МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри інженерії
програмного забезпечення

_____ Євген ДАВИДЕНКО

«___» _____ 2026 р.

КВАЛІФІКАЦІЙНА РОБОТА
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

ВЕБЗАСТОСУНОК ПРОДАЖУ ФУТБОЛЬНОЇ ЕКІПІРОВКИ

Спеціальність 121 Інженерія програмного забезпечення
Освітня програма «Інженерія програмного забезпечення»

Здобувач _____ **Михайло ВАГІН**

«__» _____ 20__ р.

Керівник роботи
д-рка технічних наук,
професорка

_____ **Альона ШВЕД**

«__» _____ 20__ р.

Миколаїв – 2026

Завдання на виконання кваліфікаційної роботи

Чорноморський національний університет імені Петра Могили

Факультет	Комп'ютерних наук
Кафедра	Інженерії програмного забезпечення
Рівень вищої освіти	Перший (бакалаврський)
Освітній ступінь	Бакалавр
Спеціальність	121 Інженерія програмного забезпечення
Освітня програма	Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри інженерії
програмного забезпечення
_____ Євген ДАВИДЕНКО
«____» _____ 2026 р.

ЗАВДАННЯ

на кваліфікаційну бакалаврську роботу здобувача

Вагіна Михайла

1. Тема кваліфікаційної роботи «Вебзастосунок продажу футбольної екіпіровки» затверджена наказом ректора ЧНУ ім. Петра Могили № _349_ від «26» грудня 2025 р.

2. Строк представлення кваліфікаційної роботи «___» _____ 2026 р.

3. Очікуваний результат роботи та початкові дані якщо такі потрібні.

Очікуваним результатом є розроблений та функціонуючий вебзастосунок інтернет-магазину футбольної екіпіровки, що забезпечує: зручний перегляд каталогу товарів; пошук і фільтрацію футбольних товарів; порівняння товарів за основними характеристиками; додавання товарів до кошика; оформлення замовлення зареєстрованими користувачами; реєстрацію та авторизацію користувачів; ведення історії замовлень; надання адміністратору інструментів для

керування товарами, категоріями, замовленнями та користувачами системи.

4. Перелік питань, що підлягають розробці:

– дослідити предметну область електронної комерції футбольної екіпіровки та провести аналіз існуючих аналогів вебзастосунків;

– провести порівняльний аналіз сучасних інтернет-магазинів футбольних товарів для визначення їх функціональних та технічних особливостей;

– сформувати специфікацію функціональних і нефункціональних вимог до програмного забезпечення;

– спроєктувати архітектуру вебзастосунку та розробити структуру бази даних для зберігання інформації про товари, користувачів, замовлення та кошики;

– розробити серверну частину вебзастосунку з реалізацією бізнес-логіки управління каталогом товарів, кошиком та обробкою замовлень;

– розробити клієнтську частину вебзастосунку з адаптивним інтерфейсом користувача;

– реалізувати механізми автентифікації та авторизації користувачів;

– провести тестування клієнтської та серверної частин системи.

5. Перелік графічних матеріалів : Презентація

6. Консультанти:

Консультантів	**Кафедра (організація)**	**Частина роботи**

Дата видачі завдання «____» _____ 2026 р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема: Вебзастосунок продажу футбольної екіпіровки

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КБР	26.12.2025	18.02.2026	виконано
2.	Огляд літератури за темою роботи	01.01.2026	05.03.2026	виконано
3.	Складання календарного плану КБР	16.02.2026	18.02.2026	виконано
4.	Аналіз предметної області	20.02.2026	10.03.2026	виконано
5.	Розробка проєктних рішень	05.03.2026	20.03.2026	виконано
6.	Моделювання та конструювання ПЗ	15.02.2026	30.02.2026	виконано
7.	Кодування, тестування та апробація розробленого ПЗ, аналіз результатів тестування, розробка керівництва користувача	01.03.2026	20.05.2026	виконано
8.	Відгук керівника КБР	22.05.2026	27.05.2026	виконано
9.	Оформлення КБР та презентації	13.05.2026	25.05.2026	виконано
10.	Попередній захист	26.05.2026	26.05.2026	виконано
11.	Завершення оформлення КБР та презентації	02.06.2026	08.06.2026	виконано
12.	Рецензування	08.06.2026	11.06.2026	виконано
13.	Захист кваліфікаційної роботи			

Здобувач _____ **Михайло ВАГІН**

«__» _____ 20__ р.

Керівник роботи

д-рка технічних наук,

професорка _____ **Альона ШВЕД**

«__» _____ 20__ р.

АНОТАЦІЯ

до кваліфікаційної бакалаврської роботи

Вебзастосунок продажу футбольної екіпіровки

Здобувач 408 гр.: Вагін Михайло

Керівник: д-рка техн. наук, професорка Швед Альона

Стрімкий розвиток електронної комерції та зростання популярності онлайн-покупок спортивних товарів зумовлюють необхідність створення зручних, функціональних та сучасних вебзастосунків для продажу футбольної екіпіровки. Аналіз існуючих рішень (ROZETKA, INTERSPORT, MEGASPORT, 4Football, Sportmaster) виявив низку недоліків: перевантаженість інтерфейсу, складність пошуку товарів за розміром і брендом, недостатню адаптивність мобільних версій та обмежені можливості управління замовленнями. Це обумовлює актуальність розробки спеціалізованого вебзастосунку, який забезпечить зручний процес вибору, фільтрації та придбання футбольної екіпіровки.

Об’єктом є процеси електронної комерції в інформаційних системах онлайн-торгівлі, зокрема процеси взаємодії користувачів з вебдодатком, обробки замовлень та управління даними.

Предметом кваліфікаційної роботи є засоби та програмні технології проєктування і реалізації інтернет-магазину футбольної екіпіровки.

Метою кваліфікаційної роботи є розробка інтернет-магазину футбольної екіпіровки, яки забезпечує зручний процес вибору товарів, оформлення замовлень, управління користувачами та інтеграцію з платіжними сервісами з дотриманням вимог інформаційної безпеки.

Кваліфікаційна робота складається із вступу, 4 розділів, висновків та переліку джерел посилання.

У вступі обґрунтовано актуальність розробки вебзастосунку інтернет-магазину футбольної екіпіровки, визначено мету, завдання, об'єкт та предмет дослідження.

У першому розділі проведено аналіз предметної області онлайн-продажу футбольної екіпіровки та існуючих аналогів інтернет-магазинів спортивних

товарів, сформовано специфікацію вимог до програмного забезпечення на основі виявлених переваг та недоліків.

Другий розділ присвячено моделюванню структури вебзастосунку, визначенню функціональних і нефункціональних вимог, а також опису ролей користувачів системи (клієнт, адміністратор, гість).

У третьому розділі описано архітектуру вебзастосунку, наведено UML-діаграми (діаграма класів, діаграма розгортання, діаграми варіантів використання), а також описано структуру бази даних FootballShopDB на основі SQL Server 2022 та Entity Framework Core.

У четвертому розділі описано процес програмної реалізації серверної частини на ASP.NET Core та клієнтської частини з використанням HTML, CSS, Bootstrap і JavaScript, а також представлено результати тестування функціональності системи.

У висновках узагальнено результати виконаної роботи, підтверджено відповідність розробленого вебзастосунку поставленим вимогам та визначено напрями подальшого розвитку системи.

Кваліфікаційна робота викладена на 80 сторінках машинописного тексту, складається із вступу, 4 розділів, загальних висновків, переліку джерел посилання з 21 найменувань та 1 додатку. Праця містить 4 таблиці та 27 рисунків.

Ключові слова: вебзастосунок, інтернет-магазин, футбольна екіпіровка, електронна комерція, ASP.NET Core, SQL Server, Entity Framework Core, каталог товарів, кошик, замовлення, спортивні товари.

ABSTRACT

to the qualifying bachelor's thesis

Web application for selling football equipment

Student of 408 group: Vagin Mykhailo

Supervisor: Doctor of Technical Sciences, Professor Shved Alena

The rapid development of e-commerce and the growing popularity of online purchases of sports goods necessitate the creation of convenient and functional web applications for selling football equipment. An analysis of existing solutions (ROZETKA, INTERSPORT, MEGASPORT, 4Football, Sportmaster) revealed several shortcomings: overloaded interfaces, difficulties in searching products by size and brand, insufficient mobile optimization and limited order management functionality. This determines the relevance of developing a specialized web application that ensures a convenient process of selecting, filtering and purchasing football equipment.

The object is the processes of e-commerce in online trading information systems, in particular the processes of user interaction with the web application, order processing and data management.

The subject of the qualification work is the tools and software technologies for designing and implementing an online store of football equipment.

The purpose of the qualification work is to develop an online store of football equipment that provides a convenient process of selecting goods, placing orders, managing users and integrating with payment services in compliance with information security requirements.

The qualification work consists of an introduction, 4 chapters, conclusions and a list of references.

The introduction substantiates the relevance of developing a web application for an online store of football equipment and defines the purpose, objectives, object and subject of research.

The first chapter analyzes the subject area of online sales of football equipment and existing analogues of sports online stores and forms the software requirements specification.

The second chapter is devoted to modeling the structure of the web application, defining functional and non-functional requirements, and describing system user roles (client, administrator, guest).

The third chapter describes the architecture of the web application, provides UML diagrams (class diagram, deployment diagram, use case diagrams), and describes the structure of the FootballShopDB database based on SQL Server 2022 and Entity Framework Core.

The fourth chapter describes the implementation of the server-side using ASP.NET Core and the client-side using HTML, CSS, Bootstrap and JavaScript, and presents the results of system functionality testing.

The conclusions summarize the results of the completed work, confirm the compliance of the developed web application with the set requirements and outline possible directions for further development of the system.

The qualification work is presented on 80 pages of typewritten text, consists of an introduction, 4 chapters, general conclusions, a list of references with 21 titles and 1 appendix. The work contains 4 tables and 27 figures.

Keywords: web application, online store, football equipment, e-commerce, ASP.NET Core, SQL Server, Entity Framework Core, product catalog, shopping cart, orders, sports goods.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

БД	база даних
ПЗ	програмне забезпечення
СКБД	система керування базами даних
CORS	Cross-Origin Resource Sharing
CRUD	Create, Read, Update, Delete
CSS	Cascading Style Sheets
EF Core	Entity Framework Core
HTML	HyperText Markup Language
HTTPS	HyperText Transfer Protocol Secure
JSON	JavaScript Object Notation
JWT	JSON Web Token
REST	Representational State Transfer
SPA	Single Page Application
SQL	Structured Query Language
SSMS	SQL Server Management Studio
UI	User Interface
URL	Uniform Resource Locator

ВСТУП

Стрімкий розвиток цифрових технологій та зростання популярності онлайн-покупок суттєво трансформували традиційні моделі роздрібної торгівлі. Електронна комерція стала важливою складовою сучасного споживчого ринку, забезпечуючи можливість придбання товарів у будь-який час і з будь-якого місця. Ринок спортивних товарів, зокрема футбольної екіпіровки, посідає значне місце серед онлайн-продажів, що зумовлено високим попитом на спеціалізовану продукцію, різноманітністю брендів і моделей, а також потребою у детальному ознайомленні з характеристиками, розмірами та матеріалами перед покупкою.

Сучасні покупці футбольної екіпіровки потребують зручних інструментів для пошуку товарів за розміром, брендом і призначенням, можливості порівняння моделей, перегляду відгуків, перевірки наявності товарів та швидкого оформлення замовлення. Водночас власники спортивних магазинів зацікавлені в автоматизації процесів управління асортиментом, контролю складських залишків, обробки замовлень та взаємодії з клієнтами для підвищення ефективності діяльності.

Аналіз існуючих інтернет-магазинів спортивних товарів (ROZETKA, INTERSPORT, MEGASPORT, 4Football, Sportmaster) виявив низку недоліків: перевантаженість інтерфейсу банерною рекламою, складність навігації через велику кількість підкатегорій, недостатньо зручну систему підбору розмірів.

Актуальність

На сучасному етапі розвитку інформаційних технологій електронна комерція є одним із найбільш динамічних напрямів цифрової економіки. Зростання кількості користувачів мережі Інтернет, розвиток онлайн-платежів та логістичних сервісів зумовлюють підвищений попит на зручні, безпечні та масштабовані вебсистеми електронної торгівлі.

Особливу актуальність має розробка спеціалізованих інтернет-магазинів, орієнтованих на конкретну предметну область, зокрема на продаж футбольної екіпіровки. Такі системи повинні забезпечувати ефективне управління каталогом

товарів, обробку замовлень, інтеграцію з платіжними сервісами, підтримку ролей користувачів та захист персональних даних.

Практичне значення роботи полягає у можливості застосування результатів дослідження та реалізації системи в реальних умовах малого та середнього бізнесу, а також використання розробленого програмного продукту як основи для подальшого розширення функціональності.

Об’єктом є процеси електронної комерції в інформаційних системах онлайн-торгівлі, зокрема процеси взаємодії користувачів з вебдодатком, обробки замовлень та управління даними.

Предметом кваліфікаційної роботи є засоби та програмні технології проєктування і реалізації інтернет-магазину футбольної екіпіровки.

Метою кваліфікаційної роботи є розробка інтернет-магазину футбольної екіпіровки, яки забезпечує зручний процес вибору товарів, оформлення замовлень, управління користувачами та інтеграцію з платіжними сервісами з дотриманням вимог інформаційної безпеки.

Для досягнення поставленої мети у кваліфікаційній роботі необхідно вирішити такі завдання:

1. провести аналіз предметної області електронної комерції у сфері спортивної екіпіровки;
2. дослідити існуючі програмні рішення та аналоги інтернет-магазинів;
3. сформувати функціональні та нефункціональні вимоги до системи;
4. спроєктувати архітектуру веб застосунку;
5. розробити структуру бази даних;
6. розробити серверну частину вебзастосунку з реалізацією бізнес-логіки управління каталогом товарів, кошиком та обробкою замовлень;
7. розробити клієнтську частину вебзастосунку з адаптивним інтерфейсом користувача;
8. провести тестування клієнтської та серверної частин системи, перевірити її коректність, продуктивність та стабільність роботи.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ЕЛЕКТРОННОЇ КОМЕРЦІЇ

1.1 Огляд предметної області та постановка завдання

Електронна комерція – це форма торгівлі, де покупець і продавець взаємодіють через Інтернет, минаючи фізичний магазин. Динаміка вражає: за даними Statista, світові продажі через інтернет-магазини у 2023 році перевалили за 5,8 трильйона доларів США, а до 2027 року аналітики прогнозують зростання до 8 трильйонів [11]. Ринок росте, бо смартфони стали доступними практично всім, цифрова грамотність населення стабільно підвищується, а онлайн-шопінг банально зручніший за похід у магазин. Покупець отримує цілодобовий доступ до товарів, не прив'язаний до географії, і за лічені хвилини порівнює ціни десятків продавців.

В Україні, попри важку економічну ситуацію через повномасштабне вторгнення, e-commerce не просто тримається – він росте. За даними EVO Company, обсяг ринку електронної комерції у 2023 році збільшився на 28% відносно попереднього року [12]. Пояснень кілька. Мільйони людей переїхали до інших міст і регіонів – і дистанційні покупки стали для них нормою. Частина фізичних магазинів закрилася. Споживча поведінка загалом стрімко цифровізується. Лідерами онлайн-продажів залишаються електроніка, одяг та взуття, товари для дому і спорту.

Спортивні товари – фактично один із найбільш стабільних сегментів онлайн-торгівлі. За аналітикою Grand View Research, світовий ринок онлайн-продажу спортивного одягу й екіпіровки у 2024 році оцінюється у 167 мільярдів доларів, а середньорічний темп зростання до 2030 року складе близько 9,7% [18]. Ринок розширюється, бо суспільний інтерес до здорового способу життя та активного відпочинку не вщухає, а провідні спортивні бренди активно нарощують присутність у цифровому просторі.

Футбол – найпопулярніший вид спорту на планеті. Не за назвою, а за цифрами: FIFA фіксує понад 265 мільйонів гравців у 200 країнах. Не дивно, що ринок футбольної екіпіровки – один із найбільших у всій спортивній індустрії.

Асортимент широкий: бутси, м'ячі, клубна та збірна форма, захисне спорядження, тренувальний інвентар, аксесуари. Онлайн-торгівля тут має очевидну перевагу – ширший вибір, нижча ціна, миттєвий пошук за брендом, розміром або клубом.

Саме тому розробка спеціалізованого вебзастосунку FootballShop – не абстрактне навчальне завдання, а практично значуще рішення. Такий застосунок має давати користувачу зручний каталог із гнучкою фільтрацією та пошуком, надійно обробляти замовлення і надавати адміністратору повноцінний інструментарій для керування асортиментом. Проєкт поєднує сучасні підходи до веброзробки з реальними потребами швидкозростаючого ринку спортивних товарів.

1.2 Аналіз існуючих рішень

Щоб сформулювати обґрунтовані функціональні вимоги до розроблюваного застосунку, було детально вивчено п'ять інтернет-магазинів, активних на українському та міжнародному ринках: ROZETKA, INTERSPORT, MEGASPORT, 4Football і Athletics. Вибір не випадковий – кожна з цих платформ займає помітну частку ринку онлайн-продажу спортивних товарів в Україні, пропонує футбольну екіпіровку та реалізує власний підхід до організації інтерфейсу й функціоналу. Платформи оцінювались за єдиними критеріями: швидкість завантаження сторінок, зручність навігації, можливості системи фільтрації, якість мобільної адаптації та повнота адміністративного інструментарію.

ROZETKA (rozetka.com.ua)

ROZETKA – найбільший маркетплейс України із широким охопленням товарних категорій, у тому числі спортивної екіпіровки.

Сильні сторони платформи:

- розгалужена система фільтрації за численними параметрами;
- зручний модуль порівняння товарів;
- підтримка різних платіжних систем;

- власний мобільний застосунок;
- система користувацьких відгуків і рейтингів.

Що функціонує невдало:

- дизайн занадто захаращений рекламними банерами;
- навігація ускладнена через надмірно розгалужену категоризацію;
- сторінки вантажаться повільно через надлишок графіки;
- інструменти підбору розміру спортивного одягу відсутні;
- адаптивність під смартфони потребує серйозного доопрацювання.

INTERSPORT (intersport.ua)

INTERSPORT – міжнародна мережа спортивних магазинів, представлена і в Україні.

Позитивні аспекти:

- вузька спеціалізація виключно на спортивних товарах;
- фахова консультація щодо підбору екіпіровки;
- широкий представлений асортимент брендів;
- діюча програма лояльності для покупців.

Головні недоробки:

- інтерфейс морально застарів і виглядає нерелевантно;
- можливості фільтрації товарів суттєво обмежені;
- порівняння характеристик між позиціями реалізоване незручно;
- фільтри гальмують при великому каталозі;
- мобільна версія працює нестабільно та повільно.

MEGASPORT (megasport.ua)

MEGASPORT – велика українська мережа спортивних магазинів із розвиненою онлайн-платформою.

Плюси платформи:

- один із найбагатших асортиментів футбольної екіпіровки;
- регулярні акції та гнучка система знижок;
- процес оформлення замовлення реалізований зрозуміло;
- картки товарів містять вичерпну інформацію.

Дефекти юзабіліті:

- головна сторінка перевантажена блоками та візуальними елементами;
- пошукова система реагує із помітною затримкою;
- процедура реєстрації надмірно ускладнена;
- персоналізація практично відсутня;
- кошик поводиться нестабільно при слабкому інтернет-з'єднанні.

4Football (4football.com.ua)

4Football – спеціалізований магазин, зосереджений виключно на футбольних товарах та аксесуарах.

Переваги вузької спеціалізації:

- чіткий фокус на футбольній тематиці без сторонніх категорій;
- детальні технічні описи кожної позиції;
- розмірні сітки для різних брендів представлені наочно;
- налагоджена швидка доставка.

Зворотний бік – мінуси:

- адміністративна панель має вкрай обмежений функціонал;
- жодних механізмів рекомендацій чи супутніх товарів не передбачено;
- зображення товарів не оптимізовані – завантажуються повільно;
- пошук примітивний: автодоповнення відсутнє;
- незареєстровані користувачі не можуть зберігати обрані товари.

Athletics (athletics.ua)

Sportmaster – міжнародна мережа спортивних магазинів із широкою лінійкою товарів різних цінових сегментів.

Що вдалося реалізувати добре:

- різноманітний асортимент у широкому ціновому діапазоні;
- категоризація товарів побудована логічно та зрозуміло;
- картки товарів містять детальну інформацію;
- бонусна програма інтегрована безпосередньо в кабінет покупця.

Проблемні зони:

- візуальна перенасиченість сторінок відволікає від товарів;
- структура меню заплутана і вимагає зайвих кліків;
- на смартфонах платформа відчутно гальмує;
- для вузькоспеціалізованих товарів бракує тематичних фільтрів;
- постійні покупці не мають можливості прискореного оформлення замовлення.

Порівняльний аналіз

На основі проведеного дослідження складено порівняльну таблицю ключових характеристик розглянутих платформ (табл. 1.1).

