгу статичну типізацію, підтримку поліморфізму, перевантажені оператори, події, вказівники на функції класів, атрибути, та властивості[15].

Спроектувавши web-сервіс та вибравши мову програмування C#, також було вибрано технологію створення web-додатків ASP.NET WebAPI. Web API представляє собою інтерфейс прикладного програмування (API) для будь-якого web-сервера або web-браузера. Ця концепція web-розробки, як правило, обмежується на стороні клієнта web-додатками і, таким чином, не включає в себе web-сервер.

Весь фукціонал web-сервісу побудований завдяки контролерам. Контролери в Web API приймають запит у вигляді об'єкта HttpRequestMessage, обробляють його за допомогою одного з методів і посилають у відповідь результат обробки у вигляді об'єкта HttpResponseMessage.

Для розмежування дій з ресурсами на рівні HTTP-методів існують такі варіанти[14]:

- GET - отримання ресурсу;
- POST - створення ресурсу;
- PUT - оновлення ресурсу;
- DELETE - видалення ресурсу.

Клас ApiController, який є основою контролерів API, дізнається з маршруту, який контролер повинен обробляти запит, і використовує метод HTTP для пошуку

відповідних дій. Методи дій контролерів API повинні містити приставку з іменем методу-дії, який вони підтримують, і тип моделі, з яким вони працюють.

В даному проекті створено такі контролери:

- BaseApiController;
- RecipesController;
- IngredientsController;
- AccountsController;
- FeedbacksController.

Контролер BaseApiController створений як базовий контролер. Він наслідується від основного класу ApiController. Також він є базовим для інших контролерів.

Даний контролер містить чотири основні властивості:

- UserManager – отримує запит зі створеним користувачем;
- RoleManager – отриує запит із ролюю створеному користувачеві;
- CurrentUser – перевіряє користувача на існування;
- DbContext – створює новий екземпляр контексту з використанням правил для створення імені бази даних.

Код роботи даного контролера представлено нижче.

```
{
protected ApplicationUserManager UserManage
{
get { return
Request.GetOwinContext().GetUserManager<ApplicationUserManager>(); }
}
protected ApplicationRoleManager RoleManager
{
get { return
Request.GetOwinContext().GetUserManager<ApplicationRoleManager>(); }
}
public ApplicationUser CurrentUser
{
get
```

```
        {
        var username =
Request.GetRequestContext().Principal.Identity.Name;
        ApplicationUser user = UserManager.FindByName(username);
        return user;
        }
        }
        private readonly TastyThingsDbContext _dbContext = new
TastyThingsDbContext();
        protected TastyThingsDbContext DbContext
        {
        get { return _dbContext; }
        }
        }
        }
```

Для роботи із рецептами реалізовано контролер RecipesController. Він містить такі основні методи:

- CreateRecipeAsync - дозволяє створювати новий рецепт;
- AddRecipeToFavoritesAsync - дозволяє додававати рецепти до улюблених;
- DeleteRecipeFromFavoritesAsync - дозволяє видаляти рецепти з улюбленх;
- GetRecipesAsync - дозволяє отримувати всі рецепти;
- CreateViewModelsAsync - створює модель відображення рецептів;
- GetFavoriteRecipesAsync- дозволяє отримувати всі улюбленні рецепти;
- GetUsersRecipesAsync - дозволяє отримувати користувачу тільки його рецепти;
- DeleteRecipe - дозволяє видаляти рецепти.

Нижче наведенокод створення рецептів:

```
        public async Task<IHttpActionResult>
        CreateRecipeAsync(CreateRecipeBindingModel model)
        {
        if (!ModelState.IsValid)
```

```
{
return BadRequest();
}

var user = CurrentUser;
var recipe = new Recipe()
{
UserId = user.Id,
Name = model.Name,
Description = model.Description
};
var ingredients = DbContext.Ingredients;
recipe.IngredientPortions = (from ingredientPortionBindingModel
in model.IngredientPortions where ingredients.Any(ingredient =>
ingredient.IngredientId ==
ingredientPortionBindingModel.IngredientId)join ingredient in
ingredients on ingredientPortionBindingModel.IngredientId
equals ingredient.IngredientId select new IngredientPortion
{
Ingredient = ingredient,
IngredientId = ingredient.IngredientId,
RecipeId = recipe.RecipeId,
Weight = ingredientPortionBindingModel.Weight
}).ToArray();
DbContext.Recipes.Add(recipe);
await DbContext.SaveChangesAsync();
var message = string.Format("{1}: Recipe created: {0}",
recipe.RecipeId, user.UserName);
return Ok(message);
}
```

Також для роботи з інгредієнтами створено контролер IngredientsController. Його основними методами є:

- CreateIngredient – доволяє створити інгредієнт;
- GetAllIngredients – дозволяє отримати всі існуючі інгредієнти.

Нижче наведено код створення інгредієнту:

```
public async Task<IHttpActionResult>
CreateIngredient(CreateIngredientBindingModel model)
{
```

```
if (ModelState.IsValid)
{
var context = new TastyThingsDbContext();
var ingredient = new Ingredient() {Name = model.Name,
Description = model.Description};
context.Ingredients.Add(ingredient);
await context.SaveChangesAsync();
return Ok(ingredient);
}
else
{
return BadRequest(ModelState);}}
```

Для коректної роботи із користувачами створено контролер AccountsController. Його основним методом є Register, який дозволяє створювати нових користувачів та валідувати токен доступу.

Авторизація користувачів відбувається за допомогою OAuth 2.0.

OAuth являє собою відкритий стандарт авторизації, який надає користувачам відкритий доступ до приватних даних, попередньо збереженим на одних сайтах іншим, без вводу логіна та паролю користувача.

Він надає користувачам токени доступу до даних, що розміщенні на сервісах. Кожен токен надає конкретний доступ до ресурсів, конкретним користувачам на певний термін За допомогою цього, користувачі дають доступ третім сайтам до інформації, не передаючи логіну та паролю [16].

Код налаштування OAuth зображено нижче:

```
{
private void ConfigureOAuth(IAppBuilder builder)
{
builder.CreatePerOwinContext(ApplicationDbContext.Create);
builder.CreatePerOwinContext<ApplicationUserManager>(ApplicationUserM
anager.Create);
builder.CreatePerOwinContext<ApplicationRoleManager>(ApplicationRoleM
anager.Create);
OAuthBearerOptions = new OAuthBearerAuthenticationOptions();
OAuthServerOptions = new OAuthAuthorizationServerOptions()
{
AllowInsecureHttp = true,
```

```
TokenEndpointPath = new PathString("/oauth/token"),
AccessTokenExpireTimeSpan = TimeSpan.FromDays(1),
Provider = new ApplicationOAuthProvider(),
};

builder.UseOAuthAuthorizationServer(OAuthServerOptions);
builder.UseOAuthBearerAuthentication(OAuthBearerOptions);
}
public OAuthAuthorizationServerOptions OAuthServerOptions {
get; set; }
public OAuthBearerAuthenticationOptions OAuthBearerOptions {
get; set; }
}
}
```

Як додатковий функціонал на етапі проектування було додано залишення коментарів. Для реалізації даного функціоналу створено контролер FeedbacksController, основними методами якого є:

- CreateFeedbacks – дозволяє злишати користувачу коментарі під рецептами;
- DeleteFeedbacks – дозволяє видаляти коментар.

Нижче наведено один із цих методів, а саме CreateFeedbacks:

```
public async Task<IHttpActionResult>
CreateFeedbacks(CreateFeedbackBindingModel model)
{
if (ModelState.IsValid)
{
var context = new TastyThingsDbContext();
bool isRecipeExisted = await context.Recipes.AnyAsync(x =>
x.RecipeId == model.RecipeId);
if (!isRecipeExisted)
{
return BadRequest();
}
var feedback = new Feedback()
{
RecipeId = model.RecipeId,
Rate = model.Rate,
Title = model.Title,
```

```
Text = model.Text
};
context.Feedbacks.Add(feedback);
await context.SaveChangesAsync();
return Ok(feedback);
}
else
{
return BadRequest(ModelState);
```

1.3.2 Реалізація бази даних та взаємодії з сервером

Існує безліч способів взаємодії з базами даних на мові C# (наприклад NHibernate або LINQ to SQL).

За допомогою NHibernate ми матимемо змогу об’єкти бізнес-логіки відбражати як реляційну базу даних. Він автоматично створює запити по опису сутностей. У LINQ to SQL модель база даних співставляється з об’єктною моделлю, яка виражена в мові програмування розробників. При запуску програми LINQ to SQL перетворює запити LINQ з об'єктної моделі в SQL і відправляє їх в базу даних для виконання. Повертаючи результати, LINQ to SQL перетворює їх назад в об'єкти, з якими можна працювати на власній мові програмування.

На основі проекту web-сервісу обрано технологію створення Entity Framework. Оскільки вона є доволі зручною, та простою у використанні.

Цу об'єктно-реляційний модуль, який за допомогою спеціалізованих об'єктів взаємодіє з даними. Це зменшує необхідність написання коду доступу. [3].

Дана технологія є спеціальною об'єктно-орієнтовану технологію на базі фреймворка .NET для роботи з даними.

За допомогою Entity Framewor ви можете такими способами взаємодіяти з базою:

- Database first
- Model first
- Code first [1].

Entity Framework пропонує зручний дизайнер, величезну кількість варіантів маппінгу, автогенерацію класів-моделей.

Для даного проекту обрано спосіб взаємодії із бд Code first, так як він є простим та зручним у користуванні, дозволяє з легкістю створювати класи-моделі на основі яких фреймворк автоматично створить бд.

DbContext є основним класом, який відповідає за взаємодію з даними як об'єктами. DbContext включи такі види діяльності:

- EntitySet: DbContext містить набір сутностей (DbSet <TEntity>) для всіх об'єктів, які відображаються в таблицях бази даних;
- автоматична генерація запитів: DbContext перетворює LINQ- to -Entities запити в SQL запити і транслює їх в базу даних;
- відстеження змін: відстежує зміни, які були внесені в об’єкти, які вже були зв’язані з контекстом бази даних;
- маніпуляції даними: дозволяє проводити операції вставки, оновлення та видалення в базі даних, на підставі станів сутностей;
- конвертація даних: DbContext перетворює вихідні дані таблиці в об'єкти сутностей.

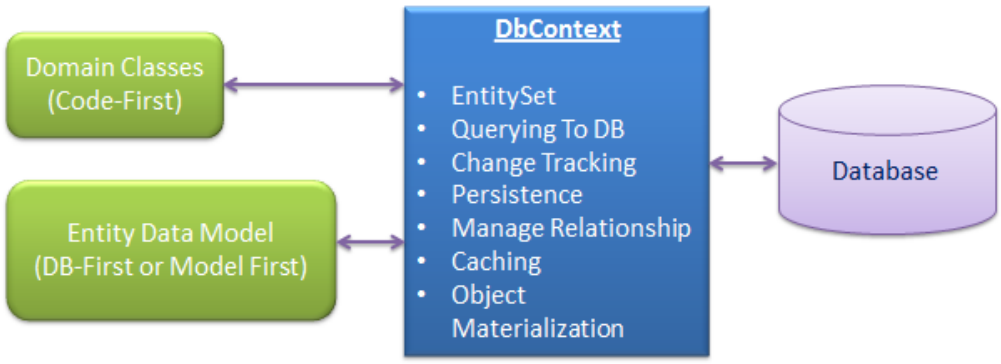

Рисунок 3.2.1- Схематичне зображення взаємодії доменів або класів із БД

Збереження бази даних відбувається у системі керування базами даних Microsoft SQL Server Expres.

Microsoft SQL Server — комерційна система керування базами даних, що розповсюджується корпорацією Microsoft. Мова, що використовується для запитів — Transact-SQL, створена спільно Microsoft та Sybase. Transact-SQL є реалізацією стандарту ANSI/ISO щодо структурованої мови запитів (SQL) із розширеннями. Використовується як для невеликих і середніх за розміром баз даних, так і для великих баз даних масштабу підприємства[18].

Код підключення до даних системи наведений нижче:

```
<connectionStrings>
<add name="TastyThingsDbContext"
providerName="System.Data.SqlClient"
connectionString="Data Source=.\SQLExpress; Integrated
Security=true; User Instance=true;" />
</connectionStrings>
```

Створення основних таблиць за допомогою Code-First реалізовано наступним чином:

```
public class TastyThingsDbContext : DbContext
{
public TastyThingsDbContext()
base("TastyThingsDbContext")
{ }
public DbSet<Recipe> Recipes { get; set; }
public DbSet<Ingredient> Ingredients { get; set; }
public DbSet<Feedback> Feedbacks { get; set; }
public DbSet<FavoriteRecipe> FavoriteRecipes { get; set; }
}
```

1.3.3 Реалізаці користувацької частини

Для тестування взаємодії web-сервісу із користувачами створено клієнтську частину, у вигляді сату. Основним завданням даного сайту є показати як працює даний web-сервіс.

Клієнська частина реалізована за допомогою наступних технологій: AngularJS, jQuery Ajax, Bootstrap 3.

AngularJS . – розширяє браузернки на основі Модель-Вид-Контролер (MVC), та спрощує їх тестування та розробку.

jQuery дозволяє створювати абстракції для низькорівневої взаємодії та створювати анімацію для ефектів високого рівня. Це сприяє створенню динамічних та потужних web-сторінок.

AJAX (Asynchronous JavaScript And XML) це сукупність методів веб-розробки, що використовують багато веб-технологій на стороні клієнта для створення асинхронних веб-додатків.

Bootstrap — створений для полегшення розробки web застосунків та сайтів. Він включає в себе CSS та HTML для типографії, кнопок, форм, сіток таблиць, , навігації тощ.

Сайт скаладається з сторінок “Рецепти”, “ Улюбленні рецепти”, “Мої рецепти” та “Панель адміністратора”.

Для незареєстрованого користувача доступна лише сторінка “Рецепти” (рисунок 1.3.3.1). Написнувши на “Інгредієнти” користувач зможе перегляну необхідні для приготування інгрудєнти та їхні порції, а також натиснувши на “Відкрити рецепт” зможе переглянути детальний опис та спосіб приготування.

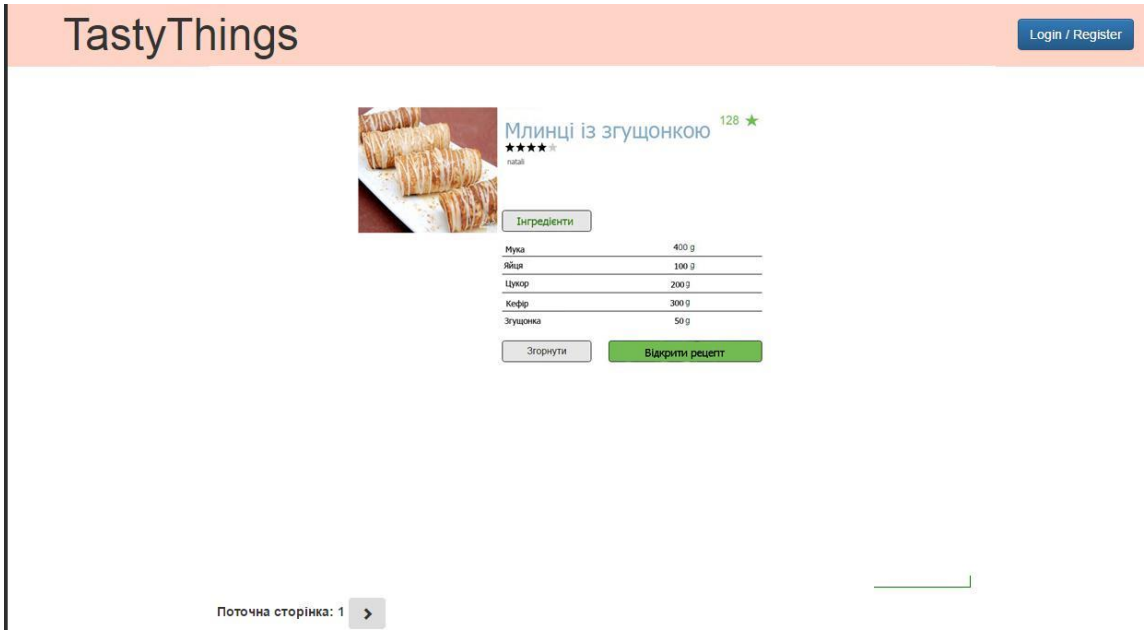

Рисунок 1.3.3.1 – Головна сторінка “Рецепти”

Для зареєстрованого користувача доступні додаткові сторінки, такі як: “Улюбленні рецепти” та “Мої рецепти”. Також для адміністратора наявна додаткова сторінка “Панель адміністратора” (див. рисунок 1.3.3.2)

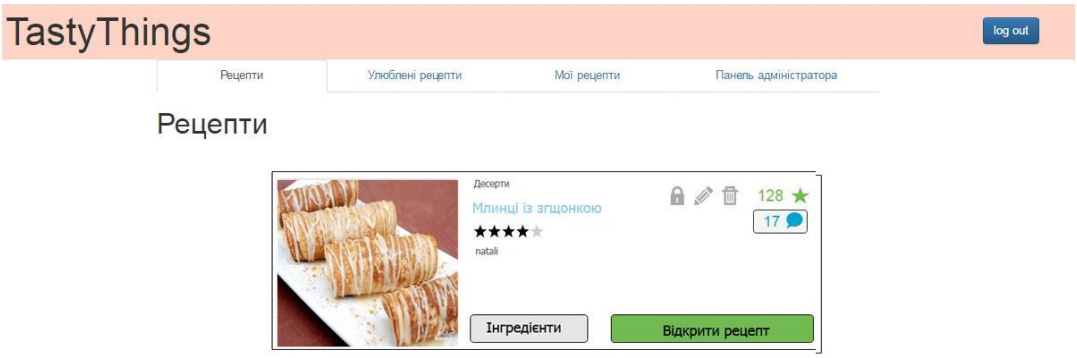

Рисунок 1.3.3.2- Головна сторінка “Рецепти” для авторизованого користувача

Ввійти на сайт чи зареєструватись користувач може за допомогою кнопки “Log in” у верхньому правому кутку сторінки, вказавши логін та пароль для авторизації, або ж логін, пароль та e-mail для реєстрації.(рисунок 1.3.3.3та 1.3.3.4).

