

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій
(повне найменування факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
ДЛЯ МОДЕЛЮВАННЯ, НАЛАШТУВАННЯ ТА ЕКСПЛУАТАЦІЇ БІЗНЕС-
СИСТЕМИ
RESEARCH AND DEVELOPMENT OF SOFTWARE FOR MODELLING,
CONFIGURATION, AND OPERATION OF A BUSINESS SYSTEM

Виконав(ла): студент(ка) 2 курсу, групи КНТ-114м
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)
Інженерія програмного забезпечення

ШИХАДЕ М.А.

(ПРІЗВИЩЕ та ініціали)

Керівник КОЛПАКОВА Т.О.

(ПРІЗВИЩЕ та ініціали)

Рецензент ІЛЬЯШЕНКО С.М.

(ПРІЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
“ ” 2025 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ШИХАДЕ Михайла Асимовича

(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та розробка програмного забезпечення для моделювання, налаштування та експлуатації бізнес-системи.
Research and Development of Software for Modelling, Configuration, and Operation of a Business System

керівник проєкту (роботи) к.т.н., доцент, КОЛПАКОВА Тетяна Олексіївна,
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “30” вересня 2025 року № 447

2. Строк подання студентом проєкту (роботи) 08 грудня 2025 року
3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми та постановка завдань дослідження.
2. Матеріали і методи дослідження. 3. Проєктування програмного забезпечення.
4. Розробка програмного забезпечення. 5. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	КОЛПАКОВА Т.О., доцент		
Нормоконтролер	КАЛІНІНА М.В., асистент		

7. Дата видачі завдання “30” вересня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	2-3 тижні	Розділ 1
3	Розробка та удосконалення методів, моделей й алгоритмів вирішення задачі.	4-5 тижні	Розділ 2
4	Розробка архітектури програми.	6 тиждень	Розділ 3
5	Розробка програми.	7-8 тижні	Розділ 4
6	Тестування та експериментальне дослідження програмного забезпечення.	9 тиждень	Розділ 5
7	Оформлення пояснювальної записки та документів до неї.	10-11 тижні	Додатки
8	Нормоконтроль та рецензування.	12 тиждень	
9	Захист роботи.	12 тиждень	

Студент(ка)

(підпис) Михайло ШИХАДЕ
(Імя ПРІЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Тетяна КОЛПАКОВА
(Імя ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
82 с., 6 табл., 21 рис., 2 дод., 23 джерела.

ІНФОРМАЦІЙНА СИСТЕМА, БІЗНЕС-СИСТЕМА,
АВТОМАТИЗАЦІЯ ПРОЄКТУВАННЯ, ШАБЛОНІЗАЦІЯ ВЕБСИСТЕМ,
ГЕНЕРАТОР ВЕБЗАСТОСУНКІВ.

Об'єкт дослідження – процес інженерії програмного забезпечення для моделювання, налаштування та експлуатації веборієнтованих бізнес-систем.

Предмет дослідження – методи та програмні засоби автоматизованого моделювання та генерації структури вебсистем на основі пресетів та шаблонів.

Мета роботи – підвищення економічної ефективності веборієнтованих бізнес-систем за рахунок зменшення часових і фінансових витрат на їх проєктування та запуск.

Матеріали, методи та технічні засоби: компонентно-орієнтоване та об'єктно-орієнтоване програмування, шаблонне проєктування, платформа .NET, мова програмування C#, система керування БД, персональний комп'ютер із процесором Intel Core i5-11400H під управлінням операційної системи Windows 10 Pro.

Результати. Створено програмне забезпечення для моделювання, налаштування та експлуатації бізнес-систем, яке на основі конфігурованого пресета бізнесу автоматично формує структуру трьохкомпонентної вебсистеми.

Висновки. Розроблено програмне забезпечення, яке оптимізує процеси моделювання, налаштування та первинного розгортання бізнес-систем, що скорочує витрати ресурсів та підвищує економічну ефективність.

Галузь використання – проєктування та розгортання веборієнтованих інформаційних систем для малого й середнього бізнесу.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 82 pages, 6 tables, 21 figures, 2 appendices, 23 sources.

INFORMATION SYSTEM, BUSINESS SYSTEM, DESIGN AUTOMATION, WEBSYSTEM TEMPLATE, WEB APPLICATION GENERATOR.

The object of the study is the software engineering process for modelling, configuring and operating web-oriented business systems.

The subject of the study is methods and software tools for automated modelling and generation of the structure of web systems based on presets and templates.

The aim of the work is to increase the economic efficiency of web-oriented business systems by reducing the time and financial costs of their design and launch.

Materials, methods and technical means: component-oriented and object-oriented programming, template-based design, .NET platform, C# programming language, database management system, personal computer with Intel Core i5-11400H processor running Windows 10 Pro.

Results. Software has been developed for modelling, configuring and operating business systems which, based on a configurable business preset, automatically forms the structure of a three-component web system.

Conclusions. Software has been developed that optimizes the processes of modelling, configuring and initial deployment of business systems, which reduces resource costs and increases economic efficiency.

Field of application – design and deployment of web-oriented information systems for small and medium-sized businesses.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	8
Вступ.....	9
1 Аналіз проблеми та постановка завдань дослідження	10
1.1 Роль веборієнтованих бізнес-систем у розвитку малого бізнесу.....	10
1.2 Потреби малого бізнесу у власному вебресурсі. Визначення моделі типової вебсистеми	13
1.3 Економічні та ресурсні бар'єри проєктування веборієнтованих бізнес- систем	16
1.4 Аналіз наявних підходів і програмних рішень	18
1.5 Специфікація вимог до програмного забезпечення	21
1.6 Висновки за розділом	23
2 Матеріали і методи дослідження.....	24
2.1 Загальна методологія дослідження	24
2.2 Вихідні дані та обґрунтування вибору тестового пресета.....	26
2.3 Програмні та технологічні засоби	27
2.4 Методи моделювання пресетів та формування структури вебсистеми	28
2.5 Методи оцінювання ефективності та постановка експерименту	29
2.6 Висновки за розділом	30
3 Проєктування програмного забезпечення	31
3.1 Загальна архітектура програмного забезпечення	31
3.2 Функціональні вимоги та сценарії використання.....	32
3.3 Модель даних і структура пресета	35
3.4 Архітектура генератора	37
3.5 Архітектура згенерованої веборієнтованої системи	38
3.6 Висновки за розділом	41
4 Розробка програмного забезпечення.....	42
4.1 Послідовність реалізації програмного забезпечення	42
4.2 Деталізована реалізація вебсистеми.....	43

4.3 Реалізація пресета та карти модулів.....	46
4.4 Реалізація застосунку-генератора DeepCore	47
4.5 Висновки за розділом	50
5 Експлуатація, тестування та експериментальне дослідження програми	51
5.1 Експлуатація програмного комплексу	51
5.2 Тестування програмного забезпечення.....	54
5.3 Експериментальне дослідження. Отримання результатів, аналіз ефективності	58
5.4 Висновки за розділом	61
Висновки	62
Перелік джерел посилань	63
Додаток А Текст програми.....	66
Додаток Б Слайди презентації.....	75

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

- CMS – Content Management System;
- CRM – Customer Relationship Management;
- CRUD – Create, Read, Update, Delete;
- CSS – Cascading Style Sheets;
- EFCore – Entity Framework Core;
- HTML – HyperText Markup Language;
- HTTP – HyperText Transfer Protocol;
- JSON – JavaScript Object Notation;
- MVC – Model–View–Controller;
- Razor – Razor-подання (технологія шаблонів ASP.NET);
- SaaS – Software as a Service;
- SKU – Stock Keeping Unit;
- slug – символічний ідентифікатор URL;
- БД – база даних;
- ВВП – валовий внутрішній продукт;
- ЄС – Європейський Союз;
- МСБ – малий і середній бізнес;
- МСП – малі та середні підприємства;
- ПЗ – програмне забезпечення.

ВСТУП

Для суб'єктів малого бізнесу наявність власного вебресурсу є однією з ключових передумов виходу на ринок і забезпечення стабільної взаємодії з клієнтами. Саме через вебсистему потенційний споживач отримує інформацію про компанію, її послуги та умови співпраці, здійснює первинний контакт та порівняння пропозицій. Відсутність повноцінного вебресурсу або його формальне, технічно чи структурно обмежене представлення істотно знижує конкурентні можливості малого бізнесу, навіть за наявності якісного товару чи послуги. Спрощення доступу до таких веборієнтованих інформаційних систем здатне послабити бар'єри входу для нових підприємницьких ініціатив, збільшити частку малого бізнесу та позитивно вплинути на загальний економічний фон.

Разом з тим створення веборієнтованої бізнес-системи для невеликих підприємств потребує значних часових і фінансових витрат. До складу такої системи зазвичай входять публічний вебресурс, база даних з відповідною структурою та внутрішні інтерфейси для наповнення і супроводу даних, а також робочі засоби для персоналу. У більшості випадків для кожного нового проєкту ці компоненти проєктуються та реалізуються окремо, що призводить до дублювання зусиль розробників, збільшення вартості кожного впровадження та збереження високого порогу входу для малих підприємств.

Аналіз практики розгортання інформаційних систем свідчить, що потреби значної частини бізнесів можуть бути представлені у вигляді типових конфігурацій, які відрізняються деталями, але зберігають загальну структурну модель, що дозволяє розглядати вебсистему як результат формалізованого пресета, реалізованого за допомогою шаблонів.

Використання спеціалізованого програмного забезпечення, здатного автоматично формувати структуру вебсистеми з формального опису, уможливорює значне скорочення часу й фінансових витрат на проєктування та розгортання, підвищення узгодженості компонентів і, як наслідок, підвищення економічної ефективності впровадження [1].

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Роль веборієнтованих бізнес-систем у розвитку малого бізнесу

1.1.1 Економічна роль малого бізнесу

Малий і середній бізнес (МСБ) є одним із ключових чинників економічного розвитку: у країнах Європейського Союзу на МСБ припадає понад 99% підприємств, близько двох третин зайнятості та понад половину створюваної доданої вартості [2]. Тобто саме від того, наскільки легко малим підприємствам виходити на ринок і масштабуватися, суттєво залежать зайнятість населення, податкові надходження та стійкість економіки загалом. Детальнішу інформацію подано в табл. 1.1.

Однією з базових передумов такого виходу на сьогодні є наявність власного вебресурсу, через який бізнес презентує себе, комунікує з клієнтами та надає частину послуг у цифровому форматі.

Таблиця 1.1 – Економічна роль МСП в Україні та ЄС

Показник	Україна	ЄС
Частка підприємств, що належать до МСП, % від загальної кількості	99,98	99,8
Частка зайнятих у МСП, % від загальної зайнятості	74	65
Частка доданої вартості МСП, % від загальної доданої вартості / ВВП	62	53

1.1.2 Веборієнтовані бізнес-системи як інфраструктура виходу на ринок

Статистичні дослідження показують, що близько 70–73% малих підприємств у розвинених країнах вже мають власний вебресурс, тоді як приблизно чверть–третина суб'єктів малого бізнесу все ще ведуть свою діяльність без нього [3]. Детальніша інформація подана у табл. 1.2.

Серед основних причин відсутності вебсистеми підприємці називають надмірну вартість її створення та нестачу фахових знань щодо організації онлайн-присутності [3]-[4].

Таблиця 1.2 – Поширеність вебприсутності та онлайн-продажів

Показник	Регіон / сукупність	Рік	Значення, %
Частка малих підприємств, що мають власний вебресурс	Світ / основні ринки	2023	71
Частка малих підприємств, що мають власний вебресурс	США	2023	73
Частка підприємств, що здійснювали онлайн-продажі	ЄС	2023	23,8
Частка населення (16–74 років), яке купувало онлайн	ЄС	2023	58,1

Таким чином, значна частина потенційно активних учасників ринку опиняється в гірших умовах конкуренції лише через те, що доступ до цифрової інфраструктури залишається ресурсно затратним.

1.1.3 Вплив вебприсутності на ефективність МСБ

Разом із тим наукові й прикладні дослідження підтверджують, що наявність усталеної онлайн-присутності безпосередньо корелює з кращими фінансовими показниками малого бізнесу. Зокрема, у роботі Гарвардської бізнес-школи наочно демонструється (рис. 1.1), як поява бізнесу в онлайн-каталогах і сервісах відгуків приводить до зростання виручки в середньому

на 5%, причому ефект має причинно-наслідковий характер [5]. Інші опитування демонструють, що підприємства з розвинутою вебприсутністю частіше декларують упевненість у поверненні маркетингових інвестицій і мають більшу частку доходів від нових клієнтів [6].

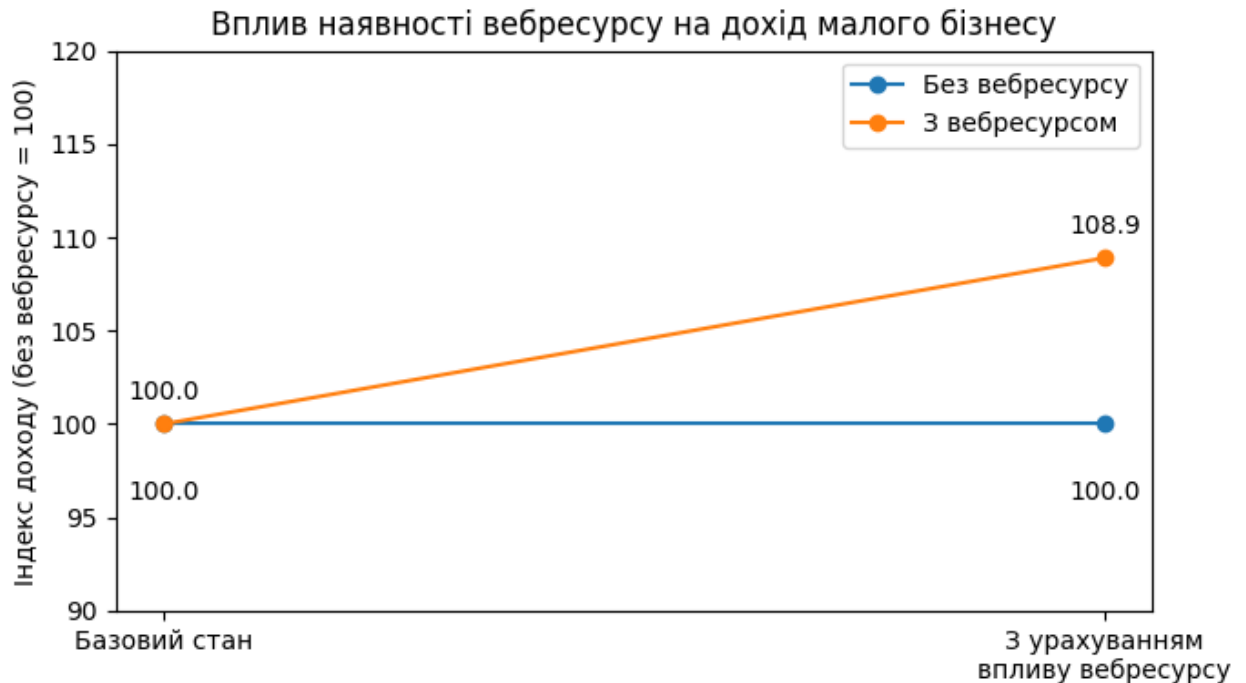


Рисунок 1.1 – Кореляція між доходом бізнесу та наявністю вебресурсу

Загалом для економіки це означає: кожний додатковий малий бізнес, який отримав повноцінну вебсистему, не лише збільшує власний дохід, а й робить свій внесок у зростання зайнятості та податкової бази.

1.1.4 Узагальнення й формулювання проблеми

Незважаючи на очевидні переваги, створення веборієнтованої бізнес-системи залишається суттєвим фінансовим та часовим бар'єром. За оцінками галузових оглядів, розробка типової вебсистеми для малого бізнесу коштує від кількох сотень до кількох тисяч доларів; додатково щомісячні витрати на хостинг, домен, підтримку та доопрацювання становлять сотні доларів [7].

Для багатьох малих підприємств такі витрати порівнянні з річним чистим прибутком, що робить запуск власної вебсистеми економічно ризикованим.

Окремою проблемою є структурна складність веборієнтованої бізнес-системи. На практиці малому підприємству рідко достатньо тільки публічного вебресурсу: потрібні ще засоби наповнення й супроводу бази даних, робочі інтерфейси для персоналу, а іноді й елементи аналітики та інтеграції з іншими сервісами. Відсутність узгодженої моделі таких компонентів призводить до фрагментації рішень: вебресурс створюється в одному середовищі, внутрішні інструменти – в іншому, а початкове наповнення баз даних часто здійснюється вручну або за допомогою разових скриптів, що збільшує сумарні витрати на впровадження і супровід, підвищує ризик помилок і обмежує масштабованість бізнесу.

1.2 Потреби малого бізнесу у власному вебресурсі. Визначення моделі типової вебсистеми

1.2.1 Зовнішні потреби малого бізнесу у вебприсутності

Для малого бізнесу вебресурс виконує передусім функцію публічного представництва: забезпечує базову видимість у пошукових системах, надає потенційним клієнтам інформацію про компанію, перелік послуг чи товарів, контакти та умови співпраці. Вебсистема виступає каналом первинної взаємодії, через який користувач може надіслати запит, залишити заявку, забронювати послугу або сформулювати попереднє замовлення [8].

Окремим аспектом є формування довіри: наявність структурованого й актуального вебресурсу з контактною інформацією, юридичними реквізитами та відгуками знижує бар'єр входу для клієнта та підвищує ймовірність конверсії [8]-[9]. Таким чином, зовнішні потреби малого бізнесу зосереджені навколо доступності, інформативності та зручності первинної взаємодії з клієнтом.

1.2.2 Внутрішні потреби: робота з даними та операційні процеси

Окрім зовнішньої комунікації, малий бізнес потребує інструментів для впорядкованої роботи з даними: номенклатурою товарів чи послуг, клієнтською базою, замовленнями, записами на послуги, станом виконання. Навіть для мікропідприємств, де частину роботи можна виконувати вручну чи у вигляді електронних таблиць, збільшення кількості клієнтів або позицій неминуче призводить до помилок, дублювань і втрати актуальності інформації [10]. Ряд досліджень показує, що переважна більшість бізнес-таблиць містять суттєві помилки, що робить їх ненадійною основою для критично важливих операційних рішень [11]-[12]. Вебсистема з внутрішніми інтерфейсами дозволяє централізувати дані у єдиній базі, запроваджувати мінімальний контроль цілісності та створювати прості звіти для прийняття рішень. Таким чином, внутрішні потреби пов'язані з уніфікацією зберігання даних, підтримкою щоденних операцій та зменшенням ризику помилок.

