Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка веб-сайту садового центру «ФасолькаСад» засобами ASP.NET Core та React

Виконав: студент IV курсу, групи СНс-41
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

_____ Титжинський А.А.
(підпис) (прізвище та ініціали)

Керівник _____ Козбур Г.В.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Литвиненко Я.В.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ Карпінський М.П.
(підпис) (прізвище та ініціали)

Тернопіль
2023

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис (прізвище та ініц
«21» червня 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Титжинському Анатолію Андрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка веб-сайту садового центру «ФасолькаСад» засобами ASP.NET Core та React

Керівник роботи Козбур Галина Володимирівна, кандидат технічних наук, доцент кафедри КН.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « 7 » лютого 2023 року № 4/7-135

2. Термін подання студентом завершеної роботи 22 червня 2023р.

3. Вихідні дані до роботи Літературні та інтернет джерела інформації про технології розробки веб-сайту садового центру «ФасолькаСад» засобами ASP.NET Core та React

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1. Аналіз предметної області та постановка задач для веб-сайту садового центру «ФасолькаСад». Аналіз садових центрів. Розробка структури сайту та постановка завдання. Стадії та етапи розробки сайту. Вибір середовища проєктування. Вибір сервісу авторизації. Висновок до першого розділу. 2. Розробка веб-сайту садового центру «ФасолькаСад». Засоби для створення веб-сайту садового центру «ФасолькаСад». Застосовані фреймворки при розробці садового центру. Файлова структура проєкту. Розробка структури бази даних. Обслуговування та наповнення сайту. Тестування сайту. Оцінка отриманих результатів веб-сайту садового центру «ФасолькаСад». Висновки до другого розділу. 3. Безпека життєдіяльності, основи хорони праці. Ергономічні проблеми безпеки життєдіяльності. Загальні вимоги безпеки з охорони праці для користувачів ПК. Висновки до третього розділу. Висновки. Перелік Джерел. Додаток А. Додаток Б.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи хорони праці		05.06.2023	08.06.2023

7. Дата видачі завдання 23 січня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	23.01.2023	*Виконано*
2.	Підбір джерел про Розробка веб-сайту садового	24.01.2023-26.01.2023	*Виконано*
	центру "ФасолькаСад"		
3.	Опрацювання джерел по темі кваліфікаційної роботи	27.01.2023-31.01.2023	*Виконано*
4.	Виконання дослідження щодо розробки веб-сайту	01.02.2023-07.02.2023	*Виконано*
	Садового центру «ФасолькаСад» засобами ASP.NET		
	Core та React		
	Розроблення веб-сайту		
5.	Оформлення розділу «Аналіз предметної області та	08.02.2023-09.02.2023	*Виконано*
	постановка задач для веб-сайту садового центру		
	«ФасолькаСад»»		
6.	Оформлення розділу «Розробка веб-сайту садового	10.02.2023-12.02.2023	*Виконано*
	центру «ФасолькаСад»»		
7.	Виконання завдання до підрозділу «Безпека	05.06.2023-06.06.2023	*Виконано*
	життєдіяльності, охорона праці»		
8.	Оформлення кваліфікаційної роботи	09.06.2023-11.06.2023	*Виконано*
9.	Нормоконтроль	12.06.2023-13.06.2023	*Виконано*
10.	Перевірка на плагіат		*Виконано*
11.	Попередній захист кваліфікаційної роботи		*Виконано*
12.	Захист кваліфікаційної роботи	22.06.2023	

Студент _____ Титжинський А. А.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Козбур Г.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка вебсайту садового центру «Фасолька Сад» засобами ASP.NET Core та React. // Кваліфікаційна робота освітнього рівня «Бакалавр» // Титжинський Анатолій Андрійович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СНс-41 // Тернопіль, 2023 // С. 40, рис. – 7, додат. – 2, бібліогр. – 26.

Ключові слова: бази даних, вебсайт, ASP.NET Core, React, HTML5, CSS3, JavaScript, SQL Server.

Кваліфікаційна робота присвячена дослідженню та розробці вебсайту садового центру «ФасолькаСад».

В першому розділі роботи проведено аналіз садових центрів, сформовано структуру сайту та вибрано середовище проєктування. В розділі розглянуто стадії та етапи розробки сайту.

Другий розділ зосереджено на розробці вебсайту засобами ASP.NET Core та React. Розглядаються застосовані при розробці фреймворки, файлова структура проєкту, розробка структури бази даних, обслуговування та наповнення сайту. Відмінна увага приділена тестуванню сайту та оцінці отриманих результатів.

У третьому розділі описані ергономічні проблеми безпеки життєдіяльності, загальні вимоги безпеки з охорони праці для користувачів ПК.

ANNOTATION

The «Fasolkasad» Garden Center Website Development by Means of ASP.NET Core and React. // Bachelor's degree thesis // Tytzhynskyi Anatolii // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Computer Science Department, group SNs-41 // Ternopil, 2023 // P. 40, fig. - 7, app. - 2, references - 26.

Keywords: database, website, ASP.NET Core, React, HTML5, CSS3, JavaScript, SQL Server.

The qualification work is dedicated to the research and development of the website of the garden center «FasolkaSad».

The first chapter of the work carries out an analysis of garden centers, formulates the website structure and chooses a design environment. The chapter considers stages and steps of the website development, including the selection of an authorization service.

The second chapter focuses on the development of the website using ASP.NET Core and React. It discusses the frameworks used in development, the project file structure, the development of the database structure, the website maintenance and filling. Particular attention is given to the website testing and evaluation of the obtained results.

The third chapter describes ergonomic safety issues, general safety requirements for PC users.

ЗМІСТ

ВСТУП

Актуальність теми. Внаслідок глобалізації та швидкого розвитку цифрових технологій, інтернет став невід'ємною частиною повсякденного життя. Веб-сайти стали важливим інструментом для бізнесу, оскільки вони надають платформу для продажу товарів та послуг. Садовий бізнес не є виключенням. Створення веб-сайту для садового центру, що забезпечує легкий доступ до широкого асортименту товарів та послуг, є актуальним напрямком сучасних досліджень в галузі інформаційних технологій та електронної комерції.

Мета і задачі дослідження. Ціль цієї кваліфікаційної роботи освітнього рівня «Бакалавр» є покращення якості послуг садового центру через створення ефективного, зручного та привабливого веб-сайту. Щоб досягти цієї мети, необхідно виконати низку завдань, включаючи: розглянути поточний стан веб-сайтів садових центрів, розробити структуру бази даних та веб-сайту, організувати обслуговування та наповнення сайту, провести його тестування та оцінити отримані результати.

Практичне значення одержаних результатів. Результати цього дослідження можуть бути використані в реальних бізнес-сценаріях. Створений веб-сайт може слугувати ефективною платформою для садового центру, сприяючи його розвитку та зростанню продажів. Крім того, здобутий досвід та знання можуть бути використані для подальшої роботи у сфері веб-розробки та електронної комерції.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧ ДЛЯ ВЕБ-САЙТУ САДОВОГО ЦЕНТРУ «ФАСОЛЬКАСАД»

1.1 Аналіз садових центрів

Для розуміння потреб ринку та формування конкурентної пропозиції, проведемо аналіз садових центрів. Основні критерії оцінки включають наявність широкого асортименту товарів, якість продукції, зручність використання веб-сайту, наявність додаткових сервісів, цінову політику та маркетингові активності.

Сайт-садовий центр «Ґалсад» відзначається великим вибором товарів для садівництва.

Дизайн сайту приємний і функціональний, що теж є важливим фактором привабливості. Кольорова палітра, використана на сайті, створює приємне візуальне враження, а легко орієнтована структура сайту полегшує пошук потрібних товарів. Дизайн головної сторінки сайту можна побачити на рисунку 1.1.

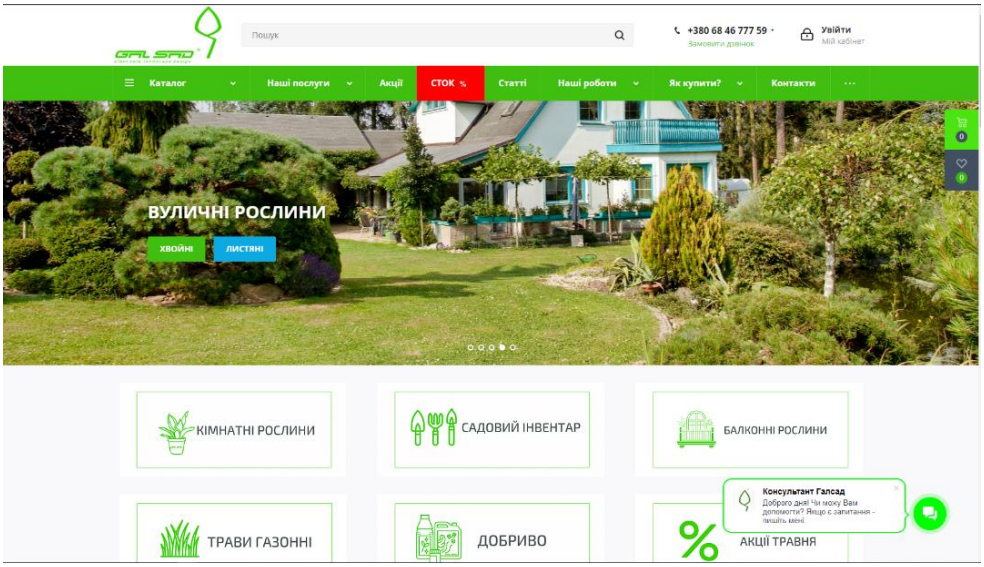

Рисунок 1.1 – Дизайн головної сторінки сайту «Ґалсад»

Проте варто зазначити про наявність графічних багів на телефонах, наприклад на рисунку 1.2, можна побачити, що спливаючий елемент з поясненням до кнопки переходу в чат з консультантом, виходить за межі екрану. Хоч даний недолік і не критичний, він може відштовхнути потенційних покупців.

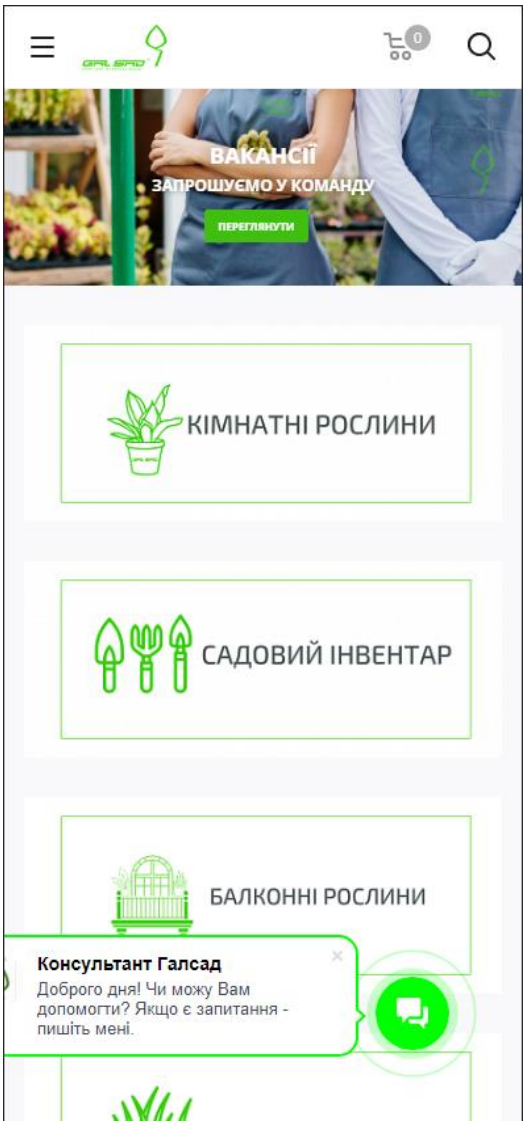

Рисунок 1.2 – Графічний баг сайту «Ґалсад» на мобільному пристрої

Сайт використовується для потреб одного продавця, тому він не підходить для розміщення власних товарів. Широкий асортимент дозволяє покрити різні потреби садівників, від різних типів рослин до садового інвентарю, що робить сайт привабливим для широкого кола користувачів.

