Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Іншина Владислава Дмитровича*
(ПІБ)

академічної групи *122-18-1*
(шифр)

спеціальності *122 Комп'ютерні науки*
(код і назва спеціальності)

освітньої програми *Комп'ютерні науки*
(назва освітньої програми)

на тему:

Розробка веб-додатку «EDA» для організації та контролю за харчуванням людини

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*доц. Кабак Л.В.*			
розділів:				
спеціальний	*доц. Кабак Л.В.*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент	*доц. Шедловський І.А.*			
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2022

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 2022 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *122-18-1* *Іншина Владислава Дмитровича*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи
Розробка веб-додатку «EDA» для організації та контролю за харчуванням людини

затверджена наказом ректора НТУ «ДП» від « » 2022 р. №

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів виробничої практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі.*** ***Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2022 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПО й тривалості його розробки***	***27.05.2022 р.***

Завдання видав _____ *доц. Кабак Л.В.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Іншин В.Д.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2022 р.*

Термін подання кваліфікаційної роботи до ЕК: *13.06.2022 р.*

РЕФЕРАТ

Пояснювальна записка: 67 с., 10 рис., 3 дод., 18 джерел.

Об'єкт розробки: web-додаток для контролю харчування користувача.

Мета кваліфікаційної роботи: надати користувачу інструмент для контролю харчування за рахунок розробки та використанню відповідного веб-додатку.

У вступі розглядається аналіз та сучасний стан проблеми, конкретизується мета кваліфікаційної роботи та галузь її застосування, наведено обґрунтування актуальності теми та уточнюється постановка завдання.

У першому розділі проведено аналіз предметної області, визначено актуальність завдання та призначення розробки, розроблена постановка завдання, задані вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі виконано аналіз існуючих рішень, обрано вибір платформи для розробки, виконано проектування і розробка програми, наведено опис алгоритму і структури функціонування підсистеми, визначені вхідні і вихідні дані, наведені характеристики складу параметрів технічних засобів, описаний виклик та завантаження застосунку, описана робота програми.

В економічному розділі визначено трудомісткість розробленої інформаційної системи, проведений підрахунок вартості роботи по створенню застосунку та розраховано час на його створення.

Практичне значення полягає в наданні користувачу інструменту для контролю харчування.

Актуальність даної роботи полягає в тому, що контроль харчування є одним з найважливіших складових життєдіяльності, тому додаток, що допомагає користувачеві контролювати харчування і стежити за його правильністю, буде корисним і затребуваним. Актуальність дпного типу додатків визначається великим попитом на подібні розробки.

Список ключових слів: ВЕБ-ДОДАТОК, ANGULAR, TYPE SCRIPT, HTML, MS SQL.

ABSTRACT

Explanatory note: 67 pp., 10 fig., 3 appendices, 18 sources.

Object of development: web-application for user nutrition control.

The purpose of the qualification work: to provide the user with a tool for nutrition control through the development and use of appropriate web application.

The introduction considers the analysis and current state of the problem, specifies the purpose of the qualification work and the field of its application, provides a justification for the relevance of the topic and clarifies the problem.

In the first section the analysis of the subject area is carried out, the urgency of the task and purpose of development are defined, the statement of the task is developed, requirements to software realization, technologies and software are set.

The second section analyzes the existing solutions, selects the platform for development, performs design and development of the program, describes the algorithm and structure of the subsystem, determines the input and output data, provides characteristics of the parameters of technical means, describes the call and application load, describes the program.

The economic section determines the complexity of the developed information system, calculates the cost of work to create an application and calculates the time to create it.

Of practical importance is to provide the user with a tool to control nutrition.

The relevance of this work is that nutrition control is one of the most important components of life, so the application that helps the user to control nutrition and monitor its correctness, will be useful and in demand. The relevance of this type of application is determined by the high demand for such developments.

Keywords: WEB APP, ANGULAR, TYPE SCRIPT, HTML, MS SQL.

ЗМІСТ

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ

БД - база даних

ЕОМ - електронно-обчислювальна машина

ІС - інформаційна система

ПЗ - програмне забезпечення

СУБД - система управління базами даних

ІТ - інформаційні технології

SEO - пошукова оптимізація

ВСТУП

Правильне харчування і здорова їжа передбачає регулярність, необхідну кількість і співвідношення речовин, що надходять в організм: води, білків, жирів, вітамінів, вуглеводів, мінеральних речовин. Все це – запорука краси, довголіття, здоров'я. Відповідно недолік чи надлишок якогось із елементів провокує різні захворювання, прискорює процес старіння, негативно впливає на організм у цілому.

Користь збалансованого харчування:

• Позбавить багатьох хвороб або попередить їх.

• Стабілізує вагу без зайвих зусиль.

• Відновлює фізичну та інтелектуальну енергію.

Підсумок – гарне здоров'я, яке сприяє відмінному самопочуттю, прекрасному зовнішньому вигляду та досягненню поставлених у житті цілей.

У кваліфікаційній роботі розробляється:

• сервер, що взаємодіє з базою даних та api;

• клієнтський web-додаток-аналізатор харчування;

• база даних користувачів та інформацією про харчування.

Актуальність даної роботи полягає в тому, що контроль харчування є одним з найважливіших складових життєдіяльності, тому додаток, що допомагає користувачеві контролювати харчування і стежити за його правильністю, буде корисним і затребуваним.

Мета роботи – розробка web-додатка контролю харчування користувача. Ця програма повинна мати інтуїтивний інтерфейс користувача, надавати всі необхідні функції для обліку харчування і витрат.

З поставленої мети випливають завдання:

• створити структуру бази даних;

• створити архітектуру проекту;

• вивчити програми-конкуренти, аналоги;

• реалізувати взаємодію з api;

• реалізувати логіку пошуку та перегляду рецептів, взаємодії з поточними продуктами, аналізу індивідуальних особливостей користувача, складання плану харчування, списку покупок, ознайомлення з витратами;

• створити зручний інтерфейс користувача.

Також дослідження затребуваності додатка контролю харчування, ознайомлення з тенденціями у харчуванні сучасного користувача, вивчення способів інтерактивної взаємодії з користувачем у web-додатку.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Загальні відомості з предметної галузі

Аналізуючи функціонал та інтерфейс сайтів-аналогів, можна дійти висновку, що більшість подібних пропозицій орієнтована на спортсменів, чиє харчування та спосіб життя певною мірою відрізняється від способу життя та харчування звичайного користувача.

Крім цього, можна врахувати неадаптованість під різні країни та регіони. Найчастіше досить складно знайти пропозицію з великою кількістю не тільки готових страв і продуктів, але й з різноманітністю інгредієнтів, які досить часто є готовим варіантом харчування.

Крім того, важливим фактором є обмежений доступ до функціонала програми у безкоштовній версії та відсутність можливості взаємодіяти з контентом без реєстрації.

Додатки-аналоги:

– Lifesum. Варто відзначити приємний інтуїтивний інтерфейс програми, розібратися в ньому не складе труднощів. Дизайн та оформлення зручний та зрозумілий Розділи, шрифти та іконки приємні, а анімовані фрукти насвистують та підморгують користувачеві, додаючи інтерактивності. Але, незважаючи на всі свої плюси, Lifesum має низку певних мінусів.

- По-перше, щоденне заповнення щоденника харчування та навантажень забирає чимало часу, а функції, яка запропонує вам раціон харчування, у додатку просто немає.
- По-друге, підрахунок калорій та БЖУ дуже спрощений і схематичний. Калорійність їжі та ефективність фізичних навантажень дуже приблизні і навряд чи відповідатимуть фактичним параметрам.
- По-третє, мізерний функціонал безкоштовної версії порівняно з Premium підпискою.

– EasyFit. EasyFit - Android додаток, здатний підраховувати придбані та витрачені калорії. З його допомогою можна контролювати зміни ваги тіла. У додатку потрібно вказати стать, вік, зростання, ступінь активності та особливості поточного положення, якщо вони є: вагітність, лактацію або використання інвалідного крісла. Для отримання індивідуального плану необхідно надати інформацію про нинішню та бажану вагу. Це допоможе сформувати програму з безпечного позбавлення зайвих кілограм. Додаток містить каталог продуктів з енергоємністю. Можна шукати позиції за назвою та додавати до денного раціону. Є можливість вибрати обсяг і вагу, вказавши розмір на око, наприклад: маленька порція, півкухлі, середня скибочка і так далі. Як і біговий трекер, утиліта зберігає відомості про фізичне навантаження. Є перелік типів активності, який включає біг, плавання, аеробіку, батут, баскетбол, серфінг, більярд, ходьбу, футбол, танці та інше. Для обліку спалених калорій слід вибрати вид навантаження та час, витрачений на тренування. Додаток фіксує та аналізує дані. На графіках відображається макро статистика та денний макрос, показуючи, скільки за тиждень та добу було вжито білків, жирів та вуглеводів. Це дозволяє контролювати баланс поживних речовин. За допомогою програми можна відслідковувати прогрес ваги та талії. Утиліта надає інформацію про спалені калорії за тиждень, місяць та рік. Передбачено можливість спостерігати добовий рівень споживання води. Особливості:

- можливість відстежувати калорії у їжі;
- враховується витрати енергії на тренування;
- доступна метрична та британська системи вимірювання;
- ведеться статистика даних;
- надсилаються повідомлення про їжу, воду або вправи;
- є різні теми оформлення;
- підтримуються віджети;
- вбудована міні гра;
- додаток скачується та використовується безкоштовно;
- сумісне з актуальними версіями Android.

– YAZIO. Лічильник калорій та щоденник харчування від YAZIO це утиліта, яка пропонує багато корисного функціоналу всім користувачам, які стежать за своєю масою тіла, хочуть скинути зайву вагу та оздоровитись. Додаток багатофункціональний і пропонує «ручні» та автоматичні можливості, просунуту довідкову систему та корисні доповнення з контролю споживаних та витрачених калорій, встановлення та дотримання необхідного раціону та низку інших корисних напрацювань. Всі навігаційні властивості виконані в красивому оформленні, користувач отримує повний контроль над своїм раціоном і зможе завжди бути в курсі того, що і коли йому можна з'їсти без ризику погладшати, зробити будь-які нотатки легко і просто в будь-якому розділі. Вводьте свої особисті дані та параметри тіла у відповідні розділи, а мод на pro та повна версія платформи подбають про всі подальші розрахунки. Вводьте свій вік і стать, масу тіла зараз і бажану вагу, окружність грудей і живота. У відповідних розділах можна зафіксувати свої гастрономічні звички та побажання, а програма сама намагатиметься все продумати до дрібниць та видасть зразковий план дій на кожен день та рекомендоване меню, яке щодня буде різним. Якщо вас не влаштовують певні продукти, можна легко згенерувати новий раціон під ваші переваги або просто замінити конкретні страви. Виставляйте обмеження за калоріями і намагайтеся досягти високих результатів за конкретний період часу.

– MyFitnessPal. На сьогоднішній день програма MyFitnessPal є популярним та затребуваним мобільним інструментом, який дозволяє вирахувати правильний баланс калорій. Загальна кількість завантажень даного сервісу становить понад 100 млн разів. За твердженнями розробників програми, до неї включена найширша база продуктів харчування, що за чисельністю перевищує 6 млн найменувань. Причому поповнення цього списку здійснюється регулярно, тому корисно завантажити програму MyFitnessPal на свій мобільний пристрій. Система не лише контролює вагу конкретної людини, а й займається підбором спеціальної щоденної програми харчування. Крім того, додаток у роботі використовує дані про фізичну активність людини та її стан здоров'я. Спочатку можна знайти і скачати програму MyFitnessPal абсолютно

безкоштовно. Базова комплектація лічильника калорій містить досить прості опції – наприклад, денну норму калорій, інформацію щодо білків, вуглеводів та жирів, а також підрахунок чистих калорій. Всі інші корисні функції нажаль присутні виключно у платній версії програми, тобто за підпискою. Заплатити доведеться за рік приблизно 30 доларів, і тоді вам будуть доступні інші опції, які додадуться вже існуючим.

1.2. Призначення розробки та область застосування

Розробка web-додатку призначено для контролю харчування користувача.

Правильне харчування і здорова їжа передбачає регулярність, необхідну кількість і співвідношення речовин, що надходять в організм: води, білків, жирів, вітамінів, вуглеводів, мінеральних речовин. Ця програма через відповідний інтуїтивний інтерфейс користувача, надаватиме всі необхідні функції для обліку харчування і витрат.

