**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**«Київський політехнічний інститут імені Ігоря Сікорського»**
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

«На правах рукопису»

УДК _____

«До захисту допущено»

Завідувач кафедри

_____ *Сергій Стіренко*
(підпис) (ініціали, прізвище)

“____” _____ 2023 р.

# Магістерська дисертація

зі спеціальності: *121. Інженерія програмного забезпечення*
(код та назва напряму підготовки або спеціальності)

Спеціалізація: *Інженерія програмного забезпечення комп'ютерних систем*

на тему: «Платформи no-code та low-code»

Виконав (-ла): студент (-ка) 2 курсу, групи ІМ-22мп
(шифр групи)

*Смучок Дарина Сергіївна*
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник *доцент каф. ОТ, к.т.н. Волокита А.М.*
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант *проф. каф. ОТ, д.т.н. Кулаков Ю.О.*
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент *доц. каф. ІСТ, к.т.н., доц. Шимкович В. М.*
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2023 року

**Національний технічний університет України**

**«Київський політехнічний інститут імені Ігоря Сікорського»**

Факультет (інститут) *Інформатики та обчислювальної техніки*
(повна назва)

Кафедра *Обчислювальної техніки*
(повна назва)

Рівень вищої освіти – другий (магістерський) за освітньо-професійною програмою
Спеціальність *121. Інженерія програмного забезпечення*
(код і назва)

Освітньо-професійна програма *Інженерія програмного забезпечення комп'ютерних систем*
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
*Сергій Стіренко*
(підпис) (ініціали, прізвище)
« » 2023 р.

**ЗАВДАННЯ**
**на магістерську дисертацію студенту**
Смучок Дарина Сергіївна
(прізвище, ім'я, по батькові)

1. Тема дисертації *Платформи no-code та low-code*

Науковий керівник дисертації *доцент каф. ОТ, к.т.н. Волоктиа А.М.*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «07» листопада 2023 р. No 5168-с

2. Строк подання студентом дисертації

3. Об'єкт дослідження *процес конструювання веб-сайту\застосунку з використанням no-code та low-code платформ*

4. Предмет дослідження *методи та алгоритми для конструювання веб-сайту\застосунку з використанням no-code та low-code платформ*

5. Перелік завдань, які потрібно розробити: - аналіз інсуючих рішень та виявлення їх проблем;

- дослідження no-code та low-code платформ;

- розробка та реалізація методів та алгоритмів для вирішення виявлених проблем;

6. Консультанти розділів дисертації:

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| Нормконтроль | | | |
| | | | |

7. Дата видачі завдання _____

Календарний план

| № з/п | Назва етапів виконання магістерської дисертації | Строк виконання етапів дисертації | Примітка |
|---|---|---|---|
| 1 | *Затвердження теми роботи* | *01.09.2023* | |
| 2 | *Вивчення та аналіз завдання* | *05.09.2023* | |
| 3 | *Розробка архітектури та загальної структури програми* | *27.09.2023* | |
| 4 | *Розробка структур окремих інтерфейсів програми* | *17.10.2023* | |
| 5 | *Програмна реалізація* | *29.11.2023* | |
| 6 | *Оформлення пояснювальної записки* | *25.12.2023* | |
| 7 | *Захист* | | |

Студент _____ Смучок Д.С.
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ Волокита А.М.
(підпис) (ініціали, прізвище)

**РЕФЕРАТ**

на магістерську дисертацію

виконану на тему: Платформи no-code та low-code

студентом: Смучок Дариною Сергіївною

Робота складається із вступу та чотирьох розділів. Загальний обсяг роботи: 113 аркуші тексту, 52 ілюстрації, 48 таблиць та 2 формули. При підготовці використовувалася література з 16 різних джерел.

**Актуальність.** Розробка без коду (або з низьким рівнем коду) — це тип веб-розробки, який дозволяє непрограмістам і програмістам створювати програмне забезпечення за допомогою взаємодії з графічним інтерфейсом користувача замість написання коду. No-code і low-code базуються на фундаментальному переконанні, що технологія має дозволяти та полегшувати створення, а не бути перешкодою для входу. Вони усунули складність, пов'язану з необхідністю знання мов програмування, і дали кожному можливість створювати прості веб-застосунки та веб-сайти незалежно від рівня знань.

**Мета і завдання дослідження.** Метою магістерської роботи є зменшення вхідного порогу для створення веб-сайтів та застосунків за допомогою можливостей no-code та low-code платформ.

Для досягнення мети дослідження поставлено і вирішено такі завдання:

- аналіз існуючих рішень та виявлення їх проблем;
- дослідження no-code та low-code платформ;
- розробка та реалізація методів та алгоритмів для вирішення виявлених проблем;
- розробка програмного забезпечення для оцінки можливостей користувачів, що бажають отримати новий для себе досівд розробки ПЗ не заглиблюючись в технічні нюанси розробки;

**Об'єкт дослідження** – процес конструювання веб-сайту\застосунку з використанням no-code та low-code платформ

**Предмет дослідження** – методи та алгоритми для конструювання веб-сайту\застосунку з використанням no-code та low-code платформ

**Методи досліджень.** Для досягнення поставлених в магістерській роботі задач, використано метод імітаційного моделювання.

Наукова новизна одержаних результатів роботи полягає полягає не стільки в нових фундаментальних наукових проривах, скільки в новому підході до дизайну, розробки та розгортання програмних продуктів. Ці платформи втілюють концепцію "абстракції від деталей", дозволяючи розробникам та не-розробникам зосереджуватися на вирішенні конкретних задач, а не на технічних деталях реалізації.

Проведене дослідження дає можливість порівняти традиційний спосіб написання коду з використанням no-code та low-code платформ, оцінити доцільність кожного з них, та створити програмний продукт, що буде мати низький поріг входу для користувача мало знайомого з програмуванням.

**Особистий внесок здобувача.** Магістерське дослідження є самостійно виконаною роботою, в якій відображено особистий авторський підхід та особисто отримані теоретичні та прикладні результати, що відносяться до використання no-code та low-code платформ для виконання задач з розробки.

**Практична цінність.** Отримані результати можуть використовуватися у майбутніх дослідженнях за напрямками:

- конструювання веб-сайтів та застосунків за допомогою no-code та low-code платформ
- оптимізація процесів розробки ПЗ
- створення інтерфейсів

**Ключові слова**

No-code, Low-code, платформа

**ABSTRACT**

on the master's thesis

done on the topic: No-code and low-code platforms

by a student Smuchok Daryna

The work consists of an introduction and four sections. Total volume of work: 113 pages of text, 52 illustrations, 48 tables and 2 formulas. Literature from 16 different sources was used in the preparation.

**Relevance.** No-code (or low-code) development is a type of web development that allows non-programmers and programmers to create software by interacting with a graphical user interface instead of writing code. No-code and low-code are based on the fundamental belief that technology should enable and facilitate creation, not be a barrier to entry. They eliminated the complexity associated with the need to know programming languages and gave everyone the opportunity to create simple web applications and websites regardless of their level of knowledge.

**The purpose and objectives of the study.** The purpose of the master's thesis is to reduce the entry threshold for creating websites and applications using the capabilities of no-code and low-code platforms.

To achieve the research goal, the following tasks were set and solved:

- analysis of existing solutions and identification of their problems;

- research of no-code and low-code platforms;

- development and implementation of methods and algorithms to solve the identified problems;

- software development for users who want to get a new software development experience without delving into the technical nuances of development;

**Object of research** - the process of designing a website/application using no-code and low-code platforms

**Subject of research** - methods and algorithms for designing a website/application using no-code and low-code platforms

**Research methods**. To achieve the tasks set in the master's thesis, the method of simulation modeling was used.

**The scientific novelty** of the obtained results is not so much in new fundamental scientific breakthroughs as in a new approach to the design, development and deployment of software products. These platforms embody the concept of "abstraction from details," allowing developers and non-developers to focus on solving specific problems rather than on the technical details of implementation.

The study makes it possible to compare the traditional way of writing code using no-code and low-code platforms, assess the feasibility of each of them, and create a software product that will have a low entry threshold for a user unfamiliar with programming.

**Personal contribution of the applicant**. The master's thesis is an independently performed work that reflects the author's personal approach and personally obtained theoretical and applied results related to the use of no-code and low-code platforms for development tasks.

**Practical value**. The results obtained can be used in future research in the following areas:

- designing websites and applications using no-code and low-code platforms
- optimization of software development processes
- creation of interfaces

**Keywords.** No-code, Low-code, platform

**ЗМІСТ**

## ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

**No-code** - Платформи, які дозволяють створювати додатки за допомогою візуальних інтерфейсів без написання ручного коду, орієнтовані на нетехнічних користувачів.

**Low code** - Інструменти розробки, які мінімізують ручне написання коду за допомогою візуальних інструментів, але дозволяють створювати кастомний код для розширених потреб.

**UI** - Відноситься до макету та інтерактивних елементів програми або веб-сайту, орієнтованих на користувача.

**MVP** - Продукт з мінімальними функціями, необхідними для запуску для збору відгуків та ітерацій.

**Drag-and-drop інтерфейс** – такий різновид інтерфейсу, що дозволяє розміщувати елементи за допомогою перетягування.

# ВСТУП

Зростаюча потреба в розробці програмного забезпечення та швидкість технологічного розвитку призвели до зростання попиту на рішення, які дозволяють створювати додатки швидше і простіше, без необхідності залучення великих команд розробників. Платформи no-code та low-code є відповіддю на цей виклик. Платформи no-code надають можливість створювати додатки без необхідності кодування. Вони мають інтуїтивно зрозумілий візуальний інтерфейс і дозволяють користувачам створювати функціональні додатки за допомогою блоків, перетягування та кастомізації. Це робить процес розробки доступним для людей без навичок програмування. Якщо колись розробка додатків і запуск веб-додатків були можливими лише для кваліфікованих програмістів, то платформи розробки без використання коду разом із безліччю навчальних посібників можуть підштовхнути будь-кого до реалізації своїх ідей.

З іншого боку, low-code платформи дозволяють розробникам прискорити процес створення додатків, надаючи заздалегідь побудовані блоки коду і компоненти, які можна використовувати для швидкого створення додатків без необхідності писати код з нуля. Це економить час і зусилля, а також забезпечує швидшу розробку і гнучкість при внесенні змін до програми.

Важливість платформ без коду та з низьким рівнем коду зумовлена кількома факторами. По-перше, вони дозволяють компаніям знизити витрати на розробку програмного забезпечення, оскільки їм не потрібно витрачати час і ресурси на пошук і наймання великих команд програмістів. По-друге, ці платформи скорочують час виходу на ринок, що дає компаніям конкурентну перевагу. По-третє, no-code та low-code платформи роблять процес розробки більш доступним для широкого кола користувачів без технічних навичок та сприяють залученню нових та інноваційних ідей.

Однією з актуальних сфер застосування no-code та low-code платформ є розробка інтерфейсів користувача (UI). Ці платформи надають можливість створювати привабливі та функціональні веб-сайти, мобільні додатки та інші інтерактивні інтерфейси без необхідності писати код. Це особливо корисно для

дизайнерів та маркетологів, які можуть самостійно розробляти і змінювати інтерфейси без залучення розробників.

Платформи веб-розробки без коду пройшли довгий шлях від редакторів WYSIWYG минулого. Хоча ці веб-сайти створювали прийнятний дизайн для того часу, ці веб-сайти були простими, пропонуючи односторонній досвід для користувача. Потім з'явилися більш динамічні конструктори веб-сайтів, які могли зробити оригінальні платформи без коду, а саме створення веб-сайтів, повних взаємодії, динамічної анімації та інших складних візуальних елементів.

Сьогодні існує низка no-code та low-code платформ, які вже успішно працюють у різних сферах, таких як розробка веб-сайтів, мобільних додатків та автоматизація бізнес-процесів. За допомогою цих платформ навіть люди без досвіду програмування можуть створювати потужні додатки та розробляти проекти.

Існує багато варіантів використання no-code платформ. Це не обмежується лише створенням веб-сайтів. Їх можна використовувати для створення мобільних додатків, веб-додатків, голосових додатків, внутрішніх інструментів, інтеграції та автоматизації завдань. Не знаючи, як написати щось складніше ніж Hello World, можна створювати чат-боти за допомогою Voiceflow, підключати кілька програм і створювати автоматизовані робочі процеси за допомогою Zapier, а також використовувати Shopify для роботи магазинів електронної комерції. Платформи low-code, такі як Airtable, в свою чергу дають змогу створювати персоналізовані інструменти для покращення бізнес-процесів команд. Обсяг того, що може робити без коду, або з зовсім невеликою його кількістю зростає дедалі більше.

Крім того, no-code та low-code платформи знаходять широке застосування в автоматизації бізнес-процесів. Вони дозволяють створювати потужні робочі потоки, додавати автоматичні оповіщення, обробляти дані та виконувати інші операції без необхідності писати складний код. Це дозволяє підприємствам покращити ефективність роботи, знизити кількість ручних операцій та зосередитися на стратегічних завданнях.

Однією з переваг no-code та low-code платформ є можливість швидко прототипувати та тестувати ідеї. Завдяки візуальним інструментам та шаблонам, розробники можуть швидко створювати прототипи програм та перевіряти їх працездатність перед повноцінною розробкою. Це дозволяє зменшити час і витрати на розробку продукту, а також залучити користувачів для отримання реального фідбеку.

Загалом, no-code та low-code платформи є актуальними і потужними інструментами, які допомагають прискорити процес розробки програмного забезпечення, знизити витрати та залучити більше людей до процесу розробки. Вони відкривають нові можливості для швидкого створення інноваційних продуктів і сприяють розвитку цифрової економіки.

Таким чином, no-code та low-code платформи відіграють важливу роль у сучасному світі програмування та сприяють тому, щоб зробити процес розробки програмного забезпечення доступнішим, швидшим та ефективнішим. Вони допомагають компаніям виконувати роботи з розробки з меншими витратами і в коротші терміни, тим самим підвищуючи їхню конкурентоспроможність на ринку.

Метою дослідження є вивчення впливу no-code та low-code платформ на розробку програмного забезпечення для користувачів, що бажають отримати новий для себе досівд розробки ПЗ не заглиблюючись в технічні нюанси розробки.

Окрім цього, дослідження охоплює аналіз сучасних тенденцій в проектуванні сайтів та застосунків без написання коду.

# РОЗДІЛ 1
# ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

## 1.1. Проблематика

Кожен, хто коли-небудь керував проектом, незалежно від дисципліни, знає, що чим більше людей вам потрібно працювати, тим складнішою стає логістика вчасної та бюджетної доставки. Традиційний процес розробки застосунку чи веб-сайту залежить щонайменше від:

— Контенту

— Дизайну

— Розробки програмного коду

А для більш комплексних проектів та кращих результатів також можуть знадобитися інженери. Крім того, кожна з трьох дисциплін, наведених вище, цілком може включати кілька спеціалістів, тому наведений вище список цілком може виглядати приблизно так:

— Контент (стратег, копірайтер, SEO, інформаційний архітектор)

— Дизайн (UX дизайнер, UI дизайнер, дизайнер взаємодії, Motion-дизайнер)

— Розробка (front-end dev, back-end dev, content dev тощо)

В свою чергу, основною аудиторією no-code і low-code платформ найчастіше є люди, які ще лише починають свій бізнес або стартап, і мають на меті лише зробити найпростіші базові лендінг сторінки свого продукту, галерею робіт, архів і тд, маючи основний контакт з клієнтами через Instagram, OLX та інші аналоги, або співпрацюючи наживу.

На початку чогось такого простого, як серія добре розроблених зображень (наприклад, прототип InVision), може бути достатньо, щоб донести основну ідею до потенційних інвесторів, перших користувачів і майбутніх членів команди. У міру розвитку концепції буде потрібна дедалі більша точність — але навіть тоді такі інструменти, як Webflow, Bubble, Glide і Voiceflow, можуть забезпечувати неймовірно надійний досвід, якого може бути більш ніж достатньо, щоб

викликати інтерес і перевірити гіпотези. І як тільки ви будете готові почати просувати свої ідеї громадськості, Webflow, Bubble і Carrd створять красиві, високоефективні інструменти для цільових сторінок, які вам знадобляться, щоб повідомити про свою основну цінність та інші переваги.

Якщо ж все так добре, і no-code інструменти повністю покривають весь необхідний функціонал, не потребують додаткових інвестицій і працюють без знань чи експертизи в певних областях, то чому досі є попит на розробників, дизайнерів та інших спеціалістів? Цьому також є цілком логічне пояснення:

1. Роль в проекті: для створення адекватного продуту, платформи low-code та no-code вимагають від вас хоча б базових навичок та знань перелічених у пунктах вище – сам собі і дизайнер, і програміст, і контент менеджер.

2. Вивчення платформ все ще вимагає часу та зусиль: no-code та low-code підвищують швидкість та продуктивність, але це не легко. Ці платформи є тривіальними, і розвиток певного рівня знань потребує часу. [1].

3. Ресурси та підтримка спільноти обмежені: багато платформ є відносно незрілими. Існують мільйони курсів, книг та інших навчальних матеріалів для вивчення традиційних мов, таких як Java або C#. Цілком інший сценарій ми бачимо для платформ low-code та no-code – особливо для більш нових платформ.

4. Ціноутворення може збивати з пантелику: корпоративні платформи low-code та no-code, як правило, невиправдано дорогі. Платформи середнього та малого ринку менш витратні, але, як правило, менш масштабовані. Залучення кількох платформ для комплексного вирішення ще більше ускладнює питання ціноутворення, тому це може бути гарним рішенням на перший час, але втратить свою актуальність в процесі розвитку.

Загалом, слід зазначити, що no-code та low-code платформи сприяють розширенню кола людей, які можуть брати участь у процесі розробки програм. Вони дозволяють непрограмістам створювати власні програми, що сприяє демократизації програмування та створює нові можливості для творчості та інновацій.

## 1.2. Low-code платформи

Low-code платформи - це фреймворки для розробки програмного забезпечення, призначені для полегшення швидкої розробки додатків з мінімальним ручним кодуванням. Вони використовують візуальні середовища розробки, перетягування компонентів і готові шаблони для спрощення створення та розгортання додатків.[1] Ось за рахунок чого low-code платформи мають значну аудиторію:

- Швидкість розробки додатків: Платформи з низьким рівнем коду дозволяють швидше розробляти та розгортати додатки порівняно з традиційними методами кодування. Це дозволяє компаніям швидко реагувати на ринкові зміни, потреби користувачів або внутрішні потреби.
- Зменшення відставання: Багато IT-відділів стикаються з великою кількістю запитів на розробку додатків. Платформи з низьким рівнем коду можуть допомогти зменшити це відставання, прискоривши процес розробки.
- Розширення можливостей громадських розробників: Бізнес-професіонали з обмеженими технічними знаннями (яких часто називають "громадськими розробниками") можуть використовувати ці платформи для створення або модифікації додатків, звільняючи IT-відділ для більш складних завдань.
- Економія ресурсів: Швидший час розробки може призвести до зменшення витрат на оплату праці. Крім того, витрати на навчання можуть бути нижчими, оскільки немає необхідності розбиратися в складному коді. Традиційна розробка додатку може бути дорогою, особливо для стартапів або малого бізнесу. Платформи без коду часто пропонують більш бюджетну альтернативу, особливо для простих додатків.

- Обслуговування: Оскільки постачальник платформи керує більшою частиною інфраструктури та оновленнями, витрати на обслуговування можуть бути значно зменшені.
- Перевірка ідеї: Компанії та стартапи можуть швидко створювати прототипи або мінімально життєздатні продукти (MVP) для перевірки ідей, збору відгуків користувачів або забезпечення фінансування перед початком повномасштабної розробки.
- Адаптивність: Платформи з низьким рівнем коду часто мають вбудовані функції масштабування, що полегшує додаткам роботу зі зростаючою кількістю користувачів.
- Інтеграція: Багато low-code платформ пропонують вбудовані інтеграції з популярними сторонніми сервісами та API, що спрощує процес підключення різних інструментів і систем.
- Уніфікованість: Оскільки багато додатків, побудованих на платформі з низьким рівнем коду, використовують стандартизовані компоненти, існує узгодженість у дизайні, користувацькому досвіді та функціональності.
- Управління та відповідність вимогам: Централізована розробка на платформі з низьким рівнем коду може полегшити дотримання бізнес-правил, стандартів і заходів відповідності в різних додатках.
- Модернізація застарілих систем та адаптація до цифрової епохи:: Компанії із застарілими системами можуть використовувати low-code платформи для модернізації своєї цифрової інфраструктури без повної перебудови. В свою чергу, для компаній, які відстають у впровадженні цифрових технологій, платформи без коду є відправною точкою для розробки цифрових інструментів і платформ без значних попередніх інвестицій у ресурси для розробки.
- Автоматизація процесів: Компанії можуть використовувати платформи з низьким рівнем коду для автоматизації повторюваних завдань, робочих процесів або процесів, підвищуючи ефективність і

точність і тим самим вивільняти ресурси розробників на імплементацію нових рішень.

- Різноманітні набори навичок: Команди з різним рівнем технічної експертизи можуть співпрацювати над розробкою додатків. Візуальна природа low-code платформ може подолати розрив між технічними та нетехнічними зацікавленими сторонами.
- Безперервний цикл зворотного зв'язку: Завдяки швидшим циклам розробки компанії можуть часто повторювати ітерації на основі відгуків користувачів, гарантуючи, що кінцевий продукт тісно пов'язаний з потребами та очікуваннями людей, що будуть використовувати даний продукт[3]

По суті, low-code платформи - це інструменти, призначені для демократизації розробки додатків, підвищення її гнучкості та задоволення мінливих потреб бізнесу в цифровому світі. Вони особливо цінні для організацій, які прагнуть впроваджувати інновації швидко, ефективно та гнучко в умовах мінливої динаміки ринку.[2]

### 1.3. No-code платформи

Платформи no-code - це фреймворки для розробки, які дозволяють приватним особам і компаніям створювати програмні додатки за допомогою візуальних інтерфейсів і попередньо визначених компонентів без написання коду вручну. Ось основні цілі та причини використання платформ без коду:

- Демократизація розвитку: Платформи без коду дозволяють людям без технічної підготовки (наприклад, навичок програмування) розробляти функціональні додатки, втілюючи цифрові ідеї в життя і вирівнюючи умови гри.
- Швидка реалізація: Стартапи, підприємці та компанії можуть швидко створювати прототипи або мінімально життєздатні продукти (MVP) для тестування ідей на ринку, отримання перших відгуків або презентації інвесторам.

- Цифрова присутність: Платформи без коду, особливо конструктори веб-сайтів, дозволяють компаніям і приватним особам легко встановлювати свою присутність в Інтернеті, будь то через веб-сайти, блоги або навіть мобільні додатки.
- Навчання та експерименти: Освітяни можуть використовувати платформи без коду, щоб познайомити учнів з концепціями, пов'язаними з розробкою цифрових продуктів, логікою та дизайном, без бар'єру, що полягає в необхідності спочатку навчитися кодувати.
- Швидке задовольняння бізнес-потреби: В організаціях, де IT-відділи завалені запитами, платформи без коду дозволяють іншим відділам розробляти рішення своїх проблем, не завжди покладаючись на IT, таким чином зменшуючи відставання.
- Адаптивні рішення: Багато платформ без коду надають функції масштабування, гарантуючи, що в міру зростання бізнесу або бази користувачів додаток може адаптуватися, не вимагаючи серйозної переробки.
- Персоналізовані рішення: Окремі особи або громадські групи можуть використовувати інструменти без коду для створення персоналізованих додатків або платформ, пристосованих до їхніх конкретних потреб або інтересів, не витрачаючи коштів на індивідуальну розробку.[2]

Таким чином, платформи без коду допомагають зробити розробку і впровадження цифрових рішень більш доступними для широкої аудиторії. Вони задовольняють потреби бізнес-середовища, що швидко змінюється, дозволяють швидко експериментувати та зменшують бар'єри для цифрових інновацій.

Low-code та no-code платформи спрощують процес розробки програмного забезпечення, абстрагуючись від складнощів ручного кодування. Давайте заглибимося в те, як працюють ці платформи, як на зовнішній стороні (інтерфейс користувача), так і на внутрішній стороні (базова реалізація).

Фронтальна сторона (користувацький інтерфейс):

- Візуальне середовище розробки: Платформи з низьким рівнем коду та без коду часто постачаються з інтерфейсом drag-and-drop. Користувачі можуть візуально створювати свій додаток, розміщуючи такі елементи, як кнопки, форми та зображення на полотні.
- Бібліотеки компонентів: Користувачі мають доступ до бібліотеки готових компонентів, які інкапсулюють загальні функціональні можливості. Це можуть бути елементи інтерфейсу, з'єднувачі даних, робочі процеси тощо.
- Властивості, що налаштовуються: Після додавання компонентів на полотно дизайну користувачі можуть налаштувати їхні властивості. Наприклад, колір кнопки, підпис і пов'язану з нею дію можна змінити за допомогою простих налаштувань.
- Візуальна побудова логіки: Багато платформ дозволяють користувачам визначати логіку програми візуально. Це можуть бути блок-схеми, тригери на основі умов або інші візуальні інструменти, які представляють базову логіку без необхідності традиційного кодування.
- Моделювання даних: Користувачі часто можуть визначати бази даних та керувати ними візуально, створюючи таблиці, встановлюючи зв'язки та керуючи правилами доступу до даних без заглиблення в SQL або інші мови баз даних.
- Попередній перегляд і тестування: Більшість платформ пропонують можливість попереднього перегляду додатку в режимі реального часу, що дозволяє користувачам тестувати користувацький досвід і функціональність перед розгортанням.

Зворотний бік (базова реалізація):

- Генерація коду: Коли користувач візуально розробляє додаток, платформа внутрішньо генерує код для представлення цього

дизайну. Цей код зазвичай оптимізований і базується на стандартних мовах програмування, таких як JavaScript, HTML, CSS тощо.[2]

- Абстракція платформи: Ці платформи абстрагуються від складнощів різних завдань, таких як управління базами даних, налаштування сервера або інтеграція API. Наприклад, коли користувач візуально визначає таблицю бази даних, платформа може за лаштунками перетворювати її на SQL-операції.
- Хостинг і розгортання: Багато з цих платформ постачаються з вбудованими хостинговими рішеннями. Коли користувач "розгортає" свій додаток, платформа піклується про забезпечення сервера, масштабування та інші пов'язані з цим завдання.
- Інтеграції з API: Платформи без коду та з низьким рівнем коду часто постачаються з попередньо вбудованими коннекторами до популярних сторонніх сервісів. Коли користувач інтегрує один з цих сервісів, платформа обробляє основні виклики API та обмін даними.
- Оптимізація та обслуговування: Для забезпечення продуктивності та безпеки ці платформи часто оптимізують згенерований код і регулярно оновлюють базову інфраструктуру. Вони прозоро керують виправленнями ОС, оновленнями безпеки та іншими технічними завданнями.
- Розширення та кастомний код: Особливо в платформах з низьким рівнем коду, якщо попередньо створені компоненти не відповідають потребам користувача, часто є можливість додати власний код. Платформа гарантує, що цей кастомний код легко інтегрується з візуально створеними компонентами.
- Масштабованість: За лаштунками ці платформи гарантують, що додатки, побудовані на них, можуть масштабуватися. Це може включати в себе балансування навантаження, шардінг баз даних та інші передові технології.

Платформи з низьким рівнем коду та без коду мають на меті спростити та пришвидшити процес розробки програмного забезпечення, зменшивши кількість необхідного ручного кодування. Однак вони орієнтовані на різні аудиторії та пропонують різний рівень кастомізації та складності. Основні відмінності між ними наведено в таблиці.[4]

*Таблиця 1.1*

**Відмінності між no-code та low-code платформами**

| | *Платформи з низьким рівнем коду* | *Платформи без коду* |
|---|---|---|
| Аудиторія | Орієнтовані на професійних розробників, які прагнуть пришвидшити процес розробки, або бізнес-професіоналів з певними технічними знаннями | Орієнтовані насамперед на бізнес-професіоналів, експертів у певній галузі або осіб без технічної освіти, які хочуть створювати програмні рішення без написання коду. |
| Кастомізація та складність | Ці платформи забезпечують основу, але вони дозволяють (і часто очікують) від розробників написання кастомного коду для більш специфічних або складних функцій. | Ці платформи мають на меті охопити широкий спектр типових випадків використання за допомогою візуальних інструментів та готових компонентів. Як наслідок, вони можуть бути не настільки гнучкими для дуже специфічних або нових вимог. |

*Таблиця 1.1 (продовження)*

**Відмінності між no-code та low-code платформами**

| | | |
|---|---|---|
| Вимоги до написання коду | Хоча більшу частину функціональності можна досягти за допомогою візуальних інструментів, все ж очікується, що програмувати окремі частини буде необхідно, особливо для складних проектів. | Як випливає з назви, платформи без коду призначені для створення функціональних додатків без будь-якого ручного написання коду. Все робиться за допомогою візуальних інтерфейсів. |
| Гнучкість | Вони пропонують більшу гнучкість, ніж платформи без коду, з точки зору інтеграції з іншими системами, реалізації кастомних функцій та оптимізації продуктивності. | Незважаючи на те, що вони стають все потужнішими,, платформи без коду можуть мати обмеження в складних налаштуваннях, розширених інтеграціях або певних оптимізаціях продуктивності. |
| Варіанти використання | Ідеально підходять для проектів, які потребують балансу між кастомізацією та швидкістю розробки. Їх можна використовувати для широкого спектру додатків, від мобільних додатків до корпоративного програмного забезпечення. | Ідеально підходять для швидкого створення MVP (мінімально життєздатних продуктів), прототипів, простих веб-додатків, мобільних додатків або інструментів автоматизації бізнесу. Гірше підходять для вузькоспеціалізованих або критично важливих до продуктивності додатків. |

Платформи без коду наголошують на простоті та доступності, дозволяючи тим, хто не має досвіду програмування, втілювати цифрові проекти в життя. Вони можуть мати обмеження в плані дуже кастомізованих функціональних можливостей. По суті, вибір між low-code і no-code значною мірою залежить від вимог проекту, цільової аудиторії та наявного набору навичок команди розробників або окремої людини.