Таблиця 1.1 – Порівняльний аналіз існуючих рішень

Параметр	**ROZETKA**	**INTERSPORT**	**MEGASPORT**	**4Football**	**Sportmaster**
Розробник (дистриб'ютор)	ROZETKA (Україна)	INTERSPORT International Corp.	MEGASPORT (Україна)	4Football (Україна)	Sportmaster Group

Архітектура	Триланкова модель (веб + API + БД); мікросервісний підхід; мобільні клієнти	Триланкова модель; e-commerce платформа; з'єднання з CRM	Триланкова модель; веб і мобільні клієнти; зв'язок з офлайн-мережею	Триланкова модель; монолітний каркас системи; вебмагазин	Триланкова модель; клієнт-серверна схема; зв'язок з ERP

Кінець таблиці 1.1

Мова реалізації	Frontend: HTML5, CSS3, JS; Mobile: Kotlin/Swift	Frontend: HTML5, CSS3, JS;	Frontend: HTML5, CSS3, JS;	Frontend: HTML5, CSS3, JS;	Frontend: HTML5, CSS3, JS;
Ключові функції	Масштабна база товарних позицій, розширені фільтри, відгуки, маркетплейс, особистий кабінет, логістика	Спортивна спрямованість, брендовий асортимент, програма лояльності, фільтрація	Акційна система, широкий товарний вибір, інформативні картки	Футбольна спеціалізація, розмірні таблиці, докладні характеристики	Великий каталог, бонусна система, структурована категоризація
Переваги	Потужна фільтрація, мобільний застосунок, швидкість	Спортивна експертиза, якісний асортимент брендів	Зрозуміле оформлення замовлення, акційні пропозиції	Чітка тематична спеціалізація, вичерпні описи товарів	Різноманітність асортименту, бонусна програма
Недоліки	Екран надмірно насичений блоками	Дизайн застарів; мобільна версія потребує доопрацювання	Пошук гальмує; реєстрація надто складна	Обмежений функціонал; пошук без автодоповнення	Заплутана навігація; мобільна версія працює повільно

Узагальнення результатів аналізу

Вивчення п'яти платформ дало змогу виявити типові слабкі місця, що повторюються на більшості майданчиків:

1. візуальний шум та надлишок реклами – практично кожен із розглянутих сайтів страждає від надмірної кількості банерів, спливаючих вікон і рекламних блоків – усе це відволікає від товарів і погіршує загальне враження від роботи з платформою;

2. заплутана навігація – глибока багаторівнева структура меню разом із надмірно дробною категоризацією змушує користувача витрачати зайвий час на пошук потрібної позиції;

3. низька швидкість завантаження – неоптимізовані зображення у високому розрізненні та підключення стороннього JS-коду суттєво уповільнюють рендеринг сторінок – особливо на слабких мережах;

4. слабка адаптація під мобільні пристрої – жодна з платформ не демонструє бездоганної роботи на смартфонах: елементи перекриваються, шрифти не масштабуються, взаємодія з фільтрами на сенсорному екрані незручна;

5. недостатні можливості фільтрації – у жодного з магазинів немає спеціалізованих інструментів для швидкого підбору футбольних товарів за комбінацією параметрів – клуб, розмір, бренд, тип покриття;

6. обтяжливий процес оформлення замовлення – багатокрокові форми з надміром обов'язкових полів відлякують покупця на фінальному етапі, підвищуючи відсоток незавершених покупок.

1.3 Аналіз засобів розробки

Технологічний стек – це не просто набір інструментів, а архітектурне рішення, що визначає продуктивність, підтримуваність і масштабованість усієї системи. Для FootballShop стек підбирався свідомо: кожен компонент має конкретне обґрунтування, а не потрапив до проекту за звичкою чи випадково.

Серверна частина

Серверна частина FootballShop обробляє HTTP-запити, виконує бізнес-логіку, взаємодіє з базою даних і формує відповіді клієнту. Фундаментом слугує платформа ASP.NET Core 8.0 з мовою C# 12.

Мова програмування C# 12

C# – вибір не випадковий. Мова від Microsoft дозволяє працювати одразу в кількох парадигмах: об'єктно-орієнтованій, функціональній та компонентній, не примушуючи обирати щось одне. Строга типізація відловлює помилки ще на етапі компіляції, а не в рантаймі. Збирач сміття (Garbage Collector) бере на себе управління пам'яттю – розробник концентрується на логіці, а не на ручному

звільненні ресурсів. Конструкції async/await роблять асинхронний код читабельним, а LINQ перетворює роботу з колекціями і запитами до БД на виразні, компактні вирази. Станом на 2024 рік C# стабільно тримається у топ-10 індексу TIOBE – і не без підстав. У FootballShop мовою C# 12 написано контролери, сервіси, репозиторії та моделі даних.

Фреймворк ASP.NET Core 8.0

ASP.NET Core – кросплатформений відкритий фреймворк Microsoft для побудови вебзастосунків і API. Восьма версія, що вийшла у листопаді 2023 року, має статус LTS із підтримкою до листопада 2026 року [5]. Це принципово: проект не залишиться без оновлень безпеки ні під час розробки, ні після розгортання.

За результатами TechEmpower Framework Benchmarks, ASP.NET Core стабільно потрапляє до топ-10 найшвидших вебфреймворків у категоріях обробки JSON-запитів і роботи з базами даних – і це при порівнянні з Node.js/Express, Python/Django, Python/Flask та Java/Spring Boot. Фреймворк реалізує патерн MVC: модель зберігає дані й логіку, представлення відповідає за відображення, контролер координує їхню взаємодію. Вбудований механізм Dependency Injection дає змогу перевернути залежності між компонентами – код стає слабко зв'язаним, а тестування перестає бути болем.

У FootballShop через ASP.NET Core розгорнуто MVC-контролери для серверного рендерингу, REST API контролери для AJAX-запитів, middleware автентифікації та авторизації, а також централізовану обробку помилок.

Entity Framework Core 8.0

Писати вручну SQL для кожної CRUD-операції – марна витрата часу, коли є EF Core. Цей ORM для платформи .NET дозволяє звертатись до реляційної бази через звичні C#-об'єкти та LINQ-запити. Підхід Code-First тут особливо зручний: структура бази даних описується через класи-сутності у коді, а система сама генерує відповідні таблиці за допомогою механізму міграцій. Вносиш зміни в модель – запускаєш міграцію – схема оновлюється автоматично. У FootballShop через EF Core 8.0 визначено моделі (User, Product, Category, Order, OrderItem, Cart,

CartItem), реалізовано CRUD через репозиторії, налаштовано зв'язки між сутностями (один-до-багатьох, багато-до-багатьох) і забезпечено автоматичне керування схемою бази даних [8].

AutoMapper

Щоб не писати монотонний код копіювання властивостей між Entity-об'єктами та DTO, у проекті задіяно бібліотеку AutoMapper. У багаторівневій архітектурі дані між рівнями передаються не напряму через сутності, а через спеціальні об'єкти – DTO (Data Transfer Object). AutoMapper конвертує їх в обидві сторони автоматично, звільняючи від рутини.

Автентифікація та авторизація (JWT + Cookie)

Безпека FootballShop побудована на двох механізмах. JWT (JSON Web Token, стандарт RFC 7519) захищає запити до REST API: токен містить claims користувача – ідентифікатор, ім'я і роль – і підписаний цифровим підписом. Cookie-автентифікація підтримує сесії при роботі з MVC-сторінками. Разом вони забезпечують рольову модель доступу (RBAC) з двома ролями: user та admin. Паролі у базі даних зберігаються виключно у хешованому вигляді – алгоритм BCrypt унеможливлює їх відновлення навіть при прямому доступі до БД.

Клієнтська частина

HTML5 та Razor Pages

За відображення даних на стороні клієнта відповідає HTML5 у зв'язці з Razor Pages – технологією шаблонів ASP.NET Core, що дозволяє вбудовувати C#-код безпосередньо в розмітку. Razor формує HTML динамічно на сервері на основі моделі, позбавляючи від дублювання логіки відображення. У FootballShop через Razor генеруються сторінки каталогу, деталей товару, кошика, профілю, адміністративної панелі та інших розділів.

CSS3 та Bootstrap 5

CSS3 відповідає за зовнішній вигляд: анімації, медіа-запити, змінні (Custom Properties), Flexbox і Grid для гнучкого компонування. Bootstrap 5 – CSS-фреймворк

від Twitter – доповнює його готовою бібліотекою компонентів, утилітами стилізування та адаптивною сіткою на 12 колонок. Завдяки Bootstrap 5 вдалося різко скоротити час на верстку, отримати кросбраузерну сумісність і коректний вигляд на будь-якому екрані – від смартфону шириною 320px до широкого монітора 1920px+. На цій основі у FootballShop реалізовано адаптивну навігацію, картки товарів, форми реєстрації й оформлення замовлення, модальні вікна, таблиці адмінпанелі та систему сповіщень Toast. Іконки – із бібліотеки Bootstrap Icons, що налічує понад 1800 векторних SVG-зображень.

JavaScript та jQuery

JavaScript забезпечує інтерактивність на стороні браузера: валідацію форм, обробку подій, динамічне оновлення вмісту без повного перезавантаження сторінки та взаємодію з серверним REST API. jQuery спрощує маніпуляції з DOM, обробку подій і написання AJAX-запитів. Попри те, що у нових проектах дедалі частіше обирають SPA-фреймворки на кшталт React, Vue.js чи Angular, для застосунку з серверним рендерингом на Razor Pages jQuery – прагматичний і виправданий вибір: він дає потрібну часткову динамічність, не змушуючи переходити на повноцінну SPA-архітектуру. AJAX-запити дозволяють спілкуватися з API без перезавантаження сторінки [10].

Засоби управління базою даних

SQLite та SQLiteStudio

Для зберігання даних FootballShop обрано SQLite – вбудовану реляційну СУБД із відкритим вихідним кодом. Головна її особливість: жодного окремого серверного процесу не потрібно – вся база живе в одному файлі на диску. SQLite підтримує стандарт SQL, гарантує ACID-властивості, а також підтримує зовнішні ключі, індекси, представлення і тригери. Порівняно з MySQL, PostgreSQL чи Microsoft SQL Server вона виграє для проектів такого масштабу за трьома параметрами: нульова конфігурація, мінімальне споживання ресурсів і відсутність необхідності разгортати окремий сервер БД.

Інтеграція з Entity Framework Core здійснюється через пакет Microsoft.EntityFrameworkCore.Sqlite – повна сумісність із міграціями, LINQ-запитами і відстеженням змін. У FootballShop SQLite зберігає дев'ять взаємопов'язаних таблиць: Users, Products, Categories, Orders, OrderItems, Carts, CartItems, Payments і DeliveryInfo. Між ними визначено зовнішні ключі, що забезпечують референційну цілісність.

Допоміжні засоби розробки

Visual Studio 2022

Основне середовище розробки проекту – Visual Studio 2022 від Microsoft. IDE надає інтелектуальне автодоповнення IntelliSense, вбудований відладчик із підтримкою точок зупинки, готові шаблони проектів ASP.NET Core і статичні аналізатори для раннього виявлення потенційних проблем у коді.

Висновки до розділу 1

У першому розділі проведено комплексний аналіз предметної області та обґрунтовано ключові рішення щодо розробки вебзастосунку FootballShop.

Дослідження п'яти платформ – ROZETKA, INTERSPORT, MEGASPORT, 4Football і Sportmaster – виявило повторювані слабкі місця: перевантажений інтерфейс, заплутану навігацію, низьку швидкість завантаження, слабку мобільну адаптацію і примітивні механізми фільтрації. Ці спостереження лягли в основу функціональних вимог до FootballShop.

Сформовано вимоги для трьох категорій користувачів – гостя, зареєстрованого покупця й адміністратора – а також нефункціональні вимоги щодо продуктивності, надійності, безпеки і зручності.

2 МОДЕЛЮВАННЯ ТА СПЕЦИФІКАЦІЯ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Функціональна модель програмної системи

Для структурного відображення процесів FootballShop обрано нотацію IDEF0 (Integrated DEFinition language 0). Вона дозволяє розбити систему на ієрархічні функціональні блоки і наочно показати, як між ними циркулює інформація. Кожен блок діаграми пов'язаний із чотирма типами стрілок: вхідні дані (Input) – те, що функція споживає; результати (Output) – те, що вона породжує; керуючі впливи (Control) – правила й умови, що визначають поведінку функції; механізми (Mechanism) – ресурси, завдяки яким функція взагалі виконується. Побудову моделі розпочато з контекстної діаграми A-0, що зображує систему як єдиний блок із усіма зовнішніми зв'язками, після чого виконується послідовна декомпозиція на нижчі рівні.

Контекстна діаграма A-0 (рис. 2.1) зводить FootballShop до одного процесу – «Забезпечення онлайн-продажу футбольної екіпіровки». На вхід надходять запити користувачів, дані товарів і дані замовлень. На виході – сформовані замовлення, підтвердження оплати та звіти для адміністратора. Механізмами виступають вебсервер, база даних SQLite і браузер користувача [21].

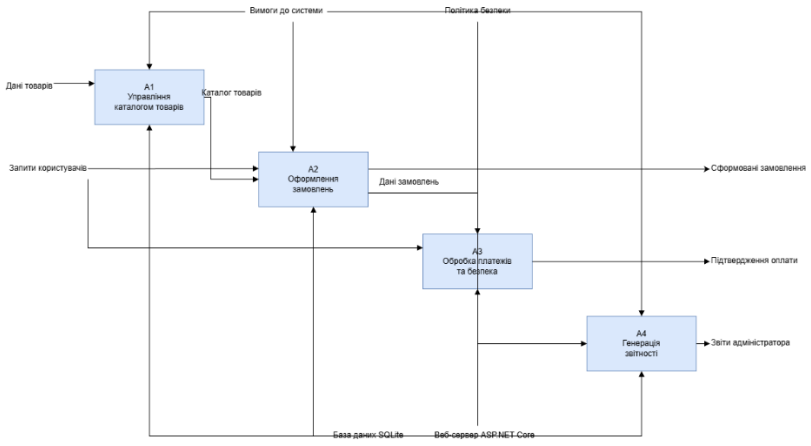

Рисунок 2.1 – Контекстна діаграма IDEF0 (рівень A-0)

Діаграма декомпозиції першого рівня A0 (рис. 2.2) розкриває внутрішню будову системи через чотири функціональні блоки:

- A1 «Управління користувачами» – охоплює реєстрацію, автентифікацію через JWT-токен, авторизацію за ролями (user/admin) і редагування профілю. Вхід – облікові дані; вихід – автентифікований сеанс роботи;
- A2 «Управління каталогом товарів» – забезпечує перегляд, пошук і фільтрацію для покупців, а також додавання, редагування і видалення позицій для адміністратора. Вхід – запити на перегляд або редагування; вихід – відфільтрований список або оновлений каталог;
- A3 «Обробка замовлень» – реалізує повний цикл від кошика до оплати: формування переліку товарів, введення адреси доставки, вибір служби, підтвердження та проведення оплати. Вхід – вміст кошика і дані доставки; вихід – замовлення зі статусом «Оплачено»;
- A4 «Адміністрування системи» – керування статусами замовлень, облік користувачів, перегляд статистики. Вхід – дані замовлень і користувачів; вихід – оновлені статуси та аналітичні звіти.

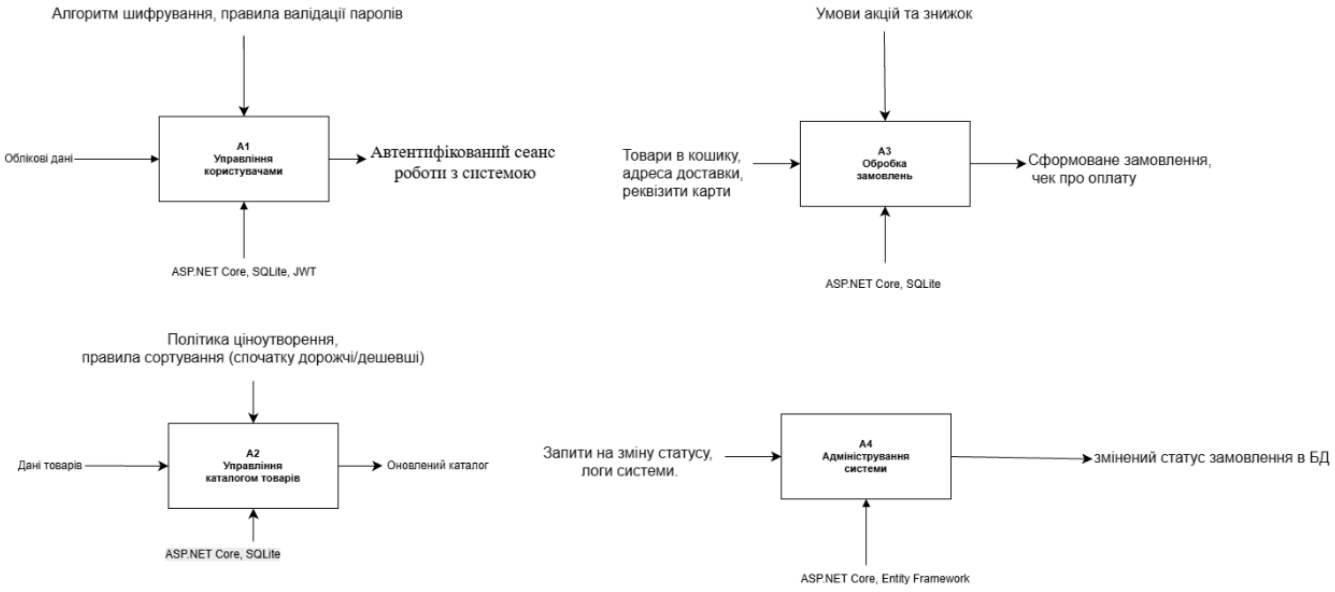

Рисунок 2.2 – Діаграма декомпозиції IDEF0 (рівень A0)

Діаграма другого рівня A3 (рис. 2.3) деталізує обробку замовлень у чотирьох підблоках:

- A31 «Формування кошика» – додавання і коригування товарів у кошику;
- A32 «Введення даних доставки» – заповнення форми з ім'ям, телефоном, адресою і вибором служби доставки;
- A33 «Підтвердження замовлення» – запис замовлення до бази даних зі статусом «Очікує оплати»;
- A34 «Обробка оплати» – введення реквізитів картки і переведення статусу в «Оплачено».

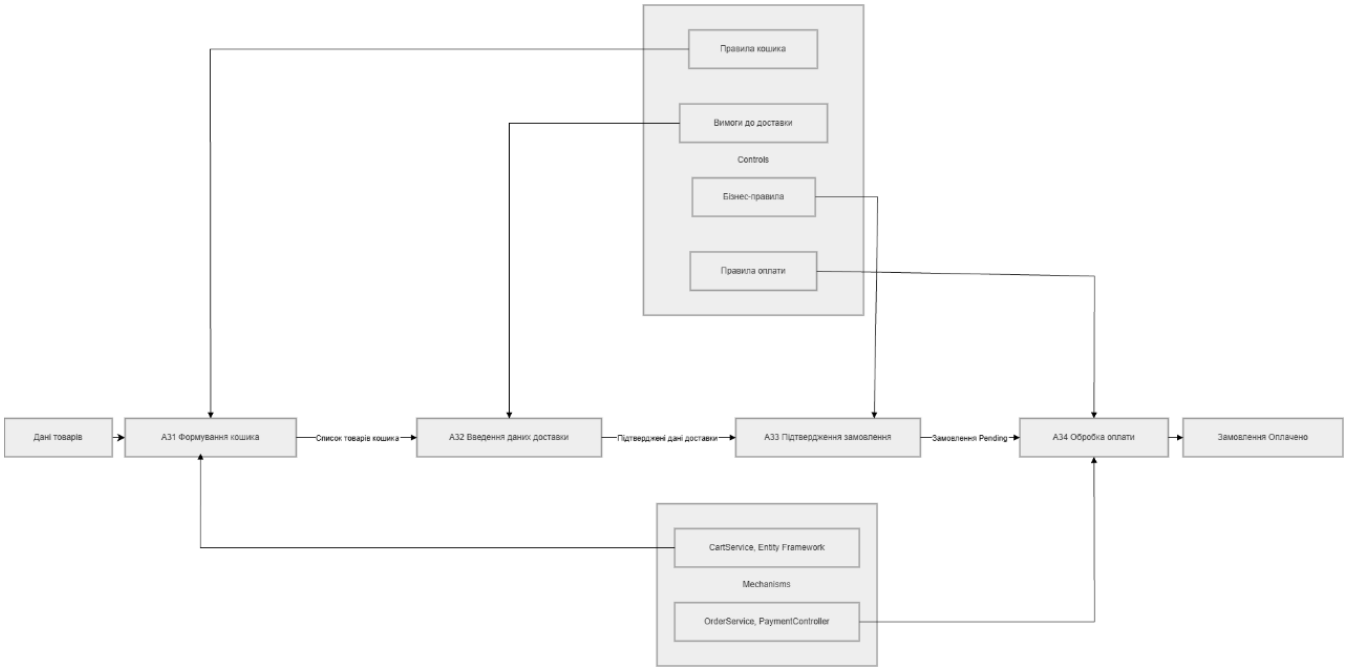

Рисунок 2.3 – Діаграма декомпозиції IDEF0 процесу обробки замовлень (рівень A3)

Побудована модель IDEF0 дає повне уявлення про склад функцій, їхні взаємозв'язки та інформаційні потоки – і саме вона слугує відправною точкою для проєктування архітектури та бази даних.

2.2 Специфікація вимог до програмного забезпечення

Призначення та межі проєкту

FootballShop автоматизує процес продажу футбольної екіпіровки через Інтернет. Система надає покупцям змогу переглядати каталог і оформлювати замовлення, а адміністраторам – керувати асортиментом і опрацьовувати заявки.

Цільова аудиторія:

- покупці футбольної екіпіровки – від аматорів до профі;
- власники або менеджери інтернет-магазину;
- адміністратори системи.

Погодження програмної документації

Розробка ведеться у відповідності до таких стандартів і рекомендацій:

- ISO/IEC 25010:2011 – для визначення характеристик якості ПЗ;
- IEEE 830-1998 – для структурування специфікації вимог;
- OWASP – для забезпечення захисту вебзастосунку;
- REST – для проєктування API [5].

Межі проєкту

До складу системи входить: вебінтерфейс для покупців, RESTful API для взаємодії клієнта з сервером, серверна логіка, база даних і адміністративна панель.

За межами проєкту залишаються: реальна інтеграція з платіжними системами (реалізується імітація) та підключення до логістичних сервісів.

Загальний опис системи

Сфера застосування

FootballShop орієнтований на малі та середні підприємства у сфері продажу спортивних товарів. Платформа може функціонувати як самостійний канал онлайн-продажів або доповнювати офлайн-мережу.

Практичні переваги від впровадження:

- автоматизація прийому та опрацювання замовлень;

- зниження операційних витрат порівняно з традиційною торгівлею;
- вихід за географічні межі місцевого ринку;
- цілодобовий доступ до каталогу без прив'язки до графіку роботи;
- зручний підбір і замовлення товарів для покупця;
- накопичення даних для аналізу продажів.

Характеристики користувачів

Таблиця 2.1 – Характеристики користувачів системи

Тип користувача	**Роль**	**Рівень технічної підготовки**	**Основні завдання**
Гість	Відвідувач	Початковий (навігація в браузері, базові онлайн-покупки)	Перегляд каталогу, пошук товарів, реєстрація
Зареєстрований користувач	Покупець	Початковий (навігація в браузері, базові онлайн-покупки)	Оформлення замовлень, перегляд історії покупок
Адміністратор	Менеджер магазину	Середній (досвід роботи з CMS або ERP-системами)	Модерація товарних позицій, опрацювання замовлень

Загальна структура системи

Система побудована за трирівневою клієнт-серверною архітектурою:

1. рівень представлення (Presentation Layer):
   - вебінтерфейс на HTML/CSS/JavaScript;

- адаптивна верстка під різні пристрої;
- клієнтська валідація форм.

2. Рівень бізнес-логіки (Business Logic Layer):
   - контролери ASP.NET Core MVC [9];
   - REST API контролери;
   - сервіси бізнес-правил;
   - маппери для конвертації між Entity і DTO.
3. Рівень доступу до даних (Data Access Layer):
   - Entity Framework Core ORM;
   - репозиторії для роботи з БД;
   - міграції схеми;
   - SQLite як сховище даних.