1.2.3 Типові ролі користувачів вебсистеми малого бізнесу

У типовій веборієнтованій бізнес-системі малого підприємства можна виділити декілька базових ролей користувачів. Власник або адміністратор бізнесу відповідає за загальні налаштування, структуру каталогу, цінову політику, права доступу інших користувачів [13]-[14]. Окрему роль часто відіграє відповідальний за наповнення даних (контент-менеджер або оператор), який створює та актуалізує записи в базі даних: опис товарів чи послуг, графіки роботи, акції. Працівники операційного рівня (менеджери, оператори замовлень, співробітники пунктів обслуговування тощо) використовують вебсистему для опрацювання заявок, зміни статусів, фіксації результатів взаємодії з клієнтами. Кінцевий клієнт у свою чергу взаємодіє з публічним вебресурсом, ініціюючи бізнес-процеси, які далі підтримуються внутрішніми ролями.

1.2.4 Типові сценарії використання веборієнтованої бізнес-системи

З точки зору бізнес-процесів більшість вебсистем малого бізнесу реалізують обмежений набір повторюваних сценаріїв. Для клієнта це, як правило, пошук певної інформації, формування запиту (заявка, бронювання, замовлення), отримання підтвердження та подальше інформування про статус. Для внутрішніх користувачів ключовими сценаріями є: додавання та оновлення записів у базі даних, обробка вхідних звернень, формування простих вибірок і звітів (наприклад, перелік активних замовлень або запланованих візитів). Адміністратор, своєю чергою, керує структурою та довідниками (категоріями, типами послуг, користувачами), забезпечуючи незмінність базових налаштувань при повсякденних змінах контенту.

1.2.5 Узагальнення потреб та формування мінімальної моделі вебсистеми

Узагальнення описаних зовнішніх і внутрішніх потреб дозволяє виділити мінімальну модель веборієнтованої бізнес-системи, спільну для широкого спектра малих підприємств. Така модель включає: публічний вебресурс для клієнтів, інтерфейс наповнення та супроводу бази даних, а також робочий інтерфейс для персоналу, орієнтований на щоденну операційну діяльність [15]. Незважаючи на різноманітність предметних областей, структура ролей, типових сценаріїв і вимог до даних у цих компонентах значною мірою повторюється, відрізняючись переважно набором сутностей та атрибутів. Це створює передумови для опису вебсистем у вигляді пресетів і реалізації їх за допомогою шаблонів, що, у свою чергу, відкриває можливість автоматизованої генерації таких систем та зниження витрат на їх проєктування й впровадження.

1.3 Економічні та ресурсні бар'єри проектування веборієнтованих бізнес-систем

Проектування повноцінної веборієнтованої бізнес-системи для малого підприємства зазвичай охоплює аналіз вимог, розроблення структури бази даних, створення публічного вебресурсу, внутрішніх інтерфейсів для персоналу, налаштування інфраструктури та подальшу підтримку. Галузеві огляди показують, що навіть «базовий» вебпроект для малого бізнесу при використанні класичної моделі розробки коштує у середньому від 3 до 10 тис. дол. США, тоді як більш складні кастомні рішення з внутрішніми панелями та інтеграціями стартують від 15–20 тис. дол. і можуть досягати десятків тисяч доларів. Для значної частини малих підприємств такі витрати є співставними з річним чистим прибутком, що фактично робить індивідуальне проектування вебсистеми економічно недосяжним.

Фінансові витрати тісно пов'язані з трудомісткістю проектування. За оцінками консалтингових агентств, розробка навіть відносно простого вебрішення «з нуля» потребує від 120–200 годин роботи, тоді як розробка системи із внутрішніми інтерфейсами, авторизацією та інтеграціями може вимагати 400–800 годин і більше. Для розробників це означає значне навантаження на команду, а для замовника — довгі терміни виведення системи в експлуатацію. У випадку малого бізнесу затримка запуску на місяці безпосередньо трансформується у втрачений дохід і потенційних клієнтів (табл. 1.3).

Окремим бар'єром є кадровий аспект. Більшість малих підприємств не мають власних ІТ-фахівців і змушені повністю покладатися на зовнішніх підрядників. Дослідження цифрової готовності МСП показують, що саме нестача внутрішніх ІТ-компетенцій та складність координації з постачальниками послуг називаються одними з ключових факторів, які гальмують цифрову трансформацію [16]. Це веде до ситуації, коли бізнес або відмовляється від комплексної вебсистеми, обмежуючись мінімальною присутністю (наприклад, сторінкою у соцмережі), або змушений

користуватися набором розрізнених сервісів, не пов'язаних єдиною моделлю даних.

Нарешті, навіть за наявності фінансування та зовнішніх підрядників значною проблемою залишається повторюваність проєктів і слабка придатність результатів до повторного використання. Кожна нова вебсистема проєктується як окремий проєкт, з власною структурою бази даних, власними інтерфейсами та локальною логікою, хоча за змістом значна частина таких рішень є варіаціями однієї й тієї ж моделі. Це призводить до неефективного використання ресурсів як з боку розробників, так і з боку бізнесу й обумовлює потребу в підходах, які дозволяють формалізувати типові вимоги та автоматизувати проєктування на основі пресетів і шаблонів.

Таблиця 1.3 – Орієнтовні витрати та трудомісткість проєктування веборієнтованих систем

Тип рішення	Орієнтовна разова вартість розробки, дол. США	Орієнтовна трудомісткість, людино-годин	Типові строки розробки, тижнів
Простий інформаційний вебресурс [17]	500–5 000	50–100	2–4
Кастомна вебсистема з окремими інтерфейсами для персоналу та логікою [18]	25 000–60 000	300–800	8–24
Малий бізнес-сайт із БД та базовою адмінкою [19]	5 000–20 000	120–250	4–8

Виходячи з даних, що наведені у таблиці, можна підтвердити те, що класичний підхід до розроблення веборієнтованих бізнес-систем супроводжується суттєвими фінансовими, часовими та кадровими витратами навіть для відносно простих рішень. Для малих підприємств такі параметри часто є критичними, оскільки строки розробки та обсяг залучених ресурсів безпосередньо впливають на економічну доцільність упровадження системи.

1.4 Аналіз наявних підходів і програмних рішень

1.4.1 Конструктори вебресурсів (SaaS-платформи)

До найпоширеніших рішень для малого бізнесу належать онлайн-конструктори вебресурсів (як приклад Wix та Squarespace), які дозволяють без програмування створити публічний вебресурс із базовими бізнес-функціями: каталогом, формами заявок, записом, онлайн-оплатою. Типові бізнес-тарифи таких сервісів коштують орієнтовно 30–40 дол. США на місяць і включають шаблони дизайну, маркетингові інструменти, базову аналітику та елементи електронної комерції [20]-[21]. Основним обмеженням цього підходу є фокус на публічному вебресурсі та інтегрованій адмінпанелі самої платформи: внутрішні процеси підприємства часто залишаються поза межами єдиної вебсистеми й на SaaS-платформах не реалізуються.

1.4.2 CMS-системи та шаблонні платформи електронної комерції

Системи керування контентом (WordPress, Joomla тощо) та готові платформи електронної комерції на їх основі дають змогу будувати більш гнучкі вебсистеми з розширюваною функціональністю, підключати модулі для інтернет-магазину, бронювання, CRM-плагіни. У типовому випадку вартість впровадження для малого бізнесу включає оплату хостингу та теми, але основні витрати формуються за рахунок розробки й інтеграції – від кількох до кількох десятків тисяч доларів залежно від обсягу робіт [22]. При цьому кожний проєкт вимагає окремого проєктування структури бази даних, налаштування ролей користувачів і логіки внутрішніх інтерфейсів, що знижує ступінь повторного використання рішень і збільшує сумарні витрати при створенні багатьох подібних вебсистем.

1.4.3 Шаблони проєктів і засоби «скелетного» генерування коду

У середовищах традиційної розробки (зокрема в екосистемі .NET) широко застосовуються шаблони проєктів та механізми метапрограмування, які вміють автоматично генерувати типові CRUD-компоненти за заданою моделлю даних. Так, в ASP.NET MVC метапрограмування дозволяє швидко створити контролери та подання для стандартних операцій над сутностями, зменшуючи обсяг рутинного коду й прискорюючи первинну стадію розробки [23]. Однак ці засоби залишаються орієнтованими на розробника, а не на бізнес-користувача: кожна вебсистема все одно проєктується як окремий проєкт, а автоматизація стосується переважно технічного, а не формалізованого опису бізнес-конфігурації, з якої могла б бути згенерована вебсистема загалом.

1.4.4 Порівняння та підводження підсумків

Узагальнення наведених підходів (табл. 1.4) показує, що SaaS-конструктори мінімізують вхідний поріг створення публічного вебресурсу, але практично не охоплюють внутрішні процеси підприємства; CMS-системи та шаблонні платформи електронної комерції забезпечують вищу гнучкість, проте кожна вебсистема вимагає окремого проєктування структури даних та інтерфейсів; інструменти шаблонів проєктів і метапрограмування зменшують обсяг роботи із кодом, однак орієнтовані переважно на технічний рівень роботи розробника. Усі розглянуті рішення здебільшого оптимізують створення одиничних вебсистем та не пропонують формалізованої моделі пресета, з якої могла б автоматично генеруватися трьохкомпонентна веборієнтована бізнес-система.

Таблиця 1.4 – Порівняння наявних підходів до створення веборієнтованих бізнес-систем для малого бізнесу

Підхід	Короткий опис	Основні переваги	Обмеження
SaaS конструктори вебресурсів	Хмарний сервіс для швидкого створення публічного вебресурсу на основі шаблонів, з базовими формами, каталогом, оплатою	Дуже низький поріг входу; не потрібні розробники; швидкий запуск (години–дні); включає хостинг та підтримку	Орієнтація переважно на публічний вебресурс; внутрішні реалізуються окремо
CMS та шаблонні платформи електронної комерції	Вебсистема на основі CMS з модулями каталогу, кошика, базової адмінпанелі	Гнучкіша структура, багатий екосистемний набір плагінів; можливість адаптації під предметну область	Кожна вебсистема проєктується окремо; структура БД, ролі та логіка внутрішніх інтерфейсів налаштовуються вручну
Шаблони проєктів і scaffolding у середовищах розробки	Автоматизоване створення каркаса проєкту	Суттєво зменшує обсяг рутинного коду; підвищує стандартизацію технічної структури	Орієнтація на розробника; необхідність повного проєктування предметної області для кожного проєкту
Проєктоване у роботі ПЗ для генерації веборієнтованих бізнес-систем	Програмне забезпечення, яке за формалізованим пресетом бізнесу генерує структуру трьохкомпонентної	Формалізований опис бізнес-конфігурації; автоматизована генерація кількох компонентів системи; можливість багаторазового використання пресетів	Потребує первинних інвестицій у розробку й підтримку ядра; ефект найбільш відчутний при багаторазовому розгортанні типових бізнес-систем

1.5 Специфікація вимог до програмного забезпечення

1.5.1 Формулювання задачі дослідження

Проведений аналіз показав, що для малого бізнесу власна веборієнтована бізнес-система є важливою умовою виходу на ринок, однак її проєктування та розгортання пов'язані зі значними часовими, фінансовими та кадровими витратами. Наявні підходи або фокусуються переважно на публічному вебресурсі, або вимагають повномасштабної індивідуальної розробки для кожного нового проєкту, що обмежує можливості масового впровадження таких систем. Задача даної роботи полягає у дослідженні й розробці програмного забезпечення, яке, спираючись на формалізований опис бізнес-конфігурації, здатне автоматизовано формувати структуру веборієнтованої бізнес-системи та тим самим зменшувати ресурсозатратність її створення.

Таким чином можна сформулювати підсумковий вид ПЗ як засіб, що приймає на вході пресет бізнес-системи та генерує на його основі проєктну структуру вебсистеми з мінімально необхідним набором компонентів. Отже, дослідження спрямоване не стільки на розширення функціональності окремої вебсистеми, скільки на зменшення витрат при багаторазовому створенні типових рішень для різних підприємств.

1.5.2 Змістові та функціональні вимоги до проєктованого ПЗ

Виходячи з проаналізованих потреб малого бізнесу та обмежень наявних підходів, до розроблюваного програмного забезпечення висуваються такі змістові та функціональні вимоги.

По-перше, система повинна забезпечувати можливість опису бізнес-системи у вигляді пресета, що включає: перелік сутностей предметної області, їх атрибути, зв'язки між сутностями, базові ролі користувачів та типові сценарії роботи з даними. Такий опис має бути достатньо формальним

для подальшої автоматизованої обробки, але водночас придатним для використання фахівцями, які не є професійними розробниками.

По-друге, програмне забезпечення має на основі заданого пресета автоматично формувати структуру щонайменше трьох компонентів веборієнтованої бізнес-системи: публічного вебресурсу для клієнтів, інтерфейсу наповнення та супроводу бази даних, а також робочого інтерфейсу для персоналу. Згенерована структура повинна забезпечувати узгодженість моделі даних між усіма компонентами та підтримувати мінімальний контроль цілісності й унікальності ідентифікаторів.

По-третє, інструмент має підтримувати повторне використання пресетів і їх варіантів: користувач повинен мати змогу створити нову конфігурацію на основі вже наявної, вносячи зміни лише у специфічні для конкретного бізнесу параметри. Це дозволить оцінювати ефективність підходу саме в умовах багаторазового розгортання подібних веборієнтованих систем.

1.5.3 Критерії досягнення мети дослідження

З огляду на мету роботи, базовим критерієм ефективності розроблюваного програмного забезпечення є скорочення сумарних часових й відповідно фінансових витрат на створення веборієнтованих бізнес-систем у порівнянні з традиційним підходом. У загальному вигляді часові витрати на розробку n систем за класичної моделі можна представити за формулою (1.1):

$$T_{\text{баз}} = n \cdot T_{\text{проект}}, \quad (1.1)$$

де $T_{\text{баз}}$ – сумарні часові витрати на створення n веборієнтованих бізнес-систем за базового підходу;

n – кількість веборієнтованих бізнес-систем, що розробляються;

$T_{\text{проект}}$ – середній час повної розробки однієї системи за базового підходу.

Для підходу на основі генератора часові витрати матимуть вигляд, що задає формула (1.2):

$$T_{\text{ген}} = T_{\text{ядро}} + n \cdot T_{\text{конф}}, \quad (1.2)$$

де $T_{\text{ген}}$ – сумарні часові витрати на створення n веборієнтованих бізнес-систем із використанням розроблюваного генератора;

$T_{\text{ядро}}$ – разові часові витрати на створення самого програмного забезпечення;

$T_{\text{конф}}$ – середній час конфігурування однієї системи на основі пресета.

Одним з критеріїв досягнення мети дослідження є оцінка різниці $T_{\text{баз}}$ й $T_{\text{ген}}$ і визначення діапазону n , для якого $\Delta T > 0$, тобто генератор забезпечує реальне скорочення затрат часового ресурсу.

Також у рамках дослідження часові показники можуть бути інтерпретовані в економічній сфері шляхом введення єдиної умовної ставки вартості години роботи фахівця, однак ключовим є саме зменшення тривалості проєктування й розгортання систем, що піддається безпосередньому вимірюванню в експериментальній частині.

1.6 Висновки за розділом

У розділі обґрунтована важливість наявності для малого й середнього бізнесу власної веборієнтованої бізнес-системи для виходу на ринок, розглянута проблема її реалізації, що пов'язана зі значними часовими та фінансовими витратами. На основі аналізу потреб сформовано мінімальну узагальнену модель такої системи, яка включатиме публічний вебресурс, інтерфейс наповнення та супроводу бази даних і робочий інтерфейс для персоналу.

Порівняння наявних підходів продемонструвало їх неспроможність у деяких аспектах та встановило напрямок нашої розробки.

2 МАТЕРІАЛИ І МЕТОДИ ДОСЛІДЖЕННЯ

2.1 Загальна методологія дослідження

Методологія дослідження ґрунтується на послідовному виконанні аналітико-оглядового, модельного, проєктно-реалізаційного та експериментального етапів. На основі результатів першого розділу формується клас цільових веборієнтованих бізнес-систем і вимоги до них, після чого розробляється формальна модель пресета бізнес-системи та підхід до автоматизованої генерації трьохкомпонентної вебсистеми. Подальші етапи включають реалізацію прототипу програмного забезпечення-генератора та порівняльну оцінку його ефективності за часовими показниками й якістю конфігурації.

2.1.1 Етапи проведення дослідження

Відокремлено наступні етапи дослідження:

а) аналітико-оглядовий етап:

1) аналіз економічної ролі МСБ, потреб у веборієнтованих системах і наявних програмних рішень;

2) формулювання проблеми та мети дослідження;

б) етап побудови концептуальної моделі:

1) виділення узагальненої структури вебсистеми;

2) формалізація пресета як вхідних даних для генератора;

в) етап проєктування та реалізації прототипу:

1) розроблення алгоритмів автоматизованої генерації;

2) реалізація програмного забезпечення;

г) експериментальний етап:

1) побудова кількох тестових пресетів;

2) порівняння часових витрат, аналіз отриманих результатів за критеріями, визначеними у розділі 1.

2.1.2 Логічна схема та узагальнення етапів дослідження

Дослідницький процес – послідовність взаємопов’язаних етапів, що відображають перехід від аналізу предметної області до побудови моделі, реалізації програмного забезпечення та оцінювання його ефективності. На логічному рівні ці етапи утворюють наступний ланцюжок: аналіз проблеми й наявних підходів – формування вимог і моделі веборієнтованої бізнес-системи – розроблення концепції пресета та алгоритмів генерації – реалізація прототипу ПЗ – експериментальне порівняння базового підходу та генератора – інтерпретація результатів (табл. 2.1).

Таблиця 2.1 – Узагальнена характеристика етапів дослідження

Етап дослідження	Поставлені завдання	Методи виконання	Бажані результати
Аналітико-оглядовий	Провести аналіз ролі МСБ, потреб у вебсистемах та огляд наявних підходів і програмних рішень	Аналіз літератури, огляд ринку, порівняльний аналіз	Сформульовані проблема, мета, первинні вимоги
Побудова концептуальної моделі	Узагальнити структури веборієнтованих систем, формалізувати пресет бізнес-системи	Абстрагування, моделювання, узагальнення	Модель пресета
Проектування та реалізація прототипу генератора	Розробити алгоритми генерації та реалізувати ПЗ	Компонентне та шаблонне проектування й програмування	Прототип ПЗ
Оцінка	Порівняти базовий та винайдений підходи за часовими витратами	Експеримент, вимірювання, аналіз	Показники ефективності, підсумки

2.2 Вихідні дані та обґрунтування вибору тестового пресета

Розглянуто клас веборієнтованих бізнес-систем типу інтернет-магазину техніки, для яких характерні розгалужений каталог товарів, фільтрація, картки товарів, оформлення замовлення та базові операційні статуси. Як репрезентативний приклад було обрано інтернет-магазин комп'ютерної техніки TELEMART.UA, який має великий категоріальний каталог, розвинену систему фільтрів, картки товарів з описами та характеристиками, а також механізми оформлення замовлень і сервісних послуг. У структурі сайту також присутні реєстрація користувача, особистий кабінет і інформаційні розділи, що відображає типовий набір функцій для веборієнтованих бізнес-систем цього класу.