В загальному, сайт садового центру «Ґалсад» здається привабливим та потенційно успішним.

Сайт «Агро-маркет» представляє собою місце для широкого асортименту товарів, але його дизайн може становити проблему для користувачів. Дизайн головної сторінки сайт «Агро-маркет» можна побачити на рисунку 1.3.

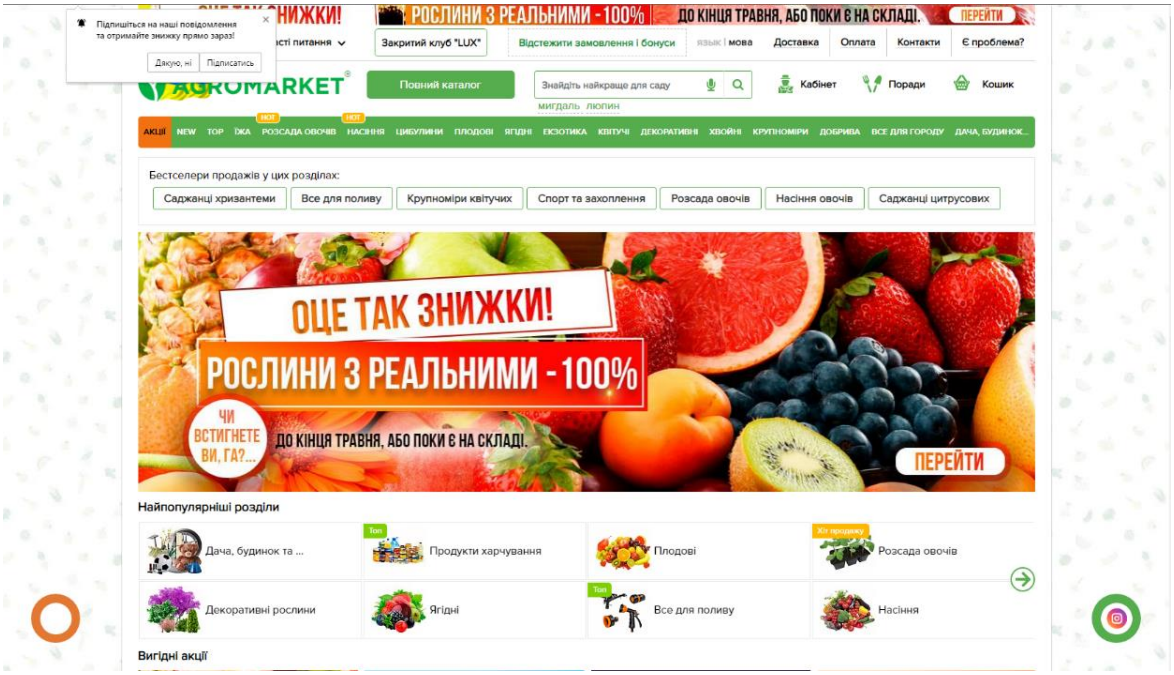

Рисунок 1.3 – дизайн головної сторінки сайту «Агро-маркет»

«Агро-маркет» пропонує широкий вибір товарів, це може привабити різні групи користувачів, оскільки тут вони можуть знайти різні товари, які вони потребують для своїх аграрних потреб. Великий вибір товарів може сприяти задоволенню широкого спектра потреб клієнтів.

Однак, на жаль, дизайн сайту «Агро-маркет» є проблемним. Перевантаженість інформацією, надто яскраві кольори картинок можуть зробити сайт неприємним для використання. Це може викликати відчуття перебільшеної зайнятості та стресу у користувачів, що може відштовхнути їх від повторного звернення до сайту. Крім того, якщо користувачам важко знайти те, що їм потрібно через слабку навігацію або організацію сайту, це може викликати

фрустрацію та незадоволення, що врешті-решт може вплинути на їх бажання повернутися на сайт.

У цілому, «Агро-маркет» має потенціал бути успішним сайтом завдяки своєму широкому асортименту товарів, але проблеми з дизайном становлять перешкоду для кращого користувацького досвіду.

1.2 Розробка структури сайту та постановка завдання

Вимоги до веб-сайту садового центру «ФасолькаСад»:

– Зручний, інтуїтивно зрозумілий інтерфейс – полегшує користування сайтом для всіх відвідувачів, незалежно від їх технічного досвіду.

– Мінімалістичний дизайн – створює приємне візуальне сприйняття і спрощує взаємодію з контентом сайту.

– Висока швидкодія – забезпечує швидке завантаження сторінок та позитивний користувацький досвід.

– Кросплатформеність – дозволяє розмістити веб-сайт на будь-якій зручній платформі.

– Адаптивний дизайн – забезпечує оптимальне відображення та функціональність сайту на різних пристроях, включаючи настільні комп'ютери, ноутбуки, планшети та мобільні телефони.

– Маштабованість – дозволяє сайту ефективно працювати при збільшенні обсягу відвідувачів або даних, не втрачаючи продуктивності чи якості обслуговування.

Структура сайту наступна:

– Каталог – ця сторінка служить як основний переглядовий інструмент для користувачів, де вони можуть ознайомитись з асортиментом товарів, що доступні на сайті. Каталог зазвичай організований за категоріями, що спрощує навігацію та пошук конкретного товару.

– Сторінка товару – кожен товар на сайті повинен мати власну сторінку, де детально описано його характеристики, включаючи ціну, доступність, опис,

фотографії тощо. Це допоможе користувачам приймати обґрунтовані рішення про покупку.

– Особистий кабінет користувача – ця сторінка служить для того, щоб користувач міг переглядати та керувати своїми даними, включаючи історію покупок, збережені товари, особисті дані, налаштування сповіщень тощо.

– Інтерфейс для додавання товарів власником – ця функція дозволяє власникам товарів додавати свої товари на сайт. Вона повинна бути інтуїтивно зрозумілою і простою в використанні, з можливістю завантажувати фотографії, додавати описи, вказувати ціну та іншу інформацію про товар.

– Сторінка «Про нас» – ця сторінка містить інформацію про компанію, її історію, місію, цінності, контактну інформацію та інші подробиці, які можуть бути цікавими для користувачів або потенційних партнерів.

– Корзина – ця сторінка призначена для показу товарів, які користувач вирішив купити, їхню загальну суму та здійснення покупки.

– Вподобані товари – ця сторінка призначена для відображення вподобаних товарів.

– Навігаційна панель – цей елемент буде присутнім в верхній частині сайту та дозволить легко переміщатись між сторінками.

– Вхід та Реєстрація – на сайті повинні бути розміщені функції входу та реєстрації для користувачів. Функція входу дозволяє зареєстрованим користувачам увійти в свій особистий кабінет, зокрема для перегляду своєї історії покупок та здійснення нових покупок. Функція реєстрації надає можливість новим користувачам створити обліковий запис на сайті, де вони зможуть зберігати свої дані, вибирати товари та здійснювати покупки. Обидві функції повинні бути зручними та безпечними, з використанням захисту даних та можливості відновлення паролю у разі необхідності.

Для кращого розуміння структури сайту була створена діаграма Site map. Її можна побачити на рисунку 1.4.

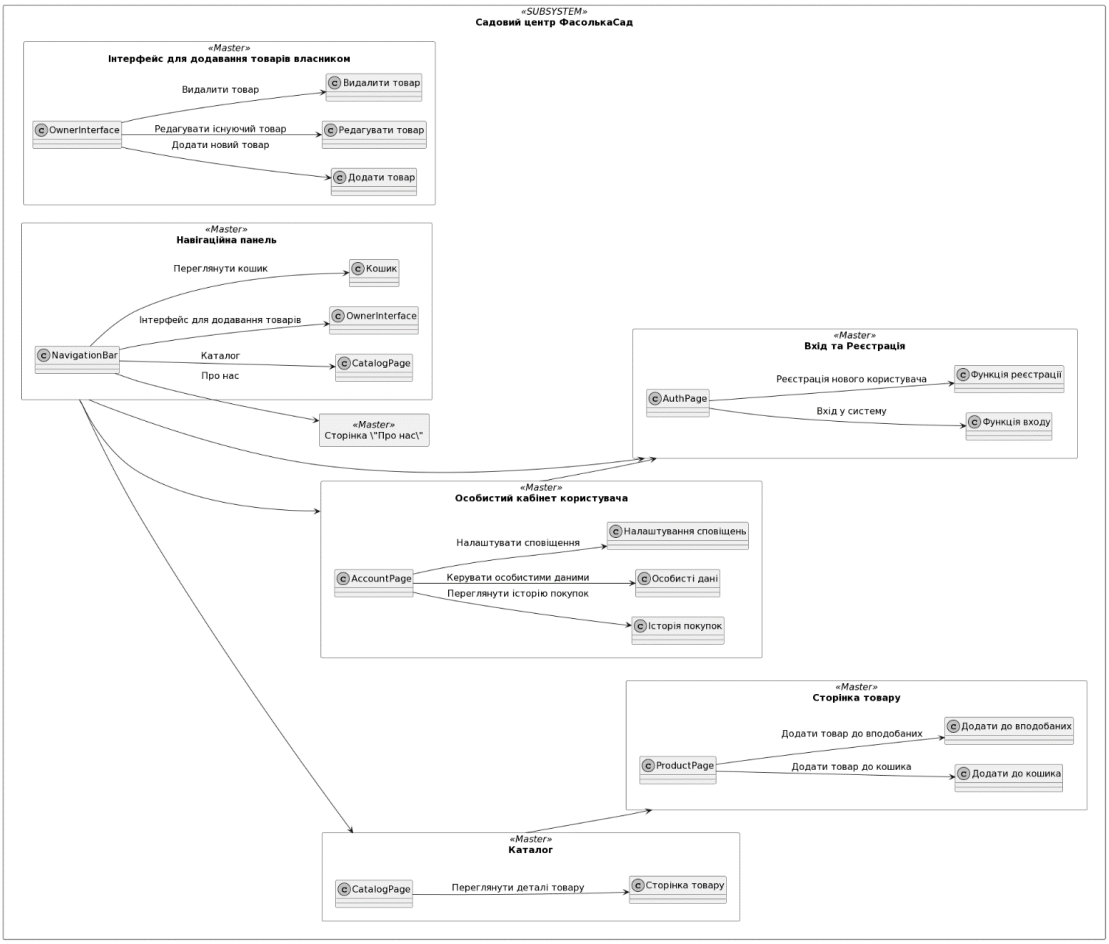

Рисунок 1.4 – діаграма Site map веб-сайту садового центру «ФасолькаСад»

Дана структура дозволить сайту бути інтуїтивно зрозумілим та зручним у використанні.

1.3 Стадії та етапи розробки сайту

Розробка веб-сайту садового центру «ФасолькаСад» передбачає дотримання наступних стадій та етапів:

– Аналіз ринку та конкурентів: Визначення сильних і слабких сторін поточних конкурентів на ринку, вивчення їхньої пропозиції та маркетингових стратегій, щоб знайти свої конкурентні переваги та виявити можливі прогалини.

– Формування вимог до сайту: На основі проведеного аналізу, встановлення ключових вимог до функціоналу, дизайну та структури сайту, а також до системи замовлень, оплати та доставки.

– Розробка структури та навігації сайту: Створення ієрархії сторінок, оптимізація навігації та взаємодії користувача з веб-сайтом, розробка макету сайту.