1.3. Підстава для розробки

Відповідно до освітньої програми, згідно навчального плану та графіків навчального процесу, в кінці навчання студент виконує кваліфікаційну роботу. Тема роботи узгоджується з керівником проекту, випускаючою кафедрою, та затверджується наказом ректора.

Отже, підставами для розробки (виконання кваліфікаційної роботи) є:

- освітня програма спеціальності 122 «Комп’ютерні науки»;
- навчальний план та графік навчального процесу;
- наказ ректора Національного технічного університету «Дніпровська політехніка» № _____ від __.__.2022 р;
- завдання на кваліфікаційну роботу на тему: «Розробка веб-додатку «EDA» для організації та контролю за харчуванням людини».

1.4. Постановка завдання

Мета роботи – розробка web-додатку контролю харчування користувача. Ця програма повинна мати інтуїтивний інтерфейс користувача, надавати всі необхідні функції для обліку харчування і витрат.

З поставленої мети випливають завдання:

• створити структуру бази даних;

• створити архітектуру проекту;

• вивчити програми-конкуренти, аналоги;

• реалізувати взаємодію з api;

• реалізувати логіку пошуку та перегляду рецептів, взаємодії з поточними продуктами, аналізу індивідуальних особливостей користувача, складання плану харчування, списку покупок, ознайомлення з витратами;

• створити зручний інтерфейс користувача.

Також дослідження затребуваності додатка контролю харчування, ознайомлення з тенденціями у харчуванні сучасного користувача, вивчення способів інтерактивної взаємодії з користувачем у web-додатку.

Впровадження всіх необхідних функцій контролю харчування.

Супровід кроків користувача спливаючими інформаційними повідомленнями.

Реалізація логіки взаємодії з API.

Прив'язка бази даних MS SQL.

Основні розділи програми повинні бути такими:

– Домашня сторінка.

– Сторінка поточних та заборонених інгредієнтів та продуктів.

– Сторінка пошуку рецептів, перегляду інформації про рецепти, перегляду улюблених рецептів.

– Сторінка роботи з планувальником живлення.

– Сторінка роботи зі списками покупок.

– Сторінка витрат.

– Сторінка входу та реєстрації.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Система повиння мати зручний та інтуітивно зрозумілий інтерфейс та забезпечувати наведені нижче функціональні характеристики.

До авторизації користувачеві надається можливість пошуку та перегляду рецептів.

Авторизація являє собою вхід з використанням адреси електронної пошти та пароля або реєстрацію з введенням імені, адреси електронної пошти, пароля та підтвердженням пошти шляхом переходу за посиланням з листа, надісланого на вказану адресу.

Після авторизації відкривається доступ до всіх розділів.

У розділі “Fridge” (холодильник) користувачеві надається можливість взаємодіяти з поточними та забороненими гредієнтами та продуктами (додавати, видаляти, редагувати, контролювати термін зберігання).

У розділі “Recipes” (рецепти) користувачеві надається можливість пошуку, детального перегляду рецептів. Крім того, користувач може додавати рецепти в обрані для подальшого спрощеного пошуку. Пошук рецептів здійснюється на основі індивідуальних особливостей користувача (непереносимість певних гредієнтів або продуктів, алергія тощо), зазначених у списку заборонених інгредієнтів та продуктів.

У розділі “Meal Planner” (планувальник харчування) користувач може працювати з планом живлення (додавати, видаляти страви). План відображається у вигляді 7 стовпців, що відсортовані по днях від поточного. Всередині кожного стовпця відображається план харчування на день. Усі страви відсортовані за порядком, вказаним користувачем. Кожен прийом їжі (сніданок, обід, вечеря) виділяється певним фоновим кольором для зручності планування.

У розділі “Shopping List” (список покупок) користувачеві надається можливість роботи з двома списками покупок (додатковий продукт). У кожному списку можна додавати та видаляти пункти.

У розділі “Expenses” (витрати) користувачеві надається можливість ознайомлюватись із витратами (на інгредієнти, продукти). Для зручності всі витрати відображаються як плитки. Плитки показують витрати на своєчасно вжиті та прострочені продукти та інгредієнти. Таким чином, можна проаналізувати правильність планування харчування.

1.5.2 Вимоги до інформаційної безпеки

Для уникнення некоректної роботи програми необхідно реалізувати:

– семантичний та синтаксичний контроль вхідних даних;

– обробку виняткових ситуацій;

– виведення повідомлень про помилки;

– можливість повторного введення даних;

– можливість безперервної роботи протягом не менше 120 годин (5 діб);

– платформну незалежність.

Як складовий елемент системи, повязаної із торгівельною діяльністю, підсистема маршрутизації запитів повинна мати наступні характеристики:

– час відновлення після збою - 5 хв;

– час відновлення після відмови одного з елементів підсистеми не повинно перевищувати половини операційного банківського дня;

– вірогідність виникнення не більше 2 логічних помилок на 1000 операторів за 1 рік експлуатації;

– забезпечення неушкодженого стану даних, що зберігаються в базі даних, у випадку відмови підсистеми.

1.5.3 Вимоги до складу та параметрів технічних засобів

Для нормального функціонування програми необхідно, щоб обчислювальна машина, на якій буде функціонувати веб-орієнтована підсистема, відповідала наступним вимогам:

- процесор класу Intel Xeon з тактовою частотою не менш 2.4 ГГц;
- не менше 2 GB оперативної пам'яті;
- рідкокристалічний монітор з діагоналлю не менше 17";
- 20 Гб вільного місця на жорсткому диску;
- доступ до мережі Internet;
- клавіатура;
- маніпулятор "миша".

Наведені вище технічні характеристики є рекомендованими, тобто при наявності технічних засобів не нижче зазначених, розроблений програмний виріб буде функціонувати відповідно до вимог щодо надійності, швидкості обробки даних і безпеки, висунутими замовником.

Також слід зазначити що більша частина подібних розробок розміщено на серверах стороніх організацій, таким чином саме вимоги до до складу та параметрів технічних засобів не є актуальними.

З точки зору користувачів програма не є вимогливою до складу та параметрів технічних засобів та може завантажуватись на ПК, ноутбуках, планшетах чи смартфонах різних типів та конфігурацій під управлінням різних ОС. Для завантаження сторінки необхідно лише веб-браузер.

1.5.4 Вимоги до інформаційної та програмної сумісності

Програма не є вимогливою до інформаційної та програмної сумісності та може Програма не є вимогливою до складу та параметрів технічних засобів та може завантажуватись на ПК, ноутбуках, планшетах чи смартфонах різних типів

та конфігурацій під управлінням різних ОС. Для завантаження сторінки необхідно лише веб-браузер.

Додатково можна зазначити лише наступне:

Вимоги до програмних характеристик клієнта:

• Браузер, що підтримує CSS/JavaScript

Вимоги до програмних характеристик сервера:

• Підтримка MS SQL

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Функціональне призначення системи

Функції розробленого web-додатку призначено для контролю харчування користувача, також надає всі необхідні функції для обліку харчування і витрат.

Система повиння має зручний та інтуітивно зрозумілий інтерфейс та забезпечує наведені нижче функціональні характеристики.

До авторизації користувачеві надається можливість пошуку та перегляду рецептів.

Авторизація являє собою вхід з використанням адреси електронної пошти та пароля або реєстрацію з введенням імені, адреси електронної пошти, пароля та підтвердженням пошти шляхом переходу за посиланням з листа, надісланого на вказану адресу.

Після авторизації відкривається доступ до всіх розділів.

У розділі “Fridge” (холодильник) користувачеві надається можливість взаємодіяти з поточними та забороненими гредієнтами та продуктами (додавати, видаляти, редагувати, контролювати термін зберігання).

У розділі “Recipes” (рецепти) користувачеві надається можливість пошуку, детального перегляду рецептів. Крім того, користувач може додавати рецепти в обрані для подальшого спрощеного пошуку. Пошук рецептів здійснюється на основі індивідуальних особливостей користувача (непереносимість певних гредієнтів або продуктів, алергія тощо), зазначених у списку заборонених інгредієнтів та продуктів.

У розділі “Meal Planner” (планувальник харчування) користувач може працювати з планом живлення (додавати, видаляти страви). План відображається у вигляді 7 стовпців, що відсортовані по днях від поточного. Всередині кожного стовпця відображається план харчування на день. Усі страви відсортовані за

порядком, вказаним користувачем. Кожен прийом їжі (сніданок, обід, вечеря) виділяється певним фоновим кольором для зручності планування.

У розділі “Shopping List” (список покупок) користувачеві надається можливість роботи з двома списками покупок (додатковий продукт). У кожному списку можна додавати та видаляти пункти.

У розділі “Expenses” (витрати) користувачеві надається можливість ознайомлюватись із витратами (на інгредієнти, продукти). Для зручності всі витрати відображаються як плитки. Плитки показують витрати на своєчасно вжиті та прострочені продукти та інгредієнти. Таким чином, можна проаналізувати правильність планування харчування.

2.2. Опис застосованих математичних методів

Оскільки особливості предметної області розв'язуваної задачі не передбачають застосування математичних методів, при розробці доного додатку математичні методи не використовувалися.

Математичні методи та функції, що використовуються системою для адаптації сторінки до різних типів пристроїв, є стандартними та знаходяться у відповідних бібліотеках.

2.3. Опис використаних технологій та мов програмування

Onion-архітектура є поділом програми на рівні. Даний проект реалізує інтерфейси, оголошені на нижніх рівнях, та пов'язуватиме їх із сховищем даних. Як сховища даних використовується БД MS SQL Server, з якою взаємодіє через Entity Framework.

Ця архітектура дає можливість правильно структурувати проект і уникати накопичень файлів в одній директорії. Зв'язок між бібліотеками у проекті реалізується за допомогою посилань. Крім того, для зручності у проекті

використовується Auto Mapper, який дозволяє проектувати моделі та спрощує роботу з ними.

Angular (зазвичай так називають фреймворк Angular 2 або вищі версії) — написаний на TypeScript ront-end фреймворк з відкритим кодом, який розробляється під керівництвом Angular Team. Angular — це AngularJS, який був переосмислений та перероблений тією ж командою розробників.

TypeScript — мова програмування, представлена Microsoft восени 2012; позиціонується як засіб розробки вебзастосунків, що розширює можливості JavaScript.

Код експериментального компілятора, котрий транслює код TypeScript в представлення JavaScript, поширюється під ліцензією Apache, розробка ведеться в публічному репозиторії через сервіс CodePlex.

TypeScript є зворотньо сумісним з JavaScript. Фактично, після компіляції програму на TypeScript можна виконувати в будь-якому сучасному браузері або використовувати спільно з серверною платформою Node.js.

2.4. Опис структури системи та алгоритмів її функціонування

Для проектування інтерфейсу було використано принципи адаптивної верстки.

Інтерфейс цієї програми є панель навігації з розділами, які направляють користувача на потрібну сторінку, де надається весь необхідний функціонал.

Панель навігації доступна з кожного розділу сайту, що дозволяє швидко переміщатися між сторінками.

Всі переміщення та операції відбуваються по одинарному кліку лівої кнопки миші (табу по екрану), що спрощує взаємодію користувача з додатком.

Програма написана з використанням фреймворку Angular, містить компоненти бібліотеки Angular Material.

Сайт містить інформацію, необхідну для правильного планування харчування, аналізу якості харчування, контролю за щоденними даними харчування користувача.

Інформація про рецепти, інгредієнти та продукти надається Spoonacular API. Інформація про поточні інгредієнти та продукти, витрати, показники, заборонені складові, улюблені рецепти зберігається в базі даних, доступ до якої здійснюється при завантаженні сторінок та при оновленні інформації.

При проектуванні бази даних було створено багато таблиць для зручного зберігання та взаємодії з даними.

Діаграма таблиць бази даних наведена на рисунку 2.1.