## 1.4. Огляд ринку

Розглянемо найбільші існуючі low-code та no-code платформи, а також основні переваги та недоліки.

### 1.4.1 Low-Code платформи

**OutSystems**

OutSystems - це універсальна платформа для швидкого створення додатків корпоративного рівня. Вона підтримує широкий спектр типів додатків, від мобільних, веб-додатків до десктопних, та є одним з провідних рішень на ринку.[5]

Плюси:

- Швидка розробка та розгортання додатків.
- Підтримує мобільні, веб- та десктопні додатки.
- Широкі можливості інтеграції з іншими платформами та системами.

Мінуси:

- Ліцензування може бути дорогим, особливо для підприємств.
- Висококласні функції можуть вимагати більш складного навчання для початківців.

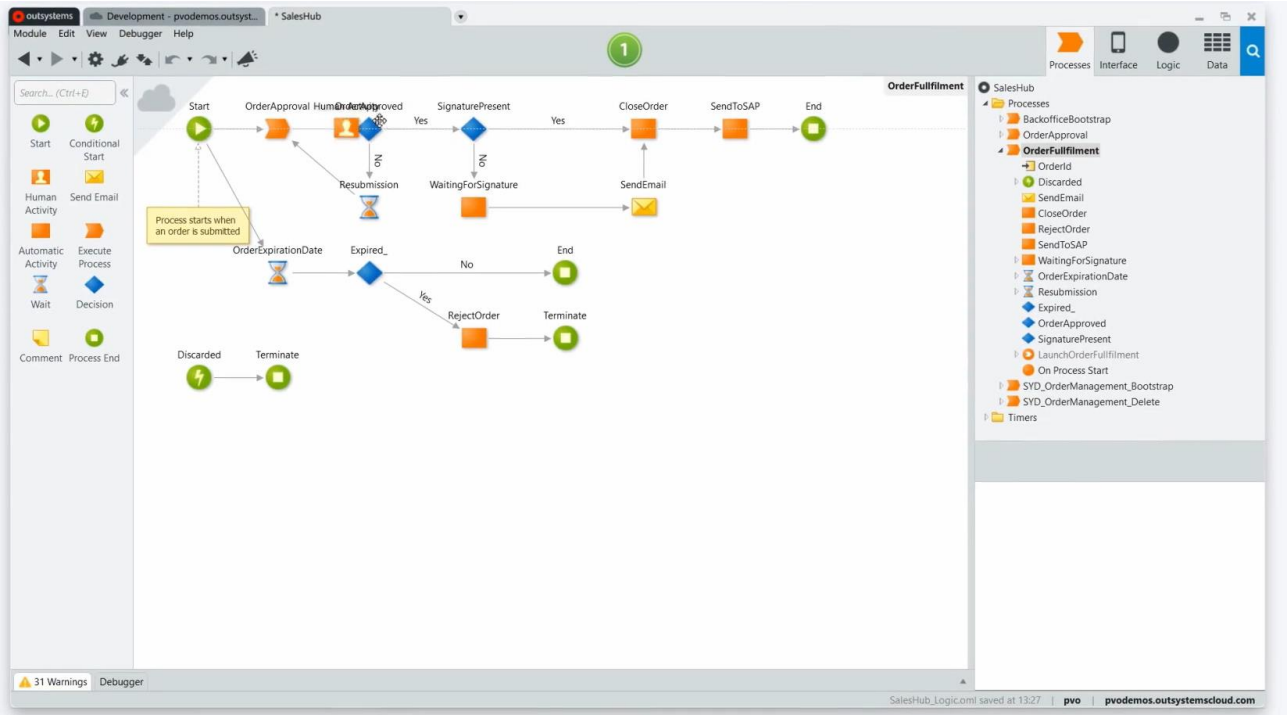


Рис. 1.1. Інтерфейс OutSystems

**Appian**

В першу чергу створений для управління бізнес-процесами, Appian розширився до простору низькорівневого коду, дозволяючи підприємствам розробляти потужні додатки з мінімальним ручним програмуванням. [6]

Плюси:

- Поєднує низькокодову розробку з управлінням бізнес-процесами.
- Велика увага приділяється безпеці та управлінню.

Мінуси:

- Інтерфейс менш інтуїтивно зрозумілий, ніж в аналогів
- Незважаючи на низький рівень коду, для деяких налаштувань може знадобитися традиційне програмування

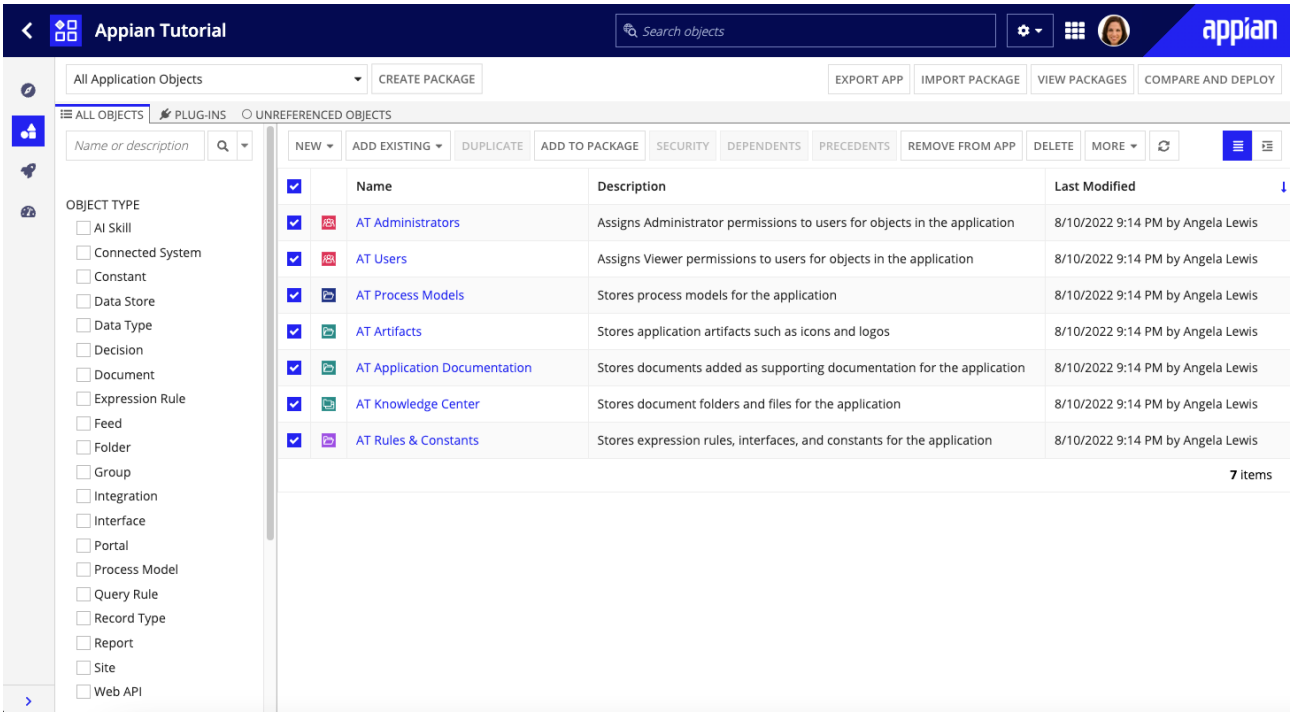


Рис. 1.2. Інтерфейс Appian

**Mendix**

Компанія Mendix, що належить Siemens, орієнтована як на професійних розробників, так і на бізнес-користувачів. Він підтримує наскрізну розробку та розгортання додатків.[7]

Плюси:

- Підтримує наскрізну розробку додатків.
- Сприяє співпраці між розробниками та бізнес-користувачами.

Мінуси:

- Програми можуть бути менш продуктивними, ніж ті, що написані власними силами.
- Плани вищого рівня можуть бути дорогими

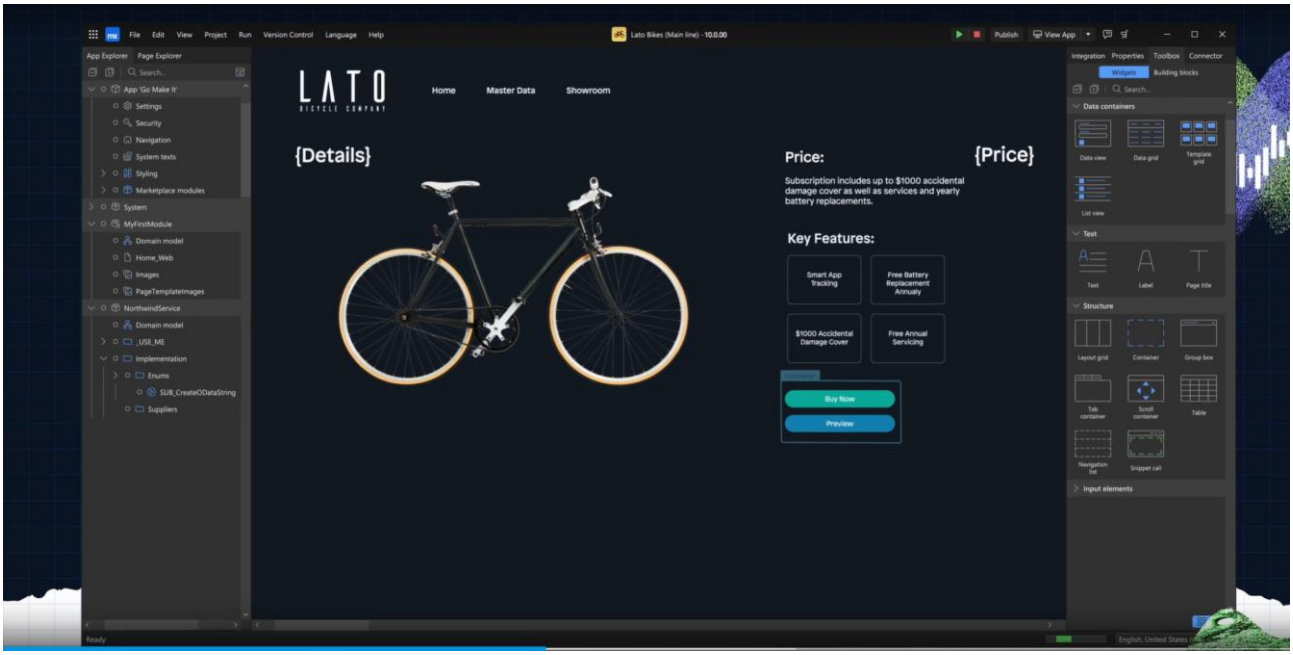


Рис. 1.3. Інтерфейс Mendix

**Salesforce Lightning**

Розширення Salesforce, ця платформа дозволяє користувачам швидко створювати додатки за допомогою перетягування компонентів. Вона інтегрована в екосистему Salesforce.

Плюси:

- Безшовна інтеграція з екосистемою Salesforce
- Компонентний підхід прискорює процес розробки

Мінуси:

- Значною мірою прив'язаний до екосистеми Salesforce
- Робота з ним вимагає знайомства з Salesforce[8]

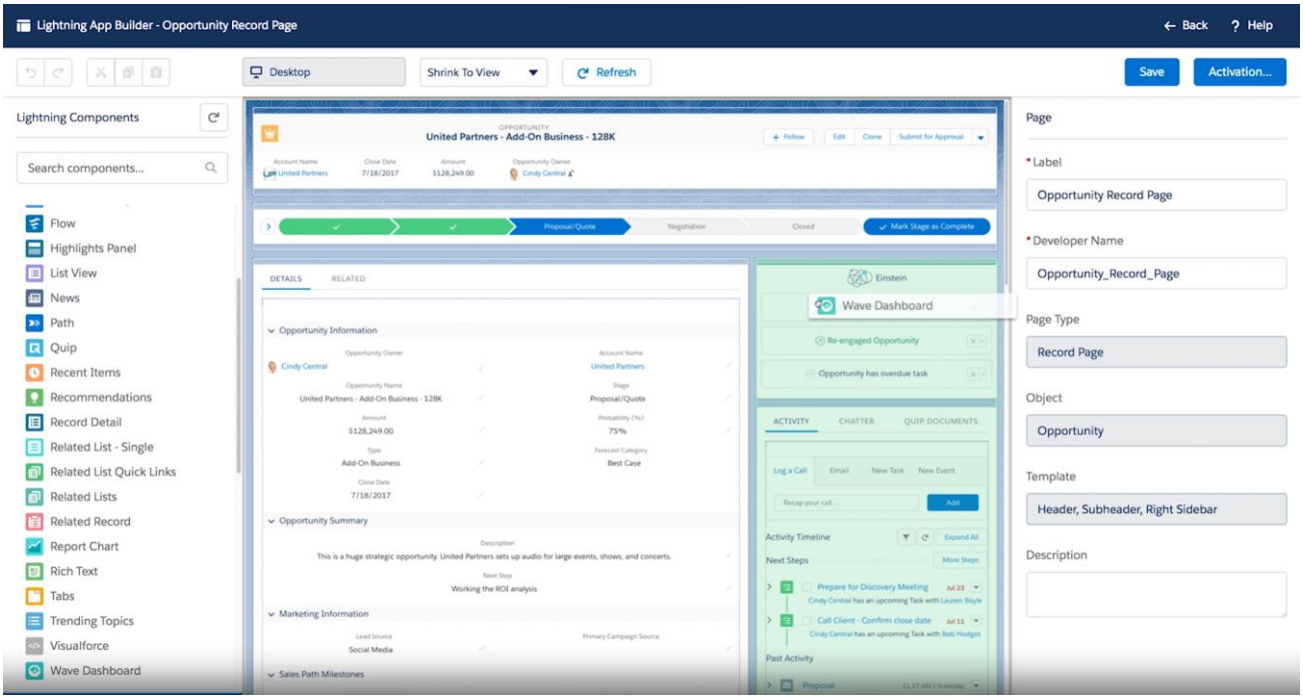


Рис. 1.4. Інтерфейс Salesforce Lightning

**Pega**

Pega - це інструмент управління бізнес-процесами, який також пропонує можливості низькокодової розробки, зокрема для взаємодії з клієнтами та автоматизації цифрових процесів.

Плюси:

- Сильна в автоматизації цифрових процесів
- Створена і орієнтована на великі підприємства

Мінуси:

- Дизайн інтерфейсу користувача може бути менш інтуїтивно зрозумілим в порівнянні з конкурентами
- Ціна може бути вищою для малих та середніх підприємств[9]

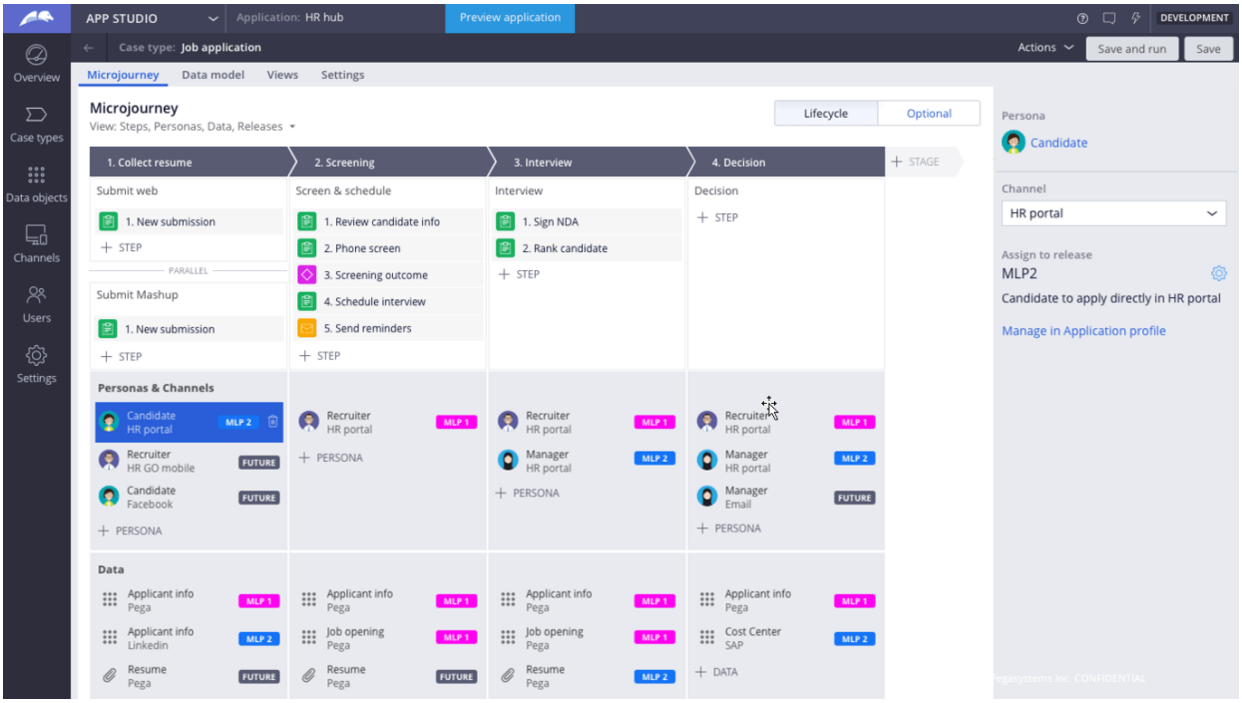


Рис. 1.5. Інтерфейс Pega

### 1.4.2. No-code платформи

**Wix**

Спочатку відомий як конструктор веб-сайтів, Wix розширив свої можливості, включивши в них більш комплексну розробку веб-додатків без необхідності програмування.[10]

Плюси:

- Зручний для користувача, інтуїтивно зрозумілий інтерфейс перетягування.
- Широкий вибір шаблонів дизайну.

Мінуси:

- Обмежена гнучкість порівняно з більш просунутими платформами для веб-розробки.
- Веб-сайти можуть бути менш оптимізованими, ніж ті, що створені вручну.

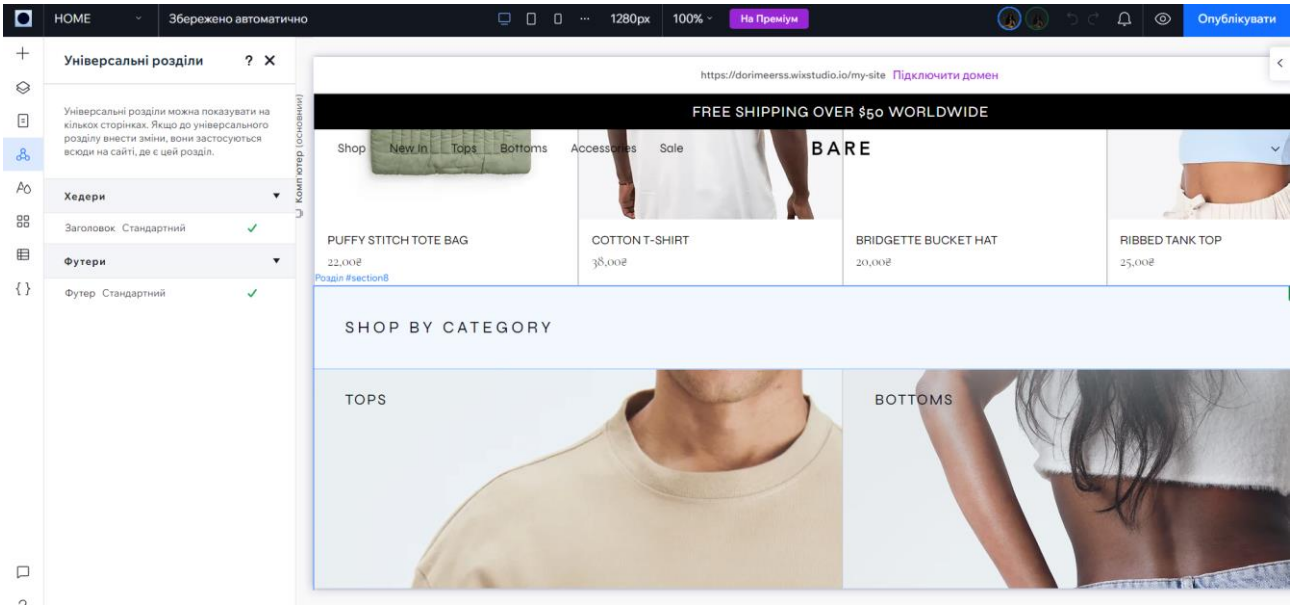


Рис. 1.6. Інтерфейс Wix

**Bubble**

Bubble - це конструктор веб-додатків, який дозволяє користувачам керувати базами даних, автентифікацією користувачів тощо за допомогою візуального інтерфейсу.[11]

Плюси:

- Пропонує більш глибоку функціональність, наприклад, бази даних та автентифікацію користувачів
- Має зростаючий ринок плагінів для додаткової функціональності

Мінуси:

- Потрібен час, щоб освоїтись, в тому числі більш просунуті функції
- Може бути менш ефективним, ніж традиційні кодовані додатки

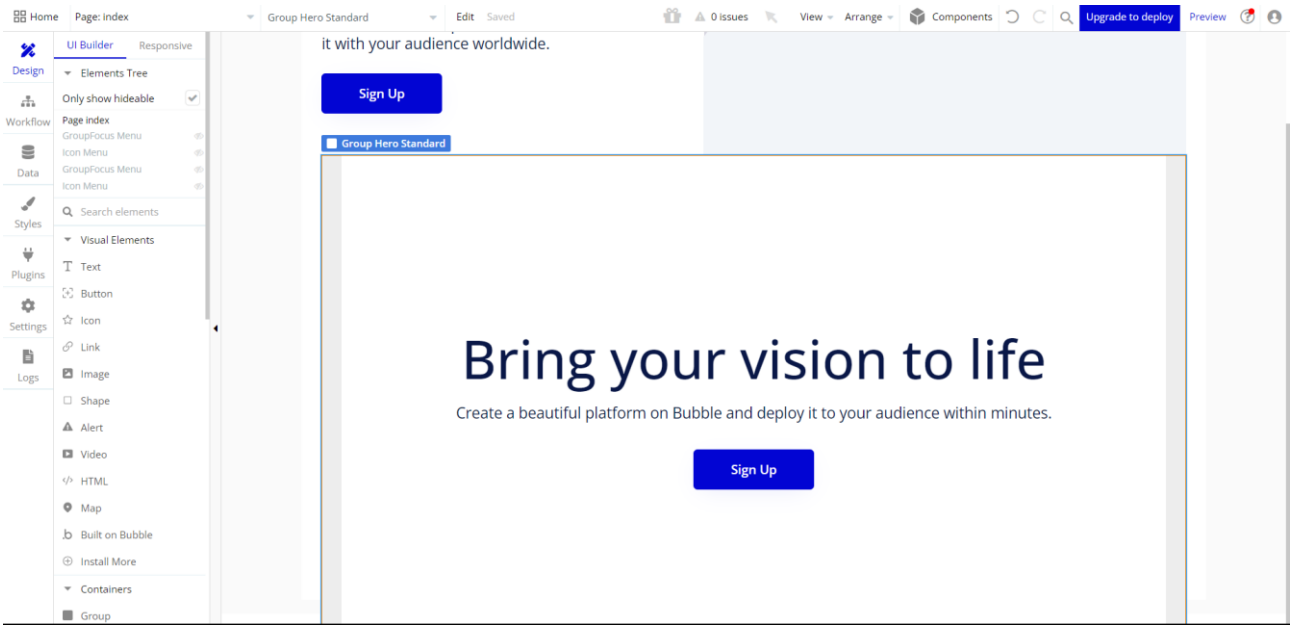


Рис. 1.7. Інтерфейс Bubble

**Webflow**

Webflow поєднує дизайн і розробку у візуальному інтерфейсі, дозволяючи дизайнерам створювати адаптивні веб-сайти без написання коду.[12]

Плюси:

- Легко поєднує дизайн і розробку завдяки наявним інструментам
- Можливість експортувати чистий, придатний для використання код

Мінуси:

- Може бути непосильним для абсолютних новачків
- Значна частина функцій входить до преміум сегменту і коштує дорожче

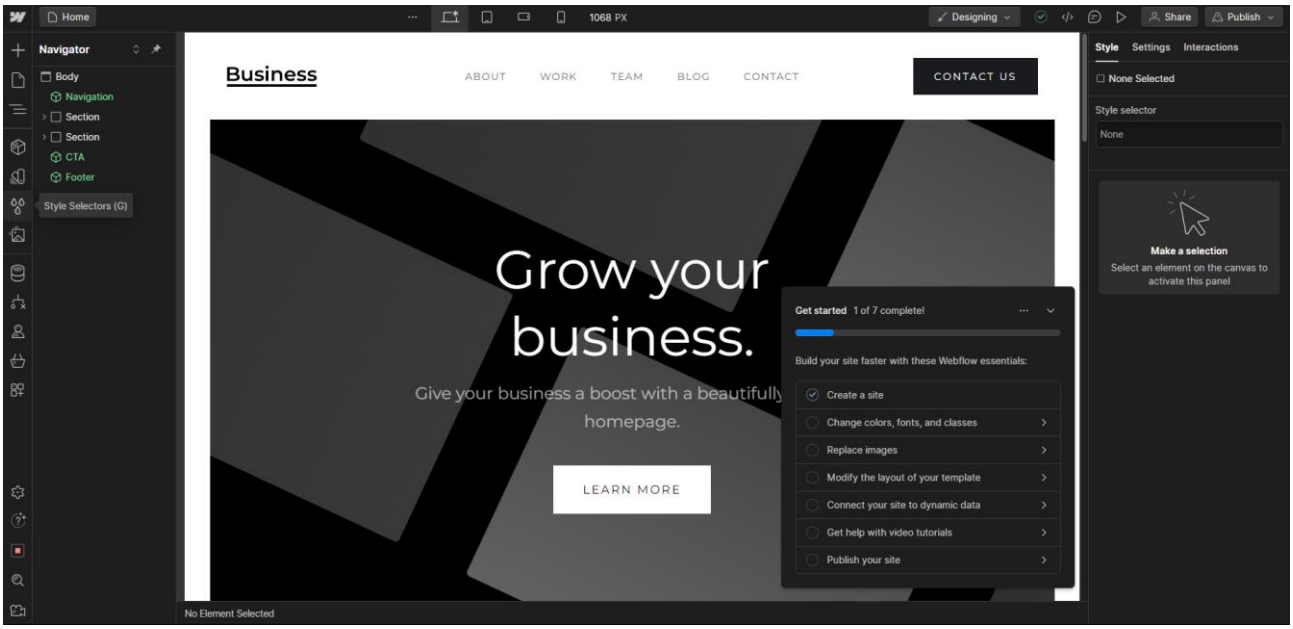


Рис. 1.8. Інтерфейс Webflow

**Adalo**

Adalo спеціалізується на створенні мобільних і веб-додатків з використанням візуального інтерфейсу. Він використовує компонентний підхід, що дозволяє створювати додатки за допомогою перетягування.[13]

Плюси:

- Орієнтований на мобільні пристрої та загалом на створення мобільних додатків.
- Пропонує ряд шаблонів для початку розробки.

Мінуси:

- Можуть виникнути проблеми з масштабуванням для дуже великої кількості користувачів.
- Глибокі налаштування можуть бути обмежені.

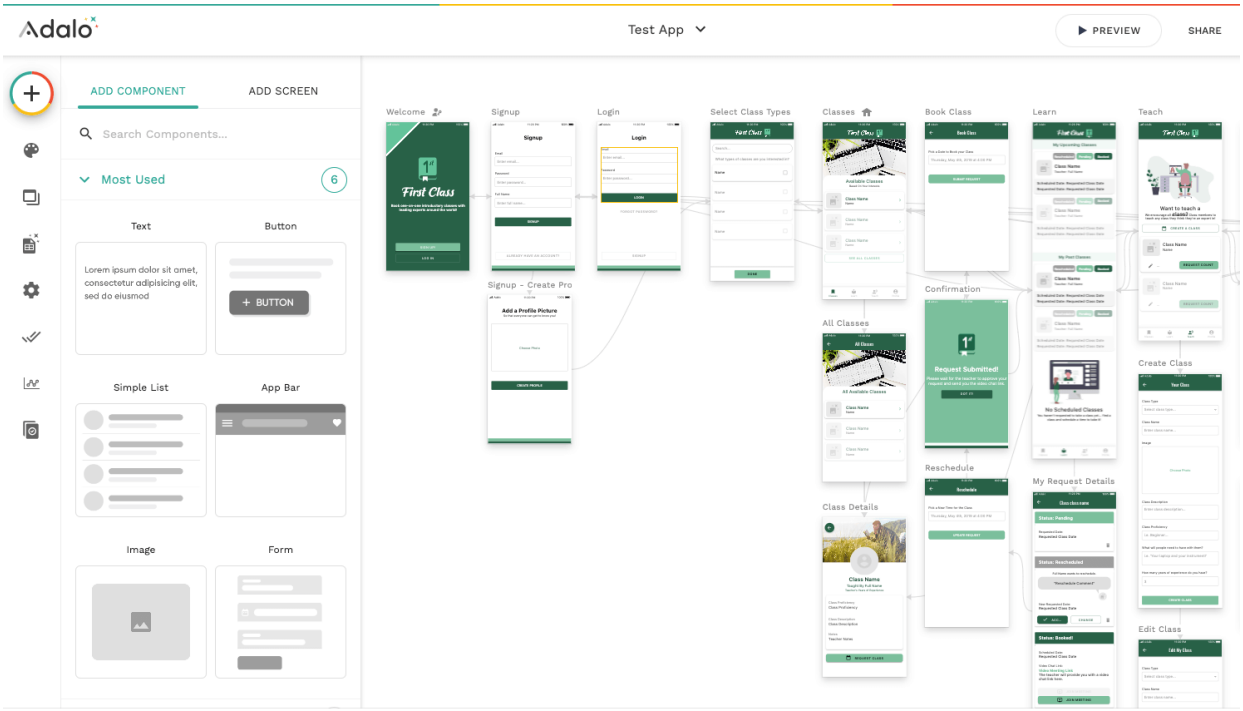


Рис. 1.9. Інтерфейс Adalo

**Glide**

Glide використовує Google Таблиці як бекенд, що дозволяє користувачам створювати мобільні додатки безпосередньо з даних своїх таблиць.[14]

Плюси:

- Простий у використанні, особливо для тих, хто знайомий з Google Таблицями.
- Незначні потреби в часі для створення додатків.