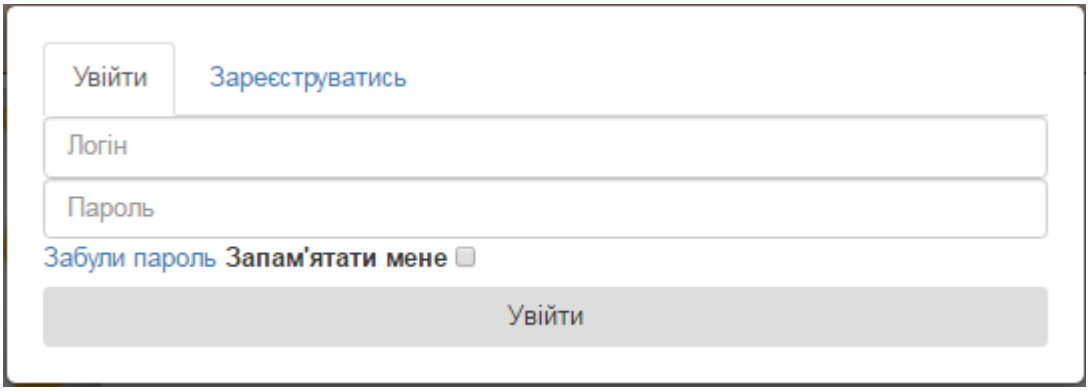

Рисунок 1.3.3.3– Панель авторизації

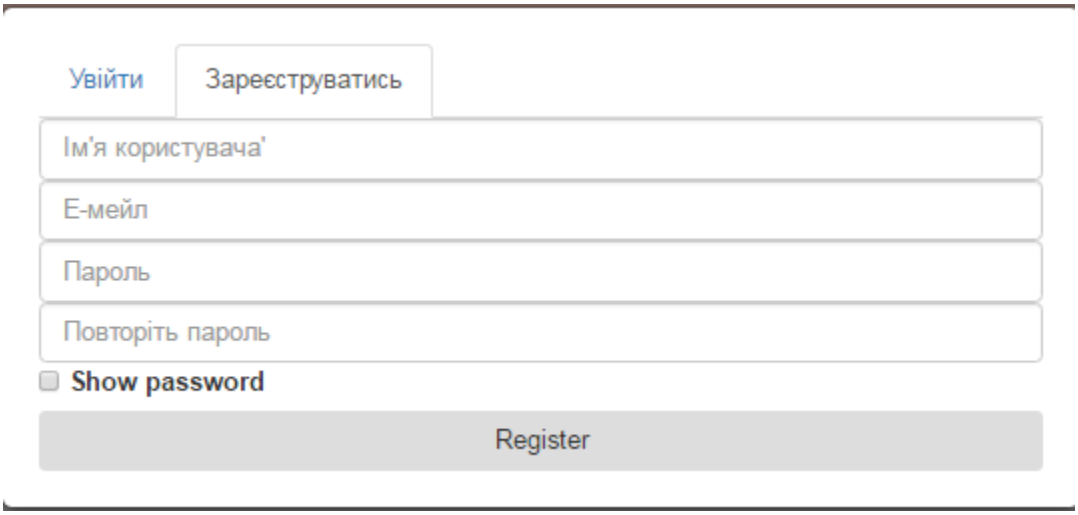

Рисунок 1.3.3.4– Панель реєстрації

На сторінці “Рецепти” відображені всі існуючі рецепти. Також авторизований користувач додатково бачить кнопки “Додати до улюблених” або ж “Видалити із улюблених”, наявні для кожного рецепту в залежності від того, чи рецепт вже доданий в улюблені, чи ні (рисунок 1.3.3.5).

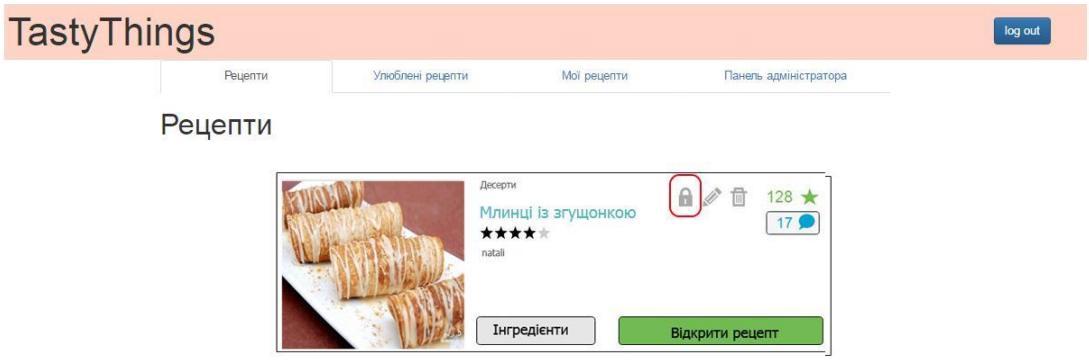

Рисунок 1.3.3.5– Кнопка “Додати до улюблених”

На сторінці “Улюблені рецепти” користувач бачить всі рецепти, які він позначив як улюбленні, для кожного з цих рецептів наявна кнопка “Видалити з улюблених”(рисунок 1.3.3.6).

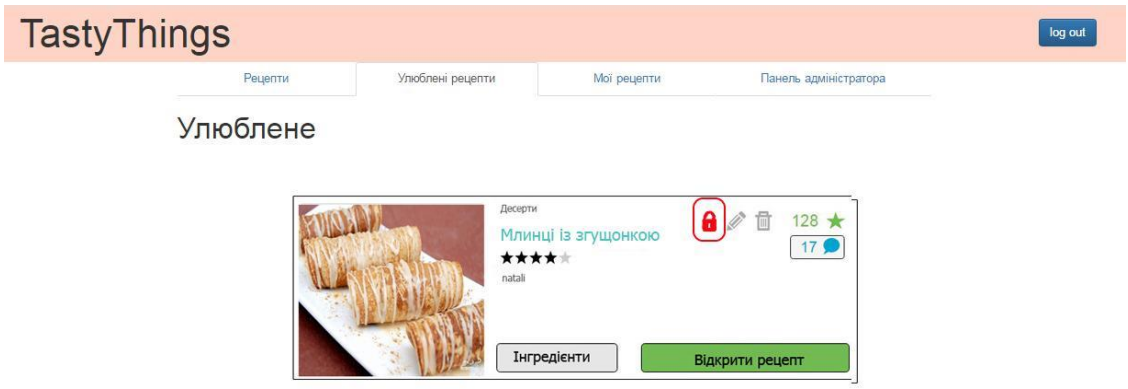

Рисунок 1.3.3.6 – Кнопка “Видалити з улюблених”

Сторінка “Мої рецепти” дає можливість переглядати рецепти, створенні поточним користувачем, додавати до улюблених, вносити до них зміни та видаляти рецепти (рисунок 1.3.3.7).

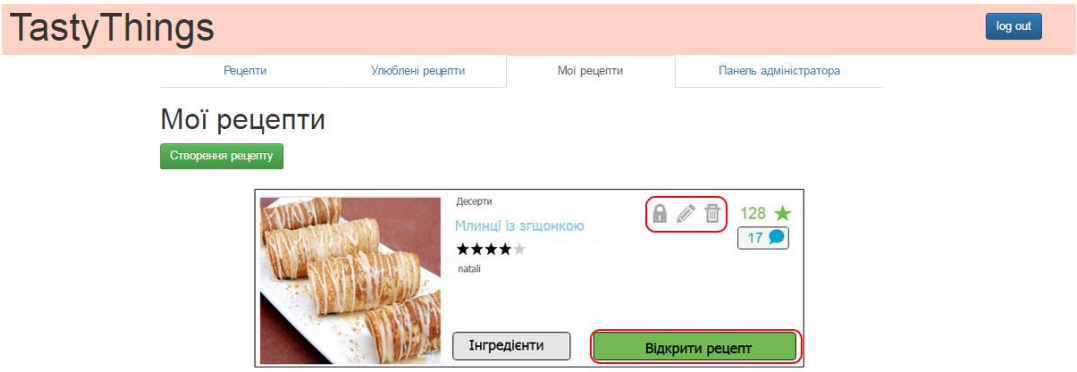

Рисунок. 1.3.3.7– Зображення доступних кнопок

Також наявна форма для створення рецептів, яка активується при натисканні кнопки “Створити рецепт” та деактивується при натисканні кнопки “Вихід”. На формі створення рецептів існують поля для назви та опису рецепту, а також елементи, призначені для вибору інгредієнтів та створення порцій з обраних інгредієнтів (рисунки 1.3.3.8 та 1.3.3.9).

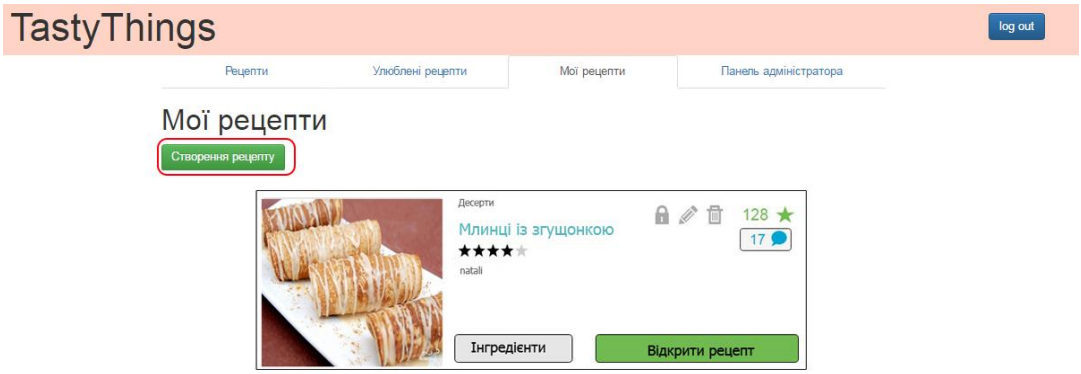

Рисунок 1.3.3.8– Кнопка “Створити рецепт”

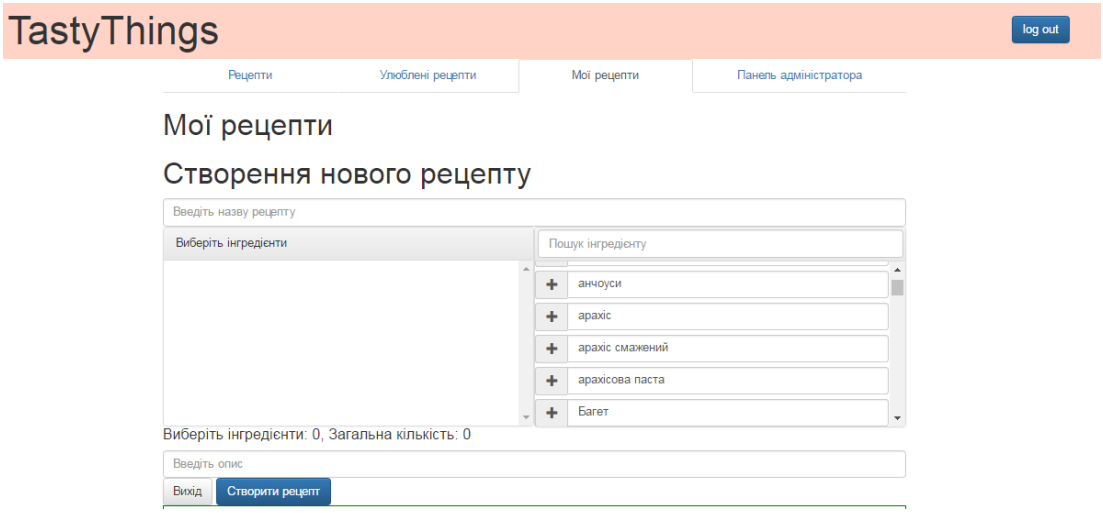

Рисунок 1.3.3.9– Панель створення рецептів

Також на рецепті наявна кнопка, яка дозволяє користувачам переглянути коментарі, залишені іншими відвідувачами (рисунок 1.3.3.10 та 1.3.3.11)

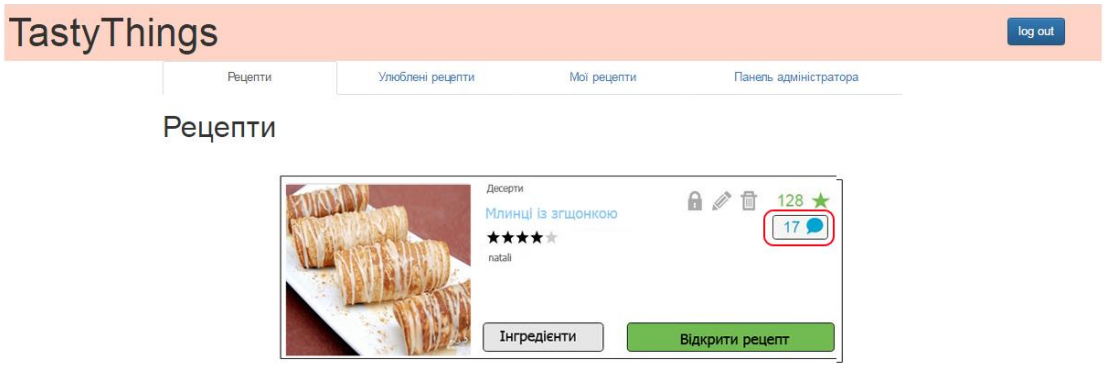

Рисунок 1.3.3.10– Кнопка “Переглянути коментарі”

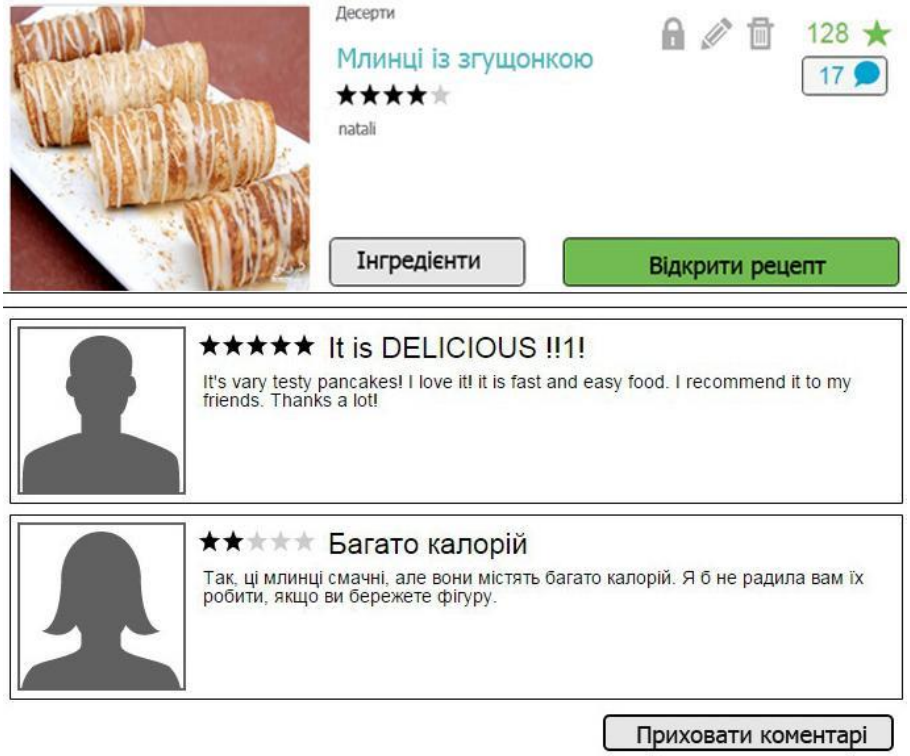

Рисунок 1.3.3.11– Панель перегляду коментарів

Сторінка “Admin panel” містить форму, яка дає можливість створювати новий інгредієнт (рисунок 1.3.3.12).