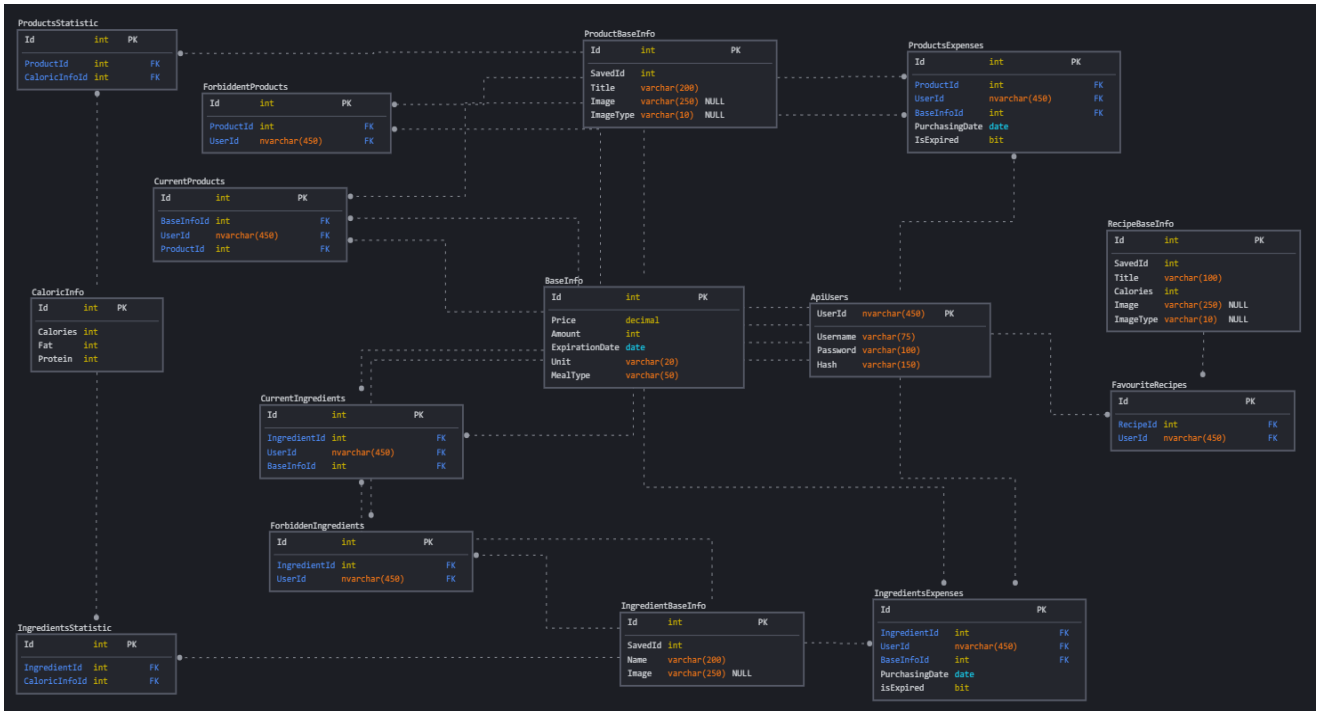

Рис. 2.1. Діаграма таблиць бази даних

Таблиці БД:

1) Таблиця ApiUsers – відомості про користувачів;

2) Таблиця BaseInfo – відомості про базову інформацію складового;

3) Таблиця ProductBaseInfo – відомості про базову інформацію товару;

4) Таблиця ProductsExpenses – відомості про витрати на продукти;

5) Таблиця ForbiddenPorducts – відомості про заборонені продукти;

6) Таблиця CurrentProducts – відомості про поточні продукти;

7) Таблиця ProductsStatistic – відомості про статистичні дані продуктів;

8) Таблиця CaloricInfo – відомості про показники складових;

9) Таблиця IngredientBaseInfo – відомості про базову інформацію інгредієнта;

10) Таблиця IngredientsExpenses – відомості про витрати на інгредієнти;

11) Таблиця ForbiddenIngredients – відомості про заборонені інгредієнти;

12) Таблиця CurrentIngredients – відомості про поточні інгредієнти;

13) Таблиця IngredientsStatistic – відомості про статистичні дані інгредієнтів;

14) Таблиця RecipeBaseInfo – відомості про базову інформацію рецепта;

15) Таблиця FavouriteRecipes – відомості про улюблені рецепти.

Завдяки сервісам клієнтської частини програми дані з бази даних, проходячи через контролери, приходять на сайт, де відображаються в необхідному форматі і обробляються користувачем.

Такий самий принцип роботи при додаванні, зміні та видаленні даних на клієнтській стороні. Всі ці дії виконуються за допомогою HTTP-запитів.

HTTP визначає безліч методів запиту, які вказують, яка бажана дія здійсниться для цього ресурсу. Кожен реалізує свою семантику, але кожна група команд поділяє загальні властивості: так, методи можуть бути безпечними, ідемпотентними або такими, що кешуються.

2.5. Обгрунтування та організація вхідних та вихідних даних програми

В якості вхідних даних системна використовує інформацію запиту від користувача. Вхідними даними також можна вважати інформацію про розмір вікна браузера для адаптації відображення сторінки на екрані пристрою, оскільки веб-додаток є адаптивним. Інформація про рецепти, інгредієнти та продукти надається Spoonacular API.

Також в якості вхідних та вихідних даних використовуються заповнена

форма користувача та результати запитів до БД.

2.6. Опис роботи розробленої системи

2.6.1. Використані технічні засоби

Програма не є вимогливою до складу та параметрів технічних засобів та може завантажуватись на ПК, ноутбуках, планшетах чи смартфонах різних типів та конфігурацій під управлінням різних ОС.

Для завантаження сторінки необхідно лише веб-браузер.

При розробці системи було використано Ноутбук Dell Vostro 15 3501 з наступними характеристиками: екран 15.6" WVA (1920x1080) Full HD, глянсовий з антивідблисковим покриттям / Intel Core i3-1005G1 (1.2 — 3.4 ГГц) / RAM 8 ГБ / SSD 256 ГБ / Intel UHD Graphics / без ОД / LAN / Wi-Fi / Bluetooth / вебкамера

Розміщення сайту організовано на завнішніх серверах за рахунок технічних засобів провайдеру.

2.6.2. Використані програмні засоби

Розроблена в рамках кваліфікаційної роботи інформаційна система була реалізована на мові HTML, CSS, TypeScript з використанням фреймворку Angular, містить компоненти бібліотеки Angular Material в середовищі Visual Studio 2019, системою керування базами даних MS SQL.

2.6.3. Виклик та завантаження програми

Спосіб виклику програми з відповідного носія даних та умови його завантаження є стандартними для запуску веб-додатків та може завантажуватись на ПК, ноутбуках, планшетах чи смартфонах різних типів та конфігурацій під управлінням різних ОС.

Веб-сайт не потребує інсталяції. Необхідно лише знати доменне ім’я та зробити до нього запит за допомогою веб-браузеру.

Для завантаження сторінки необхідно лише веб-браузер.

2.6.4. Опис інтерфейсу користувача

Веб-додаток має інтуітивно-зрозумілий інтерфейс. Інтерфейс розроблено англійською мовою для розширення аудиторії потенційних користувачів, та за для уникнення перекладу відповідних елементів, оскільки інформація про рецепти, інгредієнти та продукти надається Spoonacular API. Зображення сторінок додатку наведено на рисунках 2.2. – 2.

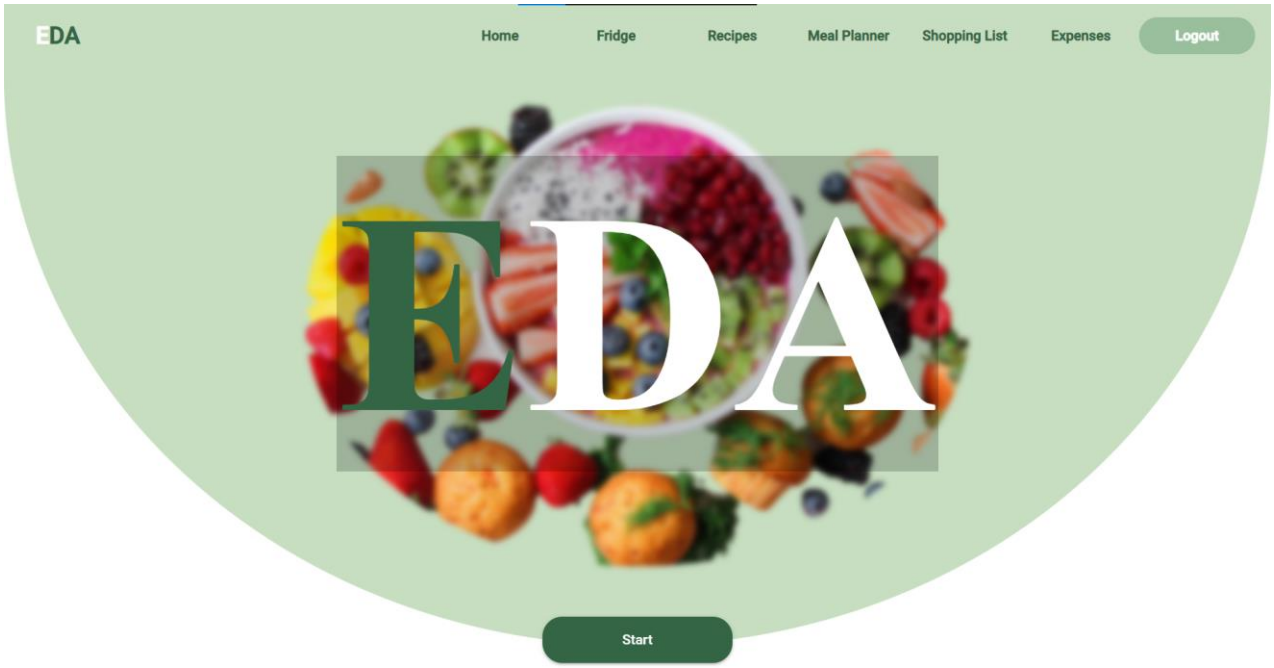

Рис. 2.2 Головна сторінка

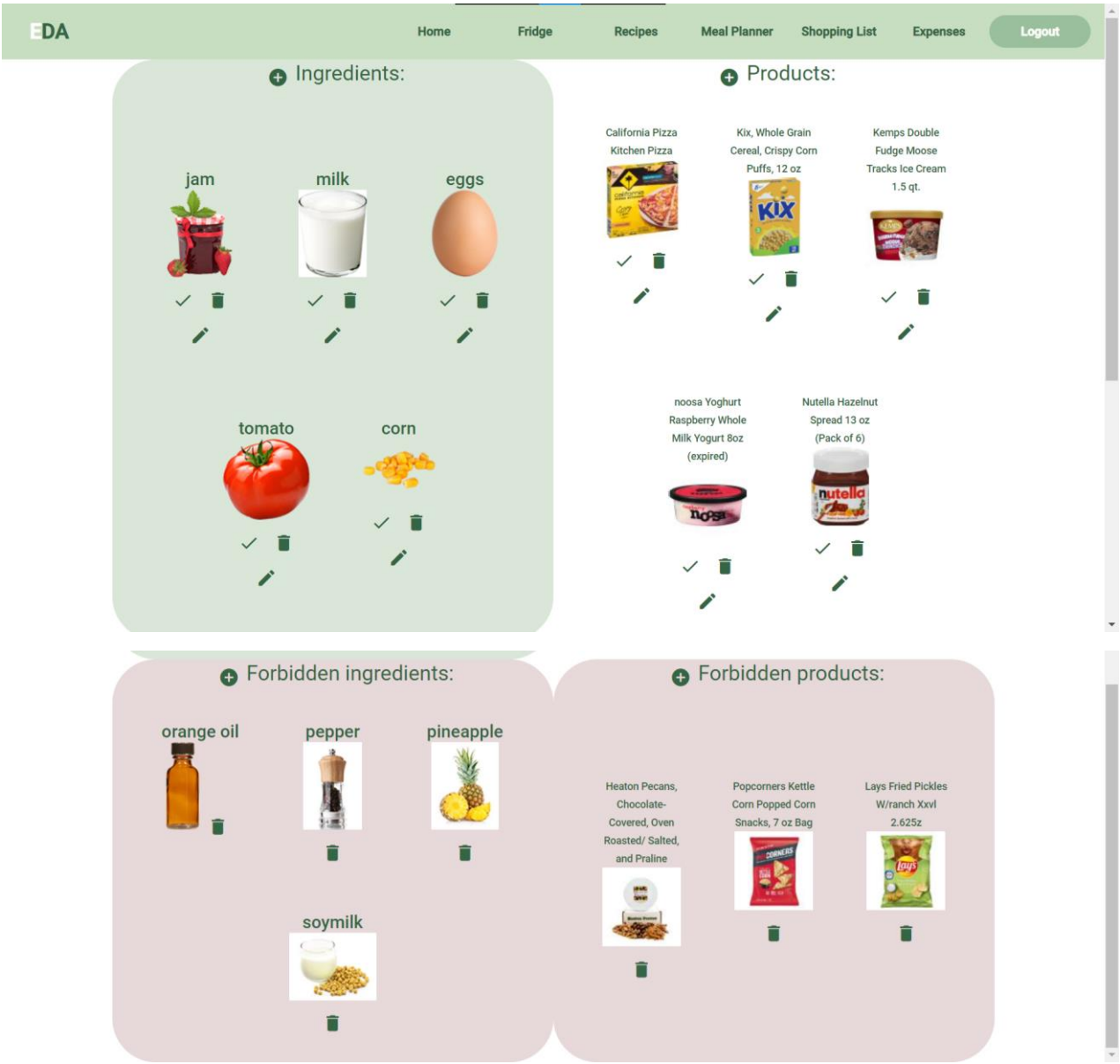

Рис. 2.3. Сорінка «Холодильник»