Мінуси:

- Покладається на Google Таблиці, які можуть бути не ідеальними для всіх наборів даних.
- Не підходить для дуже складних додатків.

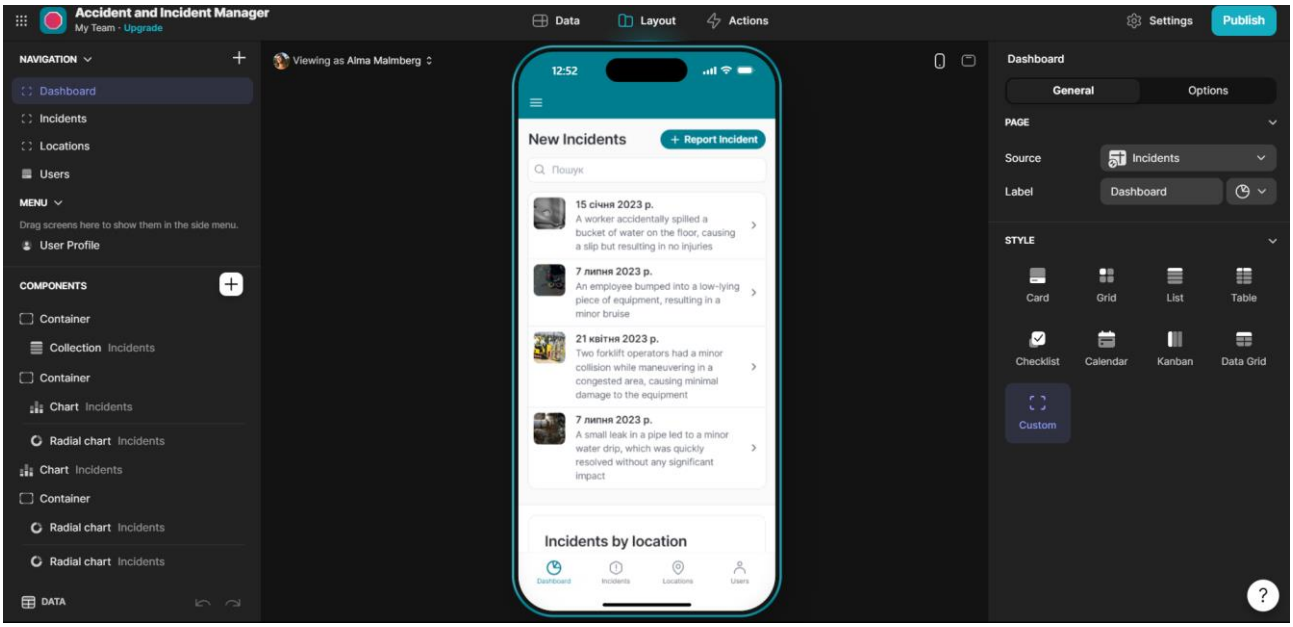


Рис. 1.10. Інтерфейс Glide

Важливо зазначити, що "найкраща" платформа часто залежить від конкретних вимог і цілей проекту. Деякі платформи можуть краще підходити для певних завдань або галузей, ніж інші.

Оскільки технологічний ландшафт постійно розвивається, можуть з'являтися нові платформи, а існуючі можуть зазнавати значних змін, тому варто бути в курсі останніх тенденцій та оглядів.

Загалом, у всіх випадках, хоч ці платформи пропонують швидку розробку та зменшення об’єму написання коду, буде необхідно йти на певні компроміси з точки зору продуктивності, гнучкості та масштабованості. Як завжди, вибір правильної платформи повинен ґрунтуватися на конкретних вимогах проекту і довгострокових цілях організації або окремої особи, що створює веб-сайт або застосунок.

## 1.5. Фінансові особливості

Вартість створення веб-сайту за допомогою платформи без коду або команди розробників може суттєво відрізнятися залежно від кількох факторів, таких як складність, функціональність, вимоги до дизайну, регіон тощо.

Нижче наведено приблизний розподіл витрат, пов'язаних з обома підходами:

**Платформа без коду:**

Підписка на платформу:

Базова: безкоштовно до $20/місяць

Бізнес/Професійна: від $20 до $500 на місяць

Enterprise: $500/місяць і вище

Доменне ім'я: від $10 до $50/рік (преміум-домени можуть коштувати значно дорожче)

Шаблон або тема (якщо не використовуєте безкоштовний):

від $0 (безкоштовно) до $200

Плагіни або доповнення (для додаткової функціональності):

Деякі з них можуть бути безкоштовними, тоді як інші можуть коштувати від $5 до $200 або більше, залежно від функціональності.

Стокові зображення або медіа: Варіюється в широких межах, але очікуйте від $1 до $50 за зображення або ресурс, якщо не використовуєте безкоштовні ресурси.

Можливі додаткові витрати:

Наймання дизайнера для подальшої кастомізації вашого сайту: від $100 до $2,000+.

Навчання або курси для вивчення платформи (для деяких користувачів): від $0 до $500

Орієнтовна сума для базового сайту без коду: від $10 до $1,500+ за перший рік (без урахування поточних витрат на підписку)

**Команда розробників:**

Доменне ім'я: від $10 до $50/рік

Веб-хостинг:

Віртуальний хостинг: від $3 до $20 на місяць

VPS-хостинг: від $20 до $100/місяць

Виділений хостинг: від $100 до $500/місяць

Дизайн:

Базовий: від $500 до $5,000

Індивідуальний та високоточний: від $5,000 до $20,000+

Розробка:

Базовий веб-сайт (кілька сторінок, стандартні функції): $1,000 до $10,000

Середньої складності (електронна комерція, інтеграція CMS): від $10,000 до $50,000

Висока складність (кастомні додатки, інтеграції, корпоративні рішення): $50 000 і вище

Створення контенту (якщо ви наймаєте професіоналів):

від $50 до $500 за одиницю, залежно від складності.

SEO та оптимізація: від $500 до $10 000+ (може значно відрізнятися залежно від цілей та регіону)

Обслуговування та оновлення:

Щомісячна або погодинна оплата: від $50 до $150/год (залежить від регіону та агентства)

Приблизна вартість базового веб-сайту, створеного розробником: від $2,000 до $70,000+ за початкову збірку (без урахування поточного обслуговування та оновлень)

Важливо зазначити, що це приблизні цифри засновані на дослідженні найпопулярніших платформ та фріланс сайтів, і фактичні витрати можуть суттєво відрізнятися залежно від регіональних цін, конкретних вимог проекту та мінливих ринкових ставок. Платформи без коду, як правило, є більш економічно вигідними для простих проектів, тоді як найм команди розробників може бути більш доцільним для складних, індивідуальних рішень. Завжди потрібно мати кілька варіантів і чітке розуміння вимог, перш ніж обирати конкретний шлях.

## 1.6 Відмінності в процесі створення

Процес створення веб-сайту за допомогою платформи без коду та командою розробників може суттєво відрізнятися. Нижче в таблиці наведено порівняння процесів розробки веб-сайту для обох методів.[2,15]

*Таблиця 1.2*

**Процес розробки веб-сайту традиційно та з використанням платформ**

| *Створення веб-сайту за допомогою платформи* | *Створення веб-сайту командою розробників* |
|---|---|
| Визначення мети: визначення мети, функції та цільової аудиторії веб-сайту. | Збір вимог: Зустрітись із зацікавленими сторонами, щоб зібрати детальні вимоги та зрозуміти цілі веб-сайту. |
| Вибір платформи: Вибір віповідної платформи без коду, яка задовольняє потребам продукту (наприклад, Wix, Webflow або Squarespace). | Структурні схеми та макети: Створення каркасу і макетів для архітектури та дизайну сайту.<br>Можна скористуватися такими інструментами, як Figma або Adobe XD. |
| Вибір шаблону: Більшість платформ без коду пропонують різноманітні шаблони, що налаштовуються. Вибрати той, який буде найбільш відповідний для продукту за цілями та аудиторією. | Дизайн: Команда дизайнерів створює детальний дизайн сайту, зосереджуючись на користувацькому досвіді та естетиці. |
| Drag-and-Drop дизайн: Скористатися інтерфейсом сайту для розміщення елементів, налаштування макетів та створення дизайну сторінок. | Розробка: Початок створення веб-сайту, з використанням такі мови та фреймворки, як HTML, CSS, JavaScript, React тощо. |
| Додавання контенту: Наповнити сайт текстом, зображеннями, відео та іншим контентом. Деякі платформи можуть пропонувати стокові зображення або інтеграцію з контент-платформами. | Інтеграція системи управління контентом (CMS): Якщо сайт потребує регулярного оновлення контенту, інтегрувати або розробити CMS. |

*Таблиця 1.2 (продовження)*

**Процес розробки веб-сайту традиційно та з використанням платформ**

| | |
|---|---|
| Інтеграція функціональності: Реалізувати такі функції, як контактні форми, розсилки або можливості електронної комерції за допомогою вбудованих інструментів або плагінів. | Backend-розробка: Налаштування баз даних, серверних скриптів та API, написання логіки продукту. |
| Адаптивність: Переконатись, що дизайн адаптивний для мобільних пристроїв. Більшість платформ автоматично справляються з цим завданням, але не зайвим буде перевірити ще раз. | Тестування: Провести ретельне тестування, включаючи функціональне тестування, UI/UX-тестування та тестування продуктивності. |
| Тестування: Перевірити функціональність, посилання та швидкість реакції сайту. | Зворотній зв'язок та ітерації: Зібрати відгуки від зацікавлених сторін та ітераційно вдосконалюйте дизайн і функціональність. |
| Домен і хостинг: Придбайте домен і використовуйте хостинг платформи або інтегруйте його зі стороннім хостингом. | Розгортання: Придбання хостингу та домену, а потім розгортання веб-сайту на сервері. |
| Запуск і підтримка: Після того, як ви будете задоволені, опублікуйте веб-сайт. Використовуйте інструменти аналітики та управління платформою для постійного оновлення та оптимізації. | Обслуговування: Регулярне оновлення сайту, виправлення вразливостей безпеки та оптимізація на основі відгуків користувачів та аналітики. |

Виходячи з означеної вище таблиці, можемо виділити такі основні відмінності:

- Час: Платформи без коду, як правило, швидше запускають базовий веб-сайт. Однак команди розробників є більш універсальними для складних проектів.
- Кастомізація: Платформи без коду можуть мати обмеження з точки зору кастомізації та унікальних функцій, в той час як команди розробників можуть створювати кастомні функції.
- Вартість: Платформи без коду можуть бути економічно вигідними для базових веб-сайтів. Команди розробників можуть мати вищі авансові витрати, але пропонують більш персоналізовані рішення.
- Масштабованість: Хоча рішення без коду вдосконалюються, традиційно кодовані веб-сайти часто пропонують кращі можливості для масштабування та оптимізації продуктивності.
- Крива навчання: Платформи без коду розроблені так, щоб бути зручними для користувача, в той час як розробка веб-сайту традиційно вимагає різноманітних навичок (дизайн, кодування, управління базами даних і т.д.).
- Обслуговування: Платформи без коду часто включають в себе інструменти для обслуговування та оновлення. За наявності команди розробників поточна підтримка може бути більш практичною, але також більш адаптованою до потреб сайту.

По суті, найкращий підхід залежить від вимог сайту, бюджету та довгострокових цілей. У той час як платформи без коду чудово підходять для швидких, економічно ефективних рішень зі стандартним функціоналом, команда розробників краще підходить для індивідуальних, масштабованих і складних проектів.

## Висновки до розділу 1

У цьому розділі оглянуто базові знання про платформи з низьким рівнем коду або платформи без коду - це інструменти, що поєднують традиційне написання коду з візуальною розробкою. Вони пропонують швидший цикл розробки, ніж традиційні методи, зберігаючи при цьому можливість заглиблюватися в код для виконання складних завдань.

Проаналізовано основні цілі використання low-code і no-code платформ, та їх роль в сучасному становищі розробки, як виступають проміжною ланкою між візуальним дизайном користувача і технічними тонкощами розробки програмного забезпечення. Вони займаються перекладом візуального дизайну у функціональні додатки, керують технічними складнощами та надають спрощений інтерфейс для користувачів, які можуть створювати свої сайти та застосунки  без глибоких технічних знань. Таким чином вони дають набагато ширший доступ до розробки для людей з мінімальними технічними знаннями, що робить нижній поріг створення додатків більш доступним для людей без специфічної освіти.

За рахунок порівняння виявлено відмінності між традиційною розробкою та розробкою з використанням low-code і no-code платформи, оцінені приблизні фінансові витрати. Проведено базовий аналіз існуючих на ринку рішень, їх переваг та недоліків.

# РОЗДІЛ 2
# МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## 2.1 Аналіз та формування вимог до програмного забезпечення

В рамках даного розділу ми розглянемо створення мінімального функціоналу веб-сайту для магазина одягу за допомогою традиційних методів розробки та no-code платформи.

Зважаючи на призначення даного веб-сайту, можемо зробити висновок, що основними функціональними вимогами мають бути:

- авторизація користувача
- перегляд товарів
- додавання, редагування та видалення товару в кошику
- можливість вибору кольору та розміру товару
- можливість адміністратору додавати, редагувати, видаляти нові позиції в кошику
- можливість адміністратору переглядати замовлення клієнтів

У таблицях нижче описані функціональні вимоги.

*Таблиця 2.1*

**Опис функціональної вимоги REQ01**

| | |
|---|---|
| Номер | REQ01 |
| Назва | Реєстрація користувача |
| Опис | Незареєстрований користувач має змогу зареєструватися за допомогою своєї електронної пошти або створити акаунт використовуючи Google |

*Таблиця 2.2*

**Опис функціональної вимоги REQ02**

| | |
|---|---|
| Номер | REQ02 |
| Назва | Авторизація користувача |
| Опис | Зареєстрований користувач має змогу авторизуватися використовуючи створений раніше акаунт або аккаунт Google. Логін та пароль зберігаються в базі даних в зашифрованому вигляді. Якщо введено правильні дані для входу, користувача буде авторизовано в застосунку |

*Таблиця 2.3*

**Опис функціональної вимоги REQ03**

| | |
|---|---|
| Номер | REQ03 |
| Назва | Редагування даних про товари |
| Опис | Адміністратор може додавати, змінювати та видаляти товари. |

*Таблиця 2.4*

**Опис функціональної вимоги REQ04**

| | |
|---|---|
| Номер | REQ04 |
| Назва | Редагування даних про товари |
| Опис | Адміністратор може додавати, змінювати та видаляти товари. |

*Таблиця 2.5*

**Опис функціональної вимоги REQ05**

| | |
|---|---|
| Номер | REQ05 |
| Назва | Формування замовлення |
| Опис | Авторизований користувач має змогу зробити замовлення |

*Таблиця 2.6*

**Опис функціональної вимоги REQ06**

| | |
|---|---|
| Номер | REQ06 |
| Назва | Вибір способу оплати |
| Опис | Авторизований користувач може вибрати спосіб оплати |

*Таблиця 2.7*

**Опис функціональної вимоги REQ07**

| | |
|---|---|
| Номер | REQ07 |
| Назва | Перегляд всіх товарів |
| Опис | Авторизований користувач має змогу переглянути список товарів. При виборі конкретного товару користувач може переглянути склад, опис та іншу інформацію. |

*Таблиця 2.8*

**Опис функціональної вимоги REQ08**

| | |
|---|---|
| Номер | REQ08 |
| Назва | Додавання товару в кошик |
| Опис | Авторизований користувач має змогу додати товар до кошика, вибрати кількість, розмір, колір. Після підтвердження товар буде додано до кошика. |

На основі описаних вимог сформуємо відповідні варіанти використання. Діаграму варіантів використання наведено у додатку.

*Таблиця 2.9*

**Варіант використання UC01**

| | |
|---|---|
| Назва | Авторизація користувача |
| Опис | Зареєстрований користувач авторизується в системі |
| Учасник | Неавторизований користувач |

*Таблиця 2.9 (продовження)*

**Варіант використання UC01**

| | |
|---|---|
| Передумови | Неавторизований користувач має аккаунт Google або зареєстрований в системі.<br>Відкрита сторінка авторизації. |
| Постумови | Користувача авторизовано в системі. |
| Основний сценарій | Користувач вводить свої дані, щоб пройти валідацію або натискає кнопку «Увійти за допомогою Google»;<br>Користувач перевіряє введенні дані і натискає кнопку «Увійти». |

*Таблиця 2.10*

**Варіант використання UC02**

| | |
|---|---|
| Назва | Авторизація користувача |
| Опис | Зареєстрований користувач авторизується в системі |
| Учасник | Неавторизований користувач |
| Передумови | Неавторизований користувач має аккаунт Google або зареєстрований в системі.<br>Відкрита сторінка авторизації. |
| Постумови | Користувача авторизовано в системі. |
| Основний сценарій | Користувач вводить свої дані, щоб пройти валідацію або натискає кнопку «Увійти за допомогою Google»;<br>Користувач перевіряє введенні дані і натискає кнопку «Увійти». |

*Таблиця 2.11*

**Варіант використання UC03**

| | |
|---|---|
| Назва | Авторизація адміністратора |
| Опис | Зареєстрований користувач авторизується в системі як адміністратор |

*Таблиця 2.11 (продовження)*

**Варіант використання UC03**

| | |
|---|---|
| Учасник | Неавторизований користувач |
| Передумови | Неавторизований користувач має аккаунт Google або зареєстрований в системі як адміністратор.<br>Відкрита сторінка авторизації. |
| Постумови | Користувач авторизувався в системі як адміністратор |
| Основний сценарій | Користувач вводить свої дані, щоб пройти валідацію або натискає кнопку «Увійти за допомогою Google»;<br>Користувач перевіряє введенні дані і натискає кнопку «Увійти». |

*Таблиця 2.12*

**Варіант використання UC04**

| | |
|---|---|
| Назва | Додавання товарів до кошика |
| Опис | Користувач додає товар до кошика |
| Учасник | Авторизований користувач |
| Передумови | Користувач знаходиться на сторінці і переглядає список товарів |
| Постумови | Користувач додав обраний товар до кошика |
| Основний сценарій | Користувач обирає бажаний товар зі списку постачальника;<br>Переглядає деталі, виставляє бажану кількість товару, колір, розмір та натискає кнопку «Додати до кошика» |

*Таблиця 2.13*

**Варіант використання UC05**

| | |
|---|---|
| Назва | Створення замовлення |
| Опис | Користувач створює нове замовлення на основі кошику |
| Учасник | Авторизований користувач |
| Передумови | Користувач знаходиться на сторінці кошику |
| Постумови | Створено нове замовлення |
| Основний сценарій | Користувач переглядає наявні у кошику товари, за необхідності змінює їх кількість, натискає кнопку «Замовити», заповнює інформацію щодо товару, номер телефону, адресу доставки, деталі замовлення, за наявності вводить промокоди та натискає кнопку «Підтвердити» |

*Таблиця 2.14*

**Варіант використання UC06**

| | |
|---|---|
| Назва | Додавання товару |
| Опис | Адміністратор додає новий товар до списку |
| Учасник | Авторизований адміністратор |
| Передумови | Користувач авторизований як адміністратор;<br>Користувач знаходиться на адміністраційній панелі |
| Постумови | До списку товарів додано новий товар |
| Основний сценарій | Адміністратор натискає кнопку «Додати новий товар», заповнює форму інформацією про товар, при наявності додаткової інформації, вносить її і натискає кнопку «Готово» |

*Таблиця 2.15*

**Варіант використання UC07**

| | |
|---|---|
| Назва | Перегляд списку замовлень |
| Опис | Адміністратор переглядає список нових замовлень |
| Учасник | Авторизований адміністратор |
| Передумови | Користувач авторизований як адміністратор;<br>Користувач знаходиться на адміністраційній панелі |
| Постумови | Адміністратор переглянув список замовлень |
| Основний сценарій | Адміністратор переглядає нові замовлення, отримує доступ до деталей замовлення та адреси доставки; |

*Таблиця 2.16*

**Варіант використання UC08**

| | |
|---|---|
| Назва | Зміна товару |
| Опис | Адміністратор замінює інформацію про існуючий товар у списку |
| Учасник | Авторизований адміністратор |
| Передумови | Користувач авторизований як адміністратор;<br>Користувач знаходиться на адміністраційній панелі |
| Постумови | Адміністратор змінив інформацію про товар |
| Основний сценарій | Адміністратор натискає кнопку «Змінити», заповнює форму іншою інформацією про товар і натискає кнопку «Готово» |

Нефункціональні вимоги для даної задачі – це англійська, українська мова інтерфейсу, та комфортний для взаємодії інтерфейс.

Таким чином, можна переходити безпосередньо до проєктування веб-сайту двома способами – з використанням no-code платформи та звичайного написання коду.

## 2.2 Розробка веб-сайту методом програмування

### 2.2.1 Вибір інструментів для програмування

При розробці клієнтської частини вибір був зроблений на користь ASP.NET MVC через наступні причини:

- Постійна підтримка платформи .NET
- Наявність багатої кількості навчальних матеріалів та розгалуженої документації.
- Досвід у використанні цієї платформи.
- Вигода в інтеграції з серверною частиною через спільне використання інструментів.

Для розробки використовувався IDE Rider від JetBrains з таких причин:

- Висока продуктивність та розширений функціонал.
- Безкоштовне використання для студентів.
- Продуктивне автодоповнення коду та спрощення навігації по проекту завдяки функціям з ReSharper.
- Підтримка фронтенд технологій, включно з HTML, JavaScript та CSS.
- Зручність роботи з плагінами, сумісними з ReSharper та IntelliJ.
- Широкий доступ до додаткових інструментів, таких як dotPeek та dotMemory.

Figma була використана для проектування UI через:

- Використання компонентів та автолейаутів для спрощення верстки.
- Безкоштовний доступ.
- Широкий вибір плагінів для поліпшення процесу.

- Активна спільнота та підтримка експорту HTML/CSS коду та графічних матеріалів.
- Інтеграція з зовнішніми сервісами для імпорту дизайну.

Razor Pages були обрані для реалізації представлень, оскільки це дозволяє інтегрувати код C# з HTML, забезпечуючи гнучкість у відображенні модельних даних.

Для серверної частини було обрано .NET 6 та C# з таких причин:

- Модульність та легкість оновлення компонентів через NuGet.
- Простота у вивченні та велика кількість доступних бібліотек.
- Активна користувацька спільнота.
- Об'єктно-орієнтованість C# відповідає вимогам розробки.

PostgreSQL була обрана для управління базами даних через відкритий код, активну спільноту та новітні функції. Для роботи з базою даних будемо використовувати DataGrip.

### 2.2.2 Архітектура програмного забезпечення

Під час встановлення вимог до проекту було вирішено використовувати монолітну структуру, яка відповідає заданим критеріям.

Шар доступу до даних — це частина програми, що спрощує взаємодію з даними, збереженими у сховищі. Цей шар забезпечує вищий рівень абстракції для користувацьких модулів.

Бізнес-логіка — це серце всіх бізнес-процесів, виконує обчислення, взаємодіє з шаром даних та управляє передачею даних до шару відображення.

Шар відображення — це верхній рівень програми, представлений як UI, з яким взаємодіють користувачі. В основі цього лежить графічне відтворення даних, доступне з різних пристроїв: телефонів, комп'ютерів тощо. У нашому контексті для взаємодії з цим рівнем використовується веб-браузер.

Основні елементи, які були використані для роботи: репозиторій, одиниця роботи, DTO, контекст даних та служба.

Репозиторій - це структура, яка втілює відомий шаблон проектування і служить для забезпечення зв’язку між джерелами даних та іншими частинами програми та забезпечує абстракцію від конкретних джерел даних.

Одиниця роботи (unit of work) - це підхід до координації роботи різних репозиторіїв з одним контекстом даних, що допомагає оптимізувати роботу з базою даних.

DTO (Об’єкт передачі даних) - це зв’язок між сервісами бізнес-логіки та шаром даних, без будь-якої вбудованої логіки.

Контекст даних - це інструмент для спілкування з базою даних, що включає в себе ряди сутностей, що представляють дані в базі.

Сервіс - це компонент, який зосереджується на виконанні певних завдань, наприклад, обробці запитів або логуванні, і може бути використаний багато разів для повторюваних операцій.

MVC - це підхід до структуризації програми на три сектори:

- Model - елемент, що відповідає за логіку та діє незалежно від View та Controller.
- View - відображає інформацію з моделі, але без прямого втручання в неї.
- Controller - опрацьовує вхідні дані, перетворюючи їх у команди для Model або View.

У процесі розробки був застосований компонент ViewModel, який представляє собою клас із полями для строго типізованого відображення та передачі даних від контролера.

Для поєднання бізнес-шару і шару відображення був застосований принцип Dependency Injection, який дозволяє інтегрувати зовнішні компоненти з використанням підходу IoC.

Для зберігання інформації про замовлення, клієнтів та товари створено базу даних, схему якої наведено на рисунку 2.1.

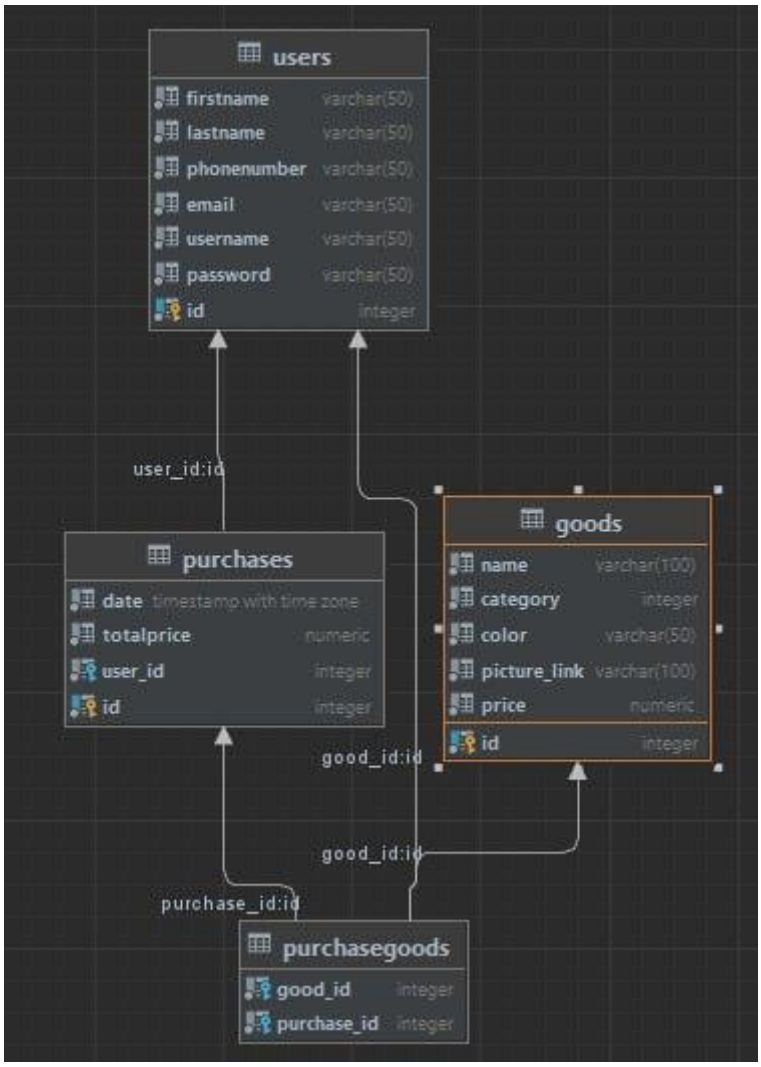


Рис. 2.1. Схема бази даних

Повну діаграму Presentation Layer наведено в додатку. На рисунках 2.2 – 2.5 наведено основні модулі даного шару представлення, які відповідають за взаємодію кінцевого користувача з нашим веб-сайтом та обробку інформації, що поступає на клієнтську частину.