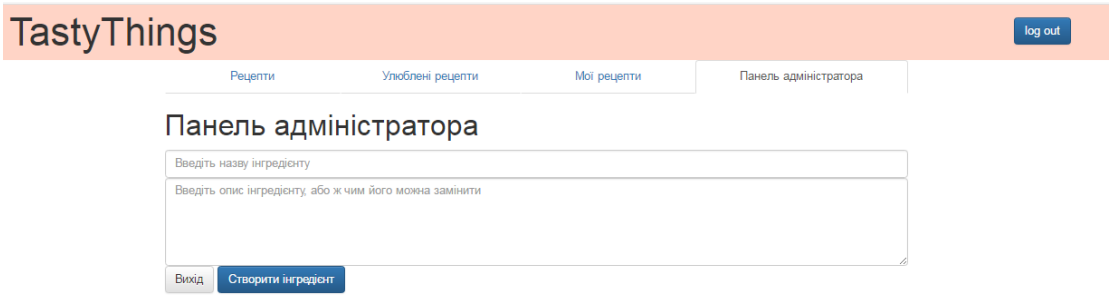

Рисунок 1.3.3.12– Панель створення інгредієнтів

1.3.4Тестування ІС

Тестування даного web-сервісу відбувалось за допомогою програмного забезпечення Fiddler.

Fiddler - це проксі, який працює з трафіком між вашим комп'ютером і віддаленим сервером, і дозволяє переглядати і змінювати його.

Можливості:

– у вікні запитів зліва можна подивитись або вибрати запити, дивитися їх заголовки, зберігати їх на диск всі разом або окремо.

– AutoResponder - дозволяє підставити свій файл замість сервера.

– Composer - дозволяє скласти запит на сервер вручну.

– Filters - дозволяє призначити дії в залежності від виду запиту. Опції стануть зрозумілі після переходу на вкладку.

Тестування web-сервісу в даному програмному середовищі розглянемо на прикладі методу CreateRecipeAsynk, який дозволяє отримувати запитані рецепти.

Для тестування методу, отримання рецептів, необхідно ввести URL http://localhost:50552/api/recipes?take=10&skip=10. Результатом вірного запиту буде код 200 – ОК – стандартна відповідь при успішному виконанні HTTP запиту. Результат тестування даної URL зображено на рисунку 1.3.4.1.

Рисунок 1.3.4.1– Зображення результату запиту

Також для перевірки правильності запиту міняємо дані у змінних take та skip, які ідповідають за кількість видачі рецептів.

Змінивши дані в take на від'ємні, що є недопустимим, отримаємо помилку. Результат перевірки зображено на рисунку 1.3.4.2.

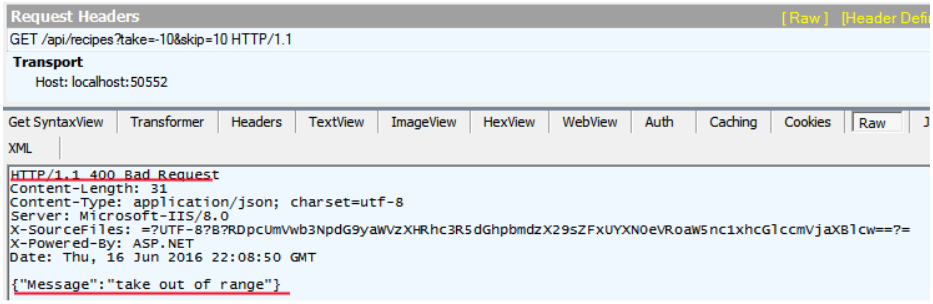

Рисунок 1.3.4.2– Зображення результату перевірки невірного запиту

Також первіримо, чи запит буде вірним, якщо ми не вкажемо всіх необхідних даних. Для цього ми введемо в URL такі дані:

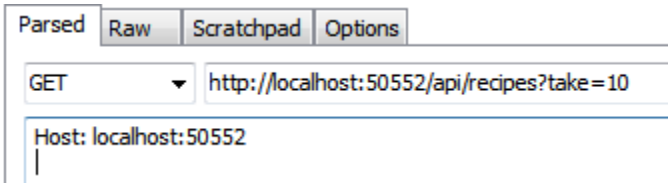

Рисунок1.3.4.13– Зміна даних в URL

Результ введених даних буде вірний, так як пропущення другої змінної є допустимим(див. рисунок 1.3.4.4).

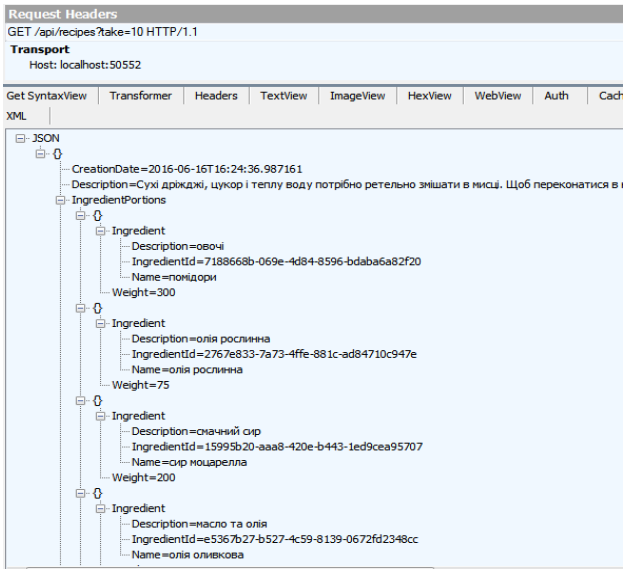

Рисунок 1.3.4.4– Результат перевірки запиту

Але забравши в запиті першу змінну запит буде не вірним (рисунок 1.3.4.5).

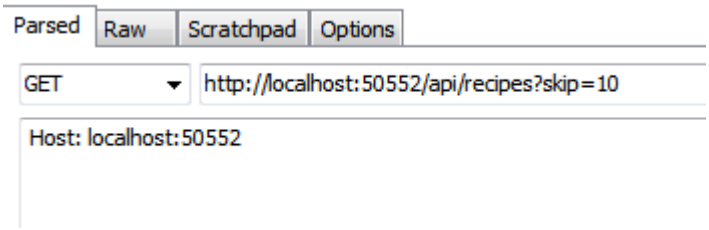

Рисунок 1.3.4.5– Перевірка запиту без змінної skip

В результаті перевірки буде помилка 500 - Внутрішня помилка серверу (рисунок 1.3.4.6).

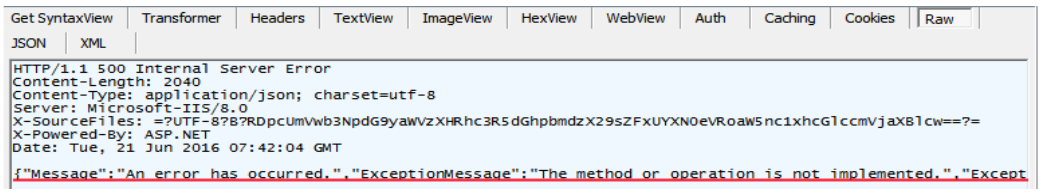

Рисунок 1.3.4.6– Результат перевірки запиту без змінної skip

Також перевіримо чи запит буде вірним не вказавши обох змінних (рисунок 1.3.4.7).

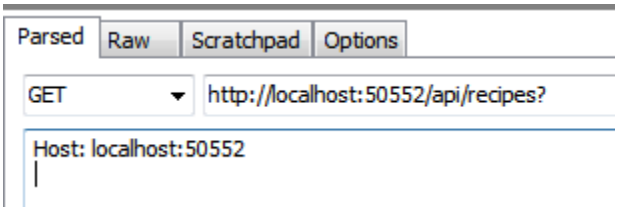

Рисунок 1.3.4.7– Перевірка запиту без змінних

Результат перевірик запиту без змінних зображений на рисунку1.3.4.18.

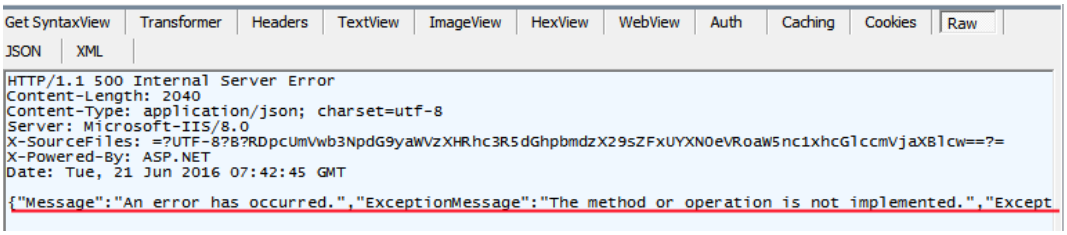

Рисунок 1.3.4.8– Результат запиту без змін

Також перевіримо запит вказавши в одній із змінних різні типи даних або вирази. Перевірка на вираз у змінній take зображена на рисунку 1.3.4.9.

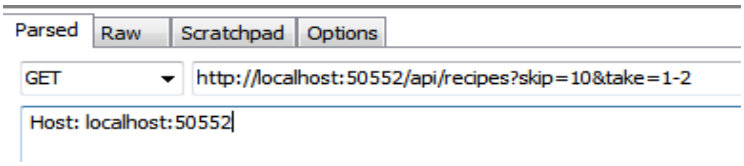

Рисунок 1.3.4.9– Перевірка запиту із виразом

Результат виконання даного запиту зображений на рисунку 1.3.4.10

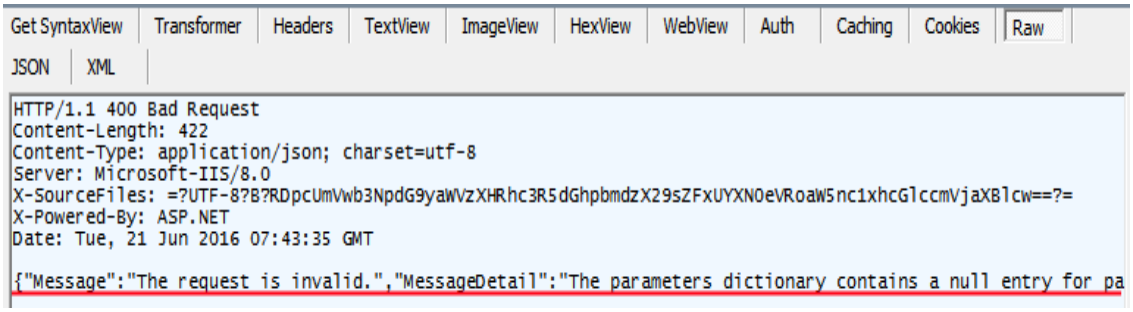

Рисунок 1.3.4.10– Результат запиту із виразом

Також перевіримо присвоєння в зміну пустої стрічки, тобто take=“”(див. рисунок 1.3.4.11)

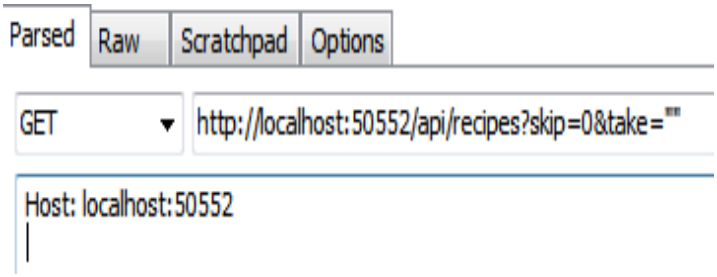

Рисунок 1.3.4.11– Перевірка запиту із take= “”

Результат запиту зображений на рисунку 1.3.4.12

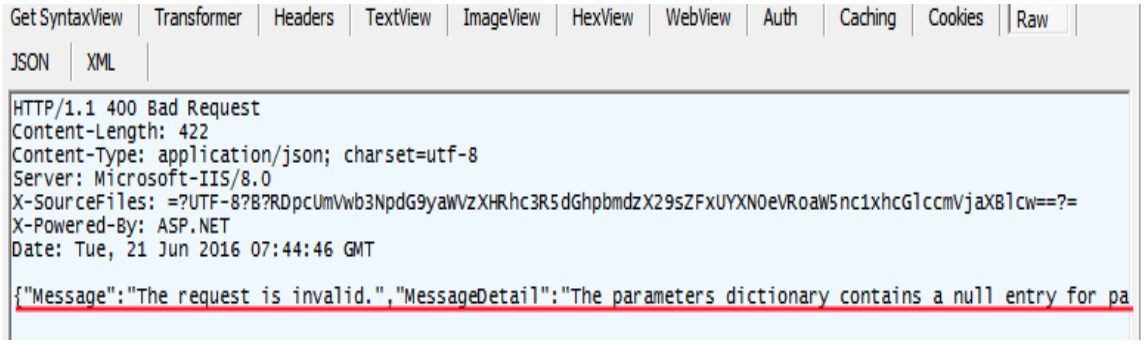

Рисунок 1.3.4.12– Результат запиту із take= “”

Також перевіримо на виконання запиту присвоївши змінній take “0” (рисунок1.3.4.13).

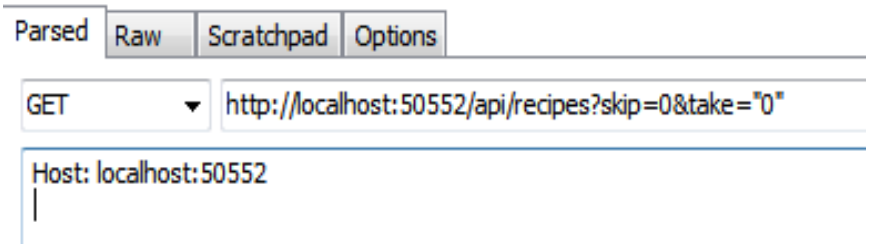

Рисунок 1.3.4.13– Перевірка запиту take=“0”

Результат даного запиту зображений на рисунку 1.3.4.14.

Рисунок 1.3.4.14– Результат запиту із take= “0”

В результаті тестування було виявлено декілька помилок, які одразу ж були виправлені.

2. СПЕЦІАЛЬНА ЧАСТИНА

2.1 Нейронні мережі та їх тлумачення

Одинним із способів реалізації штучного інтелекту (ШІ) є нейронна мережа.

Машинне навчання є однією з найбільших його областей. В разі відсутності чіткого вирішення проблеми, вона взмозі вивчати методи побудови алгоритмів для самостійного навчання. Створення механізму з самонавчанням врази простіше, ніж пошуки правильного рішення[24].

Існують такі поняття, як «глибинне» начання, його можна зустріти у більшості літератури. Іншими словами, машинне навчання, яке використовує велику кількість обчислювальни ресурсів, так звані «нейронні мережі».

Щоб не заплутатися в поняттях «штучний інтелект», «машинне навчання» та «глибоке навчання», розглнемо їхню візуалізацію розвитку:

Імітування нервової системи людини, це нейронна мережа. Вособливості її здатність самососвіти з урахуванням досвіду.

Мережеве навчання це окремі елементи на подобі нейроннів в нашій системі з яких складається мережа. Дані отримані на вході обробляються згідно полідовності поступлення на кожному шарі, не виключаючи можливості змін параметрів залежно від результатів.

Використовуючи існуючі дані, прогнозуючи поведінку прогами в майбутньому, її результати та тенденції, ми використовуємо спосіб аналізу та обробки даних такий як машинне начання. Так зване начання без явного програмування.

Прогнозуючи, машинне начання робить програму чи пристрій більш ефективним. Іншими словами, при покупках чи перегляді того чи іншого товару,

воно підкаже вам інші товари, які можуть сподобатись на основі раніше переглянутих чи придбаних. Під час користування кредитною карткою, машинне навчання може виявити зловмисників, перевіривши транзакцію. Або ж вказати на закінчення прибирання вашої кімнати роботом-пилососом.

Те, які завдання нейронна мережа буде вирішувати, визначається, тим, як вона працює та вчиться. Вона приймає та надає значення змінних які приходять до неї. Іншими словами, нейронну мереже можна використовувати, якщо володієте певною інформацією та бажаєте отримати поки не відому інформацію. Як приклад прогнозування фондового ринку. Знаючи сьогоднішнішню ціну, можна спрогнозуваи яка буде завтра. Або ж надання кредиту, так як потрібно достовірно перевірити інформацію про особу якій надають кредит та визначити всі ризики надання позики.

Коли точність вхідних або вхідних даних відома, з'єднаня моделюються безпосередньо, тому нейронна мережа зачасту використовують, коли ця інформація нам не відома. Також, те що зв'язок між вхідними та вихідними даними знаходиться в процесі вивчання, є однією з важливих особливостей мережі.

Використовуються два типи алгоритмів для навчання нейроннок, а точніше керований та не керований, так звані із вчителем та без.

Підготовка даних про навчання важливий аспект для керованого нейронного навчання, так як приклади входів і результатів є їхніми даними. Між першими та останніми мережа встановлює зв’язок. Ці дані, зазвичай, беруться із історичних данних, тобто на основі попередньо переглянутої інформації,так званого досвіду. Після чого, мережу навчають алгоритмом керованого навчання(метод зворотнього поширення- один з найвідоміших), в якому дані використовуються для мінімізації помилки на навчальному наборі.

Добре навчена мережа, здатна імітувати функцію, яка пов'язує значення вхідних та вихідних даних, з часом таку мережу можна використовувати для прогнозування невідомих значень.

2.2 Класифікація нейронної мережі

Щоб зрозуміти класифікацію за нейронних мереж, важливо дізнатися, як працюють інші алгоритми класифікації та їх сильні сторони. Через велику кількість проблем нейронна мережа може бути непридатною або «надмірна». Для інших це може бути єдиним рішенням.