– Дизайн та створення прототипу сайту: Розробка візуального стилю сайту, включаючи елементи графіки, шрифти, кольори та композицію. Створення прототипу сайту для попереднього перегляду та оцінки зручності використання.

– Вибір технологій та середовища проєктування: Обрання підходящих технологій для розробки сайту, таких як React, ASP.NET Core, MSSQL як база даних, а також інструментів для роботи в команді.

– Програмування та розробка функціоналу сайту: Написання коду, створення функціональних модулів, інтеграція з базою даних та реалізація замовлень, оплати та доставки.

– Тестування та налагодження: Проведення ретельного тестування сайту на наявність помилок, проблем з проектування.

– Випуск та реліз сайту: Після успішного завершення розробки, тестування та налагодження, сайт готовий до публічного релізу. Відбувається підготовка і перенесення сайту на веб-сервер, налаштування необхідних доменних імен та сертифікатів безпеки.

Виконання усіх пунктів дозволить отримати повністю функціональний сайт садового центру «ФасолькаСад».

1.4 Вибір середовища проєктування

Для розробки веб-сайту садового центру «ФасолькаСад» було обрано низку сучасних технологій та середовищ проєктування, що забезпечують створення відповідального, привабливого та зручного для користувачів веб-сайту. Вибір цих компонентів був здійснений з урахуванням передових технологій та вимог проекту. Ось детальний опис обраних технологій та їх переваг:

– React – ця JavaScript-бібліотека була обрана для розробки інтерактивних та динамічних веб-додатків. React пропонує модульну архітектуру, що сприяє швидкій та ефективній розробці відповідного коду. За допомогою React можна створити користувачам зручний та реактивний інтерфейс.[1]

– ASP.NET Core – цей кросплатформений фреймворк від Microsoft відповідає за розробку веб-додатків, веб-сайтів, веб-служб та прикладних програмних інтерфейсів. ASP.NET Core забезпечує високу продуктивність, масштабованість та безпеку. Його кросплатформовість дозволяє розгортати веб-сайт на різних платформах, що забезпечує гнучкість та доступність проекту.[2]

– База даних MSSQL – сучасна платформа даних для розвитку компаній із підтримкою Azure та інноваційними можливостями для підвищення продуктивності й захисту.[3] Обрано MSSQL з урахуванням сумісності з ASP.NET Core та попереднього досвіду розробників з використання Microsoft-екосистеми.

Причини вибору ASP.NET Core для проекту наступні:

– Кросплатформовість – ASP.NET Core підтримує розробку на різних платформах, включаючи Windows, macOS та Linux. Це дозволяє розгортати веб-сайт на будь-якому сервері, що сприяє гнучкості та масштабованості проекту.

– Висока продуктивність – фреймворк оптимізований для високої продуктивності та швидкодії. Він має вбудовану систему кешування, підтримку асинхронного програмування та оптимізований механізм обробки запитів, що дозволяє забезпечити швидку реакцію сайту на запити користувачів.

– Масштабованість – ASP.NET Core має вбудовану підтримку горизонтального масштабування, що дозволяє розгортати додатки на кластері серверів. Це забезпечує високу доступність та можливість обробки великого обсягу запитів.

– Безпека – фреймворк надає потужні засоби для захисту веб-додатків, включаючи вбудовану підтримку автентифікації та авторизації, захист від атак, шифрування даних та безпечне з'єднання.

– Розширюваність – ASP.NET Core має велику кількість розширень та сторонніх бібліотек, що дозволяє розширювати функціональність веб-сайту шляхом використання готових компонентів.

– Підтримка Visual Studio – ASP.NET Core є невід'ємною частиною екосистеми Visual Studio, що забезпечує розробникам потужний інструментарій для ефективної роботи та підтримки проекту.

Враховуючи ці переваги, вибір ASP.NET Core для розробки веб-сайту садового центру «ФасолькаСад» гарантує ефективну, масштабовану та безпечну розробку, що задовольнятиме потреби користувачів та забезпечить стабільну роботу сайту.

Додатково, розглянуто такі конкурентні технології:

– Angular – цей фреймворк JavaScript є альтернативою для розробки веб-додатків. Однак, з урахуванням попереднього досвіду та наявної інтеграції вибір інструменту фронтенд-розробки впав на React.

– MySQL та PostgreSQL – ці інструменти для роботи з базами даних також є популярними виборами для веб-додатків. Проте, враховуючи сумісність з ASP.NET Core та попередній досвід з використанням Microsoft-екосистеми, було обрано MSSQL.

Враховуючи переваги React, ASP.NET Core та MSSQL, такі як широка спільнота розробників, висока продуктивність, гнучкість, масштабованість та забезпечення безпеки, вибір цих технологій та середовищ проєктування гарантує ефективну розробку та функціональність веб-сайту садового центру «ФасолькаСад».

1.4.1 Вибір сервісу авторизації

При плануванні проекту веб-сайту садового центру "ФасолькаСад" були розглянуті різні варіанти сервісів авторизації, зокрема Keycloak, ActiveDirectory та IdentityServer.

Основні переваги Keycloak:

– Прогресивність – Keycloak є потужним і сучасним сервісом авторизації з підтримкою стандартів OAuth і OpenID Connect.

– Функціональність – він надає широкий спектр можливостей для керування ідентифікацією та авторизацією користувачів, включаючи управління ролями, правами доступу та аудитом.

– Інтеграція – Keycloak добре інтегрується з різними типами клієнтів, включаючи веб-додатки, мобільні додатки та API.[4]

– Основні переваги ActiveDirectory:

– Інтеграція з екосистемою Microsoft – ActiveDirectory є централізованою службою керування ідентифікацією та авторизацією в середовищі Windows.

– Розширені можливості – він пропонує обширний набір функцій, включаючи управління ролями, політиками безпеки та інтеграцію з іншими сервісами Microsoft.

– Орієнтований на корпоративні потреби – ActiveDirectory підходить для великих підприємств з потребою у централізованому управлінні користувачами та рівнями доступу.[5]

Основні переваги IdentityServer:

– Відкритість та гнучкість – IdentityServer базується на відкритих стандартах авторизації, таких як OAuth і OpenID Connect, що дозволяє інтегрувати його з різними сервісами та типами клієнтів.

– Безпека – IdentityServer забезпечує потужну систему безпеки з механізмами аутентифікації, авторизації та захисту даних.

– Розширюваність – IdentityServer має гнучку архітектуру, що дозволяє розширювати його функціональність та пристосовувати до конкретних потреб проекту.

– Підтримка спільнотою – IdentityServer має активну спільноту розробників, що забезпечує доступ до документації, підручників, форумів та інших ресурсів для розв'язання проблем та підтримки розробки.

– Інтеграція з ASP.Net Core – IdentityServer є продукцією компанії Microsoft та має готову інтеграцію з фреймворком ASP.Net Core.[6]

Після уважного аналізу та оцінки функціональності, прогресивності та підтримки спільнотою, було прийняте рішення вибрати IdentityServer як сервіс авторизації для проекту веб-сайту садового центру "ФасолькаСад". Його відкритість, гнучкість та безпека відповідають вимогам проекту та забезпечать зручну та надійну систему авторизації для наших користувачів, а наявна інтеграція з ASP.Net Core дозволить скоротити час розробки при збережені якості та безпеки кінцевого продукту.

1.5 Висновок до першого розділу

Дослідження предметної сфери та формулювання завдань для веб-сайту садового центру "ФасолькаСад" дозволяє встановити основні вимоги до функцій та дизайну сайту. Для досягнення успіху на ринку, веб-сайт має бути зручним, інтуїтивним та привабливим для користувачів, забезпечуючи широкий асортимент товарів та високу якість обслуговування.

Розробка веб-сайту передбачає проведення аналізу ринку та конкурентів, визначення вимог до сайту, формування структури та навігації, дизайну та прототипування, вибір технологій та середовища проєктування, програмування та налагодження функціоналу, тестування та запуск сайту. Успішна реалізація цих етапів дозволить створити конкурентоспроможний веб-сайт садового садового «ФасолькаСад».

РОЗДІЛ 2. РОЗРОБКА ВЕБ-САЙТУ САДОВОГО ЦЕНТРУ «ФАСОЛЬКАСАД»

2.1 Засоби для створення веб-сайту садового центру «ФасолькаСад»

При розробці веб-сайту садового центру «ФасолькаСад» було використано ряд інструментів.

Використано Visual Studio – потужне середовище розробки, яке допомагає створювати програми та веб-додатки, підтримуючи різні мови програмування та інтегруючись з різними системами контролю версій. Також даний інструмент пропонує ряд шаблонів, які можна використати, як основу для проекту, в даному проекті було використано шаблон «ASP.NET Core with React.js».[7]

Використовувався Visual Studio Code, легкий редактор вихідного коду з відкритим кодом, що надає широкі можливості для розробки веб-додатків, включаючи підтримку синтаксису HTML, CSS та JavaScript.[8] Це зручний інструмент для написання React додатків.

Важливим інструментом стало SQL Server Management Studio (SSMS), яке використовується для управління, конфігурації та моніторингу інфраструктури SQL Server, включаючи створення та управління базами даних.[9]

Важливим елементом у розробці стала використання менеджера пакунків npm. Node Package Manager (npm) використовується для управління бібліотеками та пакунками JavaScript, що дозволяє розробникам легко встановлювати, оновлювати та управляти залежностями в своїх проектах.[10]

Також, використання NuGet було критично важливим для проекту. NuGet – це менеджер пакунків для платформи .NET, який допомагає розробникам легко знаходити, встановлювати та управляти бібліотеками та пакунками .NET в їх проектах.[11]

Іншим значним інструментом, що було використано у процесі розробки «ФасолькаСад», є NSwag. NSwag – це набір відкритих інструментів для .NET, що

дозволяє генерувати специфікації OpenAPI та Swagger та інтегрувати клієнтів API та сервери на основі цих специфікацій.

NSwag дозволяє розробникам виконувати ряд задач, пов'язаних з веб-API. Зокрема, він може генерувати специфікації OpenAPI для .NET веб-серверів, автоматично створювати клієнтські бібліотеки для використання цих веб-API з різних мов програмування (наприклад, TypeScript, C#, Java і багатьох інших), та генерувати інтерфейси користувача для перевірки та тестування веб-API.[12]

У даному проекті NSwag використовувався для створення та оновлення специфікацій OpenAPI для всіх веб-API. Це не тільки забезпечує консистентність і ясність інтерфейсів API, але і дозволяє легко і швидко генерувати клієнтські бібліотеки для використання веб-API в різних частинах системи. Завдяки цьому, при розробці можна працювати ефективніше, зосереджуючись на реалізації бізнес-логіки, а не на написанні коду для взаємодії з API.

2.1.1 Застосовані фреймворки при розробці садового центру

При розробці садового центру «ФасолькаСад» важливу роль відіграли наступні фреймворки:

– .NET Core – Це загальне середовище виконання для .NET платформи, яке дозволяє розробляти різноманітні типи програм - веб-додатки, десктопні застосунки, мобільні додатки, мікросервіси та багато інших. Особливістю .NET Core є його кросплатформовість, що дозволяє розробникам використовувати різні мови програмування, такі як C#, F# та Visual Basic, та створювати програми для різних платформ та операційних систем. Використання .NET Core в процесі розробки веб-сайту садового центру «ФасолькаСад» забезпечило високу продуктивність, гнучкість та доступність для розробників.[13]

– ASP.NET Core – Це модернізована та перероблена версія ASP.NET – високопродуктивного фреймворку для побудови веб-додатків. Розроблений з урахуванням сучасних потреб розробників, ASP.NET Core включає підтримку різних парадигм програмування, таких як MVC та Web API. Він також надає

вбудовану підтримку залежностей, конфігурації та інших поліпшень, що спрощують створення надійних та безпечних веб-додатків. Застосування ASP.NET Core дозволило забезпечити ефективну розробку та функціональність веб-сайту садового центру «ФасолькаСад».