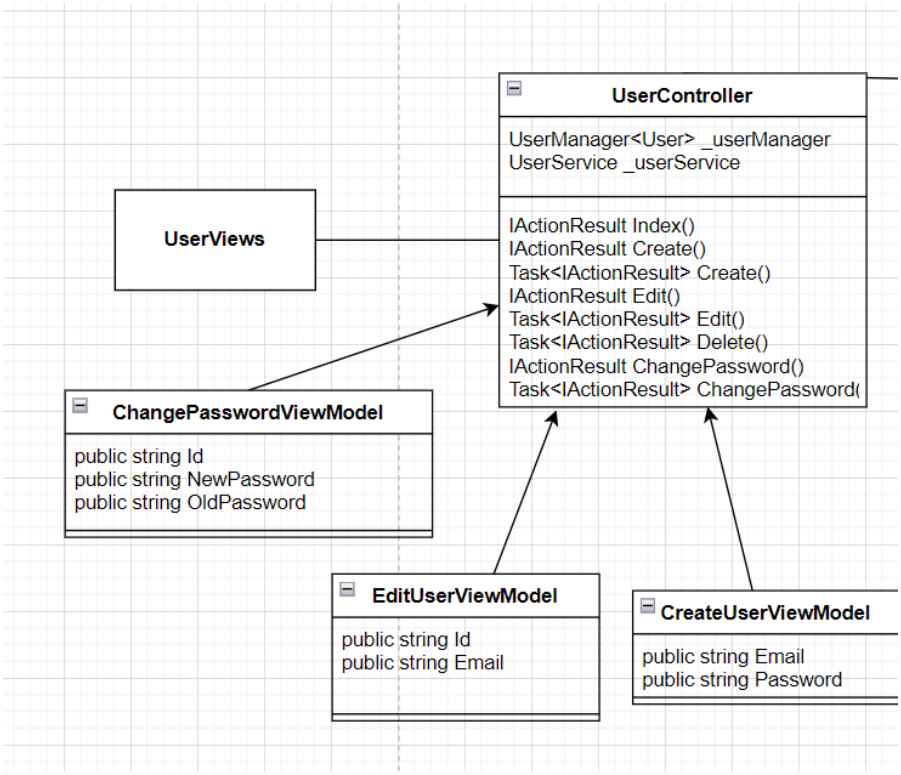


Рис. 2.2. Модуль UserController (Presentation Layer)

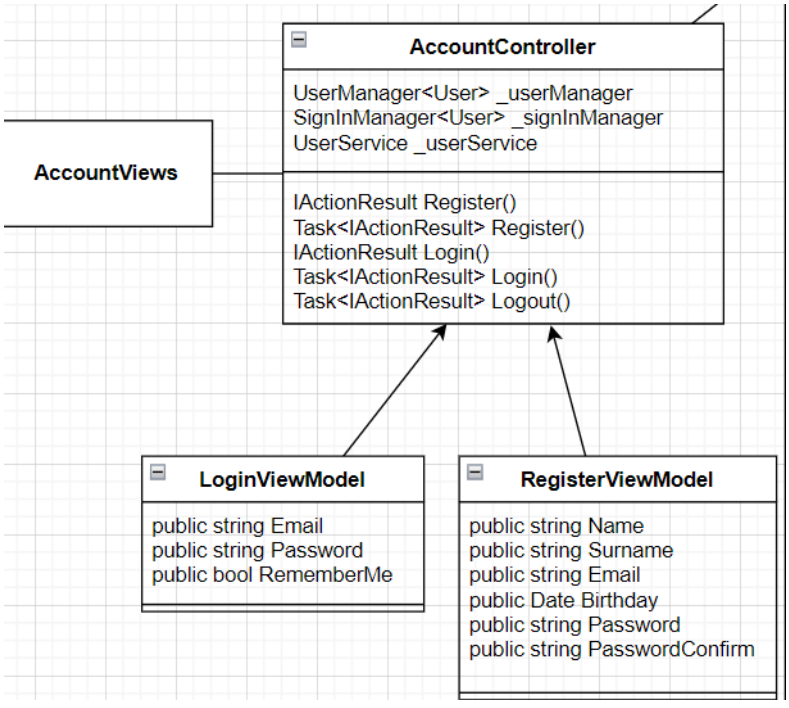


Рис. 2.3. Модуль AccountController (Presentation Layer)

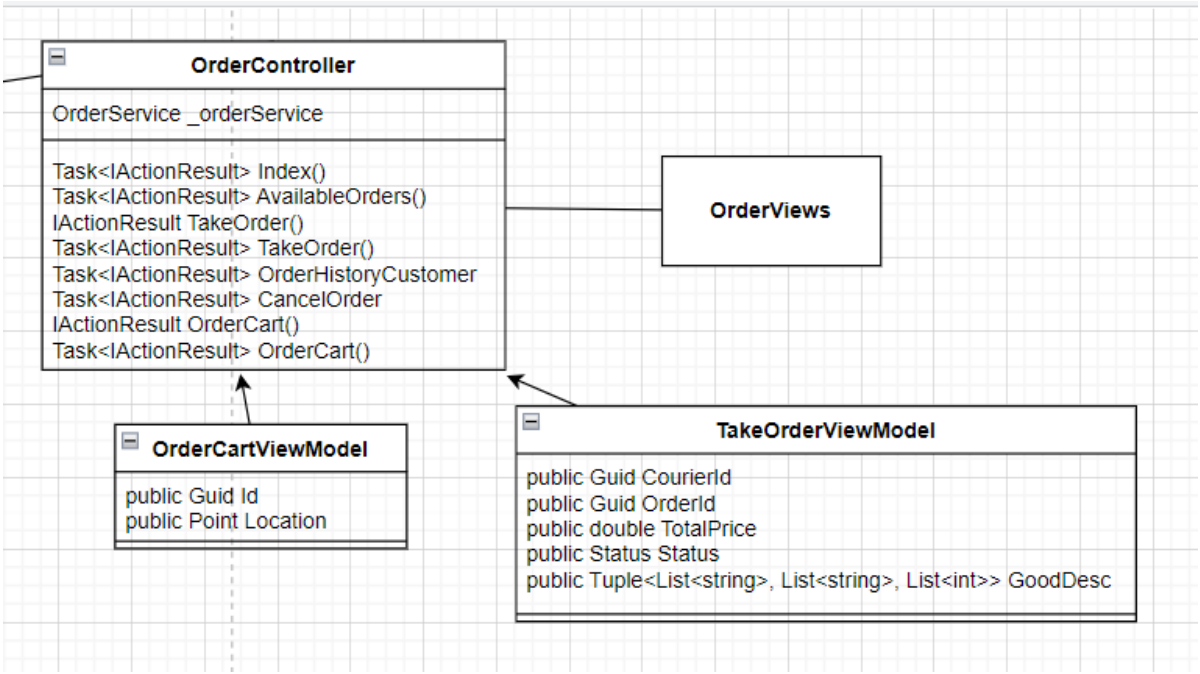


Рис. 2.4. Модуль OrderController (Presentation Layer)

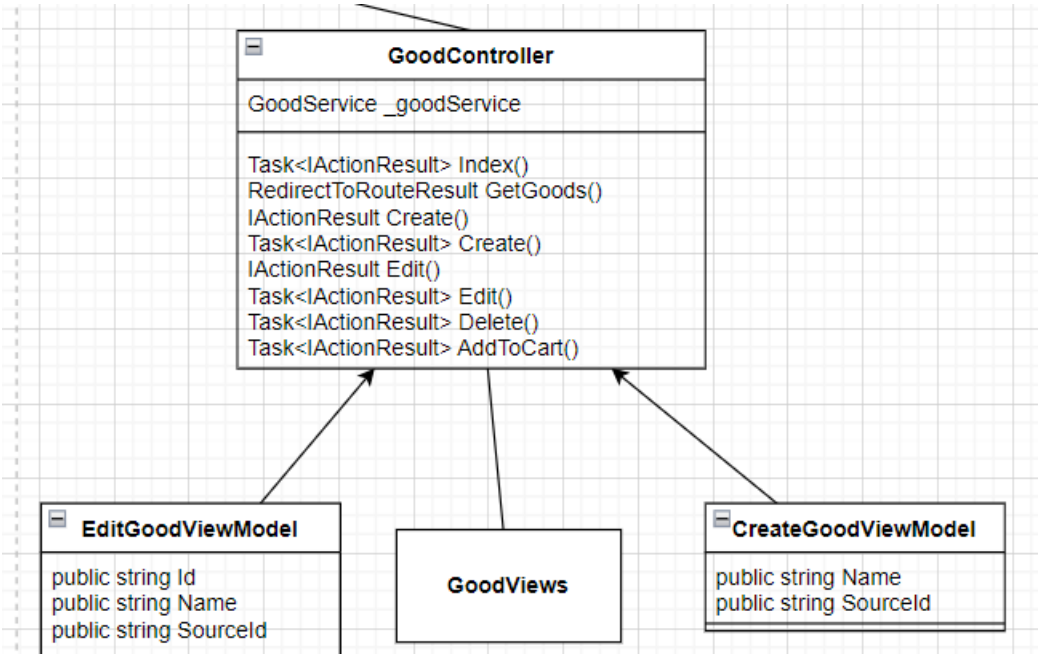


Рис. 2.5. Модуль GoodController (Presentation Layer)

Діаграму Business Logic Layer наведено в додатках.

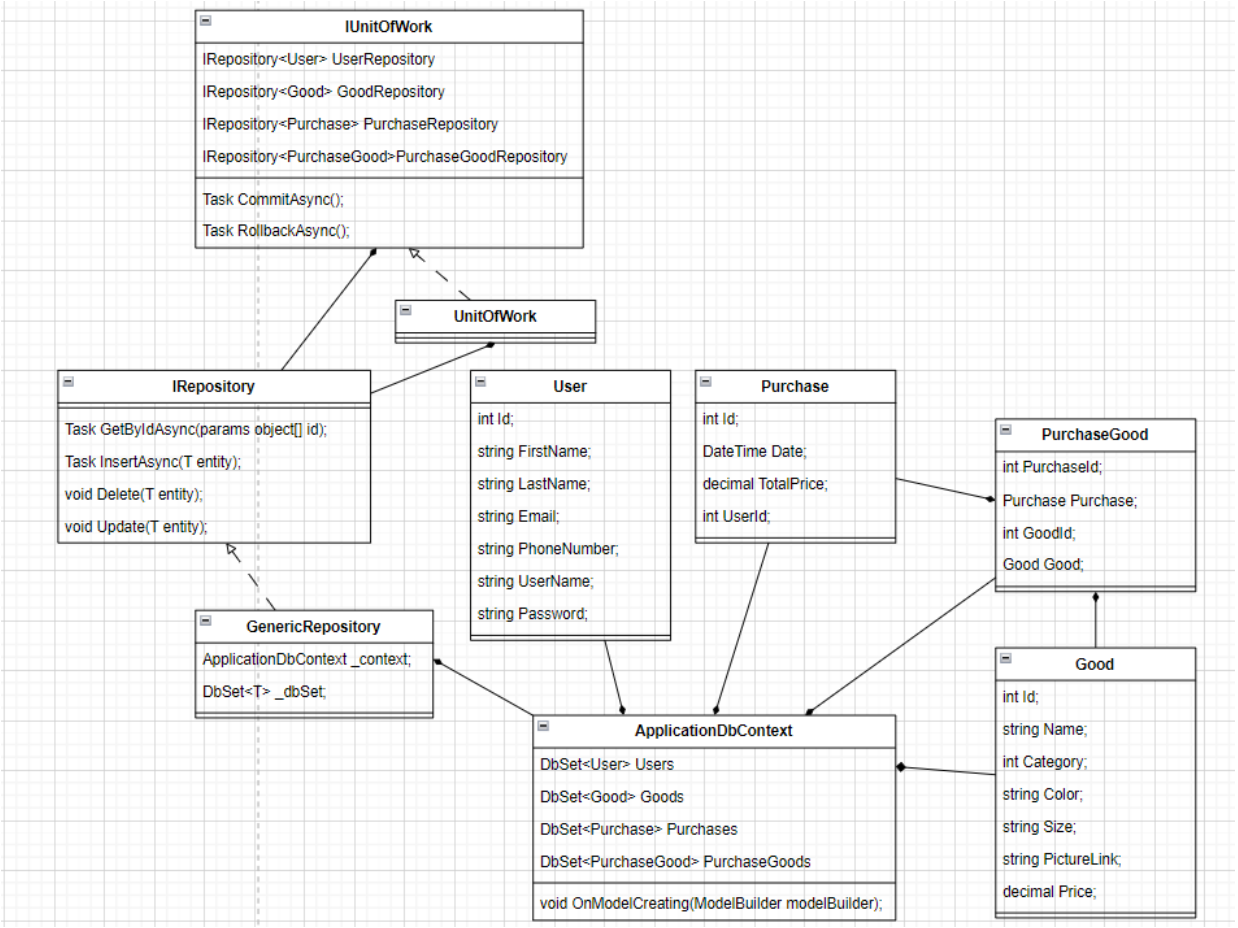


Рис. 2.6. Data Access Layer

Фінальну структуру проєкту наведено на рисунку 2.7.

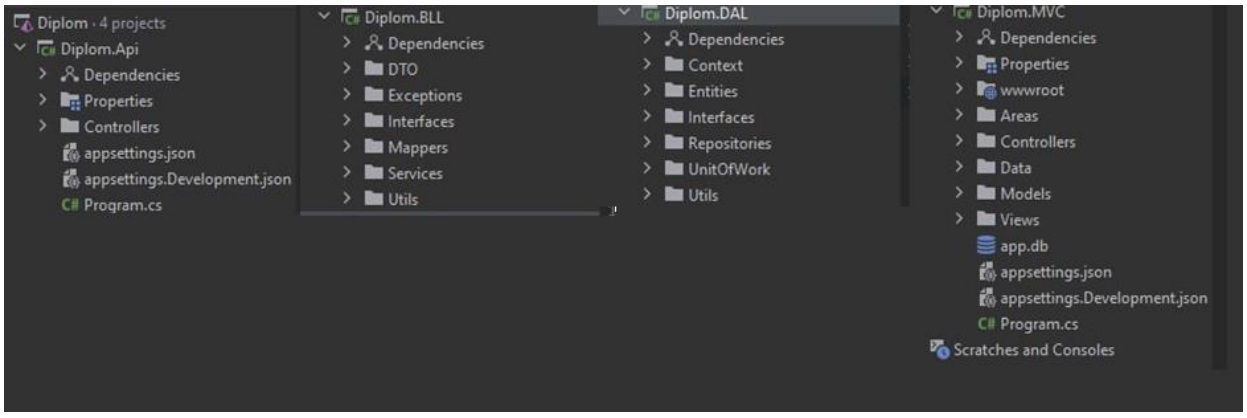


Рис. 2.7. Структура проєкту

## 2.3 Розробка веб-сайту з використанням no-code платформи

### 2.3.1 Вибір інструментів розробки для розробки з використанням no-code платформи

Середовищем розробки для no-code реалізації тієї ж задачі було обрано Wix, відомий конструктор веб-сайтів. Wix пропонує:

- інтуїтивно зрозумілий візуальний редактор з drag-and-drop інтерфейсом, що дозволяє легко створювати і налаштовувати ваш веб-сайт
- має багато готових дизайнів і шаблонів
- пропонує інтегровану платформу для створення і керування інтернет-магазином, де можна додавати товари, налаштовувати способи оплати і доставки, вести інвентаризацію тощо
- можливість за необхідності скористатися написанням коду, для того щоб доповнити існуючий функціонал.

Через це він чудово підходить для людей, мало знайомих з проектуванням веб-сайтів. Незначними недоліками можна вважати лише залежність від хостингу і лімітованість частини функцій, які входять до платних тарифів.

### 2.2.2 Проектування інтерфейсу

Проектування інтерфейсу відбувається методом drag-and drop, що являэ собою перетягування компонентів на полотні, відповідно розміщуючи всі необхідні модулі. Під час початку роботи, платформа пропонує нам почати з повністю чистого полотна (Рисунок 2.8.), або скористатись одним з наявних шаблонів для старту.

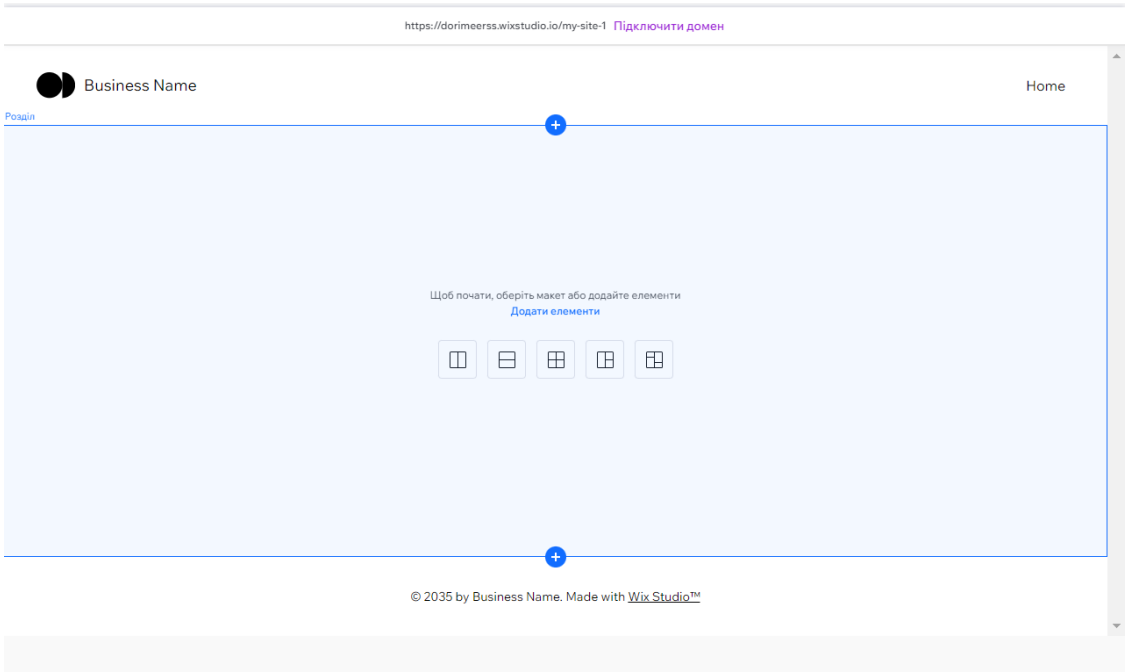


Рис. 2.8. Пустий шаблон

Стандартна сторінка, з якої ми починаємо, пропонує загальну структуру, притаманну будь-якому веб-сайту: хедер, тіло та футер. Хедер та футер налаштовуються як стандартні компоненти і використовуються на всьому сайті в подальшому.

Для наповнення сторінки в боковій вкладці натискаємо «+», де можемо ознайомитись зі списком доступних компонентів, заготованих системою. Він наведений на рисунку 2.9.

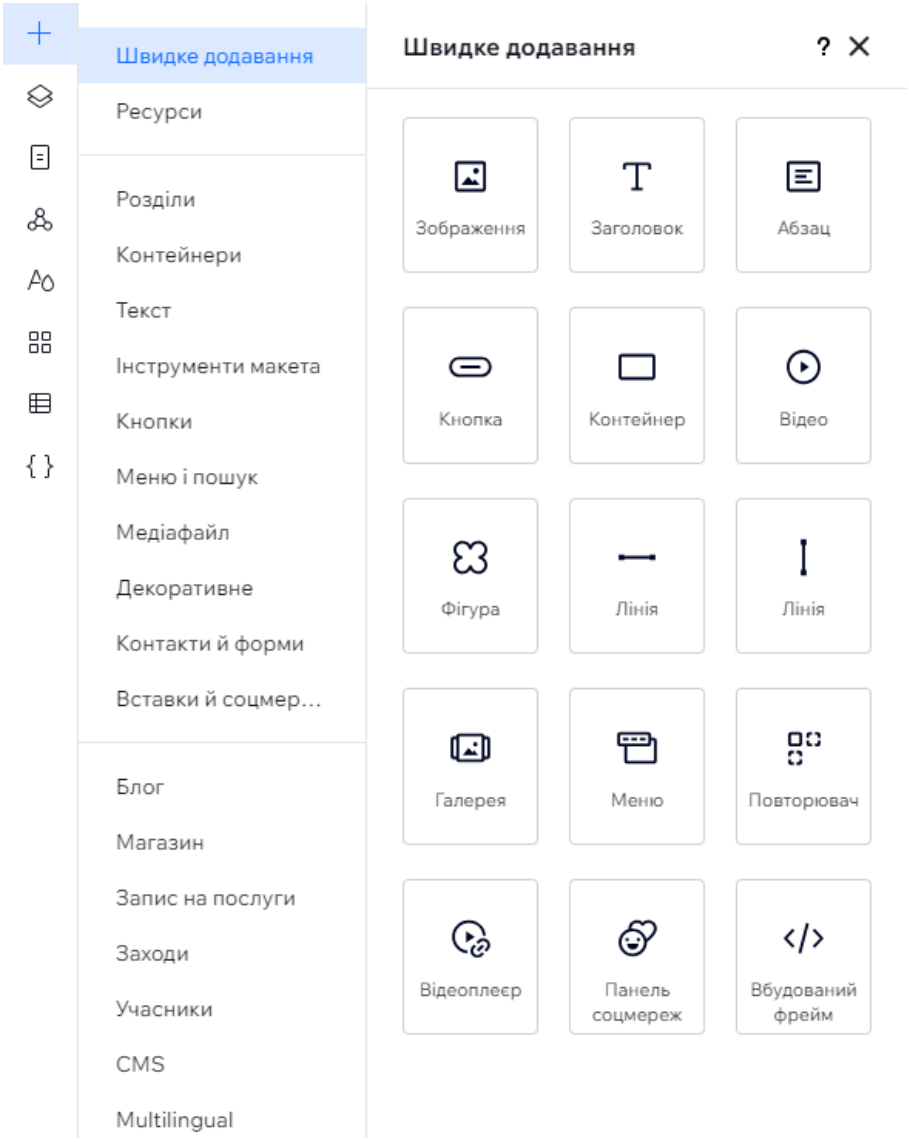


Рис. 2.9. Вкладка додавання елементів

За розміщенням елементів, відношеннями, групуванням і так далі можна стежити за допомогою вкладки шарів.

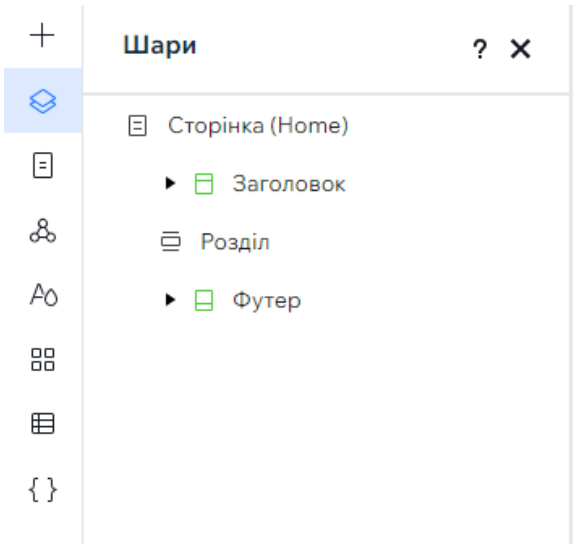


Рис. 2.10. Вкладка шарів

Оскільки сайт складатиметься не з однієї сторінки, для операцій з ними використовується вкладка сторінок сайту, через яку пізніше будуть додаватись нові сторінки: головна, картка товару, кошик, тощо.

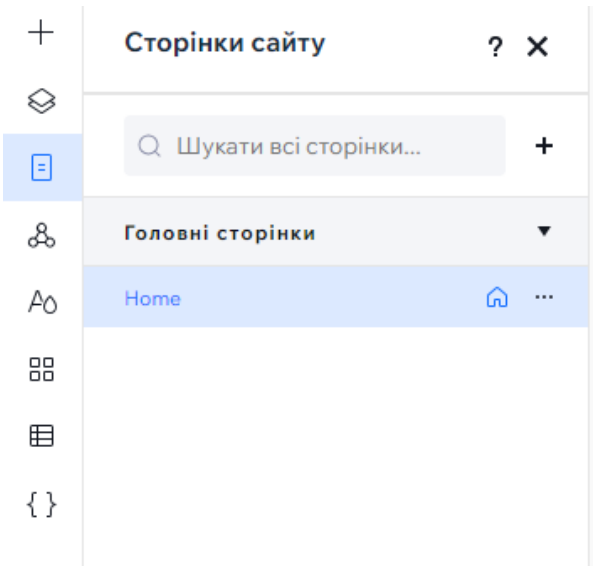


Рис. 2.11. Вкладка сторінок

Стилями сайту, шрифтами можна керувати за допомогою вкладки стилів. (рисунок 2.12.) Окрім цього, на Wix також доступна можливість доповнити можливості сайту за допомогою програмування, налаштувати CMS та встановити стандартні додатки для більш комфортного проектування.

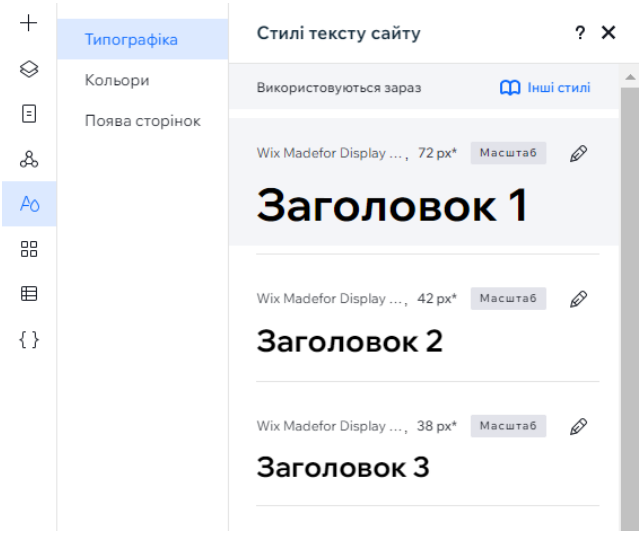


Рис. 2.12. Вкладка стилів

Для того, щоб спроектувати комфортний у використанні та візуально привабливий сайт, варто дотримуватись певних методів організації контенту на сторінці. Для цього використовуються інструменти макета, такі як сітки. Вони спрощують проектування та сприяють подальшій адаптивності сайту. Приклади таких сіток наведено нижче на рисунку 2.13

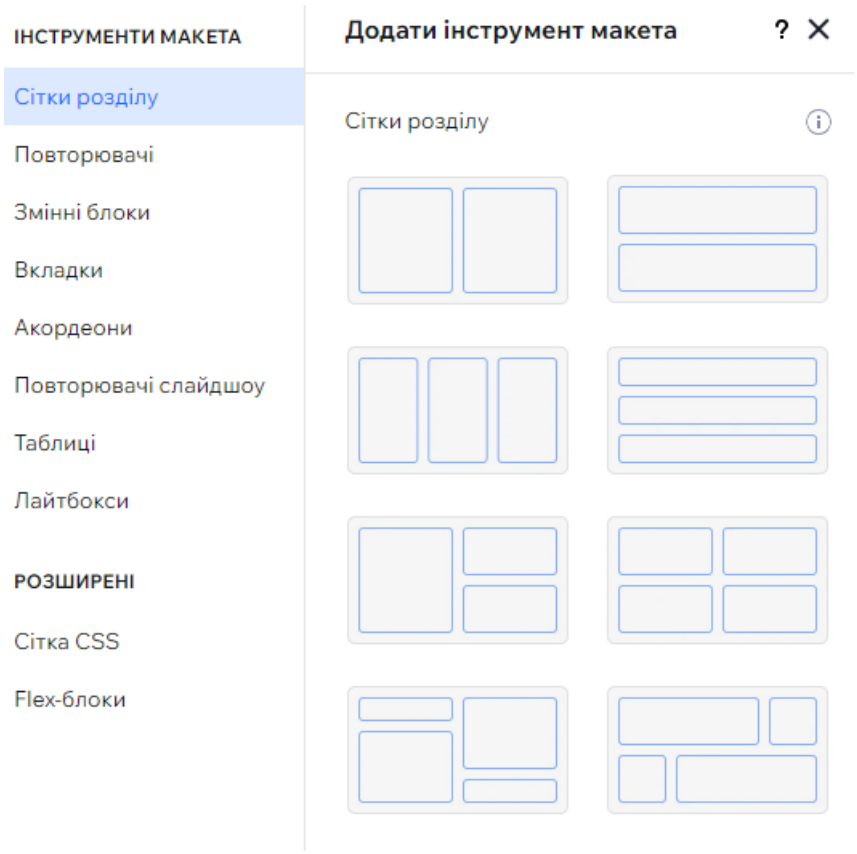


Рис. 2.13. Інструменти макету

Загалом, головна сторінка має характерні розділи. На даній ноу-код платформі підготовано декілька базових шаблонів, які можна використати для власного сайту.

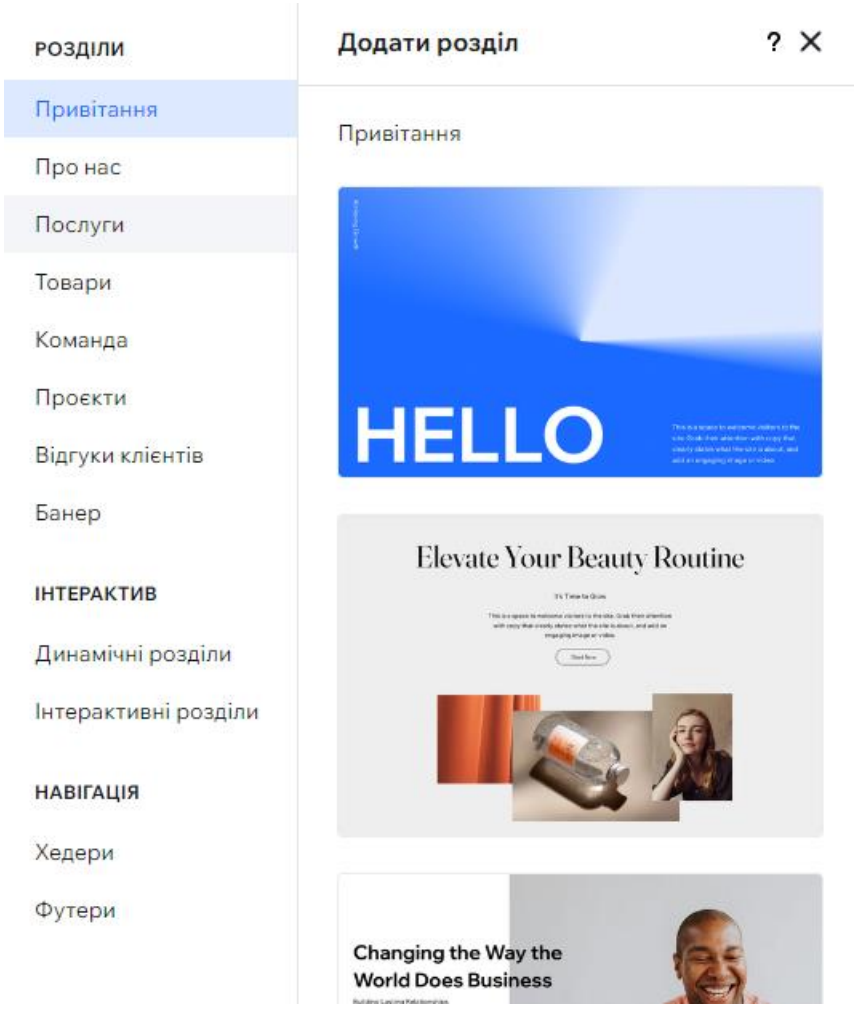


Рис. 2.14. Готові шаблони

Загалом, для мінімального прототипу нам потрібно близько п’яти сторінок – головна, каталог всіх товарів, картка товару, кошик та оплата.