- Логістична регресія: Аналізує набір точок даних з однією або декількома незалежними змінними (вхідні змінні, які можуть вплинути на результат) і знаходить найкращу модель для опису точок даних
- Алгоритм дерева рішень: Для класифікації точок даних використовується структура дерева з набором правил "якщо-тоді". Правила вивчаються послідовно з даних про навчання. Дерево побудовано зверху вниз; атрибути вгорі дерева мають більший вплив на рішення про класифікацію. Навчальний процес триває до тих пір, поки він не виконає умови припинення.
- Випадковий алгоритм лісу (Random Forest Algorithm): Більш вдосконалена версія дерева рішень, яка стосується перенасичення, використовуючи велику кількість дерев із випадковими варіаціями, а потім вибираючи та агрегуючи найкраще ефективні дерева рішень. "Ліс" - це сукупність дерев, що приймаються рішеннями, як правило, виконані за допомогою техніки, яка називається "випалювання".
- Наївний класифікатор Байєса(Naive Bayes Classifier): Класифікатор на основі ймовірності, заснований на алгоритмі Байєса. Відповідно до концепції залежної ймовірності, вона обчислює ймовірність того, що кожна з особливостей точки даних (вхідні змінні) існує у кожному із цільових класів. Потім вибирається

категорія, для якої ймовірності максимальні. Модель заснована на припущенні (що часто не відповідає дійсності), що ознаки умовно незалежні.

- k-Найближчий сусід (KNN) (k-Nearest Neighbor (KNN)) Класифікує кожну точку даних, аналізуючи найближчих сусідів з навчального набору. Поточній точці даних присвоюється клас, який найчастіше зустрічається серед сусідів. Алгоритм є непараметричним (не передбачає припущень щодо базових даних) і використовує ліниве навчання (не попередньо тренується, усі дані тренінгу використовуються під час класифікації).
- Штучні нейронні мережі та глибокі нейронні мережі (Artificial Neural Networks and Deep Neural Networks) Штучні нейронні мережі побудовані з простих елементів, званих нейронами, які приймають реальну цінність, множать її на вагу і запускають її через нелінійну функцію активації. Побудувавши кілька шарів нейронів, кожен з яких отримує частину вхідних змінних, а потім передає свої результати наступним шарам, мережа може засвоїти дуже складні функції. Теоретично нейронна мережа здатна пізнати форму будь-якої функції, враховуючи достатню обчислювальну потужність.

2.3 Проектування нейронної мережі

Ознайомившись з алгоритмами нейронних мереж, визначивши основну проблематику завдання, було прийнято рішення не писати нейронну мережу самостійно, а використати вже інснуючу. Так як це продуктивніше та якісніше. Згідно завдання яке постало та вимог, було вирішено використати нйронну мережу від Azure, а точніше Azure Mashine Learning.

«Microsoft Azure» — це хмарна платформа корпорації Microsoft, призначена для розробників застосунків хмарних обчислень мета якої спростити процес створення онлайн додатків. Платформа створена з групи трьох технологій, що забезпечують спеціалізований набір можливостей для розробників. Більше того, платформу Windows Azure можна використовувати як в локальних, так і хмпарних додатках[23].

Для того, щоб користувач міг швдко навчати та розгортати моделі машинного навчання, Azure ML надає веб-інтерфейси і пакети SDK. Використовувати дану платформу можна для машинного навчання будь-якого роду - від класичного до глибокого навчання, контрольованого і неконтрольованого. На кожному з етапів підготовки даних, навчання та оцінки моделей, можна працювати спільно з розширеними конвеєрами машинного навчання. Вони дозволяють виконувати наступні завдання:

• Автоматизація повнго циклу процесу навчання;

• Повторне використання компонентів та при необхідності повторного виконання етапів;

• Використання різного роду обчислювальних ресурсів на певному етапі;

• Виконання пакетної оцінки.

Машинне навчання Azure надає веб-інтерфейс, іменований конструктором і кілька пакетів SDK і CLI для швидкої підготовки даних, навчання та розгортання моделей. Завдяки Azure ML ви отримуєте масштабованість, підтримку декількох

платформ, розширені можливості машинного навчання, такі як автоматизоване машинне навчання, і підтримку конвеєра. CLI для Машинного навчання Azure є розширенням кроссплатформенного інтерфейсу командного рядка. Це розширення надає команди для роботи Azure ML, дозволивши автоматизувати його дії. У наступному списку наведені деякі приклади дій, які можна виконати за допомогою розширення CLI.

• виконувати експерименти для створення моделей машинного навчання;

• реєструвати моделі машинного навчання для використання клієнтами;

• упаковувати, розгортати і відслідковувати життєвий цикл моделей машинного навчання[22].

Інтерфейс командного рядка не є заміною пакету SDK для Машинного навчання Azure. Це додатковий інструмент, оптимізований для обробки строго параметрезованих завдань, які чудово підходять для автоматизації.

Взявши за основу мову програмування C# та REST архітектуру ми з легкістю можемо під єднати машинне навчання Azure. Так як при розгортанні моделі Azure ML як веб-служби, створюється інтерфейс REST API. Через цей API ви можете відправляти дані в модель і отримувати від неї прогнози.

2.3.1 Аналіз предметної обласні нейронної мережі

Живучи в настільки сучасному світі, нам завжди хочеться щоб наша робота чим раз тим більше автоматизовувалась. У буь-яких процесах ми шукаємо найшвидше вирішення проблеми. У будь-яких діях ми хочемо чим швидше отримати результат і бути ним задоволеним. Маючи неймовірно великий доступ до інформаці ми можемо в будь-який момент знайти ту чи іншу інформацію. Купити ту чи іншу річ, відвідати різні міста одним кліком на клавіатурі і подивитись фільм створений ще до нашого народження.

Кожен з нас любить смачно поїсти, тому ми завжди в пошуках нових рецептів. Нових смаків, нових вражень. Їжа один із основних факторів задоволення людини. Ми їмо щоб жити. Але вже давно люди перестали їсти лиш щоб вижити, ми почали насолоджуватись їжею. З неймовірно великим доступом до інформації ми можемо знайти будь-який рецепт в інтернеті і спробувати його приготувати. Для цього нам потрібно лиш сходити в магазин або ж замовити достаку їжі на дім та приготувати.

Але що ж робити, якщо ми не можемо знайти те, що нам дійсно подобається. Якщо ми вже спробувати стільки різних страв, що наші смакові рецептори вимагають все більше і більше нового. Звісно, ми можемо експерементувати самостійно і пробувати урізноманітнювати рецепти, створювати власні, розпитувати друзів про їхні кулінарні спроби. Або ж ми просто можемо довіритись комп'ютеру, що робимо щодня, вказати інгредієнти які в нас є, чи які хочемо поєднати і він одразу ж нам видасть рецепт який буде надзвичайно смачним та новим.

Створюючи рецепт самостійно ми витрачаємо багато часу на експеременти, на спроби, на порівнняня пропорцій, щоб в кінцевій точці вийшло те. Що ми дійсно бажали.Але ж якщо ми просто вкажемо бажане програмі. Вона нам мометально видасть те, що ми бажаємо.

Для цього нам всього-лиш необхідно прив'язати нашу програму із вже існуючими рецептами та даними про вподобання людей до нейронної мережі, яка, в свою чергу, опрацювавши всі дані навчиться самостійно їх створювати. Ми можемо навчити її на основі інших сайтів чи сервісів, давши їй велику кількість даних чим покращимо результат вибірки.

Пройшовши по рецептах, переглянувши коменатрі до них, врахувавши побажання людей та вподобання, нейронна мережа може видавати клієнту рецепт, який має високу ймовірність йому сподобатись на основі попередньо переглянувших, чи тих які клієнт попердньо зберіг у своєму кабінеті. Подібно до роботи нейронної мережі від Нетфлікс, клієнт отримувати в запропоновані максимально подібні по інгредієнтах, чи способу приготування рецепти, або ж ті, які подобались іншим клієнтам з подібними смаками.

В кожного з нас можуть бути певні вподобання які відрізяються, чи рекомендації від лікарів, дієти чи алергії. Такі люди перебувають в постійному пошуку рецептів які будуть їм доступні чи дозволені. В теперішній час можна знайти безліч рецептів на будь-який смак та урахування дієт, але ми не завжди можемо дозвлолити собі такий список продуктів і пошуки даних страв займають багато часу. Вкористовувавши нейронну мережу, ми можемо всьоголиш ввести дані з яких інгредієнтів бажаємо створити рецепт, вказати їй зовсім не поєднювальні фрукти чи овочі, вибрати спосіб приготування або ж час і вона в ту ж мить створить власний рецепт.

Так як нейронна мережа не має смакових рецепторів, вона не може з точніст передати пропорцію інгрендієнтів. Але навчившись на попередньо опрацьованих рецептах, взявши до уваги які інгрендієнти в якій пропорції з чим поєднюються вона може наближено створити рецепт та видати його користувачу, вказавши спосіб приготування та порції.

2.3.2 Проектування основних задач нейронної мережі

Основними функціями даного веб сервісу є створення, пошук, додавання, вилучення та перегляд рецептів або ж інгрендієнтів. Користувач ввійшовши в свій профіль може додавати певні рецепти до списку улюблених, створювати власні рецепти додаючи їх до бази даних та переглядати рецепти інших користувачів.

За допомогою додавання нейронної мережі та поєднання її з веб сервісом ми можемо розширити можливості даного проекту. Додавання нового функціоналу пришвидшить результати запитів клієнтів та покращить якість даного продукту. Завдяки розширеним можливостям нейронних мереж ми можемо злегкістю втілювати будь-які наші бажанн в реальність.

Взявши до уваги предметну область та оцінивши можливості було прийнято рішення виділити наступні основні функції які покращать роботу давного веб сервісу:

- Вибірка;
- Порівняння;
- Аналіз бази;
- Видача даних;
- Створення рецепту;

Завдяки REST архітектурі, коли клієнт вводить дані, запит проходить перевірку веб сервісом і викликає певний функціонал. Але в сучасному світі. Нейронні мережі настільки потужні, що злегкістю справляться із задачати набагато якісніше та швидше, ніж людина.Тому навчивши нашу нейронну мережу, ми можемо покращити результати запитів введених користувачем.

Для того щоб нейронна мережа видавала коректні результати, її необхідно навчити. Для самонавчання їй необідні дані, а точніше, наші рецепти та інгредієнти.

Вибірка. Нейронна мережа матиме доступ до бази даних веб сервісу, За допомогою чого, з легкістю отримає всі необхідні матеріали для навчання. Зробивши вибірку із наших даних, система зможе по групувати їх по критеріях, видах та смакових групах. Розділивши рецепти на категорії такі як « салат», «перші страви», «гарніри» і тому подібне, отримування конкретного запиту відбуватиметься якісніше. Також можна групувати інгредієнти по смакових якостях(кисле, солодне, гостре), так і по виду(ягоди, овочі, каші). Так як важливою функцією даної нейронної мережі є створення власного рецепту, важливим буде групування ігнредієнтвів по подібності. А точніше, по смаковим поєднаннях, способу приготування та важливістю продукту (наприклад: картопля та лавровий лист).

Порівняння. Так як клієнт переглядатиме рецепти, оцінюватиме інші та додаватиме їх до розділу улюблені, нейронна мережа на основі цих даних, порівнюючи наявність тих чи інших продукті, зможе злегкістю підказувати користувачу на 60-80% подібні рецепти.

Аналіз бази. Маючи всі дані про рецепт(інгредієнти, кількість, спосіб приготування), нейронка може аналізувати які інгредієнти з чим краще поєднюються, яка кількість того чи іншого продукту краще підходить, як готується той чи інший рецепт. На основі цих даних вона зможе видавати якісні результати.

Видача даних відбуватиметься згідно запитів клієнта. Якщо раніше система за допомогою фільтрів робила вибірку і повертала користувачу результат, тепер нейронна мережа за допомогою власних фільтрів набагато швидше повертатиме потрібні запити.

Створення рецепту. Якщо користувач створив запит і звичні функції програми не можуть дати результату, користувачу буде запропоновано створити новий рецепт. Ввівши інгредієнти які він бажає та фільтри, він надасть запит до нейронної мережі і та на основі попереднього навчання створить власний рецепт.

Даний веб сервіс дієпо даній схемі. Користував вводить дані, веб сервіс перевіряє користувача на авторизацію, після чого перевіряє правильність запиту, а вже потім направляє його опрацюванняя. Згідно до виконання функції, запит отримує дані з бази даних, та видає їх клієнту.

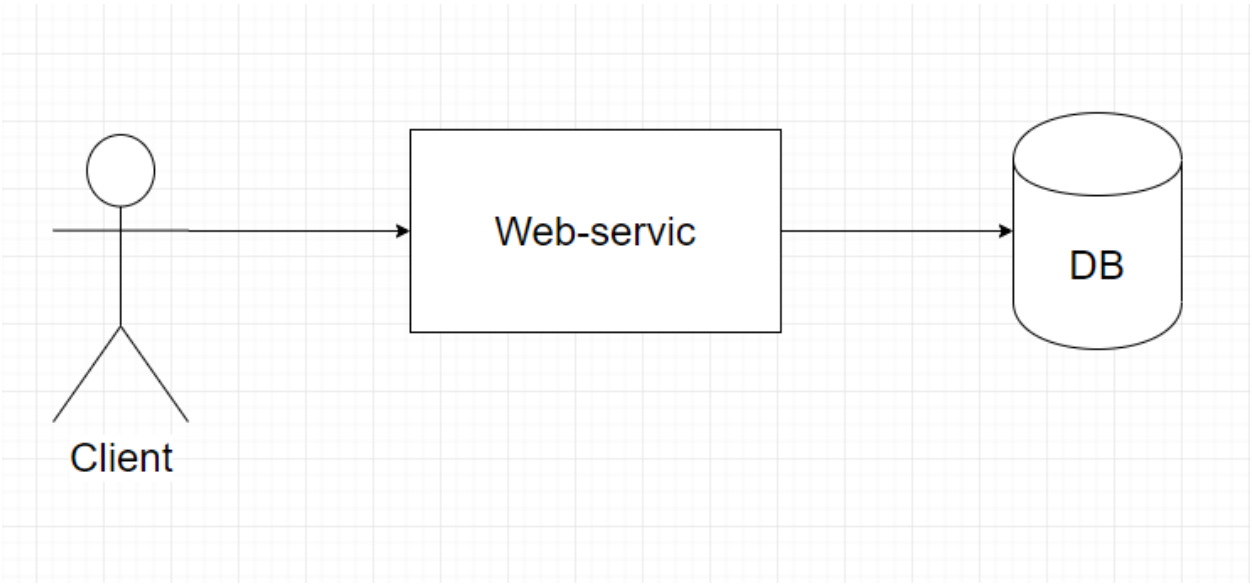

Рисунок 2.3.2.1- Схема передачі даних

Після під’єднання нейронної мережі, запит зможе як проходити звичні класи сервісу, так і згідно типу запиту перенаправлятись на нейронну мережу, та в свою чергу оброблятиме запит та видаватиме дані із власної бази даних, де дані структуровані та угруповані згідно попереднього навчання. Результати повертатимуться на сервіс та видаватимуться користувачу на його пристрій.

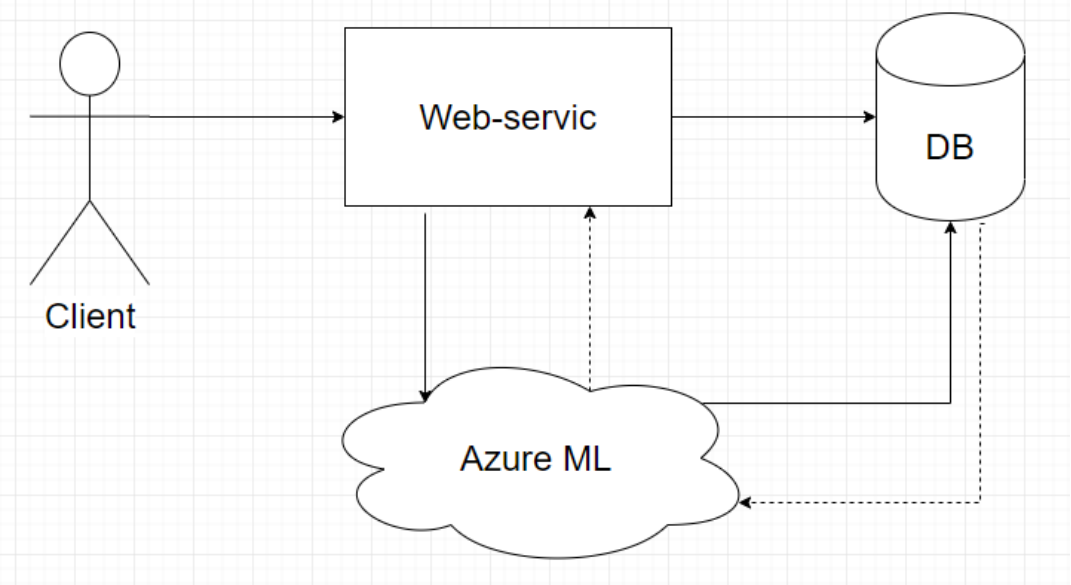

Рисунок 2.3.2.2- Схема передачі даних

Також, однією важливою функцією в пошуку рецептів, буде важливо додати фільтри. Вони пришвидшать пошук необхідного користувачу та покращать якість вибірки.

Основні фільтри:

- Оцінка
- Категорія
- Час приготування
- Спосіб приготування
- Калорійність
- Дієта

Створивши фільри, ми тим самим додаємо ще декілька функцій які покращать навчання нейронної мережі та допоможуть користувачу якісніше підібрати рецепт.