– Entity Framework Core – це фреймворк для роботи з базами даних, який забезпечує мапування об'єктно-орієнтованої моделі на реляційну базу даних. При розробці веб-сайту садового центру «ФасолькаСад», EF Core використовується для зручного та ефективного взаємодії з базою даних MSSQL. Він надає розробникам можливість працювати з базою даних через об'єктно-орієнтований підхід, спрощуючи взаємодію з даними та забезпечуючи різноманітні функціональні можливості, такі як модифікація схеми бази даних, виконання запитів та оптимізація запитів.[14]

Ці фреймворки разом створюють потужну і сучасну платформу для розробки та експлуатації веб-сайту садового центру, сприяючи формуванню надійного, безпечного та привабливого користувацького досвіду.

React часто помилково називають фреймворком, насправді є потужною JavaScript бібліотекою для побудови високоякісних, інтерактивних та динамічних інтерфейсів користувача. Розроблений і підтримуваний компанією Meta, React використовується в сотнях тисяч веб-сайтів та додатків у всьому світі і відомий своєю швидкістю, гнучкістю та ефективністю.

Основною перевагою React є його модульність. Він використовує компоненти, які можна легко повторно використовувати та комбінувати, що робить розробку більш ефективною і менш витратною за часом. Крім того, React включає підтримку реактивного програмування, де додаток автоматично відображає зміни в даних без необхідності перезавантаження сторінки. Завдяки можливостям, які надає React, створюються багатофункціональні, гнучкі та адаптивні веб-сторінки, що забезпечують користувачам простий та інтуїтивно зрозумілий досвід взаємодії з веб-сайтом.

2.1.2 Файлова структура проєкту

Оскільки в проєкті застосовуються ASP.Net Core та React файлову структуру можна поділити а дві частини:

– Бекенд частина – відповідає за структуру файлів серверної частини проекту, що використовується фреймворком ASP.Net Core.

– Фронтенд частина – відповідає за структуру файлів клієнтської частини, що використовується бібліотекою React. Зазвичай початкова структура генерується виконанням команди «npx create-react-app назва-додатка», але в даному випадку було використано готовий шаблон, наданий компанією Microsoft.

Структура бекенд частини файлової структури є наступною:

– Основна папка проєкту – основна папка, в якій розміщуються усі папки та файли, які стосуються проєкту.

– wwwroot – папка, в якій зберігаються усі файли, які використовуються при http-запитах до серверної частини. В даному проєкті тут зберігаються усі фото товарів, та файл налаштувань для Swagger.

– ClientApp – папка, в якій розміщується React проєкт, більш детально його буде описано згодом.

– Controllers – папка, в якій розміщуються контролери, які відповідають за логіку обробки запитів до серверної частини проєкту через прикладний програмний інтерфейс.

– Data – папка, в якій розміщуються усі файли, що відповідають за конфігурацію бази даних та її контексту в програмному коді.

– Interfaces – папка, в якій розміщуються інтерфейси мови програмування C#, що використовуються в проєкті та написані розробником та не імпортовані зі сторонніх бібліотек.

– Migration – в даній папці зберігаються усі міграції згенеровані за допомогою Entity Framework та використовуються для оновлення бази даних та, за потреби, повернення до попередніх версій.

– Models – в даній папці розміщуються усі класи-моделі проєкту. Дані класи відокремлені від логіки, тобто містять тільки поля.

– Pages – в даній папці розміщуються сторінки побудова яких відбувається на стороні серверу, в основному це сторінки реєстрації та входу. Також варто зазначити, що в версії ASP.Net Core 2.1.0 усі сторінки були перенесені до окремої бібліотеки – IdentityUI, тобто сторінки не наявні в файловій системі проєкту, а знаходяться в файлах бібліотеки до того моменту поки розробник не вирішить змінити стандартний вигляд сторінок.

– Repositories – в даній папці розміщуються усі класи-репозиторії, тобто класи які відповідають за зв'язок коду з базою даних за допомогою простих дій-методів, як наприклад Add – додати, Remove – прибрати та Get – отримати. Варто зазначити що усі репозиторії логічно розділені, для комфортної роботи.

– Services – в даній папці розміщуються класи-сервіси що містять допоміжний функціонал для написані коду, наприклад сервіс для роботи з налаштуваннями – перевірки чи налаштування існує, отримання його значення з файлу appsettings.json або змінних середовища.

– .gitignore – файл який визначає, які папки та файли ігноруються при створені комітів.

– appsettings.json – файл в якому зберігаються налаштування проєкту. Також наявний файл appsettings.Development.json в якому зберігаються налаштування, що використовуються проєктом в режимі Debug, тобто не будуть використовуватись проєктом після розгортання.

– nswag.json – файл конфігурацій для генератора клієнтського коду взаємодії з прикладним програмним інтерфейсом сервера – NSwag.

– Program.cs – файл класу який використовується для опису логіки запуску проєкту та налаштування обробки запитів до серверу.[15]

– Структура фронтенд частини файлової структури є наступною:

– public – в даній папці містяться іконка, яка відображається на закладці в браузері, html файл який містись базову структуру сторінки, та файл базових налаштувань проєкту.

– src – в даній папці міститься вихідний код проєкту, що використовується для клієнтської частини, звісно виключаючи сторінки побудова яких відбувається на серверній частині, тобто сторінки входу, реєстрації та особистого кабінету користувача.

– app – в даній папці розміщується код згенерований за допомогою NSwag та необхідні для його роботи компоненти.

– components – в даній папці мітяться файли, які відповідають за логіку та генерацію контенту веб-сайту. Окремо варто відзначити папку api-authorization в якій розміщуються усі файли, які відповідають за авторизацію користувача на клієнтській частині.[16]

Файлова структура даного проєкту відповідає найкращим практикам прийнятим в індустрії, тобто є доволі популярною та логічною, що дозволяє легко орієнтуватись в великому обсязі файлів.

2.2 Розробка структури бази даних

У процесі розробки веб-сайту садового центру «ФасолькаСад» особливу увагу було приділено структурі бази даних. Для цього були використані підхід Code First та Entity Framework.

Code First – це підхід, використаний в Entity Framework, який дозволяє розробникам визначати структуру бази даних за допомогою програмного коду. Спочатку створюється модель даних у програмному коді, а потім за допомогою міграцій Entity Framework створює або оновлює структуру бази даних відповідно до цієї моделі. Це дозволяє розробникам легко контролювати структуру бази даних та вносити зміни, коли це необхідно.

Entity Framework – це модуль об'єктно-реляційного відображення (ORM) для .NET платформи. Entity Framework дозволяє розробникам працювати з даними у вигляді об'єктів і властивостей, відокремлюючи їх від структури бази даних.[17] Це спрощує розробку баз даних, роблячи код більш читабельним та підтримуваним. Завдяки Entity Framework розробники можуть фокусуватися на

логіці бізнес-процесів, а не на низькорівневих деталях взаємодії з базою даних.[18]

Після визначення моделей, потрібно створити міграції. Міграції – це спосіб, яким Entity Framework відслідковує зміни у моделях коду та приймає рішення про відповідне відображення цих змін в структурі бази даних.

Кожна міграція відображає зміну у структурі бази даних, і її можна вважати як версію схеми бази даних. Якщо розробник вносить зміни в моделі коду, наприклад, додає новий клас або змінює властивості існуючого, він може створити нову міграцію, яка відобразить ці зміни.

Однак створення міграції – це тільки перший крок. Після створення міграції потрібно застосувати її до бази даних. Це означає, що Entity Framework виконає відповідні SQL команди, щоб оновити структуру бази даних відповідно до моделей коду.

Цей процес може включати створення нових таблиць, додавання або видалення стовпців, зміну типів даних стовпців та інші операції. Розробник може контролювати цей процес, перевіряючи SQL команди, що генеруються Entity Framework, і, якщо потрібно, вносячи власні зміни.

Таким чином, завдяки Code First та Entity Framework, розробники можуть впевнено працювати з моделями бази даних, знаючи, що будь-які зміни, які вони вносять у код, можуть бути безпечно та ефективно відображені в структурі бази даних.

Процес створення міграцій та оновлення бази даних є невід'ємною частиною розробки програмного забезпечення з використанням Entity Framework і Code First.[19] Це не тільки дозволяє розробникам легко вносити зміни в структуру бази даних, але й сприяє вищому рівню адаптивності програмного забезпечення до змін у бізнес-вимогах та забезпечує більш гладкий процес внесення змін в продукт.

Для створення міграції потрібно виконати команду Add-Migration в терміналі Package manager, у випадку якщо міграцію потрібно прибрати, можна виконати команду Remove-Migration.

Важливо зазначити, що міграції дозволяють міняти структуру бази даних послідовно та безпечно, без ризику втрати даних або руйнування цієї структури.[20] Кожна міграція може бути відкочена, що дозволяє розробникам відновити базу даних до попереднього стану, якщо це потрібно. Такий підхід забезпечує безпеку та надійність у процесі розробки.

Для оновлення бази даних відповідно до міграцій потрібно виконати команду Update-Database в терміналі Package manager.

Отже, використання Code First та Entity Framework у поєднанні зі створенням міграцій та оновленням бази даних виявилося ефективним рішенням при розробці садового центру «ФасолькаСад».

За допомогою Entity Framework було створено структуру бази даних, зображену на рисунку 2.1.