Використовуючи контейнери та сітку ділимо сторінку на розділи, групуючи різні об’єкти між собою. Також, можна скористатись вже готовими конструкціями з розділу «Магазин».

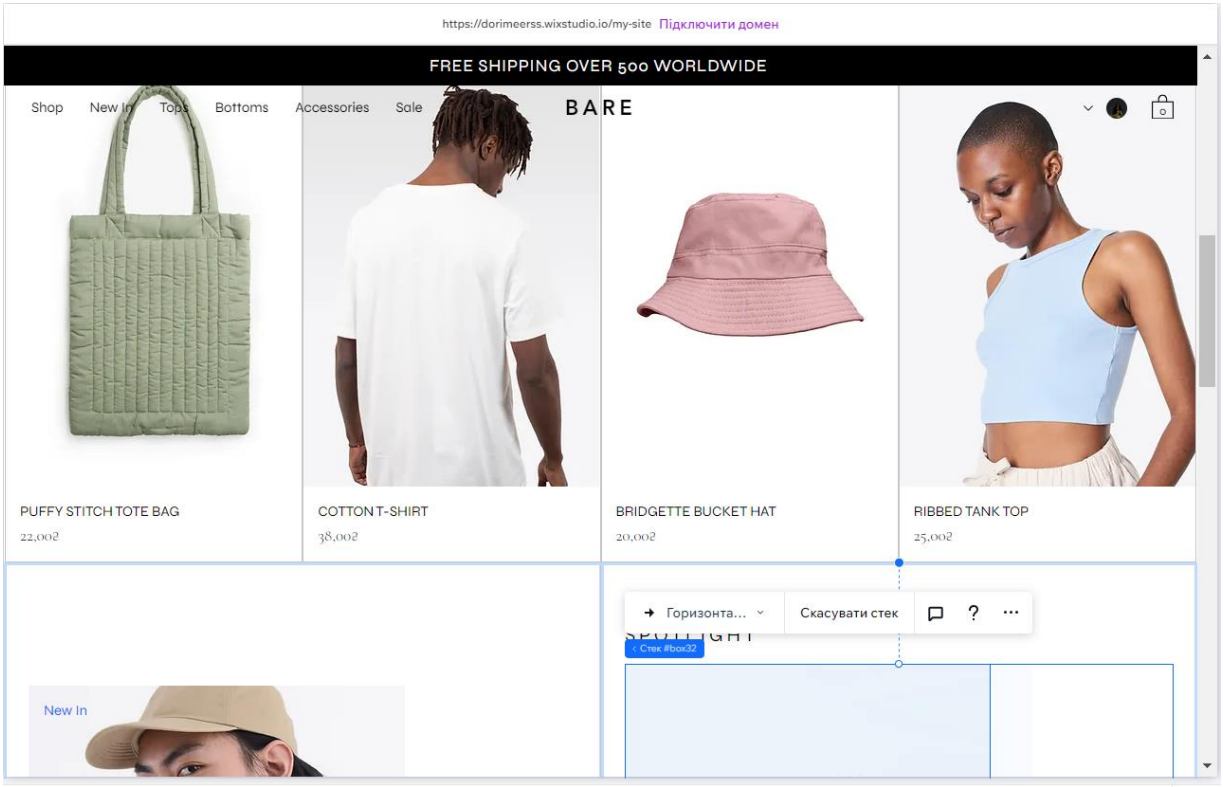


Рис. 2.15. Галерея товарів

Використовуючи стеки, можна контролювати поведінку блоків при зміні контенту, що в них знаходиться. Пізніше це можна буде використати для розстановки брейкпоїнтів, для адаптивності сайту під мобільні пристрої, як наведено на рисунку 2.16.

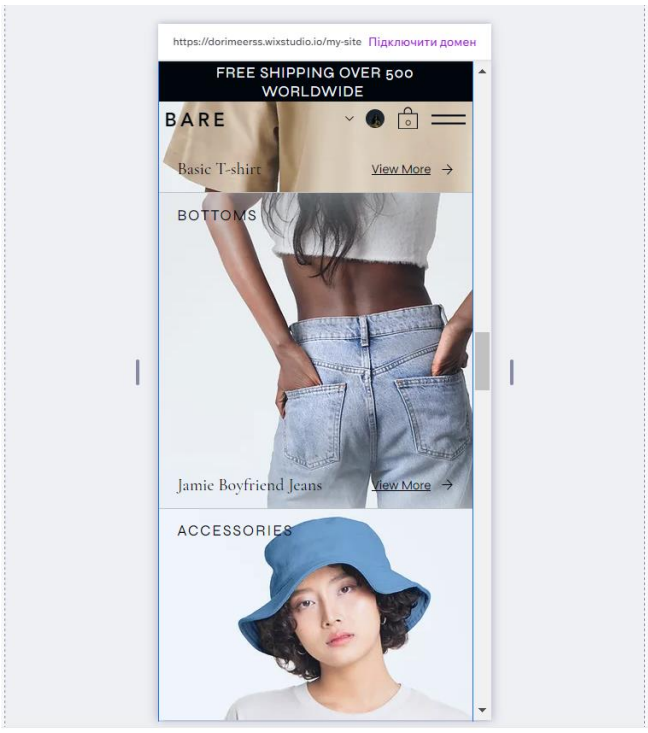


Рис. 2.16. Мобільна версія частини галереї

Для налаштування взаємодій між елементами сторінки, наприклад, переходів на інші сторінки по натисненню кнопки чи посилання

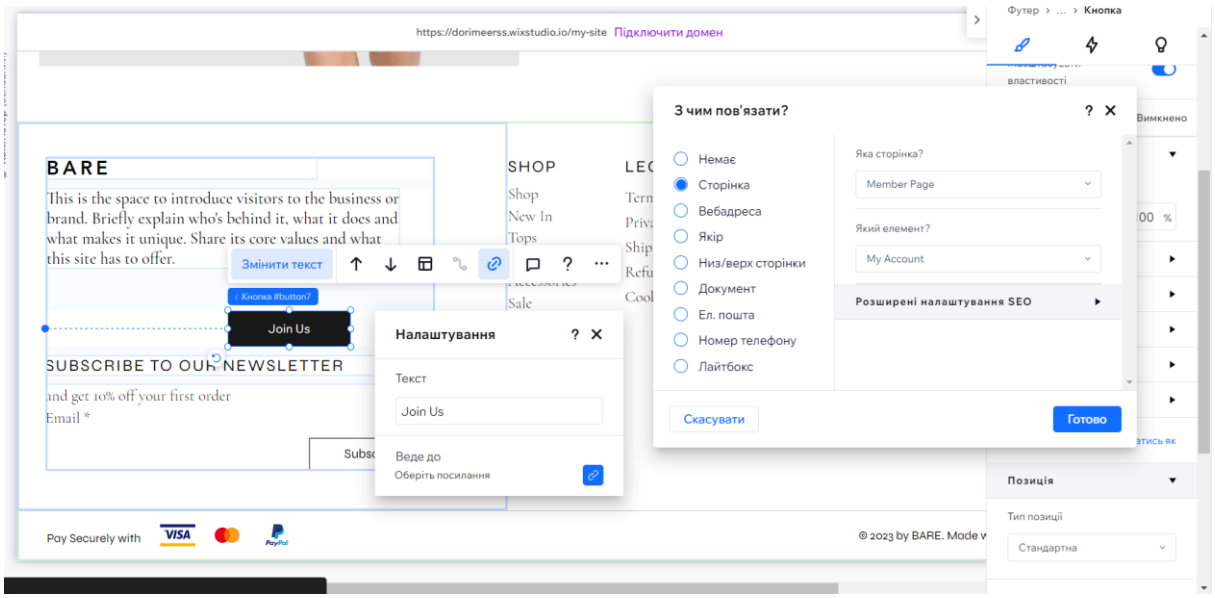


Рис. 2.17. Створення нової кнопки з ручним налаштуванням

Для категорії «Магазин» також доступні передналаштовані дії.

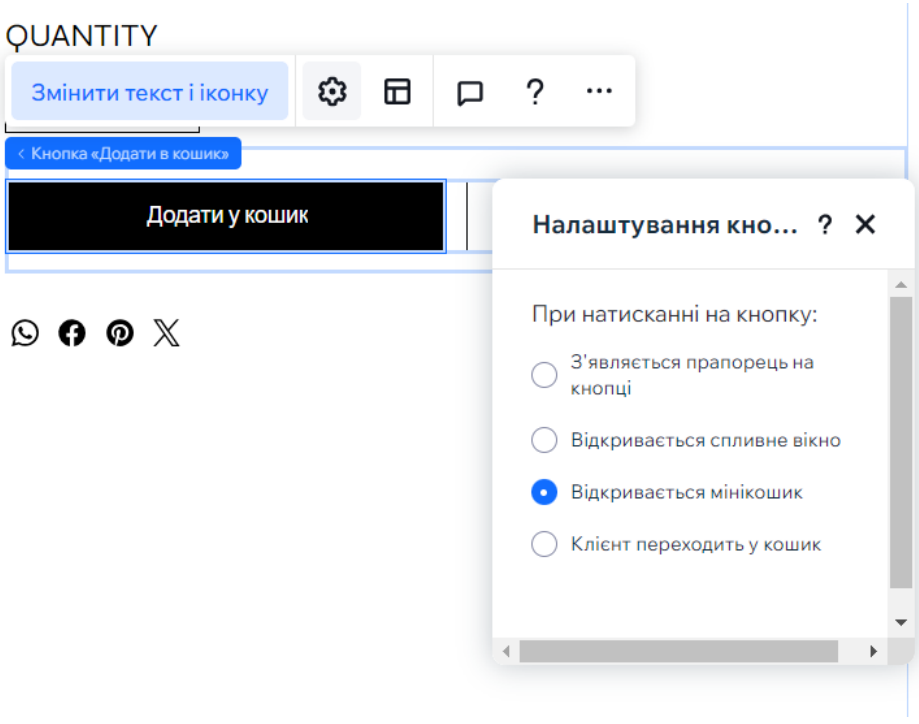


Рис. 2.18. Налаштування дії кнопки з готових наборів дій

Основними елементами сторінки з карткою товару є назва товару, ціна, можливі кольори, кількість та розмір, фотографії товару та кнопки «Купити» та «Додати в кошик». Результат моделювання даної сторінки можна побачити на рисунку 2.19.

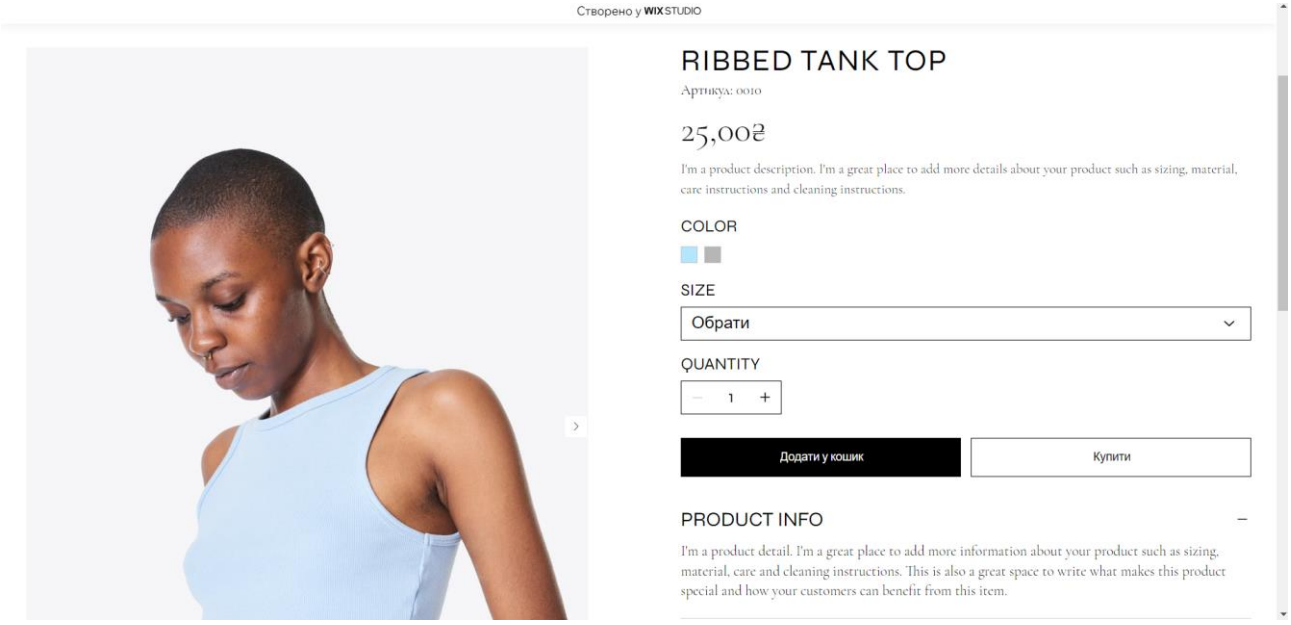


Рис. 2.19. Картка товару

Сторінка товарів включає в себе сітку з усіма товарами, наявними у магазині, відповідні категорії та фільтри за кольором, розміром. Фільтри працюють, опираючись на відповідну інформацію про товари, наявну в таблицях, які заповнюються менеджером через панель керування магазином.

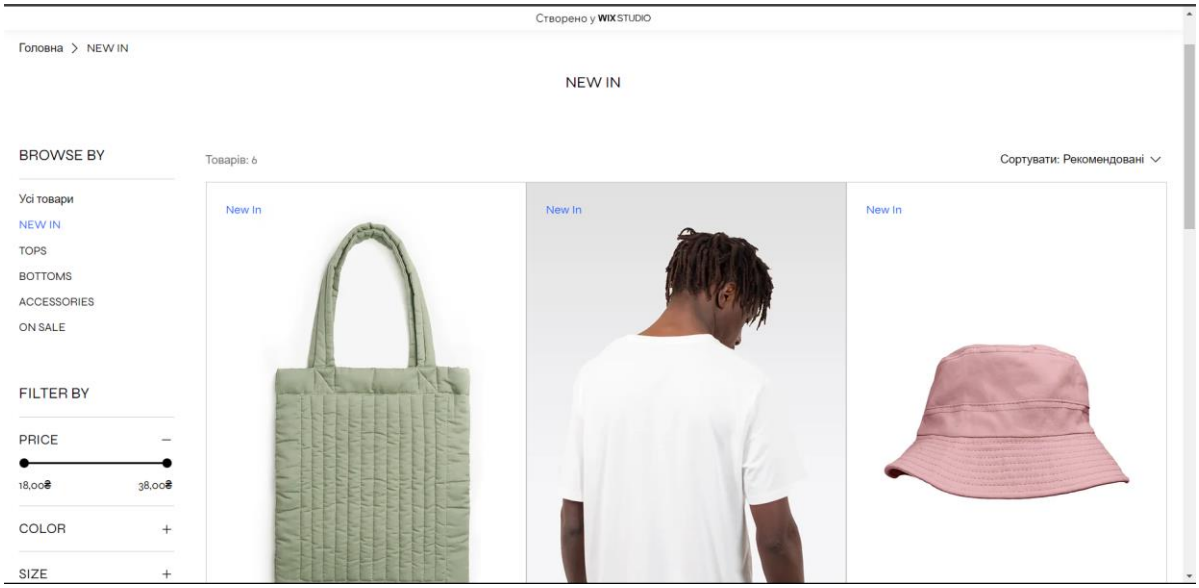


Рис. 2.20. Сторінка товарів, перегляд категорій

У сторінки «Кошик» є декілька різних станів, які включають в себе порожній та наповнений. В залежності від наявності товарів у кошику, сторінка матиме різний вигляд. Для порожньої сторінки кошику, наведеної на рисунку 2.21, окрім характерних для усіх сторінок хедера та футера бачимо лише текст та посилання в центі, щоб продовжити перегляд товарів.

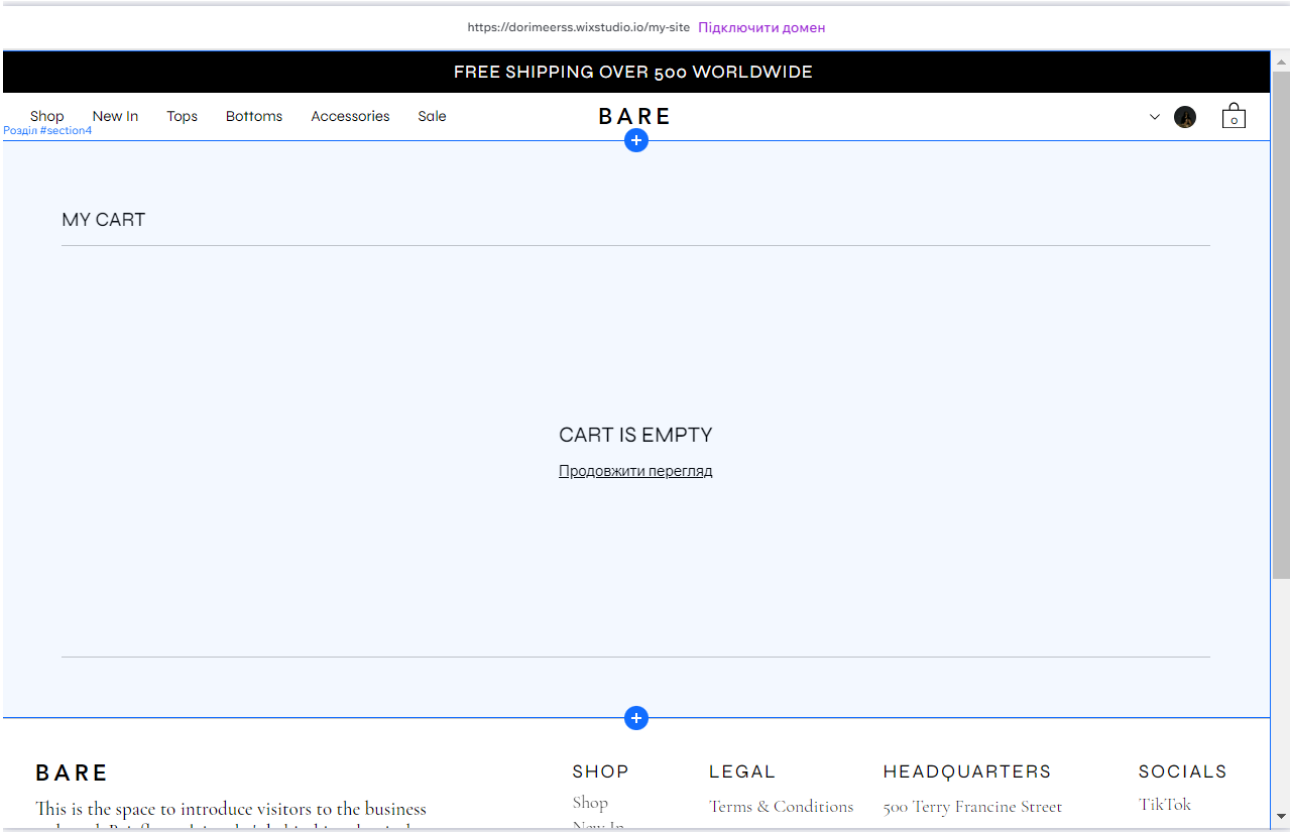


Рис. 2.21. Інтерфейс екрану порожнього кошику

Сторінка наповненого кошику має дещо іншу структуру, з'являється таблиця з даними замовлення, в кошику видно товар, його кількість та ціну, а також кнопки для введення промокоду та коментарів. Приклад такої сторінки наведено на рисунку 2.22

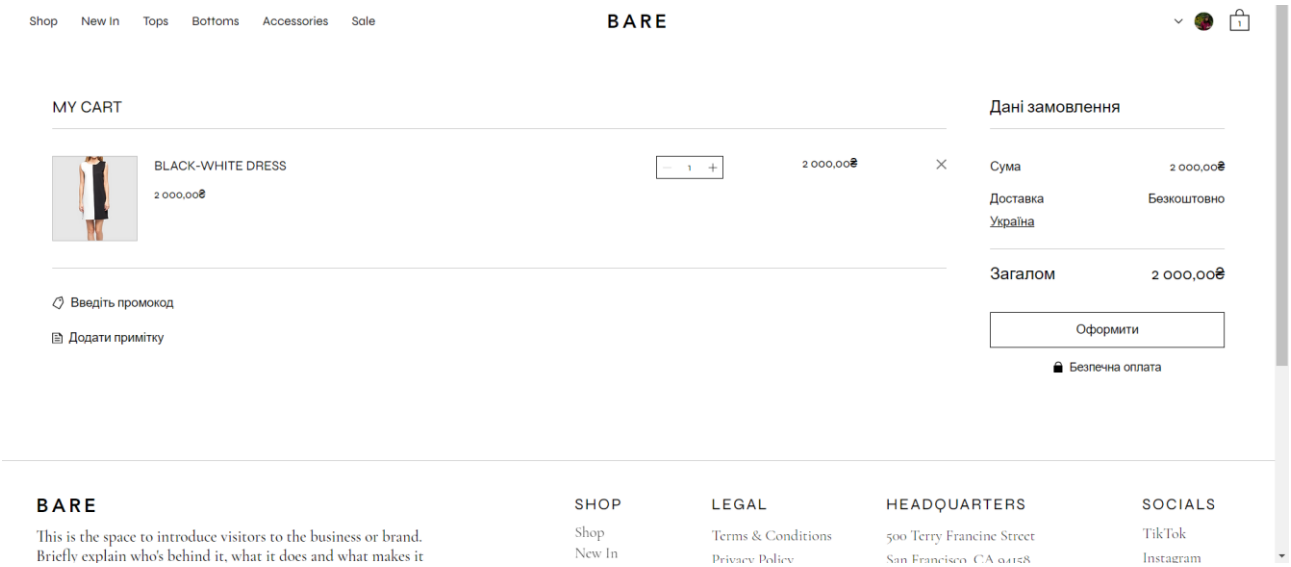


Рис. 2.22. Інтерфейс екрану кошику з товарами

Окрім цього, кошик може існувати у вигляді спливаючого вікна. Такий вигляд він має коли відчиняється на будь-якій іншій сторінці, без переходу безпосередньо на сторінку кошика.

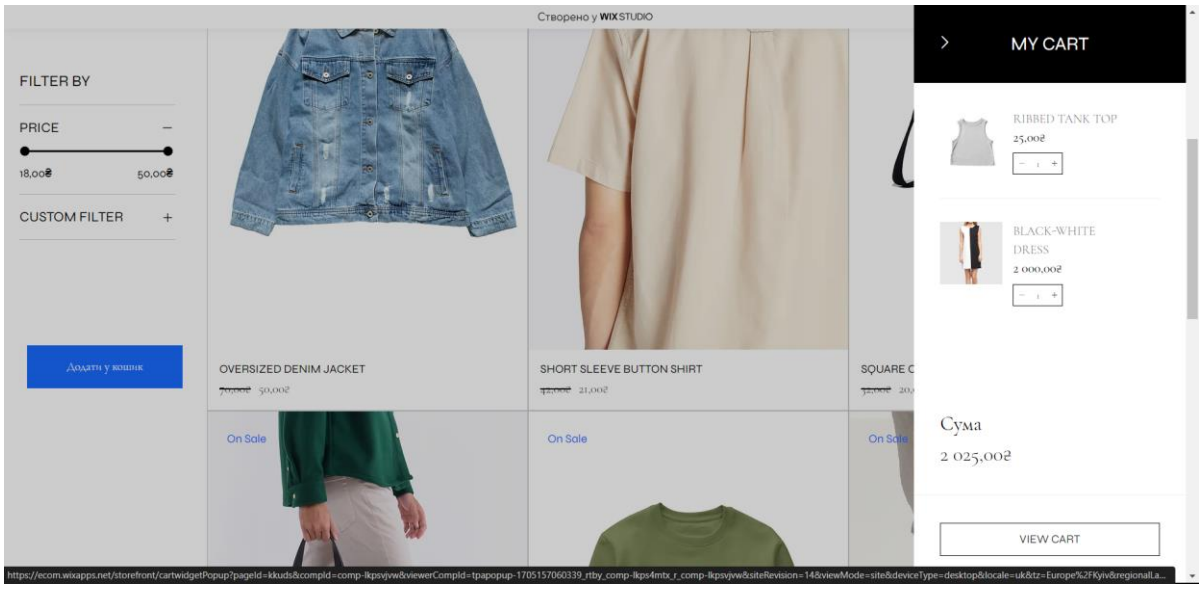


Рис. 2.23. Спливаюче вікно кошику

### 2.2.3 Інструменти адміністратора

В функціоналі платформи Wix наявні також інструменти для власника чи менеджера магазину, такі система керування магазином. Якщо у випадку з традиційним написанням коду довелось би проектувати даний розділ окремо, щоб забезпечити адміністратору можливість контролю клієнтів, замовлень, та товарообміну, тут платформа пропонує просте вбудоване рішення. Для налаштування бізнесу і початку активного використання сайту є простий покроковий список дій, доступний будь-якій людині без специфічних знань.

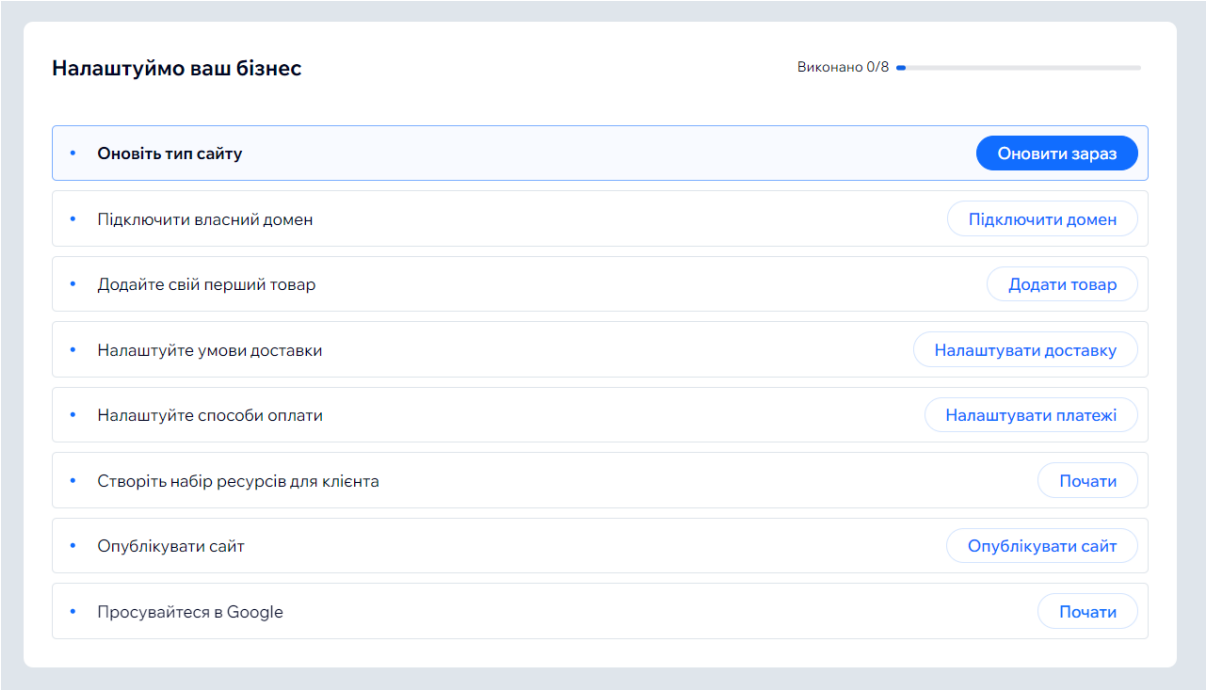


Рис. 2.24. Панель керування магазином

Для полегшення роботи з новоствореним сайтом також пропонується ряд додаткових рішень, направлених на ті чи інші види бізнесу. Приклад наведено на рисунку нижче.

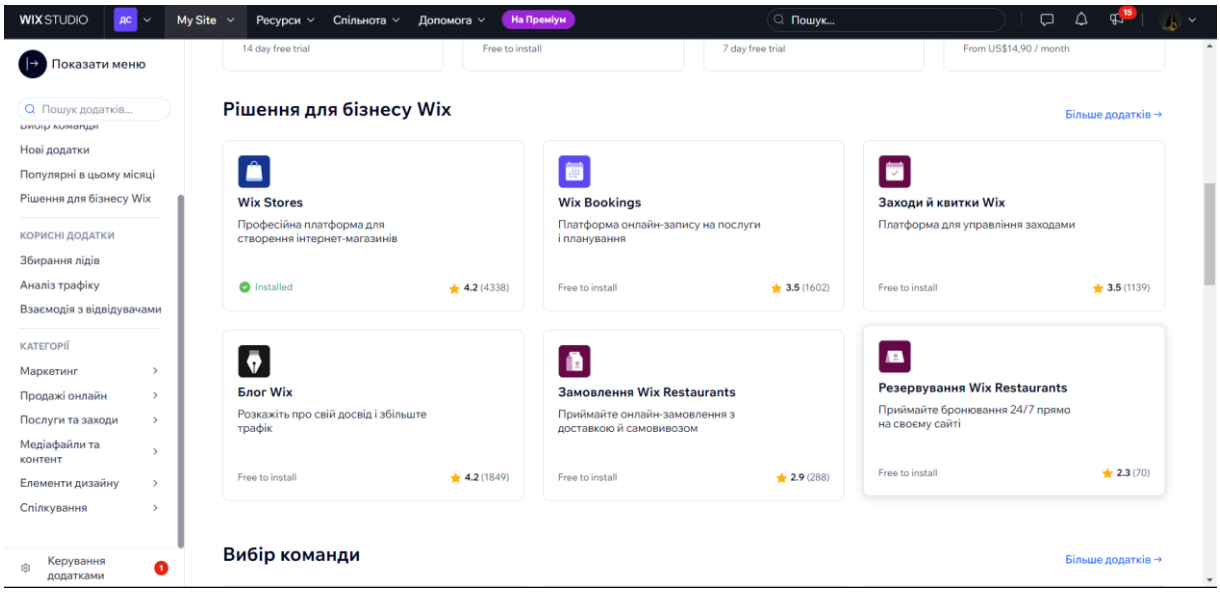


Рис. 2.25. Галерея доступних пресетів

Для управління товарами магазину, а також контролю запасів наявна окрема сторінка. Її інтерфейс показано на рисунку 2.26.