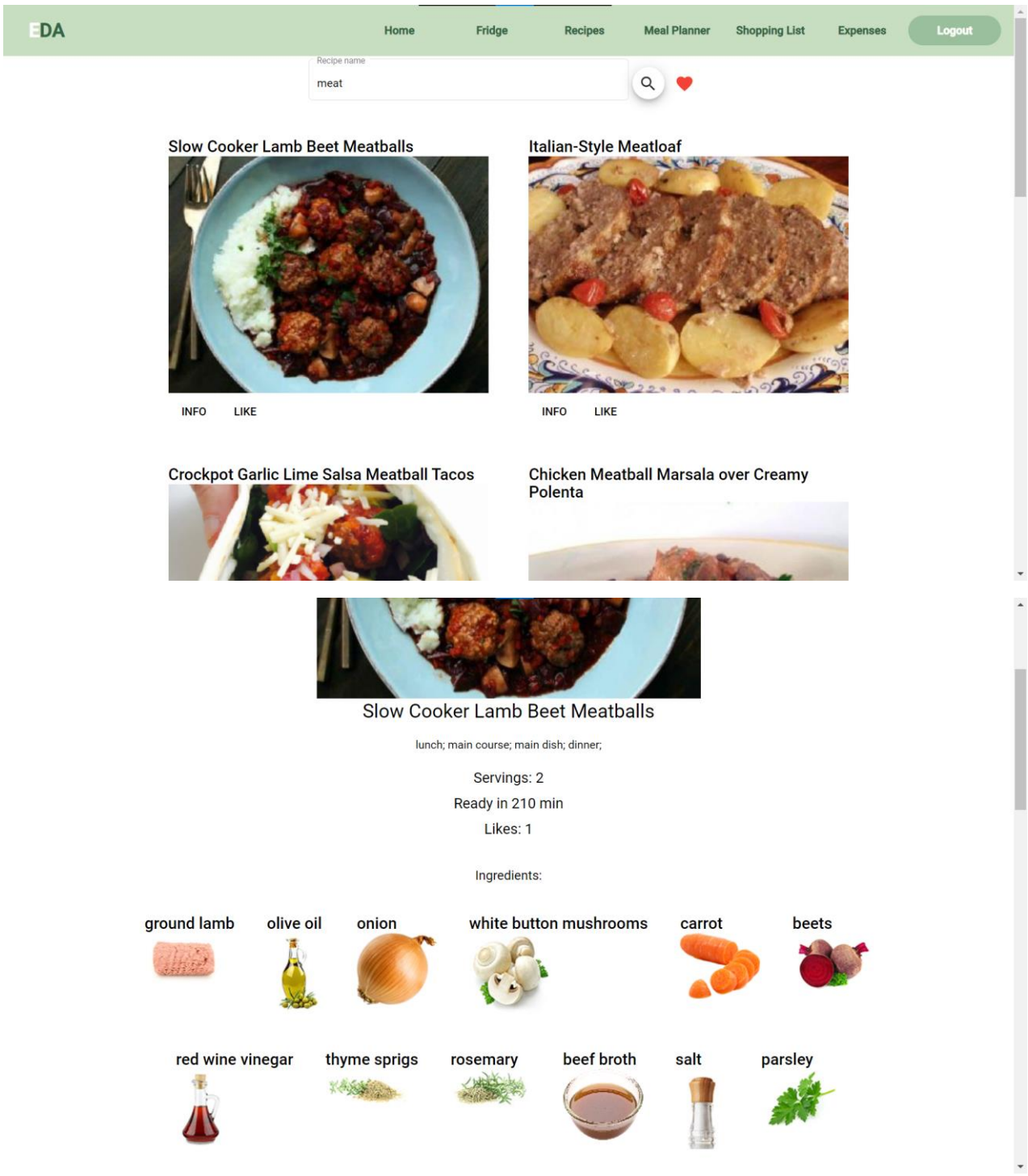

Рис. 2.4. Сторінка списку рецептів

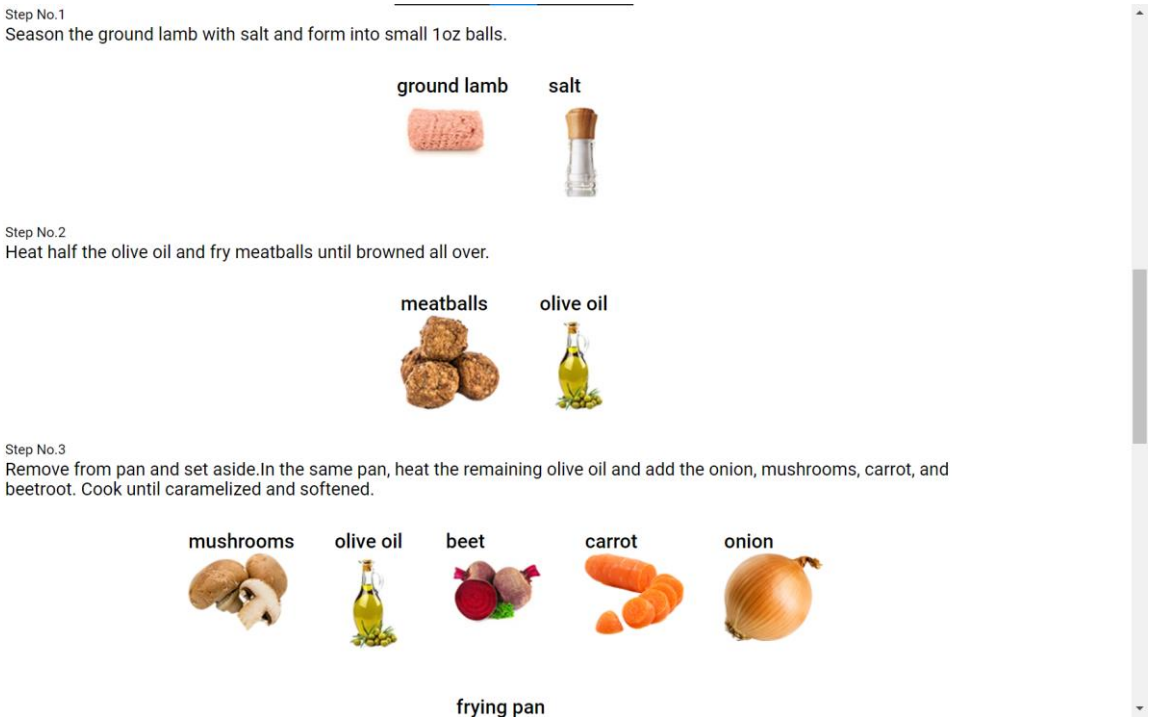

Рис. 2.5(1) Сторінка рецептів

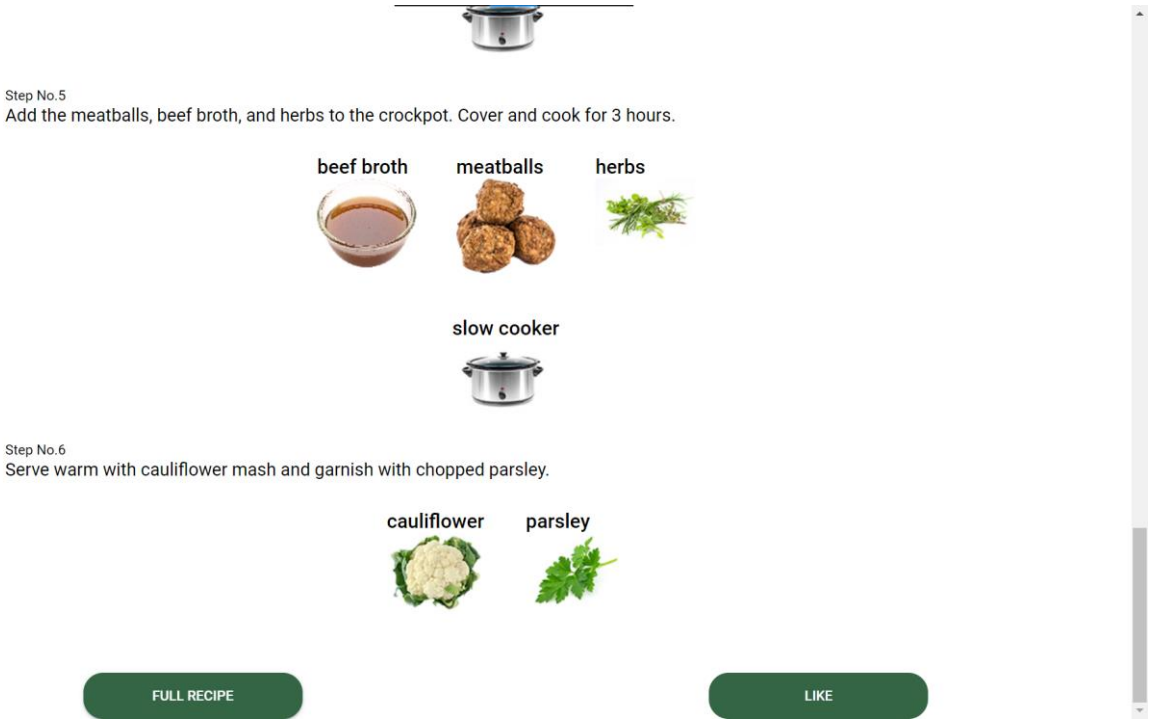

Рис. 2.5(2) Сторінка рецептів

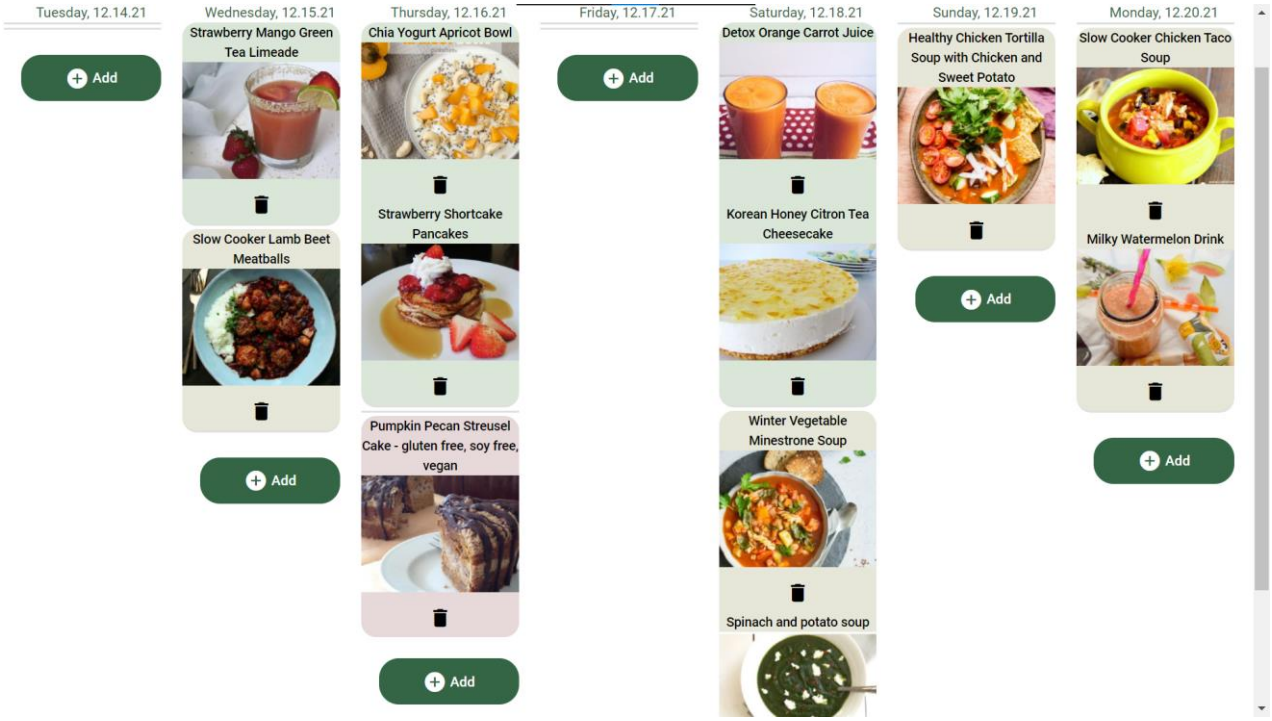

Рис. 2.6. Сторінка «Планувальник харчування»

