Міністерство освіти і науки України
Кам’янець-Подільський національний університет імені Івана Огієнка
Фізико-математичний факультет
Кафедра комп’ютерних наук

Кваліфікаційна робота магістра

з теми:

«РОЗРОБКА ПЛАТФОРМИ ДЛЯ НАДАННЯ ПОСЛУГ ЗІ СТВОРЕННЯ КАСТОМІЗОВАНИХ ОНЛАЙН-БІЗНЕСІВ НА СТЕКУ MEAN»

Виконав: здобувач вищої освіти групи Kn1-M24
спеціальності 122 Комп’ютерні науки
Продоляк Богдан Олегович
Керівник: Щирба Віктор Самуїлович,
кандидат фізико-математичних наук, доцент
Рецензент:
СЕМЕНЕЦЬ І.В., декан природничо-економічного факультету, кандидат економічних наук, доцент
СМОРЖЕВСЬКИЙ Ю.Л., завідувач кафедри математики, кандидат педагогічнх наук, доцент

Кам’янець-Подільський – 2025 р.

ЗМІСТ

АНОТАЦІЯ

Продоляк Б.О. Розробка мобільного застосунку для контролю та планування особистого бюджету на стеку MEAN. Кваліфікаційна робота бакалавра на здобуття освітньо-кваліфікаційного рівня вищої освіти спеціальності 122 «Комп’ютерні науки». Кам’янець-Подільський національний університет імені Івана Огієнка, Кам’янець-Подільський, 2024.

У роботі проаналізовано сучасні методи та технологій для розробки мобільних застосунків для управління особистими фінансами. Ретельно досліджено основні потреби користувачів і конкурентні рішення на ринку програмних засобів для фінансового менеджменту. Визначено ключові функціональні та нефункціональні вимоги до розроблюваної системи.

На основі окреслених вимог і сформованої моделі розроблено мобільний застосунок Spike, який забезпечує користувачів зручною та функціональною платформою для управління особистими фінансами, допомагає їм створювати та контролювати власний бюджет, стежити за доходами та витратами, встановлювати фінансові цілі та отримувати аналітичні звіти. «»

Ключові слова: мобільний застосунок, стек MEAN, MongoDB, Express.js, Angular, Node.js, особистий бюджет, управління фінансами.

ВСТУП

У сучасному світі цифрових технологій та електронної комерції створення онлайн-бізнесів стає все більш популярним. Підприємці прагнуть автоматизувати свої процеси та забезпечити ефективну взаємодію з клієнтами через веб-платформи. Однак розробка онлайн-бізнесу з нуля вимагає значних технічних знань і ресурсів. Це обумовлює потребу у гнучких рішеннях, що дозволяють швидко створювати кастомізовані онлайн-бізнеси без необхідності глибокого знання програмування.

Актуальність дослідження. Багато підприємців стикаються з труднощами під час розробки власних онлайн-платформ через складність інтеграції сервісів та технічні обмеження. Використання готових платформ може спростити цей процес, але вони часто мають обмежений функціонал або не повністю відповідають вимогам користувачів. Тому важливим є створення універсальної платформи, що дозволяє налаштовувати онлайн-бізнес відповідно до індивідуальних потреб підприємців.

Об'єкт дослідження – процеси створення та автоматизації онлайн-бізнесів на стеку MEAN.

Предмет дослідження – методи та технології розробки платформи для кастомізації онлайн-бізнесів з використанням стеку MEAN.

Мета роботи – розробка платформи Spike для створення кастомізованих онлайн-бізнесів на основі технологічного стеку MEAN, яка спрощуватиме процес запуску веб-проєктів та автоматизуватиме їх налаштування.

Для досягнення цієї мети були визначені такі **завдання**:

1) Дослідити основні принципи організації та ведення онлайн-бізнесу.
2) Провести аналіз існуючих рішень та визначити їхні переваги й недоліки.
3) Обґрунтувати вибір технологічного стеку для реалізації проєкту.
4) Розробити архітектуру системи та визначити її функціональні можливості.

5) Реалізувати основні компоненти платформи Spike з можливістю кастомізації.
6) Виконати тестування системи та оцінити її ефективність.

Під час виконання дипломного дослідження застосовувалися різні методи. Теоретичний аналіз охоплював вивчення наукової літератури, статей та аналітичних матеріалів, присвячених веб-платформам і управлінню онлайн-бізнесом. Практична частина включала розробку архітектури, програмування компонентів платформи Spike за допомогою стеку MEAN, а також тестування її функціональності та безпеки. Метод порівняльного аналізу дозволив оцінити переваги різних технологічних підходів, а експериментальне тестування допомогло перевірити продуктивність платформи в реальних умовах експлуатації.

У підсумку, розроблений у межах дипломної роботи проєкт платформи Spike відповідає сучасним вимогам і стандартам, забезпечуючи підприємців гнучкими та зручними інструментами для створення онлайн-бізнесів. Завдяки можливостям кастомізації та автоматизації процесів платформа сприятиме підвищенню ефективності бізнесу та залученню широкої аудиторії.

РОЗДІЛ 1
ОСНОВИ СТВОРЕННЯ, УПРАВЛІННЯ ТА ПЛАНУВАННЯ ОНЛАЙН-БІЗНЕСУ

1.1 Дослідження ринку онлайн-бізнесів та аналіз існуючих платформ для їх розробки

У наш час, коли цифрові технології розвиваються надзвичайно швидкими темпами, онлайн-бізнеси займають дедалі важливіше місце в глобальній економіці. Зростання популярності електронної комерції зумовлене кількома ключовими чинниками: широким доступом до інтернету, зручністю здійснення покупок у мережі та змінами в поведінці споживачів. Згідно з аналітичними даними платформи *Statista*, очікується, що обсяг світового ринку електронної комерції досягне 7,4 трильйона доларів США вже у 2025 році, що свідчить про високий потенціал цього сегмента [1]. Україна також демонструє динамічний розвиток у цій сфері, насамперед завдяки зростанню кількості інтернет-користувачів та поступовому вдосконаленню систем електронних платежів.