Оцінюючи чужі рецепти користувач вкаже нейронній мережі, які рецепти смачніші, тим самим, вона зможе в першу чергу брати дані з рецептів з оцінкою в 4-5 балів. А вже після переглядати в нисхідному порядку й інші.

Створивши категорії рецептів, ми розділимо всі їх по видах «салат», «горнір», «паста» і тома подібні. Що покращить перегляд всіх рецептів та краще навчить нейронну мережу.

Також додавши вид дієти, ми зможемо розежувати звичайні всім рецепти, та ті, які мають відмінності на основі тієї чи іншої дієти. Так як є люди з непереносимістю тих, чи інших інгредієнтів, заборонами від лікаря чи небажанням їсти ті чи інші продукти(такі як вегетаріанці чи вегани).

Також одним з важливих критеріїв є калорійність трави. В даний час багато людей слідкує за зовнішнім виглядом і їм важливо знати калорійність тієї чи іншої страви до її приготування ,а не вираховувати самостійно.

На основі всіх цих даних, проектована нейронна мережа зможе якісніше навчатись та видавати хороший кінцевий результат запиту користувача.

Створення нового рецепту за допомогою нейронної мережі довгий та клопіткй процес. Для цього їй потрібно самостійно навчитись поєднювати інгредієнти та зрозуміти вподобання людей. На основі всіх даних, та постійного навчання на нових доданих рецетах, нейронка завжди покращуватиме свої знання. Та щоб кінцевий результат був ще якіснішим, можна буде додати молекулярну сумісність інгредієнтів. Навчивши нейронну мережу на молекулярному рівні поєднювати інгредієнти, ми створимо нові, нікому невідомі раніше рецепти.

В даному веб сервісі, авторизваний користувач має змогу створювати рецепти. Він вибирає вкладку нового рецепту, вводить всі дані, сервіс перевіряє чи такий рецепт вже був раніше збережений та додає його до «мої рецепти» даного користувача і він стає доступним до перегляду кожному користувачу. Також можна додати кнопку скритого рецепту, тоді вони будуть видимі власнику, але не доступні іншим читачам. Таким чином, сервіс зможуть використовувати як власну кулінарну книгу. Користувач бажає мати збережені всі свої рецепти в одному місці, та не показувати їй іншим.

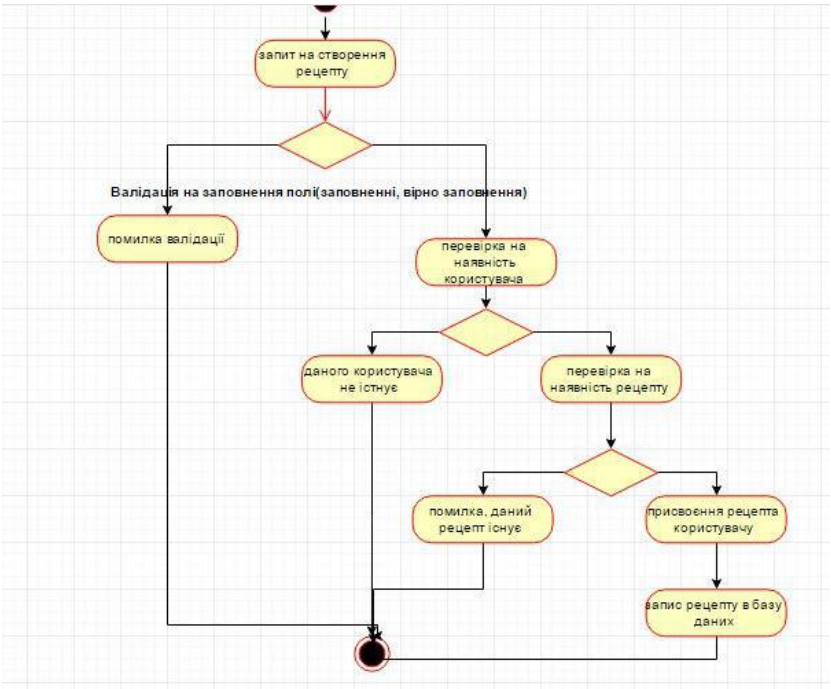

Рисунок 2.3.2.3- Юмл діаграма створення рецепту без нейронної мережі

Додавши нейронну мережу, ми додаємо нову вкладку до клієнтської частини «Рецепти нейронної мережі». Коли користувач не зможе знайти неоюхідий йому рецепт, будучи авторизованим, він матиме кнопку «створити нейронною мережею». Після чого йому буде виданий результат запиту. Користувач матиме змогу дати власну назву рецепту та бути його співавтором. Тобто після створення рецепту, він зберігається в базі даних як створений нейронною мережею, та з додаванням користувача, який ввів дані для даного рецепту та здійснив запит.

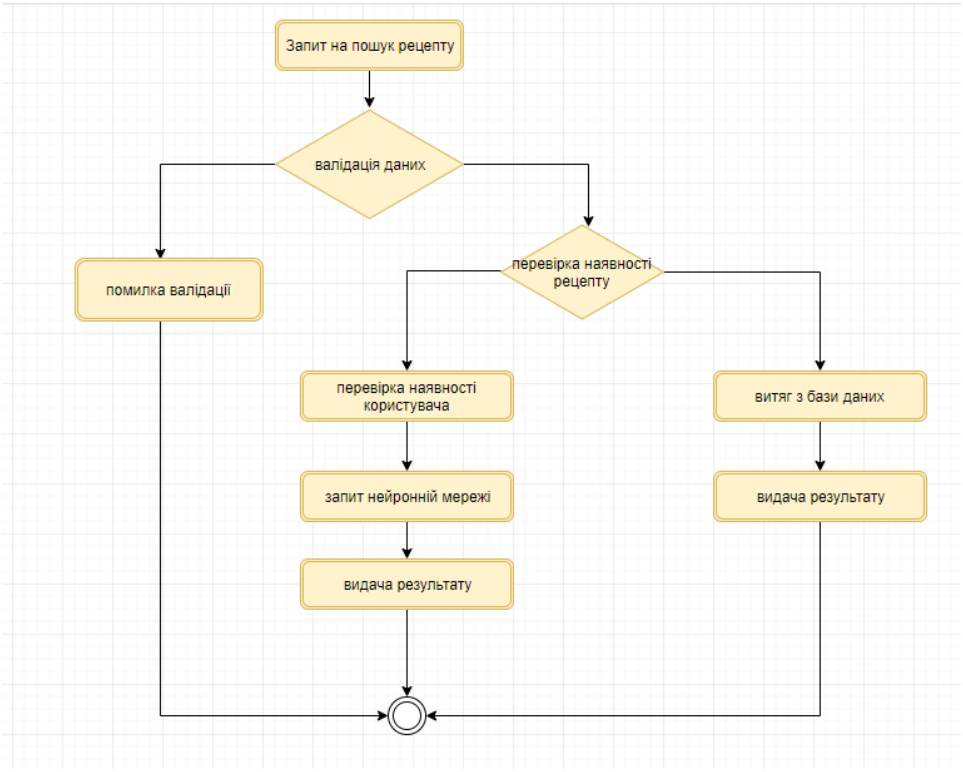

Рисунок 2.3.2.4- Юмл діаграма створення рецепту після додавання нейронної мережі.

2.3.3. Проектування реалізації нейронної мережі, згідно розробленого веб-сервісу.

Навчання на основі прогнозованих математичних моделей – це машинне навчання. При даному навчанні для прогнозування невідомих значеть, зв’язки між полями аналізуються.

Розгортання та створення моделі навчання виконується ітеративно:

Перегянувши вхідні дані, фахівці із обробки знаходять зв’язки між прогнозованими мітками та компонентами.Після чого на основі відповідних алгоритмів, перевіряють та навчають моделі, та пробують визначити оптимальну модель прогнозування. У Формі веб служби, або ж у формі іншого інкапсульованої функції дана модель розгортається у робочому середовищі. Отримуючи нові дані, модель час від часу повторно проходить навчання, що в свою міру збільшує її ефективність.

Для того, що дане завдання підтримувало масштабованість,як і попередньо розроблений сервіс, потрібно вирішити два завдання пов’язаних із машинним навчанням.

В першу чергу, навчання моделі по великим наборам даних вимагає високої продуктивності і масштабованості обчислювального кластера. Другою проблемою, орієнтовано на операціоналізацію вивченої моделі, потрібно, щоб та мала змогу масштабуватись для задоволення потреб програм, які споживають її. Зазвичай це можна досягнути шляхом розгортання можливостей прогнозування моделі як веб-сервісу, який потім можна буде зменшити.

Маючи підтримку масштабування, машинне навчання, забезпечить якісні прогностичні можливості, так як збільшивши обсяг даних, моделі здатні удосконалюватись.Після закінчення навчання модель можна розгорнути та використовувати як веб-службу без відстеження стану.

Розгортаючи модель нейронної мережі для навчання використовуючи Azure, ми можемо розгортути її як веб-службу, створивши інтерфейс REST Api. За допомогою даної служби, ми зможемо відправляти дані за допомогою запитів в попередньо створений REST сервіс.

Для навчання нейронної мережі нам необхідні дані. В даномку випадку, це буде наша база даних рецептів веб сервісу. Найчастіше, працюючи з локальними даними, використовують переваги гнучкості та масштабованості хмари для навантажень машинного навчання. Саме тому було обрано Azure ML, яка пітримує підтримує читання даних з локальної SQL Server бази даних, після чого проходить оцінку та навчання на основі цих даних. Безпосередньо із SQL Server, який використовується в реалізації попередньо створеного веб сервісу, модуль імпортування даних матиме змогу зчитувати дані.

Azure ML працює із двома способами передачі даних до rest сервісу. Це ключ, та токен. Так як весь попередньо розроблений сервіс базується на передачі запитів та токенів цей варіант чудово підходить для вирішення даної проблеми. Формуючи запит у вигляді JSON запиту, ми зможемо злегкістю як передавати, так приймати та опрацьовувати запити даним веб сервісом.

В першу чергу нам потрібно визначити робочу область. Ресурс виского рівня для Azure ML, що надає місце для роботи з артефактами, які створюються при використанні Azure ML і є робочою областю. Вона збрегіає історію всіх навчальних запусків, включаючи метрики, журнали, моментальні знімки сценаріїв та вихідні дані[21] .

Отримавши модель, ви реєструєте її в робочій області. Потім ви використовуєте зареєстровану модель і скрипти оцінки для розгортання в службі "екземпляри контейнерів Azure", "служба Kubernetes Azure" або у вигляді масиву шлюзів (FPGA), що підтримує використання полів, в якості кінцевої точки HTTP на

основі протоколу RESTFUL. Також можна розгорнути модель на Azure IoT Edge пристрої у вигляді модуля.

Модель створюється шляхом виконання в Azure ML. Також можна використовувати модель, навчену поза Azure ML. Зареєструвавши модель в Azure ML робочої області.

У простому розумінні модель являє собою фрагмент коду, який приймає вхідні дані і створює вихідні дані. Створення моделі машинного навчання включає в себе вибір алгоритму, надання йому даних і настройку гіперпараметрів. Навчання - це ітеративний процес, який створює навчену модель, що включає в себе все, чому модель навчилася в процесі навчання.

Використовуйте конвеєри машинного навчання для створення і управління робочими процесами, які поєднують етапи машинного навчання. Наприклад, конвеєр може включати в себе підготовку даних, навчання моделі, розгортання моделі, а також етапи визначення та оцінки. Кожен етап може охоплювати кілька кроків, кожен з яких може автоматично виконуватися в різних цільових об'єктах обчислення.

Етапи конвеєра можна багаторазово використовувати і можуть виконуватися без повторного виконання наступних кроків, якщо вихідні дані цього кроку не змінилися. Наприклад, можна перевчити модель без повторного виконання підготовчих кроків підготовки даних, якщо дані не змінювалися. Крім того, конвеєри дозволяють фахівцям з обробки та аналізу даних працювати в різних областях робочого процесу машинного навчання.

3. ОРГАНІЗАЦІЙНА-ЕКОНОМІЧНА ЧАСТИНА

3.1 Загальний підхід до визначення економічної ефективності розробки

Обов’язковою складовою частиною будь-якого інженерного проекту, в тому числі софтверного, є фінансові витрати на різних етапах виконання робіт. Відповідно, важливо вірно здійснити фінансову оцінку передбачуваних витрат, продуктивність, корисність та, в результаті, економічну ефективність проекту.

Наукові розробки та дослідження, на відміну від корпоративних, не завжди супроводжують за мету отримання прибутку або ж іншої матеріальної вигоди. В багатьох випадках, проекти наукового спрямування не є економічно вигідними. Однак, вони є рушійною силою прогресу, дослідженням незвіданих галузей та проблем, які, у свою чергу, майже завжди впливають на майбутні різнопланові розробки. Тому дуже часто наукова діяльність стимулюється зовнішніми інвестиціями та підтримкою держаних інститутів, міжнародних грантів. В плані використання результатів досліджень та на основі отриманих моделей можна робити прогнози по впровадженню нових методик та принципів організації роботи діагностичних установ. Різноманітні медичні центри зможуть скористатися на практиці розробленою системою для покращення існуючих та отримання нових діагностичних методик.

Оцінка вартості дослідницьких розробок базується на витратному підході: використанні первісної вартості об'єктів, виходячи з фактичних витрат на розробку та доведення до комерційного використання з урахуванням амортизації. Так, як результати роботи у вигляді математичних моделей та реалізованого на їх основі ПЗ не буде використовуватися в комерційних цілях та не підлягатиме продажу, а становить наукову та інтелектуальну цінність, то доходу від продажі ПЗ та розробки як такого не передбачається. Іншими словами, всі вкладені кошти та

витрати на розробку даного рішення є не взаємоокупними, що несуть лише витрати у кількості залучених ресурсів та матеріальних засобів.

Згідно Статті 8 Закону № 3792-12 передбачено, що твори наукового характеру та комп'ютерні програми є об'єктами авторського права [18]. У разі реєстрації виконаної роботи як інтелектуальної власності та авторського права у державній службі інтелектуальної власності України необхідно буде сплатити збори за державну реєстрацію (382,5 грн.).

Для отримання відмінних результатів експериментів та доцільності розробки такого спеціалізованого ПЗ потрібні відповідні затрати на дослідження та розробку. Це і становитиме основу витрат, які будуть здійснені протягом підготовки та виконання реалізації даного рішення. Уявно модель витрат можна поділити на дві основні частини: витрати, пов'язанані на дослідження предметної області, побудову математичних моделей, отримання попередніх результатів експериментів, та частину реалізації програмної системи, архітектури та тестування.

Враховуючи залежність якості кінцевого продукту від кваліфікації програмістів, потрібно сконцентрувати увагу на якості та результативності розробки. Для цього потрібно провести ґрунтовний аналіз предметної області, залучити найновіші та інноваційні технології, провести ґрунтовне тестування та оцінку результатів розробки. Для забезпечення хорошої результативності при розробці доводиться йти на додаткові заходи заохочення та стимулювання у вигляді преміювання працівників, підтримання наукового дослідження досвідом іноземних науковців, дорогоцінних лабораторних дослідів.

До створення ПЗ можуть бути залучені позаштатні програмісти як зареєстровані, так і не зареєстровані підприємцями. В обох випадках співпраця з ними здійснюється на підставі цивільно-правового договору, найчастіше – договорами підряду. Щодо оподаткування виплат за договором підряду, то все

залежить від того, чи зареєстрований виконавець підприємцем. Важливим етапом розробки ПЗ є його тестування, яке виконується тестувальником за певну винагороду. Якщо тестувальники не перебувають з підприємством у трудових відносинах, оплата виконується на основі договору підряду на виконання робіт з тестування ПЗ. Сам же результат розробки не оподатковується, адже не є комерційним проектом і не спрямований на продаж.

3.2 Розрахунок вартості процесу розробки та оцінка економічної ефективності проекту

Для оцінки нематеріальних активів використовують міжнародні стандарти оцінки розрахунку вартості об’єктів інтелектуальної власності, розроблені TIAVIS (The International Assets Vabulation Standarts Commitie).

Виконання розробки програмного забезпечення з огляду економічної моделі можна виконувати двома способами: процедурним та об’єктно-орієнтованим. Обидва підходи потребують залучення ресурсів у вигляді програмісті-розробників, тестувальників, керівника проекту, наукового ресурсу. Різниця виникає в самій схемі розробки, тривалості періоду розробки та відповідній вартості. Процедурний підхід для розробки ПЗ в основі якого лежать процедури і функції передбачає розробку ПЗ як монолітного композиту, що в подальшому, як правило, вимагає великих витрат на супровід та модернізацію. Об’єктно-орієнтований підхід, що ґрунтується на основі об’єктів певних класів, що описують певну область, описують певну поведінку (методи) та володіють властивостями (атрибутами), орієнтовані на варіанти використання та покроковий процес розробки [18].

Для початку робіт необхідно скласти технічне завдання на розробку, яке є основним документом, що регламентує подальшу роботу, та містить докладний опис необхідних функцій програми, інтерфейс, технології, інше. Вартість

складання технічного завдання переважно складає до 10% від планованої вартості розробки. Роботу зі складання технічного завдання веде керівник проекту разом із програмістами та консультуючись із замовником.

Усі програмісти, що працюють у штаті підприємства-розробника мають встановлено певний посадовий оклад. Місячний оклад, денна заробітна плата, трудомісткість (днів) і основна заробітна плата кожного учасника техпроцесу представлено у таблиці 3.1. Всі суми наведені в національній валюті – в гривні.