Аналіз структури дозволяє виділити базові сутності, спільні для більшості інтернет-магазинів техніки: товар, категорія, бренд, клієнт, замовлення, позиція замовлення. Для товарів використовуються атрибути на кшталт назви, опису, ціни, технічних характеристик, наявності та ідентифікаторів, для замовлень – дата, статус, сума, спосіб оплати й доставки, для клієнтів – контактні дані та історія покупок. Така структура має чітку логіку й сприяє легкій формалізації у вигляд пресета.

У розробленому програмному забезпеченні як тестовий пресет використано спрощену модель, що відтворює ключові сутності та зв'язки, які можна спостерігати в реальних системах на кшталт обраного магазину TELEMART.UA. Узагальнена структура пресета наведена в табл. 2.2.

Таблиця 2.2 – Основні сутності пресета

Сутність	Роль у системі	Приклади ключових атрибутів
Товар	Одиниця номенклатури	Назва, опис, ціна, внутрішній код товару, гарантія
Категорія	Групування товарів	Назва, опис, порядок відображення

Продовження таблиці 2.2

Сутність	Роль у системі	Приклади ключових атрибутів
Бренд	Виробник / торгова марка	Назва, країна, офіційний сайт
Клієнт	Кінцевий покупець	ПІБ, контактні дані, історія замовлень
Замовлення	Сукупність позицій, оформлена клієнтом	Дата, статус, сума, спосіб оплати й доставки
Позиція замовлення	Зв'язок між товаром і замовленням	Посилання на товар, кількість, ціна на момент

2.3 Програмні та технологічні засоби

Як програмну платформу для розробки програмного забезпечення доцільно обрати екосистему Microsoft .NET у поєднанні з мовою програмування C#. Перевагою є єдність технологічної бази для настільного компонента – засобу конфігурування пресетів – і веборієнтованого компонента, а також наявність розвинutih засобів шаблонізації та автоматизованого створення типових елементів застосунку.

Для веборієнтованого компонента передбачається застосування ASP.NET MVC, яке реалізує шаблон «модель–подання–контролер» і дозволяє чітко розмежувати рівні представлення, бізнес-логіки та доступу до даних. Доступ до даних планується здійснювати через класи моделей і контексту даних, що відображаються на структуру реляційної бази даних та забезпечують типобезпечну роботу з основними сутностями предметної області.

Клієнтська частина вебресурсу базуватиметься на стандартних вебтехнологіях HTML, CSS та JavaScript, котрі у сукупності є зручним інструментарієм для формування інтерфейсу користувача, базову валідацію введення та інтерактивну взаємодію з елементами сторінки.

2.4 Методи моделювання пресетів та формування структури вебсистеми

Моделювання пресета бізнес-системи в роботі виконується у два основні кроки. Спочатку здійснюється логічне виділення сутностей предметної області (товар, категорія, бренд, клієнт, замовлення тощо), їх атрибутів та зв'язків на основі аналізу типового інтернет-магазину техніки. Для опису використовується таблична форма (переліки сутностей та атрибутів) і спрощені діаграми зв'язків, що дозволяє отримати компактну, але формальну модель предметної області, придатну для подальшої автоматизованої обробки.

На другому кроці задаються правила перетворення пресета на структуру веборієнтованої системи. Для кожної сутності визначається її роль (довідкова, операційна, користувацька), типові операції з даними (створення, перегляд, редагування, видалення) та відображення в компонентах системи. Узагальнено передбачається, що кожна «основна» сутність породжує відповідну структуру даних і стандартний набір інтерфейсів відображення: записи у базі даних, спискове представлення, форму перегляду та форму редагування. Для пов'язаних сутностей задаються правила відображення зв'язків (наприклад, «замовлення – позиції замовлення», «товар – категорія») у вигляді зовнішніх ключів і пов'язаних елементів інтерфейсу.

Алгоритми формування структури вебсистеми будуються як застосування цих правил до пресета (рис. 2.1). На рівні методики це означає, що генератор послідовно проходить описані сутності, створює для них відповідні записи у моделі даних, шаблонні елементи вебінтерфейсу та службові елементи (маршрути, зв'язки між компонентами). Окремо передбачається етап валідації пресета, на якому перевіряється повнота опису сутностей, узгодженість зв'язків та унікальність ідентифікаторів (наприклад, кодів товарів і slug-ідентифікаторів), що зменшує ймовірність помилок ще до стадії генерації проєкту.

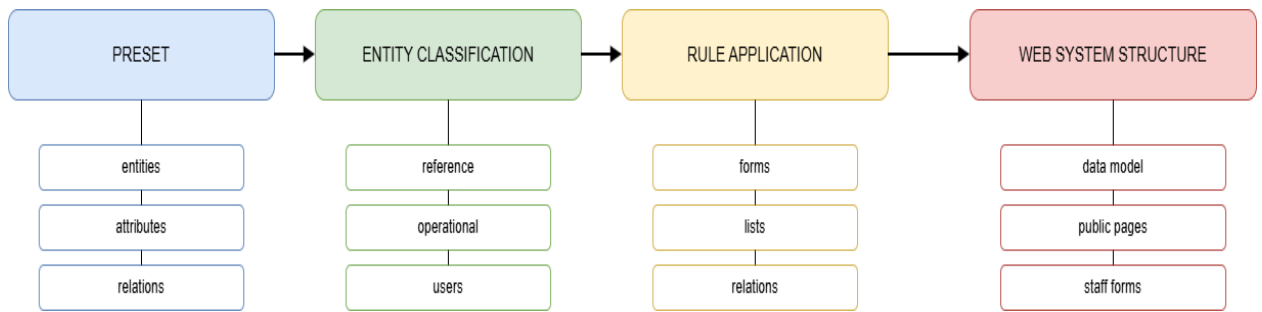


Рисунок 2.1 – Схема перетворення пресета на структуру вебсистеми

2.5 Методи оцінювання ефективності та постановка експерименту

Оцінка ефективності запропонованого підходу ґрунтується на порівнянні часових витрат на створення веборієнтованої бізнес-системи стандартним шляхом та шляхом застосування генератора, а також на аналізі помилок конфігурації. Основними показниками є:

- $T_{\text{баз}}$ оцінка часу, необхідного для створення структури вебсистеми за базового;
- $T_{\text{ген}}$ – час конфігурування пресета та запуску генерації вебсистеми з використанням розробленого програмного забезпечення;
- ΔT – перевага у часі.

Як тестовий сценарій використовується один репрезентативний пресет, описаний у підрозділі 2.2. Для нього розглядається задача побудови мінімальної трьохкомпонентної вебсистеми, що складатиметься з публічного каталогу товарів, інтерфейсу наповнення бази даних, робочого інтерфейсу обробки замовлень тощо.

Показник $T_{\text{ген}}$ передбачається вимірювати практично як сумарний час виконання користувачем операцій конфігурування пресета в настільному застосунку та запуску процедури генерації, фіксований за допомогою стандартних засобів вимірювання часу.

Для показнику $T_{\text{баз}}$ використовуватиметься теоретична оцінка на основі розкладання процесу ручного створення системи на окремі етапи та

орієнтовних тривалостей виконання кожного етапу для того самого виконавця. Таким чином, експеримент дає можливість порівняти модельні часові витрати базового підходу з фактичними витратами при використанні генератора на одному детально описаному прикладі. Отримані значення показників будуть інтерпретовані у контексті критеріїв досягнення мети, сформульованих у підрозділі 1.5.3.

2.6 Висновки за розділом

У другому розділі сформовано методологічну основу дослідження, яка включає послідовність аналітико-оглядового, модельного, проєктно-реалізаційного та експериментального етапів. Для кожного з них визначено завдання, використовувані методи та очікувані результати.

Обґрунтовано вибір класу досліджуваних систем – веборієнтовані бізнес-системи типу інтернет-магазину техніки – та сформовано репрезентативний тестовий пресет на основі аналізу структури реального магазину. Виділено базові сутності, їхні ролі в системі та ключові атрибути.

Визначено програмно-технологічну та описано методи моделювання пресетів і перетворення їх на структуру вебсистеми з урахуванням класифікації сутностей.

Сформульовано підхід до оцінювання ефективності запропонованого рішення на основі порівняння часових витрат базового підходу та генератора, а також аналізу помилок конфігурації при побудові трьохкомпонентної вебсистеми для обраного пресета.

3 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Загальна архітектура програмного забезпечення

Програмний комплекс складається з двох взаємопов'язаних компонентів: настільного застосунку-генератора пресетів бізнес-системи та веборієнтованої системи, структура якої формується на основі згенерованих артефактів. Настільний застосунок відповідає за конфігурування пресета, його збереження у структурованому вигляді, валідацію опису та побудову проміжного опису структури модулів. Вебзастосунок ASP.NET MVC використовує цю карту для узгодженого формування моделей даних, контролерів, подань і структури бази даних, реалізуючи трьохкомпонентну логіку: публічний вебресурс для клієнтів, інтерфейс наповнення та супроводу бази даних і робочий інтерфейс для персоналу.

З архітектурного боку, настільний застосунок складається з шару інтерфейсу користувача, шару доменної моделі, сервісного шару для роботи зі сховищем пресетів і файлова підсистема, а також модуля шаблонного опрацювання, який на основі карти модулів формує структуру вебпроєкту. Веборієнтована частина реалізує шар даних, шар бізнес-логіки й контролерів, шар подань для публічної частини та окрему зону адміністративних інтерфейсів, що відображають ті самі сутності у режимі керування.

Користувач описує структуру бізнес-системи в настільному застосунку, який перевіряє коректність конфігурації та зберігає її у вигляді пресета й карти модулів; далі на основі цієї карти формуються файли вебпроєкту зі стандартизованою структурою каталогів, моделей, контролерів, подань і конфігурації доступу до даних. На рис. 3.1 подано компонентну схему програмного комплексу з відображенням основних учасників, настільного генератора, сховища пресетів і карти модулів, бібліотеки шаблонів, згенерованої вебсистеми та бази даних.

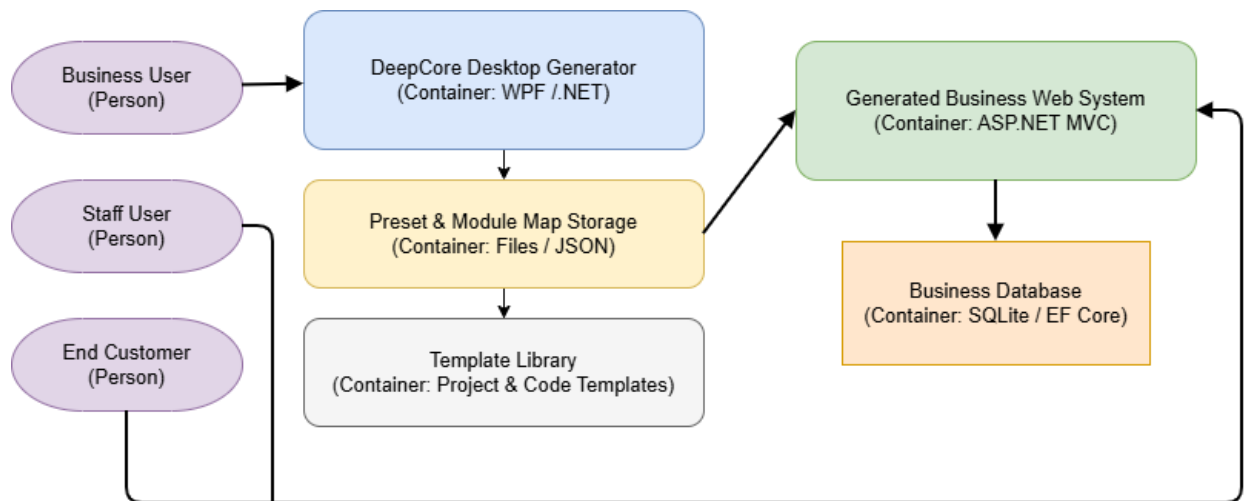


Рисунок 3.1 – Компонентна схема архітектури ПЗ

3.2 Функціональні вимоги та сценарії використання

3.2.1 Функціональні вимоги до настільного генератора

Настільний генератор має забезпечувати створення, редагування та збереження пресетів бізнес-систем, включно з описом сутностей, атрибутів, зв'язків і ролей користувачів. Система повинна виконувати валідацію пресета: перевіряти повноту опису сутностей, узгодженість зв'язків, унікальність ідентифікаторів, а також повідомляти користувача про виявлені помилки. Необхідно підтримати збереження пресетів у структурованому форматі для подальшого повторного використання та формування карти модулів, яка визначає склад компонентів цільового вебзастосунку. Генератор має надавати засоби запуску процедури автоматизованого створення структури вебпроєкту на основі пресета та фіксації.

Основні дії користувача генератора та пов'язані з ними підзадачі валідації конфігурації, підготовки початкових даних і ведення журналу подій узагальнено подано на діаграмі варіантів використання (рис. 3.2).

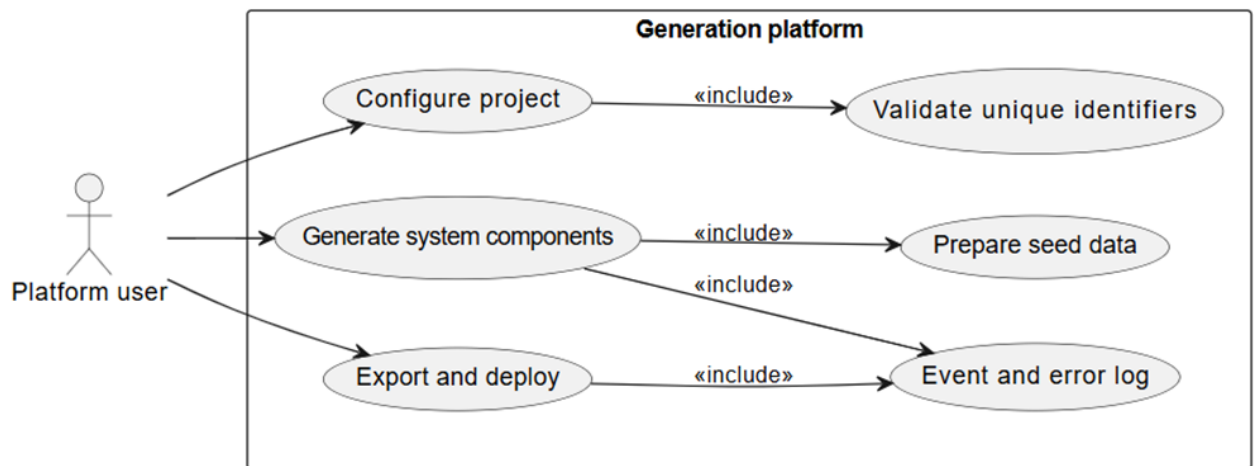


Рисунок 3.2 – Діаграма варіантів використання платформи генерації

3.2.2 Функціональні вимоги до згенерованої вебсистеми

Згенерована веборієнтована система повинна реалізовувати трикомпонентну структуру: публічний вебресурс для клієнтів, інтерфейс наповнення бази даних та робочий інтерфейс для персоналу. У публічній частині мають бути доступні перегляд каталогу, сторінки елементів, базові засоби пошуку та фільтрації відповідно до сутностей, визначених у пресеті. Інтерфейс наповнення бази даних і робочий інтерфейс для персоналу повинні забезпечувати CRUD-операції над основними сутностями, відображення зв'язків між ними та контроль доступу згідно з ролями користувачів. Вебзастосунок має працювати з єдиною базою даних, узгодженою зі структурою пресета, підтримувати перевірку унікальності критичних ідентифікаторів і коректно відображати зміни конфігурації, отримані при повторній генерації системи.

3.2.3 Основні сценарії використання

У межах функціонування розробленого програмного забезпечення доцільно виділити два базові сценарії використання. Перший сценарій охоплює конфігурування та супровід пресета бізнес-системи: бізнес-користувач у настільному генераторі задає сутності, їх атрибути, зв'язки та

ролі, виконує валідацію конфігурації, зберігає пресет у сховищі та, за потреби, вносить до нього зміни з подальшим повторним запуском генерації. Другий сценарій стосується генерації та експлуатації вебсистеми: на основі вибраного пресета автоматично формується карта модулів і структура вебзастосунку, після чого згенерована система використовується кінцевими клієнтами через публічний вебресурс і персоналом через внутрішні інтерфейси для ведення довідників та обробки операційних даних.

Структуру основних варіантів використання вебсистеми з боку клієнтів, працівників і адміністратора, а також поділ на публічну та робочу частини наведено на рис. 3.3.

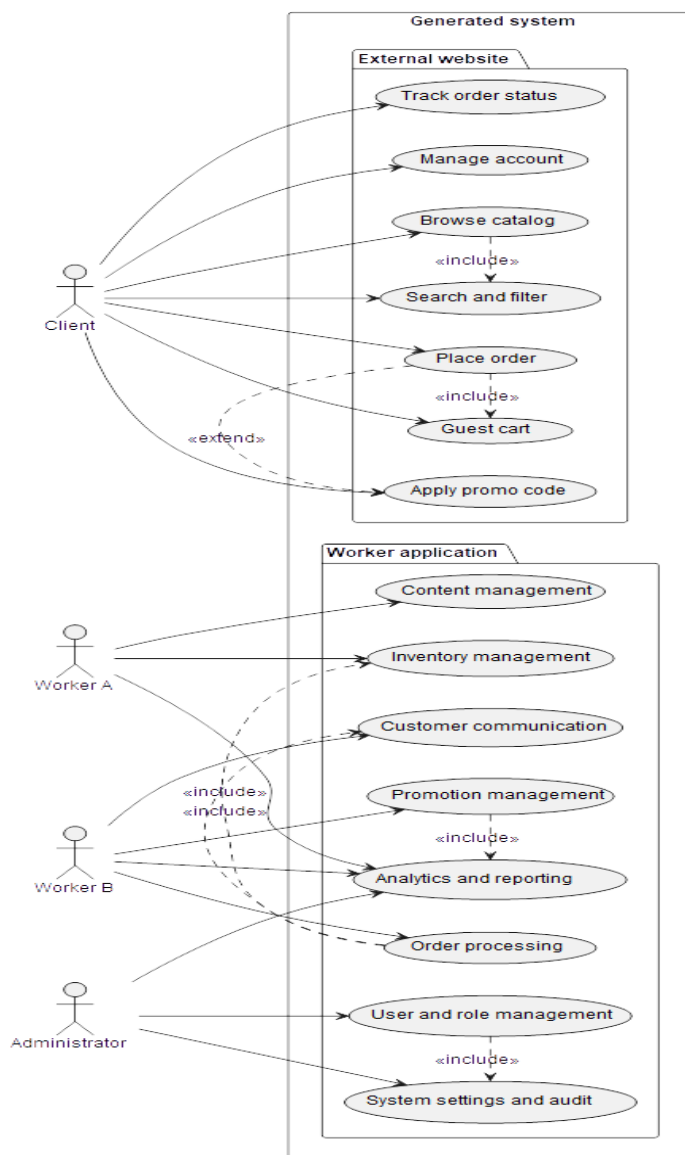


Рисунок 3.3 – Діаграма варіантів використання згенерованої системи

3.3 Модель даних і структура пресета

У розроблюваному програмному комплексі конфігурація бізнес-системи задається у вигляді пресета – формалізованого опису предметної області, який використовується настільним застосунком для генерації структури веборієнтованої системи. Пресет виступає надбудовою над конкретною СКБД та фреймворком ASP.NET MVC: він задає склад сутностей, їхні атрибути, зв'язки, ролі користувачів і призначення кожної сутності у системі.