Однак, незважаючи на зростання популярності онлайн-бізнесів, багато підприємців стикаються з серйозними викликами. Одним із головних бар’єрів є високі витрати на розробку та підтримку власних платформ. Для створення індивідуального рішення потрібні значні інвестиції в технічні знання, обладнання та програмне забезпечення. Крім того, багато малих і середніх підприємств не мають достатніх ресурсів для найму професійних розробників, що ускладнює їхній вихід на онлайн-ринок. Іншим важливим аспектом є конкуренція з великими гравцями, які вже мають стабільну позицію на ринку та можуть дозволити собі масштабні маркетингові кампанії. Це створює додатковий тиск на нових учасників ринку, які повинні шукати інноваційні підходи для залучення клієнтів.

Ще одним важливим викликом у сфері онлайн-підприємництва є технічна складність інтеграції різноманітних сервісів — зокрема платіжних систем, систем управління замовленнями та аналітичних інструментів. Багато підприємців, які не мають спеціалізованої технічної підготовки, зіштовхуються з труднощами під час налаштування та взаємодії цих компонентів, що призводить до значних витрат часу й ресурсів.

Популярні на ринку платформи, такі як Shopify, Wix або WooCommerce, хоч і пропонують готові до використання рішення, нерідко мають обмежений функціонал або не дозволяють гнучко адаптувати продукт під унікальні потреби конкретного бізнесу. Це, у свою чергу, зменшує можливості для індивідуалізації та ускладнює процес позиціонування компанії в умовах високої конкуренції.

На цьому тлі постає потреба у створенні нових інструментів, які дозволили б підприємцям швидко й ефективно запускати кастомізовані онлайн-бізнеси без необхідності володіння глибокими знаннями у сфері програмування. Такі платформи мають бути інтуїтивно зрозумілими, гнучкими у налаштуванні, масштабованими та орієнтованими на користувача. Важливо також забезпечити просту інтеграцію з популярними зовнішніми сервісами — платіжними системами, соціальними мережами, CRM та інструментами аналітики. Усе це дозволить значно полегшити запуск та подальше управління онлайн-бізнесом. Отже, розробка подібної платформи є не лише актуальним, але й стратегічно важливим завданням, яке має значний ринковий потенціал.

На сьогоднішній день існує багато готових рішень для створення онлайн-бізнесів, які дозволяють підприємцям швидко запускати свої проекти без необхідності глибоких технічних знань. Одним із найпопулярніших інструментів є платформа «Shopify», яка спеціалізується на електронній комерції. Вона пропонує зручний інтерфейс для створення інтернет-магазинів, інтеграцію з різними платіжними системами та велику кількість готових шаблонів. Однак, незважаючи на свою популярність, «Shopify» має певні обмеження. Наприклад, кастомізація платформи часто вимагає знання мови

«Liquid», що ускладнює процес для користувачів без технічної підготовки. Крім того, вартість використання «Shopify» може бути високою для малих бізнесів, особливо якщо враховувати додаткові витрати на плагіни та теми.

Іншим популярним рішенням є «Wix», який пропонує конструктор сайтів з функцією drag-and-drop. Ця платформа дозволяє створювати сайти будь-якої складності, від простих лендінгів до складних інтернет-магазинів. «Wix» відрізняється простотою використання та низьким бар'єром входу, що робить його привабливим для початківців. Проте, «Wix» має обмежену продуктивність для великих проектів, а також недостатню гнучкість у порівнянні з іншими рішеннями. Наприклад, перехід на іншу платформу після створення сайту на «Wix» може бути дуже складним через закриту архітектуру системи.

Для тих, хто працює з «WordPress», «WooCommerce» є одним із найпоширеніших рішень для створення інтернет-магазинів. Цей плагін дозволяє перетворити будь-який сайт на «WordPress» на повноцінну платформу електронної комерції. «WooCommerce» відрізняється високою гнучкістю та можливістю інтеграції з великою кількістю додаткових інструментів. Однак, для ефективного використання «WooCommerce» потрібні технічні знання, оскільки налаштування плагіну може бути досить складним для новачків.

Окрім цих платформ, існують інші рішення, такі як «Squarespace», «BigCommerce» та «Magento», кожне з яких має свої переваги та недоліки. Наприклад, «Squarespace» відрізняється естетично привабливими шаблонами, але має обмежену функціональність для складних проектів. «BigCommerce» пропонує потужні інструменти для великих бізнесів, але його вартість може бути занадто високою для малих підприємств. «Magento», з іншого боку, є дуже гнучкою платформою з відкритим вихідним кодом, але вимагає значних технічних знань для налаштування та підтримки.

Незважаючи на велику кількість існуючих рішень, багато підприємців стикаються з проблемами при їх використанні. Наприклад, обмежена

кастомізація, висока вартість, складність інтеграції з іншими сервісами та недостатня продуктивність часто стають перешкодами для успішного запуску онлайн-бізнесу. Крім того, багато платформ не дозволяють повністю адаптуватися під специфічні потреби бізнесу, що обмежує можливості підприємців щодо створення унікального продукту. Це підкреслює потребу у нових рішеннях, які б поєднували в собі зручність, гнучкість та доступність, забезпечуючи при цьому високу продуктивність та масштабованість.

1.2 Основи органцізації та управління онлайн-бізнесами

Онлайн-бізнес як сучасна форма підприємницької діяльності ґрунтується на використанні інтернет-технологій для продажу товарів або послуг, взаємодії з клієнтами та управління бізнес-процесами. Його головною особливістю є віртуальний характер, що дає змогу працювати без географічних обмежень і суттєво знижує витрати на оренду приміщень та утримання фізичної інфраструктури. Завдяки високій гнучкості та інноваційним підходам малі й середні підприємства отримують можливість успішно конкурувати з великими компаніями.