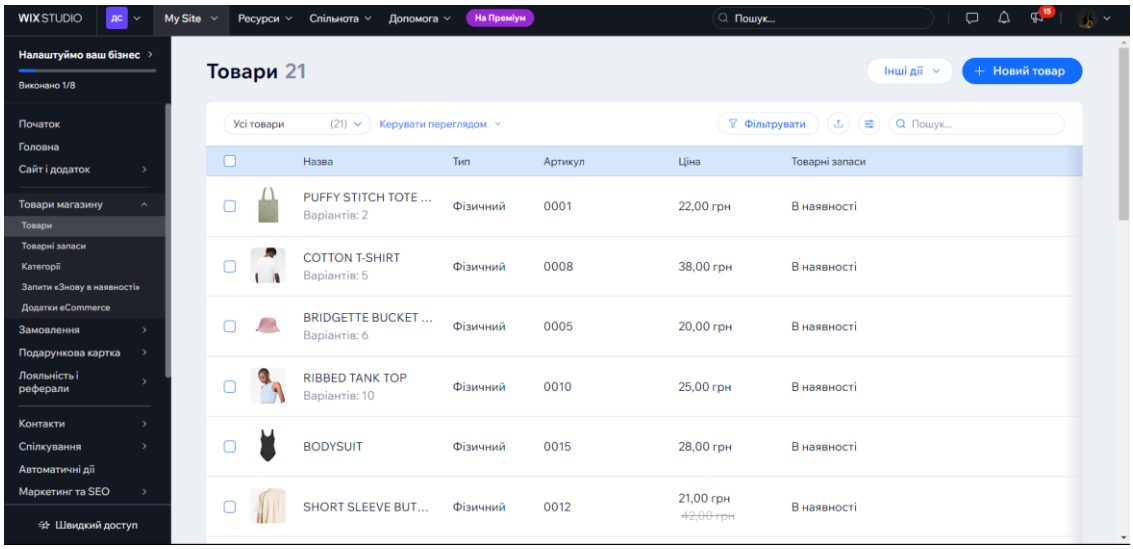


Рис. 2.26 Сторінка управління товарами

Для того щоб додати новий товар, необхідно натиснути кнопку «Новий товар», заповнити відповідні поля інформацією, та зберегти новий товар. Після цього він з’являється в списку та, якщо відмітіти прапорцем «Показувати в інтернет-магазині» стає доступним до замовлення на сайті.

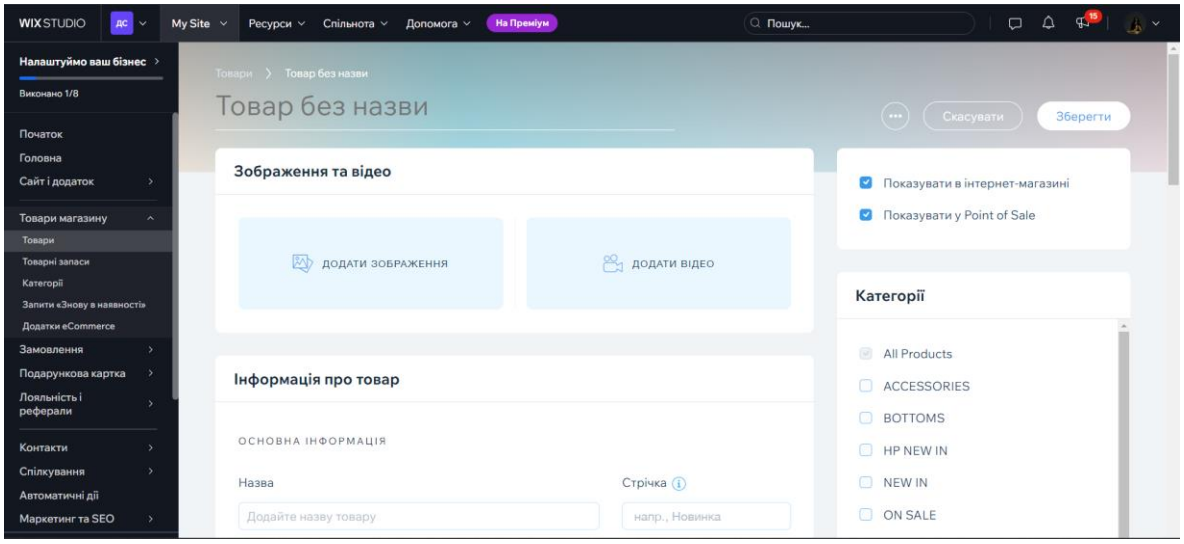


Рис. 2.27. Додавання нового товару

У разі необхідності внести зміни до інформації про товар, після наведення на поле товару, необхідно обрати символ трикрапки (показано на рисунку 2.28), і після цього «Редагувати товар».

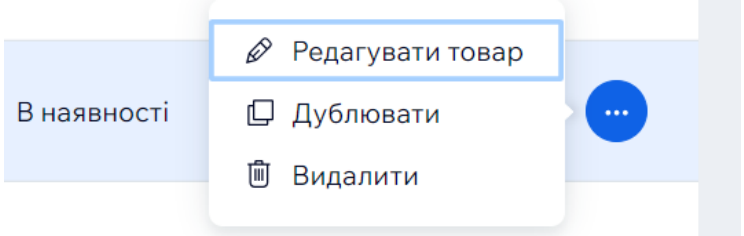

Рис. 2.28. Редагування інформації про товар

В доступних полях можна відредагувати інформацію та зберегти зміни після цього.

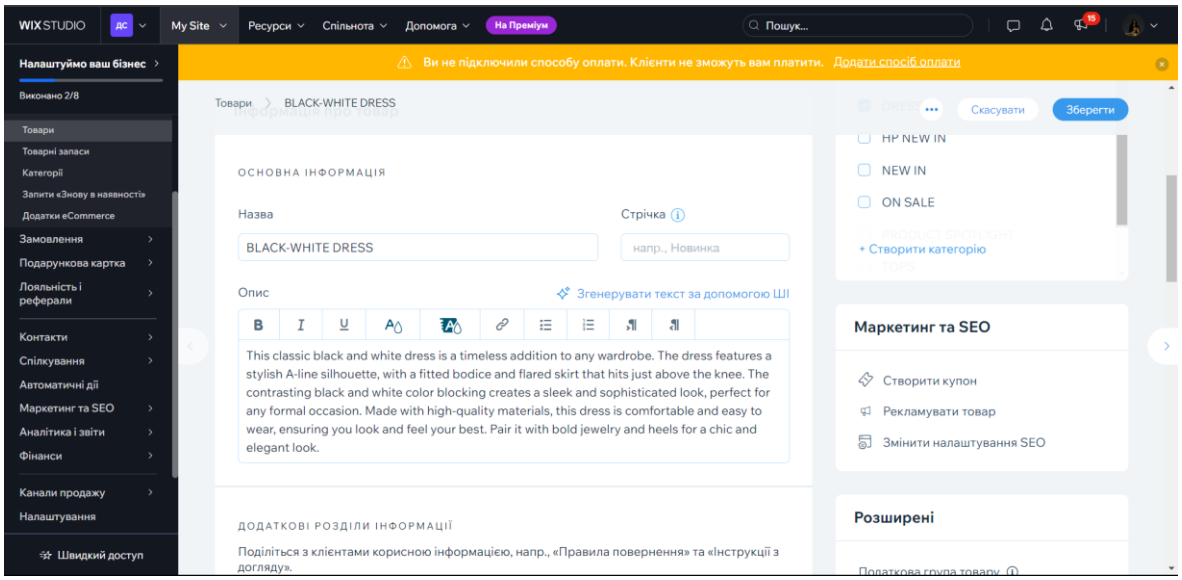


Рис. 2.29. Редагування інформації про товар

Для повноцінного функціонування магазину необхідно підключити системи оплати. В вкладці «Налаштування» обрати «Прийом платежів» і підключити довільні системи за бажанням.

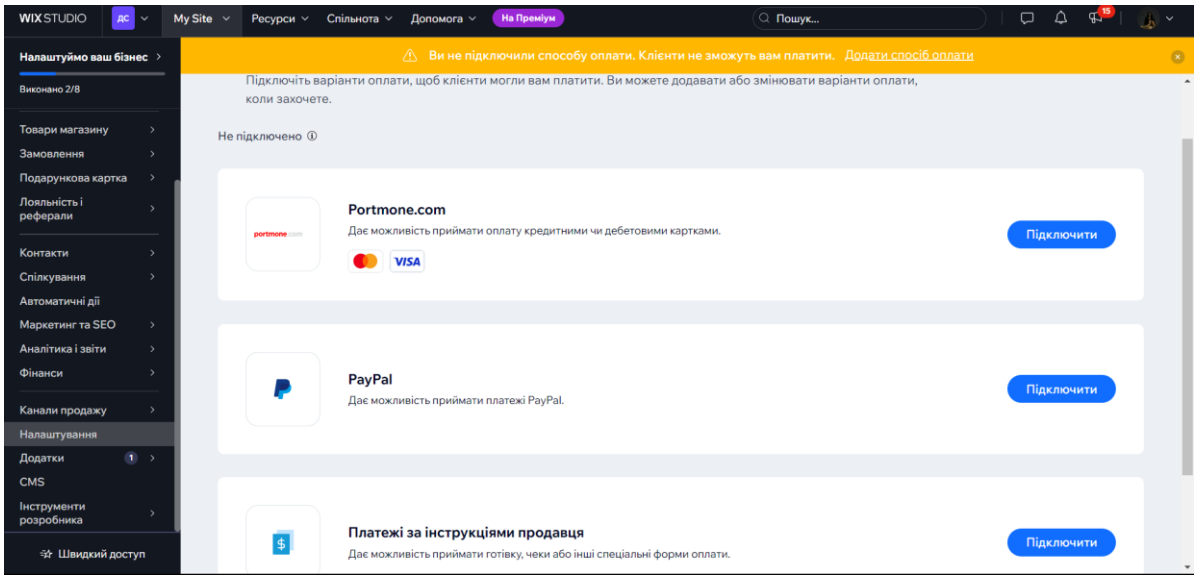


Рис. 2.30. Підключення систем оплати

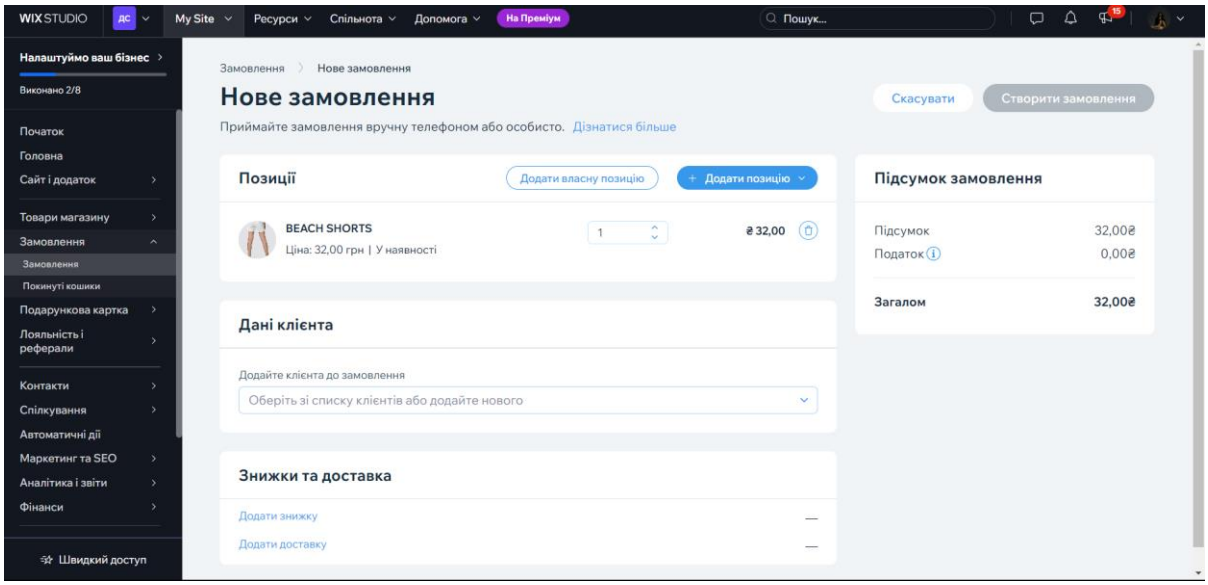


Рис. 2.31. Перегляд замовлення

Після закінчення роботи, готовий веб-сайт публікуємо. В якості хостингу виступає сам Wix, що спрощує також будь-які операції, пов’язані з розгортанням програмного забезпечення. За необхідності

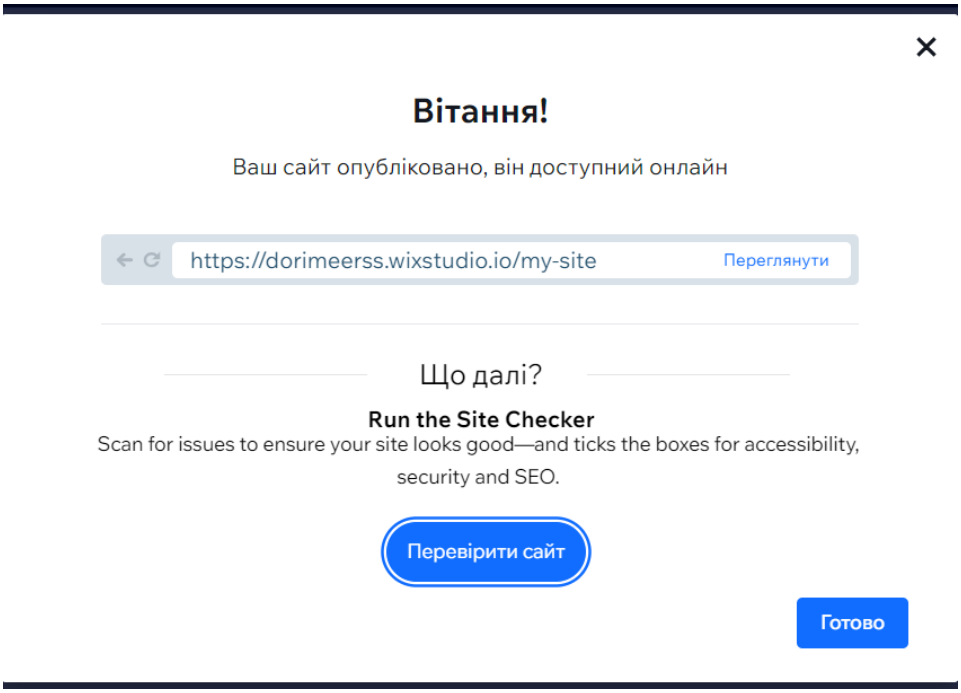


Рис. 2.32. Повідомлення про успішний запуск веб-сайту та посилання на перегляд

Переглянути готовий сайт можна за посиланням.

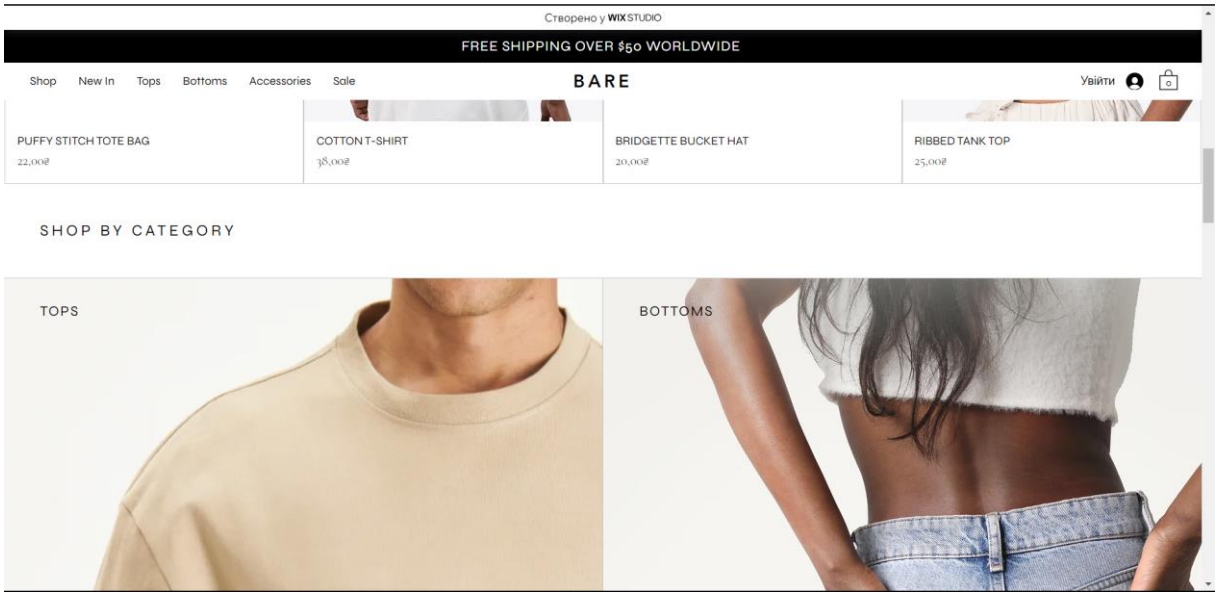


Рис. 2.33. Готовий веб-сайт

## Висновки до розділу 2

У цьому розділі оглянуто процес створення MVP сайту інтернет-магазину одягу з використанням традиційних методів розробки та no-code платформи. Сформовано функціональні та нефункціональні вимоги, визначено ролі користувачів в системі, а також описані варіанти використання.

Надано основну інформацію про ключові інструменти розробки, структуру та архітектуру проекту, надано діаграми модулів, з яких складається проект.

Також оглянуто процес конструювання веб-сайту на платформі Wix, доступні розширення та засоби проектування. Описано основні способи взаємодії з платформою, внутрішні можливості її використання для адміністратора. а також додаткові можливості для інтеграції та налаштування.

В результаті виконаної в даному розділі роботи, спроектовано дві версії вебсайту – програмна та платформена, підраховані часові витрати на обидва процеси. Платформенна версія спроектована за час близько 20 годин, в той же час, на програмну версію витрачено близько 50.

З цього випливає, що для такого типу задач використовувати no-code або low-code платформи вигідніше з точки зору часових витрат.

# РОЗДІЛ 3
# СТАТИСТИЧНІ ДОСЛІДЖЕННЯ

## 3.1 Формування вибірки

Для формування вибірки було проведено опитування групи респондентів. В ньому взяло участь 70 людей віком від 17 до 50 років, з різним досвідом у програмуванні та використанні платформ, щоб забезпечити більшу варіативність відповідей і репрезентацію кількох категорій людей, які і являються потенційною цільовою аудиторією таких платформ.

В якості завдання респондентам було запропоноване завдання, аналогічне до описаного в другому розділі даної роботи, а саме «Оцінити час, необхідний вам для створення MVP веб-сайту магазину одягу» без урахування будь-яких виправлень, комунікації з замовником та інше.

Опитування включало в себе 5 питань, результати та відповіді на них можна побачити нижче на рисунках 3.1 – 3.5.

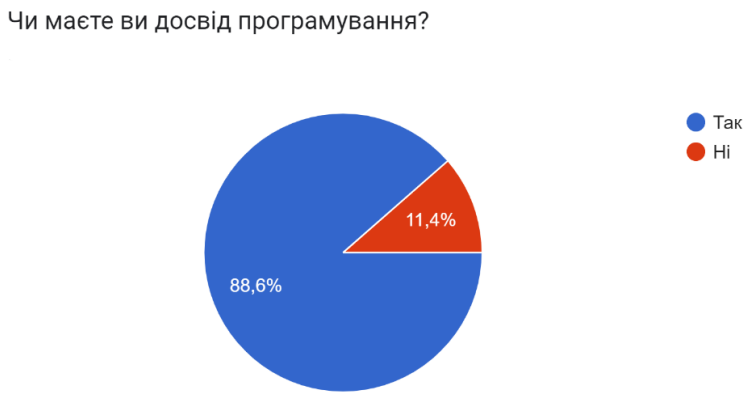


Рис. 3.1. Результати опитування 1

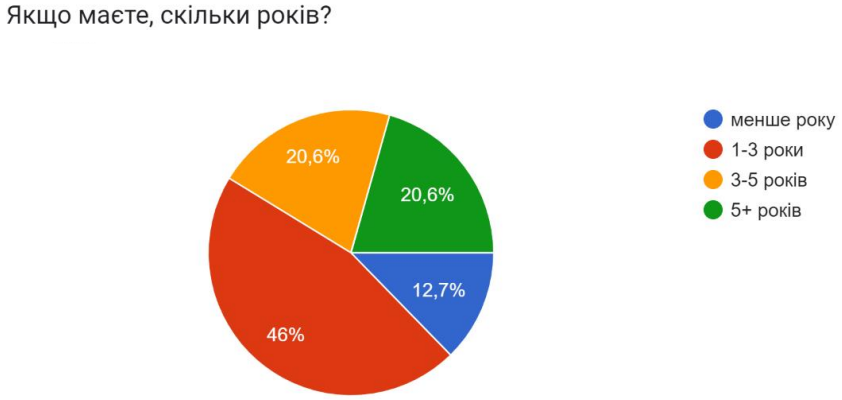


Рис. 3.2. Результати опитування 2

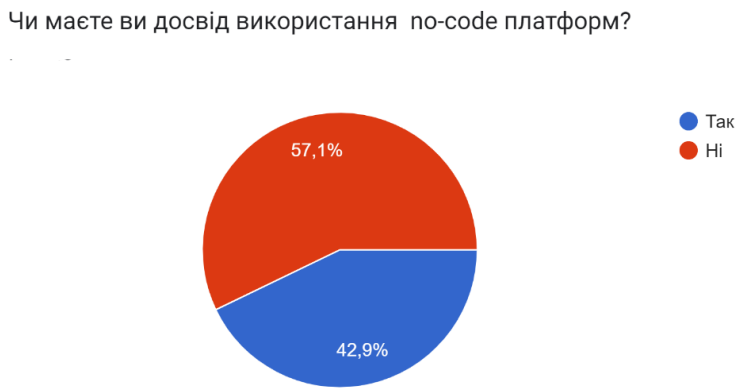


Рис. 3.3. Результати опитування 3

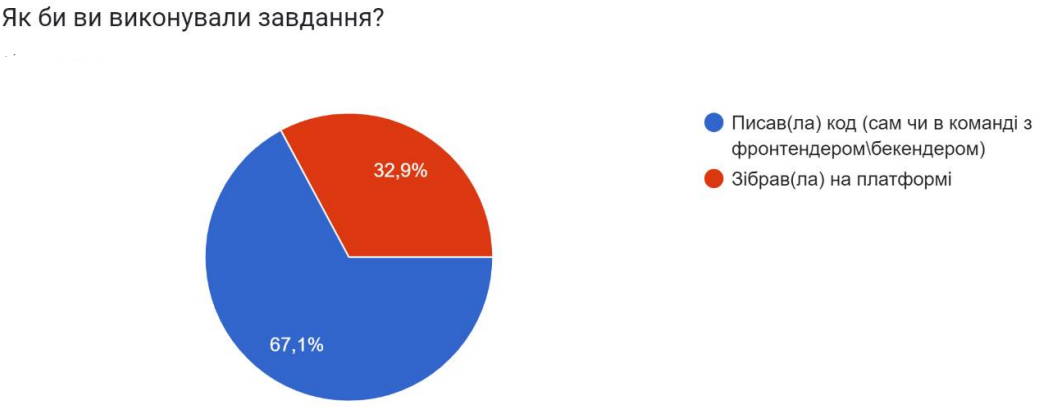


Рис. 3.4. Результати опитування 4


Рис. 3.5. Результати опитування 5

## 3.2 Обробка даних

Оскільки респонденти вказували необхідні часові проміжки різними способами, дану вибірку необхідно було модифікувати. Таким чином, була створена таблиця, відповіді респондентів структуровані та перетворені в один формат. Для цього створено таблицю в Excel та внесено в неї всю необхідну інформацію для подальшої роботи з розрахунками.

| B | C | D | E | F | G |
|---|---|---|---|---|---|
| Чи маєте ви досвід програмування? | Якщо маєте, скільки | Чи маєте ви досвід використання | Як би ви виконували завд | Скільки годин вам би знадобилось на виконання цього завдан | Адрес электронной п |
| Так | 5+ років | Ні | Писав(ла) код (сам чи в ко | 30 | |
| Так | 5+ років | Ні | Писав(ла) код (сам чи в ко | 80 | |
| Так | 1-3 роки | Ні | Писав(ла) код (сам чи в ко | 45 | |
| Так | 3-5 років | Ні | Писав(ла) код (сам чи в ко | 40 | |
| Ні | | Ні | Писав(ла) код (сам чи в ко | 124 | |
| Так | 3-5 років | Ні | Писав(ла) код (сам чи в ко | 40 | |
| Так | 1-3 роки | Так | Писав(ла) код (сам чи в ко | 10 | |
| Так | 1-3 роки | Ні | Писав(ла) код (сам чи в ко | 30 | |
| Так | 5+ років | Так | Писав(ла) код (сам чи в ко | 24 | |
| Так | 1-3 роки | Так | Писав(ла) код (сам чи в ко | 60 | |
| Так | 1-3 роки | Так | Писав(ла) код (сам чи в ко | 25 | |
| Ні | | Ні | Писав(ла) код (сам чи в ко | 32 | |
| Так | 3-5 років | Ні | Писав(ла) код (сам чи в ко | 160 | |
| Так | 1-3 роки | Ні | Писав(ла) код (сам чи в ко | 15 | |
| Так | 3-5 років | Ні | Писав(ла) код (сам чи в ко | 100 | |
| Так | 1-3 роки | Ні | Писав(ла) код (сам чи в ко | 40 | |
| Так | 1-3 роки | Ні | Писав(ла) код (сам чи в ко | 42 | |
| Так | 1-3 роки | Ні | Писав(ла) код (сам чи в ко | 7 | |
| Так | 5+ років | Ні | Писав(ла) код (сам чи в ко | 120 | |
| Так | 1-3 роки | Ні | Писав(ла) код (сам чи в ко | 40 | |
| Так | 1-3 роки | Ні | Писав(ла) код (сам чи в ко | 240 | |
| Так | 1-3 роки | Ні | Писав(ла) код (сам чи в ко | 8 | |
| Так | 1-3 роки | Ні | Писав(ла) код (сам чи в ко | 50 | |
| Ні | | Ні | Писав(ла) код (сам чи в ко | 42 | |
| Так | 1-3 роки | Ні | Писав(ла) код (сам чи в ко | 20 | |
| Так | 5+ років | Так | Писав(ла) код (сам чи в ко | 40 | |
| Так | 5+ років | Ні | Писав(ла) код (сам чи в ко | 40 | |
| Так | 5+ років | Так | Писав(ла) код (сам чи в ко | 4 | |
| Так | 1-3 роки | Ні | Писав(ла) код (сам чи в ко | 240 | |
| Так | менше року | Ні | Писав(ла) код (сам чи в ко | 20 | |
| Так | 1-3 роки | Так | Писав(ла) код (сам чи в ко | 120 | |
| Так | 3-5 років | Так | Писав(ла) код (сам чи в ко | 40 | |
| Так | менше року | Ні | Писав(ла) код (сам чи в ко | 0 | |
| Так | 1-3 роки | Так | Писав(ла) код (сам чи в ко | 12 | |
| Так | 1-3 роки | Так | Писав(ла) код (сам чи в ко | 30 | |
| Так | 3-5 років | Так | Писав(ла) код (сам чи в ко | 480 | |
| Так | 1-3 роки | Ні | Писав(ла) код (сам чи в ко | 150 | |
| Так | 3-5 років | Ні | Писав(ла) код (сам чи в ко | 30 | |

Рис. 3.6. Організована відповідним чином вибірка

Побудуємо графіки за результатами цієї таблиці, та графіки нормального розподілу до них.

Для стандартного відхилення даних, які є розподілом частот:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n} f_i (x_i - \mu)^2}{n}}$$

де:

- $f_i$ є частотою кожної точки даних,
- $x_i$ представляє кожну точку даних,
- $\mu$ є середнім значенням даних,
- n є загальною кількістю точок даних.