Таблиця 3.1 – Розрахункова вартість технологічного процесу розробки

Посада	Місячний оклад, грн.	Денна зар. плата, грн.	Об'єктно-орієнтований підхід		Процедурний підхід	
			Днів	Сума, грн.	Днів	Сума, грн.
Керівник	12100,00	550,00	15	8250,00	16	8800,00
Програміст	11660,00	530,00	21	11130,00	25	13250 ,00
Тестувальник	9680,00	440,00	7	3080,00	7	3080,00
Бізнес аналітик	10450,00	475,00	10	4750,00	10	4750,00
Додаткова зар. плата 20%			38	5442,00	42	5976,00
Фонд оплати праці 36,77%			10005,117		10986,876	
Всього витрат на зар. плату			42657,12		46842,88	
Військовий збір 1,5%			639,86		702,64	
Єдиний соціальний внесок 3.6%			1535,66		1686,34	
ПДВ, 15%			6398,57		7026,43	
Всього			**51222,21**		**54571,95**	

Згідно вимог та прорахованої кількості необхідних ресурсів на виконання, розробку, тестування та дослідницьку роботу було отримано основні часові рамки роботи над проектом. Так для об'єктно-орієнтованого підходу загальна тривалість роботи над ПЗ становить 38 робочих днів (під робочим днем розуміється 8-ми

годинний робочий день), що включає роботу програміста, який, в свою чергу, являється і керівником розробки, роботу тестувальника та наукового працівника. Сума витрат на заробітню плату становить 51222,21 гривень включаючи всі види додаткових оплат. Для процедурного підходу до розробки суми дещо більші, адже затрачається більше часу на розробку. Так, при використанні процедурного підходу сумарна тривалість часу розробки становить 42 робочі дні, та витрати у вигляді виплат заробітної плати становлять 54571,95 гривень.

Витрати на науково-дослідницьку роботу та здійснення розробки програмних продуктів і об'єктно-орієнтованим, і процедурним способом включають:

Основна заробітна плата:

$ЗП_{осн\ 1} = 27210$ грн; $ЗП_{осн\ 2} = 29880$ грн.

Додаткова заробітна плата обчислюється як $ЗП_{дод} = 0,2 \cdot ЗП_{осн}$.

$ЗП_{дод\ 1} = 0,2 \cdot 27210 = 5442$ грн; $ЗП_{дод\ 2} = 0,2 \cdot 29880 = 5976$ грн.

Нарахування на фонд оплати праці (ФОП):

$ФОП_{ЄСВ} = 0,3677 \cdot ФЗП$

$ФОП_{ЄСВ1} = 0,3677 \cdot 32652 = 10005,117$ грн;

$ФОП_{ЄСВ1} = 0,3677 \cdot 35856 = 10986,876$ грн.

Всього витрат:

$В_{ЗП1} = ЗП_1 + ФОП_{ЄСВ1} + ЗП_{дод1} = 51222,21$ грн;

$В_{ЗП2} = ЗП_2 + ФОП_{ЄСВ2} + ЗП_{дод2} = 37130,97$ грн.

З цієї суми утримуються обов'язкові відрахування на заробітну плату: єдиний соціальний внесок, який складає 3,6% від суми нарахованої заробітної плати та податок на доходи фізичних осіб, який складає 15% від суми нарахованої заробітної плати, зменшеної на суму єдиного внеску на загальнообов'язкове соціальне страхування та податкової соціальної пільги, військовий збір у розмірі 1,5%, від суми нарахувань.

До окремих витрат також відносяться витрати на куповані вироби (матеріальне забезпечення) та спец обладнання для підтримки експерименту, накладні витрати. Витрати, що будуть супроводжувати проект розробки, порівнюватимемо в двох можливих підходах розробки.

Матеріальні витрати визначаються як добуток кількості витрачених матеріалів та їх ціни (формула 3.1).

$$M_{Bi} = q_i \cdot p_i \,, \qquad (3.1)$$

де q_i – кількість витраченого матеріалу i-го виду; p_i – ціна матеріалу i-го виду.

Матеріальні витрати в рамках проекту наведені в таблиці 3.2. Загальна сума матеріальних витрат становить 1535 гривень.

Таблиця 3.2 – Матеріальні витрати

Найменування ресурсу	Кількість, шт.	Ціна одиниці, грн	Загальна сума, грн
Флешки	4	227,00	908,00
Папір для друку А4, арк	500	0,158	79,00
Тонер для принтера	1	60,00	60,00
Дошка для записів	1	450,00	450,00
Перманентний маркер	4	12,00	48,00
Всього			1535,00

Розрахунок витрат на електроенергію одиниці обладнання визначаються за формулою:

$$З_в = W * T * S, \qquad (3.2)$$

де W – необхідна потужність, кВт; T – кількість годин роботи обладнання; S – вартість кіловат-години електроенергії, S = 2,48 грн/кВт·год.

$З_{в1}$ =0.7 * 304 * 2.48 = 527,74 грн;

$З_{в2}$ =0.7 * 336 * 2.48 = 583,3 грн;

Розрахунок суми амортизаційних відрахувань. Комп'ютери та оргтехніка належать до четвертої групи основних фондів. Для цієї групи річна норма амортизації дорівнює 60 %, вартість яких перевищує 1000 грн. і визначається:

$$A = \frac{C_Б \cdot N_A \cdot T_{ФАК}}{T_{ГОД}} \qquad (3.3)$$

де $C_Б$ – балансова вартість обладнання, грн; N_A – норма амортизаційних відрахувань в рік, %; $T_{ФАК}$ – фактичний час роботи обладнання по написанню програми, год; $T_{ГОД}$ – річний робочий фонд часу, год.

У даній формулі норма відрахувань на амортизацію рівна N_A = 0,6. Балансова вартість обладнання вказана в таблиці 3.3 і рівна $C_Б$ = 22589 гривень. Річний робочий фонд часу приймемо за $T_{ГОД}$ = 2120 годин. З них реальний фактичний робочий час становить $T_{ФАК}$ = 304 години згідно об'єктно-орієнтованого підходу та $T_{ФАК}$ = 336 годин згідно процедурного підходу.

Згідно вищезгаданої формули витрати на амортизацію становлять 1943,5 гривень та 2135,3 гривень для кожного підходу відповідно.

Таблиця 3.3 – Перелік необхідного обладнання

Найменування	Кількість, шт	Ціна, грн	Сума, грн	
Комп'ютер	2	9545,00	19090,00	
Принтер	1	3499,00	3499,00	
Середовища розробки	2	безкоштовно	безкоштовно	
Операційна система (Linux)	2	безкоштовно	безкоштовно	
Всього більше 1000 грн.			22589,00	
Всього витрат на амортизацію			1943,5	2135,3
Всього			24532,5	24724,3

Накладні витрати пов’язані з обслуговуванням виробництва, утриманням апарату управління та створення необхідних умов праці та закупівлю ресурсів та обладнання для розробки наведені в таблиці 3.3.

Залежно від організаційно-правової форми діяльності господарюючого суб’єкта, накладні витрати можуть становити 20–60 % від суми основної та додаткової заробітної плати працівників. Нехай вона буде дорівнювати 30%, що становить 9795,6 грн для об’єктно-орієнтованого і 10756,3 грн для процедурного підходу розробки.

Проведемо розрахунок вартості створюваного програмного продукту. Вартість продукції включає у собі собівартість і планований прибуток. Найважливішим моментом для розробника, з економічної точки зору, є процес встановлення ціни.

Можна отримати загальні значення витрат на розробку та реалізацію проекту, враховуючи всі вище описані затрати та нарахування. Цей вид витрат складається з сум витрат на оплату праці (всього витрати на оплату праці), матеріальні затрати, затрати на електроенергію, накладні витрати, витрати на обладнання, враховуючи амортизації обладнання на час виконання проекту.

Собівартість продукції – це сума грошових витрат підприємства (фірми) на виробництво і збут одиниці продукції, виконання робіт та надання послуг.[18]

Повна собівартість програмного продукту дорівнює сумі усіх витрат на його виробництво: 87613,05 грн використовуючи об’єктно-орієнтований підхід, 92170,85 грн при процедурному підході розробки.

Ефективність виробництва – це узагальнене і повне відображення кінцевих результатів використання робочої сили, засобів та предметів праці на підприємстві за певний проміжок часу.

Економічна ефективність (Е) полягає у відношенні результату виробництва до затрачених ресурсів:

$$E = \frac{П}{Св} \qquad (3.4)$$

де *П* – прибуток, *П= В - Св; Св* – собівартість.

У випадку даної розробки, маючи некомерційний проект без економічно корисного результату, можна прогнозувати, що економічна ефективність прямує до 0 у обох випадках. Однак це не є причиною для негативного економічного висновку щодо даного проекту, адже такого плану розробки приносять користь у вигляді інтелектуальних ресурсів, і, переважно, фінансуються або виконуються на замовлення організацій, зацікавлених в отриманні результатів досліджень та розробок. Фінансування не можна вважати отриманим доходом від реалізації. Однак за надходження коштів на реалізацію ззовні можна вважати ефективність проекту рівною 1 (E = 1), що означає перекриття витрат на розробку у повній мірі, тобто фінансування.

Якщо ринкова вартість програмного продукту рівна прийнятій, то економічна ефективність визначається встановленим рівнем прибутку. Поряд із економічною ефективністю розраховують термін окупності капітальних вкладень ($T_{ок}$):

$$T_{ок} = \frac{1}{E} \qquad (3.5)$$

У нашому випадку прямого прибутку не існує. Прибуток можна прогнозувати на підприємствах, організаціях чи відомствах, що зацікавлені в дослідженні. Окупність же для розробки даного ПЗ за ефективності рівній одиниці можна вважати теж рівній 1 згідно формули 3.4.

Виходячи із експертних оцінок і складності програми, приймемо величину витрат на супровід і модернізацію програмного забезпечення, створеного за

об’єктно-орієнтованим методом 25% (21903,26 грн) від початкових витрат, а за процедурним – 30% (27651,26 грн).

Однак варто зауважити, що розробка спрямована на короткотривалу підтримку та не прогнозує модернізації. У разі ж необхідності розробки такого ж або суміжного ПЗ можна вважати за доцільно розпочинати розробку з початкових етапів, що потягне за собою нові витрати у повній мірі. Тобто у разі короткотермінової підтримки все ж за доцільніше обирати об’єктно-орієнтовану модель розробки, адже вартість короткотермінової підтримки згідно цієї моделі не вплине значно на сукупну вартість розробки.

Здана в експлуатацію система не завжди цілком завершена, її треба змінювати протягом терміну експлуатації. Внаслідок змін система стає більш складною і погано керованою. Об’єктно-орієнтоване представлення програми дозволяє навіть середньому програмісту швидко і ефективно супроводжувати і модернізувати програми, що значно скорочує подальші витрати на супровід і модернізацію [18].

Сумарні дані економічного розрахунку розробки даного проекту наведені в таблиці 3.4.

Таблиця 3.4 – Загальні витрати

Вид витрат	Об’єктно-орієнтований підхід, грн	Процедурний підхід, грн
Зарплата основна	27210,00	29880,00
Зарплата додаткова	5442,00	5976,00
Фонд заробітної плати	32652,00	35856,00
Відрахування на ФОП	10005,12	10986,88
Разом на оплату праці	51222,21	54571,95
Матеріальні витрати	1535,00	1535,00
Електроенергія	527,74	583,30
Амортизація	1943,50	2135,30
Накладні витрати	9795,60	10756,30
Обладнання	22589,00	22589,00
Разом на ін. витрати	36390,84	37598,9
Собівартість	87613,05	92170,85
Прибуток	відсутній	відсутній
Вартість розробленого ПЗ	**87613,05**	**92170,85**
Модернізація і супровід	21903,26	27651,26
Загальні витрати на розробку	109516,30	119822,11
Економія ($ЗВ_1$-$ЗВ_2$)	**10305,81**	

Загальна вартість пропонованих робіт становить 119822,11 гривень для процедурного і 109516,30 гривень для об’єктно-орієнтованого підходів розробки. В даному випадку реалізації проекту варто вибрати об’єктно орієнтований підхід для розробки даного ПЗ, адже фінансово це більш вигідно. Також у даній методиці розробки кращі часові рамки та перспективи підтримки і модернізації. У оцінці вартості продукту варто враховувати можливість фінансування та спонсорування проекту, що дозволить гнучко та ефективно підійти до розробки та організації праці.

4.ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Охорона праці

Для реалізації даного проекту по розробці програмної системи керування контентом, необхідне комп’ютерне обладнання. Слід зауважити, що виконуючи дану роботу програміст проводить за робочим місцем достатньо велику кількість годин. Тому організація його робочого місця має відповідати всім нормах для збереження його здоров’я.

Існує ряд певних вимог для користувачів щодо збереження здоров’я та працездатності при роботі з комп’ютером. Потрібно правильно спланувати і розмістити робочі місця для роботи з комп’ютером. Їх заборонено облаштовувати у підвальних або цокольних приміщеннях будинків відповідно до ДСанПіН 3.3.2-007-98 "Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин". В обладнанні приміщень діагностичного центру забороняється використання полімерних матеріалів, що виділяють шкідливі хімічні речовини. Також слід приділити увагу забезпеченню достатнім для здійснення роботи рівнем освітлення (природного та штучного – у темну пору доби) та звукоізоляції відповідно до ДБН В.2.5-28:2016. Для регуляції рівня освітлення природним світлом бажано застосовувати жалюзі. Окрім того, у приміщеннях, де здійснюється робота з комп’ютерами, щодня має здійснюватися вологе прибирання з метою недопущення запиленості підлоги та меблів.

Приміщення з комп’ютерами повинні бути обладнані системами опалення, кондиціювання повітря або припливно-витяжною вентиляцією, а параметри мікроклімату, іонного складу повітря, вмісту шкідливих речовин на робочих місцях, оснащених відеотерміналами, повинні відповідати вимогам пункту 2.4 СН 4088-86 "Санітарні норми мікроклімату виробничих приміщень", ГОСТ 12.1.005-88

та СН 2152-80 "Санітарно-гігієнічні норми допустимих рівнів іонізації повітря виробничих та громадських приміщень".

Заземлені конструкції, що знаходяться в приміщеннях, де розміщені робочі місця операторів (батареї опалення, водопровідні труби, кабелі із заземленим відкритим екраном), мають бути надійно захищені діелектричними щитками або сітками з метою недопущення потрапляння людини під напругу.

Для виконання даного проекту необхідні потужні ЕОМ які безперервно включенні в систему живлення, тому необхідно дотримуватись всіх правил безпеки[26].

Особливої уваги заслуговують заходи дотримання протипожежної безпеки. Так, у всьому робочому приміщенні лінії електромережі забезпечуються від виникнення короткого замикання, а також від перепадів мережевої напруги, що може спричинити збої в роботі електронно-обчислювальної техніки. Приміщення (окрім тих, де розташовуються сервери) мають бути оснащені системою автоматичної пожежної сигналізації та вогнегасниками. Під час монтажу та експлуатації ліній електромережі необхідно повністю унеможливити виникнення електричного джерела загоряння внаслідок короткого замикання та перевантаження проводів, обмежувати застосування проводів з легкозаймистою ізоляцією і, за можливості, застосовувати негорючу ізоляцію. У приміщенні, де одночасно експлуатуються понад п'ять комп’ютерів, на помітному та доступному місці встановлюється аварійний резервний вимикач, який може повністю вимкнути електричне живлення приміщення, крім освітлення.

Вимоги щодо організації та обладнання робочих місць програміста модуля розмежуванн згідно з ДСанПіН 3.3.2-007-98: площа, відведена на одне робоче місце має становити не менше 6 кв. м., а об’єм – не менше 20 куб. м. Конструкція робочого місця повинна забезпечувати підтримання оптимальної робочої пози (тобто такої, яка дозволяє працівникові виконувати роботу з мінімальним

напруженням тіла, і яка дозволяє уникнути перевтоми в ході і після закінчення робочого процесу).

За потреби особливої концентрації уваги під час виконання робіт суміжні робочі місця операторів ЕОМ необхідно відділяти одне від одного перегородками висотою 1,5 - 2 м.

Робочі місця слід розташовувати відносно джерела природного світла (вікон) таким чином, щоб світло падало збоку. Також робоче місце має відповідати сучасним вимогам ергономіки згідно з ДСанПіН 3.3.2.007-98:

Через одноманітуну, монотонну роботу в одному положенні тіла, програмісту необхідно виконувати періодичні перерви. Під час регламентованих перерв рекомендується виконувати комплекси вправ для очей, рук, хребта, поліпшення мозкового кровообігу тощо. Про виявлення несправності обладнання або інших факторів, які створюють загрозу для життя або здоров'я працівників, необхідно негайно інформувати свого безпосереднього керівника.

4.2 Фактори що впливають на функціональний стан користувачів комп'ютерів

Трудовий процес суттєво впливає на психофізіологічні можливості користувачів комп'ютерів, оскільки їх діяльність характеризується значними статичними фізичними навантаженнями; недостатньою руховою активністю; напруженнями сенсорного апарату, вищих нервових центрів, які забезпечують функції уваги, мислення, регуляції рухів. Окрім того, трудовий процес користувачів комп'ютерів відзначається значними інформаційними навантаженнями.