Однією з ключових характеристик онлайн-бізнесу є різноманіття моделей його ведення. Найпоширенішою є модель B2C (Business-to-Consumer), за якої товари чи послуги реалізуються безпосередньо кінцевим споживачам. Ця модель охоплює інтернет-магазини, сервіси доставки їжі, потокові платформи тощо. Прикладами B2C-компаній є Amazon і Zalando. За даними eMarketer, у 2023 році обсяг продажів у B2C-сегменті склав 3,6 трильйона доларів США, що становить близько 60% загального обсягу електронної комерції [2].

Ще однією популярною моделлю є B2B (Business-to-Business), що передбачає взаємодію між компаніями — наприклад, у постачанні сировини, програмного забезпечення чи маркетингових послуг. Яскравим прикладом B2B-платформи є Alibaba, яка забезпечує зв’язок між виробниками та

оптовими покупцями. За прогнозами Forrester Research, до 2027 року обсяг B2B-ринку електронної комерції може досягти 20,9 трильйона доларів США [3].

Модель C2C (Consumer-to-Consumer) охоплює транзакції між окремими споживачами, що реалізуються через платформи для продажу вживаних речей (наприклад, OLX) або для оренди житла (наприклад, Airbnb).

Кожна з описаних моделей має свої особливості, вимоги до організації бізнес-процесів та рівень технічної підтримки, що необхідно враховувати при створенні й масштабуванні онлайн-платформ.

Життєвий цикл онлайн-бізнесу починається з формування ідеї та аналізу ринкових можливостей. На цьому етапі важливо визначити цільову аудиторію, провести дослідження конкурентів та розробити бізнес-план. Наприклад, перед запуском нового інтернет-магазину необхідно дослідити попит на товари, визначити переваги перед конкурентами та оцінити можливі ризики. За даними Harvard Business Review, близько 42% стартапів зазнають невдач через відсутність ринкового попиту [4]. Після цього відбувається розробка онлайн-платформи, яка включає проектування інтерфейсу, інтеграцію необхідних сервісів (наприклад, платіжних систем, систем управління замовленнями) та тестування функціональності. Запуск платформи супроводжується маркетинговими активностями, спрямованими на залучення клієнтів. Наприклад, використання соціальних мереж, контекстної реклами та SEO-оптимізації дозволяє привернути увагу до нового бізнесу. За даними HubSpot, компанії, які використовують інтегровані маркетингові стратегії, збільшують свої продажі в середньому на 19% [5]. Після запуску важливим етапом є постійне вдосконалення платформи, збір зворотного зв’язку від користувачів та масштабування бізнесу. Наприклад, компанія Netflix починала як сервіс для доставки DVD, але згодом перетворилася на одну з найпопулярніших потокових платформ завдяки постійному вдосконаленню своїх сервісів.

Організація онлайн-бізнесу потребує ефективного управління ресурсами, серед яких ключову роль відіграють фінансові, технічні та людські ресурси.

Фінансове управління охоплює планування бюджету, контроль витрат і аналіз прибутковості. Застосування сучасних інструментів фінансового аналізу дозволяє виявляти найбільш ефективні канали продажу, оптимізувати витрати й підвищувати загальну рентабельність бізнесу. За даними McKinsey, компанії, які впроваджують передові методи фінансового аналізу, демонструють зростання прибутковості на 20–30% [6].

Технічне управління включає підтримку стабільної роботи онлайн-платформи, забезпечення її кібербезпеки та можливостей масштабування. Використання хмарних технологій, зокрема сервісів AWS або Google Cloud, дозволяє досягти високої доступності, швидкої адаптації до навантажень та зменшення витрат на утримання фізичної інфраструктури.

Управління людськими ресурсами передбачає формування професійної команди, здатної реалізувати всі етапи життєвого циклу онлайн-бізнесу — від розробки продукту до його просування та аналізу ефективності. Успішна реалізація бізнес-ідеї вимагає злагодженої співпраці маркетологів, розробників, аналітиків та інших фахівців, що забезпечує комплексний підхід до розвитку платформи.

Одним із ключових елементів успішного онлайн-бізнесу є використання сучасних інструментів для автоматизації та оптимізації процесів. Наприклад, системи управління замовленнями (Order Management Systems) дозволяють автоматизувати обробку замовлень, відстежувати їх статус та інтегрувати дані з іншими системами. Інструменти аналітики, такі як Google Analytics чи Tableau, допомагають аналізувати поведінку користувачів, визначати найефективніші маркетингові кампанії та приймати обґрунтовані бізнес-рішення. Крім того, важливим є використання штучного інтелекту та машинного навчання для персоналізації сервісів. Наприклад, рекомендаційні системи, які використовуються Amazon чи Netflix, дозволяють підвищити

задоволеність клієнтів та збільшити продажі. За даними Accenture, компанії, які використовують AI для персоналізації, збільшують свої доходи на 6-10% [7].

Таким чином, організація та управління онлайн-бізнесом вимагають комплексного підходу, який включає не лише технічні аспекти, але й ефективне управління ресурсами, маркетингові стратегії та постійне вдосконалення платформи. Це дозволяє підприємцям адаптуватися до швидкозмінного ринку, задовольняти потреби клієнтів та досягати стабільного розвитку. До прикладу, компанія Shopify, яка почала як невеликий стартап, згодом стала однією з найпопулярніших платформ для створення інтернет-магазинів завдяки постійному вдосконаленню своїх сервісів та орієнтації на потреби клієнтів.

1.3 Планування та наукові підходи до розробки платформ для онлайн-бізнесів

Створення та розвиток онлайн-бізнесів супроводжуються низкою викликів, які вимагають комплексного підходу до їх вирішення. Ці виклики можна умовно розділити на три основні категорії: **технічні**, **організаційні** та **ринкові**. Кожна з них має свої особливості та вимагає використання сучасних підходів для ефективного подолання.

Технічні виклики

Технічна складність є одним із головних бар’єрів для створення онлайн-платформ. Багато підприємців, особливо малого та середнього бізнесу, стикаються з труднощами при інтеграції різних сервісів, таких як платіжні системи, системи управління замовленнями та інструменти аналітики. Це пов’язано з тим, що кожен із цих сервісів має свої специфічні вимоги до інтеграції, що вимагає глибоких технічних знань. Наприклад, інтеграція

платіжної системи Stripe чи PayPal вимагає не лише налаштування API, але й забезпечення відповідного рівня безпеки для обробки фінансових даних.