Основу моделі пресета становить набір описів сутностей. Для кожної сутності визначаються: унікальний ідентифікатор, людиночитна назва, набір атрибутів із типами даних, обмеженнями та службовими ознаками (наприклад, «видимість» у публічній частині або в інтерфейсі персоналу). Окремо задаються зв'язки між сутностями, які надалі відображаються на зовнішні ключі в базі даних та навігаційні властивості в моделях вебзастосунку.

Сутності додатково класифікуються за роллю в системі на довідкові, операційні та користувацькі. Така класифікація потрібна генератору для застосування різних шаблонів: для довідкових сутностей створюються переважно списки й прості форми редагування, для операційних – повні CRUD-інтерфейси з відображенням зв'язків, для користувачів – форми реєстрації та авторизації, прив'язка до ролей доступу.

Окремою групою атрибутів у пресеті є ідентифікатори, що мають бути унікальними у межах системи: SKU для товарів, slug для веб-адрес сторінок тощо. Для таких полів фіксуються правила перевірки унікальності, які використовуються як на етапі валідації пресета в настільному застосунку, так і в згенерованій вебсистемі при роботі з даними. Це забезпечує узгодженість конфігурації та знижує ризик помилок при розгортанні кількох схожих інстансів системи. Концептуальна модель пресета представлена на рис. 3.4.

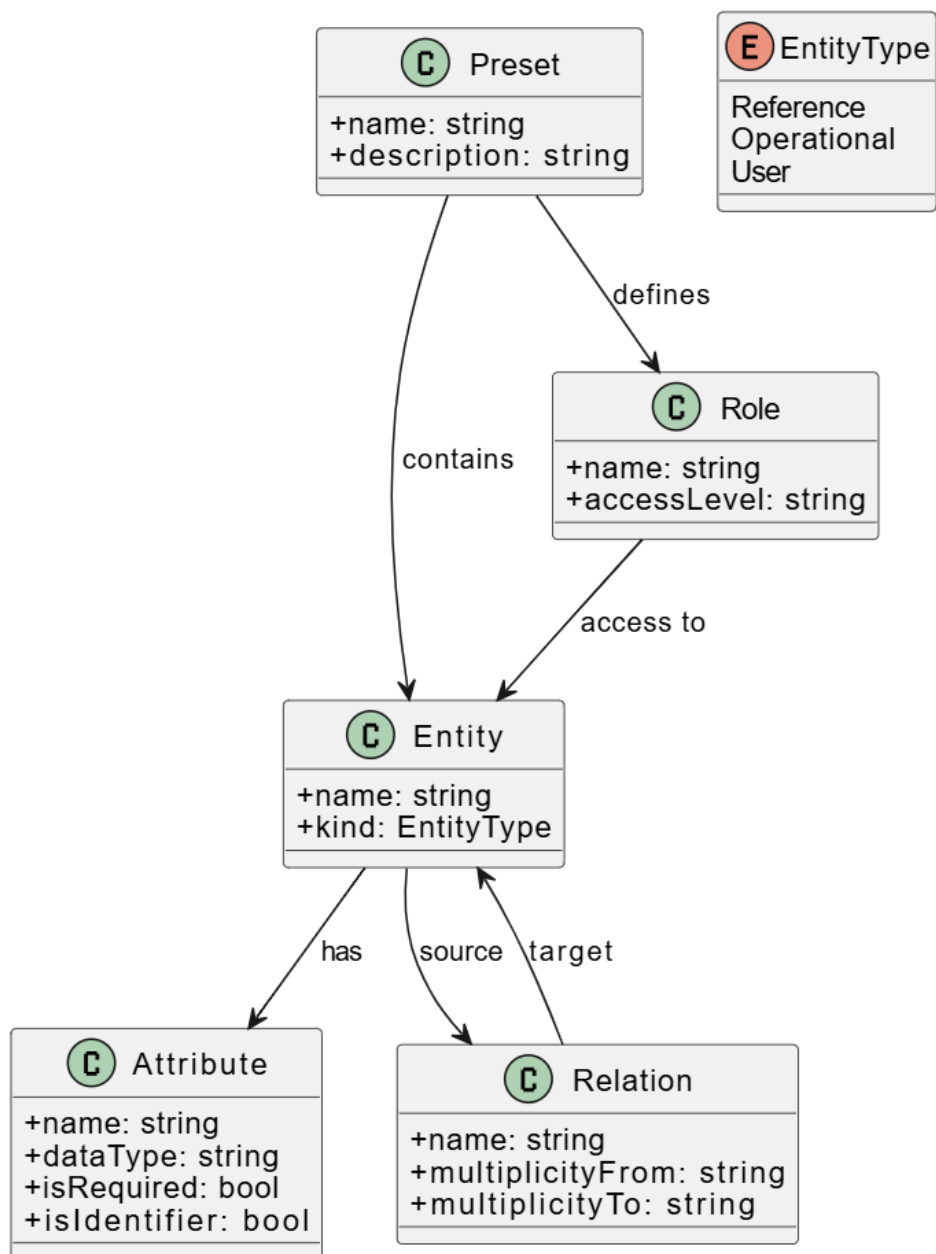


Рисунок 3.4 – Концептуальна модель пресета

Пресет представлено набором C#-класів доменної моделі, які можуть серіалізуватися у формат JSON для збереження та подальшої обробки. На основі цих класів формується карта модулів, де для кожної сутності фіксується, які компоненти вебпроєкту мають бути згенеровані: таблиця бази даних, модель даних у вебзастосунку, контролери та подання публічної частини, елементи адміністративного інтерфейсу. Відповідність між елементами пресета й артефактами вебсистеми показано на рис. 3.2.

3.4 Архітектура генератора

Настільний застосунок генератора реалізовано як багат шаровий WPF-застосунок на платформі .NET, у якому чітко виділено шар інтерфейсу користувача, шар доменної моделі пресета, сервісний шар та модуль шаблонного генерування. Такий поділ дозволяє ізолювати роботу з пресетом і правилами генерації від технічних деталей збереження, журналювання та формування файлів вебпроєкту, що спрощує супровід і дає змогу розвивати окремі частини системи незалежно.

Шар інтерфейсу користувача забезпечує екрани для створення й редагування пресета: введення сутностей, атрибутів, зв'язків, ролей користувачів, параметрів цільової системи та каталогу генерації. Події елементів керування не містять бізнес-логіки, а лише делегують операції сервісним компонентам, що працюють з об'єктами доменної моделі. Доменний шар представлено класами, які описують структуру пресета, класифікацію сутностей, параметри ідентифікаторів та карту модулів, що визначає склад компонентів майбутньої вебсистеми.

Сервісний шар відповідає за збереження пресетів та карт модулів у файловому сховищі (серіалізація у JSON), завантаження існуючих конфігурацій, ведення журналу подій і реалізацію валідації пресета. Саме тут виконуються перевірки повноти опису сутностей, коректності зв'язків, унікальності критичних ідентифікаторів (наприклад, SKU та slug), а результат валідації повертається до інтерфейсу у вигляді повідомлень для користувача.

Модуль шаблонного генерування реалізує перетворення пресета й карти модулів на структуру проєкту ASP.NET MVC. Для кожного типу сутності та ролі в системі обираються відповідні шаблони моделей, контролерів, подань і конфігураційних файлів, у які підставляються назви сутностей, атрибути, типи даних, зв'язки та службові ознаки. На виході формується готовий каркас вебзастосунку зі стандартизованою структурою каталогів і початковою реалізацією CRUD-операцій. Структурну схему

основних модулів настільного застосунку-генератора та їх взаємодії подано на рис. 3.5.

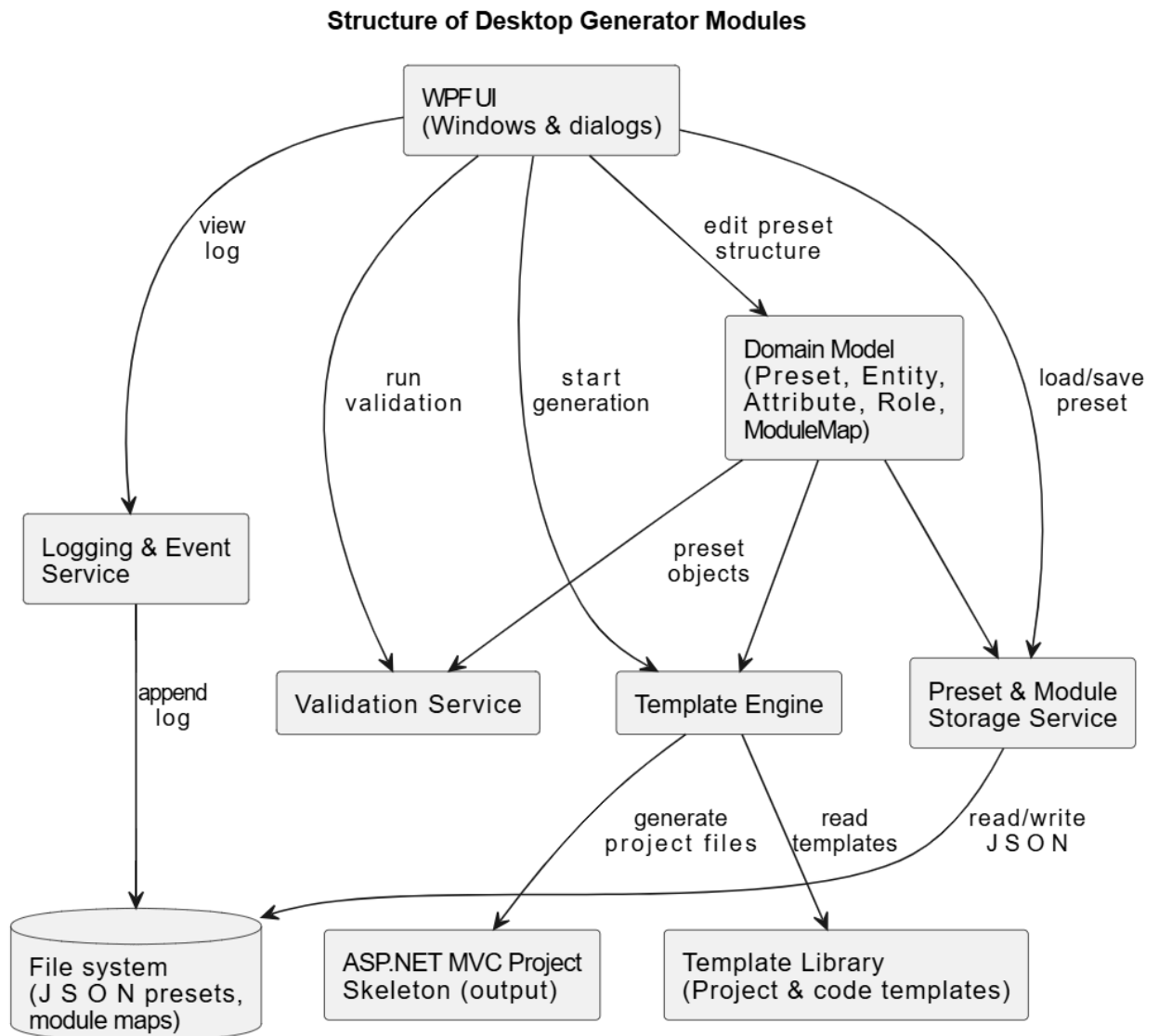


Рисунок 3.5 – Структурна схема модулів застосунку-генератора

3.5 Архітектура згенерованої веборієнтованої системи

Згенерована веборієнтована система реалізована як багат шаровий вебзастосунок на платформі ASP.NET MVC, структура якого формується на основі пресета та карти модулів. Логічно вона поділяється на три компоненти: публічний вебресурс для клієнтів, внутрішній застосунок для персоналу та шар доступу до даних, що забезпечує роботу обох інтерфейсів із єдиною базою даних.

Шар даних побудовано на основі об'єктно-реляційного перетворювача Entity Framework Core: для кожної сутності, описаної у пресеті, створюється модель даних і відповідна колекція у класі контексту БД. Структура таблиць бази даних, зовнішні ключі та обмеження формуються відповідно до зв'язків між сутностями, заданих у пресеті. У базі даних зберігаються як довідкові сутності, так і операційні записи, а також облікові записи користувачів і працівників.

Шар контролерів реалізує бізнес-логіку вебсистеми та розділений за зонами відповідальності. У публічній частині контролери забезпечують відображення каталогу, сторінок елементів, пошук і фільтрацію відповідно до атрибутів сутностей. В адмін-області розміщено контролери для CRUD-операцій над довідковими й операційними сутностями, обробки замовлень та базових функцій адміністрування. Імена контролерів і склад дій генеруються на основі карти модулів, тому структура HTTP-інтерфейсу безпосередньо відображає структуру пресета.

Презентаційний шар побудовано на razor-видах, згрупованих за контролерами й зонами. Для кожної сутності формуються типові подання списку, створення чи редагування й перегляду, у яких автоматично відображаються атрибути, визначені пресетом, з урахуванням їхніх типів і службових ознак. Окремі подання забезпечують специфічні сценарії.

Узагальнену шарову архітектуру згенерованої веборієнтованої системи наведено на рис. 3.6.

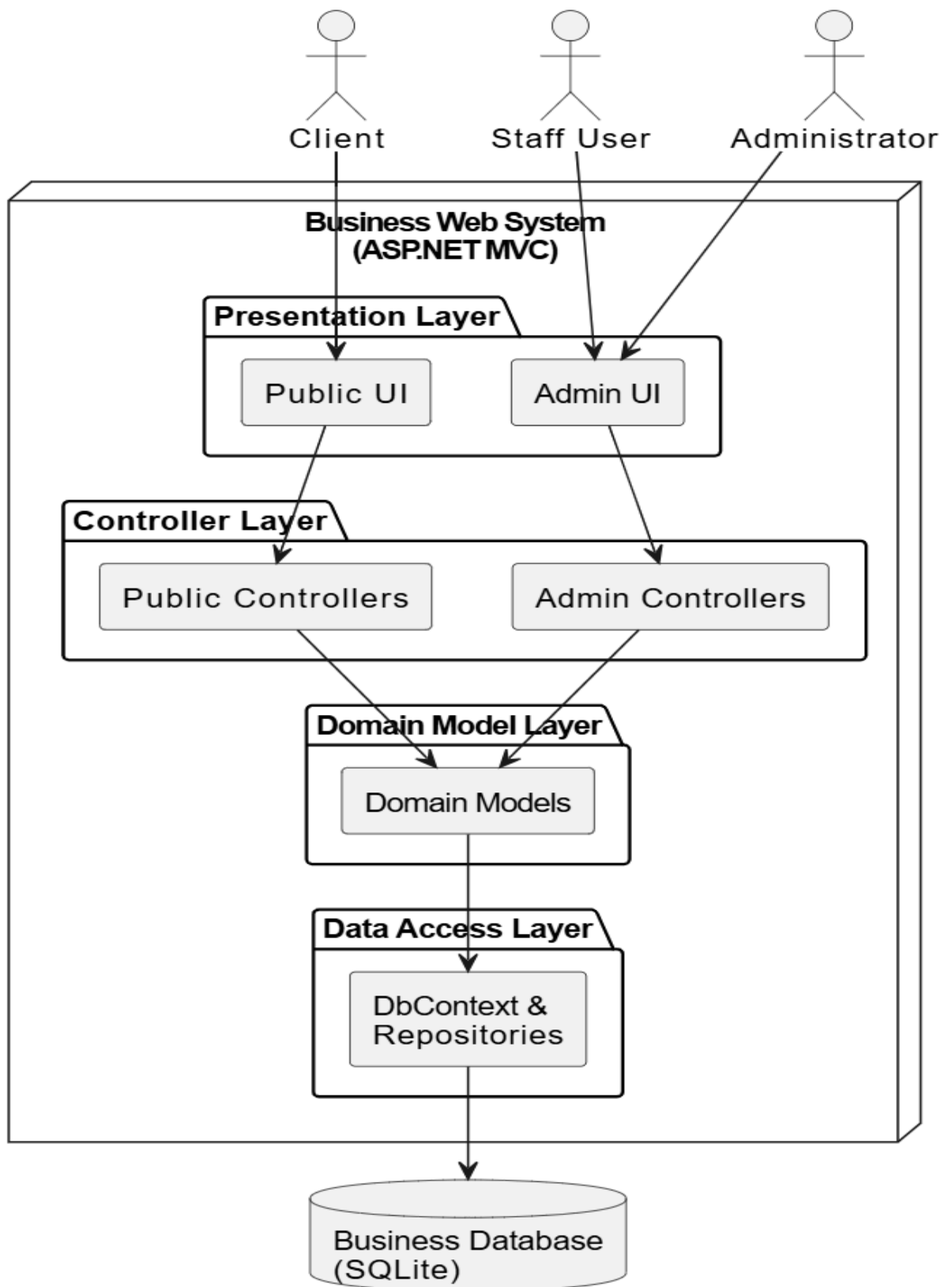
Layered architecture of the generated web system

Рисунок 3.6 – Шарова архітектура згенерованої веборієнтованої системи

3.6 Висновки за розділом

У цьому розділі було спроектовано архітектуру програмного забезпечення, яке поєднує настільний застосунок-генератор пресетів та згенеровану веборієнтовану систему, що працюють як єдиний інструмент для побудови бізнес-орієнтованих вебресурсів. Визначено загальну компонентну структуру комплексу, ролі основних учасників і потоки даних між генератором, сховищем пресетів, бібліотекою шаблонів, вебзастосунком і базою даних.

Сформовано функціональні вимоги до настільного генератора та веборієнтованої системи, а також основні сценарії їх використання з позицій бізнес-користувача, персоналу та клієнтів, що дало змогу узгодити архітектурні рішення з практичними потребами експлуатації. На основі метамоделі пресета побудовано модель даних, яка забезпечує пряме відображення сутностей, атрибутів, зв'язків і ролей користувачів на структуру таблиць бази даних, контролерів та подань вебзастосунку.