Загальні обмеження

Технологічні:

- середовище виконання – .NET 8.0;
- СУБД – SQLite (файлове сховище, без окремого сервера);
- підтримувані браузери – Chrome 90+.

Організаційні:

- розробку веде один розробник;
- комерційні бібліотеки не залучаються;
- терміни визначені академічним роком.

Функції системи

Функція «Перегляд каталогу товарів»

Суть операції: система надає доступ до повного каталогу з підтримкою фільтрації та сортування.

Що надходить на вхід:

- параметри фільтрації (категорія, мінімальна і максимальна ціна);
- параметр сортування (за ціною, назвою або новизною);
- номер сторінки для пагінації.

Результат генерації:

- список товарів із ключовими атрибутами (назва, ціна, зображення);
- загальна кількість знайдених позицій;
- метадані поточної сторінки.

Функціональні вимоги:

- FR-1.1: товари відображаються у вигляді сіткової розкладки карток;
- FR-1.2: кожна картка містить зображення, назву, ціну та кнопку «Детальніше»;
- FR-1.3: сторінка вміщає 9 товарів, решта доступна через пагінацію;
- FR-1.4: фільтри спрацьовують миттєво, без перезавантаження сторінки;
- FR-1.5: якщо за заданими критеріями товарів не знайдено, система виводить відповідне повідомлення.

Функція «Управління кошиком»

Призначення алгоритму: надати покупцю можливість формувати перелік обраних товарів, коригувати кількість і видаляти позиції.

Вхідні параметри:

- ідентифікатор товару;
- потрібна кількість одиниць;
- тип дії (додати, скоригувати кількість, видалити).

Вихідні дані системи:

- актуальний вміст кошика;
- розрахована загальна вартість;

- кількість позицій у кошику.

Функціональні вимоги:

- FR-2.1: для авторизованих користувачів вміст кошика зберігається у базі даних;
- FR-2.2: якщо товар уже є в кошику, система збільшує кількість, а не дублює запис;
- FR-2.3: загальна вартість перераховується автоматично при кожній зміні кількості;
- FR-2.4: допустимий діапазон кількості одного товару – від 1 до 99 одиниць;
- FR-2.5: перед видаленням товару система запитує підтвердження дії.

Функція «Обробка замовлень»

Що виконує цей модуль: адміністратор отримує інструменти для перегляду всіх замовлень і керування їхніми статусами.

Дані, що подаються на вхід:

- ідентифікатор замовлення;
- новий статус, що присвоюється.

Дані на виході:

- оновлена картка замовлення;
- підтвердження успішного збереження статусу.

Функціональні вимоги:

- FR-3.1: адмінпанель відображає повний список замовлень із фільтрацією за статусом;
- FR-3.2: адміністратор змінює статус через випадний список безпосередньо у таблиці;
- FR-3.3: після зміни статусу список замовлень оновлюється автоматично;

- FR-3.4: дозволені переходи між статусами: Pending –>Processing, Processing –>Shipped, Shipped –>Delivered; з будь-якого статусу можливий перехід у Cancelled;
- FR-3.5: момент зміни статусу фіксується у базі даних із позначкою часу.

Вимоги до інформаційного забезпечення

Джерела вхідної інформації

1. дані товарів – вводяться адміністратором через вебінтерфейс.
2. дані користувачів – надходять при реєстрації та редагуванні профілю.
3. дані замовлень – формуються автоматично з вмісту кошика і форми доставки.
4. зображення товарів – підключаються за URL-посиланням.

Нормативно-довідкова інформація

Система спирається на три довідники:

1. довідник категорій – назви та ідентифікатори категорій товарів.
2. довідник статусів замовлень – перелік: Pending, Processing, Shipped, Delivered, Cancelled.
3. довідник ролей – типи облікових записів: User, Admin.

Вимоги до зберігання та ведення інформації

1. усі дані зберігаються в реляційній базі SQLite.
2. цілісність забезпечується зовнішніми ключами та обмеженнями.
3. паролі зберігаються виключно у хешованому вигляді (BCrypt).
4. резервне копіювання бази виконується щоденно.
5. критичні операції (зміна статусу замовлення, видалення товарів) підлягають логуванню.

Вимоги до технічного забезпечення

Серверне обладнання:

1. процесор: мінімум 2 ядра, рекомендовано 4 ядра;
2. оперативна пам'ять: мінімум 4 ГБ, рекомендовано 8 ГБ;

3. вільний дисковий простір: мінімум 20 ГБ.

Клієнтське обладнання:

1. будь-який пристрій із сучасним браузером (ПК, ноутбук, смартфон);
2. мінімальна роздільна здатність екрану: 320×568 пікселів.

Вимоги до програмного забезпечення

Архітектура програмної системи

Система реалізує патерн MVC із чітким поділом відповідальності:

1. Model: Entity-класи доменних об'єктів, DTO для передачі між рівнями, ViewModels для передачі у представлення.
2. View: Razor Pages для серверного рендерингу, HTML/CSS/Bootstrap для структури та стилів, JavaScript/jQuery для клієнтської інтерактивності.
3. Controller: MVC-контролери для HTTP-запитів від представлень, API-контролери для AJAX-запитів, Middleware для автентифікації та обробки помилок [13].

Системне програмне забезпечення

Серверне середовище:

1. ОС: Windows Server 2019+ або Linux (Ubuntu 20.04+);
2. .NET 8.0 Runtime;
3. ASP.NET Core 8.0.

Середовище бази даних:

1. SQLite – легка файлова СУБД без окремого сервера;
2. SQLiteStudio – для адміністрування та перегляду даних;
3. Entity Framework Core 8.0 – ORM для роботи з базою [8];
4. EF Core Migrations – для керування схемою;
5. LINQ – для побудови запитів.

Мова і технології розробки

Backend:

1. мова: C# 12;
2. фреймворк: ASP.NET Core 8.0;

3. ORM: Entity Framework Core 8.0 [14];
4. маппінг: AutoMapper 12.0.

Frontend:

1. розмітка: HTML5;
2. стилізація: CSS3, Bootstrap 5.3 [15];
3. скриптова мова: JavaScript (ES6+);
4. бібліотека: jQuery 3.7.

Вимоги до зовнішніх інтерфейсів

Інтерфейс користувача

Загальні вимоги до UI:

- інтерфейс не вимагає попереднього навчання – логіка взаємодії має бути очевидною;
- єдиний стиль оформлення витримується на всіх сторінках;
- адаптивна верстка коректно відображається на екранах різних розмірів;
- підтримка Chrome, Firefox, Safari і Edge.

Обов'язкові елементи:

- навігаційне меню з посиланнями на основні розділи;
- рядок пошуку для швидкого знаходження товарів;
- панель фільтрів для відбору за параметрами;
- кошик із відображенням поточної суми;
- форми з валідацією введених даних.

Програмний інтерфейс

RESTful API організовано за такими групами endpoints:

1. /api/products – операції з товарами (GET, POST, PUT, DELETE);
2. /api/categories – операції з категоріями (GET, POST, PUT, DELETE);
3. /api/cart – операції з кошиком (GET, POST, PUT, DELETE);
4. /api/orders – операції із замовленнями (GET, POST, PUT);
5. /api/users – операції з користувачами (GET, PUT).

Формат обміну даними: JSON, стандартні HTTP-методи і коди відповіді, Content-Type: application/json.

Протокол: HTTP/HTTPS; для передачі чутливих даних – виключно HTTPS; підтримка cookies для збереження сесії.

Властивості програмного забезпечення

Доступність

Плановий показник аптайму (готовності системи) закладається на рівні не нижче 99%, що відповідає максимальному часу простою близько 7 годин на рік. Доступ – цілодобовий, без технологічних вікон у розкладі.

Супроводжуваність

Модульна структура коду дозволяє вносити зміни локально, не зачіпаючи несуміжні компоненти. Усі публічні методи і класи документуються. Система логування забезпечує діагностику проблем без доступу до продуктового середовища.

Переносимість

Застосунок розгортається як на Windows, так і на Linux без модифікацій. Прив'язки до конкретного обладнання немає; у роботі використовуються стандартні протоколи та формати даних.

Продуктивність

- час відгуку на запит користувача – не більше 2 секунд;
- завантаження головної сторінки – до 3 секунд;
- стабільна робота при одночасному навантаженні до 100 сеансів без деградації.

Надійність

- вміст кошика зберігається автоматично навіть при втраті з'єднання;
- операції з базою даних виконуються транзакційно;
- помилки перехоплюються і відображаються у вигляді зрозумілих повідомлень;

- передбачено автоматичне резервне копіювання даних.

Безпека

- паролі хешуються алгоритмом BCrypt – відновлення відкритого тексту виключене;
- параметризовані запити закривають вектор SQL-ін'єкцій;
- вхідні дані екрануються для захисту від XSS-атак;
- форми захищені CSRF-токенами;
- доступ розмежований за ролями (RBAC);
- спроби несанкціонованого доступу фіксуються в журналі подій [17].

Висновкидо розділу 2

У другому розділі виконано моделювання предметної області та сформовано повну специфікацію вимог до вебзастосунку FootballShop.

Функціональну структуру системи описано засобами нотації IDEF0. Побудовано контекстну діаграму рівня A-0, що фіксує зовнішні взаємодії системи, та діаграми декомпозиції рівнів A0 і A3, які деталізують чотири ключові процеси: управління користувачами, управління каталогом, обробку замовлень і адміністрування. Модель унаочнює інформаційні потоки та взаємозв'язки між компонентами.

Специфікацію вимог складено відповідно до міжнародних стандартів ISO/IEC 25010:2011 та IEEE 830-1998. Вона охоплює призначення системи, характеристики трьох категорій користувачів (гість, зареєстрований покупець, адміністратор), функціональні вимоги з прив'язкою до конкретних ідентифікаторів (FR-1.1–FR-3.5), а також вимоги до інформаційного, технічного та програмного забезпечення.

Визначено трирівневу архітектуру на базі патерну MVC – рівні представлення, бізнес-логіки та доступу до даних – і описано RESTful API як інтерфейс взаємодії між ними.

Сформульовані вимоги до якісних характеристик – доступності, супроводжуваності, переносимості, продуктивності, надійності та безпеки – слугуватимуть критеріями оцінки системи на етапі тестування. Усі результати цього розділу є безпосередньою основою для проєктування архітектури та реалізації у наступних розділах роботи.

3 ПРОЄКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ

3.1 Діаграма варіантів використання

Для формалізованого опису поведінки FootballShop розроблено комплекс UML-діаграм. Система охоплює три ролі користувачів, нетривіальну логіку оформлення замовлень і багаторівневий бекенд – усе це зумовило вибір конкретних типів діаграм: варіантів використання, послідовності, діяльності та класів.

Діаграма варіантів використання фіксує межі системи і показує, які дії доступні кожному з трьох акторів – Гостю, Зареєстрованому користувачу та Адміністратору (рис. 3.1).

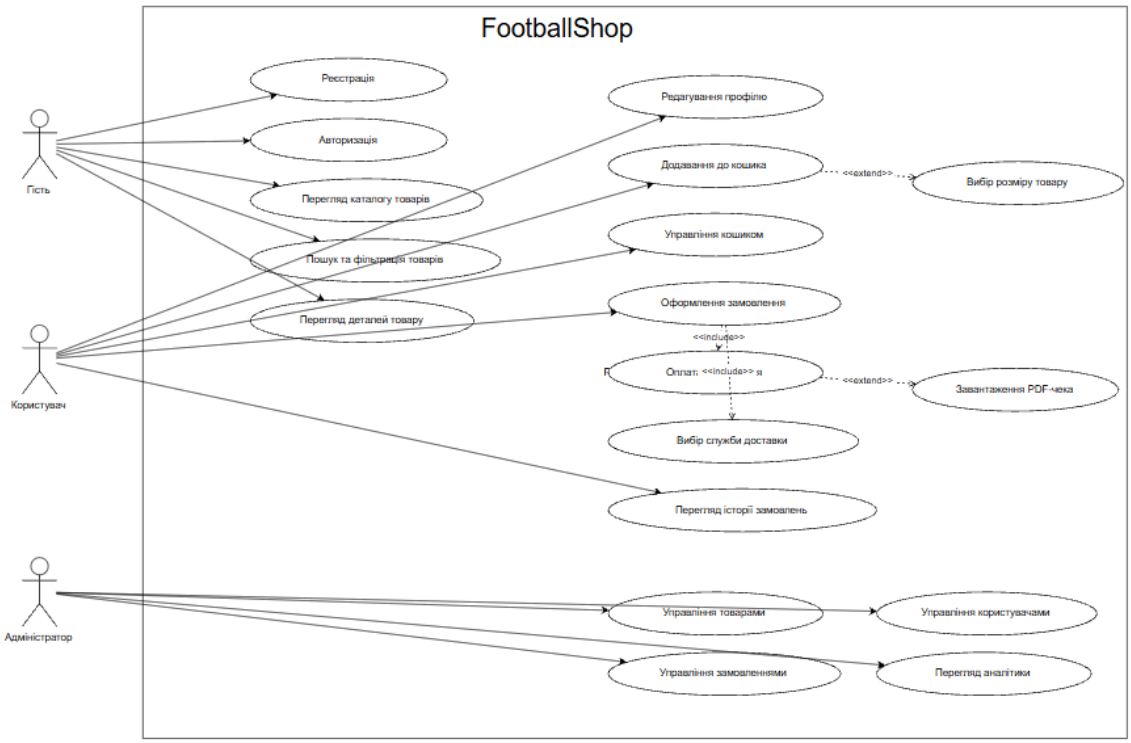

Рисунок 3.1 – Діаграма варіантів використання FootballShop

Гість взаємодіє з системою без реєстрації: переглядає каталог із фільтрацією за категорією та ціною, шукає товари за назвою, відкриває детальну картку позиції – розмірну сітку, опис, посилання на сайт бренду – а також реєструється або входить в акаунт.

Зареєстрований користувач успадковує весь функціонал гостя і отримує доступ до персонального простору: додає товари до кошика з вибором розміру, керує його вмістом, проводить оплату через форму з даними картки, відстежує статуси замовлень на таймлайні, завантажує PDF-чек після підтвердженої оплати, управляє списком обраного і редагує профіль.

Адміністратор працює в окремому модулі керування: додає, редагує і видаляє товарні позиції (включно з URL зображення та категорією), переглядає всі замовлення з фільтрацією за статусом і змінює їх, управляє обліковими записами та аналізує дашборд – графіки продажів по місяцях, топ-5 товарів і статистику нових користувачів.

3.2 Діаграми послідовності

Діаграми послідовності показують часовий порядок обміну повідомленнями між об'єктами у конкретному сценарії. Для FootballShop побудовано три такі діаграми, що покривають ключові бізнес-процеси.

Автентифікація користувача (рис. 3.2)

Процедура входу й перевірки прав побудована таким чином: користувач вводить облікові дані і браузер надсилає POST-запит на /api/account/login. AccountController приймає запит і передає дані до UserService, який звертається до ApplicationDbContext для пошуку запису за email у таблиці Users. Якщо запис знайдено, UserService верифікує пароль через BCrypt.Verify, порівнюючи введене значення з хешем у базі. Успішна перевірка запускає генерацію JWT-токена з claims – ідентифікатором, ім'ям і роллю користувача. Токен повертається клієнту, зберігається в Cookie, і браузер перенаправляє сесію на головну сторінку.

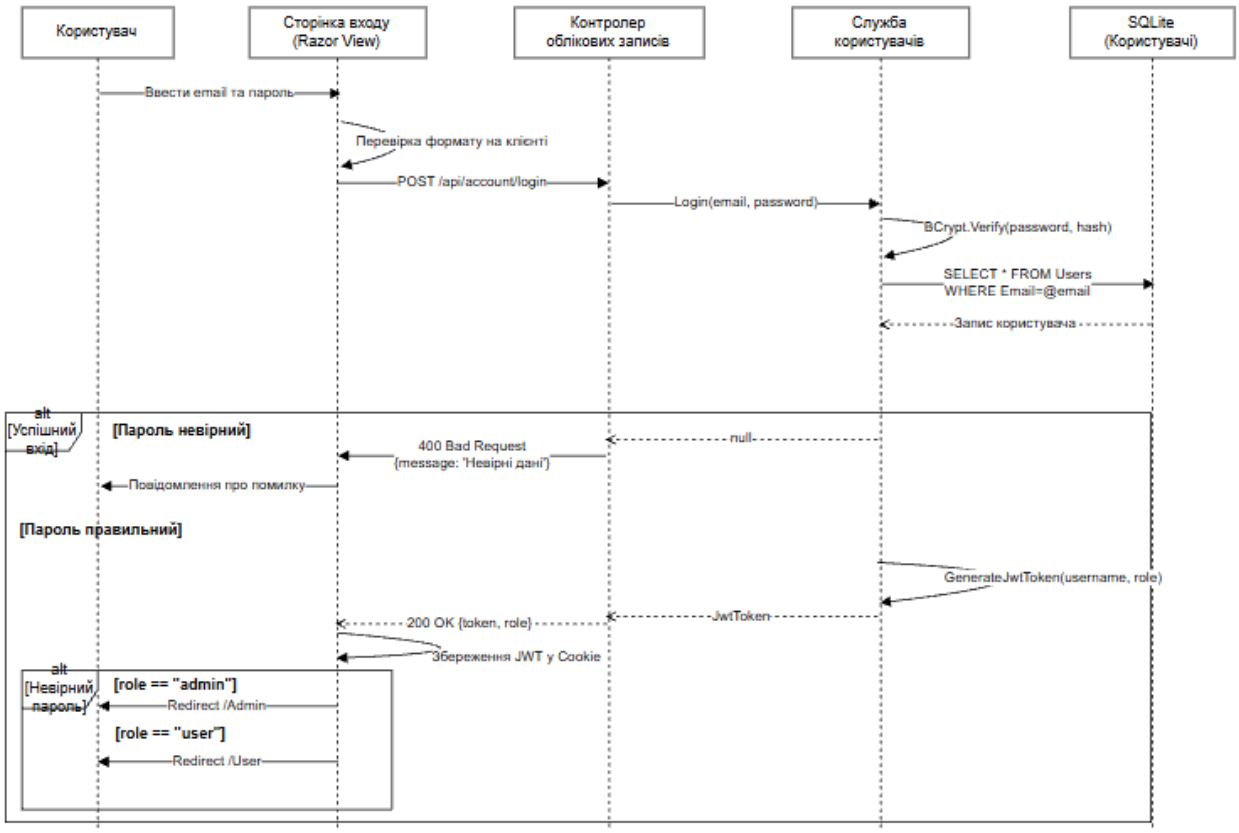

Рисунок 3.2 – Діаграма послідовності автентифікації користувача

Оформлення та оплата замовлення (рис. 3.3)

Сценарій купівлі та взаємодії користувача з кошиком відображає наступну логіку: авторизований покупець натискає «Оформити замовлення», і OrderController отримує POST-запит. Через OrderService зчитується поточний вміст кошика з таблиці CartItems. Сервіс формує новий запис у таблиці Orders зі статусом Pending і паралельно записує позиції до OrderItems – із зафіксованою ціною кожного товару на момент оформлення. Після збереження відбувається перехід на сторінку деталей замовлення. Покупець натискає «Оплатити» – відкривається двокрокове модальне вікно: перший крок збирає дані доставки та зберігає їх у Session, другий – реквізити картки, після чого надсилається POST-запит на /api/orders/{id}/pay. OrderApiController переводить статус у Processing і перенаправляє на сторінку успішної оплати з деталями й кнопкою завантаження PDF-чека.

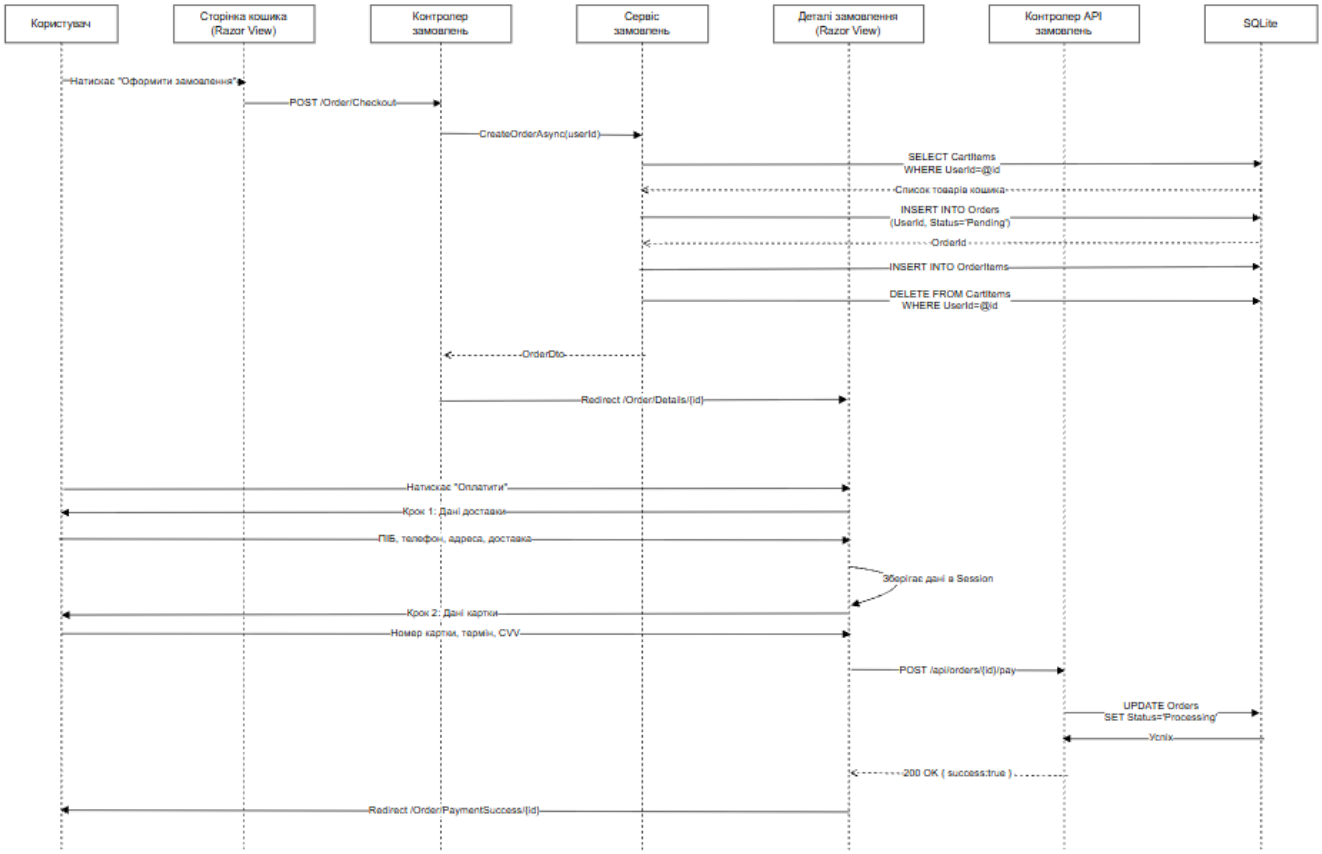

Рисунок 3.3 – Діаграма послідовностей оформлення та оплати замовлення

Управління товарами адміністратором (рис. 3.4)

Адміністратор відкриває /Admin/ManageProducts. AdminController перевіряє роль через атрибут [Authorize(Roles = "admin")] і рендерить сторінку. Одразу після завантаження виконується GET-запит на /api/products – повертається повний список товарів у JSON. Натискання «Додати товар» відкриває модальне вікно; адміністратор заповнює назву, категорію, ціну, URL зображення і опис та підтверджує форму. POST-запит іде на /api/products: ProductApiController через ProductService зберігає новий запис у таблиці Products і повертає успішну відповідь. Таблиця товарів оновлюється без перезавантаження сторінки через повторний AJAX-запит.