Крім того, обмежена кастомізація існуючих платформ часто не дозволяє повністю реалізувати бізнес-ідеї. Багато готових рішень, таких як Shopify чи Wix, пропонують лише базовий функціонал, який не завжди відповідає специфічним потребам бізнесу. Це обмежує можливості підприємців щодо створення унікального продукту та диференціації на ринку.

Ще одним важливим аспектом є висока вартість розробки та підтримки платформ. Для створення індивідуального рішення потрібні значні інвестиції в технічні знання, обладнання та програмне забезпечення. Наприклад, використання хмарних технологій, таких як AWS чи Google Cloud, хоча й дозволяє знизити витрати на інфраструктуру, все ж вимагає значних фінансових ресурсів для налаштування та підтримки. Це є особливо важливим для малих бізнесів, які часто мають обмежений бюджет.

Організаційні аспекти

Організаційні аспекти також є важливим бар’єром для розвитку онлайн-бізнесів. Управління ресурсами, включаючи фінансові, технічні та людські, вимагає професійного підходу. Відсутність чіткої стратегії масштабування може призвести до неефективного використання ресурсів та зниження прибутковості. Наприклад, багато стартапів зазнають невдач через те, що не мають чіткого плану розвитку та не можуть ефективно розподіляти ресурси між різними напрямками діяльності.

Крім того, конкуренція з великими гравцями ринку, такими як Amazon чи Alibaba, створює додатковий тиск на нових учасників. Великі компанії мають значні ресурси для проведення масштабних маркетингових кампаній, що ускладнює залучення клієнтів для малих бізнесів. Щоб виділитися на тлі конкурентів, підприємцям необхідно використовувати інноваційні підходи, такі як персоналізація сервісів, використання соціальних мереж для просування та забезпечення високого рівня обслуговування клієнтів.

Ринкові виклики

Ринкові умови також є важливим фактором, який впливає на розвиток онлайн-бізнесів. Швидкі зміни в поведінці споживачів, поява нових технологій та зростання конкуренції вимагають від підприємців постійної адаптації. Наприклад, зростання популярності мобільних пристроїв призвело до того, що більшість онлайн-платформ повинні бути оптимізовані для мобільних користувачів. Це вимагає додаткових інвестицій у розробку мобільних додатків чи адаптивних веб-інтерфейсів.

Крім того, збільшення вимог до безпеки даних та конфіденційності користувачів вимагає від компаній інвестувати в сучасні системи захисту. Наприклад, впровадження механізмів двофакторної автентифікації, шифрування даних та регулярне оновлення програмного забезпечення є необхідними кроками для забезпечення безпеки платформи.

Наукові підходи до розробки платформ

Для подолання цих викликів необхідний науковий підхід до розробки платформ, який би враховував сучасні тенденції та технології. Одним із ключових напрямків є використання **методологій розробки програмного забезпечення**, таких як Agile, Scrum чи Waterfall. Agile, зокрема, дозволяє гнучко реагувати на зміни вимог та забезпечує швидку реалізацію нових функцій. Це особливо важливо для онлайн-бізнесів, які повинні швидко адаптуватися до змін на ринку.

Іншим важливим аспектом є використання **сучасних технологій для автоматизації та оптимізації процесів**. Системи управління замовленнями (Order Management Systems) дозволяють автоматизувати обробку замовлень, відстежувати їх статус та інтегрувати дані з іншими системами. Інструменти аналітики, такі як Google Analytics чи Tableau, допомагають аналізувати поведінку користувачів, визначати найефективніші маркетингові кампанії та приймати обґрунтовані бізнес-рішення.

Крім того, використання **штучного інтелекту (AI) та машинного навчання (ML)** дозволяє персоналізувати сервіси та підвищити задоволеність клієнтів. Наприклад, рекомендаційні системи, які використовуються Amazon чи Netflix, дозволяють збільшити продажі завдяки персоналізації.

Важливим науковим підходом є також **використання хмарних технологій** для забезпечення масштабованості та високої доступності платформ. Хмарні сервіси, такі як AWS, Google Cloud чи Microsoft Azure, дозволяють знизити витрати на інфраструктуру та забезпечити високу продуктивність. Крім того, важливим є **забезпечення безпеки платформ**, що включає захист даних користувачів, шифрування платежів та захист від кібератак. Використання сучасних методів шифрування, таких як SSL/TLS, та механізмів автентифікації, таких як OAuth чи JWT (JSON Web Tokens), дозволяє забезпечити надійний захист даних.

Отже, розділ дослідження присвячено аналізу ринку онлайн-бізнесів, теоретичним основам їх організації та управління, а також проблематиці, яка виникає під час їх створення та розвитку. Було встановлено, що онлайн-бізнеси є важливим елементом сучасної економіки, який продовжує стрімко розвиватися. Однак їх створення супроводжується низкою викликів, включаючи технічну складність, високу вартість розробки та організаційні труднощі. Для подолання цих викликів необхідний науковий підхід, який би враховував сучасні тенденції та технології. Використання методологій розробки, таких як Agile, сучасних технологій, таких як AI та хмарні сервіси, а також інструментів аналітики дозволяє забезпечити стабільний розвиток онлайн-платформ. Це створює основу для подальшого дослідження, яке буде спрямоване на розробку конкретних рішень для створення платформи Spike.

РОЗДІЛ 2
ТЕХНОЛОГІЇ РОЗРОБКИ ПЛАТФОРМ ДЛЯ ОНЛАЙН БІЗНЕСІВ

2.1 Аналіз існуючих рішень для створення онлайн-бізнесів

Сучасні платформи для створення онлайн-бізнесів формуються на основі різних технологічних підходів, серед яких ключовими є SaaS-рішення, конструктори сайтів та open-source системи з можливістю самостійного розгортання. Вибір тієї чи іншої моделі визначає рівень гнучкості, витрат і технічних вимог до користувача. Важливо підкреслити, що різні рішення не лише орієнтовані на різні сегменти ринку, а й суттєво відрізняються за архітектурою, принципами масштабування та можливостями кастомізації.