Окремо було розроблено архітектуру настільного генератора як багатошарового застосунку та шарову архітектуру згенерованої вебсистеми на базі ASP.NET MVC з єдиною базою даних і розподіленням на публічний інтерфейс і робочий застосунок для персоналу. Запропоновані рішення забезпечують можливість багаторазового використання пресетів, автоматизовану побудову каркаса вебсистеми та зменшення обсягу ручного проєктування.

4 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Послідовність реалізації програмного забезпечення

Розробка програмного забезпечення проходить поетапно, починаючи з побудови базової вебсистеми-прототипу та завершуючи створенням настільного генератора, що автоматизує формування подібних систем.

На першому етапі було створено вебзастосунок на основі шаблону ASP.NET MVC, у якому сформовано початкову структуру проєкту з каталогами моделей, контролерів, подань, даних та конфігураційних файлів. Паралельно було виділено окреме рішення для настільного застосунку «DeerCore», який надалі мав виконувати роль генератора пресетів і модулів.

Другим етапом стало формування предметної області. У вебпроєкті було створено набір класів-моделей, зокрема класів Product, Category, Customer, Order, OrderItem, Employee, які описують товари, категорії, клієнтів, замовлення, позиції замовлень та працівників. На основі цих класів було реалізовано клас контексту бази даних, що задає колекції сутностей та зв'язки між ними, після чого автоматично сформовано структуру таблиць у базі даних та забезпечено зв'язки «один-до-багатьох» або «багато-до-одного».

Третім етапом було розроблено вебінтерфейси, які безпосередньо працюють з цими моделями. Для публічної частини створено перелік контролери та відповідні подання для відображення каталогу товарів, сторінок окремого товару й оформлення замовлення. Для персоналу реалізовано окрему адміністративну область у каталозі Areas/Admin з контролерами керування товарами, категоріями та замовленнями, а також з поданнями для виконання типових CRUD-операцій. Таким чином, предметна модель почала використовуватися як основа одночасно для рівня даних і для інтерфейсів різних ролей користувачів.

На четвертому етапі було проведено аналіз реалізованої вебсистеми з точки зору повторюваних шаблонів. Для кожного класу-моделі визначено типовий набір артефактів: таблиця в базі даних, клас-модель у вебзастосунку,

контролер, набір подань списку й редагування, а також елементи адміністративного інтерфейсу. На основі цього було сформовано структуру пресета та проміжний опис у вигляді карти модулів, а у вебпроєкті створено конфігураційний файл, у якому фіксується склад модулів для кожної сутності.

П'ятим етапом стала реалізація настільного застосунку DeepCore. У цьому проєкті було створено класи доменної моделі пресета (зокрема класи `Preset`, `EntityDefinition`, `AttributeDefinition`, `RoleDefinition`, `ModuleMap`), сервіси для серіалізації пресета й карти модулів у формат JSON, а також сервіси валідації та журналювання. Поверх цих класів реалізовано графічний інтерфейс на WPF: головне вікно з елементами для редагування сутностей, атрибутів і ролей, вибору цільового каталогу вебпроєкту та запуску генерації. Останнім кроком було додано модуль шаблонного генерування, який, використовуючи дані пресета й карти модулів, підставляє відповідні параметри у шаблони контролерів, моделей і подань, формуючи або оновлюючи структуру вебзастосунку. Це забезпечило можливість багаторазового створення подібних веборієнтованих бізнес-систем із мінімальними ручними змінами.

4.2 Деталізована реалізація вебсистеми

4.2.1 Моделі даних і контекст бази даних

У вебпроєкті було створено набір класів моделей, які описують предметну область магазину (рис. 4.1). Зокрема, клас «`Product`» зберігає основні характеристики товару (ідентифікатор, назву, опис, ціну, SKU, посилання на категорію), клас «`Category`» описує групи товарів, клас «`Customer`» – дані клієнта, клас «`Order`» – замовлення з датою, станом і прив'язкою до клієнта, а клас «`OrderItem`» – окремі позиції замовлення з посиланням на товар і кількість. Для цих моделей визначено навігаційні властивості, що задають зв'язки «категорія – товари», «клієнт – замовлення», «замовлення – позиції».

У каталозі даних реалізовано клас контексту бази даних, який успадковується від контексту Entity Framework Core і містить колекції сутностей для кожного з перелічених класів. Цей контекст відповідає за створення таблиць бази даних, накладання зовнішніх ключів та узгодження цілісності даних. Структура таблиць безпосередньо відображає модель предметної області: таблиця товарів містить посилання на таблицю категорій, таблиця замовлень – посилання на таблицю клієнтів, а таблиця позицій замовлення – посилання і на замовлення, і на товари.

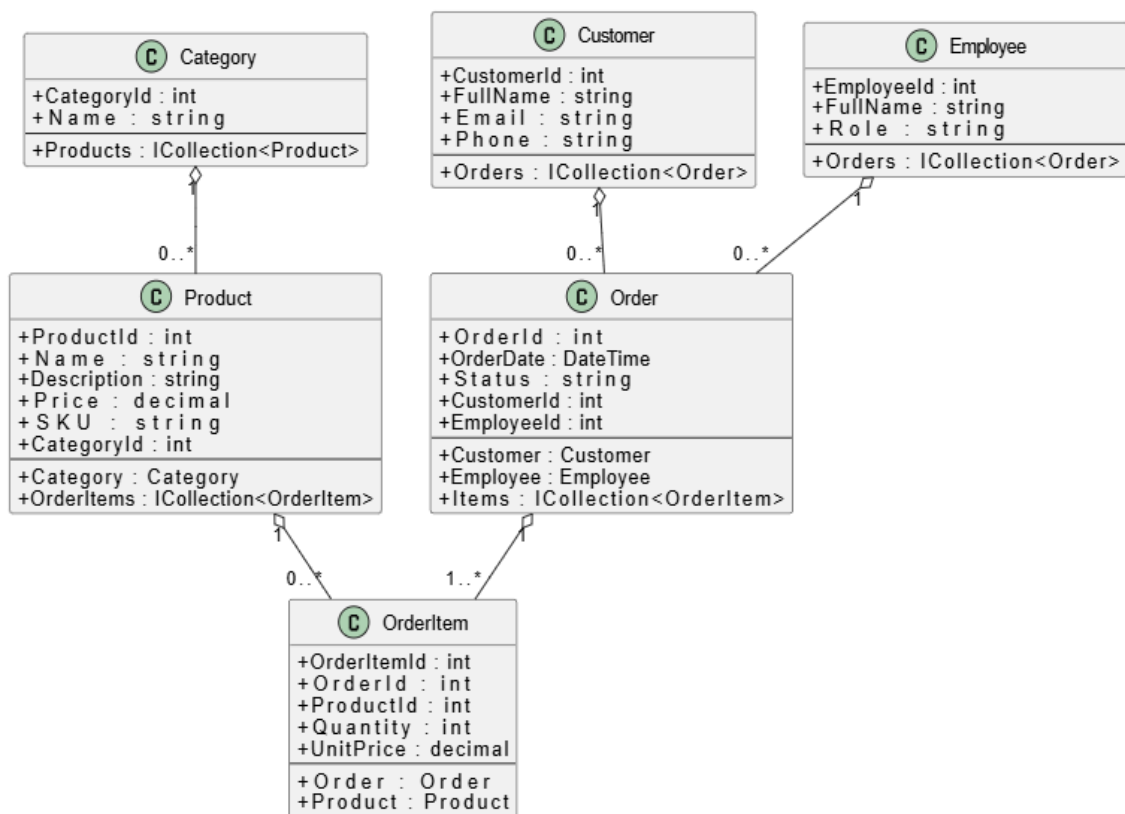


Рисунок 4.1 – Діаграма класів доменної моделі вебсистеми

4.2.2 Публічний інтерфейс

Публічна частина вебсистеми реалізує сценарії роботи клієнта з каталогом товарів. Для цього створено контролери, відповідальні за перегляд списку товарів і детальних сторінок, пошук та фільтрацію за категоріями. Кожен контролер отримує доступ до даних через контекст бази даних,

формує вибірки товарів і передає їх до Razor-подань у вигляді моделей подання. Подання каталогу відображає перелік товарів з основними атрибутами, а подання детальної сторінки – повний опис товару, включно з ціною, зображенням і кнопкою додавання до замовлення.

Для оформлення замовлення реалізовано окремий контролер, який обробляє введення контактних даних клієнта, формує об'єкт замовлення з набором позицій та зберігає його в базі даних. Відповідні подання забезпечують послідовні кроки: перегляд кошика, введення реквізитів, підтвердження замовлення. Усі ці сценарії спираються на ті самі моделі «Product», «Customer», «Order» і «OrderItem», що забезпечує узгодженість між відображенням інформації на сторінках і її збереженням у базі даних.

4.2.3 Інтерфейс для персоналу

Для роботи працівників магазину реалізовано окрему адміністративну область, що розміщена в каталозі Areas/Admin і має власні контролери та подання. Тут передбачено контролери керування довідковими сутностями та операційними сутностями. Кожен контролер адміністративної області надає повний набір CRUD-операцій: перегляд списку записів, створення, редагування та видалення, причому для замовлень додатково реалізовано зміну статусу й перегляд пов'язаних позицій (рис. 4.2).

Подання адміністративної частини побудовані як таблиці з можливістю переходу до форм редагування, у яких відображаються всі ключові атрибути відповідних моделей. Завдяки тому, що адміністратор працює з тими самими класами «Product», «Category», «Order» і «OrderItem», що й публічна частина, будь-які зміни в довідкових або операційних даних одразу відображаються на публічному вебресурсі. Таким чином, прототип вебсистеми реалізує повноцінний цикл роботи магазину техніки – від перегляду каталогу клієнтом до обробки замовлень персоналом.

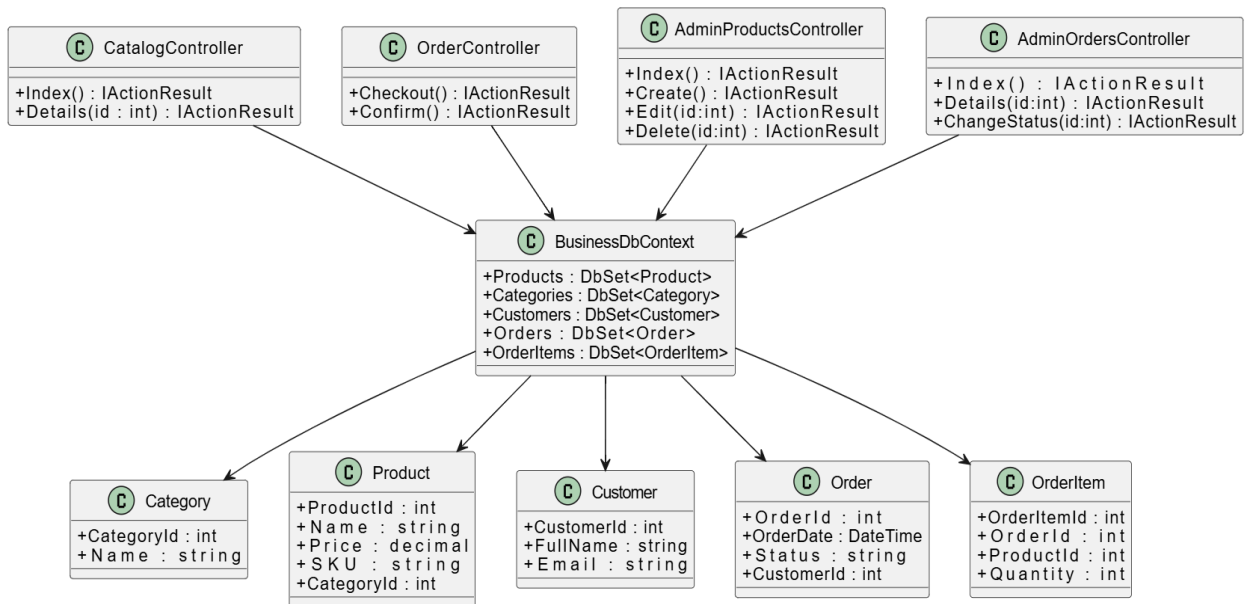


Рисунок 4.2 – Діаграма класів контролерів вебсистеми

4.3 Реалізація пресета та карти модулів

4.3.1 Відображення моделей вебсистеми на елементи пресета

Після реалізації вебсистеми-прототипу було виконано відображення її внутрішніх моделей на елементи пресета. Для кожного класу моделі вебзастосунку у настільному генераторі створюється окремий опис сутності у вигляді об'єкта класу «EntityDefinition». У цьому описі фіксуються логічне ім'я сутності, її тип – довідкова, операційна або користувацька – а також службові ознаки, необхідні для подальшого генерування компонентів вебсистеми.

Кожен атрибут класів вебмоделей відображається на об'єкт класу «AttributeDefinition», у якому задаються ім'я поля, тип даних, ознака обов'язковості, можливі обмеження та позначка, чи є цей атрибут ідентифікатором (наприклад, поле SKU для товару або slug для URL-адреси). Зв'язки між моделями вебзастосунку (такі як зв'язок між класом «Product» і класом «Category» або між класом «Order» і класом «OrderItem») представляються в пресеті об'єктами класу «RelationshipDefinition» із зазначенням пов'язаних сутностей і кратності.

У результаті формується пресет, який не дублює готовий код вебзастосунку, а містить його узагальнений опис у термінах сутностей, атрибутів і зв'язків. Саме цей опис використовується настільним застосунком для побудови карти модулів і подальшої генерації структури проєкту. Таким чином, пресет виступає формальним містком між предметною моделлю вебсистеми та механізмом шаблонного генерування.

4.3.2 Структура конфігурації

Для збереження конфігурації було обрано формат JSON, у якому представлено як сам пресет, так і карту модулів. У файлі пресета зберігаються списки сутностей, атрибутів, зв'язків і ролей користувачів; для кожної сутності вказуються її тип, набір атрибутів, правила валідації ідентифікаторів та службові ознаки. Файл пресета може зберігатися й завантажуватися настільним застосунком, що дозволяє повторно використовувати одну й ту саму конфігурацію для генерування або оновлення вебсистем.

Карта модулів у складі вебпроєкту містить інформацію про те, які саме компоненти мають бути створені для кожної сутності пресета. Для кожного запису в карті модулів фіксується ім'я сутності, список необхідних компонентів, а також додаткові параметри, що впливають на структуру згенерованих файлів.

4.4 Реалізація застосунку-генератора DeepCore

4.4.1 Моделі та сервіси генератора

У настільному застосунку-генераторі логіка роботи побудована навколо доменної моделі пресета, реалізованої у вигляді окремих класів. Клас «Preset» описує конфігурацію бізнес-системи загалом і містить колекції описів сутностей, атрибутів, зв'язків та ролей користувачів. Клас «EntityDefinition» представляє окрему сутність з логічним іменем, типом та

набором ознак, важливих для подальшого генерування. Для кожного поля сутності використовується клас «AttributeDefinition» з інформацією про назву, тип даних, обов'язковість та статус ідентифікатора (наприклад SKU або slug). Зв'язки між сутностями зберігаються у вигляді об'єктів класу «RelationshipDefinition», де фіксуються пов'язані сутності та тип зв'язку.

Окремим елементом моделі є клас «ModuleMap», який відображає елементи пресета на артефакти вебпроєкту: для кожної сутності вказується, чи потрібна таблиця бази даних, модель у вебзастосунку, контролер, комплект подань та елементи адміністративного інтерфейсу. Роботу з цими моделями інкапсулюють сервіси, розміщені в окремому каталозі. Сервіс збереження пресетів відповідає за серіалізацію об'єктів класів «Preset» і «ModuleMap» у формат JSON та їх завантаження з файлів, сервіс валідації перевіряє коректність опису, а сервіс журналювання фіксує помилки й попередження, що виникають у процесі редагування та генерації.

4.4.2 Графічний інтерфейс генератора

Графічний інтерфейс настільного застосунку реалізовано на основі WPF як віконну структуру з розподілом на функціональні області. Головне вікно – клас «MainWindow» – містить панель навігації за сутностями пресета, область редагування атрибутів і зв'язків для вибраної сутності, а також блок налаштувань цільового вебпроєкту. Для роботи з кожним елементом пресета використовуються спеціалізовані елементи керування: таблиця для переліку сутностей, форми введення для атрибутів, списки для зв'язків і ролей користувачів.

Події інтерфейсу напряду пов'язані з доменною моделлю: вибір сутності оновлює відображення її атрибутів, редагування полів одразу змінює відповідні об'єкти класів «EntityDefinition» та «AttributeDefinition», а натискання кнопок «Зберегти пресет» і «Завантажити пресет» викликає сервіси серіалізації. Кнопка запуску генерації використовує поточний стан пресета та карти модулів як вхідні дані для шаблонного механізму. Така

організація інтерфейсу дозволяє поєднати редагування конфігурації, її валідацію й запуск процесу генерування в межах одного робочого вікна без необхідності оперувати безпосередньо файлами чи параметрами командного рядка.

4.4.3 Шаблонний механізм формування вебпроєкту

Шаблонний механізм генерації реалізовано як окремий модуль, що працює з текстовими шаблонами файлів проєкту. У каталозі шаблонів зберігаються файли-каркаси контролерів, моделей, Razor-подань та конфігураційних елементів, у яких зарезервовано позиції для підстановки імен сутностей, списків атрибутів, назв маршрутів та інших параметрів (рис. 4.3). Під час запуску генерації модуль читає пресет і карту модулів, для кожної сутності визначає необхідний набір артефактів та формує на основі відповідних шаблонів конкретні файли з підставленими значеннями.

Алгоритм роботи механізму включає послідовні кроки: побудову внутрішньої моделі контексту для кожної сутності, вибір потрібних шаблонів, підстановку даних і запис файлів у структуру каталогу вебпроєкту. При цьому підтримується можливість часткового оновлення: якщо певні файли вже існують, генератор може або перезаписати їх повністю, або оновити окремі фрагменти, що відповідають автоматизованій частині, залишаючи ручні зміни розробника. Завдяки цьому настільний застосунок не лише створює початкову версію вебсистеми, а й дозволяє повторно застосовувати пресети для модифікації або розгортання нових інстансів бізнес-систем з однаковою структурою.