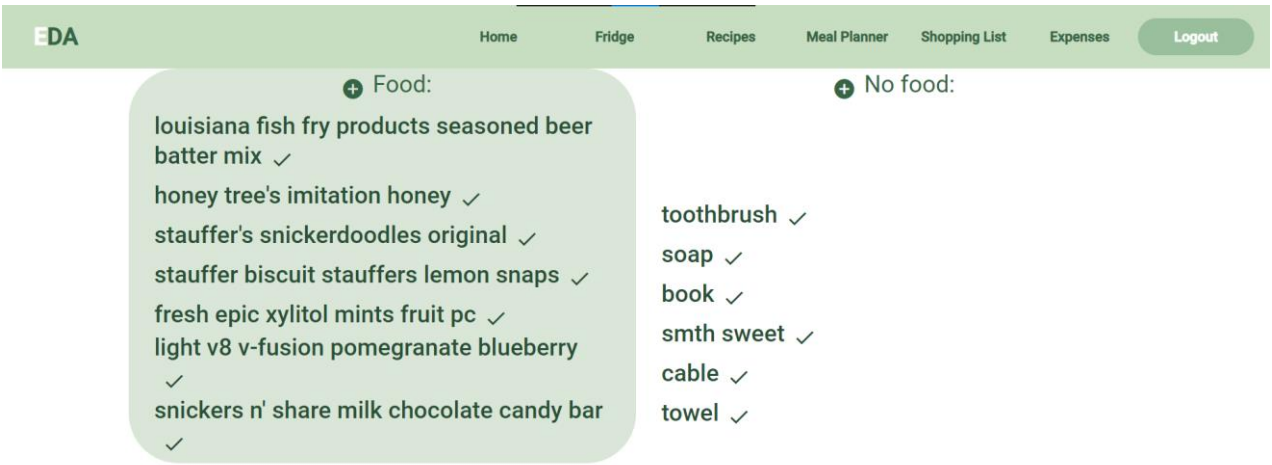

Рис. 2.7. Сторінка «Список покупок»

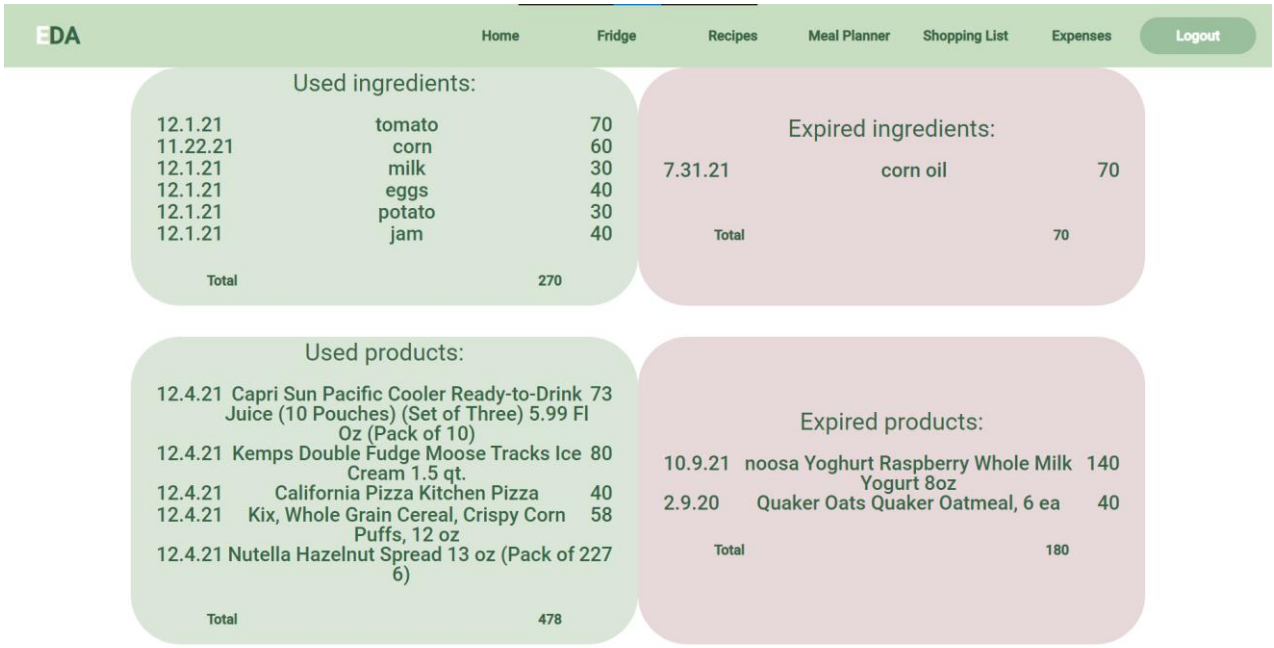

Рис. 2.8. Сторінка расходів

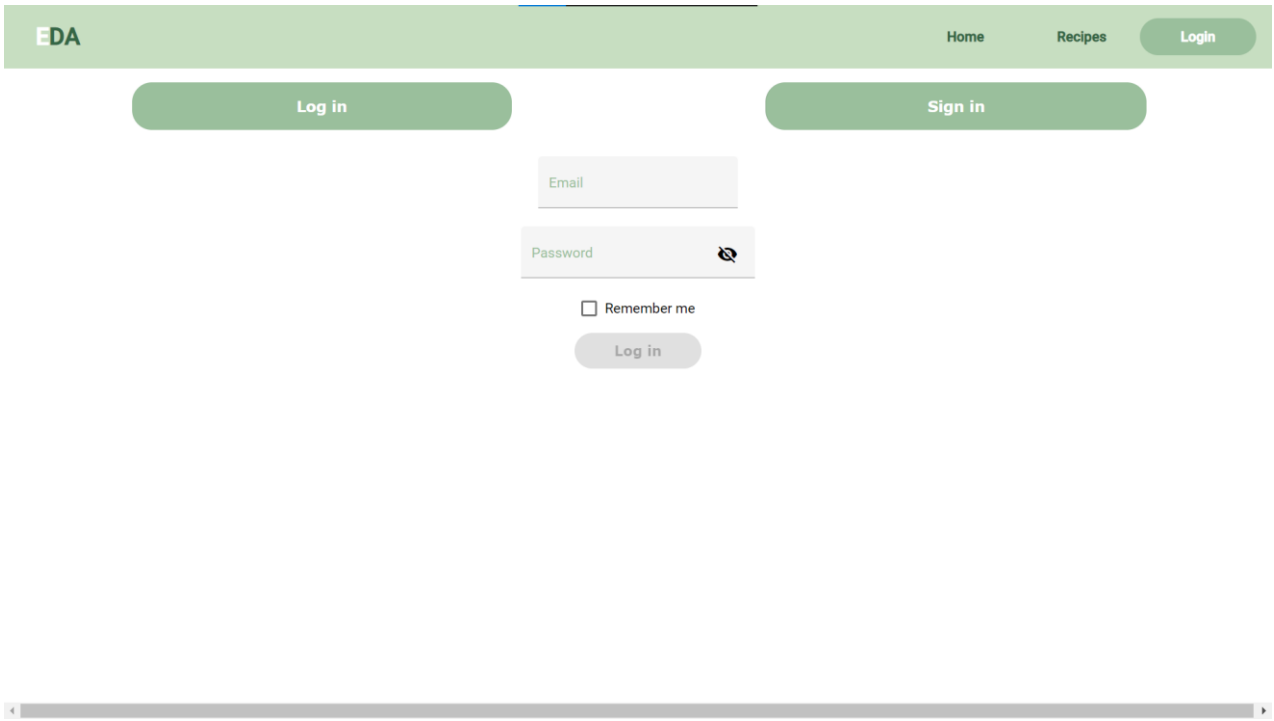

Рис. 2.9. Сторінка авторизації

РОЗДІЛ 3
ЕКОНОМІКА

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Початкові дані:

1. передбачуване число операторів програми – 1200;
2. коефіцієнт складності програми – 1,6;
3. коефіцієнт корекції програми в ході її розробки – 0,05;
4. годинна заробітна плата програміста – 120 грн/год;
5. коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,2;
6. коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1,2;
7. вартість машино-години ЕОМ – 13 грн/год.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д \text{, людино-годин,} \quad (3.1)$$

де t_o - витрати праці на підготовку й опис поставленої задачі (приймається 50 людино-годин);

t_u - витрати праці на дослідження алгоритму рішення задачі;

t_a - витрати праці на розробку блок-схеми алгоритму;

t_n - витрати праці на програмування по готовій блок-схемі;

$t_{отл}$ - витрати праці на налагодження програми на ЕОМ;

$t_д$ - витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у програмногму забезпеченні, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C \cdot (1 + p),$$

де q - передбачуване число операторів (1200);

C - коефіцієнт складності програми (1,6);

p - коефіцієнт корекції програми в ході її розробки (0,05).

Звідси умовне число операторів в програмі:

$$Q = 1{,}6 \cdot 1200 \cdot (1 + 0{,}05) = 2016$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \cdot B}{(75..85) \cdot k}, \text{ людино-годин,}$$

де B - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

k - коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності. При стажі роботи від 3 до 5 років він складає 1,2.

Приймемо збільшення витрат праці внаслідок недостатнього опису завдання не більше 50% (B = 1.2). З урахуванням коефіцієнта кваліфікації k = 1,2, отримуємо витрати праці на вивчення опису завдання:

$$t_u = (2016 \cdot 1{,}2) / (75 \cdot 1{,}2) = 26{,}88 \text{ людино-годин}$$

Витрати праці на розробку алгоритму рішення задачі визначаються за формулою:

$$t_a = \frac{Q}{(20\ldots25)\cdot k}\ \text{, людино-годин,}\quad (3.2)$$

де Q – умовне число операторів програми;

k – коефіцієнт кваліфікації програміста.

Підставивши відповідні значення в формулу (3.2), отримаємо:

$$t_a = 2016 / (20 \cdot 1{,}2) = 84 \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20\ldots25)\cdot k}\ \text{, людино-годин.}$$

$$t_n = 2016 / (25 \cdot 1{,}2) = 67{,}2 \text{ людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

- за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4..5)\cdot k}\ \text{, людино-годин.}$$

$$t_{отл} = 2016 / (5 \cdot 1{,}2) = 336 \text{ людино-годин.}$$

- за умови комплексного налагодження завдання:

$$t_{отл}^{k} = 1{,}5 \cdot t_{отл}\ \text{, людино-годин.}$$

$$t^{k}_{отл} = 1,5 \cdot 336 = 504 \text{ людино-годин.}$$

Витрати праці на підготовку документації визначаються за формулою:

$$t_{д} = t_{др} + t_{до} \text{ , людино-годин,}$$

де $t_{др}$ - трудомісткість підготовки матеріалів і рукопису:

$$t_{др} = \frac{Q}{(15..20) \cdot k} \text{ , людино-годин,}$$

$t_{до}$ - трудомісткість редагування, печатки й оформлення документації:

$$t_{до} = 0,75 \cdot t_{др} \text{ , людино-годин.}$$

Підставляючи відповідні значення, отримаємо:

$$t_{др} = 2016 / (18 \cdot 1,2) = 93,33 \text{ людино-годин.}$$

$$t_{до} = 0,75 \cdot 93,33 = 70 \text{ людино-годин.}$$

$$t_{д} = 93,33 + 70 = 163,33 \text{ людино-годин.}$$

Повертаючись до формули (3.1), отримаємо повну оцінку трудомісткості розробки програмного забезпечення:

$$t = 50 + 26,88 + 84 + 67,2 + 336 + 163,33 = 727,41 \text{ людино-годин.}$$

3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ $К_{ПО}$ включають витрати на заробітну плату виконавця програми $З_{ЗП}$ і витрат машинного часу, необхідного на налагодження програми на ЕОМ:

$$К_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн.}$$

Заробітна плата виконавців визначається за формулою:

$$З_{ЗП} = t \cdot С_{ПР}, \text{ грн,}$$

де: t - загальна трудомісткість, людино-годин;

$С_{ПР}$ - середня годинна заробітна плата програміста, грн/година

З урахуванням того, що середня годинна зарплата програміста становить 120 грн / год, отримуємо:

$$З_{ЗП} = 727{,}41 \cdot 120 = 87\,289{,}2 \text{ грн.}$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ, визначається за формулою:

$$З_{мв} = t_{отл} \cdot С_{мч}, \text{ грн,} \quad (3.3)$$

де $t_{отл}$ - трудомісткість налагодження програми на ЕОМ, год;

$С_{мч}$ - вартість машино-години ЕОМ, грн/год (13 грн/год).