Трудова діяльність користувачів комп'ютерів (ВДТ) відбувається у певному виробничому середовищі, яке впливає на їх функціональний стан. Найбільш значимі — фізичні фактори виробничого середовища, до яких належать електромагнітні хвилі різних частотних діапазонів, електростатичні поля, шум, параметри мікроклімату та ціла низка світлотехнічних показників. Вплив хімічних та, особливо, біологічних факторів виробничого середовища на користувачів комп'ютерів — значно менший.

Професійні якості та виробничий досвід, які визначають внутрішні засоби діяльності, обумовлюють надійну та безпомилкову діяльність користувачів комп'ютерів, дозволяють знаходити безпечні методи розв'язання виробничих завдань навіть у нестандартних ситуаціях.

Зовнішні засоби діяльності, які в основному визначаються ергономічними показниками щодо організації робочого місця, форми та параметрів його елементів, просторового розташування основного ідопоміжного устаткування, можуть суттєво знизити фізичні та психофізіологічні навантаження, що діють на користувачів комп'ютерів.

Особливості роботи користувачів комп'ютерів

У професійних операторів частіше зустрічаються порушення органів зору, опорно-рухового апарату, центральної нервової, серцево-судинної, імунної та статевої систем, захворювання шкіри. Зафіксована значна кількість скарг операторського персоналу на загальне недомагання, передчасне стомлювання, головний біль, порушення функцій органів зору, які здійснювали несприятливий психофізіологічний вплив на самопочуття та працездатність операторів.

Сучасна професія користувача ВДТ належить до розумової праці, яка характеризується: високою напруженістю зорових функцій; одноманітною позою; великою кількістю стереотипних висококоординованих рухів, що виконуються лише м'язами кистей рук на фоні малої загальної рухової активності; значним нервово-емоційним компонентом, особливо в умовах дефіциту часу; роботою з великими масивами інформації, що викликає активізацію уваги та інших вищихпсихічних функцій. Крім того, при роботі з дисплеями на електронно-променевих трубках виникає вплив на користувача цілої низки факторів фізичної природи — електростатичні поля, радіочастотне та рентгенівське випромінювання тощо.

Діяльність професіоналів можна поділити на три групи:

1. Діяльність, яка пов'язана з виконанням нескладних багаторазово повторюваних операцій, що не вимагають великого розумового напруження. Наприклад, робота операторів комп'ютерного набору, працівників довідкових служб.

2. Діяльність, яка пов'язана із здійсненням логічних операцій, що постійноповторюються. Це робота інженера-економіста, інженера-проектувальника, оператора автоматизованого виробництва.

3. Діяльність, коли в процесі роботи необхідно приймати рішення за відсутності заздалегідь відомого алгоритму. Наприклад, робота інженера-програміста, диспетчерів руху залізничного транспорту, аеропортів тощо.

У користувачів, які інтенсивно використовують комп'ютер в умовах значних розумових напружень досить часто (40—70%) виникають психологічні та поведінкові порушення (нервозність, роздратування, тривога, нерішучість, замкнутість тощо). Серед користувачів ВДТ в США і Європі значного поширення набуло специфічне захворювання, яке отримало назву синдром комп'ютерного стресу (СКС). СКС супроводжується головним болем, запаленням очей, алергією, роздратованістю, млявістю і депресією. Інформаційне перевантаження користувачів ВДТ супроводжується низкою специфічних захворювань, які називають інформаційними. Першим симптомом їх є головний біль. Дослідження, проведені в США, Німеччині, Швейцарії та інших країнах, показали, що робота з обслуговування ВДТ супроводжується підвищеним напруженням зору, інтенсивністю і монотонністю праці, збільшенням статичних навантажень, нервово-психічним напруженням, впливом різного виду випромінювань та ін. Внаслідок цього серед операторів ВДТ, як зазначають фахівці Всесвітньої організації охорони здоров'я, частіше, ніж в інших групах працюючих, трапляються такі професійні захворювання, як передчасна стомлюваність, погіршення зору, м'язові і головні болі, психічні й нервові розлади, хвороби серцево-судинної системи, онкологічні захворювання та ін. Вважається, що стан організму операторів ВДТ визначається комплексним впливом факторів трудового процесу і середовища, значення яких є неоднаковим. На операторів з малим стажем роботи на ВДТ домінуючий вплив чинять фактори середовища, а на операторів зі стажем понад 5 років - фактори трудового процесу.

Розлади здоров'я користувачів, що формуються під впливом роботи за комп'ютером**.** Зоровий дискомфорт

Комп’ютерний зоровий синдром (КЗС) – комплекс порушень здоров’я, який може виникати у користувачів персональних комп’ютерів (ПК). Діагноз ставлять,

якщо людина, що працює за ПК протягом двох годин, висловлює хоча б дві з десяти скарг:

— головний біль

— сльозотеча

— різь

— туман

— двоїння

— свербіж

— важкість в очах

— фотофобія

— миготіння знаків на екрані

— нудота.

У користувачів ПК дуже поширені кон’юнктивіти і блефарити, патогенетично пов’язані з КЗС.

Синдром розвивається при умові, що робоче місце організовано неправильно – у користувача незручне крісло, відсутні пюпітри для паперів, підставки для ніг та кистей рук, не встановлена висота і нахил монітора відносно очей, відстань від очей до екрана. За таких умов тіло людини при роботі займає вимушене положення: спина статично напружена, шия витягнута, плечі жорстко фіксовані. Напружені м’язи погіршують кровотік у сонних артеріях, а недостатнє кровозабезпечення головного мозку веде до очманіння, появи головного болю. На фоні шийного остеохондрозу з’являється відчуття випирання очних яблук, туману в очах, мушок та райдужних кіл у полі зору. Розвитку КЗС сприяє поганий мікроклімат приміщення, значна загальна іонізація та мікробне забруднення, а також куріння.

Національною радою з наукових досліджень США для стану зорового дискомфорту був уведений термін "астенопія", який означає "будь-які суб'єктивні

зорові симптоми чи емоційний дискомфорт, що є результатом зорової діяльності". Симптоми астенопії були класифіковані на "очні" (біль, печія та різь в очах, почервоніння повік та очних яблук, ломота у надбрівній частині тощо) та "зорові" (пелена перед очима, мерехтіння, швидка втома під час зорової роботи та ін.).

У операторів ВДТ "очні" симптоми трапляються частіше, ніж "зорові", причому частота проявів астенопії вища у жінок, ніж у чоловіків і більше виражена в осіб середнього і старшого віку. Причиною вважається електромагнітне випромінювання від ВДТ.

При роботі з ВДТ основне навантаження припадає на всі елементи зорового аналізатора.

Робота з ВДТ може призвести до розвитку короткозорості, так як у користувачів комп'ютерів, в основному, "працює" ближній зір.

При аналізі зорової роботи операторів ВДТ, встановлено, що через дві години частота флуктуацій акомодації зменшується, а внесок низькочастотної компоненти підвищується. Це може бути причиною скарг на втому зорового аналізатора. Тривала робота на ВДТ може призвести до розвитку короткозорості, оскільки у користувачів ВДТ головним чином "працює" ближній зір.

У 100 пацієнтів із 150, які працювали на ВДТ по шість годин на день протягом чотирьох років, були виявлені проблеми з фокусуванням зору.

Робота за комп'ютером характеризується також тим, що постійний напружений погляд на екран монітора зменшує частоту моргання. При цьому погіршується зволоження поверхні очного яблука сльозовою рідиною, яка захищає рогівку ока від висихання, пилу та інших забруднень. Це може призвести до виникнення так званого синдрому Сікка: рогівка висихає і мутніє, і як наслідок розвивається сліпота.[25]

ВИСНОВКИ

В ході виконання даної магістерської роботи було створено web-сервіс спроектований за архітектурним шаблоном REST та спроектоване покращення роботи веб сервіса за допомогою нейронної мережі.

Метою даної роботи було проектування та реалізація web-сервісу кулінарних рецептів та проектування подальшого його покращення за допомогою нейонної мережі. Для досягнення поставленої мети було досліджено проблему та шляхи її вирішення, проведено аналіз предметної області та розглянуто принципи роботи web-сервісів та нейронних мереж.

Було проаналізовано можливі способи взаємодії програмних клієнтів з сервером через програмний інтерфейс. Складено вимоги до функціоналу web-сервісу та спроектовано програмний інтерфейс сервісу для взаємодії з зовнішніми системами (API). Спроектовано модель бази даних і структуру сервісу,а також проведено їх реалізацію та тестування. Спроектовано основні функції нейронної мережі для даного веб сервісу

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1 Code-first в Entity Frameiwork [Електронний ресурс]: режим доступу: https://habrahabr.ru/post/121132/

2 Entity Framework Code First на практике [Електронний ресурс]: режим доступу: https://habrahabr.ru/post/236037/

3 Entity Frameiwork [Електронний ресурс]: режим доступу: http://metanit.com/sharp/entityframework/1.1.php

4 Введение в Entity Framework 6 [Електронний ресурс]: режим доступу: METANIT.COM

5 Learn About ASP.NET Web API [Електронний ресурс]: режим доступу: http://www.asp.net/web-api

6 Введение в Web API - METANIT.COM [Електронний ресурс]: режим доступу: http://metanit.com/sharp/mvc/12.1.php

7 Web API - Wikipedia, the free encyclopedia [Електронний ресурс]: режим доступу: https://en.wikipedia.org/wiki/Web_API

8 Разработка web API [Електронний ресурс]: режим доступу: https://habrahabr.ru/post/181988/ Разработка web API / Хабрахабр

9 Норми освітленн офісних приміщень [Електронний ресурс]: режим доступу: http://gc.ua/business-news/oxorona-praci-v-ofisi-vimogi-do-robochogo-miscya-ofisnogo-pracivnika/ охорона праці

10 REST [Електронний ресурс]: режим доступу: https://uk.wikipedia.org/wiki/REST

11 Coocking chanel [Електронний ресурс]: режим доступу: http://www.cookingchanneltv.com/home.html

12 Рецепстер [Електронний ресурс]: режим доступу: http://recipester.ru/

13 Jamie Oliver [Електронний ресурс]: режим доступу: http://www.jamieoliver.com/recipes/

14 HTTP [Електронний ресурс]: режим доступу: https://uk.wikipedia.org/wiki/HTTP

15 C# [Електронний ресурс]: режим доступу: https://uk.wikipedia.org/wiki/C_Sharp

16 OAuth [Електронний ресурс]: режим доступу: https://uk.wikipedia.org/wiki/OAuth

17 ADO.NET Entity Fraimwork [Електронний ресурс]: режим доступу: https://ru.wikipedia.org/wiki/ADO.NET_Entity_Frameworkx

18 Методичні вказівки для виконання розділу дипломної роботи щодо техніко-економічного обґрунтування вибору проектного рішення розробки та оцінки якості програмного забезпечення / Упор. Петрик М.Р., Кінах Я.І., Головатий А.І., Рогатинська Л.Р. – Тернопіль: Вид-во ТНТУ ім. І. Пулюя. – 2013. – 34 с.

19 Microsoft SQL Server [Електронний ресурс]: режим доступу: https://uk.wikipedia.org/wiki/Microsoft_SQL_Server\

20 Стаття з Вікіпедії. Rapid application development. [Електронний ресурс] - Режим доступу: URL: https://en.wikipedia.org/wiki/Rapid_application_development

21 Автоматичне навчання [Електронний ресурс]: режим доступу: https://docs.microsoft.com/ru-ru/azure/machine-learning/service/how-to-auto-train-forecast

22 Доступ до даниї [Електронний ресурс]: режим доступу: https://docs.microsoft.com/ru-ru/azure/machine-learning/service/how-to-access-data

23 What is Azure Machine Learning [Електронний ресурс]: режим доступу:https://docs.microsoft.com/uk-ua/azure/machine-learning/service/overview-what-is-azure-ml

24 Azure Machine Learning: Розробка сервісів машинного навчання [Електронний ресурс]: режимдоступу :https://habr.com/ru/company/microsoft/blog/275475/

25 Фактори що впливають на функціональний стан користувачів комп'ютерів [Електронний ресурс]: https://studopedia.org/4-188366.html

26 Правил безпечної експлуатації електроустановок споживачів [Електронний ресурс]: https://zakon.rada.gov.ua/laws/show/z0093-98

ДОДАТКИ

Додаток А

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
Факультет комп'ютерно-інформаційних систем і програмної інженерії
Кафедра програмної інженерії

ЗАТВЕРДЖУЮ
Завідувач кафедрою
програмної інженерії

"___ "_____ 2019 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської дипломної роботи

на тему: «Розробка програмної системи керування контентом із використанням ASP.NET WEB API»

Сороцькій Наталі Тарасівній

Керівник роботи:
к.т.н, доцент Кінах Я.І.
"___" _____ 2019р.
_____

Виконавець:
студентка групи СПм-62
Сороцька Наталя Тарасівна
"___" _____ 2019р.
_____

м. Тернопіль – 2019

ЗМІСТ

1 ПІДСТАВИ ДО РОЗРОБКИ

Розробка проводиться у відповідності до графіку навчального плану на 2019 рік, та згідно наказу на виконання дипломної роботи студента-магістра.

Тема проекту: «Розробка програмної системи керування контентом із використанням ASP.NET WEB API».

2 ПРИЗНАЧЕННЯ РОЗРОБКИ

Поява web-технологій відіграла важливу роль у формуванні сьогоденного світу інформаційних технологій. Сучасні web-ресурси надають їх користувачам широкий спектр можливостей, пов’язаних з інформацією. Головним завданням будь-якого інтернет-сервісу є інформаційне забезпечення своїх користувачів.

Метою даної роботи є проектування та реалізація програмної системи керування контентом, тобто web-сервіс кулінарних рецептів та проектування подальшого його покращення за допомогою нейронної мережі. Для досягнення поставленої мети необхідно дослідити проблему та шляхи її вирішення, провести аналіз предметної області та розглянути принципи роботи web-сервісів та нейронних мереж.

Проаналізувати можливі способи взаємодії програмних клієнтів з сервером через програмний інтерфейс. Скласти вимоги до функціоналу web-сервісу та спроектувати програмний інтерфейс сервісу для взаємодії з зовнішніми системами (API). Спроектувати модель бази даних і структуру сервісу,а також провести їх реалізацію та тестування. Спроектувати основні функції нейронної мережі для даного веб-сервісу

3 ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

3.1 Функціональні характеристики

Програмне забезпечення має виконувати наступні дії:

- додавання рецептів;
- редагування рецептів;
- видалення рецептів;
- читання та пошук рецептів та інгредієнтів;
- додавання інгредієнтів.
- створнення коментарів;
- редагування коментарів;
- видалення коментарів;
- додавання рецептів у список улюблених.
- реєстрація;
- авторизація.

3.2 Склад та параметри технічних засобів

1) ПК, 8GB оперативної пам’яті, встановленою системою Windows 10, 64-бітна система, Visual Studio, SQLServer,. Не менше 200 Мб вільного місця на жорсткому диску. Двоядерний процесор з тактовою частотою від 2.49 GHz і більше.

3.3 Інформаційна та програмна сполучність

Програмний продукт повинен коректно функціонувати в операційній системі Windows, на яких доступне для встановлення SQL Manager. Розроблювана бібліотека класів повинна бути пристосована до використання у інформаційних системах та програмних засобах. Розробку виконувати з використанням бібліотек

та технологій мови C# в середовищі програмування Visual Studio 2015 з використанням технології ASP.NET WebAPI.

4. СТАДІЇ РОЗРОБКИ

В ходів реалізації роботи проект повинен пройти крізь наступні стадії розробки:

— аналіз предметної області;
— проектування основних задач;
— проектування архітектури;
— реалізація класів програмної системи;
— реалізація візуального представлення веб-сервісу;
— тестування результатів розробки;
— Проектування нейронної мережі
— оформлення супровідної документації;
— здача роботи.

5. ПРОГРАМНА ДОКУМЕНТАЦІЯ

Для програмного продукту повинні бути розроблені наступні документи:

— Пояснювальна записка;
— Технічне завдання;
— Презентаційний матеріал;
— Додатки.

6. ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Розроблений програмний продукт має виконувати всі вимоги, що складаються з перерахованих у п. 3.1 характеристик.

Приймання проводиться спеціально створеною екзаменаційною комісією в термін до:

“__” лютого 2019р.

Додаток В

УДК 658.012.011.56:681.3.06
Н.Т. Сороцька – магістрантка
Тернопільський національний технічний університет імені Івана Пулюя, Україна
Я.І. Кінах – кандидат технічних наук, доцент
Тернопільський національний технічний університет імені Івана Пулюя, Україна

ПРОГРАМНА СИСТЕМА КЕРУВАННЯ КОНТЕНТОМ ІЗ ВИКОРИСТАННЯМ ТЕХНОЛОГІЇ ASP.NET WEB API

N.T. Sorotska – magistrate, I.I. Kinakh – Ph.D, Assoc. Prof.

SOFTWARE CONTROL SYSTEM USING ASP.NET WEB API TECHNOLOGY

Для вирішення проблеми програмного керування контентом може бути декілька способів, наприклад: складання бази даних власноручно, або ж пошук по існуючим сайтам рецептів. Так само розробник зможе написати парсер сайтів, який буде наповнювати базу даних в автоматичному режимі. Найкращим виходом в цій ситуації є використання сайтів з існуючими програмними інтерфейсами для доступу до потрібної розробнику інформації, за допомогою якого його сайт або мобільна аплікація зможе здійснювати пошук контентк у вже існуючій базі даних.