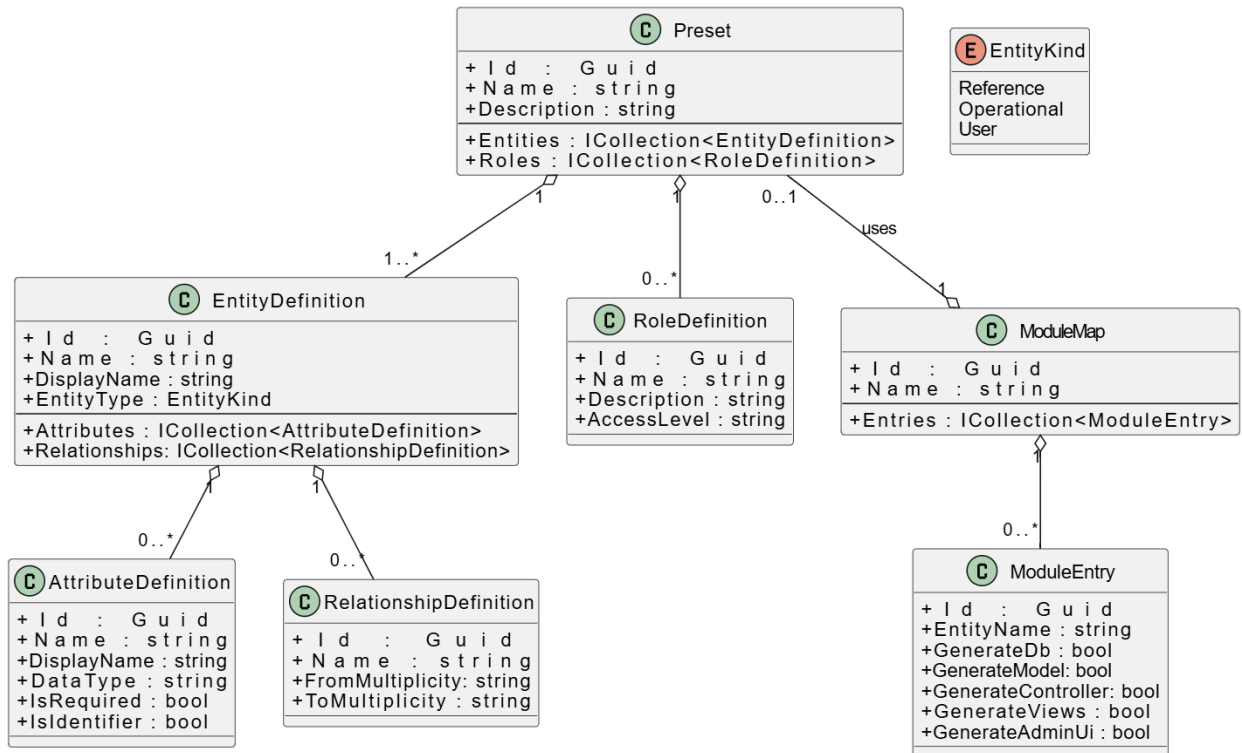


Рисунок 4.3 – Діаграма класів доменної моделі пресета в DeepCore

4.5 Висновки за розділом

У четвертому розділі було реалізовано повний програмний комплекс, який складається з вебсистеми та настільного застосунку-генератора DeepCore. Описано побудову предметної моделі, контексту бази даних і вебінтерфейсів публічної та адміністративної частин, що разом забезпечують повний цикл роботи бізнес-системи – від перегляду каталогу до обробки замовлень. На основі цих моделей сформовано пресет і карту модулів, які стали формальним описом конфігурації для подальшої автоматизованої генерації.

Настільний застосунок DeepCore реалізує доменну модель пресета, сервіси збереження, валідації та шаблонний механізм формування файлів вебпроєкту, що дає змогу генерувати або оновлювати структуру ASP.NET MVC-застосунку на основі конфігурації, заданої користувачем.

5 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

5.1 Експлуатація програмного комплексу

5.1.1 Підготовка середовища

Для експлуатації програмного комплексу необхідно мати встановлене середовище розробки Microsoft Visual Studio і веббраузер для роботи зі згенерованою веборієнтованою системою. Вебчастина розгортається як типове ASP.NET MVC-рішення: відкривається відповідний проєкт, виконується ініціалізація бази даних, після чого вебзастосунок запускається у вбудованому вебсервері й стає доступним за локальною адресою. Настільний застосунок DeerpCore встановлюється або запускається як окремий Windows-застосунок і використовує спільну з вебпроєктом файлову структуру для розміщення пресетів, карти модулів та згенерованих файлів.

5.1.2 Робота з настільним генератором DeerpCore

Експлуатація генератора передбачає послідовність дій користувача, яка повторюється щоразу при створенні або зміні конфігурації бізнес-системи. Спочатку користувач створює новий пресет або відкриває наявний файл конфігурації, після чого за допомогою елементів інтерфейсу формує перелік сутностей, їхніх атрибутів, зв'язків та ролей користувачів. Далі виконується валідація пресета, за результатами якої користувач або усуває виявлені помилки, або переходить до вибору каталогу цільового вебпроєкту і запуску генерації структури вебсистеми.

Узагальнений алгоритм використання програмного забезпечення подано у вигляді схеми на рис. 5.1.

Головне вікно настільного застосунку відображає список сутностей пресета, панель редагування атрибутів і зв'язків, а також елементи керування вибором каталогу вебпроєкту та запуском процесу генерації.

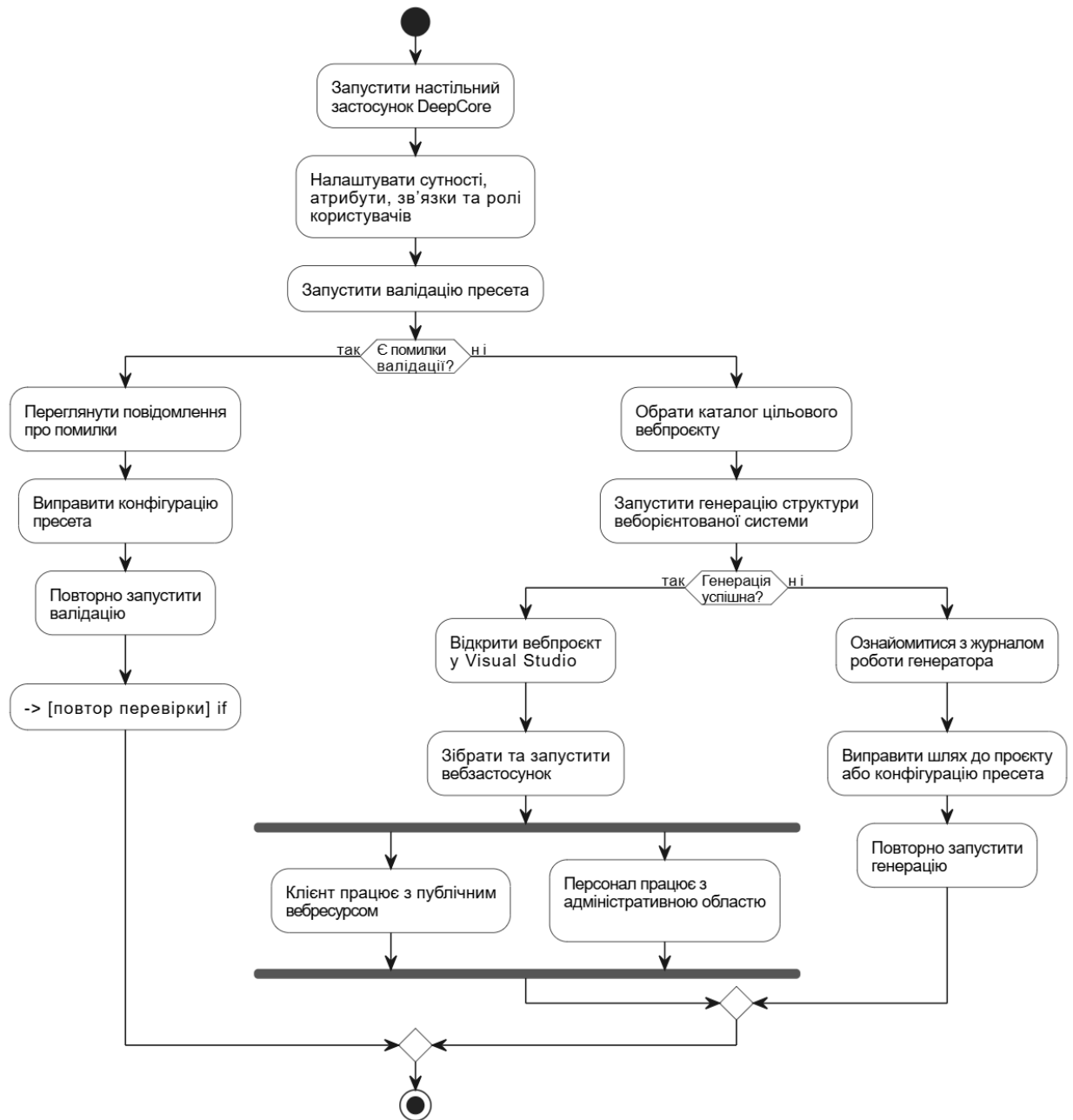


Рисунок 5.1 – Алгоритм використання програмного забезпечення

На рис. 5.2 наведено приклад головного вікна DeepCore з основними елементами інтерфейсу, які використовуються під час повсякденної роботи користувача.

Оберіть модулі: ☐ Каталог товарів ☐ Замовлення / корзини ☐ Користувачі та ролі ☐ Звіти та аналітика Оберіть типи акаунтів: ☐ Клієнт ☐ Адміністратор ☐ Менеджер	Назва проєкту / рішення	Контактний email
	Назва вебсистеми (заголовок сайту)	Контактний телефон
	Шлях до папки вебпроєкту	Адреса компанії
	Назва бізнесу / компанії	Текст ліцензії / копірайту
	Шлях до файлу імпорту товарів (CSV)	Шлях до папки з зображеннями товарів

Рисунок 5.2 – Головне меню застосунку-генератору DeepCore

5.1.3 Інструкція з експлуатації для користувача

Експлуатація ПЗ проводиться виконанням наступних кроків:

- запустити настільний застосунок DeepCore. У полі «Назва проєкту / рішення» ввести логічну назву рішення, у межах якого буде розгортатися вебсистема;
- у полі «Назва вебсистеми» задати текст, який відображатиметься як основний заголовок вебресурсу;
- у полі «Шлях до папки вебпроєкту» вказати повний шлях до кореневої папки вебзастосунку ASP.NET MVC, у який потрібно згенерувати структуру;
- у полі «Назва бізнесу / компанії» ввести назву організації, що використовуватиметься у підписах та текстах вебресурсу.
- у полях «Контактний email», «Контактний телефон» та «Адреса компанії» ввести відповідні контактні дані;

- у полі «Текст ліцензії / копірайту» ввести текст, який буде відображатися у нижній частині сторінок;
- у полі «Шлях до файлу імпорту товарів (CSV)» вказати шлях до файлу з початковим переліком товарів, який має бути використаний для заповнення бази даних, а у полі «Шлях до папки з зображеннями товарів» – каталог, що містить відповідні зображення;
- у блоці «Оберіть модулі» відмітити галочками ті функціональні модулі, які мають бути згенеровані;
- у блоці «Оберіть типи акаунтів» визначити, які типи облікових записів будуть доступні в системі, залишивши необхідні ролі позначеними;
- після заповнення всіх полів натиснути кнопку «START».

Застосунок виконує перевірку коректності введених даних, формує пресет і карту модулів та генерує/оновлює структуру вебпроєкту у вказаній папці; у разі успішного завершення користувач може перейти до тестування згенерованої вебсистеми.

5.2 Тестування програмного забезпечення

Проведемо тестування програмного комплексу, яке відтворює типову послідовність дій користувача: від налаштування пресета в генераторі DeerCore до роботи з отриманою вебсистемою. Мета тесту – підтвердити, що за заданою конфігурацією успішно формується працездатний вебзастосунок.

5.2.1 Генерація вебсистеми

Підготовка пресета та запуск генерації: у головному вікні DeerCore були заповнені всі поля конфігурації, обрані потрібні модулі й типи акаунтів. Після натискання кнопки START застосунок сформував пресет і карту модулів, згенерував структуру вебпроєкту у вказаній папці; успішне завершення операції підтверджено відсутністю помилок. На рисунку 5.3 зафіксовано заповнене головне вікно та результат генерації.

Оберіть модулі:

☒ Каталог товарів

☒ Замовлення / корзини

☒ Користувачі та ролі

☐ Звіти та аналітика

Оберіть типи акаунтів:

☒ Клієнт

☒ Адміністратор

☐ Менеджер

BusinessSystem	demo@business.local
Demo Business System	+380000000000
C:\Projects\BusinessSystem	Demo city, Demo street 1
Demo Business	© Demo Business, 2025
C:\Projects\BusinessSystem\Data\demo-products.csv	C:\Projects\BusinessSystem\wwwroot\img

START

Рисунок 5.3 – Головне меню генератора із задіяними налаштуваннями

5.2.2 Запуск і перевірка публічної та адміністративної частин вебсистеми

Згенерований вебпроект було відкрито у Visual Studio, виконано збірку й запуск. У браузері перевірено завантаження головної сторінки, відображення каталогу товарів, коректність переходу на сторінку окремого товару та роботу корзини. Було оформлено тестове замовлення з введенням контактних даних клієнта; система сформувала підтвердження й запис замовлення в базу даних, що зафіксовано на скріншотах публічної частини.

Далі виконано вхід до admin-області під обліковим записом працівника, перевірено відображення списку товарів і замовлень. Створено новий тестовий товар, змінено ціну існуючого, оновлено статус щойно оформленого замовлення. Усі зміни коректно відобразилися в інтерфейсі й у базі даних; на рисунках демонструються форми редагування та таблиці admin-інтерфейсу.

На рисунку 5.4 наведено публічну сторінку каталогу з фільтрами та картками послуг, згенеровану на основі пресета.

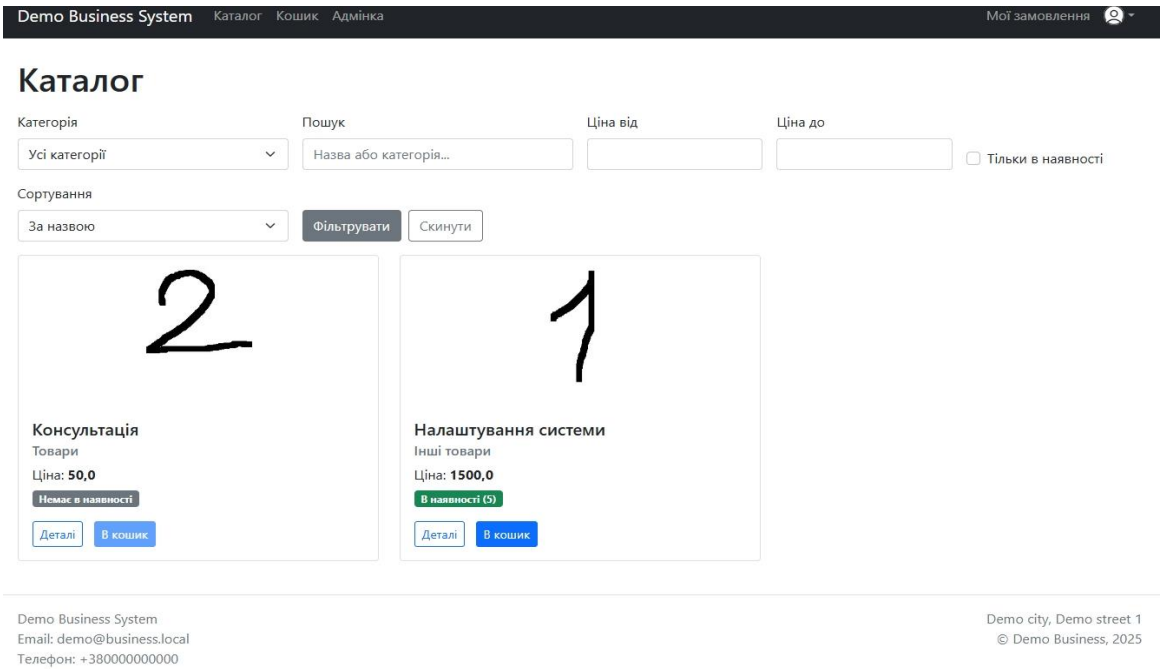


Рисунок 5.4 – Вебсторінка каталогу магазину

На рисунку 5.5 показано сторінку кошика з підсумком замовлення перед переходом до оформлення.



Рисунок 5.5 – Вебсторінка кошику

На рисунку 5.6 наведено адміністративний інтерфейс керування товарами з можливістю додавання, редагування та видалення позицій каталогу.

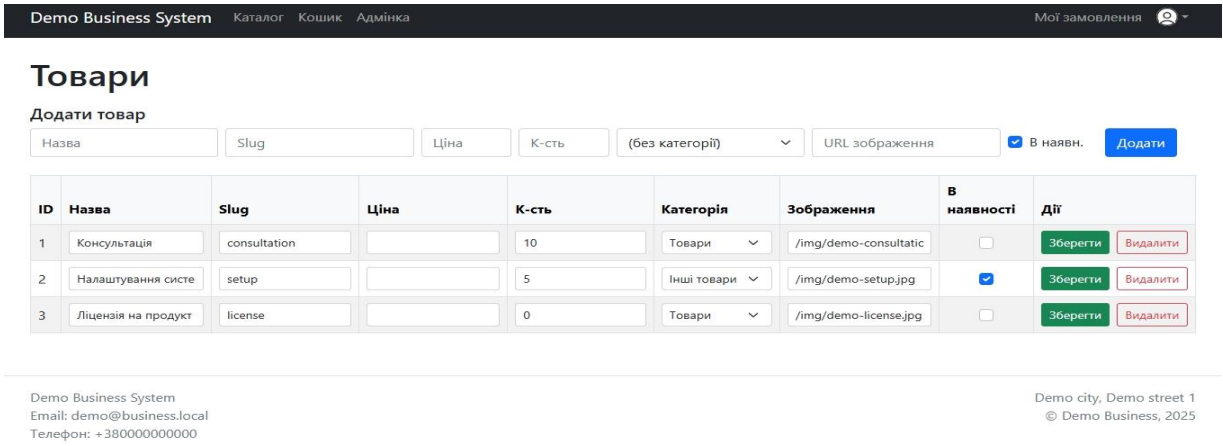


Рисунок 5.6 – Адміністративний інтерфейс керування товарами

На рисунку 5.7 показано форму оформлення замовлення, де клієнт вводить контактні дані для підтвердження покупки.