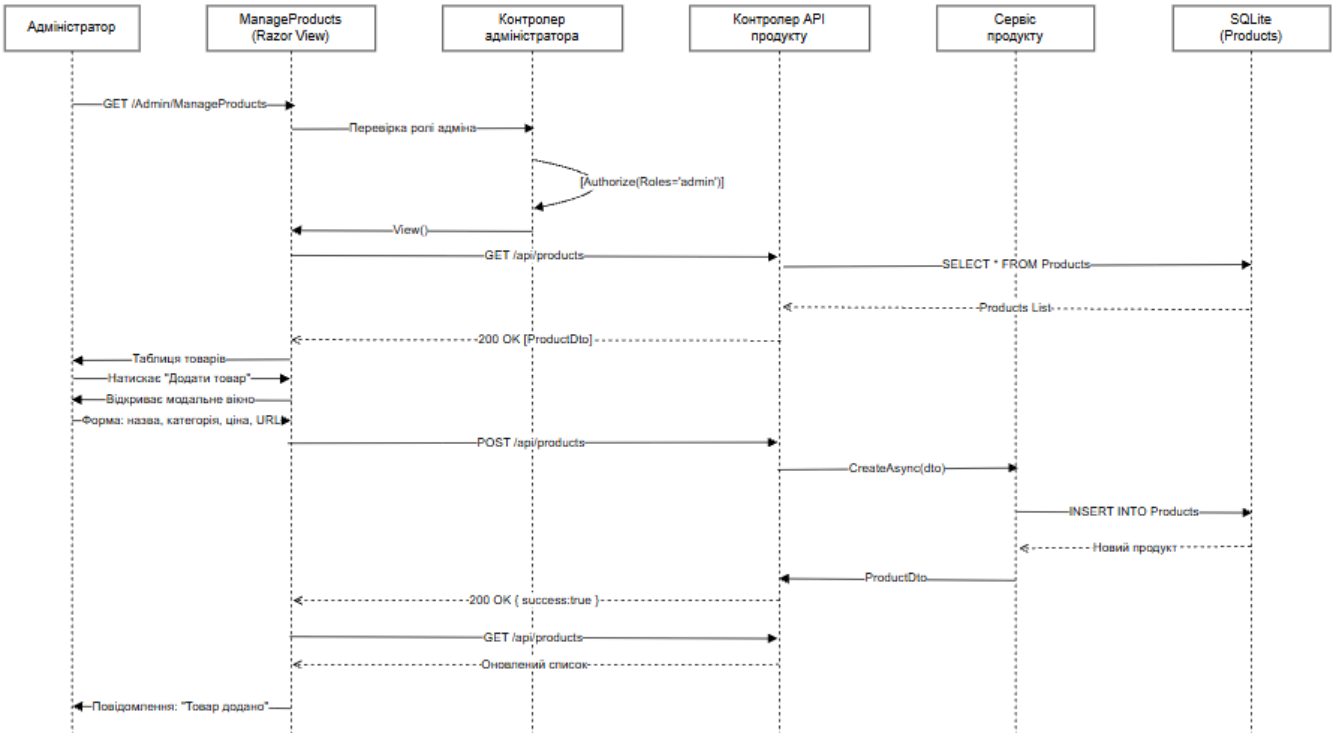

Рисунок 3.4 – Діаграма послідовності управління товарами адміністратором

3.3 Діаграми діяльності

Діаграми діяльності описують покроковий перебіг процесів із умовними переходами та альтернативними гілками. Для FootballShop побудовано три такі діаграми.

Авторизація і розподіл за ролями (рис. 3.5)

Після відкриття сторінки входу користувач вводить email і пароль. Клієнтська валідація перевіряє формат – помилка повертає до форми з підказкою. Коректні дані йдуть на сервер: якщо email не знайдено в базі, відображається відповідне повідомлення. Знайдений запис проходить BCrypt-перевірку пароля – невідповідність дає помилку автентифікації. При успішному збігу система зчитує роль: адміністратор отримує редирект на /Admin, звичайний користувач – на головну сторінку.

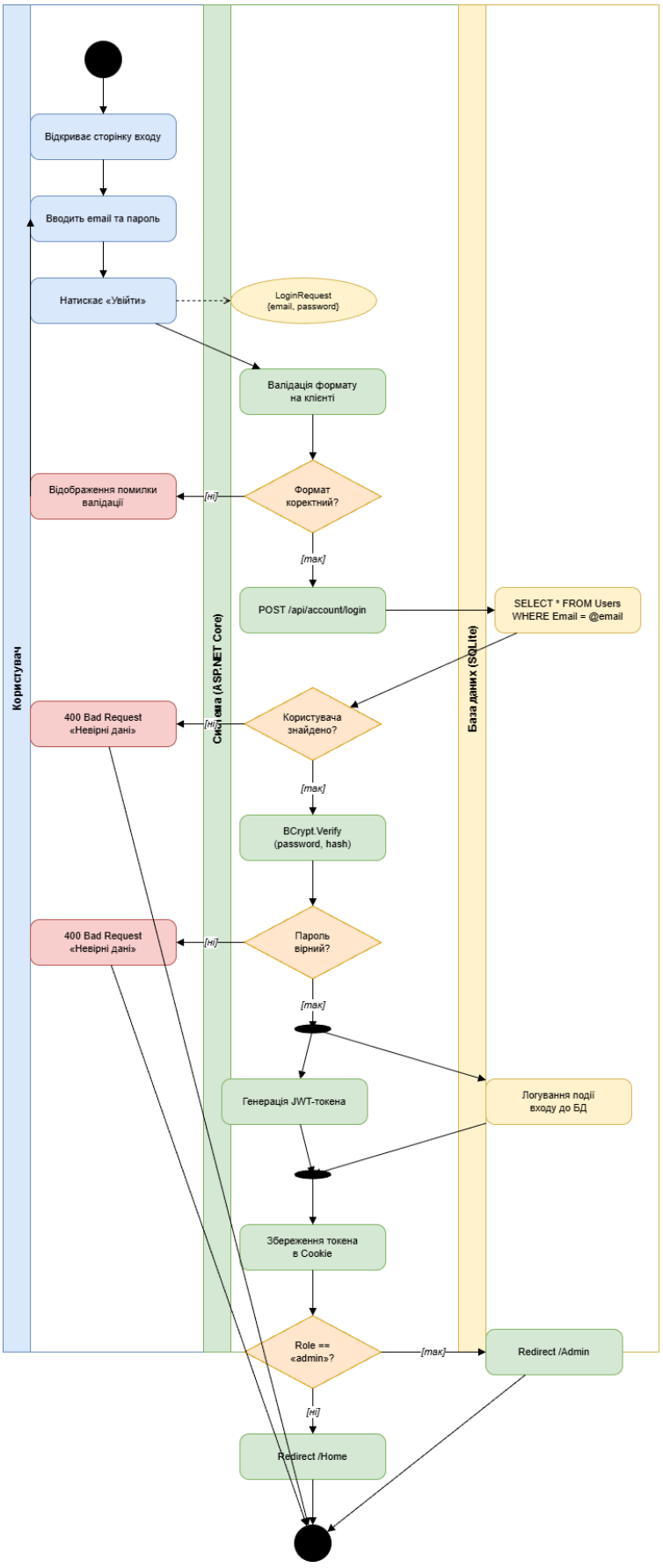

Рисунок 3.5 – Діаграма діяльності процесу авторизації

Додавання до кошика та оформлення замовлення (рис. 3.6)

Сценарій купівлі та взаємодії покупця з кошиком розгортається таким чином: відвідувач відкриває картку конкретного товару, і платформа автоматично визначає категорію, відображаючи відповідну розмірну сітку. Взуття отримує лінійку US 33–47, одяг — S–XXL, позиції без розмірної специфіки не потребують вибору взагалі. Якщо розмір є обов'язковим, але покупець його не обрав, блок вибору підсвічується червоним контуром із попереджувальним повідомленням — кнопка додавання в кошик залишається заблокованою. Щойно розмір зафіксовано або його вибір не потрібен, натискання «Додати до кошика» запускає перевірку сесії. Неавторизований користувач отримує редирект на сторінку входу; авторизований — потрапляє до кошика, де може скоригувати кількість одиниць або прибрати непотрібні позиції. Натискання «Оформити замовлення» створює запис у базі зі статусом Pending і переводить покупця на сторінку деталей, де через двокрокове модальне вікно виконується оплата.

Обробка замовлення адміністратором (рис. 3.7)

Робочий процес адміністратора з замовленнями починається на сторінці /Admin/ManageOrders. Система підвантажує весь перелік замовлень із бази і розміщує їх у таблиці з фільтрами за статусом. Клік на конкретному рядку відкриває модальне вікно з повною деталізацією: склад товарів, загальна сума і поточний статус. Набір доступних дій залежить від стану замовлення — для Processing активна кнопка «Відправити», що переводить статус у Shipped; для Shipped — «Доставлено», що завершує цикл переходом у Delivered. Кожна зміна фіксується PUT-запитом на /api/orders/{id}, після чого таблиця оновлюється автоматично без перезавантаження сторінки.

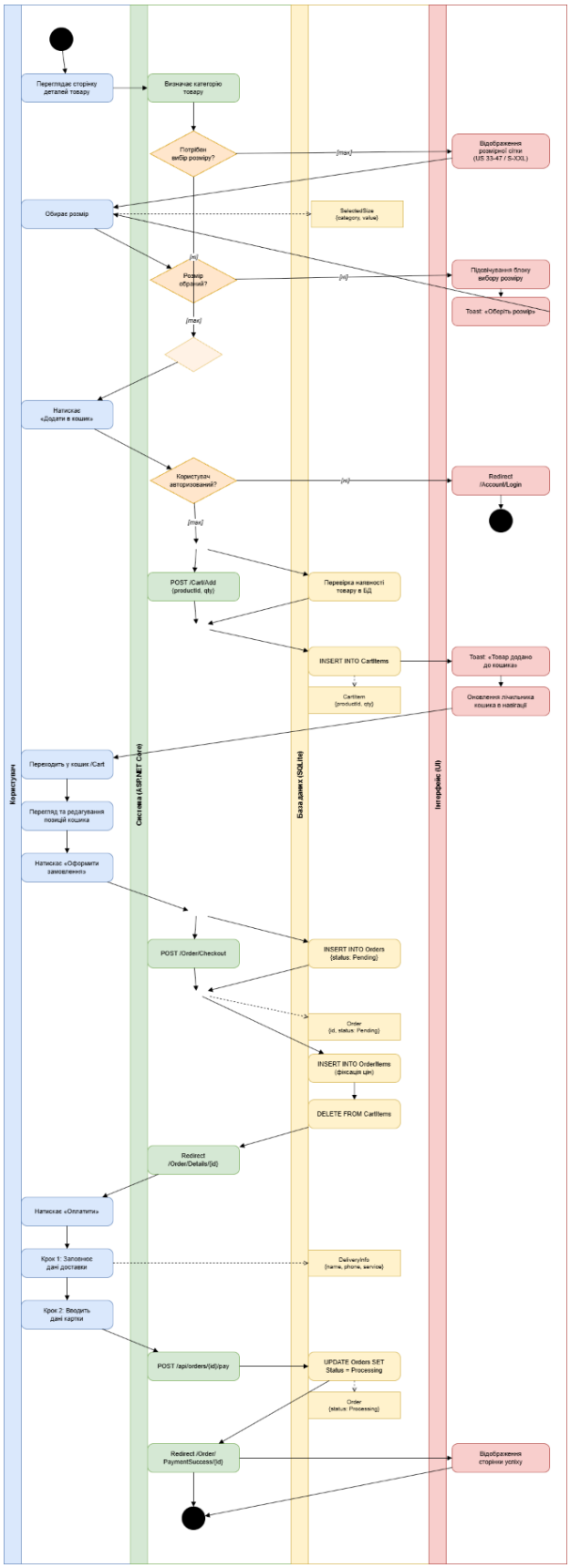

Рисунок 3.6 – Діаграма діяльності процесу додавання до кошика та оформлення замовлення

Рисунок 3.7 – Діаграма діяльності процесу обробки замовлення адміністратором

3.4 Діаграма класів

Діаграма класів відображає статичну структуру серверної частини: дев'ять класів предметної області, їхні атрибути та зв'язки між ними (рис. 3.8). Кожен клас відповідає окремій таблиці бази даних і реалізований як Entity-модель у проекті.

Центр моделі – клас Product із атрибутами Id, Name, Description, Price (decimal), Stock (int), ImageUrl (string) і CreatedAt (DateTime). Product пов'язаний з Category відношенням «багато до одного»: категорія об'єднує довільну кількість товарів, а кожен товар належить рівно одній категорії. Зв'язок реалізується через атрибут CategoryId і навігаційну властивість Category.

Клас User – Id, Username, Email, PasswordHash, Role, CreatedAt – є джерелом трьох відношень «один до багатьох»: до Order (через UserId – один користувач може мати безліч замовлень), до Cart (через UserId – персональний кошик покупця) і до Wishlist (через UserId – список обраного).

Клас Order з атрибутами Id, UserId, Username, TotalAmount, Status і CreatedAt використовує перелічуваний тип OrderStatus із п'ятьма значеннями. Order виступає власником для OrderItem – позиції в чеку не мають сенсу без самого факту замовлення, що й реалізує відношення композиції «один до багатьох».

Аналогічна логіка діє між Cart і CartItem: елементи кошика існують виключно в контексті конкретного кошика. Клас Wishlist зв'язує User і Product через атрибути UserId, ProductId і AddedAt. Клас Payment перебуває у відношенні «один до одного» з Order – кожному замовленню відповідає один запис про оплату.

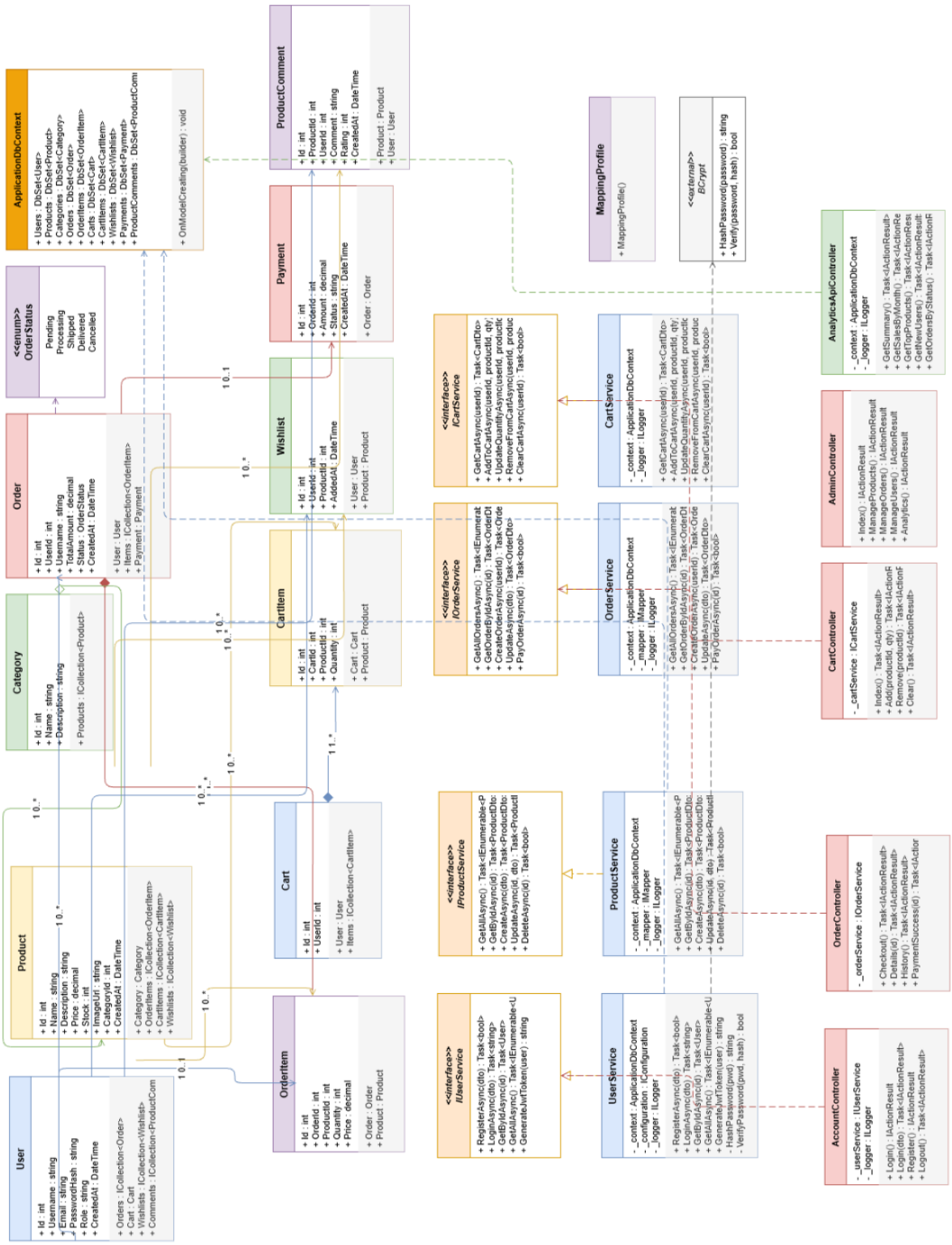

Рисунок 3.8 – Діаграма класів серверної частини FootballShop

3.5 Проєктування бази даних

База даних структурована за трьома функціональними блоками: «Користувачі», «Каталог» та «Продажі і клієнтський досвід» (рис. 3.9).

База даних FootballShop зберігається у вигляді локального SQLite-файлу (.db) безпосередньо в директорії проекту – це усуває потребу в окремому сервері СУБД і спрощує розгортання. Структура організована за трьома функціональними блоками.

Блок «Користувачі»

Єдина таблиця цього блоку – Users – зберігає облікові записи всіх зареєстрованих у системі. До її структури входять первинний ключ Id, унікальний Username, унікальний Email, хеш пароля PasswordHash отриманий через BCrypt, роль Role з двома можливими значеннями («user» / «admin»), а також службова мітка дати реєстрації CreatedAt. Зберігання ролі безпосередньо в цій таблиці забезпечує просту і ефективну модель доступу без додаткової таблиці ролей.

Блок «Каталог»

Складається з двох таблиць. Categories є довідником категорій: поля Id, Name і Description. Products – центральна сутність каталогу: Name, Description, Price, Stock, ImageUrl, зовнішній ключ CategoryId на таблицю Categories і службове поле CreatedAt. Один запис у Categories може охоплювати довільну кількість Products, тоді як кожен товар належить рівно одній категорії.

Блок «Продажі і клієнтський досвід»

П'ять таблиць. Orders фіксує замовлення: зовнішній ключ UserId, денормалізоване поле Username, загальна сума TotalAmount, статус Status (Pending / Processing / Shipped / Delivered / Cancelled) і дата створення CreatedAt. OrderItems зберігає склад кожного замовлення: OrderId, ProductId, Quantity і Price – ціна фіксується на момент оформлення, що унеможливлює її подальший перерахунок при зміні прайсу. Carts прив'язує кошик до конкретного користувача через UserId. CartItems містить позиції кошика із посиланнями на Carts і Products та кількість Quantity.

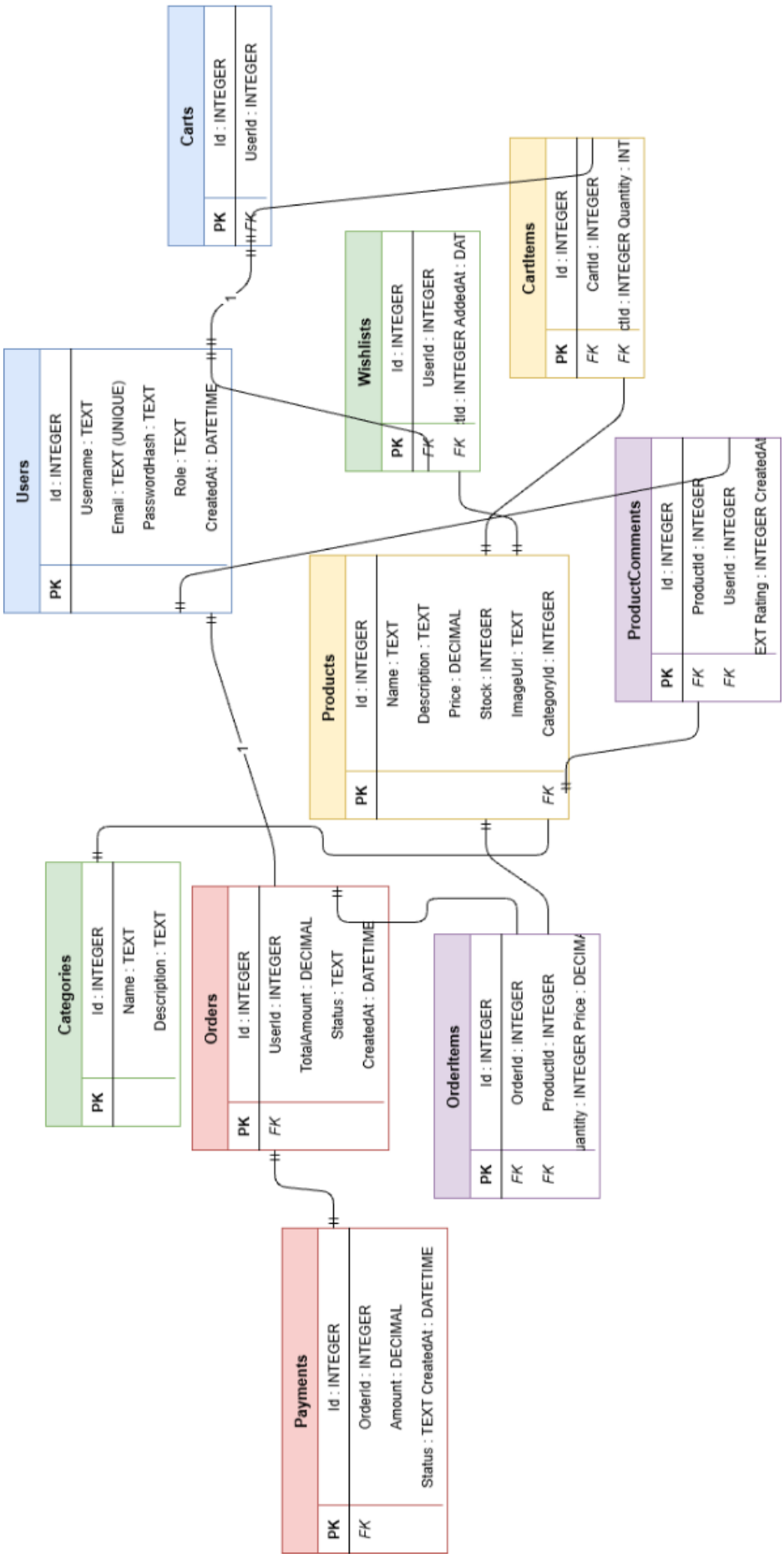

Рисунок 3.9 – ER-діаграма бази даних FootballShop

Первинні ключі всіх таблиць – цілочисельні ідентифікатори з автоінкрементом. Зовнішні ключі забезпечують референційну цілісність. Поля, що беруть участь у пошуку та фільтрації, індексуються для швидкодії запитів. Повну структуру таблиць наведено в таблиці 3.1.