Середнє значення $\mu$ розраховується як:

$$\mu = \frac{\sum_{i=1}^{n} f_i x_i}{n}$$

де:

- $f_i$ є частотою кожної точки даних,
- $x_i$ представляє кожну точку даних,
- n є загальною кількістю точок даних.[16]

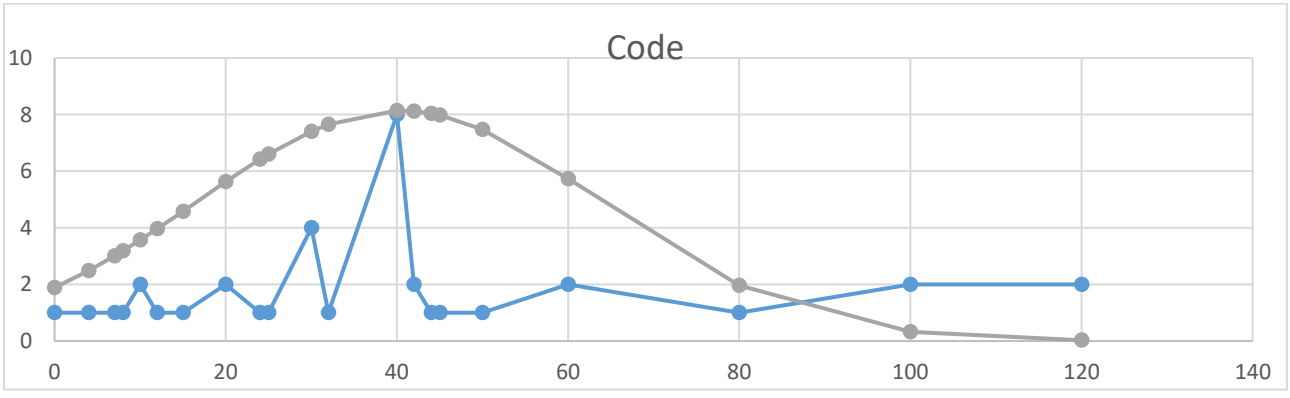


Рис. 3.7. Графік оцінки часових витрат для програмування

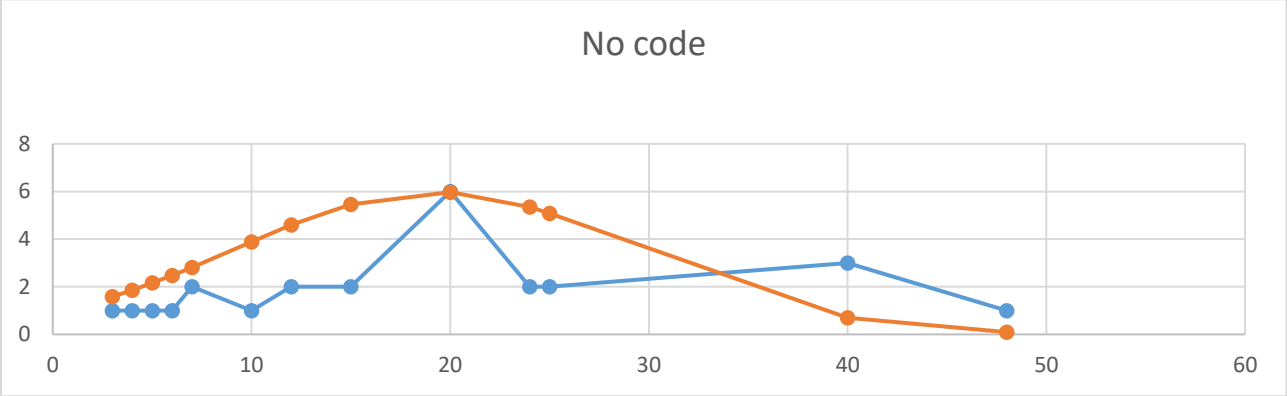


Рис. 3.8. Графік оцінки часових витрат для no-code платформ

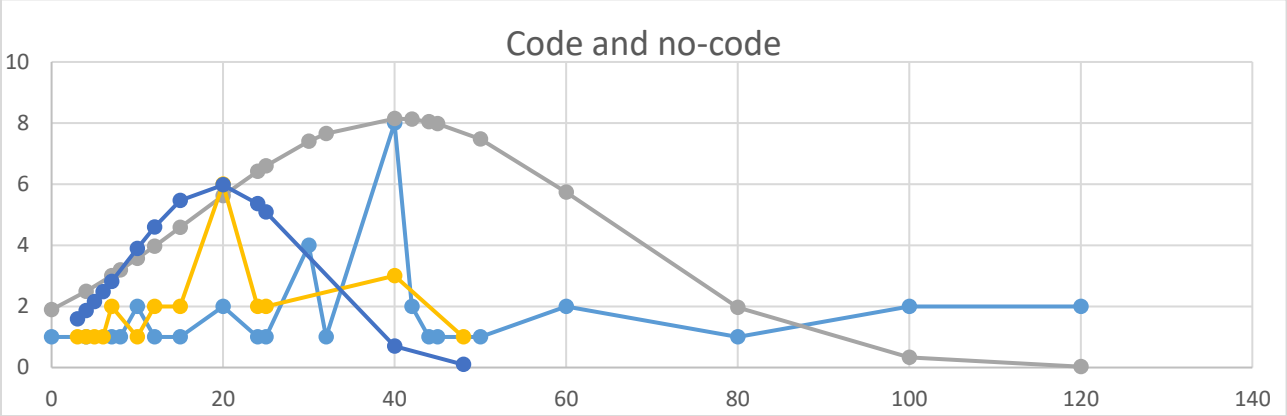


Рис. 3.9. Загальний графік

З наведених рисунків можна побачити, що ці графіки близькі до натурального розподілу, з тим, що no-code рішення займає менше часу, тому для малих сайтів і прототипів no-code рішення є значно ефективнішими.

Медіана часу для проектування такого сайту з використанням no-code рішення буде дорівнювати 19.28, для програмування медіана матиме значення 40.27. Таким чином, можемо зробити висновок, що різниця витрат у часі майже в два рази говорить про високу ефективність no-code платформ для більшості задач подібного типу.

Специфіка та порівняно невеликий розмір вибірки може частково впливати на кінцевий результат, але близькість до нормального розподілу загалом підтверджує правдивість результатів.

## Висновки до розділу 3

У цьому розділі описано результати проведеного опитування стосовно використання програмування або low-code та no-code платформ для виконання простої задачі на прикладі MVP інтернет-магазину одягу.

Описані основні питання, надана статистика відповідей, інформація, отримана в результаті проаналізована та узагльнена для подальшої роботи з нею.

На основі отриманих даних побудовано графіки часових витрат на проектування веб-сайту двома способами за означеною задачею, а також нормальні розподіли для цих відповідних значень.

# РОЗДІЛ 4
# РОЗРОБКА СТАРТАП-ПРОЄКТУ

## 4.1 Інформаційна карта проєкту

*Таблиця 4.1*

**Інформаційна картка проєкту**

| | |
|---|---|
| *1. Назва проєкту* | No-code платформа “Droplet” |
| *2. Автори проєкту* | Волокита Артем Миколайович, Смучок Дарина Сергіївна |
| *3. Коротка анотація* | Метою даного стартапу є розробка no-code платформи. Вона пропонує інтуїтивний драг-енд-дроп інтерфейс, широкий набір вбудованих шаблонів та компонентів, а також можливість інтеграції з третіми сервісами. З її допомогою користувачі можуть швидко і ефективно розробляти мобільні та веб-додатки, автоматизувати робочі процеси та оптимізувати бізнес-задачі без глибоких знань в програмуванні. Використання шаблонів допоможе зекономити час на продумуванні загальної структури для окремих осіб або малих підприємців, які вперше створюють сайт для свого бізнесу, а також вивільнити ресурси програмістів у більших компаніях, дозволивши ним займатись реалізацією нового функціоналу. |
| *4. Термін реалізації проєкту* | 4 місяці |

*Таблиця 4.1 (продовження)*

**Інформаційна картка проєкту**

| | |
|---|---|
| *5. Необхідні ресурси* | *Фінансові ресурси* |
| | 1. Витрати на співробітників на місяць:<br>− 2 back-end розробника – 60 000 грн х 2 = 120 000 грн<br>− 1front-end розробник – 60 000 грн<br>− 1 тестувальник – 40 000 грн<br>− 1 UI\UX дизайнер – 50 000 грн<br>− 1 проєктний менеджер – 80 000 грн<br>− 1 маркетолог – 30 000 грн<br>2. Оренда та обслуговування приміщення (на місяць) – 0 грн<br>(команда працюватиме дистанційно)<br>3. Оформлення торгової марки – 11 105 грн<br>4. Оформлення авторського плану – 10 000 грн<br>5. Оренда домену (на рік) – 1 013,11 грн<br>6. Оренда хмарних сервісів Azure (на рік) – 28986,89 грн<br>*Всього:* 1 601 105 грн. |
| | *Матеріальні ресурси* |
| | 1. Ноутбук Acer Aspire 5 Spin 14– 34 999 грн (7 шт.)<br>2. Комп'ютерна миша Миша Logitech M100 USB Black – 289 грн (7 штук)<br>3. Гарнітура Gembird MHS-123 Black – 140 грн (7 штук)<br>4. Операційна система Windows – 1649 грн (7 ліцензій)<br>Всього: 259 539 грн. |

*Таблиця 4.1 (продовження)*

**Інформаційна картка проєкту**

| | |
|---|---|
| | Інтелектуальні ресурси |
| | Навчально-методичне забезпечення, онлайн-документація, наукові статті, тощо – безкоштовно. |
| | Загальна сума: 1 860 644грн. |
| *6. Опис проблеми, яку вирішує проєкт* | No-code платформа “Droplet” дозволить знизити вимоги до рівня технічних знань, якими має володіти користувач для створення власних програм. Конструювання веб-сайтів на абстрактному та візуальному рівні дозволить не вдаватися в деталі реалізації, а фокусуватись на потребах бізнесу. |
| *7. Головні цілі та завдання проєкту* | Головною ціллю стартап-проєкту є створення MVP no-code платформи<br>Завдання проєкту:<br>1. Створення базового функціоналу платформи: візуальний редактор, робота з базами даних, респонсивний дизайн, шаблони, аутентифікація користувачів<br>2. Інтеграція з зовнішніми сервісами та соцмережами, системи управління контентом, аналітика<br>3. Створення сучасного і зручного інтерфейсу<br>4. Здобуття 1000 активних користувачів системи |
| *8. Очікувані результати* | Користувачі зекономлять багато часу на навчанні користування інструментами для розробки, та зможуть протестувати свої ідеї і спробувати себе в ролі розробника без великого обсягу технічних знань, що понизить поріг входу у розробку, вивільнивши спеціалістів для більш серйозних задач. |

## 4.2 Формування команди стартапу

*Таблиця 4.2*

**Склад команди стартапу**

| *№ п/п* | *Роль* | *Зона відповідальності* | *Кількість* |
|---|---|---|---|
| 1 | Back-end розробник | Розробка серверного коду | 2 людини |
| 2 | Front-end розробник | Розробка клієнтського коду | 1 людина |
| 3 | Тестувальник | Перевірка зручність інтерфейсу користувача та відповідність програмного забезпечення вимогам до нього | 1 людина |
| 4 | UI\UX дизайнер | Розробка інтерфейсу користувача, користувацього досвіду і дизайну | 1 людина |
| 5 | Проєктний менеджер | Управління командою, генерування ідей; несе відповідальність за проєкт | 1 людина |
| 6 | Маркетолог | Просування продукту на ринок | 1 людина |
| | | Разом: | 7 людей |

Отже, стартап складатиметься з 7 людей, але фактично над розробкою продукту працюватимуть шестеро, оскільки маркетолог не є технічним працівником. Хоча його внесок в продукт і не безпосередній, але також важливий для життєздатності стартапу.

Визначимо важливість факторів реалізації стартап-проєкту.

*Таблиця 4.3*

**Визначення важливості факторів щодо їх вкладу у створення та реалізацію інноваційного проєкту**

| *№ п/п* | *Фактор* | *Важливість фактору (1- 10 балів)* |
|---|---|---|
| 1 | Створення та просування ідеї | 7 |
| 2 | Формування бізнес-плану | 5 |

*Таблиця 4.3 (продовження)*

**Визначення важливості факторів щодо їх вкладу у створення та реалізацію інноваційного проєкту**

| | | |
|---|---|---|
| 3 | Компетентність співробітників | 7 |
| 4 | Співпраця та комунікація | 7 |
| 5 | Виконання обов’язків | 9 |

Проаналізуємо важливість факторів для кожного зі співробітників та побудуємо діаграму особистого внеску співробітників у проєкт.

*Таблиця 4.4*

**Оцінювання особистого внеску кожного партнера у створення та реалізацію стартапу**

| *Фактор* | *Back-end розробник* | *Front-end розробник* | *Тестувальник* | *UI\UX дизайнер* | *Проєктний менеджер* | *Маркетолог* | *Разом* |
|---|---|---|---|---|---|---|---|
| Створення та просування ідеї | 30 | 40 | 10 | 40 | 80 | 5 | |
| Формування бізнес-плану | 20 | 20 | 20 | 20 | 80 | 10 | |
| Компетентність співробітників | 90 | 90 | 70 | 90 | 80 | 50 | |
| Співпраця та комунікація | 75 | 75 | 60 | 75 | 100 | 20 | |
| Виконання обов’язків | 90 | 90 | 40 | 90 | 80 | 30 | |
| Разом | 305 | 315 | 200 | 315 | 420 | 115 | 1670 |
| У відсотках | 18,3 | 18,8 | 12 | 18,8 | 25,2 | 6,9 | 100 |

Як можна бачити, найбільше внеску у проєкт вносять проєктний менеджер, back-end та front-end розробники і UI\UX дизайнер. Однак, применшувати вплив тестувальника та маркетолога не варто, оскільки без них стартапу буде важче реалізуватися на ринку.

## 4.3 Формулювання основної ідеї проєкту

У 1-ому розділі дисертації ми дійшли висновку, що на ринку присутні багато додатків-аналогів, що використовуються для створення сайтів. Але попри свою популярність і багатство функцій, вони мають мінуси, деякі з яких можна усунути у нашому стартап-проєкті. Завдяки цьому, наш продукт може бути конкурентоспроможним на ринку.

Серед переваг можна відмітити також той факт, що велика частина функціоналу нашого додатку буде повторювати існуючі в аналогах. А це означає, що ми уже знаємо, які функції і як вони повинні виглядати, тому для нас це зробити буде простіше і дешевше, ніж для додатків-аналогів. В той же час, так як наповнення програми відповідними можливостями як і в конкурентів, та імплементація нового функціоналу потребує значних часових витрат навіть з урахуванням розуміння необхідного функціоналу.

Отже, підсумовуючи усі плюси і мінуси, складемо таблицю.

*Таблиця 4.5*

**Плюси та мінуси стартап-проєкту**

| *№ п/п* | *Плюси* | *Мінуси* |
|---|---|---|
| 1 | Імплементація інтерфейсів методом drag-and-drop | Складність розробки візуального моделювання проєкту |
| 2 | Спрощення системи для користувача | Складність проєктування продукту |
| 3 | Копіювання в продукт уже існуючих рішень з аналогів | Менший набір можливостей застосунку |
| 4 | Шаблони і уніфікація | Невисока конкурентоспроможність на старті |

*Таблиця 4.5 (продовження)*

**Плюси та мінуси стартап-проєкту**

| | | |
|---|---|---|
| 5 | Повторне використання існуючих рішень | Складнощі з нарощуванням клієнтської бази |
| 6 | Швидкість розгортання | |
| 7 | Адаптивність | |
| 8 | Розширюваність | |
| *Сума* | 8 | 5 |

Для успішної реалізації ідеї необхідно визначити всі сильні, нейтральні та слабкі техніко-економічні характеристики і пропрацювати слабкі, якщо вони мають вагомий вплив на фінальний результат. Наведемо результати аналізу у наступній таблиці.

*Таблиця 4.6*

**Техніко-економічні характеристики стартапу та продуктів аналогів**

| *№ п/п* | *Характеристика* | *Власна розробка* | *Аналоги* | | | | *W* | *N* | *S* |
|---|---|---|---|---|---|---|---|---|---|
| | | | *1* | *2* | *3* | *4* | | | |
| 1 | Легкий і інтуїтивно зрозумілий інтерфейс | + | + | | + | + | | | + |
| 2 | Drag-and-drop моделювання | + | + | + | + | + | | | + |
| 3 | Доступ до додатку з будь-якого пристрою | | + | | + | + | | + | |
| 4 | Адміністрування доступу | | + | | + | + | + | | |
| 5 | Робота з базами даних | + | + | + | + | + | | + | |
| 6 | Інтеграція з зовнішніми додатками | | + | | + | + | | + | |
| 7 | Спрощена система для користувача | + | + | | | + | | + | |
| 8 | Шаблони | + | + | + | + | + | | | + |

*Таблиця 4.6 (продовження)*

**Техніко-економічні характеристики стартапу та продуктів аналогів**

| *№ п/п* | *Характеристика* | *Власна розробка* | *Аналоги* | | | | *W* | *N* | *S* |
|---|---|---|---|---|---|---|---|---|---|
| | | | *1* | *2* | *3* | *4* | | | |
| 9 | Уніфікація | + | + | + | | + | | + | |
| 10 | Повторне використання | + | + | | + | | | | + |
| 11 | Швидкість розгортання | + | + | | + | + | | | + |
| 12 | Адаптивність | + | + | + | + | | | | + |
| 13 | Розширюваність | + | + | + | + | + | | + | |

Дослідивши технічні та економічні аспекти, ми можемо закласти основу для забезпечення конкурентних переваг нашого стартапу. З'ясувалося, що наша продукція має чимало переваг у власній категорії (включаючи ті, що вже присутні у конкурентів, та ексклюзивні), тоді як характеристики, які ми не впроваджуємо, переважно не є критично важливими або є невизначеними.

## 4.4 Аналіз ринкових можливостей

*Таблиця 4.7*

**Попередня характеристика потенційного ринку стартап проєкту**

| *№ п/п* | *Показники стану ринку (найменування)* | *Характеристика* |
|---|---|---|
| 1 | Кількість головних гравців, од | 6 |
| 2 | Загальний обсяг продаж від реклами, грн/ум. од. | 100грн*1000 реклам = 100000 грн (в місяць) |
| 3 | Динаміка ринку (якісна оцінка) | Зростає |
| 4 | Наявність обмежень для входу (вказати характер обмежень) | Відсутні |
| 5 | Специфічні вимоги до стандартизації та сертифікації | Відсутні |
| 6 | Середня норма рентабельності в галузі (або по ринку), % | (689000/1000000)*100 = 68.9% за рік |

*Таблиця 4.8*

**Характеристика потенційних клієнтів стартап-проєкту**

| *№ п/п* | *Потреба, що формує ринок* | *Цільова аудиторія (цільові сегменти ринку)* | *Відмінності у поведінці різних цільових груп клієнтів* | *Вимоги до споживачів товару* |
|---|---|---|---|---|
| 1 | Спрощення процесу створення ПЗ | Компанії, що займаються розробкою ПЗ, індивідуальні підприємці, користувачі з низьким рівнем технічних навичок | Відсутні | Широкий функціонал, зручний інтерфейс, широкі можливості для створення ПЗ |
| 2 | Використання простих для розуміння інструментів | Компанії, що хочуть вивільнити ресурс програмістів, користувачі з низьким рівнем технічних навичок | Відсутні | Можливість не заглиблюватись в технічні деталі реалізації |
| 3 | Точність планування та розподілу бюджету | Фактично, усі потенційні користувачі | Особливо актуально для користувачів, у яких кошти обмежені | Враховування досвіду співробітників, прив’язка оплати до співробітника, а не тільки до задач |

*Таблиця 4.9*

**Фактори загроз**

| *№ п/п* | *Фактор* | *Зміст загрози* | *Можлива реакція компанії* |
|---|---|---|---|
| 1 | Поява конкурентів із передовими технологіями | Користувачі перейдуть до компанії, що запропонує їм кращі умови | Регулярне оновлення та удосконалення нашого продукту |
| 2 | Неефективна реклама | Лише невелика частина цільової аудиторії дізнається про продукт | Глибше вивчення аудиторії та коригування рекламних стратегій |
| 3 | Обмеженість функціоналу | Недостача необхідних функцій може відштовхнути користувачів до альтернативних продуктів. | Ретельний аналіз потреб користувачів і прискорення розробки ключових особливостей і необхідного функціоналу. |
| 4 | Неякісний продукт | Недостатня якість продукту змусить користувачів звертатись до конкурентів | Зосередження на якості, посилення тестування та підвищення кваліфікації команди. |

*Таблиця 4.10*

**Фактори можливостей**

| *№ п/п* | *Фактор* | *Зміст можливості* | *Можлива реакція компанії* |
|---|---|---|---|
| 1 | Постійний розвиток No-code методів розробки | Продукт забезпечує кращі можливості ніж у поточних і нових конкурентів | Подальший розвиток функцій, введення нових можливостей |
| 2 | Використання можливостей запропонованих конкурентами | Додавання можливостей з конкурентних продуктів для зменшення відмінностей у функціоналі | Зосередження зусиль на створенні унікальних особливостей замість загальноприйнятих |
| 3 | Освоєння ринків, де конкуренція слабка | Продукт отримує популярність серед користувачів деяких ринків | Сфокусування на опрацюванні цього ринку та схожих з ним ринків. |

*Таблиця 4.11*

**Ступеневий аналіз конкуренції на ринку**

| *Особливості конкурентного середовища* | *В чому проявляється дана характеристика* | *Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентноспроможною)* |
|---|---|---|
| Монополістичне | Багато продавців пропонують аналогічний товар і конкурують за покупців | Необхідність постійно зростати в продажах через простий доступ до ринку |
| Глобальне | Конкуренти представлені у різних країнах та мають клієнтів з усього світу | Потреба в адаптації додатка до різних мов |

*Таблиця 4.11 (продовження)*

**Ступеневий аналіз конкуренції на ринку**

| *Особливості конкурентного середовища* | *В чому проявляється дана характеристика* | *Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентноспроможною)* |
|---|---|---|
| Внутрішньогалузеве | Основні конкуренти зосереджені в сфері IT, маркетингу та на загальному рівні | Необхідність розробки особливостей, характерних для цих галузей |
| Товарно-родове | Змагання з іншими платформами для створення застосунків без коду з низьким порогом входу | Інтеграція можливостей, які пропонують конкуренти |
| Нецінове | Змагання базується на якості програмного забезпечення | Фокус на покращенні якості та додаванні нових функцій |
| Немарочне | Торгова марка не є ключовим фактором вибору | Спрямованість на високу якість продукту замість просування бренду. |

*Таблиця 4.12*

**Аналіз конкуренції в галузі за М. Портером**

| *Складові аналізу* | *Прямі конкуренти в галузі* | *Потенційні конкуренти* | *Поста-чальники* | *Клієнти* | *Товаро-замінники* |
|---|---|---|---|---|---|
| | Wix, Webflow, Bubble та інші | Будь-які low-code платформи | Відсутні | Розмір закупівель, змінні витрати, чутливість до зміни цін, прибутки | Ціна Змінні витрати |
| *Висновки* | Інтенсив-ність висока | Можливість входу в ринок присутня, потенційні конкуренти наявні, але пропонують складніші опції з вищим порогом входу | Постача льники не диктують умови роботи на ринку | Клієнти обирають найдоступні ші варіанти із функціями, які їм потрібні | Обмежень немає, основна конкуренція іде від прямих конкурентів |

*Таблиця 4.13*

**Фактори конкурентоспроможності**

| *№ п/п* | *Фактор конкурентоспроможності* | *Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)* |
|---|---|---|
| 1 | Легкий і інтуїтивно зрозумілий інтерфейс | Спрощує сприйняття нового інструменту, зменшує витрати часу на те, щоб розібратись з новою програмою |
| 2 | Drag-and-drop моделювання | Дозволяює користувачам фокусуватися на бізнес-логіці застосунку, замість написання коду. |
| 3 | Уніфікація | Приведення системи до єдиного виду покращує якість комунікації між її складовими |
| 4 | Робота з базами даних | Можливість використання таблиць для зручного наповнення, представлення або візуалізації, збереження інформації, тощо |
| 5 | Інтеграція з зовнішніми додатками | Вбудовані засоби для інтеграції з іншими популярними сервісами, дозволяючи користувачам легко об'єднувати різні технологічні рішення. |
| 6 | Спрощена система для користувача | Процес розробки доступнішим для людей без технічної освіти, дозволяючи їм створювати застосунки на основі візуальних інтерфейсів. |
| 7 | Шаблони | Наявність готових варіантів з можливістю кастомізації спрощує підбір необхідних блоків і часові витрати на архітектуру. |
| 8 | Адаптивність | Інструменти для створення адаптивних дизайнів, які автоматично оптимізуються для різних пристроїв та екранів. |

*Таблиця 4.13 (продовження)*

**Фактори конкурентоспроможності**

| *№ п/п* | *Фактор конкурентоспроможності* | *Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)* |
|---|---|---|
| 9 | Повторне використання | No-code платформиа пропонують модульну структуру, де готові блоки можуть бути легко повторно використані в різних проектах. |
| 10 | Швидкість розгортання | Централізовані ресурси та готові до використання компоненти дозволяють значно прискорити процес виведення продукту на ринок. |
| 11 | Цінова політика | Розумна вартість у порівнянні з можливостями продукту сприяє його вибору серед нових користувачів. |
| 12 | Розширюваність | Забезпечення засобами для скальованості застосунків, коли це потрібно. |

*Таблиця 4.14*

**Порівняльний аналіз сильних та слабких сторін**

| *№ п/п* | *Фактор конкурентноспроможності* | *Бали 1-20* | *Рейтинг товарів-замінників у порівнянні із стартапом* | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | -3 | -2 | -1 | 0 | +1 | +2 | +3 |
| 1 | Легкий і інтуїтивно зрозумілий інтерфейс | 10 | | | | + | | | |
| 2 | Drag-and-drop моделювання | 15 | | | | | | | + |
| 3 | Уніфікація | 10 | | | | | | + | |

*Таблиця 4.14 (продовження)*

**Порівняльний аналіз сильних та слабких сторін**

| *№ п/п* | *Фактор конкурентноспроможності* | *Бали 1-20* | *Рейтинг товарів-замінників у порівнянні із стартапом* | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | -3 | -2 | -1 | 0 | +1 | +2 | +3 |
| 4 | Робота з базами даних | 5 | | | + | | | | |
| 5 | Інтеграція з зовнішніми додатками | 15 | | | | | | | + |
| 6 | Спрощена система для користувача | 10 | | | | | + | | |
| 7 | Шаблони | 15 | | + | | | | | |
| 8 | Адаптивність | 10 | | | | + | | | |
| 9 | Повторне використання | 15 | | | | | + | | |
| 10 | Швидкість розгортання | 15 | | | | | | + | |
| 11 | Цінова політика | 20 | + | | | | | | |
| 12 | Розширюваність | 10 | | | | | + | | |

*Таблиця 4.15*

**SWOT-аналіз стартап-проєкту**

| | |
|---|---|
| *Сильні сторони:*<br>Легкий і інтуїтивно зрозумілий інтерфейс<br>Drag-and-drop моделювання<br>Уніфікація<br>Робота з базами даних<br>Інтеграція з зовнішніми додатками<br>Спрощена система для користувача<br>Шаблони<br>Адаптивність<br>Повторне використання<br>Швидкість розгортання<br>Цінова політика<br>Розширюваність | *Слабкі сторони:*<br>Малі можливості на старті<br>Доступний функціонал не перекриває переваги конкурентів<br>Не розвинені базові функції, що є в конкурентів |
| *Можливості:*<br>Постійний розвиток<br>No-code методів розробки<br>Використання можливостей запропонованих конкурентами<br>Освоєння ринків, де конкуренція слабка | *Загрози:*<br>Поява конкурентів із передовими технологіями<br>Неефективна реклама<br>Обмеженість функціоналу<br>Неякісний продукт |

*Таблиця 4.16*

**Альтернативи ринкового впровадження стартап-проєкту**

| *№ п/п* | *Альтернатива (орієнтовний комплекс заходів) ринкової поведінки* | *Ймовірність отримання ресурсів* | *Строки реалізації* |
|---|---|---|---|
| 1 | Пошук інвесторів, реалізація MVP проєкту найнятою командою, активна реклама протягом 6 місяців | 75% | 12 місяців |
| 2 | Взяття кредиту, реалізація MVP проєкту найнятою командою, активна реклама протягом 6 місяців | 50% | 10 місяців |
| 3 | Реалізація MVP самостійно, пошук інвесторів для проєкту, активна реклама протягом 6 місяців | 85% | 3 роки |
| 4 | Краудфандинг, реалізація MVP проєкту найнятою командою, активна реклама протягом 6 місяців | 20% | 12 місяців |
| 5 | Участь у спеціалізованих програмах фінансування стартапів, реалізація MVP проєкту найнятою командою, активна реклама протягом 6 місяців | 30% | 15 місяців |

Серед запропонованих альтернатив найкращою видається 1-ша – хоч і трохи довша за другу, але вона має більшу імовірність отримання ресурсу.