Підставивши в формулу (3.3) відповідні значення, визначимо вартість необхідного для налагодження машинного часу:

$$З_{мв} = 336 \cdot 13 = 4368 \text{ грн.}$$

Звідси витрати на створення програмного продукту:

$$К_{ПО} = 87\ 289{,}2 + 4368 = 91\ 657{,}2 \text{ грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p}, \text{ міс.}$$

де B_k - число виконавців (дорівнює 1);

F_p - місячний фонд робочого часу (при 40 годинному робочому тижні F_p=176 годин).

Звідси витрати на створення програмного продукту:

$$T = 727{,}41 / 1 \cdot 176 \approx 4 \text{ міс.}$$

Висновок. Програмне забезпечення призначене для контролю харчування користувача. Вартість даного програмного забезпечення становить 91 657,2. грн. і не вимагає додаткових витрат як при розробці програми. Очікуваний час розробки становить 4 місяці. Цей термін пов'язаний зі значним числом операторів, і включає час на дослідження і розробку алгоритму вирішення поставленого завдання, програмування по готовому алгоритму, налагодження програми і підготовку документації.

ВИСНОВКИ

Створення затребуваного продукту у сфері контролю харчування користувача є відповідальним процесом, у якому важлива участь самого користувача споживача контенту. Для проектування інтерфейсу та бази даних необхідний аналіз використання програми великою кількістю користувачів.

Для правильного зв'язку сервера з клієнтським додатком було створено DTO-об'єкти, контролери, сервіси та інтерфейси, які забезпечують правильне проектування об'єктів, відображення їх на стороні клієнта та взаємодію користувача з ними.

Даний додаток відповідає всім нинішнім стандартам створення web-додатків і надає всі необхідні функції для контролю живлення користувача.

Розроблена в рамках кваліфікаційної роботи інформаційна система була реалізована на мові TypeScript з використанням фреймворку Angular, містить компоненти бібліотеки Angular Material в середовищі Visual Studio 2019, системою керування базами даних MS SQL.

Всі поставлені в роботі завдання виконано в повному обсязі.

Також у кваліфікаціфйній роботі було визначено трудомісткість розробленої системи (727 люд-год), проведений підрахунок вартості роботи по створенню програми (91 657,2 грн) та розраховано час на її створення (4 міс).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Інькова Н. А. Створення Web-сайтів: Навчально-методичний посібник [Електронний ресурс] / Інькова Н.А., Зайцева О.О., Кузьміна Н.В., Толстих С.Г. // Режим доступу: http://club-edu.tambov.ru/methodic/fio/p5.doc
2. Ланг К., Чоу Дж. Публікація баз даних в Інтернеті – СПб.: Символ-Плюс, 1998. – 206 с.
3. Ледет З. Створення Web-сайту з урахуванням бази даних // Комп'ютерПрес, 1997, N 6, з. 270-273.
4. Макфарланд Д. Велика книга CSS [Текст] / Д. Макфарланд. - СПб.: Пітер, 2010. - 512 с.
5. Режепп А. Типові помилки при створенні корпоративних web-сайтів [Текст] / О. Режепп, Ю. Степанов, О. Павлова // Світ Internet. – 2001. – № 2. – С. 70-73.
6. Шарма В., Шарма Р. Розробка Web-серверів для електронної комерції: комплексний підхід: Навчальний посібник – М.: Вільямс, 2001. – 336 с.
7. Мальчук О.В. HTML та CSS. Самовчитель/Є. В. Мальчук - М.: Вільямс, 2008. - 416 с.
8. Методичні вказівки з виконання економічного розділу у дипломних проектах студентів спеціальності «Комп'ютерні системи» / О.Г. Вагонова, О.Б. Нікітіна, Н.М. Романюк; М-во освіти та науки України, ДВНЗ «Нац. гірн. ун-т». - Д.: НГУ, 2013. - 11 с.
9. П'юрівал С. Основи розробки веб-додатків / С. П'юрівал. - СПб.: Пітер, 2015. - 272 с.
10. Скотт Б. Проектування веб-інтерфейсів /Б. Скотт, Т. Нейл. -Спб.: Символ-Плюс, 2010. - 352 с.
11. Хенік Б. HTML та CSS. Шлях до досконалості / Б. Хенік - СПб .: Пітер, 2011. - 336 с.
12. Шкріль А. А. Програмуємо для web-сайту/А.А. Шкриль - М.: Діалектика, 2006. - 368 с: іл.

13. Ayman H. Learning Website Development with Django. - Packt Publishing Ltd.: Birmingham, 2008. - 261c.

14. Фрейн Б. HTML5 та CSS3. Розробка сайтів для будь-яких браузерів та пристроїв. 2-ге вид. - СПб.: Пітер, 2017. - 272 с.

15. Фленаган Д. JavaScript. Детальний посібник, 6-е видання. - Пров. з англ. - СПб: Символ-Плюс, 2012. - 1080 с., іл.

16. https://spoonacular.com/food-api

17. https://mixsport.pro/

18. https://qalight.ua/

ДОДАТОК А

КОД ПРОГРАМИ

Лістинг сайту

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>DietAnalyzer</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
  <link rel="preconnect" href="https://fonts.gstatic.com">
  <link
href="https://fonts.googleapis.com/css2?family=Roboto:wght@300;400;500&display=swap
" rel="stylesheet">
  <link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
</head>
<body class="mat-typography">
  <app-root></app-root>
</body>
</html>
```

```
<body>
    <app-nav-menu></app-nav-menu>
    <div class="container">
      <router-outlet></router-outlet>
    </div>
  </body>
```

```
        import { enableProdMode } from '@angular/core';
        import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';

        import { AppModule } from './app/app.module';
        import { environment } from './environments/environment';

        if (environment.production) {
          enableProdMode();
        }

        platformBrowserDynamic().bootstrapModule(AppModule)
          .catch(err => console.error(err));
```

```
import { HttpClient, HttpHeaders } from "@angular/common/http";
import { Injectable } from "@angular/core";
import { environment } from "src/environments/environment";
import { ApiUser } from "../../interfaces/api-user.interface";

@Injectable()
export class ApiUsersService
{
   url: string;

   constructor(private http: HttpClient){
      this.url = environment.API_URL + '/api-users';
   }

   getApiUsers() {
      return this.http.get<ApiUser[]>(this.url, { withCredentials: true });
   }

   getApiUserById(id: string) {
      return this.http.get<ApiUser>(this.url + '/' + id, { withCredentials: true });
   }

   createApiUser(user: ApiUser) {
      var body = JSON.stringify(user);
      var headerOptions = new HttpHeaders({'Content-Type':'application/json'});

      return this.http.post(this.url, body, {
         headers: headerOptions,
         withCredentials: true
      });
   }

   deleteApiUser(id: string) {
      return this.http.delete(this.url + '/' + id, { withCredentials: true });
   }
}


import { HttpClient, HttpHeaders } from "@angular/common/http";
```

```
import { Injectable } from "@angular/core";
import { environment } from "src/environments/environment";
import { BaseInfo } from "../../interfaces/base-info.interface";

@Injectable()
export class BaseInfoService
{
   url: string;

   constructor(private http: HttpClient){
      this.url = environment.API_URL + '/base-info';
   }

   getBaseInfo() {
      return this.http.get<BaseInfo[]>(this.url, { withCredentials: true });
   }

   getBaseInfoById(id: number) {
      return this.http.get<BaseInfo>(this.url + '/' + id, { withCredentials: true });
   }

   getBaseInfoByFields(info: BaseInfo) {
      var body = JSON.stringify(info);
      var headerOptions = new HttpHeaders({'Content-Type':'application/json'});

      return this.http.post<BaseInfo>(this.url + '/fields', body, {
         headers: headerOptions,
         withCredentials: true
      });
   }

   createBaseInfo(info: BaseInfo) {
      var body = JSON.stringify(info);
      var headerOptions = new HttpHeaders({'Content-Type':'application/json'});

      return this.http.post(this.url, body, {
         headers: headerOptions,
         withCredentials: true
      });
   }

   updateBaseInfo(info: BaseInfo) {
```

```
    var body = JSON.stringify(info);
    var headerOptions = new HttpHeaders({'Content-Type':'application/json'});

    return this.http.put(this.url, body, {
      headers: headerOptions,
      withCredentials: true
    });
  }

  deleteBaseInfo(id: number) {
    return this.http.delete(this.url + '/' + id, { withCredentials: true });
  }
}


import { HttpClient, HttpHeaders } from "@angular/common/http";
import { Injectable } from "@angular/core";
import { environment } from "src/environments/environment";
import { CaloricInfo } from "../../interfaces/caloric-info.interface";

@Injectable()
export class CaloricInfoService
{
  url: string;

  constructor(private http: HttpClient){
    this.url = environment.API_URL + '/caloric-info';
  }

  getCaloricInfo() {
    return this.http.get<CaloricInfo[]>(this.url, { withCredentials: true });
  }

  getCaloricInfoById(id: number) {
    return this.http.get<CaloricInfo>(this.url + '/' + id, { withCredentials: true });
  }

  getCaloricInfoByFields(caloricInfo: CaloricInfo) {
    var body = JSON.stringify(caloricInfo);
    var headerOptions = new HttpHeaders({'Content-Type':'application/json'});

    return this.http.post<CaloricInfo>(this.url + '/fields', body, {
```

```
            headers: headerOptions,
            withCredentials: true
        });
    }

    createCaloricInfo(info: CaloricInfo) {
        var body = JSON.stringify(info);
        var headerOptions = new HttpHeaders({'Content-Type':'application/json'});

        return this.http.post(this.url, body, {
            headers: headerOptions,
            withCredentials: true
        });
    }

    updateCaloricInfo(info: CaloricInfo) {
        var body = JSON.stringify(info);
        var headerOptions = new HttpHeaders({'Content-Type':'application/json'});

        return this.http.put(this.url, body, {
            headers: headerOptions,
            withCredentials: true
        });
    }

    deleteCaloricInfo(id: number) {
        return this.http.delete(this.url + '/' + id, { withCredentials: true });
    }
}

import { HttpClient, HttpHeaders } from "@angular/common/http";
import { Injectable } from "@angular/core";
import { environment } from "src/environments/environment";
import { CurrentIngredient } from "../../interfaces/current-ingredient.interface";

@Injectable()
export class CurrentIngredientsService
{
    url: string;

    constructor(private http: HttpClient){
        this.url = environment.API_URL + '/current-ingredient';
```

```
  }

  getCurrentIngredients() {
    return this.http.get<CurrentIngredient[]>(this.url, { withCredentials: true });
  }

  getCurrentIngredientById(id: number) {
    return this.http.get<CurrentIngredient>(this.url + '/' + id, { withCredentials: true });
  }

  getCurrentIngredientByIngredientBaseInfoId(infoId: number, userId: string) {
    return this.http.get<CurrentIngredient>(this.url + '/ingr-base-info/' + infoId + '/user/' +
userId,
    { withCredentials: true });
  }

  getCurrentIngredientsByUserId(id: string) {
    return this.http.get<CurrentIngredient[]>(this.url + '/user/' + id, { withCredentials: true
});
  }

  createCurrentIngredient(ingredient: CurrentIngredient) {
    var body = JSON.stringify(ingredient);
    var headerOptions = new HttpHeaders({'Content-Type':'application/json'});

    return this.http.post(this.url, body, {
      headers: headerOptions,
      withCredentials: true
    });
  }

  updateCurrentIngredient(ingredient: CurrentIngredient) {
    var body = JSON.stringify(ingredient);
    var headerOptions = new HttpHeaders({'Content-Type':'application/json'});

    return this.http.put(this.url, body, {
      headers: headerOptions,
      withCredentials: true
    });
  }

  deleteCurrentIngredient(id: number) {
```

```
        return this.http.delete(this.url + '/' + id, { withCredentials: true });
    }
}

import { HttpClient, HttpHeaders } from "@angular/common/http";
import { Injectable } from "@angular/core";
import { environment } from "src/environments/environment";
import { CurrentProduct } from "../../interfaces/current-product.interface";