Таблиця 3.1 – Структура таблиць бази даних FootballShop

Таблиця	**Поля**	**Зв'язки**
Users	Id, Username, Email, PasswordHash, Role, CreatedAt	Orders, Carts, Wishlists
Categories	Id, Name, Description	Products
Products	Id, Name, Description, Price, Stock, ImageUrl, CategoryId, CreatedAt	Categories, OrderItems, CartItems, Wishlists
Orders	Id, UserId, Username, TotalAmount, Status, CreatedAt	Users, OrderItems
OrderItems	Id, OrderId, ProductId, Quantity, Price	Orders, Products
Carts	Id, UserId	Users, CartItems
CartItems	Id, CartId, ProductId, Quantity	Carts, Products
Wishlists	Id, UserId, ProductId, AddedAt	Users, Products

3.6 Архітектура програмного забезпечення

FootballShop побудований за триланковою клієнт-серверною схемою з чітким поділом на рівень представлення, бізнес-логіки та даних. Кожен рівень незалежний від деталей реалізації сусіднього – це спрощує масштабування і підтримку. Фізичне розміщення компонентів відображено на діаграмі розгортання (рис. 3.10).

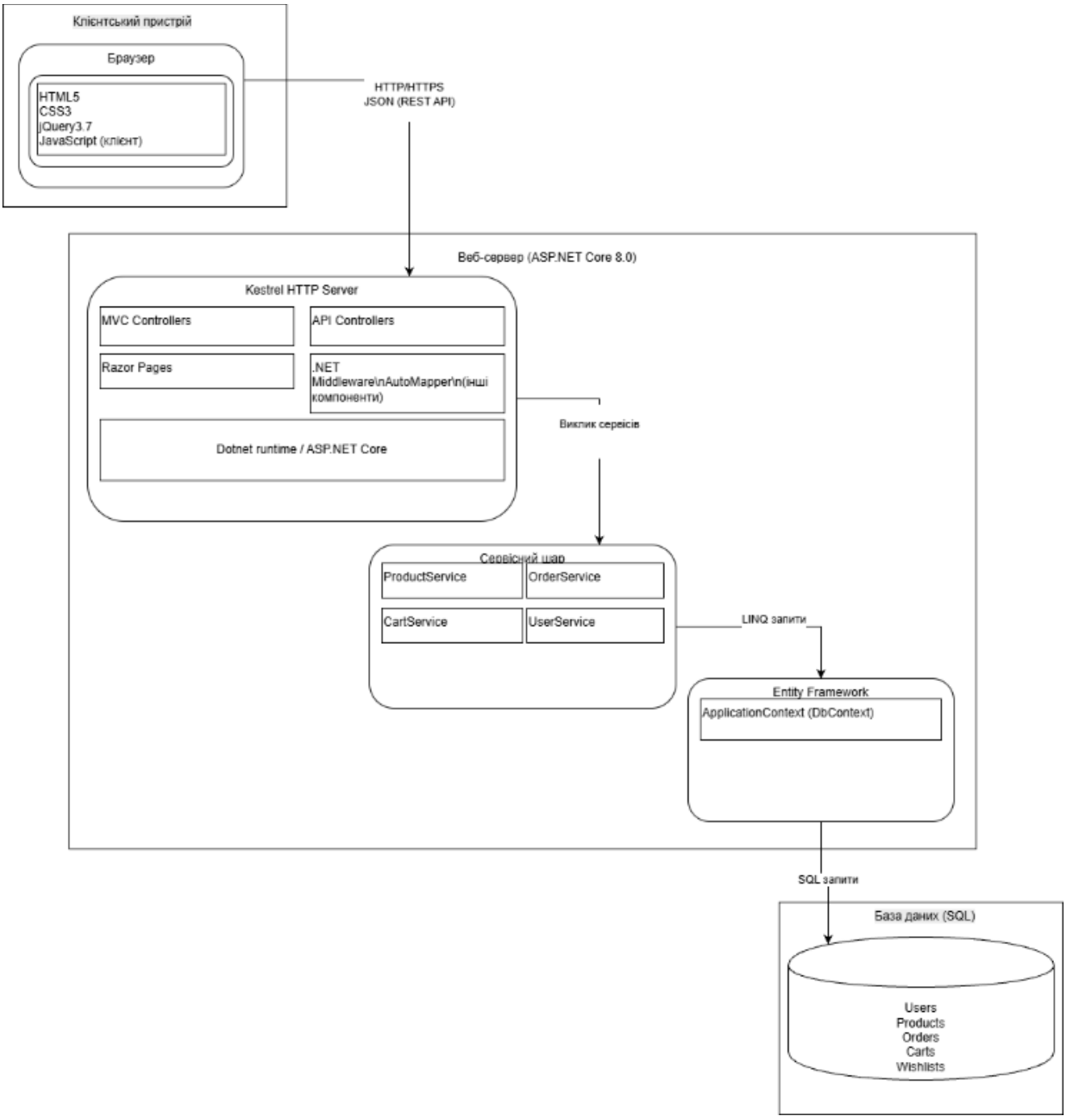

Рисунок 3.10 – Діаграма розгортання вебзастосунку FootballShop

Клієнтський рівень спирається на зв'язку HTML5/CSS3 і Bootstrap 5.3 для формування адаптивного інтерфейсу, коректного на екранах від 320px до 1920px і ширше. За динамічну поведінку відповідає JavaScript із бібліотекою jQuery 3.7. Усі асинхронні звернення до серверного API йдуть через AJAX – вміст сторінок оновлюється без повного перезавантаження. Серверний рендеринг HTML реалізований через Razor Pages: технологія дозволяє вбудовувати C#-логіку безпосередньо в шаблон.

За обробку правил і координацію даних відповідає бекенд-шар на ASP.NET Core 8.0 із патерном MVC. Він містить: MVC-контролери для HTTP-запитів від браузера, що повертають Razor-сторінки з моделями; API-контролери для AJAX-запитів, що повертають JSON; сервіси (ProductService, OrderService, CartService, UserService) з бізнес-логікою; бібліотеку AutoMapper для автоматичного перетворення між Entity-об'єктами і DTO. Автентифікація поєднує JWT-токени для API-запитів і Cookie – для MVC-сторінок. Паролі захищені хешуванням через BCrypt.

Рівень даних будується на Entity Framework Core 8.0 з підходом Code-First. ApplicationDbContext описує набори сутностей і налаштовує зв'язки між ними. Взаємодія з базою даних здійснюється через провайдер Microsoft.EntityFrameworkCore.Sqlite – вся база живе в одному .db-файлі в директорії проекту. Схема версіонується разом із вихідним кодом через механізм міграцій EF Core.

Загальний ланцюг взаємодії виглядає так: браузер надсилає HTTP/HTTPS-запит до ASP.NET Core, контролер маршрутизує його до відповідного сервісу, сервіс звертається до бази SQLite через EF Core і повертає результат – у вигляді HTML-сторінки або JSON-об'єкта залежно від типу запиту.

3.7 Проєктування інтерфейсу застосунку

Візуальна концепція FootballShop побудована на принципах мінімалізму та зручності: темна кольорова схема на основі відтінків зеленого, чіткі акценти й відсутність зайвих елементів. Bootstrap 5.3 забезпечує правильне розташування компонентів на будь-якому екрані – від смартфону шириною 320px до широкоформатного монітора 1920px і ширше. Кожна сторінка відкривається з єдиною навігаційною панеллю: логотип магазину, посилання на каталог, іконка кошика з лічильником поточних позицій і блок авторизації, що динамічно перемикається між кнопками входу та посиланням на профіль залежно від стану сесії. Підвал однаковий для всіх сторінок – він містить посилання на офіційні сайти брендів-партнерів і підключається через Razor Layout як єдиний спільний компонент. Нижче наведено опис п'яти ключових екранних форм застосунку.

Головна сторінка (рис. 3.11)

Простір головної сторінки відкривається з Hero-секції – великого акцентного блоку з вітальним текстом і двома кнопками-закликами: «Переглянути каталог» і «Дізнатися більше». Нижче вбудовано секцію «Чому обирають нас» – чотири лаконічні картки переваг: швидка доставка, гарантія якості, легке повернення і цілодобова підтримка. Наступний логічний крок для відвідувача супроводжується динамічною сіткою «Популярні товари»: вісім позицій, що завантажуються через AJAX-запит до /api/products і відображаються в адаптивній розкладці карток. Завершує сторінку блок категорій – їхній перелік підтягується з бази даних і виводиться в один горизонтальний рядок.

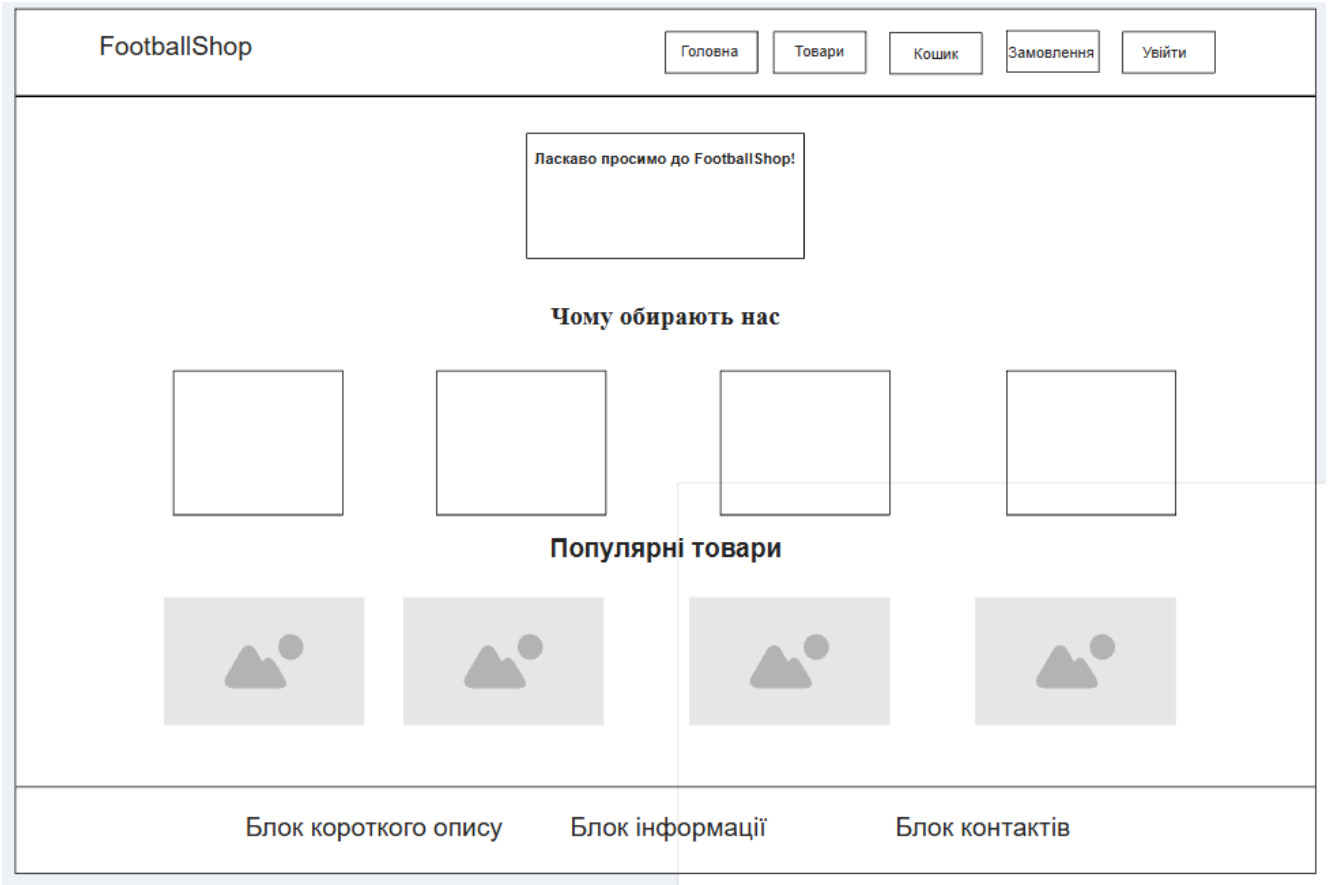

Рисунок 3.11 – Макет головної сторінки

Сторінка каталогу товарів (рис. 3.12)

Каталог відкривається рядком пошуку з живими підказками. Одразу під ним розгортається панель фільтрів: випадний список категорій, поля мінімальної та максимальної ціни, чекбокс «Тільки в наявності» і список сортування за ціною або назвою. Фільтрація відпрацьовує миттєво завдяки JavaScript-обробці локального масиву даних – жодного перезавантаження сторінки. Основну площу займає адаптивна сітка карток товарів. Кожна картка несе зображення, назву, короткий опис, ціну й індикатор наявності; користувачу доступні інтерактивні елементи для переходу до повної картки товару або миттєвого додавання позиції в кошик.

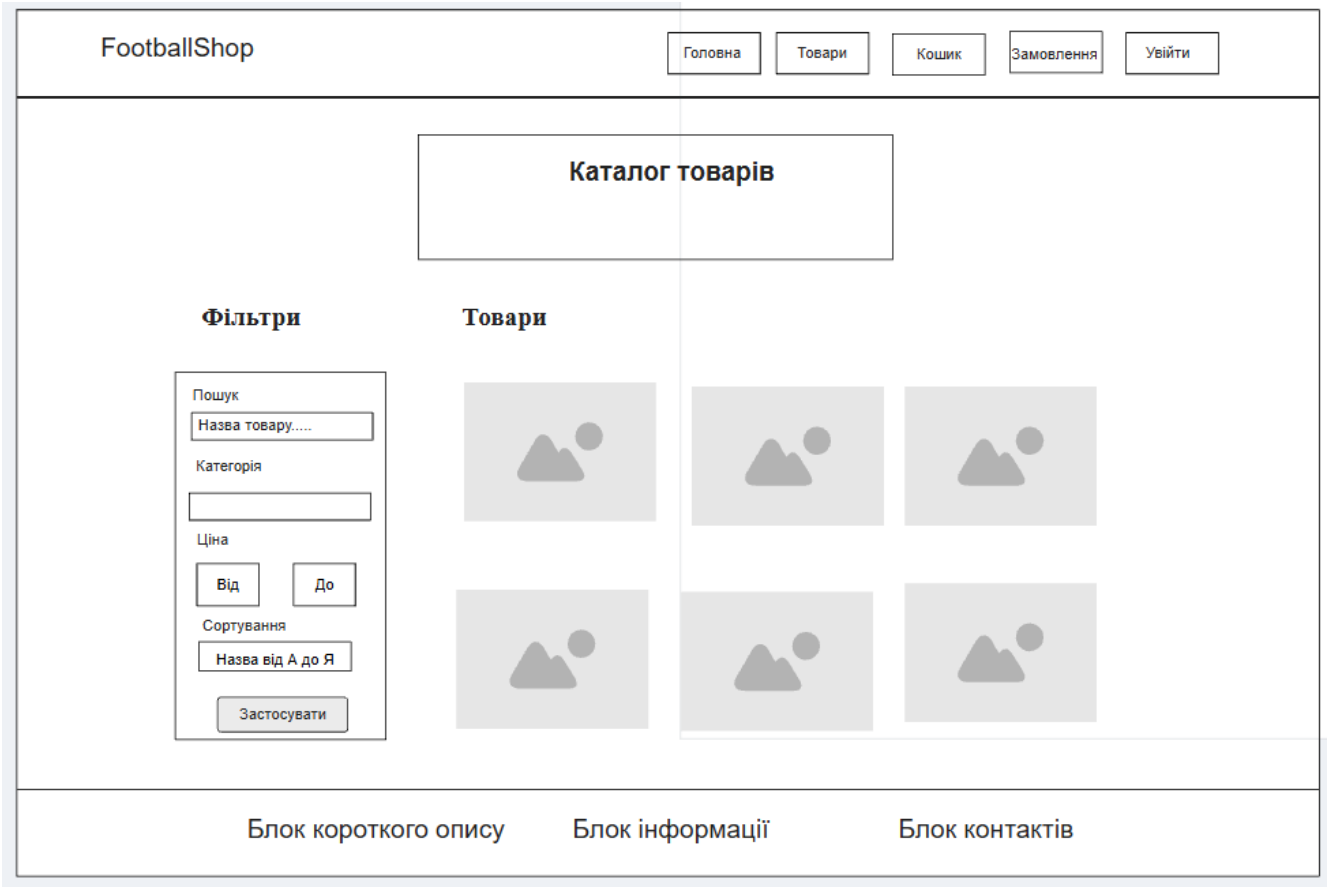

Рисунок 3.12 – Макет сторінки каталогу товарів

Сторінка деталей товару (рис. 3.13)

Ліва колонка відведена під галерею: велике головне зображення і чотири мініатюри – клік на будь-яку з них підмінює головне фото. Права частина концентрує всю комерційну інформацію: назва товару, зірковий рейтинг, ціна, блок вибору розміру, що автоматично підлаштовується під категорію (взуття – US 33–47, одяг – S–XXL, інші – без вибору розміру), розгорнутий опис і таблиця характеристик. Якщо розмір обов'язковий, але не обраний, система блокує додавання до кошика і підсвічує блок вибору червоним контуром. Поруч із кнопкою «В кошик» розміщено лічильник кількості і кнопка «В обране» для додавання до Wishlist. Нижня секція сторінки динамічно підвантажує схожі товари через AJAX з фільтром за тією самою категорією.

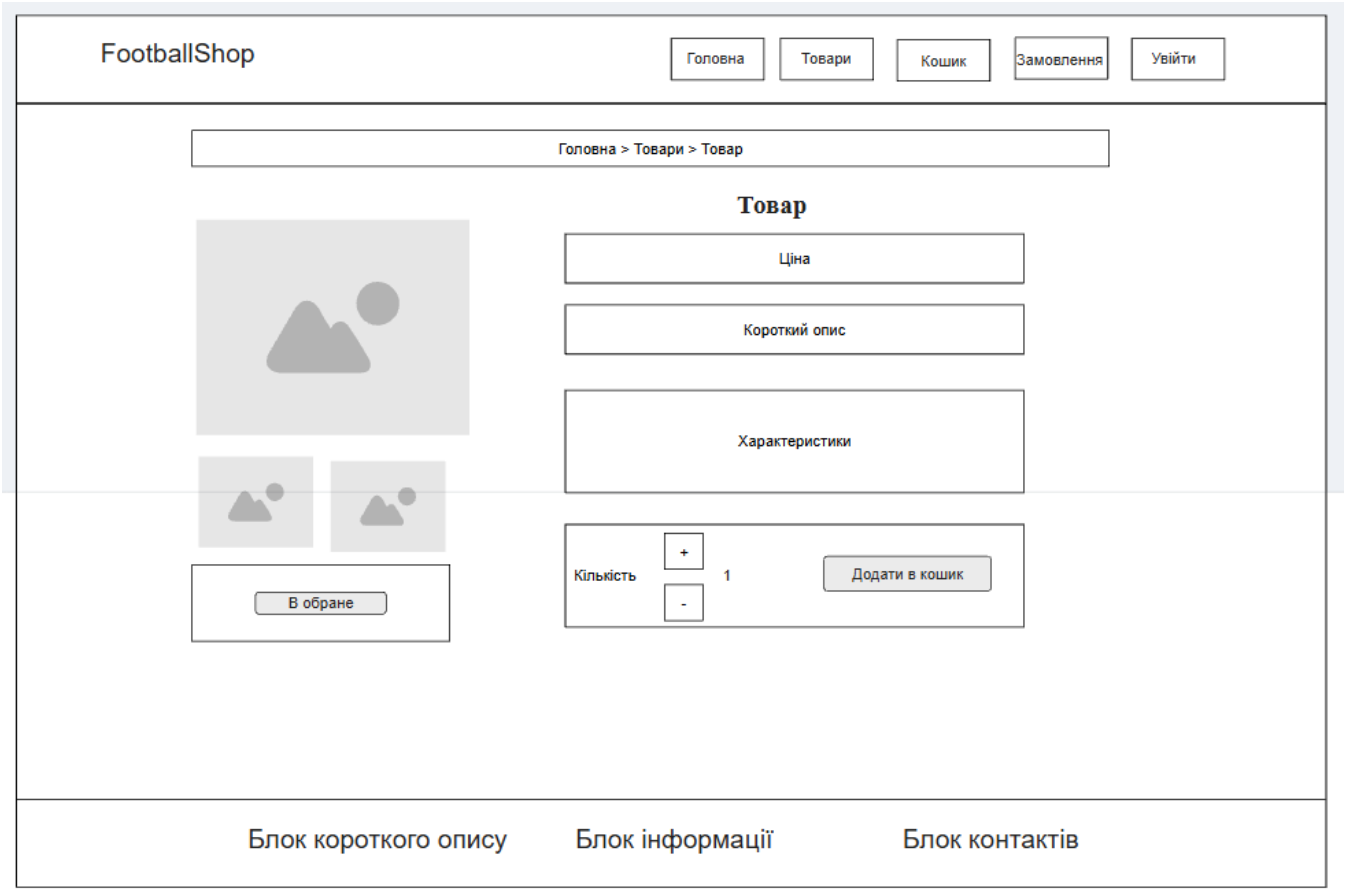

Рисунок 3.13 – Макет сторінки деталей товару

Сторінка кошика (рис. 3.14)

Інтерфейс кошика поділений на дві зони. Ліва займає більшу частину екрану – тут покупець бачить усі додані позиції: зображення, назву, лічильник кількості з кнопками «+» і «–», ціну за одиницю і підсумок по позиції. Кожну позицію можна прибрати окремою кнопкою видалення. Права зона – блок підсумку: загальна сума, рядок «Доставка: Безкоштовно» і кнопка «Оформити замовлення», яка запускає подальший флоу.