Shopify є однією з найбільш поширених SaaS-платформ, яка працює за моделлю «усе в одному». Вона пропонує готову хмарну інфраструктуру, інтегровану систему управління товарами та шлюзи для обробки платежів і маркетингові інструменти. Основна технічна перевага Shopify полягає в її централізованій архітектурі, де більшість налаштувань виконується через API або спеціалізовану мову Liquid. Такий підхід спрощує початкове налаштування та дає змогу швидко запускати магазини, проте він обмежує можливості індивідуальної кастомізації та створює залежність від екосистеми платформи.

Wix застосовує інший підхід, роблячи акцент на drag-and-drop інтерфейсі та візуальній побудові сторінок. З технічної точки зору, ця система орієнтована на швидке формування статичних або напівдинамічних сайтів, що добре підходить для лендінгів, невеликих вітрин та персональних проєктів. Проте у випадку високих навантажень або потреби в складних інтеграціях виникають проблеми: архітектура Wix є закритою, а доступ до серверної логіки практично відсутній. Це робить платформу малопридатною для розробки гнучких корпоративних рішень, хоча вона залишається привабливою завдяки низькому порогу входу.

WooCommerce як розширення WordPress базується на open-source підході і дозволяє створювати платформи електронної комерції з широким спектром інтеграцій. На відміну від SaaS-систем, WooCommerce не має централізованих обмежень: користувач може самостійно обирати сервер, налаштовувати базу даних, оптимізувати роботу бекенду та інтегрувати сторонні сервіси. Технічна складність полягає у необхідності постійного адміністрування — оновлення плагінів, захист від вразливостей і забезпечення продуктивності. Проте саме ця архітектурна відкритість робить WooCommerce одним із найбільш кастомізованих рішень на ринку.

Magento, на відміну від попередніх прикладів, створювалася як корпоративна система з акцентом на масштабованість та складні бізнес-процеси. Вона підтримує модульну архітектуру, багатомовність, мультивалютність і роботу з великими каталогами товарів. У технічному плані Magento вимагає значних серверних ресурсів, а також професійних навичок для налаштування та оптимізації. Для малих бізнесів це може бути надмірним, проте для великих компаній система забезпечує високий рівень кастомізації, контроль над даними та інтеграцію з ERP чи CRM-рішеннями.

Крім зазначених платформ, важливу нішу займають рішення на кшталт Squarespace, BigCommerce та OpenCart. Squarespace робить ставку на естетику та простоту, однак у технічному плані обмежується фіксованим набором функцій. BigCommerce має архітектуру, орієнтовану на інтеграцію з великими системами, і пропонує API-first підхід, проте його цінова політика робить його менш доступним для малого бізнесу. OpenCart, як і WooCommerce, належить до open-source рішень і надає широкі можливості для кастомізації, проте також вимагає від користувача технічної підготовки й регулярної підтримки.

Аналіз показує, що сучасні платформи поступово переходять від традиційних монолітних архітектур до більш гнучких моделей. Поширюється тренд headless-розробки, коли фронтенд і бекенд відокремлені один від одного, що забезпечує більшу свободу у виборі інструментів і кращу масштабованість. Дедалі більш популярним стає JAMstack-підхід, який

передбачає використання JavaScript, API та попередньо згенерованих статичних сторінок, що забезпечує швидкодію та безпеку. Однак більшість комерційних рішень ще не повністю інтегрували ці підходи, зберігаючи залежність від закритих екосистем та обмежених інструментів кастомізації.

Таким чином, огляд існуючих рішень демонструє, що хоча сучасний ринок пропонує широкий вибір платформ для створення онлайн-бізнесів, жодна з них не здатна одночасно забезпечити простоту використання, повну гнучкість кастомізації, масштабованість та економічну доступність. Це відкриває перспективи для нових підходів, які спиратимуться на відкриті технології та підтримуватимуть глибоку кастомізацію при збереженні простоти для кінцевого користувача

2.2 Ключові аспекти розробки веб-платформ

Створення веб-платформи не зводиться лише до написання коду чи побудови інтерфейсу, воно охоплює цілу систему взаємопов’язаних рішень, які визначають стабільність, продуктивність і конкурентоспроможність продукту. На перших етапах найбільший вплив має вибір архітектури. Традиційний монолітний підхід передбачає розробку платформи як єдиного цілого, де всі компоненти тісно пов’язані між собою. Це спрощує початкову реалізацію, але з часом робить підтримку та розширення значно складнішими, адже будь-яка зміна у коді може вплинути на роботу всієї системи. У відповідь на ці обмеження все більше проєктів переходять на мікросервісну архітектуру, коли функціонал поділений на окремі модулі, які взаємодіють через API. Такий підхід дозволяє незалежно оновлювати й масштабувати окремі частини системи, а використання контейнеризації та оркестрації, зокрема Docker і Kubernetes, забезпечують гнучке управління ресурсами та швидке розгортання нових версій [8].

Важливим критерієм успіху веб-платформи є користувацький досвід. Навіть найфункціональніший продукт не буде ефективним, якщо він складний

у використанні. Дизайн інтерфейсу має бути інтуїтивним і однаково зручним як на комп’ютері, так і на мобільному пристрої. В епоху мобільного інтернету більше половини користувачів заходять на платформи зі смартфонів, тому адаптивність є критичною вимогою. При цьому продуктивність і швидкість завантаження сторінок також відіграють ключову роль: статистика Google показує, що навіть затримка у дві секунди може знизити ймовірність продовження взаємодії користувача з ресурсом на третину. Це пояснює, чому сучасні веб-додатки активно використовують асинхронні технології, кешування та CDN, що дозволяють скоротити час відгуку.