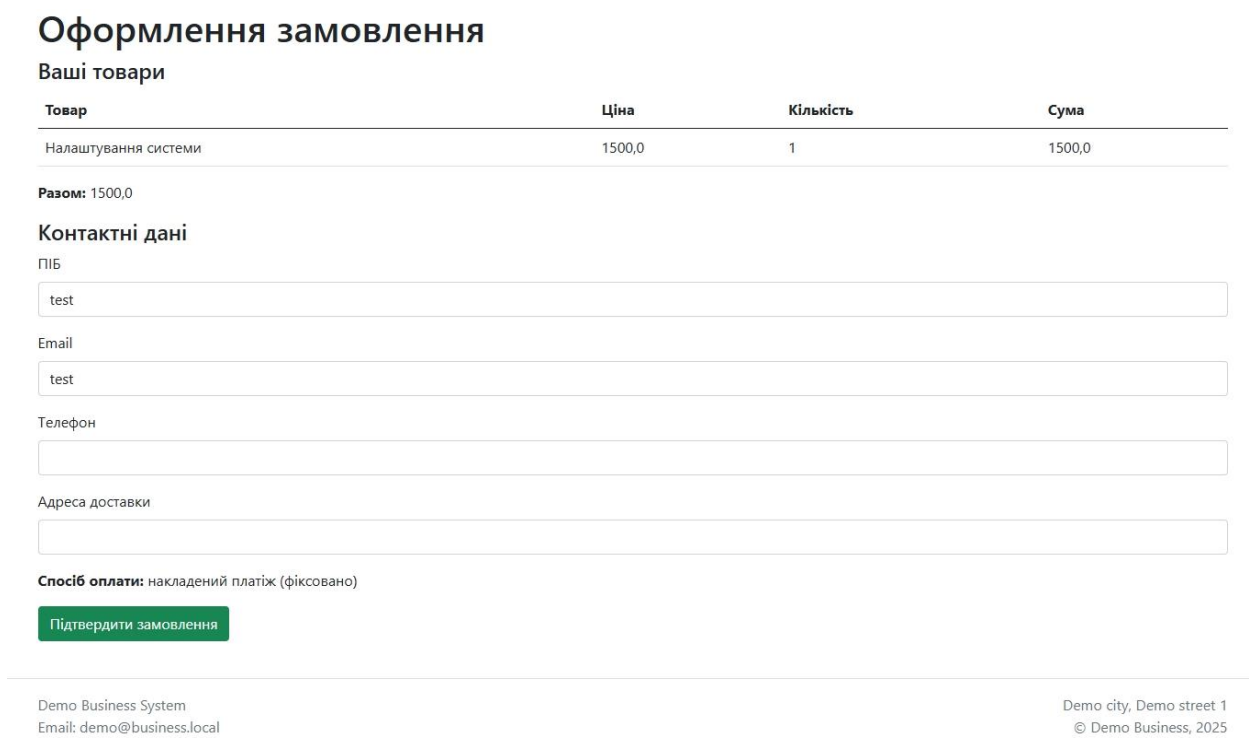


Рисунок 5.7 – Форма оформлення замовлення

На рисунку 5.8 наведено сторінку «Мої замовлення», яка відображає історію замовлень поточного користувача.

Мої замовлення

ID	Дата	Статус	Сума
7	05.12.2025 3:10:49	New	1500,0
6	30.11.2025 18:36:19	New	6000,0
4	30.11.2025 16:01:35	New	2500,0

Demo Business System
Email: demo@business.local
Телефон: +380000000000

Demo city, Demo street 1
© Demo Business, 2025

Рисунок 5.8 – Вебсторінка замовлень клієнта

На рисунку 5.9 показано сторінку профілю користувача з відображенням основних даних облікового запису та переходом до списку замовлень.

Мій профіль

 test

Зареєстрований: 30.11.2025 16:01:03

Email: test

Роль: Admin

Мої замовлення

Demo Business System
Email: demo@business.local
Телефон: +380000000000

Demo city, Demo street 1
© Demo Business, 2025

Рисунок 5.9 – Вебсторінка профілю користувача

5.3 Експериментальне дослідження. Отримання результатів, аналіз ефективності

Узагальнимо експериментальне дослідження ефективності розробленого генератора. Згідно з підрозділом 1.5.3 базовим критерієм є скорочення сумарних часових витрат на створення веборієнтованих бізнес-систем у порівнянні з традиційним підходом, що описується формулами (1.1) та (1.2).

У методичному підрозділі 2.6 для експерименту обрано один репрезентативний пресет інтернет-магазину, а показниками є оцінка часу

повної ручної розробки системи, час конфігурування пресета та запуску генерації та різниця у витраченому часі, порівнюючи обидва підходи.

Для оцінки $T_{\text{баз}}$ процес ручної розробки системи розкладено на етапи, що відповідають структурі реалізованого у розділі 4 прототипу:

- проєктування моделі даних і структури БД;
- реалізація моделей і контексту доступу до даних;
- створення контролерів і подань публічної частини;
- реалізація адміністративної області для роботи персоналу;
- налаштування навігації, авторизації та базових перевірок,

початкове тестування;

На основі досвіду розробки аналогічних навчальних проєктів і трудомісткості реалізованого прототипу ці етапи оцінені у 6–8 нормо-годин кожен; сумарно це дає приблизно 40 годин для одного інтернет-магазину. Таким чином, для одного проєкту базові часові витрати становлять $T_{\text{баз}} \approx 40$ год.

Показник часу $T_{\text{ген}}$ вимірювався за сценарієм з підрозділу 2.6: конфігурування тестового пресета в настільному застосунку DeepCore, запуск генерації вебпроєкту й первинне розгортання прототипу з перевіркою працездатності базових сценаріїв каталогу та оформлення замовлення. Для цього циклу, виконаного одним користувачем, сумарний час склав близько $T_{\text{ген}} \approx 8$ годин, що включає повторні запуски генерації після дрібних коригувань пресета. Отримане значення можна інтерпретувати як $T_{\text{конф}}$ у формулі (1.2) для одного екземпляра системи.

Отже, для одиничної системи генератор скорочує трудомісткість приблизно на $\Delta T \approx 32$ години, або близько 80 % від базового підходу.

Щоб оцінити повноцінний сценарій промислового впровадження, введемо окремо час створення самого генератора. Припустимо, що над платформою працює команда з двої розробників протягом одного року. За умови близько 1600 робочих годин на людину це дає $T_{\text{ядро}} \approx 3200$ робочих годин. Тоді сумарні витрати часу на традиційну розробку n подібних систем

описуються залежністю $T_{\text{баз}}(n) = n \cdot T_{\text{баз}}$, а для підходу з генератором – $T_{\text{ген}}(n) = T_{\text{ядро}} + n \cdot T_{\text{ген}}$.

Точка, у якій підхід з генератором стає вигіднішим за базовий, визначається з нерівності $T_{\text{ген}}(n) < T_{\text{баз}}(n)$. Підставляючи числові значення ($T_{\text{ядро}} = 3200$ год, $T_{\text{баз}} = 40$ год, $T_{\text{ген}} = 8$ год), одержуємо $n > 3200 / (40 - 8) \approx 100$. Це означає, що на короткому відрізку для певної кількості перших проєктів традиційний підхід виглядає економічно простішим та надійнішим, але починаючи приблизно з соті згенерованої системи сумарні витрати часу при використанні генератора стають меншими, а далі розрив лише зростає.

Графічно ці залежності можуть бути представлені у вигляді двох ліній на координатній площині «кількість систем n – сумарні трудові витрати T »: пряма $T_{\text{баз}}(n) = 40n$ проходить через початок координат, тоді як пряма $T_{\text{ген}}(n) = 3200 + 8n$ стартує вище, але має значно менший нахил. У точці $n \approx 100$ ці лінії перетинаються: лівіше від цієї точки нижче розташована крива базового підходу, правіше – крива підходу з генератором, що наочно демонструє зміну економічної доцільності залежно від масштабу використання платформи (рис. 5.10).

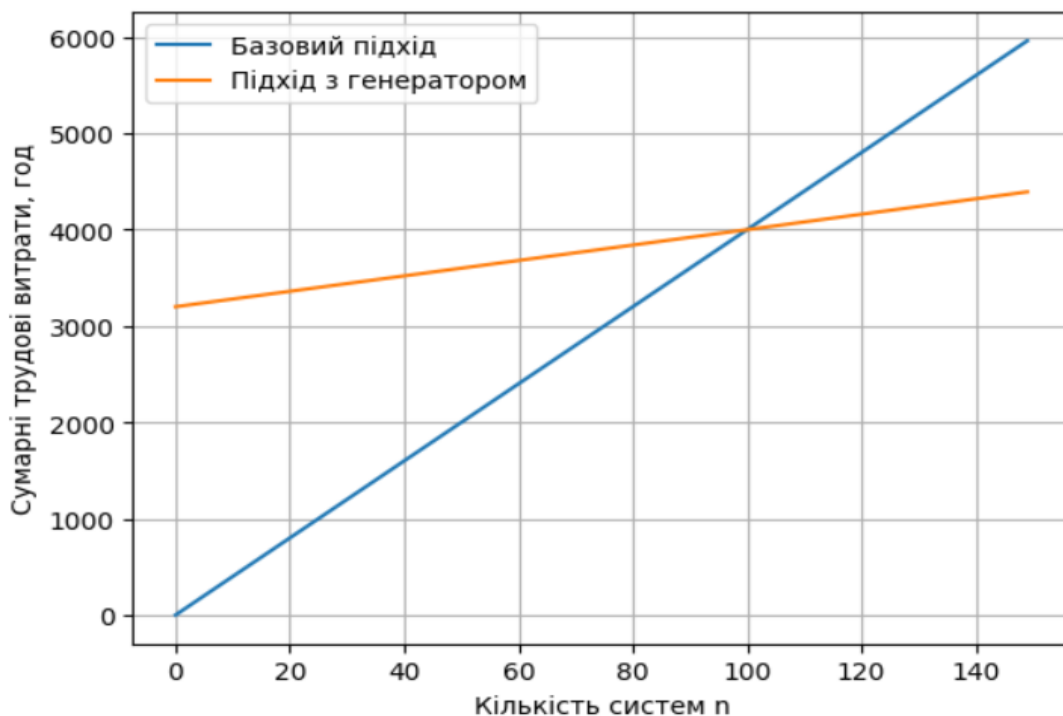


Рисунок 5.10 – Порівняльний графік методів

5.4 Висновки за розділом

У розділі було описано умови експлуатації програмного забезпечення, послідовність роботи користувача в настільному генераторі DeerCore та процедуру розгортання згенерованої веборієнтованої системи. Проведене тестування показало, що за заданим пресетом коректно формується повноцінний вебзастосунок із каталогом товарів, кошиком, оформленням замовлення, адміністративною областю для керування товарами та перегляду історії замовлень, причому всі основні сценарії роботи відпрацьовують без збоїв.

Експериментальне дослідження продемонструвало, що трудомісткість створення однієї системи за допомогою генератора зменшується приблизно з 40 до 8 нормо-годин, тобто орієнтовно на 80%. З урахуванням витрат на розробку ядра генератора (приблизно 3200 годин) встановлено порогове значення близько 100 згенерованих систем, після якого сумарні трудові витрати підходу з генератором стають меншими, ніж при традиційній розробці, а подальше збільшення кількості систем лише посилює цей ефект.

Таким чином, експеримент підтверджує як працездатність запропонованого програмного комплексу, так і його потенційну економічну доцільність у разі багаторазового використання для типових веборієнтованих бізнес-систем.

ВИСНОВКИ

У результаті дослідження опрацьовано сучасні підходи до проєктування та впровадження веборієнтованих бізнес-систем і визначено ключові проблеми їх розробки у випадках, коли організаціям потрібні не поодинокі, а серійні рішення з подібною структурою. Показано, що традиційний підхід із ручним створенням кожної системи окремо призводить до повторюваних витрат часу на типові модулі, налаштування ролей, базові CRUD-сценарії, конфігурації та узгодження структури даних. Це формує потребу в інструменті, який дозволяє стандартизувати та пришвидшувати запуск бізнес-систем без втрати можливості адаптації під конкретні вимоги.

Розроблено концепцію та архітектуру програмного комплексу генераторного типу, орієнтованого на моделювання, налаштування та експлуатацію бізнес-систем. Запропонований підхід базується на використанні пресетів і шаблонів, що дає змогу формалізувати склад системи, структуру модулів, організацію даних і базові правила взаємодії компонентів. Реалізовано програмний прототип, який поєднує настільну частину генератора та цільовий вебзастосунок. У межах реалізації забезпечено формування каркаса вебсистеми, налаштування типових сутностей, ролей і ключових сценаріїв роботи.

Проведене тестування підтвердило коректність основних функцій генерації та стабільність запуску сформованого вебзастосунку. Отримані результати свідчать, що застосування генераторного підходу є доцільним у середовищах, де потрібне багаторазове створення однотипних або близьких за структурою бізнес-рішень. У таких умовах зменшується трудомісткість стартового етапу, підвищується узгодженість архітектурних рішень і покращується керованість конфігурацій.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. European Investment Fund. EIF Working Paper 2024/101 [Electronic resource] // European Investment Fund. – Access mode: https://www.eif.org/news_centre/publications/eif-working-paper-2024-101.pdf.
2. Di Bella, L. Annual Report on European SMEs 2023 [Electronic resource] / L. Di Bella, A. Katsinis, J. Lagüera-González, L. Odenthal, M. Hell, B. Lozar // European Commission. – Access mode: https://single-market-economy.ec.europa.eu/system/files/2023-08/Annual%20Report%20on%20European%20SMEs%202023_FINAL.pdf.
3. Ko, A. Small business website statistics [Electronic resource] / A. Ko // Wix Blog. – Access mode: <https://www.wix.com/blog/small-business-website-statistics>.
4. Adnyana, R. The ultimate list of online business statistics [Electronic resource] / R. Adnyana // WPForms Blog. – Access mode: <https://wpforms.com/the-ultimate-list-of-online-business-statistics/>.
5. Luca, M. Getting on the Map: The Impact of Online Listings on Business Performance [Electronic resource] / M. Luca, A. Nagaraj, G. Subramani // Harvard Business School Working Paper. – Access mode: https://www.hbs.edu/ris/Publication%20Files/23-031_d3d279ef-59ce-43f8-9917-0387ae133a06.pdf.
6. Sharabati, A.-A. A. The Impact of Digital Marketing on the Performance of SMEs [Electronic resource] / A.-A. A. Sharabati, A. A. A. Ali, M. I. Allahham, A. A. Hussein, A. F. Alheet, A. S. Mohammad // Sustainability. – Access mode: <https://www.mdpi.com/2071-1050/16/19/8667>.
7. Leadpages. Average cost of website design for small business [Electronic resource] // Leadpages Blog. – Access mode: <https://www.leadpages.com/blog/average-cost-of-website-design-for-small-business>.

8. Rico, A. R. Website statistics [Electronic resource] / A. R. Rico // Fit Small Business. – Access mode: <https://fitsmallbusiness.com/website-statistics/>.
9. Rendorio, I. Surprising small business website stats [Electronic resource] / I. Rendorio // Podium. – Access mode: <https://www.podium.com/article/surprising-small-business-website-stats>.
10. Cin7 Team. Essential inventory management guide for SMB manufacturers [Electronic resource] // Cin7 Blog. – Access mode: <https://www.cin7.com/blog/essential-inventory-management-guide-for-smb-manufacturers/>.
11. Panko, R. R. Spreadsheet Errors: What We Know. What We Think We Can Do [Electronic resource] / R. R. Panko // arXiv. – Access mode: <https://arxiv.org/pdf/0802.3457>.
12. PlanGuru. Spreadsheet errors [Electronic resource] // PlanGuru Blog. – Access mode: <https://www.planguru.com/blog/88-spreadsheets-errors/>.
13. Indeed Editorial Team. Small business owner responsibilities [Electronic resource] // Indeed Career Guide. – Access mode: <https://uk.indeed.com/career-advice/career-development/small-business-owner-responsibilities>.
14. Small Business Choice. Business owner responsibilities [Electronic resource] // Small Business Choice Blog. – Access mode: <https://www.smallbusinesschoice.com/blog/business-owner-responsibilities>.
15. Tidio. Multichannel customer service [Electronic resource] // Tidio Blog. – Access mode: <https://www.tidio.com/blog/multichannel-customer-service/>.
16. OECD. The digital transformation of SMEs [Electronic resource] // OECD Publishing. – Access mode: https://www.oecd.org/en/publications/the-digital-transformation-of-smes_bdb9256a-en.html.
17. SiteMile. How many hours does it take to make a website? [Electronic resource] // SiteMile Blog. – Access mode: <https://sitemile.com/how-many-hours-does-it-take-to-make-a-website/>.

18. MobileAppDaily. Website development cost [Electronic resource] // MobileAppDaily. – Access mode: <https://www.mobileappdaily.com/knowledge-hub/website-development-cos>.
19. Wix. Wix Premium plans [Electronic resource] // Wix Blog. – Access mode: <https://www.wix.com/blog/wix-premium-plans>.
20. Golden Launch Creative. How much is a Squarespace website? [Electronic resource] // Golden Launch Creative Blog. – Access mode: <https://www.goldenlaunchcreative.com/blog/how-much-is-a-squarespace-website>.
21. Microsoft. Implement CRUD functionality with ASP.NET Core MVC and Entity Framework [Electronic resource] // Microsoft Learn. – Access mode: <https://learn.microsoft.com/en-us/aspnet/core/data/ef-mvc/crud?view=aspnetcore-10.0>.
22. Leadpages. Average cost of website design for small business [Electronic resource] // Leadpages Blog. – Access mode: <https://www.leadpages.com/blog/average-cost-of-website-design-for-small-business>.
23. Microsoft. Implement CRUD functionality with ASP.NET Core MVC and Entity Framework [Electronic resource] // Microsoft Learn. – Access mode: <https://learn.microsoft.com/en-us/aspnet/core/data/ef-mvc/crud?view=aspnetcore-10.0>.