## 4.5 Розробка ринкової стратегії проєкту

*Таблиця 4.17*

**Вибір цільових груп потенційних споживачів**

| *№ п/п* | *Опис профілю цільової групи потенційних клієнтів* | *Готовність споживачів сприйняти продукт* | *Орієнтовний попит в межах цільової групи (сегменту)* | *Інтенсивність конкуренції в сегменті* | *Простота входу в сегмент* |
|---|---|---|---|---|---|
| 1 | Великі ІТ-бізнеси | + | Високий | Висока | Середня |
| 2 | Нетехнічні спеціалісти | + | Середній | Середня | Легка |
| 3 | Малі підприємці в сфері ПЗ | + | Висока | Висока | Легка |
| Обрані цільові групи: Малі підприємці, нетехнічні спеціалісти | | | | | |

*Таблиця 4.18*

**Визначення базової стратегії проєкту**

| *№ п/п* | *Обрана альтернатива розвитку проєкту* | *Стратегія охоплення ринку* | *Ключові конкурентноспроможні позиції відповідно до обраної альтернативи* | *Базова стратегія розвитку* |
|---|---|---|---|---|
| 1 | Пошук інвесторів, реалізація | Стратегія концентрованого маркетингу | Легкий і інтуїтивно зрозумілий інтерфейс | Стратегія диференціації |

*Таблиця 4.18 (продовження)*

**Визначення базової стратегії проєкту**

| | | | | |
|---|---|---|---|---|
| | MVP проєкту найнятою командою, активна реклама протягом 6 місяців | | Спрощена система для користувача<br>Шаблони<br>Швидкість розгортання<br>Цінова політика | |

*Таблиця 4.19*

**Визначення стратегії конкурентної поведінки**

| *Чи є проєкт першопрохідцем на ринку?* | *Чи буде компанія шукати нових споживачів, чи забирати існуючих у конкурентів?* | *Чи буде компанія копіювати основні риси товару конкурента, і які?* | *Стратегія конкурентної поведінки* |
|---|---|---|---|
| Ні | Компанія буде шукати і нових споживачів, і буде забирати існуючих у конкурентів | Так: базові функції проектування, зокрема drag-and-drop інтерфейс, шаблони | Стратегія диференціації |

*Таблиця 4.20*

**Визначення стратегії позиціонування**

| *Вимоги до товару цільової аудиторії* | *Базова стратегія розвитку* | *Ключові конкурентноспроможні позиції власного стартап-проєкту* | *Вибір асоціацій, які мають сформувати комплексну позицію власного проєкту (три ключових)* |
|---|---|---|---|
| Якість, зручний інтерфейс, зручна навігація | Стратегія диференціації | Легкий і інтуїтивно зрозумілий інтерфейс<br>Спрощена система для користувача<br>Шаблони | Зручність, простота, доступність для будь-кого, відсутність необхідності довго навчатись |

## 4.6 Маркетингова програма стартап-проєкту

*Таблиця 4.21*

**Визначення ключових переваг концепції потенційного товару**

| *№ п/п* | *Потреба* | *Вигода, яку пропонує товар* | *Ключові переваги перед конкурентами (існуючі або такі, які можна створити)* |
|---|---|---|---|
| 1 | Проектування без спеціалізованих знань | Простота та інтуїтивність | Враховування досвіду конкурентів, складність платформи і необхідність часу на вивчення |
| 2 | Побудова власного сайту | Спрощення процесу | Враховування досвіду співробітників, використання шаблонів та drag-and-drop |

*Таблиця 4.22*

**Опис трьох рівнів моделі товару**

| *Рівні товару* | *Сутність та складові* | | |
|---|---|---|---|
| I. Товар за задумом | No-code платформа | | |
| II. Товар у реальному виконанніТх | *Властивості/характеристики* | *М/Нм* | *Вр/Тх/Тл/Е/Ор* |
| | Зручність | Нм | Е |
| | Надійність | Нм | Е |
| | Відсутність візуальної перевантаженості | Нм | Ор |
| | Наявність шаблонів | Нм | Тх |
| | Уніфікація | Нм | Тх |
| | Робота з базами даних | Нм | Тх |
| III. Товар із підкріпленням | Адаптивність | Нм | Тх |
| | Спрощена система | Нм | Тл |
| | *Якість:* програмне забезпечення має відповідати стандарту ISOIEC_25010 | | |
| | *Пакування:* веб-додаток | | |
| | *Марка:* Droplet | | |
| | *До продажу:* стабільність | | |
| | *Після продажу:* технічна підтримка | | |
| *За рахунок чого товар буде захищено від копіювання:* за рахунок торгової марки та комплексного поєднання властивостей і характеристик товару. | | | |

*Таблиця 4.23*

**Концепція маркетингових комунікацій**

| *Специфіка поведінки цільових клієнтів* | *Канали комунікацій, якими користуються цільові клієнти* | *Ключові позиції, обрані для позиціонування* | *Завдання рекламного повідомлення* | *Концепція рекламного звернення* |
|---|---|---|---|---|
| Цільові клієнти дізнаються про товар з реклами в Інтернеті та рекламних статей у виданнях | Інтернет, Facebook, Instagram, X(раніше Твіттер) | Інтуїтивність інтерфейсу, простота розуміння | Профінформувати про існування продукту та функції, що вирізняють продукт серед конкурентів | Швидкість і простота. Рекламне звернення: “Дропни новий сайт з Droplet!” |

## 4.7 Виробничий план

*Таблиця 4.24*

**Календарний план-графік реалізації стартап-проєкту**

| *№ п/п* | *Етапи реалізації* | *Період реалізації проєкту* | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | *0-ий рік* | | | | *1-ий рік* | *2-ий рік* | *3-ій рік* |
| | | *1-ий кв.* | *2-ий кв.* | *3-ий кв.* | *4-ий кв.* | | | |
| 1 | Проведення науково-дослідницьких робіт | + | | | | | | |
| 2 | Розробка ТЗ та ТЕО | + | | | | | | |
| 4 | Створення компанії | + | | | | | | |

*Таблиця 4.24 (продовження)*

**Календарний план-графік реалізації стартап-проєкту**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | Придбання нематеріальних активів, отримання дозвільних документів, тощо | + | + | | | | | |
| 7 | Придбання обладнання | | + | | | | | |
| 8 | Передвиробничі маркетингові дослідження | | + | + | | | | |
| 9 | Робоче проєктування та тестування | | + | + | | | | |
| 10 | Впровадження системи | | | + | + | | | |
| 11 | Рекламна кампанія | | | + | + | + | | |
| 12 | Продаж продукції | | | | + | + | + | + |

*Таблиця 4.25*

**Планова потреба у виробничому обладнанні та устаткуванні**

| *№ п/п* | *Вид обладнання (устаткування, пристрою)* | *Тип (модель)* | *Виробник обладнання (устаткування, пристрою)* | *Терміни постачання* | *Вартість за 1 шт., грн.* | *Кількість, шт.* |
|---|---|---|---|---|---|---|
| 1 | Ноутбук | Aspire 5 Spin 14 | Acer | 7 днів | 34 999 | 7 |
| 2 | Комп’ютерна миша | M100 USB Black | Logitech | 7 днів | 289 | 7 |
| 3 | Гарнітура | MHS-123 Black | Gembird | 7 днів | 140 | 7 |
| 4 | Операційна система | Windows 10 | Microsoft | 1 день | 1649 | 7 |
| | | | | *Разом:* | 259 539 грн | |

*Таблиця 4.26*

**Планова вартість нематеріальних активів**

| *№ п/п* | *Вид активів* | *Активи, що можуть бути віднесені до даного типу* | *Вартість, грн.* |
|---|---|---|---|
| 1 | Авторське право та суміжні з ним права | Право власності на код ПЗ, що стосується продукту | 10 000 |
| 2 | Право інтелектуальної власності | Торгова марка | 11 105 |
| | | *Всього:* | 21 105 |

*Таблиця 4.27*

**Плановий обсяг виробництва продукції стартап-проєкту**

| *Вид продукції* | *Одиниця виміру* | *Обсяг виробництва за період* | | |
|---|---|---|---|---|
| | | *1-ий рік* | *2-ий рік* | *3-ій рік* |
| Програмне забезпечення | Підписка | 1000 | 4000 | 10000 |

*Таблиця 4.28*

**Планова потреба та витрати на персонал**

| *№ п/п* | *Категорія персоналу* | *Чисельність* | *Заробітна плата, грн/міс.* | *Відрахування на соціальні заходи, грн/міс.* | *Виплати на оплату праці за період, грн.* | | |
|---|---|---|---|---|---|---|---|
| | | | | | *1-ий рік* | *2-ий рік* | *3-ій рік* |
| 1 | Back-end розробник | 2 | 60 000 | 4 474 | 773 688 | 773 688 | 773 688 |

*Таблиця 4.28 (продовження)*

**Планова потреба та витрати на персонал**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 2 | Front-end розробник | 1 | 60 000 | 4 474 | 773 688 | 773 688 | 773 688 |
| 3 | UI\UX дизайнер | 1 | 50 000 | 4 474 | 653 688 | 653 688 | 653 688 |
| 4 | Тестувальник | 1 | 40 000 | 3 474 | 521 688 | 521 688 | 521 688 |
| 5 | Проєктний менеджер | 1 | 80 000 | 5 474 | 1 025 688 | 1 025 688 | 1 025 688 |
| 6 | Маркетолог | 1 | 30 000 | 2 974 | 395 688 | 395 688 | 395 688 |
| *Разом за весь персонал:* | | 7 | 380 000 | 29 818 | 4 917 816 | 4 917 816 | 4 917 816 |

Соціальні заходи рахуються, як ЄСВ + 5% від зарплати, так як усі співробітники компанії працюють, як ФОПи. Мінімальний внесок ЄСВ на момент написання роботи становить 1 474 грн.

*Таблиця 4.29*

**Загальні початкові витрати проєкту**

| *№ п/п* | *Стаття витрат* | *Обсяги витрат в 0-ий рік, грн.* |
|---|---|---|
| 1 | Проведення науково-дослідницьких робіт | 2 000 |
| 2 | Розробка ТЗ і ТЕО | - |
| 3 | Робоче проєктування та тестування | Входить у витрати персоналу |
| 4 | Витрати на придбання обладнання | 259 539 |
| 5 | Витрати на хостинг і хмарні сервіси | 30 000 |
| 6 | Витрати на придбання нематеріальних активів | 21 105 |
| 7 | Витрати на передвиробничі маркетингові дослідження і створення збутної мережі | - |
| 8 | Витрати, пов’язані з діяльністю персоналу | 4 917 816 |
| | *Разом:* | 5 230 460 грн |

*Таблиця 4.30*

**Планові загальногосподарські витрати**

| *№ п/п* | *Стаття витрат* | *Витрати за період, тис. грн.* | | |
|---|---|---|---|---|
| | | *1-ий рік* | *2-ий рік* | *3-ій рік* |
| 1 | Витрати на хостинг і хмарні сервіси | 50 000 | 70 000 | 100 000 |
| 2 | Витрати на персонал (на відрядження, соціальні заходи тощо) | 4 917 816 | 4 917 816 | 4 917 816 |
| 3 | Витрати на просування та рекламу | 50 000 | 50 000 | 50 000 |
| *Разом:* | | 5 017 816 | 5 037 816 | 5 087 816 |

## Висновки до 4 розділу

У цьому розділі було розроблено стартап-проєкт, що та проведено його маркетинговий аналіз. Було створено інформаційну картку проєкту, сформовано опис необхідної команди проєкту, сформульовано основні ідеї, проаналізовано ринкові можливості, розроблено ринкову стратегію, маркетингову програму та сформовано виробничий план.

В ході аналізу було сформовано необхідність в команді з 7 людей для реалізації проєкту. Серед запропонованих альтернатив впровадження стартап проєкту було обрано наступну: пошук інвесторів, реалізація MVP проєкту найнятою командою, активна реклама продукту протягом 6 місяців. Цільовими групами проєкту було визначено людей пов’язаних з IT, що мають нетехнічні спеціальності, та широкий загал, Стратегією охоплення ринку було обрано стратегію концентрованого маркетингу, базовою стратегією було обрано стратегію диференціації, а стратегію конкурентної поведінки було теж обрано стратегію диференціації.

Було визначено ключові характеристики, які вигідно вирізнятимуться серед конкурентів і стимулюватимуть зацікавленість в продукті. Серед них: простота та інтуїтивність використання, краща цінова політика.

Обчислено, що початкові витрати проєкту в нульовий рік становитимуть 5 230 460 грн, в 1-ий рік – 5 017 816грн, в 2-ий рік – 5 037 816грн, а в 3-ій рік - 5 087 816 грн.

# ВИСНОВКИ

Магістерська робота присвячена no-code та low-code платформам, як засобам проектування програмного забезпечення. Метою магістерської роботи є зменшення вхідного порогу для створення веб-сайтів та застосунків за допомогою можливостей no-code та low-code платформ

У роботі проаналізовано основні цілі використання low-code і no-code платформ, та їх роль в сучасному становищі розробки. Виконано огляд існуючих рішень. За рахунок порівняння виявлено відмінності між традиційною розробкою та розробкою з використанням low-code і no-code платформи, оцінені приблизні фінансові витрати. Проведено базовий аналіз існуючих на ринку рішень, їх переваг та недоліків.

Виконано технологічний огляд засобів реалізації, обґрунтовано вибір мови програмування, архітектури та сервісів. Виконано опис структури програмного додатку та реалізації його компонент. Роглянуто процес створення програмного забезпечення за допомогою програмування та no-code платформи. Сформовано функціональні та нефункціональні вимоги, визначено ролі користувачів в системі, а також описані варіанти використання.

Також оглянуто процес конструювання веб-сайту на платформі, доступні розширення та засоби проектування. Описано основні способи взаємодії з платформою, внутрішні можливості її використання для адміністратора. а також додаткові можливості для інтеграції та налаштування.

Спроектовано дві версії вебсайту – програмна та платформена, підраховані часові витрати на обидва процеси.

Проведено соціологічне опитування, оброблені результати.

Результати досліджень пізніше можуть бути використані для покращення процесів проектування програмного забезпечення.

# СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ


1. J.Bloomberg, C.Tomasi, Brad Tilton, “ Low-Code For Dummies: Creating Applications using Visual Tools and Model-Driven Processes” John Wiley & Sons, Inc.,2020
2. R. Picek, "Low-code/No-code Platforms and Modern ERP Systems," 2023 International Conference on Information Management (ICIM), Oxford, United Kingdom, 2023,
3. The Defenitive Guide to Low-Code [Електронний ресурс] – Режим доступу до ресурсу: https://www.mendix.com/low-code-guide/
4. No-code and low-code similarities and differences [Електронний ресурс] – Режим доступу до ресурсу: https://www.mendix.com/blog/understand-no-code-vs-low-code-development-tools/
5. The OutSystems platform brings high-performance to low-code [Електронний ресурс] – Режим доступу до ресурсу: https://www.outsystems.com/low-code-platform/
6. Appian Total Experience [Електронний ресурс] – Режим доступу до ресурсу: https://docs.appian.com/suite/help/23.4/total-experience.html
7. Mendix Platform Overview [Електронний ресурс] – Режим доступу до ресурсу: https://www.mendix.com/platform/
8. Get started with Lightning Experience [Електронний ресурс] – Режим доступу до ресурсу: https://www.salesforce.com/campaign/lightning/
9. Low-code app development [Електронний ресурс] – Режим доступу до ресурсу: https://www.pega.com/products/platform/low-code-app-development
10. Developer-first advanced web creation [Електронний ресурс] – Режим доступу до ресурсу: https://www.wix.com/studio/development
11. Bubble for developers [Електронний ресурс] – Режим доступу до ресурсу: https://bubble.io/for-developers?ref=design-header
12. Experience the power of code. Without writing it. [Електронний ресурс] – Режим доступу до ресурсу: https://webflow.com/designer

13. The Future of No-Code Report [Електронний ресурс] – Режим доступу до ресурсу: https://ru.adalo.com/the-future-is-no-code/conclusions
14. Glide Docs [Електронний ресурс] – Режим доступу до ресурсу: https://www.glideapps.com/docs/quick-start
15. No Code / Low Code vs. Custom Traditional Development [Електронний ресурс] – Режим доступу до ресурсу https://themobilereality.com/blog/no-code-low-code-vs-traditional-development
16. Probability of one random variable being greater than another [Електронний ресурс] – Режим доступу до ресурсу: https://stats.stackexchange.com/questions/50501/probability-of-one-random-variable-being-greater-than-another

# ДОДАТОК А

Платформи no-code та low-code

Діаграма варіантів використання

Аркушів: 1

Київ – 2024 року

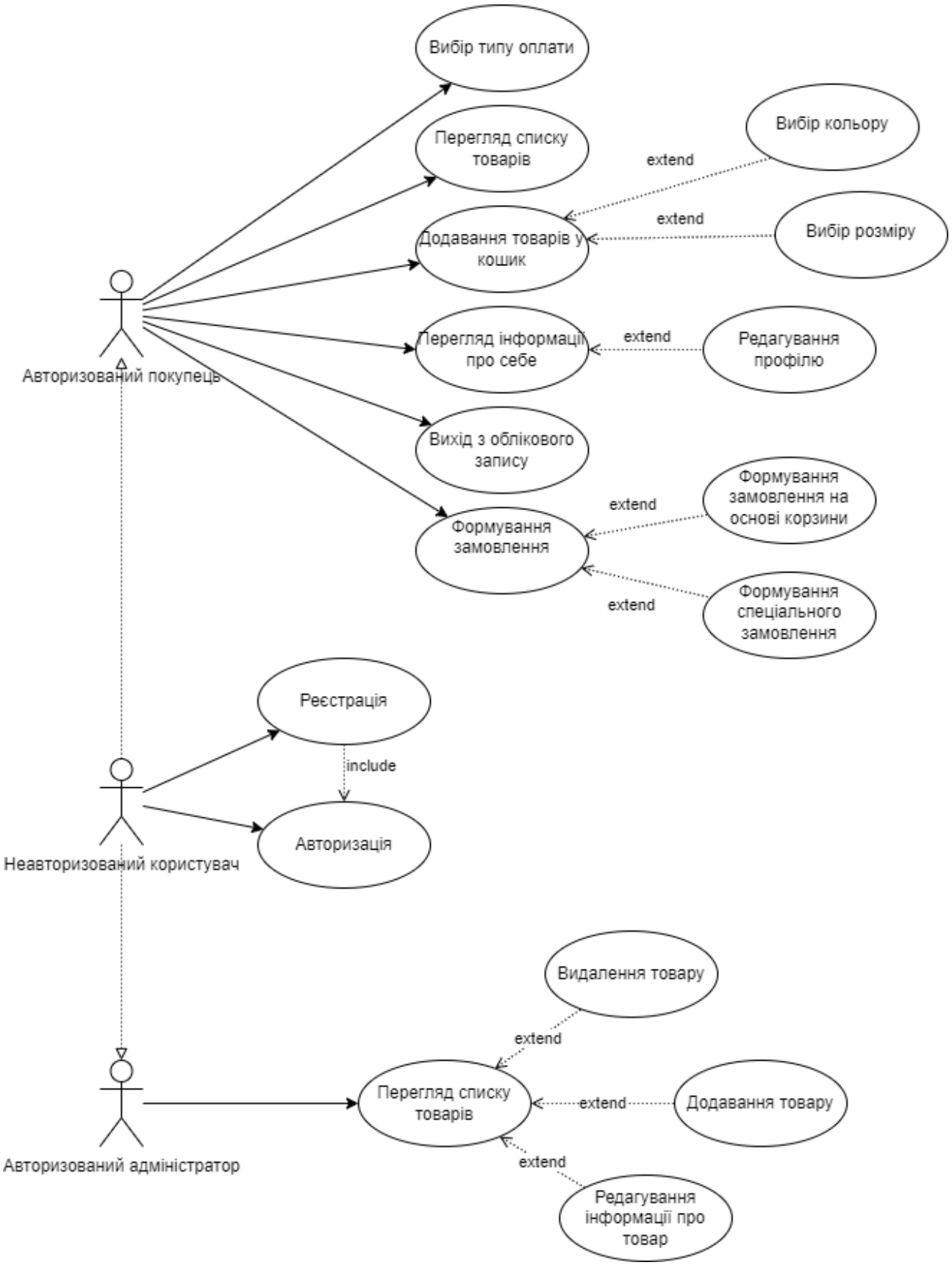
Вибір типу оплати
Перегляд списку товарів
Вибір кольору
extend
Додавання товарів у кошик
extend
Вибір розміру
Авторизований покупець
Перегляд інформації про себе
extend
Редагування профілю
Вихід з облікового запису
Формування замовлення на основі корзини
extend
Формування замовлення
extend
Формування спеціального замовлення
Реєстрація
include
Неавторизований користувач
Авторизація
Видалення товару
extend
Авторизований адміністратор
Перегляд списку товарів
extend
Додавання товару
extend
Редагування інформації про товар

Рис.А.1

# ДОДАТОК Б

Платформи no-code та low-code

Діаграма Presentation Layer

Аркушів: 1

Київ – 2024 року

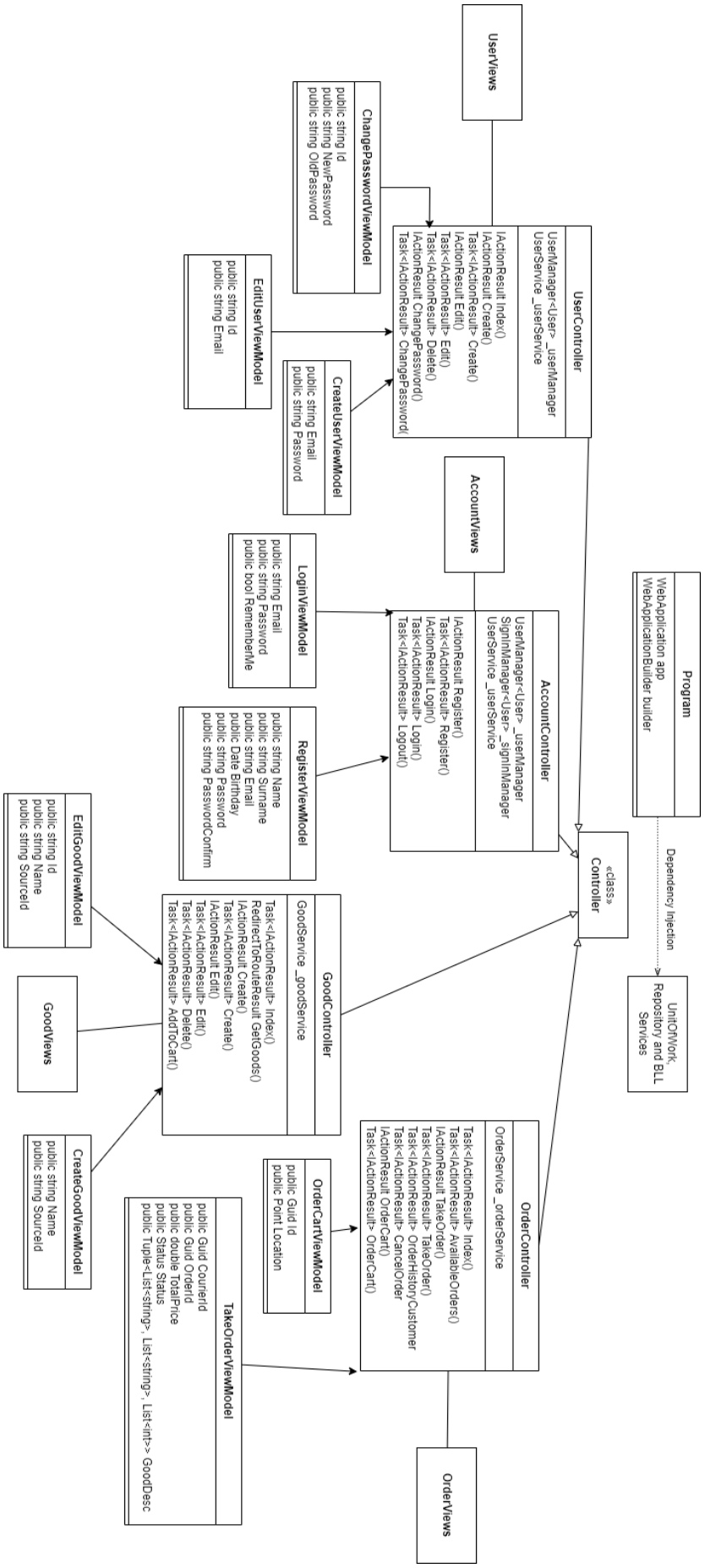
Program
WebApplication app
WebApplicationBuilder builder
Dependency Injection
«class»
Controller
UnitOfWork,
Repository and BLL
Services
UserViews
UserController
UserManager<User> _userManager
UserService _userService
IActionResult Index()
IActionResult Create()
Task<IActionResult> Create()
IActionResult Edit()
Task<IActionResult> Edit()
Task<IActionResult> Delete()
IActionResult ChangePassword()
Task<IActionResult> ChangePassword(
ChangePasswordViewModel
public string Id
public string NewPassword
public string OldPassword
EditUserViewModel
public string Id
public string Email
CreateUserViewModel
public string Email
public string Password
AccountViews
AccountController
UserManager<User> _userManager
SignInManager<User> _signInManager
UserService _userService
IActionResult Register()
Task<IActionResult> Register()
IActionResult Login()
Task<IActionResult> Login()
Task<IActionResult> Logout()
LoginViewModel
public string Email
public string Password
public bool RememberMe
RegisterViewModel
public string Name
public string Surname
public string Email
public Date Birthday
public string Password
public string PasswordConfirm
GoodController
GoodService _goodService
Task<IActionResult> Index()
RedirectToRouteResult GetGoods()
IActionResult Create()
Task<IActionResult> Create()
IActionResult Edit()
Task<IActionResult> Edit()
Task<IActionResult> Delete()
Task<IActionResult> AddToCart()
EditGoodViewModel
public string Id
public string Name
public string SourceId
GoodViews
CreateGoodViewModel
public string Name
public string SourceId
OrderController
OrderService _orderService
Task<IActionResult> Index()
Task<IActionResult> AvailableOrders()
IActionResult TakeOrder()
Task<IActionResult> TakeOrder()
Task<IActionResult> OrderHistoryCustomer
Task<IActionResult> CancelOrder
IActionResult OrderCart()
Task<IActionResult> OrderCart()
OrderCartViewModel
public Guid Id
public Point Location
TakeOrderViewModel
public Guid CourierId
public Guid OrderId
public double TotalPrice
public Status Status
public Tuple<List<string>, List<string>, List<int>> GoodDesc
OrderViews

Рис. Б.1

# ДОДАТОК В

Платформи no-code та low-code

Діаграма Business Logic Layer

Аркушів: 1

Київ – 2024 року

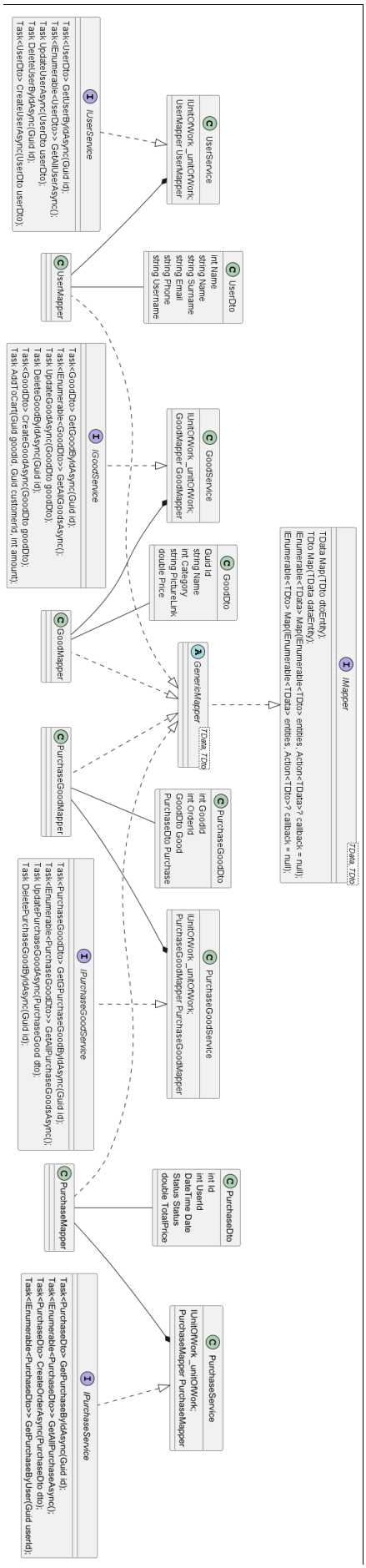
IMapper
TData, TDto
TData Map(TDto dtoEntity);
TDto Map(TData dataEntity);
IEnumerable<TData> Map(IEnumerable<TDto> entities, Action<TData>? callback = null);
IEnumerable<TDto> Map(IEnumerable<TData> entities, Action<TDto>? callback = null);
GenericMapper
TData, TDto
IUserService
Task<UserDto> GetUserByIdAsync(Guid id);
Task<IEnumerable<UserDto>> GetAllUserAsync();
Task UpdateUserAsync(UserDto userDto);
Task DeleteUserByIdAsync(Guid id);
Task<UserDto> CreateUserAsync(UserDto userDto);
UserService
IUnitOfWork _unitOfWork;
UserMapper UserMapper
UserMapper
UserDto
int Name
string Name
string Surname
string Email
string Phone
string Username
IGoodService
Task<GoodDto> GetGoodByIdAsync(Guid id);
Task<IEnumerable<GoodDto>> GetAllGoodsAsync();
Task UpdateGoodAsync(GoodDto goodDto);
Task DeleteGoodByIdAsync(Guid id);
Task<GoodDto> CreateGoodAsync(GoodDto goodDto);
Task AddToCart(Guid goodId, Guid customerId, int amount);
GoodService
IUnitOfWork _unitOfWork;
GoodMapper GoodMapper
GoodMapper
GoodDto
Guid Id
string Name
int Category
string PictureLink
double Price
IPurchaseGoodService
Task<PurchaseGoodDto> GetGPurchaseGoodByIdAsync(Guid id);
Task<IEnumerable<PurchaseGoodDto>> GetAllPurchaseGoodsAsync();
Task UpdatePurchaseGoodAsync(PurchaseGood dto);
Task DeletePurchaseGoodByIdAsync(Guid id);
PurchaseGoodService
IUnitOfWork _unitOfWork;
PurchaseGoodMapper PurchaseGoodMapper
PurchaseGoodMapper
PurchaseGoodDto
int GoodId
int OrderId
GoodDto Good
PurchaseDto Purchase
IPurchaseService
Task<PurchaseDto> GetPurchaseByIdAsync(Guid id);
Task<IEnumerable<PurchaseDto>> GetAllPurchaseAsync();
Task<PurchaseDto> CreateOrderAsync(PurchaseDto dto);
Task<IEnumerable<PurchaseDto>> GetPurchaseByUser(Guid userId);
PurchaseService
IUnitOfWork _unitOfWork;
PurchaseMapper PurchaseMapper
PurchaseMapper
PurchaseDto
int Id
int UserId
DateTime Date
Status Status
double TotalPrice

Рис.В.1