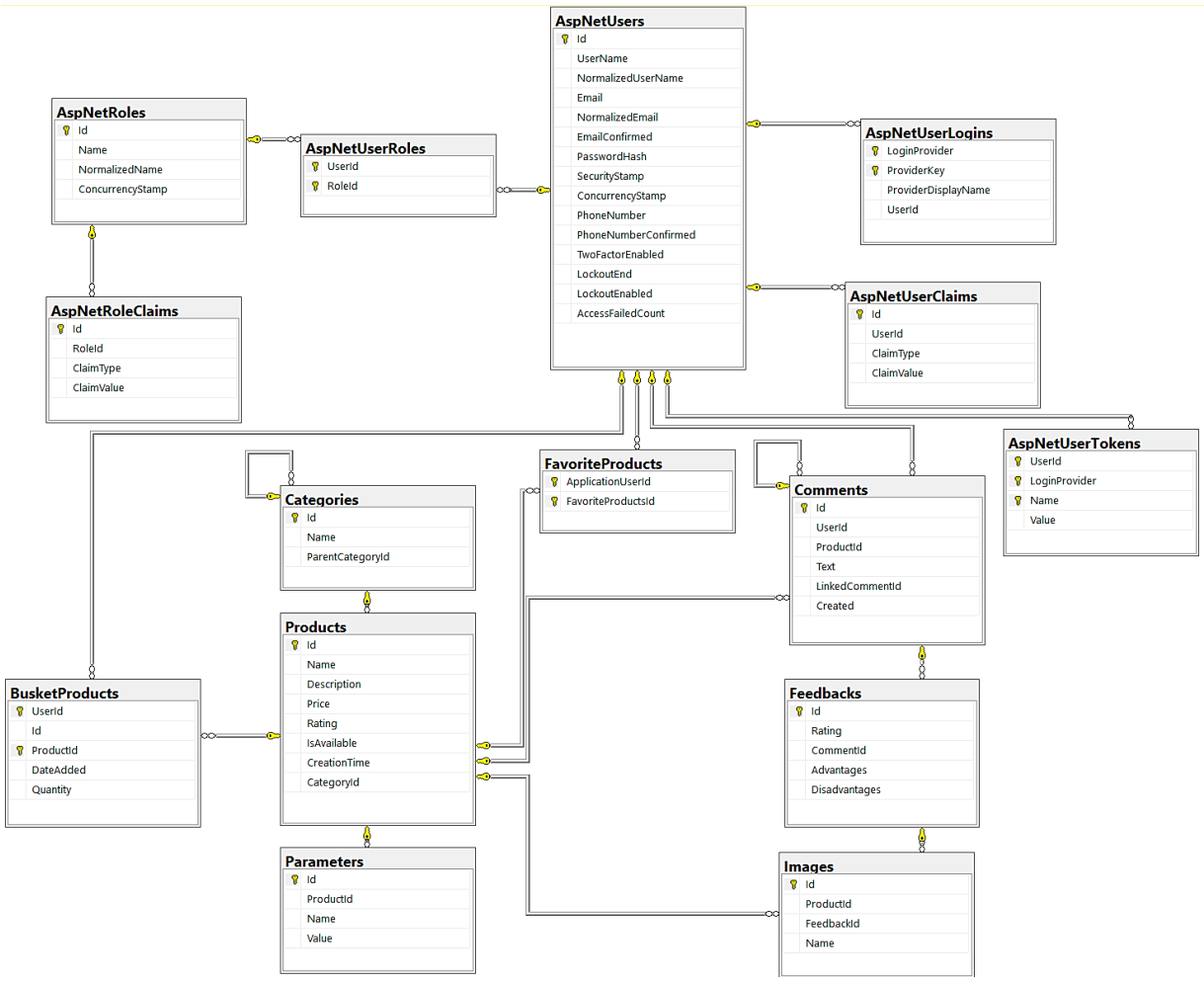

Рисунок 2.1 – структура бази даних веб-сайту садового центру «ФасолькаСад»

Ця конструкція відображає відносини між моделями у проекті, що спрощує взаємодію з базою даних за допомогою коду.

Також в даній структурі варто відмітити наступні таблиці:

– AspNetRoleClaims.

– AspNetRoles.

– AspNetUserRoles.

– AspNetUserLogins.

– AspNetUserClaims.

– AspNetUserTokens.

– AspNetUsers.

Дані таблиці використовуються системою автентифікації Identity Server та зберігають інформацію про користувачів.[21]

Для налаштування складних відносин між таблицями було використано можливості OnModelCreating. Це метод класу контексту який надається Entity Framework для визначення різних конфігурацій бази даних. Також варто зазначити що в даному класі прописуються усі моделі які розробник бажає бачити в базі даних у вигляді таблиці.

Код методу OnModelCreating наведено в лістингу 2.1.

Лістинг 2.1 – код методу OnModelCreating

```
protected override void OnModelCreating(ModelBuilder builder)
{
base.OnModelCreating(builder);

builder.Entity<ApplicationUser>()
                .HasMany(u => u.FavoriteProducts)
                .WithMany()
                .UsingEntity(j => j.ToTable("FavoriteProducts"));

builder.Entity<BusketProduct>()
                .HasKey(bp => new { bp.ProductId, bp.UserId });

builder.Entity<BusketProduct>()
                .HasOne(bp => bp.User)
                .WithMany(u => u.BusketProducts)
                .HasForeignKey(bp => bp.UserId);
```

```
builder.Entity<BusketProduct>()
                .HasOne(bp => bp.Product)
                .WithMany()
                .HasForeignKey(bp => bp.ProductId);
}
```

В даному коді прописані налаштування для моделі BusketProduct в контексті бази даних та відносин ApplicationUser і FavoriteProducts об’єкти останньої не мають окремого представлення в якості моделі, натомість ця таблиця використовується для зберігання вподобаних товарів користувачів зберігаючи тільки ідентифікатори, тобто це реалізація відношення «Багато до багатьох».

2.3 Обслуговування та наповнення сайту

Обслуговування та наповнення сайту садового центру «ФасолькаСад» відіграють центральну роль в забезпеченні стабільної роботи ресурсу і його привабливості для користувачів. Ці два етапи вимагають безперервної уваги, професійного підходу і ретельного планування.

Обслуговування сайту включає в себе регулярний моніторинг системи і сервера, раннє виявлення та виправлення можливих проблем або збоїв. Критично важливими є такі процеси як резервне копіювання даних та своєчасне встановлення оновлень, що допомагають зберегти інформацію безпечною і актуальною. Це в свою чергу гарантує стабільну роботу сайту і забезпечує комфорт для користувачів, які відвідують сайт в пошуках товарів або інформації.

Що стосується наповнення сайту, цей процес охоплює додавання нового контенту на веб-ресурс. Він може включати в себе оновлення інформації про товари, розширення каталогу, публікацію інформативних статей та інших матеріалів, пов'язаних з садівництвом. Цей аспект є надзвичайно важливим, оскільки актуальний та релевантний контент не тільки приваблює нових відвідувачів, але і поліпшує позицію сайту в результатах пошуку, що сприяє збільшенню видимості сайту.

2.4 Тестування сайту

Тестування є критично важливим етапом у процесі розробки веб-сайту, і для проекту «ФасолькаСад» було застосовано два основних підходи до тестування – юніт тестування для серверної частини та мануальне тестування для клієнтської частини.

Для серверної частини сайту було написано юніт-тести. Юніт-тестування - це процес перевірки окремих компонентів коду, зазвичай на рівні функцій або методів. Це дає можливість переконатися, що кожна частина серверної логіки працює правильно та відповідає вимогам. Регулярне юніт-тестування може підвищити надійність коду, спрощуючи процес виявлення та виправлення помилок на ранніх та пізніх етапах розробки.[22]

З іншого боку, клієнтську частину сайту було протестовано за допомогою мануального тестування. Мануальне тестування включає в себе процес перевірки функціональності та дизайну сайту, як це б виконував реальний користувач. Це включає перевірку інтерфейсу, перехід по посиланням, використання форм введення даних, тестування реакції сайту на різні дії користувача та перевірку загального користувацького досвіду. Це дозволило переконатись, що сайт є дружнім до користувача, інтуїтивно зрозумілим та легким у використанні.[23]

Дизайн повинен бути адаптивним, тобто коректно виглядати, як на мобільний пристроях так і на персональних комп’ютерах. Для цього розглянемо вигляд сайту на телефоні, що зображено на Рисунку 2.2 та вигляд сайту на комп’ютері. Подивитись на вигляд сайту на телефонах можна використовуючи інструменти налагодження вбудовані в браузер Google Chrome, перевага використання цих інструментів в тому, що можна побачити як сайт виглядатиме при розширенні реальних телефонів та планшетів без необхідності мати їх в наявності при тестуванні.

Рисунок 2.2 – вигляд каталогу веб-сайту садового центру «ФасолькаСад» на телефоні iPhone 12 Pro

Можна побачити, що при розширенні для телефонів відсутні графічні баги та наявні колапсуючі кнопки, наприклад в правому верхньому кутку, що дозволяють адекватно відобразити усю необхідну інформацію на екрані без перевантаження інтерфейсу.

Також розглянемо вигляд сайту при розширенні для комп’ютерів, зображеного на рисунку 2.3.

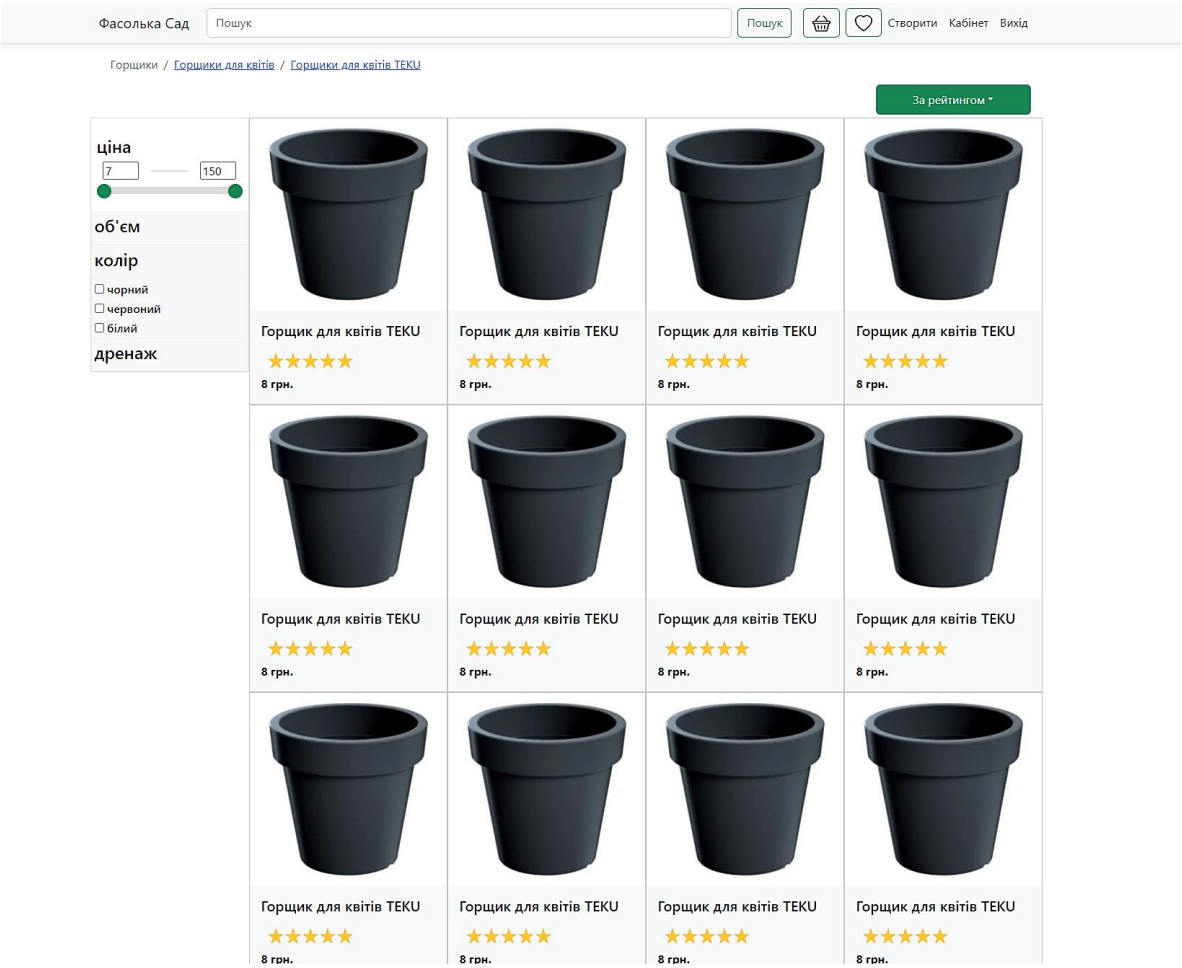

Рисунок 2.3 – вигляд каталогу веб-сайту садового центру «ФасолькаСад» для адміністратора при розширенні для комп’ютерів

Можемо побачити, що вимогу мінімалістичного дизайну, яка була зазначена раніше, було дотримано. Також, що увесь контент, який був схований за колапсуючими кнопками на мобільній версії, відображається коректно.

Також варто навести приклад сценарію мануального тестування, в даному випадку «Додавання товару до вподобаних». Його кроки наступні:

– Увійти в аккаунт для тестування.

– Переглянути наявність товарів у списку вподобаних, якщо присутні, то прибрати їх.

– Обрати товар з каталогу, та перейти на його сторінку.

– Натиснути на зображення серця для додавання товару до вподобаних.

– Перейти до списку вподобаних товарів та впевнитись що присутній обраний товар.