Безпека є ще одним стовпом у процесі розробки. Сучасні веб-платформи працюють з чутливими даними, тому вони повинні бути стійкими до атак і відповідати міжнародним стандартам. Використання SSL/TLS, двофакторної аутентифікації, токенів доступу (OAuth 2.0, JWT), регулярне оновлення компонентів і моніторинг активності є обов’язковими практиками. Крім того, у багатьох країнах діють суворі нормативи щодо захисту персональних даних, такі як GDPR у ЄС, що вимагає від розробників впровадження додаткових заходів прозорості та контролю. Недостатня увага до безпеки не лише ставить під загрозу довіру користувачів, а й може призвести до значних фінансових втрат у випадку витоку даних.

Окрему роль у сучасних платформах відіграє інтеграція з іншими сервісами. Бізнес рідко працює ізольовано, і саме завдяки API-підходу платформи можуть легко підключати платіжні системи, CRM-системи, сервіси електронної пошти чи аналітичні інструменти. Вдалим прикладом є використання REST або GraphQL API, які дають можливість оптимізувати роботу з даними та зменшити навантаження на систему. API-first підхід також дозволяє швидко масштабувати продукт та підключати сторонні сервіси без глибоких змін у коді.

Ще одним ключовим чинником є масштабованість. Веб-платформа, що запускається з невеликою кількістю користувачів, може у майбутньому обслуговувати тисячі або навіть мільйони відвідувачів щодня. Для цього

необхідно передбачати можливість горизонтального та вертикального масштабування, використання хмарних сервісів, систем балансування навантаження та баз даних, здатних працювати з великим обсягом інформації. На практиці це означає впровадження кластерних рішень, реплікації даних, механізмів кешування і систем моніторингу, які виявляють проблеми ще до того, як вони стають критичними.

Важливою частиною життєвого циклу платформи є підтримка та розвиток. Тут на перший план виходять сучасні підходи DevOps, які поєднують процеси розробки та операційного управління. Використання CI/CD-практик забезпечує швидке й безпечне впровадження оновлень, що особливо важливо для платформ, які постійно розширюють функціонал. Регулярні оновлення дозволяють не лише покращувати продукт, а й підтримувати довіру користувачів. Водночас впровадження Agile-методологій у командній роботі забезпечує гнучкість і швидке реагування на зміни ринку.

Ще один важливий аспект — вимоги до доступності та багатомовності. Сучасні платформи повинні бути однаково зручними для користувачів із різними можливостями, що передбачає дотримання стандартів WCAG для доступності, а також забезпечення підтримки кількох мов і валют. Для міжнародних проєктів це стає обов’язковою умовою, яка дозволяє охоплювати ширшу аудиторію та формувати конкурентні переваги.

У підсумку можна сказати, що ключові аспекти розробки веб-платформ охоплюють архітектуру, користувацький досвід, безпеку, інтеграцію, масштабованість, підтримку й розвиток, а також додаткові вимоги на кшталт багатомовності та доступності. Всі ці елементи формують єдину систему, і нехтування хоча б одним із них знижує ефективність продукту. Комплексний підхід, заснований на сучасних технологіях і методологіях управління, дозволяє створювати платформи, здатні витримувати високі навантаження, забезпечувати безпеку й залишатися конкурентними у динамічному цифровому середовищі.

2.3 Огляд технологічного стеку MEAN та його можливостей

Технологічний стек MEAN, який включає MongoDB, Express.js, Angular та Node.js, став одним із найпоширеніших рішень для створення сучасних веб-платформ завдяки поєднанню гнучкості, швидкості розробки та єдності середовища. Його ключовою перевагою є використання мови JavaScript на всіх рівнях застосунку — від клієнтської частини до серверної логіки й роботи з базою даних. Це дозволяє розробникам уникати постійного перемикання між мовами програмування, знижує ризик помилок і прискорює створення прототипів.

MongoDB у цьому стеку виконує роль бази даних, яка відрізняється від класичних реляційних систем. Вона належить до класу NoSQL-рішень і працює з документами у форматі BSON (JSON-подібна структура). Це робить зберігання даних більш гнучким, адже структура може змінюватися динамічно, що є зручним для стартапів і проєктів, які перебувають у постійному розвитку. MongoDB підтримує горизонтальне масштабування та розподілене зберігання, що дозволяє обробляти великі обсяги даних із мінімальними затримками.

Express.js, як серверний фреймворк, забезпечує легке управління маршрутизацією, обробкою HTTP-запитів і middleware-функціями. На відміну від важких корпоративних рішень, Express пропонує мінімалістичну основу, яку можна доповнювати залежно від потреб. Це робить його гнучким інструментом для створення як невеликих REST API, так і складних серверних застосунків. Його популярність пояснюється простотою інтеграції з іншими бібліотеками, активною спільнотою та наявністю великої кількості готових рішень.

Angular, розроблений Google, використовується для побудови динамічних інтерфейсів. На відміну від простих бібліотек для роботи з UI, Angular є повноцінним фронтенд-фреймворком, що пропонує модульну структуру, двостороннє зв’язування даних, систему компонентів і потужні

інструменти для тестування. Він підходить для розробки як невеликих клієнтських додатків, так і масштабних корпоративних систем, де важлива стабільність, чітка структура та можливість повторного використання компонентів. Додатковою перевагою є постійна підтримка з боку Google та регулярні оновлення, що забезпечують сумісність із сучасними технологіями.

Node.js завершує цей стек як середовище виконання JavaScript поза браузером. Його неблокуюча, подієво-орієнтована архітектура дозволяє ефективно працювати з великою кількістю одночасних запитів, що робить його оптимальним для створення високонавантажених систем, таких як чати, онлайн-магазини або потокові сервіси. Саме завдяки Node.js розробники можуть поєднувати простоту роботи з JavaScript із продуктивністю, необхідною для масштабованих веб-додатків.

Важливо відзначити, що MEAN забезпечує не лише технічну цілісність, а й широку підтримку з боку спільноти. Тисячі розробників по всьому світу створюють бібліотеки та плагіни, що значно скорочує час розробки. Крім того, відкритість цього стеку робить його більш привабливим для компаній, які прагнуть уникнути залежності від закритих екосистем.