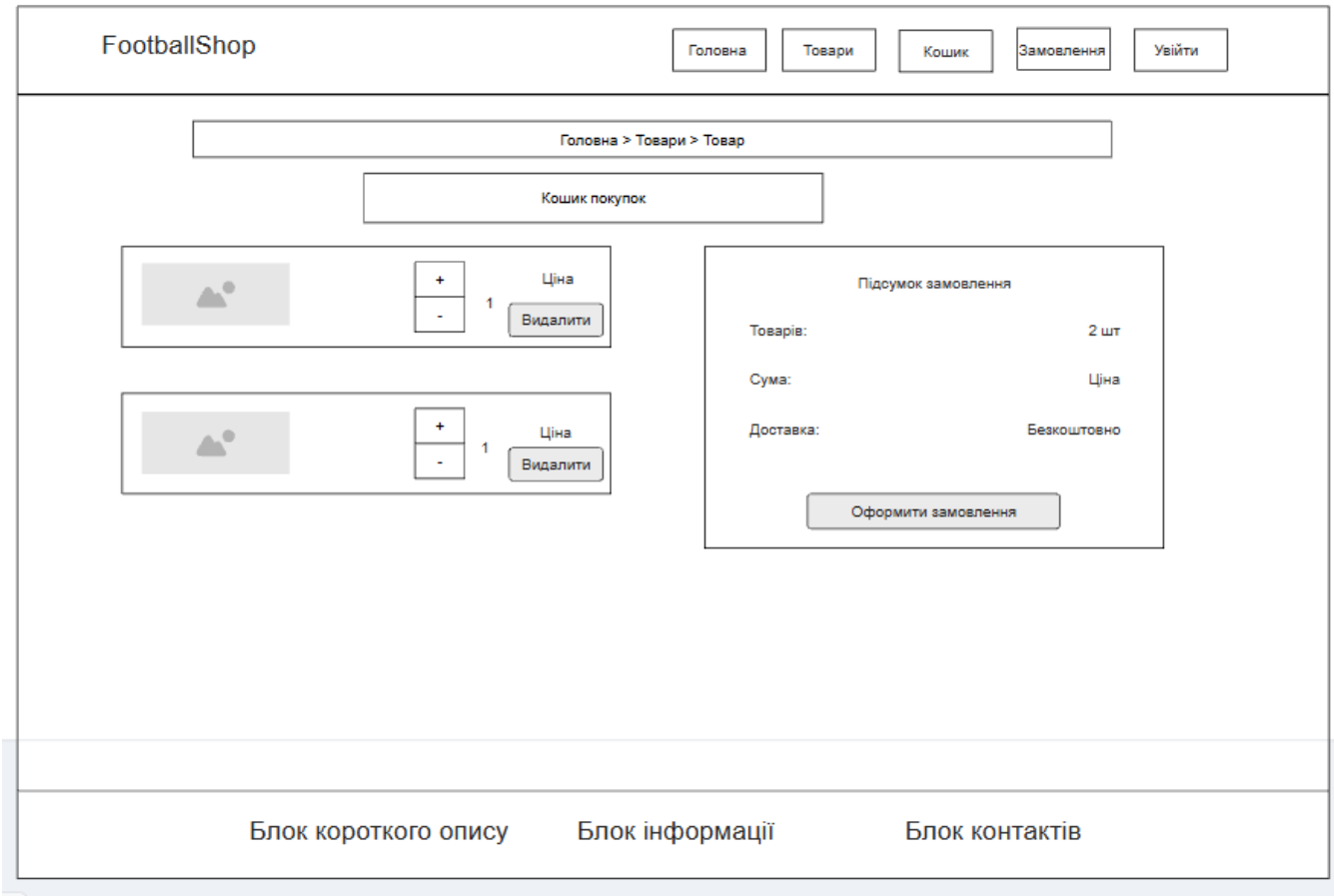

Рисунок 3.14 – Макет сторінки кошика

Сторінка замовлень (рис. 3.15)

Шапка сторінки показує загальну кількість оформлених замовлень і їхній розподіл за статусами. Основна область відображає замовлення у вигляді карток – кожна містить номер, дату, перелік товарів, загальну вартість, спосіб доставки і кольоровий бейдж статусу (Pending, Processing, Shipped, Delivered, Cancelled). Натискання на замовлення розгортає деталі: найменування позицій, кількість, ціна за одиницю і підсумкова сума. Для активних замовлень доступний таймлайн обробки й доставки. Фільтрація за статусом і сортування за датою реалізовані у верхній частині без перезавантаження сторінки. Якщо у покупця ще немає жодного замовлення – система відображає інформаційний порожній стан із пропозицією перейти до каталогу.

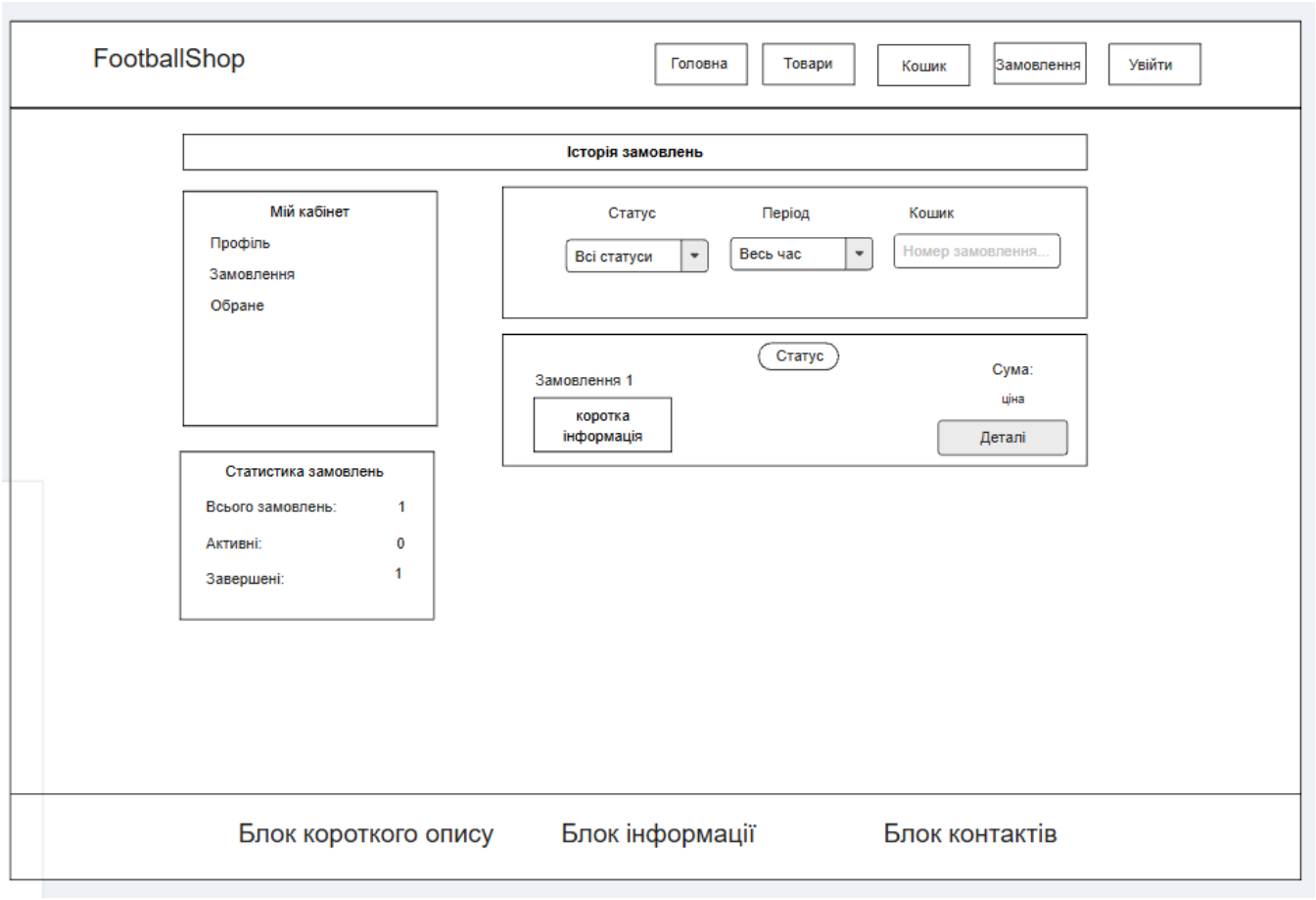

Рисунок 3.15 – Макет історії замовлень

Висновки до розділу 3

Підсумовуючи архітектурний етап розробки FootballShop, можна стверджувати, що створена інженерна база є повною і достатньою для переходу до програмної реалізації.

Триланкова клієнт-серверна архітектура чітко розподіляє відповідальність: клієнтський рівень спирається на HTML5, CSS3, Bootstrap 5.3 і jQuery 3.7 для формування адаптивного інтерфейсу; бекенд-шар на ASP.NET Core 8.0 з патерном MVC координує бізнес-правила і обробку запитів; рівень даних будується на Entity Framework Core 8.0 з провайдером SQLite – вся база зберігається у локальному .db-файлі без потреби в окремому сервері СУБД. Комбінація JWT-токенів для API-запитів і Cookie-автентифікації для MVC-сторінок охоплює обидва сценарії

взаємодії з системою. BCrypt гарантує, що навіть при прямому доступі до бази жоден пароль не може бути відновлений.

Реляційна схема бази даних включає дев'ять взаємопов'язаних таблиць, розподілених за трьома функціональними блоками: «Користувачі», «Каталог» і «Продажі та клієнтський досвід». ER-діаграма фіксує всі сутності й зовнішньоключові зв'язки між ними. Окремої уваги заслуговує рішення щодо фіксації ціни товару в таблиці OrderItems на момент оформлення – це захищає історичні дані від будь-яких подальших коригувань прайсу.

Комплекс UML-діаграм охоплює всі ключові виміри системи: діаграма варіантів використання описує функціональні можливості для трьох ролей, три діаграми послідовності деталізують сценарії автентифікації, оформлення замовлення і управління товарами адміністратором, три діаграми діяльності відображають алгоритми з умовними переходами, а діаграма класів документує структуру дев'яти Entity-моделей і зв'язки між ними.

П'ять ключових екранних форм – головна сторінка, каталог, картка товару, кошик і сторінка замовлень – спроєктовані з єдиним дизайн-кодом і спільними компонентами через Razor Layout. Аналітичний дашборд адміністратора з графіками на базі Chart.js доповнює інтерфейс інструментами для аналізу продажів. Усі отримані проєктні рішення формують готову основу для програмної реалізації у наступному розділі.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ

4.1 Структура проєкту

FootballShop реалізований як єдиний ASP.NET Core 8.0 проєкт за патерном MVC. Така організація спрощує розгортання і водночас зберігає чіткий поділ відповідальності між шарами.

Вихідний код підпорядкований стандартним угодам ASP.NET Core MVC. У директорії Controllers логіку розділено на два типи класів. MVC-контролери обробляють HTTP-запити від браузера і повертають Razor-сторінки: HomeController відповідає за головну сторінку, ProductController – за каталог і картку товару, CartController – за кошик, OrderController – за оформлення і перегляд замовлень, AccountController – за реєстрацію й вхід, WishlistController – за список обраного. API-контролери винесено в підпапку Api – вони обслуговують AJAX-запити і повертають JSON: ProductApiController, CategoryApiController, CartApiController, OrderApiController, WishlistApiController, CommentsApiController і AnalyticsApiController.

Бізнес-правила ізольовано в директорії Services, де інтерфейси (IProductService, IOrderService, ICartService, IUserService) відокремлені від реалізацій у підпапках Interfaces і Implementation відповідно.

Директорія Models зберігає Entity-класи, що відображають таблиці бази даних: User, Product, Category, Order, OrderItem, Cart, CartItem, Wishlist, Payment і ProductComment. Об'єкти передачі даних зібрано у DTO і згруповано за функціональними областями: Products, Orders, Cart, Users, Wishlist і Comments.

У директорії Data міститься клас ApplicationDbContext з усіма DbSet-властивостями та налаштуванням зв'язків між сутностями. Поруч у Migrations лежать файли міграцій EF Core для версіонування схеми SQLite-бази.

Views – шаблони Razor, згруповані за контролерами: Home, Product, Cart, Order, Account, Wishlist і Shared. Статичні ресурси – CSS, JavaScript і зображення – зберігаються у wwwroot.

4.2 Реалізація серверної частини

Серверна частина FootballShop побудована на платформі .NET 8 із застосуванням ASP.NET Core MVC. Нижче розглянуто реалізацію чотирьох ключових компонентів: контексту бази даних, сервісу замовлень, контролера автентифікації та API товарів.

ApplicationDbContext

Клас ApplicationDbContext базується на базовому класі DbContext з Entity Framework Core і є головним інструментом рівня доступу до даних. Він оголошує десять DbSet-властивостей для всіх сутностей системи, а в методі OnModelCreating налаштовує їхні зв'язки і обмеження. На поле Email таблиці Users встановлено унікальний індекс. Для поля Price у Products і OrderItems вказано тип decimal(18,2). Зв'язок між Orders і Users реалізовано з поведінкою DeleteBehavior.Restrict – це запобігає каскадному видаленню замовлень при видаленні облікового запису.

Лістинг коду:

```
public class ApplicationDbContext : DbContext
{
    public ApplicationDbContext(
        DbContextOptions<ApplicationDbContext>options)
        : base(options) { }

    public DbSet<User>Users { get; set; }
    public DbSet<Product>Products { get; set; }
    public DbSet<Category>Categories { get; set; }
    public DbSet<Order>Orders { get; set; }
    public DbSet<OrderItem> OrderItems { get; set; }
    public DbSet<Cart>Carts { get; set; }
    public DbSet<CartItem> CartItems { get; set; }
    public DbSet<Wishlist> Wishlists { get; set; }
    public DbSet<Payment>Payments { get; set; }
    public DbSet<ProductComment> ProductComments { get; set; }

    protected override void OnModelCreating(
        ModelBuilder modelBuilder)
    {
        modelBuilder.Entity<User>()
            .HasIndex(u => u.Email)
            .IsUnique();

        modelBuilder.Entity<Product>()
            .Property(p => p.Price)
            .HasColumnType("decimal(18,2)");

        modelBuilder.Entity<OrderItem>()
            .Property(oi => oi.Price)
            .HasColumnType("decimal(18,2)");
```

```
        modelBuilder.Entity<Order>()
            .HasOne(o => o.User)
            .WithMany(u => u.Orders)
            .HasForeignKey(o => o.UserId)
            .OnDelete(DeleteBehavior.Restrict);
    }
}
```

OrderService – метод CreateOrderAsync

OrderService інкапсулює логіку керування замовленнями. Залежності ApplicationDbContext і IMapper впроваджуються на рівні конструктора (DI). Метод CreateOrderAsync зчитує поточний вміст кошика разом із товарними позиціями через ланцюжок Include/ThenInclude, формує новий запис Order зі статусом Pending і фіксує ціну кожного товару безпосередньо в OrderItem – це унеможливлює перерахунок вартості при майбутніх змінах прайсу. Кошик очищується в тому самому виклику SaveChangesAsync, що забезпечує атомарність усієї операції.

Лістинг коду:

```
public async Task<OrderDto?>CreateOrderAsync(int userId)
{
    var cart = await _context.Carts
        .Include(c => c.Items)
            .ThenInclude(i => i.Product)
        .FirstOrDefaultAsync(c => c.UserId == userId);

    if (cart == null || !cart.Items.Any())
        return null;

    var order = new Order
    {
        UserId = userId,
        Status = OrderStatus.Pending,
        TotalAmount = cart.Items.Sum(
            i => i.Product.Price * i.Quantity),
        CreatedAt = DateTime.Now
    };

    _context.Orders.Add(order);
    await _context.SaveChangesAsync();

    foreach (var item in cart.Items)
    {
        _context.OrderItems.Add(new OrderItem
        {
            OrderId = order.Id,
            ProductId = item.ProductId,
            Quantity = item.Quantity,
            Price = item.Product.Price
        });
    }
```

```
    _context.CartItems.RemoveRange(cart.Items);
    await _context.SaveChangesAsync();

    return _mapper.Map<OrderDto>(order);
}
```

AccountController – генерація JWT-токена

AccountController відповідає за реєстрацію і вхід. При реєстрації контролер перевіряє унікальність email, хешує пароль через BCrypt.Net.BCrypt.HashPassword і зберігає нового користувача з роллю user. Одночасно автоматично створюється порожній кошик. При вході пароль верифікується через BCrypt.Net.BCrypt.Verify, після чого виклик GenerateJwtToken формує підписаний токен із набором claims – ідентифікатор, ім'я, роль і email – і зберігає його в Cookie.

Лістинг коду:

```
private string GenerateJwtToken(User user)
{
    var claims = new[]
    {
        new Claim(ClaimTypes.NameIdentifier,
            user.Id.ToString()),
        new Claim(ClaimTypes.Name,
            user.Username),
        new Claim(ClaimTypes.Role,
            user.Role),
        new Claim(ClaimTypes.Email,
            user.Email)
    };

    var key = new SymmetricSecurityKey(
        Encoding.UTF8.GetBytes(
            _configuration["Jwt:SecretKey"]!));

    var token = new JwtSecurityToken(
        issuer: _configuration["Jwt:Issuer"],
        audience: _configuration["Jwt:Audience"],
        claims: claims,
        expires: DateTime.UtcNow.AddHours(24),
        signingCredentials: new SigningCredentials(
            key, SecurityAlgorithms.HmacSha256));

    return new JwtSecurityTokenHandler()
        .WriteToken(token);
}
```

ProductApiController – метод GetProducts

ProductApiController декорований атрибутами [ApiController] і [Route("api/products")] та надає повний CRUD для товарів. Метод GetProducts

приймає три необов'язкові query-параметри – categoryId, minPrice і maxPrice – і динамічно добудовує LINQ-запит через AsQueryable(). Умови фільтрації додаються тільки за наявності конкретного значення параметра. Результуючий список конвертується в колекцію DTO через _mapper.Map<IEnumerable<ProductDto>>. Весь метод огорнутий у try-catch з поверненням структурованої JSON-відповіді і кодом 500 при непередбаченій помилці.

Лістинг коду:

```
[HttpGet]
public async Task<IActionResult>GetProducts(
    [FromQuery] int? categoryId,
    [FromQuery] decimal? minPrice,
    [FromQuery] decimal? maxPrice)
{
    try
    {
        var query = _context.Products
            .Include(p => p.Category)
            .AsQueryable();

        if (categoryId.HasValue)
            query = query.Where(
                p => p.CategoryId == categoryId.Value);

        if (minPrice.HasValue)
            query = query.Where(
                p => p.Price >= minPrice.Value);

        if (maxPrice.HasValue)
            query = query.Where(
                p => p.Price <= maxPrice.Value);

        var products = await query.ToListAsync();
        var result = _mapper.Map
            IEnumerable<ProductDto>>(products);

        return Ok(result);
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, "Error getting products");
        return StatusCode(500,
            new { message = "Помилка отримання товарів" });
    }
}
```

AnalyticsApiController – метод GetTopProducts

AnalyticsApiController захищений атрибутом [Authorize(Roles = "admin")] і надає п'ять аналітичних endpoints для адмінського дашборду. Endpoint

/api/analytics/summary агрегує шість показників: загальний дохід, кількість замовлень, користувачів, товарів, замовлень у статусі очікування і дохід поточного місяця. Endpoint /api/analytics/sales-by-month повертає дані продажів за останні шість місяців для побудови графіків через Chart.js. Метод GetTopProducts групує записи з OrderItems за ідентифікатором і назвою товару, виключає скасовані та незавершені замовлення, підраховує загальну кількість і виручку по кожній позиції, а потім повертає п'ять лідерів у порядку спадання кількості продажів.

Лістинг коду:

```
[HttpGet("top-products")]
public async Task<IActionResult>GetTopProducts()
{
    try
    {
        var topProducts = await _context.OrderItems
            .Include(oi => oi.Product)
            .Where(oi =>
                oi.Order.Status != OrderStatus.Cancelled &&
                oi.Order.Status != OrderStatus.Pending)
            .GroupBy(oi =>new
            {
                oi.ProductId,
                oi.Product.Name
            })
            .Select(g => new
            {
                productName = g.Key.Name,
                totalQuantity = g.Sum(oi => oi.Quantity),
                totalRevenue = g.Sum(
                    oi => oi.Price * oi.Quantity)
            })
            .OrderByDescending(x => x.totalQuantity)
            .Take(5)
            .ToListAsync();

        return Ok(topProducts);
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, "Error getting top products");
        return StatusCode(500,
            new { message = ex.Message });
    }
}
```

4.3 Керівництво користувача

Реєстрація та вхід (рис. 4.1 та рис. 4.2)

Взаємодія з платформою для нового відвідувача починається на сторінці /Account/Register. Реєстраційна форма містить поля імені, електронної пошти, пароля і його підтвердження. Первинна клієнтська перевірка – відповідність паролів і коректність формату email – спрацьовує ще до відправки запиту, знижуючи зайве навантаження на сервер. Після підтвердження форми дані надходять на /api/account/register: пароль хешується через BCrypt, новий обліковий запис зберігається з роллю user, а паралельно автоматично ініціалізується порожній кошик. Браузер перенаправляється на головну сторінку з підтвердженням успішної реєстрації.