Сценарій мануального тестування "Додавання товару до вподобаних" є важливим ілюстративним прикладом, який демонструє, як повинна працювати одна з ключових функцій веб-сайту садового центру «ФасолькаСад». Кроки, описані вище, прямолінійно ведуть тестувальника через процес взаємодії з сайтом, як це б робив звичайний користувач. Кожен крок цього процесу вимагає належної роботи різних елементів сайту. Від функції входу до системи до детальної сторінки товару, а також самого процесу додавання товару до списку вподобаних – усе це формує одну велику, узгоджену систему.

Виконання даного сценарію мануального тестування дозволяє впевнитись, що функціонал, що відноситься до списку вподобаних товарів, працює коректно.

Об'єднуючи юніт тестування серверного коду і мануальне тестування клієнтської частини, можна забезпечити якість та надійність розробленого веб-сайту.

2.4.1 Оцінка отриманих результатів веб-сайту садового центру «ФасолькаСад»

Оцінка отриманих результатів є важливою частиною процесу розробки, особливо після фази тестування. Це допомагає розробнику зрозуміти, наскільки успішним був проект і які можуть бути наступні кроки для його поліпшення.

Після аналізу отриманих результатів, можна зробити наступні підсумки:

– Функціональність – Було успішно реалізовано основні функціональні можливості садового центру "ФасолькаСад", включаючи каталог товарів, сторінку товару, особистий кабінет користувача, інтерфейс для додавання товарів власником, сторінку "Про нас", корзину та вподобані товари. Всі ці функції були вдало реалізовані та працюють належним чином.

– Відповідність вимогам – Веб-сайт відповідає поставленим вимогам та специфікаціям проекту. Функціональність та дизайн відповідають очікуванням

користувачів, а також враховують потреби власників товарів та адміністраторів сайту.

– Користувацький досвід – Під час мануального тестування було отримано позитивний зворотний зв'язок щодо користувацького досвіду на сайті. Користувачі зазначили зручність навігації, чіткість та зрозумілість інтерфейсу, а також швидкість завантаження сторінок. Це позитивно впливає на задоволеність користувачів та ймовірність повторного відвідування сайту.

– Мобільна оптимізація – Веб-сайт було оптимізовано для різних розмірів екранів, включаючи мобільні пристрої. Така оптимізація дозволяє користувачам безперешкодно користуватися сайтом на різних пристроях, забезпечуючи хороший користувацький досвід незалежно від використаного пристрою.

– Безпека – Веб-сайт реалізує необхідні протоколи безпеки та захисту даних, включаючи захист від SQL Injection, XSS та CSRF атак. Така міцна система безпеки дозволяє забезпечити захист даних користувачів та довіру до веб-сайту.

– Виявлені проблеми – Під час тестування було виявлено та виправлено кілька незначних проблем, таких як неправильне відображення деяких елементів на різних пристроях та дрібні помилки в тексті. Однак, загальний обсяг виявлених проблем був незначним, що свідчить про високу якість розробленого веб-сайту.

На основі цих результатів можна зробити висновок, що розробка садового центру "ФасолькаСад" була успішною. Проект відповідає вимогам та очікуванням, забезпечуючи функціональність та зручний користувацький досвід. За необхідності, можна розглянути поліпшення та вдосконалення на основі отриманого зворотного зв'язку.

2.5 Висновки до другого розділу

У процесі розробки веб-сайту садового центру «ФасолькаСад» було зрозуміло, що ефективне поєднання різних технологій та інструментів, які

включають фреймворки, менеджери пакунків, бази даних та редактори коду, є важливим для створення якісного веб-рішення.

Під час розробки було використано Code First та Entity Framework, які спрощують процес створення бази даних і роблять його більш інтуїтивно зрозумілим. Використання системи міграцій дозволило контролювати зміни в структурі бази даних, що сприяє гнучкості і масштабованості проекту.

Процес обслуговування та наповнення сайту було здійснено самостійно. Це дозволило повністю контролювати кожен аспект проекту, а також отримати цінний досвід з керування всіма сторонами розробки веб-сайту.

Під час тестування веб-сайту було використано юніт-тестування для серверної частини та мануальне тестування для клієнтської частини. Це підходи дозволили забезпечити надійність та якість веб-сайту.

Оцінка отриманих результатів показала, що встановлені цілі були досягнуті, і сайт «ФасолькаСад» відповідає вимогам та очікуванням. Отримані знання та досвід є цінними для подальшого розвитку та поліпшення веб-сайту.

У підсумку цей проект показав значимість правильного вибору технологій, ефективного проектування бази даних, адекватного обслуговування та тестування сайту для досягнення якісного результату.

РОЗДІЛ 3. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

3.1 Ергономічні проблеми безпеки життєдіяльності

У контексті розробки веб-сайту, важливо враховувати ергономічні проблеми безпеки життєдіяльності, що стосуються роботи розробника. Ергономіка спрямована на створення оптимальних умов праці, що сприяють збереженню здоров'я та забезпечують комфорт та ефективність працівника.

Ключові ергономічні проблеми, які можуть виникнути під час розробки веб-сайту, включають розташування робочого місця, освітлення, розміщення монітора та клавіатури, регулярні паузи та управління стресом. Розробник повинен мати комфортне та безпечне робоче місце з відповідною організацією простору, правильною висотою столу та підтримкою відповідної позиції тіла. Надійне та природне освітлення є важливим, а також зручне розміщення монітора та клавіатури, що дозволяє підтримувати правильну позицію без напруги. Регулярні паузи та вправи можуть допомогти запобігти м'язовим напругам та розвитку синдрому карпального каналу, а також інших проблем, пов'язаних з роботою за комп'ютером. Нарешті, розробники повинні знаходити способи керування стресом та підтримувати позитивну робочу атмосферу, щоб зберегти психічне та емоційне благополуччя.[24]

Також важливо забезпечити розробникам веб-сайту адекватну організацію робочого процесу, раціональне планування завдань та часу, що сприятиме підвищенню продуктивності та зниженню ризику виникнення втоми та перевантаження.

Забезпечення ергономічних принципів під час розробки веб-сайту допомагає зберегти здоров'я розробників та підвищує ефективність їх роботи. Дотримання принципів охорони праці та безпеки життєдіяльності є важливим кроком у забезпеченні стабільної та продуктивної розробки веб-сайту, а також відповідності законодавству України.

3.2 Загальні вимоги безпеки з охорони праці для користувачів ПК

Загальні вимоги безпеки з охорони праці для користувачів ПК є основою для створення безпечного та здорового робочого середовища. Компанії та працівники повинні дотримуватись цих вимог, забезпечуючи безпеку, захист здоров'я та ефективність праці при роботі з комп'ютером.

Загальні вимоги:

– дія інструкції з охорони праці при роботі з персональним комп’ютером поширюється на всі підрозділи підприємства, де виконують роботи з комп’ютером;

– інструкція з охорони праці для користувачів комп’ютерів розроблена відповідно до нормативних документів, таких як Положення про розробку інструкцій з охорони праці, Типове положення про проведення навчання і перевірки знань з питань охорони праці та інші відповідні нормативні акти;

– користувачів, які використовують персональні комп'ютери, необхідно інструктувати згідно з цією інструкцією перед початком роботи та регулярно кожні 6 місяців. Результати інструктажу фіксуються в Журналі реєстрації інструктажів з питань охорони праці на робочому місці;

– користувачі зобов'язані дотримуватись правил внутрішнього трудового розпорядку, не допускати сторонніх осіб до свого робочого місця, виконувати правила охорони праці та пожежної безпеки, мати знання правил надання домедичної допомоги та вміти користуватись первинними засобами пожежогасіння;

– користувачам необхідно дбати про особисту безпеку і здоров'я, а також про безпеку і здоров'я довкілля під час виконання робіт на персональному комп'ютері та перебування на території підприємства.

Основні небезпечні та шкідливі виробничі фактори:

– підвищений рівень статичної електрики;

– нерівномірність розподілу яскравості в полі зору;

– підвищена яскравість світлового зображення;

– ураження електричним струмом;

– напруга зору та уваги;

– тривалі статичні навантаження.

Розміщення та організація робочого місця:

– робоче місце повинно забезпечувати оптимальну робочу позу користувача, простір для розміщення та зручне користування обладнанням;

– монітор повинен бути встановлений на відстані 600-700 мм від очей користувача, залежно від розміру екрана;

– клавіатуру розміщують на робочому або окремому столі на відстані 100-300 мм від краю з боку користувача, кут нахилу клавіатури повинен забезпечувати комфортне користування;

– робочий стіл повинен бути конструктивно пристосованим для розміщення обладнання та забезпечувати зручний доступ до необхідних ресурсів;

– крісло повинно забезпечувати підтримку раціональної робочої пози та можливість зміни пози під час тривалої роботи;

– забезпечення оптимальної робочої пози користувача полягає в розташуванні основних засобів праці у зоні зорового спостереження та моніторного поля;

– ПК та оргтехніку слід розміщувати на рівній та стійкій поверхні;

– розетка біля ПК повинна бути доступною для відключення в разі аварійних ситуацій;

– при переміщенні ПК та периферійних пристроїв слід відключати їх від живлення та дотримуватись правил безпеки щодо кабелів живлення;

– при підключенні ПК до електромережі необхідно використовувати справні штепсельні з'єднання та електророзетки зі спеціальними контактами для під'єднання нульового захисного провідника.[25]

Ці вимоги є обов'язковими для всіх користувачів персональних комп'ютерів та спрямовані на забезпечення безпеки та здоров'я працівників під час роботи з комп'ютерами.

Огляд робочого місця та підготовка до роботи:

– перед початком роботи необхідно оглянути робоче місце і впевнитись, що на ньому немає сторонніх предметів. Також важливо переконатись, що всі обладнання і блоки ПК з'єднані з системним блоком за допомогою з'єднувальних шнурів;

– слід перевірити надійність встановлення апаратури на робочому столі. Монітор не повинен стояти на краю стола;

– перевірити загальний стан апаратури, справність електропроводки, з'єднувальних шнурів, штепсельних вилок, розеток та заземлення захисного екрана;

– вставити вилку в розетку та переконатись, що вона міцно тримається, важливо уникати вставляння або виймання вилки мокрими руками;

– налаштувати і зафіксувати висоту крісла та зручний для користувача нахил спинки;

– відрегулювати яскравість свічення та контрастність монітора;

– при виявленні будь-яких несправностей слід повідомити керівника робіт та утриматись від початку роботи до їх усунення.[26]

Інструкції з питань безпеки під час роботи:

– під час роботи на ПК необхідно стійко встановити клавіатуру на робочому столі, не допускаючи її хитання. При цьому необхідно передбачити можливість поворотів та переміщень клавіатури. Клавіатуру слід розмістити на відстані не менше 100 мм від краю столу в оптимальній зоні моніторного поля. Під час роботи на клавіатурі слід сидіти рівно і не напружуватися. Щоб зменшити навантаження під час роботи з комп'ютерною мишею, необхідно забезпечити велику вільну поверхню столу для переміщення миші та зручний упор ліктьового суглоба. Періодично при вимкненому комп'ютері слід прибирати пил із поверхонь апаратури за допомогою спеціальних серветок;

– при роботі з ПК заборонено самостійно розбирати або ремонтувати системний блок, монітор, клавіатуру, комп'ютерну мишу та іншу апаратуру;

– тривалість безперервної роботи за ПК не повинна перевищувати 2 годин. Після цього необхідно зробити 15-хвилинну перерву. Якщо виникає зоровий дискомфорт або інші неприємні відчуття, слід зробити коротку перерву;

– для зменшення нервово-емоційного напруження, стомлення зорового аналізатора, покращення мозкового кровообігу та запобігання втомі, під час декількох перерв виконуйте комплекс вправ.