У практичному плані стек MEAN дозволяє реалізувати швидке прототипування, створювати адаптивні та масштабовані системи, інтегруватися з зовнішніми API та розгортати застосунки у хмарних середовищах. Це робить його придатним як для невеликих стартапів, так і для великих організацій, які потребують кастомізованих рішень. У контексті розробки платформи Spike саме використання MEAN відкриває можливості для реалізації модульної архітектури, впровадження кастомізації на рівні клієнтського інтерфейсу та забезпечення високої продуктивності на стороні сервера.

Таким чином, стек MEAN поєднує універсальність, простоту та продуктивність, а його поширення пояснюється не лише технічними перевагами, а й активною підтримкою спільноти та наявністю великої кількості прикладних бібліотек. Він забезпечує розробникам єдиний

інструментарій для роботи з даними, серверною логікою та інтерфейсами, що робить його одним із найефективніших підходів до створення сучасних кастомізованих онлайн-платформ.

2.4 Вимоги до створюваного програмного продукту та його технічні характеристики

Розробка платформи для створення кастомізованих онлайн-бізнесів передбачає чітке визначення вимог, що охоплюють функціональні, нефункціональні та технічні аспекти системи. На відміну від стандартних веб-додатків, платформа повинна мати універсальний характер, забезпечуючи можливість створення індивідуальних бізнес-рішень без потреби у переписуванні основного коду. Це вимагає особливого підходу до проектування архітектури, вибору технологій та організації процесів взаємодії між клієнтською, серверною та базовою частинами системи.

Функціональні вимоги до платформи визначають набір можливостей, які повинні бути реалізовані в межах основного продукту. Система має забезпечувати створення, редагування та налаштування власного онлайн-бізнесу з урахуванням специфіки користувача. Це передбачає можливість вибору шаблонів, керування контентом, підключення платіжних систем, управління замовленнями, ведення клієнтської бази та інтеграцію з зовнішніми API. Важливим елементом є наявність адміністративної панелі, що дозволяє користувачу контролювати усі процеси у межах створеного бізнесу, не маючи технічної підготовки.

Нефункціональні вимоги визначають якісні характеристики системи, серед яких особливе місце займають надійність, масштабованість, безпека та швидкодія. Платформа повинна стабільно працювати при великій кількості одночасних користувачів, забезпечувати швидкий відгук інтерфейсу та мінімізувати затримки при взаємодії з базою даних. Зважаючи на динамічність сучасного бізнесу, важливо передбачити можливість швидкого оновлення

функціоналу без зупинки роботи системи. Для цього доцільно реалізувати модульну архітектуру, яка дозволить додавати або змінювати окремі компоненти без впливу на решту підсистем.

З технічної точки зору, платформа Spike повинна бути побудована на основі технологічного стеку MEAN, який забезпечує єдність середовища розробки та гнучкість взаємодії між клієнтською і серверною частинами. Використання Node.js та Express.js дозволить створити продуктивний серверний шар із підтримкою REST API, який забезпечить швидку обробку запитів і зручну інтеграцію з фронтендом. Angular буде відповідати за побудову клієнтського інтерфейсу з адаптивним дизайном, що коректно відображається на всіх типах пристроїв, а MongoDB забезпечить динамічне управління даними без необхідності суворої схеми, що особливо важливо при створенні кастомізованих бізнес-рішень.

З точки зору архітектури, доцільно використовувати мікросервісний підхід із розмежуванням відповідальності між компонентами системи. Це дозволить підвищити масштабованість і спростить технічну підтримку. Серверна частина повинна бути побудована у вигляді окремих сервісів, що відповідають за різні підсистеми — автентифікацію користувачів, управління проєктами, інтеграцію з платіжними системами та аналітику. Така структура створює умови для незалежного оновлення компонентів і забезпечує безперервну роботу системи навіть при розгортанні нових модулів.

Особливу увагу слід приділити вимогам до безпеки. Платформа має підтримувати систему авторизації та автентифікації з використанням JSON Web Tokens (JWT), що дозволить захистити сесії користувачів. Усі дані повинні передаватися через захищений протокол HTTPS із застосуванням шифрування SSL/TLS. Для захисту від зовнішніх атак важливо реалізувати обмеження запитів, фільтрацію вхідних даних і механізми запобігання XSS- та CSRF-атакам. Крім того, система повинна забезпечувати резервне копіювання бази даних та відновлення у разі збоїв, що є необхідною умовою для стабільної роботи у середовищі з постійним потоком клієнтів.

З метою забезпечення масштабованості і стабільності роботи передбачається розгортання платформи у хмарному середовищі. Це дозволить гнучко розподіляти ресурси, автоматично балансувати навантаження та швидко масштабувати систему відповідно до кількості активних користувачів. Для цього можуть бути використані сервіси AWS, Google Cloud або DigitalOcean, які підтримують автоматичне горизонтальне масштабування, контейнеризацію та інтеграцію з CI/CD-середовищами.

З боку продуктивності платформа повинна забезпечувати мінімальний час відгуку при роботі з даними, що досягається шляхом використання асинхронних запитів, кешування та оптимізації запитів до MongoDB. Для аналітики ефективності системи необхідно реалізувати збір метрик у реальному часі — кількість запитів, середній час обробки, частоту помилок та навантаження на сервер. Це дозволить оперативно виявляти проблеми та покращувати продуктивність.

Також варто врахувати вимоги до інтерфейсу користувача. Оскільки система орієнтована на широке коло підприємців, важливо забезпечити зрозумілу логіку роботи, адаптивний дизайн і можливість кастомізації візуальних елементів. Інтерфейс повинен підтримувати багатомовність, а також функціональність drag-and-drop для налаштування структури сторінок і модулів бізнесу без програмування. Це сприятиме зниженню порогу входу для користувачів, які не мають технічної освіти.