@Injectable()
export class CurrentProductsService
{
    url: string;

    constructor(private http: HttpClient){
        this.url = environment.API_URL + '/current-product';
    }

    getCurrentProducts() {
        return this.http.get<CurrentProduct[]>(this.url, { withCredentials: true });
    }

    getCurrentProductById(id: number) {
        return this.http.get<CurrentProduct>(this.url + '/' + id, { withCredentials: true });
    }

    getCurrentProductByProductBaseInfoId(infoId: number, userId: string) {
        return this.http.get<CurrentProduct>(this.url + '/prod-base-info/' + infoId + '/user/' +
userId,
        { withCredentials: true });
    }

    getCurrentProductsByUserId(id: string) {
        return this.http.get<CurrentProduct[]>(this.url + '/user/' + id, { withCredentials: true });
    }

    createCurrentProduct(product: CurrentProduct) {
        var body = JSON.stringify(product);
        var headerOptions = new HttpHeaders({'Content-Type':'application/json'});

        return this.http.post(this.url, body, {
            headers: headerOptions,
```

```
            withCredentials: true
        });
    }

    updateCurrentProduct(product: CurrentProduct) {
        var body = JSON.stringify(product);
        var headerOptions = new HttpHeaders({'Content-Type':'application/json'});

        return this.http.put(this.url, body, {
            headers: headerOptions,
            withCredentials: true
        });
    }

    deleteCurrentProduct(id: number) {
        return this.http.delete(this.url + '/' + id, { withCredentials: true });
    }
}

import { HttpClient, HttpHeaders } from "@angular/common/http";
import { Injectable } from "@angular/core";
import { environment } from "src/environments/environment";
import { FavouriteRecipe } from "../../interfaces/favourite-recipe.interface";

@Injectable()
export class FavouriteRecipesService
{
    url: string;

    constructor(private http: HttpClient){
        this.url = environment.API_URL + '/favourite-recipe';
    }

    getFavouriteRecipes() {
        return this.http.get<FavouriteRecipe[]>(this.url, { withCredentials: true });
    }

    getFavouriteRecipeById(id: number) {
        return this.http.get<FavouriteRecipe>(this.url + '/' + id, { withCredentials: true });
    }

    getFavouriteRecipesByUserId(id: string) {
```

```
    return this.http.get<FavouriteRecipe[]>(this.url + '/user/' + id, { withCredentials: true });
  }

  getFavouriteRecipeByRecipeBaseInfoId(infoId: number, curUserId: string) {
    return this.http.get<FavouriteRecipe>(this.url + '/recipe-base-info/' + infoId + '/user/' +
curUserId, { withCredentials: true });
  }

  createFavouriteRecipe(recipe: FavouriteRecipe) {
    var body = JSON.stringify(recipe);
    var headerOptions = new HttpHeaders({'Content-Type':'application/json'});

    return this.http.post(this.url, body, {
      headers: headerOptions,
      withCredentials: true
    });
  }

  updateFavouriteRecipe(recipe: FavouriteRecipe) {
    var body = JSON.stringify(recipe);
    var headerOptions = new HttpHeaders({'Content-Type':'application/json'});

    return this.http.put(this.url, body, {
      headers: headerOptions,
      withCredentials: true
    });
  }

  deleteFavouriteRecipe(id: number) {
    return this.http.delete(this.url + '/' + id, { withCredentials: true });
  }
}

import { HttpClient, HttpHeaders } from "@angular/common/http";
import { Injectable } from "@angular/core";
import { environment } from "src/environments/environment";
import { ForbiddenIngredient } from "../../interfaces/forbidden-ingredient.interface";

@Injectable()
export class ForbiddenIngredientsService
{
  url: string;
```

```
  constructor(private http: HttpClient){
    this.url = environment.API_URL + '/forbidden-ingredient';
  }

  getForbiddenIngredients() {
    return this.http.get<ForbiddenIngredient[]>(this.url, { withCredentials: true });
  }

  getForbiddenIngredientById(id: number) {
    return this.http.get<ForbiddenIngredient>(this.url + '/' + id, { withCredentials: true });
  }

  getForbiddenIngredientsByUserId(id: string) {
    return this.http.get<ForbiddenIngredient[]>(this.url + '/user/' + id, { withCredentials:
true });
  }

  getForbiddenIngredientByIngredientBaseInfoId(ingrBInfoId: number, userId: string) {
    return this.http.get<ForbiddenIngredient>(this.url + '/ingredient-base-info/' +
ingrBInfoId + '/user/' + userId,
    { withCredentials: true });
  }

  createForbiddenIngredient(ingredient: ForbiddenIngredient) {
    var body = JSON.stringify(ingredient);
    var headerOptions = new HttpHeaders({'Content-Type':'application/json'});

    return this.http.post(this.url, body, {
      headers: headerOptions,
      withCredentials: true
    });
  }

  updateForbiddenIngredient(ingredient: ForbiddenIngredient) {
    var body = JSON.stringify(ingredient);
    var headerOptions = new HttpHeaders({'Content-Type':'application/json'});

    return this.http.put(this.url, body, {
      headers: headerOptions,
      withCredentials: true
    });
```

```
    }

    deleteForbiddenIngredient(id: number) {
        return this.http.delete(this.url + '/' + id, { withCredentials: true });
    }
}

import { HttpClient, HttpHeaders } from "@angular/common/http";
import { Injectable } from "@angular/core";
import { environment } from "src/environments/environment";
import { ForbiddenProduct } from "../../interfaces/forbidden-product.interface";

@Injectable()
export class ForbiddenProductsService
{
    url: string;

    constructor(private http: HttpClient){
        this.url = environment.API_URL + '/forbidden-product';
    }

    getForbiddenProducts() {
        return this.http.get<ForbiddenProduct[]>(this.url, { withCredentials: true });
    }

    getForbiddenProductById(id: number) {
        return this.http.get<ForbiddenProduct>(this.url + '/' + id, { withCredentials: true });
    }

    getForbiddenProductsByUserId(id: string) {
        return this.http.get<ForbiddenProduct[]>(this.url + '/user/' + id, { withCredentials: true
});
    }

    getForbiddenProductByProductBaseInfoId(prodBInfoId: number, userId: string) {
        return this.http.get<ForbiddenProduct>(this.url + '/product-base-info/' + prodBInfoId +
'/user/' + userId,
        { withCredentials: true });
    }

    createForbiddenProduct(product: ForbiddenProduct) {
        var body = JSON.stringify(product);
```

```
    var headerOptions = new HttpHeaders({'Content-Type':'application/json'});

    return this.http.post(this.url, body, {
      headers: headerOptions,
      withCredentials: true
    });
  }

  updateForbiddenProduct(product: ForbiddenProduct) {
    var body = JSON.stringify(product);
    var headerOptions = new HttpHeaders({'Content-Type':'application/json'});

    return this.http.put(this.url, body, {
      headers: headerOptions,
      withCredentials: true
    });
  }

  deleteForbiddenProduct(id: number) {
    return this.http.delete(this.url + '/' + id, { withCredentials: true });
  }
}

import { HttpClient } from "@angular/common/http";
import { Injectable } from "@angular/core";
import { environment } from "src/environments/environment";
import { IngredientInfo } from "../../interfaces/ingredient-info.interface";
import { ShortIngredient } from "../../interfaces/short-ingredient.interface";

@Injectable()
export class IngredientsService
{
  url: string;

  constructor(private http: HttpClient){
    this.url = environment.API_URL + '/ingredients';
  }

  getIngredientsByName(name: string) {
    return this.http.get<ShortIngredient[]>(this.url + '/search/name/' + name,
{withCredentials: true});
  }
```

```
    getIngredientById(id: number) {
        return this.http.get<IngredientInfo>(this.url + '/search/id/' + id, {withCredentials: true});
    }
}

import { HttpClient, HttpHeaders } from "@angular/common/http";
import { Injectable } from "@angular/core";
import { environment } from "src/environments/environment";
import { IngredientBaseInfo } from "../../interfaces/ingredient-base-info.interface";

@Injectable()
export class IngredientsBaseInfoService
{
    url: string;

    constructor(private http: HttpClient){
        this.url = environment.API_URL + '/ingredient-base-info';
    }

    getIngredientsBaseInfo() {
        return this.http.get<IngredientBaseInfo[]>(this.url, { withCredentials: true });
    }

    getIngredientBaseInfoById(id: number) {
        return this.http.get<IngredientBaseInfo>(this.url + '/' + id, { withCredentials: true });
    }

    getIngredientBaseInfoByApiId(id: number) {
        return this.http.get<IngredientBaseInfo>(this.url + '/api/' + id, { withCredentials: true });
    }

    createIngredientBaseInfo(info: IngredientBaseInfo) {
        var body = JSON.stringify(info);
        var headerOptions = new HttpHeaders({'Content-Type':'application/json'});

        return this.http.post(this.url, body, {
            headers: headerOptions,
            withCredentials: true
        });
    }
```

```
  updateIngredientBaseInfo(info: IngredientBaseInfo) {
    var body = JSON.stringify(info);
    var headerOptions = new HttpHeaders({'Content-Type':'application/json'});

    return this.http.put(this.url, body, {
      headers: headerOptions,
      withCredentials: true
    });
  }

  deleteIngredientBaseInfo(id: number) {
    return this.http.delete(this.url + '/' + id, { withCredentials: true });
  }
}

import { HttpClient, HttpHeaders } from "@angular/common/http";
import { Injectable } from "@angular/core";
import { environment } from "src/environments/environment";
import { IngredientsExpense } from "../../interfaces/ingredients-expense.interface";

@Injectable()
export class IngredientsExpensesService
{
  url: string;

  constructor(private http: HttpClient){
    this.url = environment.API_URL + '/ingredients-expense';
  }

  getIngredientsExpenses() {
    return this.http.get<IngredientsExpense[]>(this.url, { withCredentials: true });
  }

  getIngredientsExpenseById(id: number) {
    return this.http.get<IngredientsExpense>(this.url + '/' + id, { withCredentials: true });
  }

  getIngredientsExpenseByIngredientBaseInfoId(infoId: number, curUserId: string) {
    return this.http.get<IngredientsExpense>(this.url + '/ingr-base-info/' + infoId + '/user/' +
curUserId,
    { withCredentials: true });
  }
```

```
  getIngredientsExpensesByUserId(id: string) {
    return this.http.get<IngredientsExpense[]>(this.url + '/user/' + id, { withCredentials: true
});
  }

  createIngredientsExpense(expense: IngredientsExpense) {
    var body = JSON.stringify(expense);
    var headerOptions = new HttpHeaders({'Content-Type':'application/json'});

    return this.http.post(this.url, body, {
      headers: headerOptions,
      withCredentials: true
    });
  }

  updateIngredientsExpense(expense: IngredientsExpense) {
    var body = JSON.stringify(expense);
    var headerOptions = new HttpHeaders({'Content-Type':'application/json'});

    return this.http.put(this.url, body, {
      headers: headerOptions,
      withCredentials: true
    });
  }

  deleteIngredientsExpense(id: number) {
    return this.http.delete(this.url + '/' + id, { withCredentials: true });
  }
}

import { HttpClient, HttpHeaders } from "@angular/common/http";
import { Injectable } from "@angular/core";
import { environment } from "src/environments/environment";
import { IngredientStatistic } from "../../interfaces/ingredient-statistic.interface";