ДОДАТОК А
Текст програми

```

using System.IO;
using System.Text.Json;
using DeepCore.Models;
namespace DeepCore.TemplateEngine
{
    public static class TemplateService
    {
        // корінь шаблону
        private const string TemplateRoot =
@"C:\Users\username\source\repos\BusinessSystem";

        public static void Generate(ProjectConfig config)
        {
            var targetRoot = Path.Combine(config.OutputPath,
config.ProjectName);

            // 1. Копіювання
            CopyDirectory(TemplateRoot, targetRoot);

            // 2. Підставка токенів в текстових файлах
            ReplaceTokens(targetRoot, config);

            // 3. Запис налаштувань
            WriteSettingsJson(targetRoot, config);

            // 4. Запис ролей адмін-сектору
            WriteRolesJson(targetRoot, config);

            // 5. Копія CSV й підтягування картинок товарів
            ImportProductsAssets(targetRoot, config);

            // 6. Вимкнення зайвих модулів
            ApplyModules(targetRoot, config);
        }

        private static void CopyDirectory(string sourceDir, string targetDir)
        {
            Directory.CreateDirectory(targetDir);

            foreach (var file in Directory.GetFiles(sourceDir))
            {
                var dest = Path.Combine(targetDir, Path.GetFileName(file));
                File.Copy(file, dest, overwrite: true);
            }
        }
    }
}

```

```

    }

    foreach (var dir in Directory.GetDirectories(sourceDir))
    {
        var dest = Path.Combine(targetDir, Path.GetFileName(dir));
        CopyDirectory(dir, dest);
    }
}

private static void ReplaceTokens(string root, ProjectConfig config)
{
    foreach (var file in Directory.GetFiles(root, ".*.*",
SearchOption.AllDirectories))
    {
        var ext = Path.GetExtension(file).ToLowerInvariant();
        if (ext is not (".cs" or ".cshtml" or ".json" or ".config" or
".cspj"))
            continue;

        var text = File.ReadAllText(file);

        text = text
            .Replace("__PROJECT_TITLE__", config.SiteTitle)
            .Replace("__PROJECT_NAME__", config.ProjectName)
            .Replace("__BUSINESS_NAME__", config.BusinessName);

        File.WriteAllText(file, text);
    }
}

private static void WriteSettingsJson(string targetRoot, ProjectConfig
config)
{
    var settingsPath = Path.Combine(targetRoot,
"deepcore.settings.json");
    var settingsJson = JsonSerializer.Serialize(new
    {
        config.BusinessName,
        config.Email,
        config.Phone,
        config.Address,
        config.LicenseInfo
    }, new JsonSerializerOptions { WriteIndented = true });

```

```

        File.WriteAllText(settingsPath, settingsJson);
    }

    private static void WriteRolesJson(string targetRoot, ProjectConfig
config)
    {
        var rolesPath = Path.Combine(targetRoot, "deepcore.roles.json");
        var rolesJson = JsonSerializer.Serialize(config.AdminAreaRoles,
            new JsonSerializerOptions { WriteIndented = true });

        File.WriteAllText(rolesPath, rolesJson);
    }

    private static void ImportProductsAssets(string targetRoot,
ProjectConfig config)
    {
        // CSV
        if (!string.IsNullOrEmpty(config.ProductsCsvPath) &&
            File.Exists(config.ProductsCsvPath))
        {
            File.Copy(
                config.ProductsCsvPath,
                Path.Combine(targetRoot, "deepcore.items.csv"),
                overwrite: true);
        }

        // Зображення
        if (!string.IsNullOrEmpty(config.ProductImagesPath) &&
            Directory.Exists(config.ProductImagesPath))
        {
            var destImgs = Path.Combine(targetRoot, "wwwroot", "images",
"products");
            CopyDirectory(config.ProductImagesPath, destImgs);
        }
    }

    // опис модулів
    private class ModuleMap
    {
        public string Id { get; set; } = "";
        public List<string> Paths { get; set; } = new();
    }

    private static void ApplyModules(string targetRoot, ProjectConfig
config)

```

```

    {
        var mapPath = Path.Combine(targetRoot, "modules.map.json");
        if (!File.Exists(mapPath)) return;

        var json = File.ReadAllText(mapPath);
        var modules = JsonSerializer.Deserialize<List<ModuleMap>>(json) ?? new();

        var disabled = new List<string>();
        if (!config.UseCatalog) disabled.Add("Catalog");
        if (!config.UseOrders) disabled.Add("Orders");
        if (!config.UseAccounts) disabled.Add("Accounts");
        if (!config.UseReports) disabled.Add("Reports");

        foreach (var m in modules.Where(m => disabled.Contains(m.Id)))
        {
            foreach (var rel in m.Paths)
            {
                var full = Path.Combine(targetRoot, rel);
                if (Directory.Exists(full)) Directory.Delete(full, true);
                else if (File.Exists(full)) File.Delete(full);
            }
        }
    }
}

using BusinessSystem.Data;
using BusinessSystem.Services;
using Microsoft.AspNetCore.Authentication.Cookies;
using Microsoft.EntityFrameworkCore;
var builder = WebApplication.CreateBuilder(args);
// MVC + БД
builder.Services.AddControllersWithViews();
builder.Services.AddDbContext<AppDbContext>(options =>

options.UseSqlite(builder.Configuration.GetConnectionString("DefaultConnection
"))));

```

```

// Cookie-auth
builder.Services.AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme)
    .AddCookie(options =>
    {
        options.LoginPath = "/Account/Login";
    });
builder.Services.AddHttpContextAccessor();
builder.Services.AddScoped<CartService>();
var app = builder.Build();
// Ініціалізація БД
using (var scope = app.Services.CreateScope())
{
    var db = scope.ServiceProvider.GetRequiredService<AppDbContext>();
    db.Database.EnsureCreated();
    DbInitializer.Seed(db);
}
// Pipeline
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Home/Error");
    app.UseHsts();
}

app.UseHttpsRedirection();
app.UseStaticFiles();
app.UseRouting();
app.UseAuthentication();
app.UseAuthorization();
app.MapControllerRoute(
    name: "areas",

```

```

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

namespace BusinessSystem.Data
{
    using BusinessSystem.Models;
    using Microsoft.EntityFrameworkCore;

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

        public DbSet<Category> Categories => Set<Category>();
        public DbSet<Item> Items => Set<Item>();
        public DbSet<Order> Orders => Set<Order>();
        public DbSet<OrderItem> OrderItems => Set<OrderItem>();
        public DbSet<Setting> Settings => Set<Setting>();
        public DbSet<User> Users => Set<User>();

        public DbSet<Cart> Carts => Set<Cart>();
        public DbSet<CartItem> CartItems => Set<CartItem>();
    }
}

using BusinessSystem.Models;
using System.Text.Json;
using System.IO;

```

```

namespace BusinessSystem.Data
{
    public static class DbInitializer
    {

        public static void Seed(AppDbContext db)
        {
            if (!db.Categories.Any() && !db.Items.Any())
            {

                var path = Path.Combine(AppContext.BaseDirectory,
"deepcore.settings.json");
                Setting setting;
                if (File.Exists(path))
                {
                    var json = File.ReadAllText(path);
                    var cfg =
JsonSerializer.Deserialize<SettingConfigDto>(json)!;
                    setting = new Setting
                    {
                        BusinessName = cfg.BusinessName,
                        Email = cfg.Email,
                        Phone = cfg.Phone,
                        Address = cfg.Address,
                        LicenseInfo = cfg.LicenseInfo
                    };
                }
                else
                {

```

```

        setting = new Setting
        {
            BusinessName = "Demo Business System",
            Email = "demo@business.local",
            Phone = "+3800000000000",
            Address = "Demo city, Demo street 1",
            LicenseInfo = "© Demo Business, 2025"
        };
    }

    db.Settings.Add(setting);
}

db.SaveChanges();
}

public class SettingConfigDto
{
    public string BusinessName { get; set; } = "";
    public string Email { get; set; } = "";
    public string Phone { get; set; } = "";
    public string Address { get; set; } = "";
    public string LicenseInfo { get; set; } = "";
}

}

}

```

ДОДАТОК Б
Слайди презентації

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Кафедра програмних засобів

Дослідження та розробка програмного
забезпечення для моделювання, налаштування
та експлуатації бізнес-системи

Виконав студент КНТ-114м
Керівник: доцент

Михайло ШИХАДЕ
Тетяна КОЛПАКОВА

Слайд Б.1 – Титульна сторінка

2

Об'єкт, предмет й мета дослідження

- Об'єкт дослідження – процес інженерії програмного забезпечення для моделювання, налаштування та експлуатації веборієнтованих бізнес-систем.
- Предмет дослідження – методи та програмні засоби автоматизованого моделювання та генерації структури вебсистем на основі пресетів та шаблонів.
- Мета роботи – підвищення економічної ефективності веборієнтованих бізнес-систем за рахунок зменшення часових і фінансових витрат на їх проектування та запуск.

Слайд Б.2 – Об'єкт, предмет й мета дослідження

Постановка завдання

Хід роботи:

- аналіз: встановлення проблематики, порівняння аналогів, підсумкова специфікація вимог;
- моделювання пресету: обґрунтування обраного підходу та формування структури вебсистеми;
- архітектура генератора: проектування згідно із встановленими специфікацією та моделлю;
- експериментальне дослідження: отримання й аналіз результатів, вираження ефективності методу.

Слайд Б.3 – Постановка завдання

Економічні бар'єри

Показник	Україна	ЄС	Тип рішення	Орієнтовна разова вартість розробки, дол. США	Орієнтовна трудомісткість, людино-годин	Типові строки розробки, тижд.
Частка підприємств, що належать до МСП, % від загальної кількості	99,98	99,8	Простий інформаційний вебресурс	500–5 000	50–100	2–4
Частка зайнятих у МСП, % від загальної зайнятості	74	65	Малий бізнес-сайт із БД та базовою адмінкою	5 000–20 000	120–250	4–8
Частка доданої вартості МСП, % від загальної доданої вартості / ВВП	62	53	Кастомна вебсистема з окремими інтерфейсами для персоналу та логікою	25 000–60 000	300–800	8–24

Слайд Б.4 – Економічне питання

Типова модель вебсистеми малого бізнесу

Компонент	Призначення	Основні функції	Типові користувачі
Публічний вебресурс	Представлення бізнесу в інтернеті, робота з клієнтом	Перегляд каталогу товарів/послуг, пошук, оформлення замовлення, зворотний зв'язок	Кінцеві клієнти, відвідувачі
Інтерфейс наповнення БД	Первинне заповнення та періодичне оновлення даних	Додавання/редагування товарів, цін, описів, зображень, довідників	Контент-менеджер, адміністратор
Робочий інтерфейс персоналу	Операційна робота з замовленнями та клієнтами	Перегляд і обробка замовлень, зміна статусів, робота з клієнтами, базові звіти	Менеджери, оператори, адміністратори

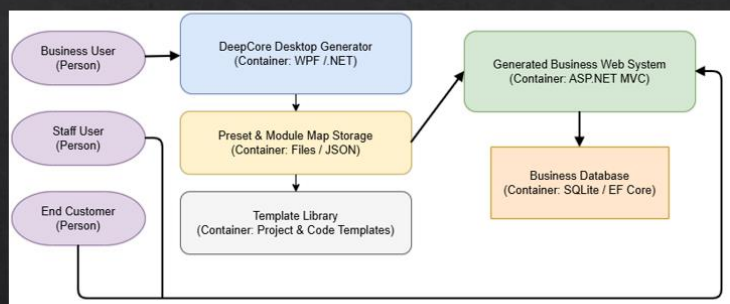
Слайд Б.5 – Опис типової моделі вебсистеми малого бізнесу

Наявні підходи до створення вебсистем

Підхід	Короткий опис	Основні переваги	Обмеження
SaaS конструктори вебресурсів	Хмарний сервіс для швидкого створення публічного вебресурсу на основі шаблонів, з базовими формами, каталогом, оплатою	Дуже низький поріг входу; не потрібні розробники; швидкий запуск (години-дні); включає хостинг та підтримку	Орієнтація переважно на публічний вебресурс; внутрішні реалізуються окремо
CMS та шаблонні платформи електронної комерції	Вебсистема на основі CMS з модулями каталогу, кошика, базової адмінпанелі	Гнучкіша структура, багатий екосистемний набір плагінів; можливість адаптації під предметну область	Кожна вебсистема проєктується окремо; структура БД, ролі та логіка внутрішніх інтерфейсів налаштовуються вручну
Шаблони проєктів і scaffolding у середовищах розробки	Автоматизоване створення каркаса проєкту	Суттєво зменшує обсяг рутинного коду; підвищує стандартизацію технічної структури	Орієнтація на розробника; необхідність повного проєктування предметної області для кожного проєкту
Проектване у роботі ПЗ для генерації веборієнтованих бізнес-систем	Програмне забезпечення, яке за формалізованим пресетом бізнесу генерує структуру трьохкомпонентної	Формалізований опис бізнес-конфігурації; автоматизована генерація кількох компонентів системи; можливість багаторазового використання пресетів	Потребує первинних інвестицій у розробку й підтримку ядра; ефект найбільш відчутний при багаторазовому розгортанні типових бізнес-систем

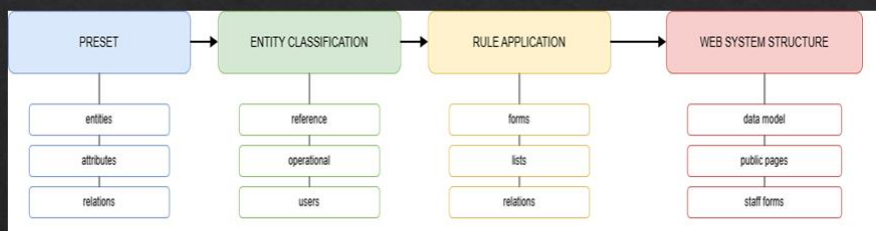
Слайд Б.6 – Порівняльний опис наявних підходів розробки вебсистем

Архітектура комплексу



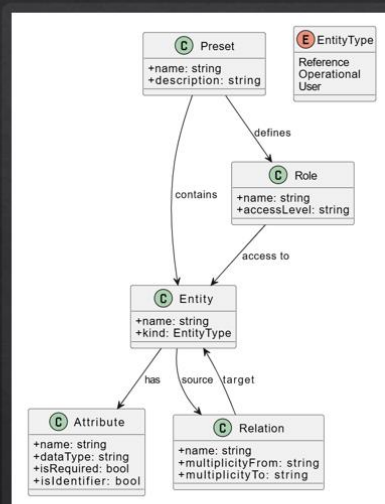
Слайд Б.7 – Архітектура комплексу

Перетворення пресета на структуру вебсистеми



Слайд Б.8 – Опис перетворення пресету на структуру вебсистеми

Модель пресета бізнес-системи

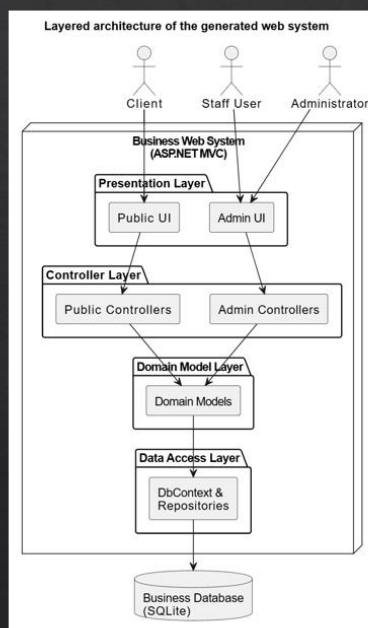


Класифікація сутностей:

- конфігураційна сутність верхнього рівня, що визначає ролі та містить набір сутностей моделі;
- керуюча сутність доступу, яка визначає рівень доступу до конкретних сутностей;
- базова доменна сутність, що містить атрибути та бере участь у зв'язках;
- описова сутність, що визначає структуру даних: тип, обов'язковість, ідентифікацію;
- зв'язувальна сутність, що задає напрям і кратність зв'язків.

Слайд Б.9 – Опис моделі пресету

Архітектура згенерованої вебсистеми



Складові вебсистеми:

а) користувачі:

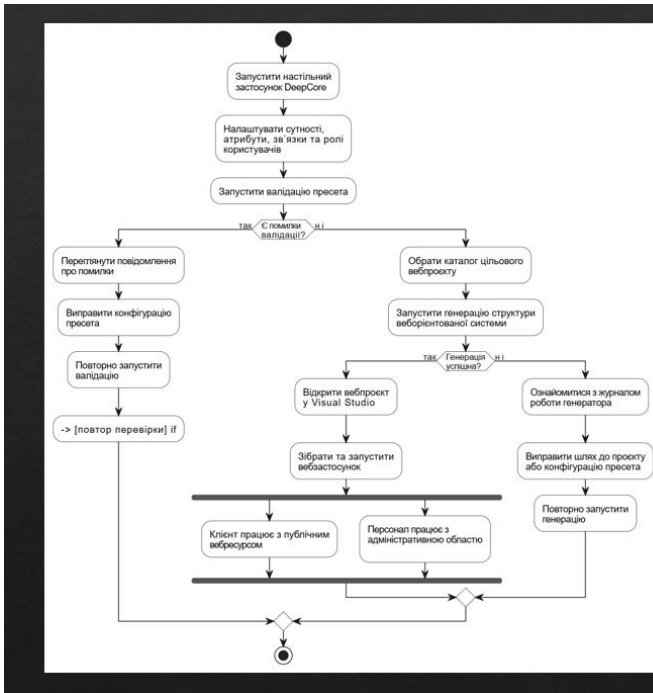
- 1) клієнт;
- 2) працівник;
- 3) адміністратор;

б) рівні системи:

- 1) рівень подання;
- 2) рівень контролерів;
- 3) рівень доменної моделі;
- 4) рівень доступу до даних;

в) сховище даних.

Слайд Б.10 – Архітектура згенерованої вебсистеми



Реалізація: генератор

Слайд Б.11 – Реалізація генератора вебсистем

Елементи інтерфейсу вебсистеми

Оформлення замовлення

Ваші товари

Товар	Ціна	Кількість
Налаштування системи	1500,0	1

Результат: 1500,0

Контактні дані

Ім'я:

Електронна пошта:

Телефон:

Адреса доставки:

Спеціальні вимоги до надання послуг (опціонально):

Каталог

Категорія: Пошук: Ціна від:

Сортування:

2

Консультація

Товари:

Ціна: 50,0

1

Налаштування системи

Товари:

Ціна: 1500,0

Demo Business System
Email: demo@business.local
Телефон: +380000000000

Товари

Додати товар

Назва: Slug: Ціна: К-сть: (без категорії)

ID	Назва	Slug	Ціна	К-сть	Категорія	Зображення
1	Консультація	consultation	50	1	Товари	
2	Налаштування системи	setup	1500	1	Товари	
3	Обслуговування	support	50	1	Товари	

Demo Business System
Email: demo@business.local
Телефон: +380000000000

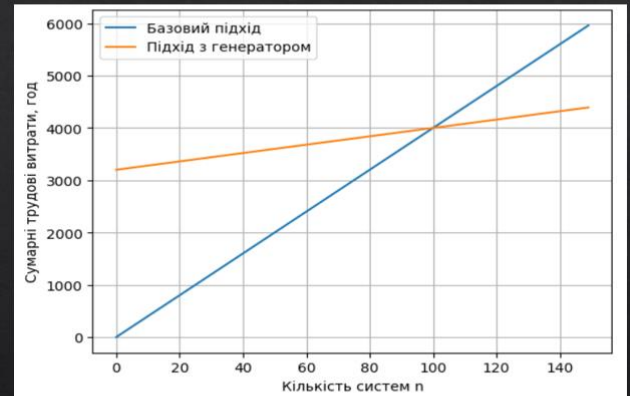
Ціна	Кількість	Сума
1500,0	1	1500,0

Слайд Б.12 – Опис елементів інтерфейсу вебсистеми

Експеримент: методика та результати

$$T_{\text{баз}} = n \cdot T_{\text{проект}}$$

$$T_{\text{ген}} = T_{\text{ядро}} + n \cdot T_{\text{конф}}$$



Слайд Б.13 – Методика проведення експерименту, визначення й аналіз результатів

Висновки

- Проаналізовано наявні підходи до створення веборієнтованих бізнес-систем і виявлено економічні та часові бар'єри їх упровадження для малого бізнесу;
- запропоновано модель пресета та класифікацію сутностей, що дозволяє формалізувати структуру бізнес-системи й автоматизувати її генерацію;
- розроблено архітектуру генератора та багатошарову архітектуру згенерованої вебсистеми на основі ASP.NET MVC;
- експериментальні результати підтвердили зменшення витрат часу й підвищення економічної ефективності проектування та запуску вебсистем.

Слайд Б.14 – Висновки