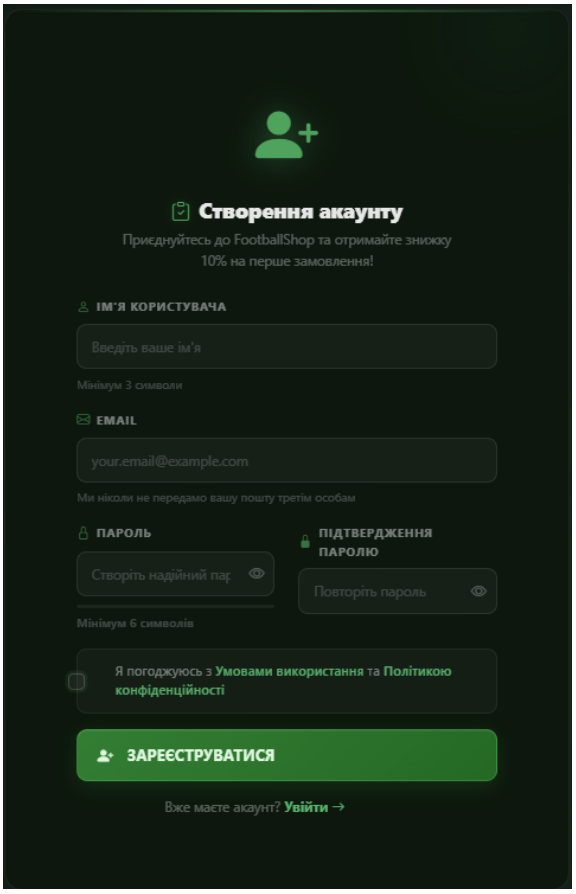

Рисунок 4.1 – Форма реєстрації нового користувача

Повторний вхід відбувається через /Account/Login. Після введення email і пароля запит іде на сервер, де BCrypt.Verify звіряє введене значення з хешем у базі.

При успішній перевірці формується JWT-токен із claims – ідентифікатор, роль і ім'я користувача – і автоматично зберігається у Cookie браузера з обмеженим терміном дії. Активна сесія підтримується без повторного введення облікових даних при переходах між сторінками. Роль визначає подальший маршрут: адміністратор потрапляє в панель керування, звичайний користувач – на головну сторінку магазину. Введення хибного пароля або неіснуючого email не пропускає до системи – відображається повідомлення про помилку з пропозицією повторити спробу.

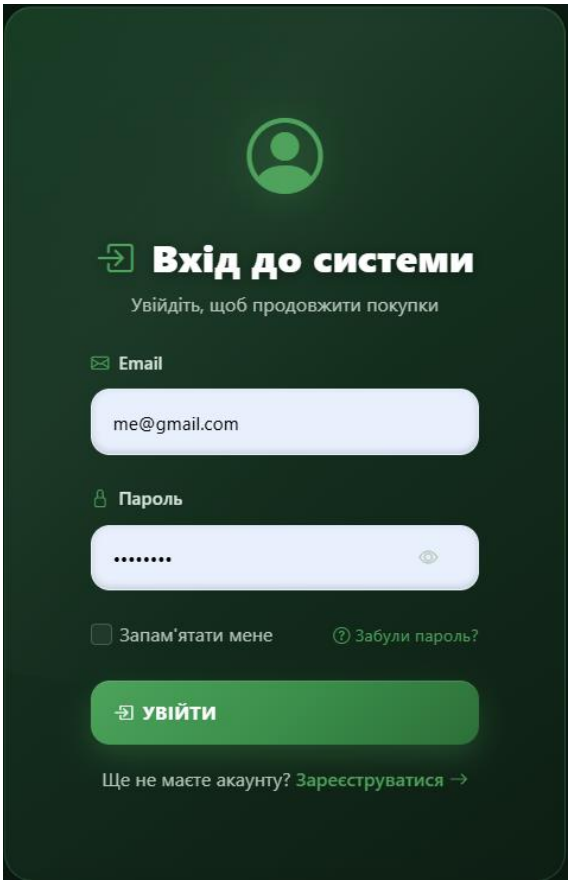

Рисунок 4.2 – Форма входу до облікового запису

Головна сторінка (рис. 4.3)

Перше, що бачить відвідувач, – Hero-секція з акцентним заголовком і двома кнопками: «Переглянути каталог» веде до повного асортименту, «Дізнатися більше» плавно прокручує сторінку до блоку переваг. Той складається з чотирьох карток: доставка по Україні за 1–3 дні, оригінальна продукція від офіційних постачальників, повернення протягом 14 днів і цілодобова підтримка. Нижче через

AJAX-запит до /api/products динамічно підвантажується сітка з восьми популярних товарів. Завершує сторінку горизонтальний рядок категорій із іконками, що також тягнеться з бази даних.

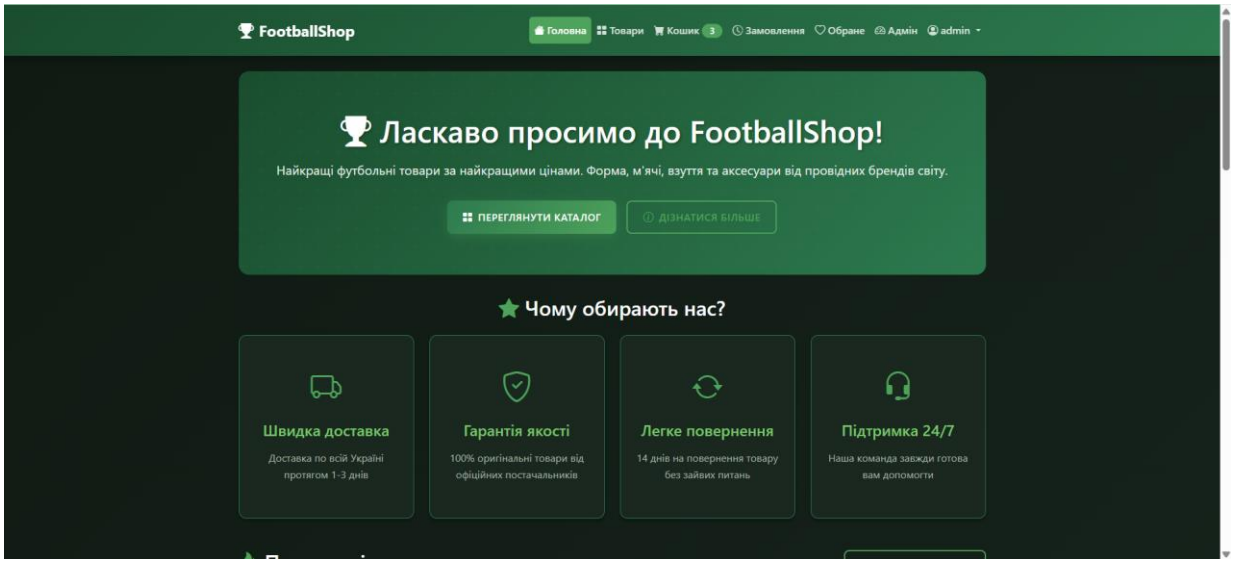

Рисунок 4.3 – Головна сторінка застосунку

Каталог товарів (рис. 4.4)

Пошук і підбір продукції – головне завдання сторінки каталогу. Вгорі – рядок пошуку за назвою. Одразу під ним розгортається панель фільтрів: випадний список категорій, поля цінового діапазону, чекбокс «Тільки в наявності» і список сортування (за ціною зростання/спадання або за назвою). Весь каталог завантажується одним AJAX-запитом у локальний масив, тому фільтрація JavaScript-обробкою відпрацьовує миттєво – без жодного перезавантаження. Кожна картка несе зображення, назву, короткий опис, ціну й індикатор наявності; покупцю доступні кнопка переходу до повної картки товару і кнопка миттєвого додавання в кошик.

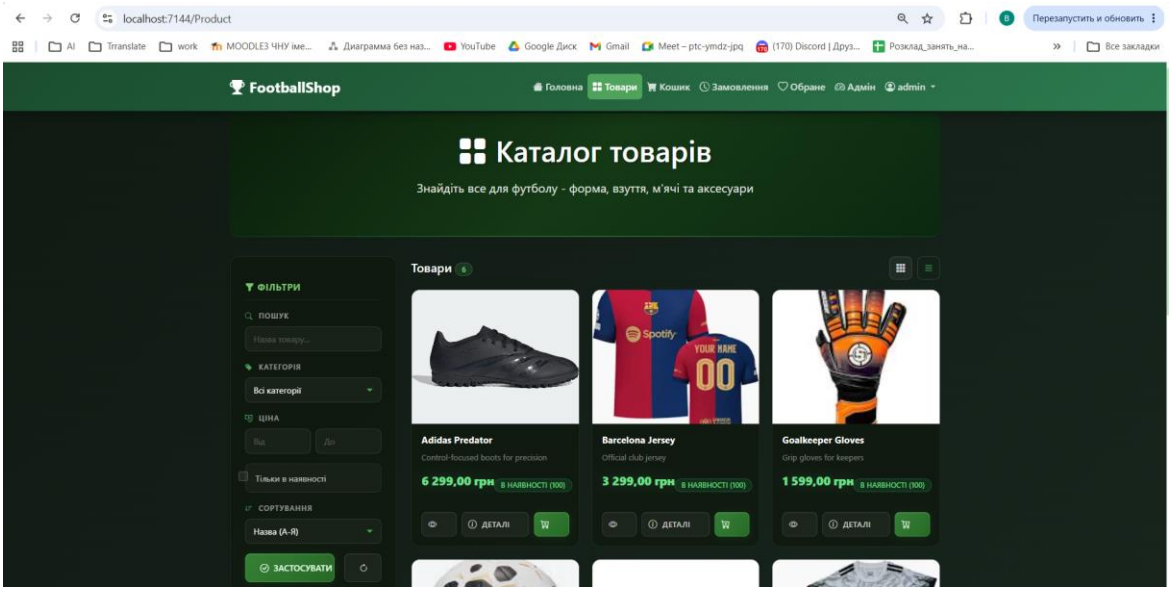

Рисунок 4.4 – Сторінка каталогу товарів з фільтрами

Деталі товару (рис. 4.5)

Розгорнута картка товару дає вичерпну інформацію для прийняття рішення. Галерея зліва – велике головне фото і чотири мініатюри, клік на будь-яку підмінює головне зображення. Права колонка концентрує назву, зірковий рейтинг, ціну і блок вибору розміру: для взуття – US 33–47, для одягу – S/M/L/XL/XXL, для решти категорій вибір розміру не відображається. Якщо розмір обов'язковий, але не обраний, система заблокує додавання в кошик і підсвітить блок червоним із пояснювальним повідомленням. Поруч – кнопки «В обране» (Wishlist) і «Поділитися» для копіювання посилання. Нижня частина сторінки містить секцію коментарів для авторизованих користувачів і добірку схожих товарів тієї самої категорії.

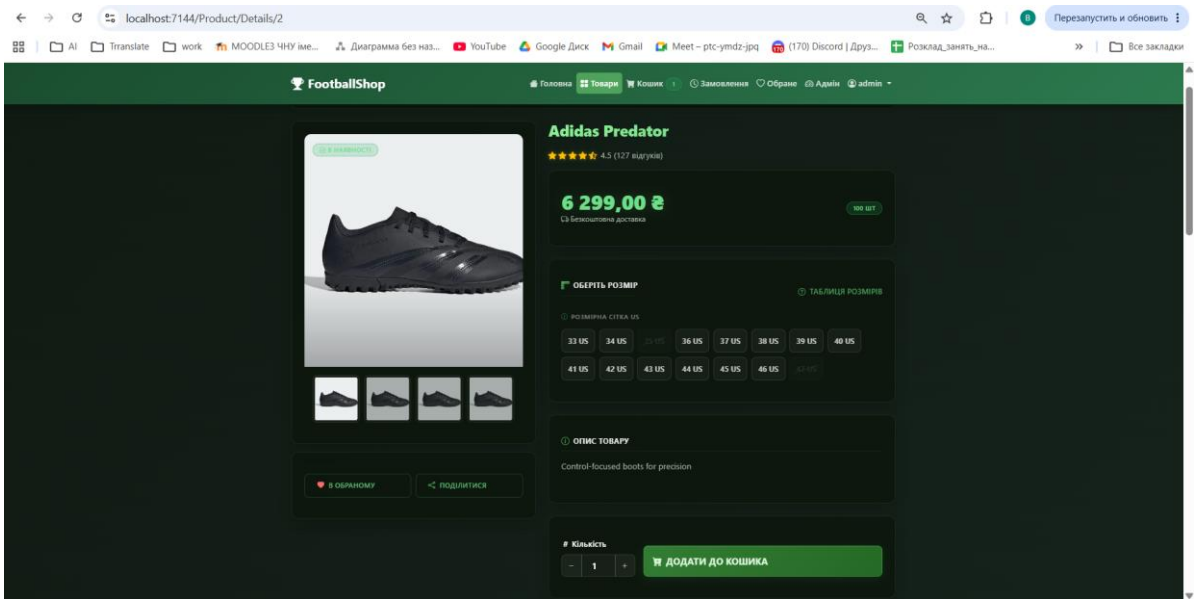

Рисунок 4.5 – Сторінка деталей товару з вибором розміру

Кошик (рис. 4.6)

На сторінці кошика покупець бачить усі відібрані позиції: зображення, назву, лічильник кількості з кнопками «+» і «−», ціну за одиницю і підсумок по рядку. Видалити позицію можна окремою кнопкою. Блок підсумку праворуч показує загальну суму і рядок «Доставка: Безкоштовно». Натискання «Оформити замовлення» створює запис у базі зі статусом Pending і переводить на сторінку деталей замовлення.

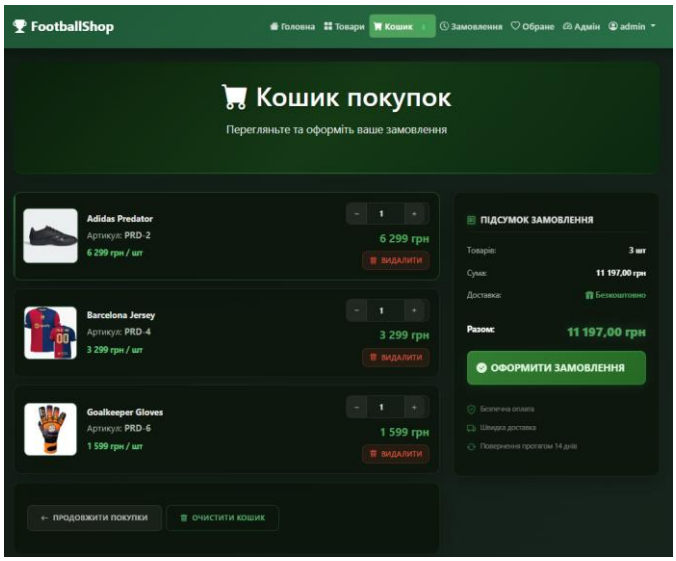

Рисунок 4.6 – Сторінка кошика

Деталі замовлення (рис. 4.7)

Після оформлення покупець потрапляє на сторінку з таймлайном із чотирьох кроків: «Замовлення створено» –> «Оплата» –> «Відправлення» –> «Доставлено». Поточний активний крок підсвічується зеленим відповідно до актуального статусу з бази. Для замовлень у статусі Pending відображається кнопка «Оплатити», що відкриває двокрокове модальне вікно. Перший крок збирає дані доставки – повне ім'я, телефон, email, адресу. Другий – реквізити картки з автоматичним форматуванням номера і дати. Після підтвердження статус переходить у Processing, браузер перенаправляється на сторінку успішної оплати.

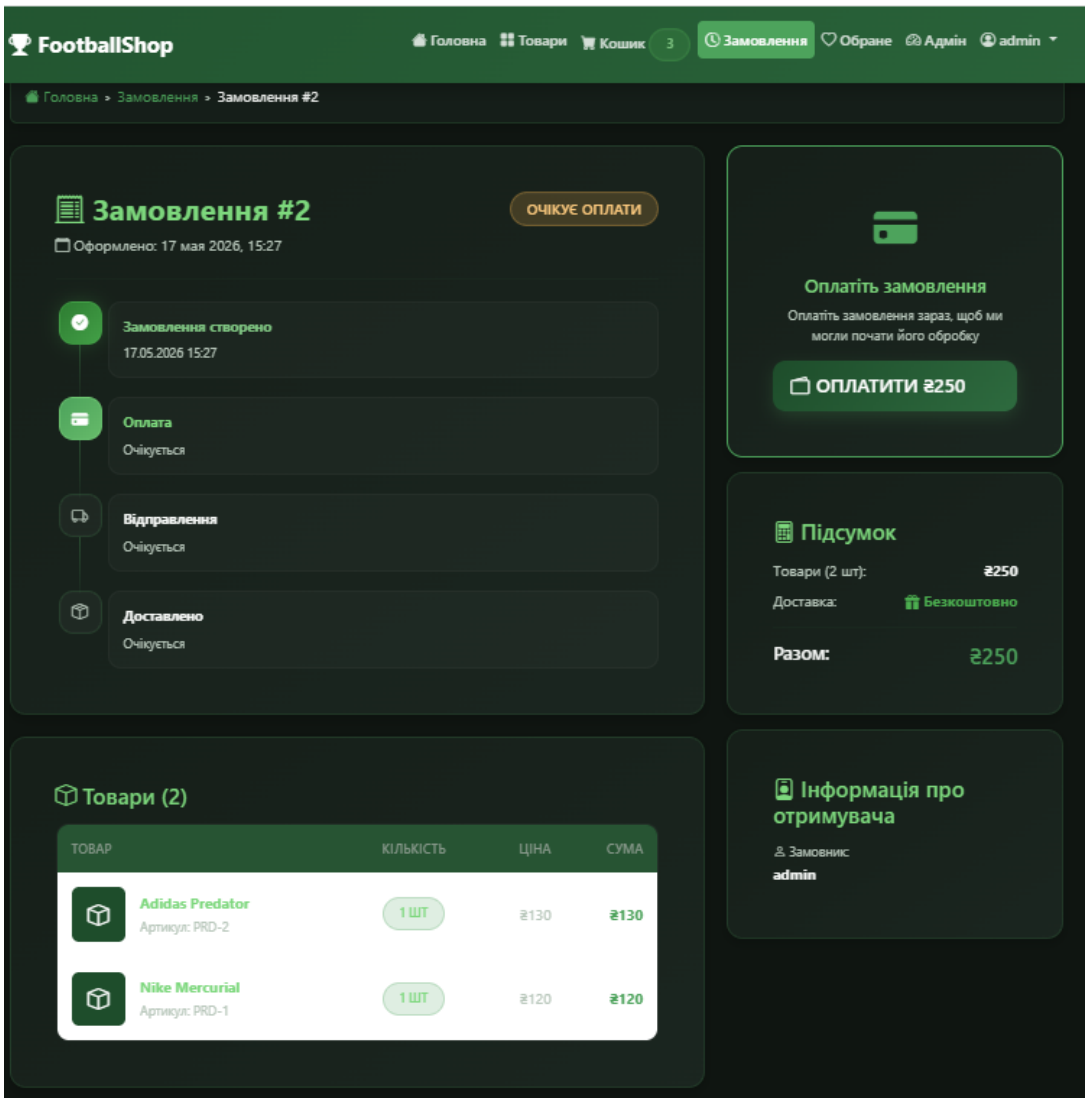

Рисунок 4.7 – Сторінка деталей замовлення з таймлайном статусів

Список обраного (рис. 4.8)

Wishlist відображає збережені товари в адаптивній сітці карток. Кожна картка показує зображення, назву, ціну, дату додавання; у правому верхньому куті – іконка серця для видалення з обраного. Перейти до картки товару або одразу додати його в кошик можна безпосередньо звідси, без зайвих переходів. Вгорі сторінки – лічильник збережених позицій і кнопка «Очистити все». При порожньому списку відображається інформаційний стан із пропозицією повернутися до каталогу.

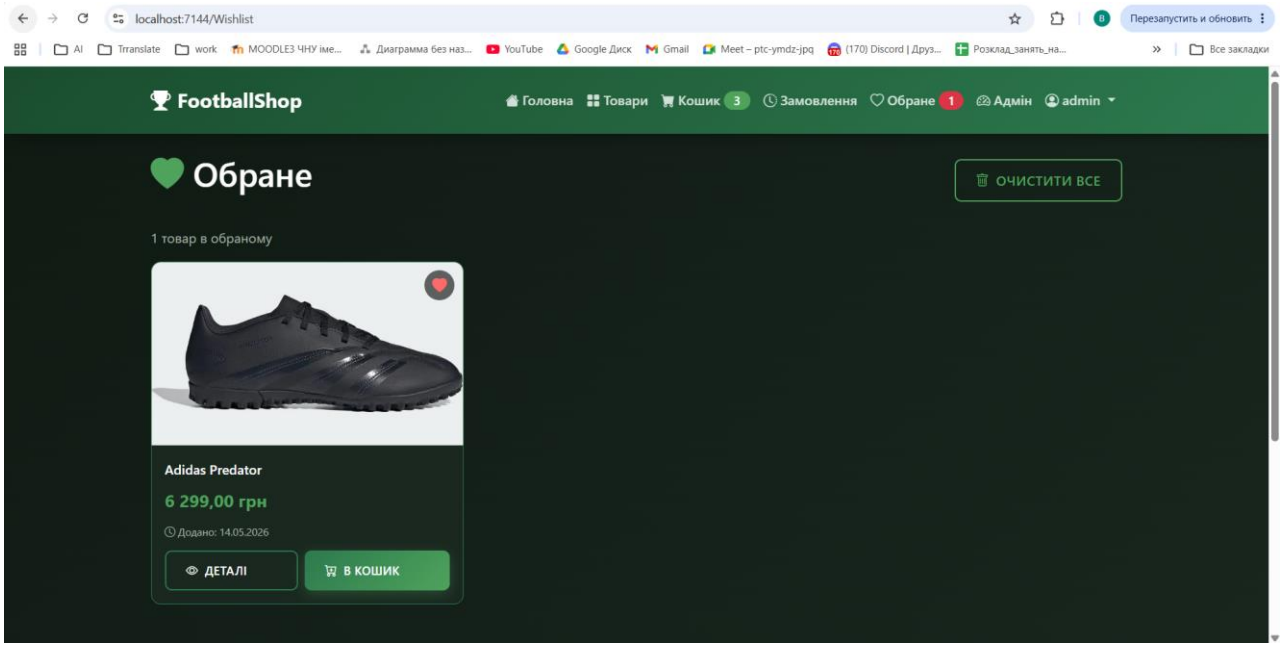

Рисунок 4.8 – Сторінка обраного

Адміністративна панель (рис. 4.9)

Доступ до панелі відкривається виключно для облікових записів з роллю admin. Дашборд зустрічає шістьма статистичними картками: загальний дохід, кількість замовлень, користувачів, товарів, замовлень що очікують оплати і дохід поточного місяця. Нижче – чотири інтерактивні графіки на базі Chart.js: комбінована діаграма продажів по місяцях, кругова за статусами замовлень, горизонтальний стовпчастий рейтинг топ-5 товарів і лінійний графік приросту користувачів. Розділ замовлень дає таблицю з фільтрами за статусом і кнопками швидкого переведення в наступний статус. Розділ товарів дозволяє додавати нові позиції через модальне вікно з полями назви, категорії, ціни, URL зображення і опису, а також редагувати й видаляти наявні.