Так само перед розробником системи може бути інша проблема – міграція сайту з однієї платформи на іншу, у випадку якщо початкова платформа не в змозі забезпечити доволі просте розширення функціоналу, редизайн, або ж розробка додаткового програмного продукту, який повинен використовувати ту ж саму базу даних. У випадку, якщо відбувається міграція сайту з однієї платформи на іншу - це призводить до переписування серверного коду сайту під цільову платформу. Або ж у випадку написання мобільного додатку виникає потреба забезпечити функціонал для взаємодії з системою що призводить до дублювання вже інсуючого функціоналу. Шляхом вирішення цієї проблеми може бути web-сервіс, який інкапсулює в собі всю необхідну бізнес логіку для створення, збереження, читання, пошуку, редагування та видалення інформації. Це дозволить безболісно проводити зміну дизайну сайту, оскільки бізнес логіка буде відокремлена від користувацької браузерної частини. Так само це дозволить мобільній аплікації в повній мірі використовувати той самий функціонал, що і сайту, уникаючи його дублювання.

Web-сервіс, який буде спроектований технологією ASP.NET Web API. Під час виконання проекту надаватиме стороннім сервісам змогу використовувати свій функціонал шляхом взаємодії через програмний інтерфейс. Це позбавляє розробників необхідності створювати власну базу даних, піклуватись про її цілісність та внутрішню будову. Web-сервіс також забезпечує використання спільної бази знань для усіх клієнтів, що дозволяє уникнути проблем з пошуком та дублюванням інформації. Для роботи web-сервісу необхідна база даних, яка дозволить зберігати дані, отримані від сторонніх систем і які будуть в подальшому оброблятись цими системами.

Також необхідно спроектувати програмний інтерфейс, завдяки якому буде відбуватись взаємодія між web-сервісом та сторонніми системами. Для проектуваня програмного інтерфейсу необхідно визначити функціонал,який буде надаватись цим web-сервісом.

Література

1. Learn About ASP.NET Web API [Електронний ресурс]: режим доступу: http://www.asp.net/web-api
2. Юдін О.К. Кодування в інформаційно-комунікаційних мережах: – Монографія. - К.:НАУ,. 2007.- 308с.

Додаток С

Лістинг контролера роботи з акаунтами

```
    public class AccountsController : BaseApiController
    {
    [HttpPost]
    [Route("api/accounts/register")]
    public async Task<IHttpActionResult>
Register(UserRegisterBindingModel model)
    {
    if (!ModelState.IsValid)
    {
    return BadRequest(ModelState);
    }

    var user = new ApplicationUser()
    {
    UserName = model.UserName,
    Email = model.Email,
    EmailConfirmed = true
    };

    IdentityResult registerResult = UserManager.Create(user,
model.Password);

    if (registerResult.Succeeded)
    {
    var role = await RoleManager.FindByNameAsync("User");
    var applyRoleResult = await UserManager.AddToRoleAsync(user.Id,
role.Name);

    if (applyRoleResult.Succeeded)
    {
    return Ok(string.Format("User has been created with role \"{0}\"",
role.Name));
    }
    else
    {
    return Ok("User has been created without any roles");
    }
    }
    else
    {
    var errors = "Errors: " + registerResult.Errors.Count();

    foreach (var error in registerResult.Errors)
    {
```

```
    errors += "\n" + error;
    }
    return BadRequest(errors);
    }
    }

    [HttpGet]
    [Authorize]
    [Route("api/validateToken")]
    public async Task<IHttpActionResult> ValidateToken()
    {
    return Ok();
    }
    }
    }
```

Лістинг основного контролера

```
    public class BaseApiController : ApiController
    {
    protected ApplicationUserManager UserManager
    {
    get { return
Request.GetOwinContext().GetUserManager<ApplicationUserManager>(); }
    }

    protected ApplicationRoleManager RoleManager
    {
    get { return
Request.GetOwinContext().GetUserManager<ApplicationRoleManager>(); }
    }

    public ApplicationUser CurrentUser
    {
    get
    {
    var username =
Request.GetRequestContext().Principal.Identity.Name;
    ApplicationUser user = UserManager.FindByName(username);
    return user;
    }
    }

    private readonly TastyThingsDbContext _dbContext = new
TastyThingsDbContext();

    protected TastyThingsDbContext DbContext
    {
```

```
    get { return _dbContext; }
    }
    }
    }
```

Лістинг контролера роботи з коментарями

```
    public class FeedbacksController : BaseApiController
    {
    public async Task<IHttpActionResult>
CreateFeedbacks(CreateFeedbackBindingModel model)
    {
    if (ModelState.IsValid)
    {
    var context = new TastyThingsDbContext();
    bool isRecipeExisted = await context.Recipes.AnyAsync(x =>
x.RecipeId == model.RecipeId);

    if (!isRecipeExisted)
    {
    return BadRequest();
    }

    var feedback = new Feedback()
    {
    RecipeId = model.RecipeId,

    Rate = model.Rate,
    Title = model.Title,
    Text = model.Text
    };
    context.Feedbacks.Add(feedback);
    await context.SaveChangesAsync();

    return Ok(feedback);
    }
    else
    {
    return BadRequest(ModelState);
    }
    }
    }
    }
```

Лістинг контролера роботи з інгредієнтами

```
    [RoutePrefix("api/ingredients")]
    public class IngredientsController : BaseApiController
    {
```

```
    [HttpPost]
    [Authorize]
    public async Task<IHttpActionResult>
CreateIngredient(CreateIngredientBindingModel model)
    {
    if (ModelState.IsValid)
    {
    var context = new TastyThingsDbContext();
    var ingredient = new Ingredient() {Name = model.Name, Description
= model.Description};
    context.Ingredients.Add(ingredient);
    await context.SaveChangesAsync();
    return Ok(ingredient);
    }
    else
    {
    return BadRequest(ModelState);
    }
    }

    [HttpGet]
    public async Task<IHttpActionResult> GetAllIngredients()
    {
    var ingredients = await DbContext.Ingredients.ToArrayAsync();
    return Ok(ingredients);
    }
    }
    }
```

Лістинг контролера роботи з рецептами

```
    public class RecipesController : BaseApiController
    {

    [HttpPost]
    [Route("api/recipes")]
    [Authorize(Roles = "User")]
    public async Task<IHttpActionResult>
CreateRecipeAsync(CreateRecipeBindingModel model)
    {
    if (!ModelState.IsValid)
    {
    return BadRequest();
    }

    var user = CurrentUser;

    var recipe = new Recipe()
    {
```

```
    UserId = user.Id,
    Name = model.Name,
    Description = model.Description
    };

    var ingredients = DbContext.Ingredients;

    recipe.IngredientPortions = (from ingredientPortionBindingModel in
model.IngredientPortions
    where ingredients.Any(ingredient => ingredient.IngredientId ==
ingredientPortionBindingModel.IngredientId)
    join ingredient in ingredients
    on ingredientPortionBindingModel.IngredientId equals
ingredient.IngredientId
    select new IngredientPortion
    {
    Ingredient = ingredient,
    IngredientId = ingredient.IngredientId,
    RecipeId = recipe.RecipeId,
    Weight = ingredientPortionBindingModel.Weight
    }).ToArray();

    DbContext.Recipes.Add(recipe);
    await DbContext.SaveChangesAsync();

    var message = string.Format("{1}: Recipe created: {0}",
recipe.RecipeId, user.UserName);
    return Ok(message);
    }

    [HttpPost]
    [Route("api/favoriterecipes/{recipeId}")]
    [Authorize(Roles = "User")]
    public async Task<IHttpActionResult>
AddRecipeToFavoritesAsync(string recipeId)
    {
    ApplicationUser user = CurrentUser;
    Recipe selectedRecipe = await
DbContext.Recipes.FirstOrDefaultAsync(recipe => recipe.RecipeId ==
recipeId);
    FavoriteRecipe favoriteRecipe = await
DbContext.FavoriteRecipes.FirstOrDefaultAsync(fvoriteRecipe =>
fvoriteRecipe.UserId == user.Id && fvoriteRecipe.RecipeId ==
recipeId);

    bool alreadyInFavorite = favoriteRecipe != null;
    bool recipeExists = selectedRecipe != null;
```

```
    if (alreadyInFavorite)
    {
    if (recipeExists)
    {
    return BadRequest("This recipe is already favorite");
    }
    return BadRequest("This recipe is already removed");
    }

    if (!recipeExists)
    {
    return BadRequest("Recipe not found");
    }

    favoriteRecipe = new FavoriteRecipe()
    {
    UserId = user.Id,
    Recipe = selectedRecipe
    };

    DbContext.FavoriteRecipes.Add(favoriteRecipe);
    await DbContext.SaveChangesAsync();

    return Ok();
    }

    [HttpDelete]
    [Route("api/favoriterecipes/{recipeId}")]
    [Authorize(Roles = "User")]
    public async Task<IHttpActionResult>
DeleteRecipeFromFavoritesAsync(string recipeId)
    {
    ApplicationUser user = CurrentUser;
    Recipe selectedRecipe = await
DbContext.Recipes.FirstOrDefaultAsync(x => x.RecipeId == recipeId);
    FavoriteRecipe favoriteRecipe = await
DbContext.FavoriteRecipes.FirstOrDefaultAsync(x => x.UserId ==
user.Id && x.RecipeId == recipeId);

    bool alreadyInFavorite = favoriteRecipe != null;
    bool recipeExists = selectedRecipe != null;

    if (!alreadyInFavorite)
    {
    if (recipeExists)
    {
    return BadRequest("This recipe is not favorite");
    }
```

```
    return BadRequest("Original recipe not found");
    }

    if (!recipeExists)
    {
    return BadRequest("Original recipe not found");
    }

    DbContext.FavoriteRecipes.Remove(favoriteRecipe);
    await DbContext.SaveChangesAsync();

    return Ok();
    }

    [HttpGet]
    [Route("api/recipes")]
    public async Task<IHttpActionResult> GetRecipesAsync(int take, int
skip = 0)
    {
    if (take <= 0 || skip < 0)
    {
    return BadRequest(string.Format("{0} out of range",
take<=0?"take": "skip"));
    }

    bool nonAuthorized = CurrentUser == null;

    IEnumerable<Recipe> recipes = await
DbContext.Recipes.OrderByDescending(recipe =>
recipe.CreationDateTicks).Skip(skip).Take(take).ToArrayAsync();

    IEnumerable<Tuple<Recipe, bool>> recipesWithFavorieFlag = null;

    if (nonAuthorized)
    {
    recipesWithFavorieFlag = recipes.Select(recipe => new
Tuple<Recipe, bool>(recipe, false));
    }
    else
    {
    IQueryable<FavoriteRecipe> favoriteRecipes =
DbContext.FavoriteRecipes.Where(favoriteRecipe =>
favoriteRecipe.UserId == CurrentUser.Id);
    recipesWithFavorieFlag = (from recipe in recipes
    join favoriteRecipe in favoriteRecipes
    on recipe.RecipeId equals favoriteRecipe.RecipeId into frs
    from tmpfr in frs.DefaultIfEmpty()
    select new Tuple<Recipe, bool>(recipe, tmpfr != null));
```

```
    }

    var userIds = recipes.Select(recipe => recipe.UserId).Distinct();
    var users = UserManager.Users;

    var userViewModels = from userId in userIds
    join user in users
    on userId equals user.Id
    select new UserViewModel()
    {
    UserId = user.Id,
    UserName = user.UserName,
    Email = user.Email
    };

    var recipeViewModels = from recipeTuple in recipesWithFavorieFlag
    join user in userViewModels
    on recipeTuple.Item1.UserId equals user.UserId
    select
    new RecipeViewModel()
    {
    RecipeId = recipeTuple.Item1.RecipeId,
    User = user,
    Name = recipeTuple.Item1.Name,
    Description = recipeTuple.Item1.Description,
    IngredientPortions =
recipeTuple.Item1.IngredientPortions.Select(ingredientportion => new
IngredientPortionViewModel()
    {
    Ingredient = new IngredientViewModel()
    {
    IngredientId = ingredientportion.Ingredient.IngredientId,
    Name = ingredientportion.Ingredient.Name,
    Description = ingredientportion.Ingredient.Description,
    },
    Weight = ingredientportion.Weight
    }),
    CreationDate = new DateTime(recipeTuple.Item1.CreationDateTicks),
    IsFavorite = recipeTuple.Item2
    };

    return Ok(recipeViewModels);
    }

    [HttpGet]
    [Route("api/recipes")]
    public async Task<IEnumerable<RecipeViewModel>> GetRecipesAsync()
    {
```

```
    throw new NotImplementedException();
    }

    private async Task<IEnumerable<RecipeViewModel>>
CreateViewModelsAsync(IEnumerable<Recipe> recipes)
    {
    string[] userIds = recipes.Select(recipe =>
recipe.UserId).Distinct().ToArray();

    UserViewModel[] userViewModels = await
UserManager.Users.Where(user => userIds.Contains(user.Id))
    .Select(user => new UserViewModel()
    {
    UserName = user.UserName,
    UserId = user.Id,
    Email = user.Email
    }).ToArrayAsync();

    IEnumerable<RecipeViewModel> recipeViewModels =
recipes.Select(recipe =>
    new RecipeViewModel()
    {
    RecipeId = recipe.RecipeId,
    User = userViewModels.FirstOrDefault(userViewModel =>
userViewModel.UserId == recipe.UserId),
    CreationDate = new DateTime(recipe.CreationDateTicks),
    Name = recipe.Name,
    Description = recipe.Description,

    IngredientPortions = recipe.IngredientPortions
    .Select(ip =>
    new IngredientPortionViewModel()
    {
    Ingredient = new IngredientViewModel()
    {
    IngredientId = ip.Ingredient.IngredientId,
    Name = ip.Ingredient.Name,
    Description = ip.Ingredient.Description
    },
    Weight = ip.Weight,
    })
    .ToArray()
    });
    return recipeViewModels;
    }
    [HttpGet]
    [Route("api/favoriterecipes")]
    [Authorize]
```

```
    public async Task<IHttpActionResult> GetFavoriteRecipesAsync(int
take, int skip = 0)
    {
    if (take <= 0 || skip < 0)
    {
    return BadRequest();
    }

    var favoriteRecipes = await DbContext.FavoriteRecipes
    .Where(favoriteRecipe => favoriteRecipe.UserId == CurrentUser.Id)
    .OrderByDescending(favoriteRecipe =>
favoriteRecipe.Recipe.CreationDateTicks)
    .Skip(skip)
    .Take(take)
    .ToArrayAsync();
    var userIds = favoriteRecipes
    .Select(favoriteRecipe => favoriteRecipe.Recipe.UserId)
    .Distinct()
    .ToArray();
    var userViewModels = await UserManager.Users.Where(user =>
userIds.Contains(user.Id))
    .Select(user => new UserViewModel()
    {
    UserName = user.UserName,
    UserId = user.Id,
    Email = user.Email
    })
    .ToArrayAsync();

    var recipes = favoriteRecipes
    .Select(favoriteRecipe => favoriteRecipe.Recipe);

    var recipeViewModels = recipes.Select(recipe =>
    new RecipeViewModel()
    {
    RecipeId = recipe.RecipeId,
    User = userViewModels.FirstOrDefault(userViewModel =>
userViewModel.UserId == recipe.UserId),
    CreationDate = new DateTime(recipe.CreationDateTicks),
    Name = recipe.Name,
    Description = recipe.Description,
    IngredientPortions = recipe.IngredientPortions
    .Select(ip =>
    new IngredientPortionViewModel()
    {
    Ingredient = new IngredientViewModel()
    {
    IngredientId = ip.Ingredient.IngredientId,
```

```
    Name = ip.Ingredient.Name,
    Description = ip.Ingredient.Description
    },
    Weight = ip.Weight,
    })
    .ToArray()
    });

    return Ok(recipeViewModels);
    }

    [HttpGet]
    [Route("api/users/{userId}/recipes")]
    [Authorize]
    public async Task<IHttpActionResult> GetUsersRecipesAsync(string
userId, int take, int skip = 0)
    {
    if (take <= 0 || skip < 0)
    {
    return BadRequest();
    }

    userId = CurrentUser.Id;

    var recipes = await DbContext.Recipes
    .Where(x => x.UserId == userId)
    .OrderByDescending(x => x.CreationDateTicks)
    .Skip(skip)
    .Take(take).ToArrayAsync();

    var recipeViewModels = await CreateViewModelsAsync(recipes);

    return Ok(recipeViewModels);
    }
    [HttpDelete]
    [Route("api/recipes/{recipeId}")]
    [Authorize(Roles = "User")]
    public async Task<IHttpActionResult> DeleteRecipe(string recipeId)
    {
    string currentUserId = CurrentUser.Id;
    var recipe = await DbContext.Recipes.FirstOrDefaultAsync(x =>
x.RecipeId == recipeId && x.UserId == currentUserId);

    if (recipe == null)
    {
    return BadRequest("Recipe not found");
    }
    else
```

```
    {
    IQueryable<FavoriteRecipe> allRelatedFavoriteRecipes =
DbContext.FavoriteRecipes.Where(favoriteRecipe =>
favoriteRecipe.RecipeId == recipeId);
    DbContext.FavoriteRecipes.RemoveRange(allRelatedFavoriteRecipes);

    DbContext.Recipes.Remove(recipe);
    await DbContext.SaveChangesAsync();
    return Ok(string.Format("Recipe with id: {0} was removed",
recipeId));
    }}}}
```

Додаток D