@Injectable()
export class IngredientsStatisticService
{
  url: string;

  constructor(private http: HttpClient){
```

```
    this.url = environment.API_URL + '/ingredient-statistic';
  }

  getIngredientsStatistic() {
    return this.http.get<IngredientStatistic[]>(this.url, { withCredentials: true });
  }

  getIngredientsStatisticById(id: number) {
    return this.http.get<IngredientStatistic>(this.url + '/' + id, { withCredentials: true });
  }

  createIngredientsStatistic(statistic: IngredientStatistic) {
    var body = JSON.stringify(statistic);
    var headerOptions = new HttpHeaders({'Content-Type':'application/json'});

    return this.http.post(this.url, body, {
      headers: headerOptions,
      withCredentials: true
    });
  }

  updateIngredientsStatistic(statistic: IngredientStatistic) {
    var body = JSON.stringify(statistic);
    var headerOptions = new HttpHeaders({'Content-Type':'application/json'});

    return this.http.put(this.url, body, {
      headers: headerOptions,
      withCredentials: true
    });
  }

  deleteIngredientsStatistic(id: number) {
    return this.http.delete(this.url + '/' + id, { withCredentials: true });
  }
}

import { HttpClient, HttpHeaders } from "@angular/common/http";
import { Injectable } from "@angular/core";
import { environment } from "src/environments/environment";
import { IngredientStatistic } from "../../interfaces/ingredient-statistic.interface";

@Injectable()
```

```
export class IngredientsStatisticService
{
   url: string;

   constructor(private http: HttpClient){
      this.url = environment.API_URL + '/ingredient-statistic';
   }

   getIngredientsStatistic() {
      return this.http.get<IngredientStatistic[]>(this.url, { withCredentials: true });
   }

   getIngredientsStatisticById(id: number) {
      return this.http.get<IngredientStatistic>(this.url + '/' + id, { withCredentials: true });
   }

   createIngredientsStatistic(statistic: IngredientStatistic) {
      var body = JSON.stringify(statistic);
      var headerOptions = new HttpHeaders({'Content-Type':'application/json'});

      return this.http.post(this.url, body, {
         headers: headerOptions,
         withCredentials: true
      });
   }

   updateIngredientsStatistic(statistic: IngredientStatistic) {
      var body = JSON.stringify(statistic);
      var headerOptions = new HttpHeaders({'Content-Type':'application/json'});

      return this.http.put(this.url, body, {
         headers: headerOptions,
         withCredentials: true
      });
   }

   deleteIngredientsStatistic(id: number) {
      return this.http.delete(this.url + '/' + id, { withCredentials: true });
   }
}

import { HttpClient, HttpHeaders } from "@angular/common/http";
```

```
import { Injectable} from "@angular/core";
import { environment } from "src/environments/environment";
import { SavingRecipe } from "../../interfaces/saving-recipe.interface";
import { Day } from "../../interfaces/week-plan.interface";

@Injectable()
export class MealPlannerService
{
   url: string;

   constructor(private http: HttpClient){
      this.url = environment.API_URL + '/meal-planner';
   }

   getWeekPlan(startDate: string) {
      return this.http.get<Day[]>(this.url + '/week/' + startDate, { withCredentials: true });
   }

   getDayPlan(date: string) {
      return this.http.get<Day>(this.url + '/day/' + date, { withCredentials: true });
   }

   addRecipeToMealPlan(savingRecipe: SavingRecipe) {
      var body = JSON.stringify(savingRecipe);
      var headerOptions = new HttpHeaders({'Content-Type':'application/json'});

      return this.http.post(this.url, body, {
         headers: headerOptions,
         withCredentials: true
      });
   }

   deleteRecipeFromMealPlan(id: number) {
      return this.http.delete(this.url + '/delete/' + id, { withCredentials: true });
   }
}

import { HttpClient } from "@angular/common/http";
import { Injectable } from "@angular/core";
import { environment } from "src/environments/environment";
import { ProductInfo } from "../../interfaces/product-info.interface";
import { ShortProduct } from "../../interfaces/short-product.interface";
```

```
@Injectable()
export class ProductsService
{
   url: string;

   constructor(private http: HttpClient){
      this.url = environment.API_URL + '/products';
   }

   getProductsByTitle(title: string) {
      return this.http.get<ShortProduct[]>(this.url + '/search/title/' + title, {withCredentials:
true});
   }

   getProductById(id: number) {
      return this.http.get<ProductInfo>(this.url + '/search/id/' + id, {withCredentials: true});
   }
}

import { HttpClient, HttpHeaders } from "@angular/common/http";
import { Injectable } from "@angular/core";
import { environment } from "src/environments/environment";
import { ProductBaseInfo } from "../../interfaces/product-base-info.interface";

@Injectable()
export class ProductsBaseInfoService
{
   url: string;

   constructor(private http: HttpClient){
      this.url = environment.API_URL + '/product-base-info';
   }

   getProductsBaseInfo() {
      return this.http.get<ProductBaseInfo[]>(this.url, { withCredentials: true });
   }

   getProductBaseInfoById(id: number) {
      return this.http.get<ProductBaseInfo>(this.url + '/' + id, { withCredentials: true });
   }
```

```
  getProductBaseInfoByApiId(id: number) {
    return this.http.get<ProductBaseInfo>(this.url + '/api/' + id, { withCredentials: true });
  }

  createProductBaseInfo(info: ProductBaseInfo) {
    var body = JSON.stringify(info);
    var headerOptions = new HttpHeaders({'Content-Type':'application/json'});

    return this.http.post(this.url, body, {
      headers: headerOptions,
      withCredentials: true
    });
  }

  updateProductBaseInfo(info: ProductBaseInfo) {
    var body = JSON.stringify(info);
    var headerOptions = new HttpHeaders({'Content-Type':'application/json'});

    return this.http.put(this.url, body, {
      headers: headerOptions,
      withCredentials: true
    });
  }

  deleteProductBaseInfo(id: number) {
    return this.http.delete(this.url + '/' + id, { withCredentials: true });
  }
}

import { HttpClient, HttpHeaders } from "@angular/common/http";
import { Injectable } from "@angular/core";
import { environment } from "src/environments/environment";
import { ProductsExpense } from "../../interfaces/products-expense.interface";

@Injectable()
export class ProductsExpensesService
{
  url: string;

  constructor(private http: HttpClient){
    this.url = environment.API_URL + '/products-expense';
  }
```

```
  getProductsExpenses() {
    return this.http.get<ProductsExpense[]>(this.url, { withCredentials: true });
  }

  getProductsExpenseById(id: number) {
    return this.http.get<ProductsExpense>(this.url + '/' + id, { withCredentials: true });
  }

  getProductsExpensesByUserId(id: string) {
    return this.http.get<ProductsExpense[]>(this.url + '/user/' + id, { withCredentials: true
});
  }

  getProductsExpenseByProductBaseInfoId(infoId: number, curUserId: string) {
    return this.http.get<ProductsExpense>(this.url + '/prod-base-info/' + infoId + '/user/' +
curUserId,
    { withCredentials: true });
  }

  createProductsExpense(expense: ProductsExpense) {
    var body = JSON.stringify(expense);
    var headerOptions = new HttpHeaders({'Content-Type':'application/json'});

    return this.http.post(this.url, body, {
      headers: headerOptions,
      withCredentials: true
    });
  }

  updateProductsExpense(expense: ProductsExpense) {
    var body = JSON.stringify(expense);
    var headerOptions = new HttpHeaders({'Content-Type':'application/json'});

    return this.http.put(this.url, body, {
      headers: headerOptions,
      withCredentials: true
    });
  }

  deleteProductsExpense(id: number) {
    return this.http.delete(this.url + '/' + id, { withCredentials: true });
```

```
    }
}

import { HttpClient, HttpHeaders } from "@angular/common/http";
import { Injectable } from "@angular/core";
import { environment } from "src/environments/environment";
import { ProductStatistic } from "../../interfaces/product-statistic.interface";

@Injectable()
export class ProductsStatisticService
{
    url: string;

    constructor(private http: HttpClient){
        this.url = environment.API_URL + '/product-statistic';
    }

    getProductsStatistic() {
        return this.http.get<ProductStatistic[]>(this.url, { withCredentials: true });
    }

    getProductStatisticById(id: number) {
        return this.http.get<ProductStatistic>(this.url + '/' + id, { withCredentials: true });
    }

    getProductStatisticByProductBaseInfoId(id: number) {
        return this.http.get<ProductStatistic>(this.url + '/prod-base-info/' + id, {
withCredentials: true });
    }

    createProductStatistic(statistic: ProductStatistic) {
        var body = JSON.stringify(statistic);
        var headerOptions = new HttpHeaders({'Content-Type':'application/json'});

        return this.http.post(this.url, body, {
            headers: headerOptions,
            withCredentials: true
        });
    }

    updateProductStatistic(statistic: ProductStatistic) {
        var body = JSON.stringify(statistic);
```

```
        var headerOptions = new HttpHeaders({'Content-Type':'application/json'});

        return this.http.put(this.url, body, {
            headers: headerOptions,
            withCredentials: true
        });
    }

    deleteProductStatistic(id: number) {
        return this.http.delete(this.url + '/' + id, { withCredentials: true });
    }
}

import { HttpClient } from "@angular/common/http";
import { Injectable, OnInit } from "@angular/core";
import { environment } from "src/environments/environment";
import { FullRecipe } from "../../interfaces/full-recipe.interface";
import { RecipeInfo } from "../../interfaces/recipe-info.interface";
import { Recipe } from "../../interfaces/recipe.interface";

@Injectable()
export class RecipesService
{
    url: string;

    constructor(private http: HttpClient){
        this.url = environment.API_URL + '/recipes';
    }

    getRandomRecipes(amount: number) {
        return this.http.get<FullRecipe[]>(this.url + '/random/' + amount, { withCredentials: true
});
    }

    getRecipesByTitle(title: string) {
        return this.http.get<Recipe[]>(this.url + '/search/title/' + title, {withCredentials: true});
    }

    getRecipeById(id: number) {
        return this.http.get<RecipeInfo>(this.url + '/search/id/' + id, {withCredentials: true});
    }
}
```

```
import { HttpClient, HttpHeaders } from "@angular/common/http";
import { Injectable } from "@angular/core";
import { environment } from "src/environments/environment";
import { RecipeBaseInfo } from "../../interfaces/recipe-base-info.interface";

@Injectable()
export class RecipesBaseInfoService
{
   url: string;

   constructor(private http: HttpClient){
      this.url = environment.API_URL + '/recipe-base-info';
   }

   getRecipesBaseInfo() {
      return this.http.get<RecipeBaseInfo[]>(this.url, { withCredentials: true });
   }

   getRecipeBaseInfoById(id: number) {
      return this.http.get<RecipeBaseInfo>(this.url + '/' + id, { withCredentials: true });
   }

   getRecipeBaseInfoByApiId(id: number) {
      return this.http.get<RecipeBaseInfo>(this.url + '/api/' + id, { withCredentials: true });
   }

   createRecipeBaseInfo(info: RecipeBaseInfo) {
      var body = JSON.stringify(info);
      var headerOptions = new HttpHeaders({'Content-Type':'application/json'});

      return this.http.post(this.url, body, {
         headers: headerOptions,
         withCredentials: true
      });
   }

   updateRecipeBaseInfo(info: RecipeBaseInfo) {
      var body = JSON.stringify(info);
      var headerOptions = new HttpHeaders({'Content-Type':'application/json'});

      return this.http.put(this.url, body, {
```

```
            headers: headerOptions,
            withCredentials: true
        });
    }

    deleteRecipeBaseInfo(id: number) {
        return this.http.delete(this.url + '/' + id, { withCredentials: true });
    }
}

import { HttpClient, HttpHeaders } from "@angular/common/http";
import { Injectable} from "@angular/core";
import { environment } from "src/environments/environment";
import { Aisle } from "../../interfaces/aisle.interface";
import { ShoppingItem } from "../../interfaces/shopping-item.interface";

@Injectable()
export class ShoppingListService
{
    url: string;

    constructor(private http: HttpClient){
        this.url = environment.API_URL + '/shopping-list';
    }

    getShoppingList() {
        return this.http.get<Aisle[]>(this.url, { withCredentials: true });
    }

    addToShoppingList(item: ShoppingItem) {
        var body = JSON.stringify(item);
        var headerOptions = new HttpHeaders({'Content-Type':'application/json'});

        return this.http.post(this.url, body, {
            headers: headerOptions,
            withCredentials: true
        });
    }

    deleteFromShoppingList(id: number) {
        return this.http.delete(this.url + '/delete/' + id, { withCredentials: true });
    }
```

```
}
```

ДОДАТОК Б

Відзив керівника економічного розділу

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ОПТИЧНОМУ НОСІЇ

Ім’я файла	Опис
Пояснювальні документи	
Диплом.doc	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Диплом.pdf	Пояснювальна записка до кваліфікаційної роботи. в форматі PDF
Програма	
Diplom.zip	Архів. Містить коди програми і откомпільовану програму
Презентація	
Презентація.ppt	Презентація кваліфікаційної роботи.