Таким чином, щоб забезпечити безпечне та продуктивне робоче середовище, слід уважно стежити за дотриманням цих вимог. Компанії та їх працівники повинні постійно робити це пріоритетом, щоб забезпечити здоров'я та благополуччя всіх, хто працює з ПК.

3.3 Висновки до третього розділу

У розділі «Безпека життєдіяльності, основи охорони праці» були розглянуті ергономічні проблеми безпеки життєдіяльності та загальні вимоги безпеки з охорони праці для користувачів персональних комп'ютерів. Ці аспекти є важливими для забезпечення безпеки та здоров'я працівників, особливо тих, хто виконує роботу, пов'язану з використанням комп'ютера.

Ергономічні проблеми безпеки життєдіяльності включають аналіз робочого місця, розміщення обладнання, належну позицію тіла під час роботи, а також забезпечення оптимального освітлення та вентиляції. Ці заходи спрямовані на попередження негативного впливу робочого середовища на здоров'я працівників та зменшення ризику травм.

Висновки з цього розділу підкреслюють важливість дотримання вимог безпеки життєдіяльності та охорони праці під час роботи з комп'ютером. Це сприяє забезпеченню безпеки, здоров'я та комфорту працівників, а також попередженню можливих травм та професійних захворювань, пов'язаних з використанням комп'ютерної техніки.

ВИСНОВКИ

Усі цілі та задачі були успішно виконані в ході виконання кваліфікаційної роботи для отримання бакалаврського ступеня, яка включала розробку веб-сайту садового центру «ФасолькаСад» за допомогою ASP.NET Core та React.

В першому розділі було досягнуто наступне:

– проведено аналіз садових центрів та виявлено особливості їх діяльності;

– розроблено структуру веб-сайту та поставлено завдання для подальшої реалізації;

– визначено стадії та етапи розробки веб-сайту;

– обрано середовище проєктування та сервіс для авторизації.

В другому розділі виконано наступні дії:

– обрано засоби розробки, в тому числі фреймворки;

– сформовано базу даних для садового центру «ФасолькаСад»;

– організовано процес обслуговування та наповнення сайту;

– проведено всебічне тестування та оцінено кінцевий продукт.

У розділі «Безпека життєдіяльності, основи охорони праці» були вивчені ергономічні питання, пов'язані з безпекою життєдіяльності, а також були висвітлені універсальні вимоги до захисту праці для користувачів персональних комп'ютерів.

Таким чином, у процесі виконання роботи вдалося розробити функціональний та зручний в користуванні веб-сайт садового центру «ФасолькаСад», що задовольняє всі вимоги до сучасних веб-ресурсів.

ПЕРЕЛІК ДЖЕРЕЛ

1. React – JavaScript-бібліотека для створення користувацьких інтерфейсів. URL: https://uk.legacy.reactjs.org/ (дата звернення: 15.04.2023).

2. ASP.NET | Open-source web framework for .NET. URL: https://dotnet.microsoft.com/apps/aspnet (дата звернення: 16.04.2023).

3. Microsoft Data Platform | Microsoft. URL: https://www.microsoft.com/uk-ua/sql-server/ (дата звернення: 16.04.2023).

4. Keycloak. URL: https://www.keycloak.org/ (дата звернення: 18.04.2023).

5. Active Directory Domain Services Overview | Microsoft Learn. URL: https://docs.microsoft.com/uk-ua/windows-server/identity/ad-ds/get-started/virtual-dc/active-directory-domain-services-overview (дата звернення: 18.04.2023).

6. Duende Software. URL: https://duendesoftware.com/products/identityserver (дата звернення: 15.05.2023).

7. Visual Studio: IDE and Code Editor for Software Developers and Teams. URL: https://visualstudio.microsoft.com/ (дата звернення: 20.04.2023).

8. Visual Studio Code – Code Editing. Redefined. URL: https://code.visualstudio.com/ (дата звернення: 20.04.2023).

9. Download SQL Server Management Studio (SSMS) - SQL Server Management Studio (SSMS) | Microsoft Learn. URL: https://docs.microsoft.com/uk-ua/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver15 (дата звернення: 20.04.2023).

10. npm. URL: https://www.npmjs.com/ (дата звернення: 21.04.2023).

11. NuGet Gallery | Home. URL: https://www.nuget.org/ (дата звернення: 21.04.2023).

12. GitHub - RicoSuter/NSwag: The Swagger/OpenAPI toolchain for .NET, ASP.NET Core and TypeScript. URL: https://github.com/RicoSuter/NSwag (дата звернення: 22.04.2023).

13. .NET Core – загальне середовище виконання для .NET платформи. URL: https://dotnet.microsoft.com/ (дата звернення: 23.04.2023).

14. EF Core – фреймворк для роботи з базами даних. URL: https://dotnet.microsoft.com/apps/aspnet/entity-framework (дата звернення: 23.04.2023).

15. ASP.NET Core – Project Structure. URL: https://www.tutorialsteacher.com/core/aspnet-core-application-project-structure (дата звернення: 24.04.2023).

16. How to Structure Your React Project. URL: https://daveceddia.com/react-project-structure/ (дата звернення: 24.04.2023).

17. ADO.NET. URL: https://uk.zahn-info-portal.de/wiki/ADO.net (дата звернення 25.04.2023).

18. Creating and Configuring a Model – EF Core | Microsoft Learn. URL: https://docs.microsoft.com/en-us/ef/core/modeling/ (дата звернення: 25.04.2023).

19. How to create React and Asp.Net Core App. URL: https://dotnetdetail.net/how-to-create-react-and-asp-net-core-app-using-visual-studio-2017/ (дата звернення 25.04.2023).

20. Migrations Overview – EF Core | Microsoft Learn. URL: https://docs.microsoft.com/en-us/ef/core/managing-schemas/migrations/ (дата звернення: 25.04.2023).

21. Introduction to Identity on ASP.NET Core | Microsoft Learn. URL: https://docs.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-5.0&tabs=visual-studio (дата звернення: 25.04.2023).

22. Testing in .NET – .NET | Microsoft Learn. URL: https://docs.microsoft.com/en-us/dotnet/core/testing/ (дата звернення: 25.04.2023).

23. Manual Testing Tutorial: What is, Types, Concepts. URL: https://www.guru99.com/manual-testing.html (дата звернення: 25.04.2023).

24. Ергономічні проблеми безпеки життєдіяльності. URL: http://um.co.ua/13/13-9/13-91108.html.

25. Інструкція з охорони праці при роботі на ПК/персональному комп’ютері. URL: https://pro-op.com.ua/article/485-nstruktsya-z-ohoroni-prats-pri-robot-na-personalnomu-kompyuter.

26. Інструкція з охорони праці при роботі з комп'ютером. URL: https://osvita-docs.com/node/41.

ДОДАТКИ

Додаток А

Код програми в файлі Program.cs

```
using AutoMapper;
using FasolkaSadProject;
using FasolkaSadProject.Data;
using FasolkaSadProject.Interfaces;
using FasolkaSadProject.Models;
using FasolkaSadProject.Repositories;
using FasolkaSadProject.Services;
using Microsoft.AspNetCore.ApiAuthorization.IdentityServer;
using Microsoft.AspNetCore.Authentication;
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.AspNetCore.Identity;
using Microsoft.EntityFrameworkCore;
using NSwag;
using NSwag.Generation.Processors.Security;

var builder = WebApplication.CreateBuilder(args);
builder.Services.AddOpenApiDocument(configure =>
{
    configure.Title = "HikeApp API";
    configure.GenerateEnumMappingDescription = true;
    configure.OperationProcessors.Add(new
OperationSecurityScopeProcessor("Bearer"));
    configure.DocumentProcessors.Add(new
SecurityDefinitionAppender("Bearer", new OpenApiSecurityScheme
    {
        Type = OpenApiSecuritySchemeType.ApiKey,
        Name = "Authorization",
        In = OpenApiSecurityApiKeyLocation.Header
    }));
    configure.OperationProcessors.Add(new
SampleHeaderOperationProcessor());
});

// Add services to the container.
var                        connectionString                          =
builder.Configuration.GetConnectionString("DefaultConnection")   ??
throw     new     InvalidOperationException("Connection     string
'DefaultConnection' not found.");
builder.Services.AddDbContext<ApplicationDbContext>(options =>
    options.UseSqlServer(connectionString));
builder.Services.AddDatabaseDeveloperPageExceptionFilter();

builder.Services.AddDefaultIdentity<ApplicationUser>(options     =>
options.SignIn.RequireConfirmedAccount = true)
    .AddRoles<IdentityRole>()
    .AddEntityFrameworkStores<ApplicationDbContext>();
```

```
builder.Services.AddIdentityServer()
    .AddApiAuthorization<ApplicationUser, ApplicationDbContext>();

builder.Services.AddAuthentication()
.AddIdentityServerJwt();

builder.Services.Configure<JwtBearerOptions>(
    IdentityServerJwtConstants.IdentityServerJwtBearerScheme,
    options =>
    {
        var onTokenValidated = options.Events.OnTokenValidated;

        options.Events.OnTokenValidated = async context =>
        {
            await onTokenValidated(context);
        };

        options.TokenValidationParameters = new
Microsoft.IdentityModel.Tokens.TokenValidationParameters
        {
            ValidateAudience = false,
            ValidateIssuer = false
        };
    });

builder.Services.AddControllersWithViews();

builder.Services.AddScoped<ICategoryRepository,
CategoryRepository>();
builder.Services.AddScoped<IProductRepository,
ProductRepository>();
builder.Services.AddScoped<ICommentRepository,
CommentRepository>();
builder.Services.AddScoped<IFeedbackRepository,
FeedbackRepository>();
builder.Services.AddScoped<IImageRepository, ImageRepository>();
builder.Services.AddScoped<IFavoriteProductRepository,
FavoriteProductRepository>();
builder.Services.AddScoped<IBusketProductRepository,
BusketProductRepository>();

builder.Services.AddSingleton<SettingsService>();

builder.Services.Configure<IdentityOptions>(options =>
{
    options.User.RequireUniqueEmail = true;
});

var mapperConfig = new MapperConfiguration(mc =>
{
    mc.AddProfile(new MappingProfile());
});
```

```
IMapper mapper = mapperConfig.CreateMapper();
builder.Services.AddSingleton(mapper);

builder.Services.AddRazorPages();

var app = builder.Build();

using var serviceScope = app.Services.CreateScope();

CreateRoles(serviceScope.ServiceProvider).Wait();

// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseMigrationsEndPoint();
}
else
{
    // The default HSTS value is 30 days. You may want to change
this for production scenarios, see https://aka.ms/aspnetcore-hsts.
    app.UseHsts();
}

app.UseHttpsRedirection();
app.UseStaticFiles();

app.UseSwaggerUi3(settings =>
{
    settings.Path = "/swagger";
    settings.DocumentPath = "/swagger/specification.json";
});

app.UseRouting();

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

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

app.MapFallbackToFile("index.html");

app.Run();

CreateRoles(app.Services).Wait();

async Task CreateRoles(IServiceProvider serviceProvider)
{
```

```
    //initializing custom roles
    var                          RoleManager                          =
serviceProvider.GetRequiredService<RoleManager<IdentityRole>>();
    var                          UserManager                          =
serviceProvider.GetRequiredService<UserManager<ApplicationUser>>()
;
    var                       settingsService                         =
serviceProvider.GetRequiredService<SettingsService>();

    string[] roleNames = { "Admin", "Store-Manager", "Member" };
    IdentityResult roleResult;

    foreach (var roleName in roleNames)
    {
        var              roleExist              =              await
RoleManager.RoleExistsAsync(roleName);
        // ensure that the role does not exist
        if (!roleExist)
        {
            //create the roles and seed them to the database:
            roleResult    =    await    RoleManager.CreateAsync(new
IdentityRole(roleName));
        }
    }

    if (!settingsService.SettingExists("adminEmail"))
    {
        return;
    }

    var                           adminEmail                           =
settingsService.GetSettingStringValue("adminEmail");
    // find the user with the admin email
    var _user = await UserManager.FindByEmailAsync(adminEmail);

    // check if the user exists
    if (_user == null)
    {
        //Here you could create the super admin who will maintain
the web app
        var poweruser = new ApplicationUser
        {
            UserName = adminEmail,
            Email = adminEmail,
        };
        string adminPassword = "Password123!";

        var            createPowerUser            =            await
UserManager.CreateAsync(poweruser, adminPassword);
        if (createPowerUser.Succeeded)
        {
            //here we tie the new user to the role
```

```
            await UserManager.AddToRoleAsync(poweruser, "Admin");
            var              token              =              await
UserManager.GenerateEmailConfirmationTokenAsync(poweruser);
            await UserManager.ConfirmEmailAsync(poweruser, token);
        }
    }
}
```