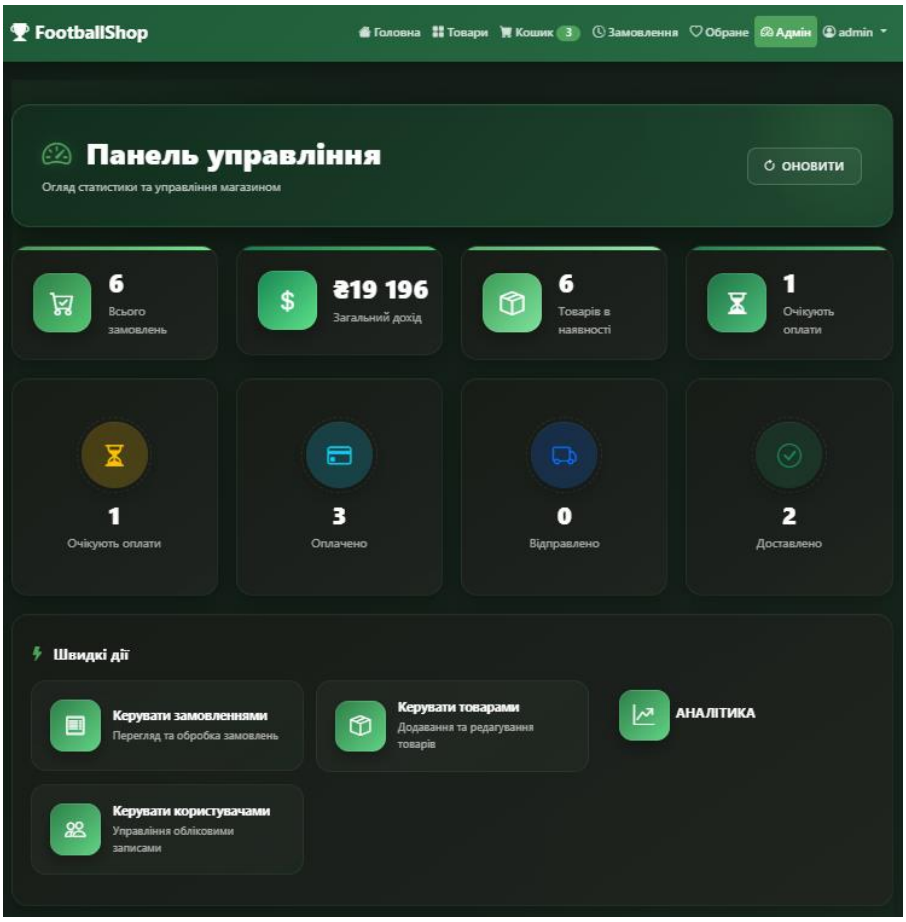

Рисунок 4.9 – Адміністративна панель з аналітичним дашбордом

4.4 Тестування вебзастосунку

Функціональне тестування FootballShop проводилося вручну через браузер і Postman. Для кожного сценарію перевірялися HTTP-код відповіді, структура JSON і відповідність фактичної поведінки системи очікуваній. Захищені endpoints тестувалися після попередньої автентифікації із передачею JWT-токена в заголовку запиту. Зведений перелік сценаріїв наведено в таблиці 4.1.

Таблиця 4.1 – Тест-кейси вебзастосунку FootballShop

№	Назва тесту	Передумови	Кроки тестування	Очікуваний результат	Фактичний результат
1	Реєстрація нового користувача	Обліковий запис у системі не створено	POST /api/account/register з валідними даними	200 OK, токен у Cookie, роль user	Повністю збігається з очікуваним
2	Реєстрація з наявним email	Акаунт із цією адресою вже існує	POST /api/account/register з дублікатом email	400 Bad Request, повідомлення про конфлікт	Результат успішний, дефекти відсутні
3	Вхід із коректними даними	Зареєстрований обліковий запис наявний	POST /api/account/login із правильними обліковими даними	200 OK, JWT-токен, редірект на головну	Поведінка системи коректна
4	Автентифікація за умови некоректного пароля	Зареєстрований обліковий запис наявний	POST /api/account/login із хибним паролем	400 Bad Request, повідомлення про помилку	Очікуваний ефект досягнуто
5	Отримання переліку товарів	БД містить товарні позиції	GET /api/products	200 OK, колекція об'єктів перенесення даних товарів	Повністю збігається з очікуваним
6	Фільтрація каталогу за категорією	БД містить товари різних категорій	GET /api/products?categoryId=1	200 OK, відфільтрована вибірка товарів	Результат успішний, дефекти відсутні
7	Додавання позиції до кошика	Користувач автентифікований	POST /Cart/Add з productId та quantity	200 OK, товар збережено в CartItems	Поведінка системи коректна

Кінець таблиці 4.1

8	Спроба додавання без авторизації	Сесія користувача відсутня	POST /Cart/Add без Cookie-токена	401 Unauthorized, редірект на сторінку входу	Очікуваний ефект досягнуто
9	Оформлення замовлення	Автентифікований користувач, кошик не порожній	POST /Order/Checkout	200 OK, замовлення зі статусом Pending	Повністю збігається з очікуваним
10	Проведення оплати	Замовлення у статусі Pending	POST /api/orders/{id}/pay із даними доставки	200 OK, статус переведено в Processing	Результат успішний, дефекти відсутні
11	Оновлення статусу адміністратором	Автентифікований адмін, замовлення Processing	PUT /api/orders/{id} зі статусом Shipped	200 OK, статус переведено в Shipped	Поведінка системи коректна
12	Додавання товару до обраного	Користувач автентифікований	POST /api/wishlist/{productId}	200 OK, запис збережено в таблиці Wishlists	Очікуваний ефект досягнуто
13	Публікація коментаря	Користувач автентифікований	POST /api/comments із productId та текстом	200 OK, коментар збережено в БД	Повністю збігається з очікуваним
14	Запит аналітики без токена	JWT-токен відсутній	GET /api/analytics/summary без заголовка авторизації	401 Unauthorized	Результат успішний, дефекти відсутні
15	Адміністративна дія з токеном рядового користувача	JWT-токен містить роль user	PUT /api/orders/{id} із токеном user	403 Forbidden	Поведінка системи коректна

Усі п'ятнадцять перевірених сценаріїв завершилися успішно. Система коректно відпрацьовує сценарії успішної реєстрації, входу, оформлення замовлення, оплати й управління обраним. Не менш важливо, що негативні сценарії – реєстрація з дублікатом email, хибний пароль, звернення до захищених endpoints без токена і спроба виконати адміністративну дію з роллю user – також повертають

очікувані коди та повідомлення. Механізм JWT у поєднанні з Cookie і розмежуванням за ролями RBAC функціонує відповідно до вимог специфікації.

Висновки до розділу 4

Завершальний етап розробки FootballShop дозволив успішно трансформувати архітектурні моделі у працездатний програмний продукт.

Організація вихідного коду підпорядкована стандартним угодам ASP.NET Core MVC: контролери розділені на MVC і API-підшар, бізнес-правила ізольовані в сервісах із чітким розмежуванням інтерфейсів і реалізацій, Entity-моделі зосереджені у директорії Models, а міграції EF Core версіонують схему SQLite-бази поряд із вихідним кодом.

Чотири ключові компоненти описані з фрагментами коду. ApplicationDbContext декларує десять DbSet-властивостей і через OnModelCreating налаштовує унікальні індекси, типи числових полів і поведінку зовнішніх ключів. OrderService забезпечує атомарне оформлення замовлення: зчитує кошик, фіксує ціни в OrderItems і очищає CartItems в межах одного виклику SaveChangesAsync. AccountController реалізує реєстрацію з BCrypt-хешуванням і генерацію JWT-токенів, підписаних алгоритмом HMAC-SHA256. ProductApiController і AnalyticsApiController надають захищені REST-endpoints із динамічною LINQ-фільтрацією і агрегацією даних для Chart.js-дашборду.

Керівництво користувача охоплює опис базових сторінок застосунку: реєстрацію й авторизацію, головну сторінку, каталог із миттєвою фільтрацією, картку товару з вибором розміру, кошик, сторінку деталей замовлення з таймлайном статусів, список обраного і адміністративну панель з аналітичними графіками.

ВИСНОВКИ

Результатом виконаної кваліфікаційної бакалаврської роботи стало проєктування повноцінного вебзастосунку FootballShop – спеціалізованого інтернет-магазину футбольної екіпіровки, що охоплює весь ланцюжок онлайн-торгівлі: від перегляду каталогу й підбору розміру до оплати замовлення й аналітики продажів у адмінській панелі.

Аналітичний огляд ринку електронної комерції у сфері спортивних товарів дозволив виявити слабкі місця існуючих платформ. Дослідження лідерів вітчизняного e-commerce та спеціалізованих майданчиків – великих мереж спортивного ритейлу ROZETKA, INTERSPORT, MEGASPORT, Sportmaster і профільного магазину 4Football – показало спільні вади: візуальна перевантаженість інтерфейсів рекламними блоками, заплутана навігація через надмірну категоризацію, повільна фільтрація та незадовільна адаптація під мобільні пристрої. Саме ці спостереження склали основу функціональних і нефункціональних вимог до FootballShop.

Проєктування базувалося на принципах чіткого розмежування відповідальності. Технологічний стек підбирався свідомо, а не за шаблоном. Серверну частину побудовано на ASP.NET Core 8 і C# 12, що дало змогу поєднати строгу типізацію, асинхронне програмування через async/await і продуктивний рендеринг Razor Pages. Для роботи з даними обрано Entity Framework Core 8.0 з провайдером SQLite – легкою вбудованою реляційною СУБД, що зберігає базу в єдиному файлі, не вимагає окремого серверного процесу і суттєво спрощує розгортання. Клієнтський рівень сформований зв'язкою HTML5/CSS3, Bootstrap 5.3 і JavaScript ES6+ з jQuery, що забезпечило адаптивний інтерфейс у діапазоні від 320px до 1920px. Архітектурні патерни MVC, Repository і Dependency Injection забезпечили слабку зв'язаність компонентів і чисту структуру коду. Безпеку системи закрито комбінацією JWT-автентифікації для REST API і Cookie – для MVC-сторінок; паролі зберігаються виключно у вигляді BCrypt-хешів.

Моделювання системи охопило функціональні діаграми IDEF0 трьох рівнів декомпозиції та повний комплекс UML-діаграм: варіантів використання, послідовності, діяльності й класів. Схема реляційної бази даних складається з дев'яти взаємопов'язаних таблиць; зв'язки між сутностями і обмеження налаштовано через Fluent API Entity Framework Core. Ключове архітектурне рішення – фіксація ціни товару безпосередньо в OrderItems на момент оформлення – захищає цілісність історичних даних від будь-яких подальших коригувань прайсу.

Програмна реалізація включає повноцінний REST API для керування товарами, категоріями, кошиком, замовленнями і користувачами. Реалізовано реєстрацію й авторизацію з рольовою моделлю доступу (RBAC), пошук і миттєву клієнтську фільтрацію каталогу, двокроковий флоу оплати через модальне вікно, таймлайн статусів замовлення і адміністративний дашборд з інтерактивними графіками на Chart.js.

Функціональне тестування п'ятнадцяти сценаріїв через браузер і Postman підтвердило: система коректно опрацьовує як штатні шляхи – реєстрація, вхід, оформлення і оплата замовлення – так і граничні випадки: дублікат email, хибний пароль, звернення до захищених endpoints без токена і спроба виконати адміністративну дію з роллю user. Усі 15 перевірок завершено без зауважень; патерни Repository і Dependency Injection виправдали себе на практиці, спрощуючи ізоляцію компонентів під час тестування.

Мету кваліфікаційної роботи досягнуто – FootballShop є працездатним і технічно обґрунтованим програмним рішенням, що відповідає сучасним вимогам до вебсистем електронної комерції. Перспективи розвитку проєкту – інтеграція реальних платіжних сервісів, підключення API служб доставки, механізм персоналізованих рекомендацій на основі історії покупок і розширення аналітичного модуля.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Bucko A., Vishi K., Krasniqi B. Enhancing JWT authentication and authorization in web applications based on user behavior history. Computers. 2023. Vol. 12, № 4. P. 78–92.

2. Hakiki R., Delianti V. I., Marta R. Smart financial management for cooperatives: a web and payment gateway integration approach. Journal of Hypermedia&Technology-Enhanced Learning. 2024. Vol. 3, № 1. P. 16–36.

3. Sehgal R., Mehrotra V. A comparative study of REST and GraphQL APIs for web application development. International Journal of Advanced Computer Science and Applications (IJACSA). 2022. Vol. 13, № 4. P. 310–317.

4. Alshammari M., Alqahtani A. Security analysis of RESTful APIs in modern web applications. IEEE Access. 2023. Vol. 11. P. 45231–45244.

5. Strauss D. Creating ASP.NET Core web applications. Cham : Springer, 2021. 372 p.

6. Hoffman A. Web application security. Sebastopol : O’Reilly Media Inc., 2024. 350 p.

7. Frain B. Responsive web design with HTML5 and CSS: build future-proof responsive websites using the latest HTML5 and CSS techniques. Birmingham : Packt Publishing Ltd, 2025. 500 p.

8. Smith J. P. Entity Framework Core in Action. New York : Simon and Schuster, 2021. 528 p.

9. Freeman A. Pro ASP.NET Core 7. Berkeley : Apress, 2023. 1018 p.

10. Ranjan A., Sinha A., Battewad R. JavaScript for modern web development: building a web application using HTML, CSS, and JavaScript. New Delhi : BPB Publications, 2020. 356 p.

11. Ullah S. E., Alauddin T., Zaman H. U. Developing an e-commerce website. 2016 International Conference on Microelectronics, Computing and Communications (MicroCom). Durgapur, 2016. P. 1–4.

12. Ritonummi S., Niininen O. User experience of an e-commerce website: a case study. Contemporary Issues in Digital Marketing. London : Routledge, 2021. P. 61–71.

13. Microsoft. ASP.NET Core documentation – URL: https://docs.microsoft.com/aspnet/core (Accessed: 10.04.2025).

14. Microsoft. Entity Framework Core documentation – URL: https://docs.microsoft.com/ef/core (Accessed: 10.04.2025).

15. Bootstrap. Bootstrap 5 official documentation – URL: https://getbootstrap.com/docs/5.3 (Accessed: 10.04.2025).

16. SQLite. SQLite documentation – URL: https://www.sqlite.org/docs.html (Accessed: 10.04.2025).

17. OWASP. OWASP top ten web application security risks – URL: https://owasp.org/www-project-top-ten (Accessed: 10.04.2025).

18. Statista. Global e-commerce market size 2023 – URL: https://www.statista.com/statistics/379046/worldwide-retail-e-commerce-sales (Accessed: 10.04.2025).

19. Chart.js. Chart.js documentation – URL: https://www.chartjs.org/docs/latest (Accessed: 10.04.2025).

20. JWT.io. Introduction to JSON web tokens – URL: https://jwt.io/introduction (Accessed: 10.04.2025).

21. Кондіус О. Методичні рекомендації з функціонального моделювання IDEF0 – URL: https://elib.lntu.edu.ua (дата звернення: 10.04.2025).

ДОДАТОК Лістинг коду

Лістинг коду Product.cs :

```
using System;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace FootballShop.Models
{
    public class Product
    {
        [Key]
        public int Id { get; set; }

        [Required]
        [ForeignKey(nameof(Category))]
        public int CategoryId { get; set; }

        [Required, MaxLength(200)]
        public string Name { get; set; } = null!;

        public string? Description { get; set; }

        [Column(TypeName = "decimal(18,2)")]
        public decimal Price { get; set; }

        public string? ImageUrl { get; set; }

        public int Stock { get; set; } = 0;

        public DateTime CreatedAt { get; set; } = DateTime.UtcNow;

        // Навігаційні властивості
        public virtual Category? Category { get; set; }
    }
}
```

Лістинг коду Product_DTOs.cs :

```
// DTO/Products/CreateProductDto.cs
namespace FootballShop.DTO.Products
{
    public class CreateProductDto
    {
        public string Name { get; set; } = string.Empty;
        public string Description { get; set; } = string.Empty;
        public decimal Price { get; set; }
        public string ImageUrl { get; set; } = string.Empty;
        public int CategoryId { get; set; }
    }
    public class ProductDto
    {
        public int Id { get; set; }
        public string Name { get; set; } = string.Empty;
        public string Description { get; set; } = string.Empty;
        public decimal Price { get; set; }
        public string ImageUrl { get; set; } = string.Empty;
```

```
        public int CategoryId { get; set; }
        public string CategoryName { get; set; } = string.Empty;
        public int Stock { get; set; }
        public DateTime CreatedAt { get; set; }
    }
    public class UpdateProductDto
    {
        public int Id { get; set; }
        public string? Name { get; set; }
        public string? Description { get; set; }
        public decimal? Price { get; set; }
        public string? ImageUrl { get; set; }
        public int? CategoryId { get; set; }
    }
}
```

Лістинг коду ProductController.cs :

```
using Microsoft.AspNetCore.Mvc;
using FootballShop.Services.Interfaces;

namespace FootballShop.Controllers
{
    public class ProductController : Controller
    {
        private readonly IProductService _productService;

        public ProductController(IProductService productService)
        {
            _productService = productService;
        }

        public async Task<IActionResult>Index()
        {
            var products = await _productService.GetAllAsync();
            return View(products);
        }

        public async Task<IActionResult>Details(int id)
        {
            var product = await _productService.GetByIdAsync(id);
            if (product == null)
                return NotFound();

            return View(product);
        }
    }
}
```

Лістинг коду ProductApiController.cs :

```
using Microsoft.AspNetCore.Mvc;
using FootballShop.Services.Interfaces;
using FootballShop.DTO.Products;

namespace FootballShop.Controllers.Api
{
    [Route("api/products")]
```

```
    [ApiController]
    public class ProductApiController : ControllerBase
    {
        private readonly IProductService _productService;

        public ProductApiController(IProductService productService)
        {
            _productService = productService;
        }

        [HttpGet]
        public async Task<IActionResult> GetAll() =>
            Ok(await _productService.GetAllAsync());

        [HttpGet("{id}")]
        public async Task<IActionResult>Get(int id)
        {
            var product = await _productService.GetByIdAsync(id);
            return product == null ? NotFound() : Ok(product);
        }

        [HttpPost]
        public async Task<IActionResult>Create(CreateProductDto dto)
        {
            var created = await _productService.CreateAsync(dto);
            return CreatedAtAction(nameof(Get), new { id = created.Id }, created);
        }

        [HttpPut("{id}")]
        public async Task<IActionResult> Update(int id, UpdateProductDto dto)
        {
            dto.Id = id;
            var updated = await _productService.UpdateAsync(dto);
            return updated == null ? NotFound() : Ok(updated);
        }

        [HttpDelete("{id}")]
        public async Task<IActionResult>Delete(int id)
        {
            var deleted = await _productService.DeleteAsync(id);
            return deleted ? NoContent() : NotFound();
        }
    }
}
```

Лістинг коду IProductService.cs :

```
// Services/Interfaces/IProductService.cs
using FootballShop.DTO.Products;

namespace FootballShop.Services.Interfaces
{
    public interface IProductService
    {
        Task<IEnumerable<ProductDto>> GetAllAsync();
        Task<ProductDto?> GetByIdAsync(int id);
        Task<ProductDto> CreateAsync(CreateProductDto dto);
        Task<ProductDto?> UpdateAsync(UpdateProductDto dto);
        Task<bool> DeleteAsync(int id);
    }
}
```

Лістинг коду ProductService.cs :

```
// Services/ProductService.cs
using FootballShop.DTO.Products;
using FootballShop.Models;
using FootballShop.Services.Interfaces;
using Microsoft.EntityFrameworkCore;

namespace FootballShop.Services
{
    public class ProductService : IProductService
    {
        private readonly ApplicationDbContext _context;

        public ProductService(ApplicationDbContext context)
        {
            _context = context;
        }

        public async Task<IEnumerable<ProductDto>> GetAllAsync()
        {
            return await _context.Products
                .Include(p => p.Category)
                .Select(p => new ProductDto
                {
                    Id = p.Id,
                    Name = p.Name,
                    Description = p.Description,
                    Price = p.Price,
                    ImageUrl = p.ImageUrl,
                    CategoryId = p.CategoryId,
                    CategoryName = p.Category.Name,
                    Stock = p.Stock,
                    CreatedAt = p.CreatedAt
                }).ToListAsync();
        }

        public async Task<ProductDto?> GetByIdAsync(int id)
        {
            var p = await _context.Products.Include(p => p.Category).FirstOrDefaultAsync(p
=> p.Id == id);
            if (p == null) return null;

            return new ProductDto
            {
                Id = p.Id,
                Name = p.Name,
                Description = p.Description,
                Price = p.Price,
                ImageUrl = p.ImageUrl,
                CategoryId = p.CategoryId,
                CategoryName = p.Category.Name,
                Stock = p.Stock,
                CreatedAt = p.CreatedAt
            };
        }

        public async Task<ProductDto> CreateAsync(CreateProductDto dto)
        {
```

```
            var product = new Product
            {
                Name = dto.Name,
                Description = dto.Description,
                Price = dto.Price,
                ImageUrl = dto.ImageUrl,
                CategoryId = dto.CategoryId
            };
            _context.Products.Add(product);
            await _context.SaveChangesAsync();

            return await GetByIdAsync(product.Id);
        }

        public async Task<ProductDto?> UpdateAsync(UpdateProductDto dto)
        {
            var product = await _context.Products.FindAsync(dto.Id);
            if (product == null) return null;

            if (dto.Name != null) product.Name = dto.Name;
            if (dto.Description != null) product.Description = dto.Description;
            if (dto.Price.HasValue) product.Price = dto.Price.Value;
            if (dto.ImageUrl != null) product.ImageUrl = dto.ImageUrl;
            if (dto.CategoryId.HasValue) product.CategoryId = dto.CategoryId.Value;

            await _context.SaveChangesAsync();
            return await GetByIdAsync(product.Id);
        }

        public async Task<bool> DeleteAsync(int id)
        {
            var p = await _context.Products.FindAsync(id);
            if (p == null) return false;

            _context.Products.Remove(p);
            await _context.SaveChangesAsync();
            return true;
        }
    }
}
```