Додаток Б

Текст програми в файлі Home.js

```
import React, { useEffect, useState } from 'react';
import { Collapse, Button } from 'reactstrap';
import { RatingStar } from "rating-star";
import { CategoryClient, SearchClient, SortType } from "../app/web-
api-client.ts";
import createApiClient from "../app/create-api-client.ts";
import RangeSlider from './RangeSlider'


export default function Home(props) {

    const [orderSetting, setOrderSetting] =
useState(SortType.Rating);
    const [currentCategoryId, setCurrentCategoryId] = useState(1);
    const [categories, setCategories] = useState([]);
    const [orderButtonText, setOrderButtonText] = useState("За
рейтингом");
    const [products, setProducts] = useState([]);
    const [currentPage, setCurrentPage] = useState(1);
    const [totalPages, setTotalPages] = useState();
    const [priceMaxRange, setPriceMaxRange] = useState(['', '']);
    const [priceRange, setPriceRange] = useState(['', '']);
    const [parameters, setParameters] = useState([]);
    const [checkedParameters, setCheckedParameters] = useState([]);
    const [fetchToggle, setFetchToggle] = useState(false);

    const PRODUCTS_PER_PAGE = 24

    const [isOpen, setIsOpen] = useState(true);

    useEffect(() => {
        async function fetchCategories() {
            const response = await
createApiClient(CategoryClient).get(currentCategoryId, false,
true);
            if (response) {
                setCategories(response);
            }
        }

        fetchCategories();
    }, [currentCategoryId]);

    useEffect(() => {
        getSearchResult();
    }, [orderSetting, currentCategoryId, priceRange, currentPage,
fetchToggle]);
```

```
    useEffect(() => {
        getPameters();
    }, [currentCategoryId]);

    useEffect(() => {
        setFetchToggle((prevFetchToggle) => !prevFetchToggle);
    }, [checkedParameters]);

    useEffect(() => {
        setPriceRange([priceMaxRange[0], priceMaxRange[1]]);
    }, [priceMaxRange]);

    useEffect(() => {
        const handleResize = () => {
            setIsOpen(window.innerWidth > 576);
        };

        window.addEventListener('resize', handleResize);

        handleResize();

        return () => window.removeEventListener('resize',
handleResize);
    }, []);

    async function getSearchResult() {
        const checkedParametersArray = parameters.filter((_, index)
=> checkedParameters[index]);

        const response = await
createApiClient(SearchClient).search(PRODUCTS_PER_PAGE,
priceRange[0], priceRange[1], currentCategoryId,
checkedParametersArray, orderSetting, (currentPage - 1) *
PRODUCTS_PER_PAGE, PRODUCTS_PER_PAGE);
        if (response == null) {
            return
        }

        if (response.minPrice !== priceMaxRange[0] ||
response.maxPrice !== priceMaxRange[1]) {
            setPriceMaxRange([response.minPrice,
response.maxPrice]);
        }

        setProducts(response.products);
        setTotalPages(response.pageCount);
    }

    async function getPameters() {
        const response = await
createApiClient(CategoryClient).getParameters(currentCategoryId);
```

```
        setCheckedParameters(response.map(() => false));
        setParameters(response);
    }

    function groupParametersByName(parameters) {
        return parameters.reduce((groups, parameter) => {
            const groupName = parameter.name;
            if (!groups[groupName]) {
                groups[groupName] = [];
            }
            groups[groupName].push(parameter);
            return groups;
        }, {});
    }

    function getProductsView() {
        let productItems = ''
        if (Array.isArray(products)) {
            productItems = products.map(
                (item, index) => {
                    return (
                        <div key={index} className="product-item">
                            <div className="card rounded-0 catalog-
item bg-light d-flex flex-column">
                                <img className="d-flex mt-auto mb-
auto                                align-content-center"
src={`${window.location.origin}/api/image/${item.images[0].name}`}
width='100%' />
                                <div className="d-flex">
                                    <div  className="card-text  p-3
">
                                        <h5   className="card-title
text-start">{item.name}</h5>
                                        <RatingStar   id={"catalog-
item-" + index} className="rating" rating={item.rating} />
                                        <br />
                                        <span        className="fw-
bold">{item.price} грн.</span>
                                        <a
href={'product?ProductId=' + item.id} className='stretched-link' />
                                    </div>
                                </div>
                            </div>
                        </div>
                    )
                }
            )
        }
        else {
            productItems    =    <div><h1>Товарів    не    знайдено
:(</h1></div>
        }
```

```
        return (
            <div   className="products-grid"   style={{   display:
'flex', justifyContent: 'flex-start', flexWrap: 'wrap' }}>
                {productItems}
            </div>
        );
    }


    const handleCategoryClick = (categoryId) => {
        setCurrentCategoryId(categoryId);
    };

    function CategoryBreadcrumb() {
        return (
            <nav aria-label="breadcrumb">
                <ol className="breadcrumb">
                    {categories.map((category, index) => (
                        <li   key={index}   className={`breadcrumb-
item${currentCategoryId === category.id ? " active" : ''}`}>
                            {currentCategoryId  ===  category.id  ?
category.name      :      (<a      href='#'      onClick={()      =>
handleCategoryClick(category.id)}>{category.name}</a>)}
                        </li>
                    ))}
                </ol>
            </nav>
        );
    }

    function handleCheckboxChange(index) {
        setCheckedParameters((prevCheckedParameters) => {
            const               newCheckedParameters               =
Array.from(prevCheckedParameters);
            newCheckedParameters[index]                            =
!newCheckedParameters[index];
            return newCheckedParameters;
        });
    }

    function renderCheckboxes() {
        const                  groupedParameters                   =
groupParametersByName(parameters);

        return        Object.entries(groupedParameters).map(([name,
parameterGroup], groupIndex) => (
            <>
                <div    className="parameterCollapseButton    p-1"
key={groupIndex}>
                    <h4
                        data-bs-toggle="collapse"
```

```
                        data-bs-
target={`#collapseParams${groupIndex}`}
                        aria-expanded="false"
                        onClick={(e) => {
                            const             expanded             =
e.currentTarget.getAttribute('aria-expanded') === 'true';
                            e.currentTarget.setAttribute('aria-
expanded', !expanded);
                        }}
                    >
                        {name}
                    </h4>
                </div>
                <div                          className="childWrapper
parameterExpandContent">
                    {parameterGroup.map((parameter, index) => {
                        const             checkedIndex             =
parameters.indexOf(parameter);
                        return (
                            <div key={index} className="collapse p-
1" id={`collapseParams${groupIndex}`}>
                                <label>
                                    <input
                                        type="checkbox"

checked={checkedParameters[checkedIndex]}
                                        onChange={()             =>
handleCheckboxChange(checkedIndex)}
                                    />
                                    {` ${parameter.value}`}
                                </label>
                            </div>
                        );
                    })}
                </div>
            </>
        ));

    }

    function getPageSwitcher() {
        const pageNumbers = [];
        if (totalPages <= 1) {
            return
        }

        if (currentPage > 1) {
            pageNumbers.push(<div     className='pageswitch-button'
onClick={() => setCurrentPage(currentPage - 1)}> &#60; </div>);
            pageNumbers.push(<div     className='pageswitch-button'
onClick={() => setCurrentPage(1)}> 1 </div>);
        }
```

```
        if (currentPage > 3) {
            pageNumbers.push(<div className='m-2'>...</div>);
        }

        if (currentPage > 2) {
            pageNumbers.push(<div     className='pageswitch-button'
onClick={() => setCurrentPage(currentPage - 1)}> {currentPage - 1}
</div>);
        }

        pageNumbers.push(<div className='m-2'>{currentPage}</div>);
        if ((totalPages > 2 || currentPage === 1) && currentPage <
totalPages) {
            pageNumbers.push(<div     className='pageswitch-button'
onClick={() => setCurrentPage(currentPage + 1)}> {currentPage + 1}
</div>);
        }

        if (currentPage < totalPages - 1) {
            pageNumbers.push(<div className='m-2'>...</div>);
            pageNumbers.push(<div     className='pageswitch-button'
onClick={() => setCurrentPage(totalPages)}> {totalPages} </div>);
        }

        if (currentPage < totalPages) {
            pageNumbers.push(<div     className='pageswitch-button'
onClick={() => setCurrentPage(currentPage + 1)}> &#62; </div>);
        }
        return (<>{pageNumbers}</>)
    }


    function toggle() {
        setIsOpen(!isOpen);
    }


    return (
        <>
            <div className='row mb-1 mx-1'>
                <div>{CategoryBreadcrumb()}</div>
                <Button onClick={toggle} className="w-100 w-sm-auto
d-sm-none btn bg-success mb-1">Фільтр пошуку</Button>
                <div className="dropdown col-12 col-sm-2 ps-0 pe-0
ms-auto">
                    <button className="btn btn-secondary dropdown-
toggle bg-success p-2 text-white w-100 w-sm-auto" type="button"
data-bs-toggle="dropdown" aria-expanded="false">
                        {orderButtonText}
                    </button>
```

```
                    <ul className="dropdown-menu">
                        <li><a            className="dropdown-item"
onClick={()    =>    {    setOrderSetting(SortType.PriceHighToLow);
setOrderButtonText("Спершу дорожчі") }}>Спершу дорожчі</a></li>
                        <li><a            className="dropdown-item"
onClick={()    =>    {    setOrderSetting(SortType.PriceLowToHigh);
setOrderButtonText("Спершу дешевші") }}>Спершу дешевші</a></li>
                        <li><a            className="dropdown-item"
onClick={()      =>      {      setOrderSetting(SortType.Rating);
setOrderButtonText("За рейтингом") }}>За рейтингом</a></li>
                        <li><a            className="dropdown-item"
onClick={()    =>    {    setOrderSetting(SortType.CreationTime);
setOrderButtonText("Новинки") }}>Новинки</a></li>
                    </ul>
                </div>
            </div>


            <div className='row flex-column-reverse flex-sm-row'>
                <Collapse  className='col-12  col-sm-2  pe-0  ps-0
order-2 order-sm-1' isOpen={isOpen}>
                    <div className='border p-0 mb-auto'>
                        <div className='mb-0 pt-2 h6'>
                            <div className="p-2 mt-2 mb-3">
                                <h4>ціна</h4>
                                <RangeSlider
                                    value={priceRange}
                                    range={priceMaxRange}
                                    onAfterChange={(e)          =>
setPriceRange(e)}
                                />
                            </div>
                            {renderCheckboxes()}
                        </div>
                    </div>
                </Collapse>
                <div className='col pe-0 ps-0 order-1 order-sm-2'>
                    {getProductsView()}
                </div>
            </div>

            <div  className='d-flex  justify-content-center  m-2  mt-
5'>
                {getPageSwitcher()}
            </div>
        </>
    );
}
```