Отже, технічні характеристики та вимоги до створюваного програмного продукту базуються на принципах надійності та масштабованості, безпеки і гнучкості. Реалізація системи Spike із використанням стеку MEAN дозволить створити універсальну платформу, здатну задовольнити потреби різних типів користувачів, від малих підприємців до компаній, що працюють на міжнародному ринку. Поєднання відкритих технологій, модульної архітектури та інтуїтивного інтерфейсу забезпечить платформі конкурентні переваги та високу ефективність у практичному застосуванні.

2.5 Інструменти та технології для реалізації кастомізованих рішень

Розроблення кастомізованих онлайн-платформ передбачає створення такої архітектури, де користувач може самостійно формувати бізнес-логіку, структуру сторінок та набір функцій без необхідності втручання у код. Це досягається через поєднання концепцій модульності, компонентності, шаблонізації та інтегрованих API. Згідно з дослідженнями Gartner (2023), понад 65 % сучасних бізнес-платформ включають елементи low-code або no-code кастомізації, що істотно скорочує час розробки та знижує технічний бар'єр входу для підприємців [9].

Ключовим принципом побудови таких систем є **модульна архітектура**, що дає змогу динамічно підключати, відключати або налаштовувати функціональні блоки без перезбирання всієї системи. Для цього використовуються підходи *plugin-based architecture* і *dynamic module loading*, описані у роботах Марка Річардса та Ніла Форда «Software Architecture: The Hard Parts» (O'Reilly, 2021) [10]. У межах платформи Spike цей принцип означає, що користувач може обирати, які саме модулі будуть активовані — наприклад, модуль оплати, аналітики чи керування клієнтами — і конфігурувати їх параметри через інтерфейс.

Другим важливим інструментом реалізації кастомізації є **API-орієнтована модель**, що дозволяє кожному компоненту взаємодіяти через стандартизовані інтерфейси. Підхід *API-first*, рекомендований Cloud Native Computing Foundation (2022), передбачає створення повнофункціонального бекенду, який може бути адаптований під різні фронтенд-середовища. У випадку Spike це означає, що кожен користувач, створюючи власний онлайн-бізнес, взаємодіє із сервером через уніфікований REST або GraphQL API, отримуючи можливість будувати унікальні бізнес-процеси без зміни основного коду.

Ще однією технологією, що визначає кастомізованість, є **система шаблонізації (template & theme engine)**. Вона дозволяє швидко створювати

індивідуальні інтерфейси на основі попередньо визначених шаблонів. Сучасні рішення — зокрема Handlebars.js та EJS — підтримують динамічну підстановку контенту, що дає змогу користувачу самостійно налаштовувати вигляд сторінок. У контексті Spike планується застосування Angular Dynamic Components, які забезпечують підвантаження шаблонів у реальному часі, що відповідає концепції low-code-редагування.

Важливою складовою кастомізації є **збереження користувацьких конфігурацій**. Для цього використовуються підходи на кшталт *User Configuration Storage*, коли кожен об’єкт інтерфейсу чи функціональний блок зберігає налаштування у форматі JSON-документів у базі MongoDB. Це дає змогу миттєво відтворювати користувацьке середовище після повторного входу або перенесення бізнес-моделі на інший домен.

Серед інструментів, які безпосередньо забезпечують кастомізованість у сучасних системах, варто відзначити бібліотеки **Drag-and-Drop інтерфейсів**. Зокрема, Angular CDK DragDrop або Interact.js дозволяють користувачам динамічно формувати макет сторінки, змінювати порядок елементів, редагувати блоки контенту. У платформі Spike така функціональність дає можливість будувати інтерфейс бізнесу інтуїтивно, без знання програмування, що узгоджується з концепцією no-code-підходу.

Не менш важливою є підтримка **конфігураційного розширення функціоналу** через систему плагінів. Для цього доцільно реалізувати спеціальний інтерфейс *Plugin API*, який дозволить розробникам сторонніх модулів інтегрувати власні рішення у платформу. Такий підхід активно використовується у сучасних CMS, наприклад WordPress та Strapi, і описаний у праці С. Девіса «Extensible Web Systems and Modular Integration Patterns» (IEEE Access, 2021) [11].

Окрему роль у побудові кастомізованих систем відіграють інструменти автоматизованого тестування та розгортання. Використання CI/CD-практик через GitHub Actions або Jenkins забезпечує стабільність під час оновлення та мінімізує ризики при впровадженні нових плагінів чи користувацьких

шаблонів. Автоматичні юніт-тести, побудовані на Jest та Cypress, гарантують коректність роботи після зміни параметрів конфігурації.

Питання безпеки також має безпосереднє відношення до кастомізації. Оскільки користувачі самостійно змінюють структуру своїх бізнес-модулів, система має вбудовані механізми контролю доступу (Role-Based Access Control — RBAC) та валідації даних. Реалізація цих механізмів через middleware Express.js і Angular Guards забезпечує ізольованість користувацьких даних та попереджає помилки, які можуть вплинути на роботу всієї платформи.

Таким чином, інструменти та технології, які використовуються для реалізації кастомізованих рішень, охоплюють широкий спектр підходів — від модульної архітектури та API-орієнтованих систем до drag-and-drop інтерфейсів і конфігураційного збереження налаштувань. Їхнє поєднання формує основу для створення платформи Spike, що надає користувачам можливість будувати власні онлайн-бізнеси без глибоких технічних знань, одночасно зберігаючи стабільність, безпеку й масштабованість системи.

РОЗДІЛ 3
РОЗРОБКА І ТЕСТУВАННЯ ПЛАТФОРМИ

3.1 Призначення платформи та її основні функції. Технічне завдання

Платформа Spike розробляється як багатофункціональне середовище для створення, конфігурації та управління онлайн-бізнесами різного типу. Її основне призначення полягає у спрощенні процесу цифрової трансформації малого та середнього бізнесу, а також у забезпеченні підприємців, маркетологів і консультантів інструментами, які дозволяють створювати власні веб-системи без глибоких технічних знань. Spike поєднує технологічну складність бекенду з простотою користувацької взаємодії, створюючи універсальну платформу, здатну охопити потреби від базового сайту-представництва до складного корпоративного рішення.

Платформа орієнтована передусім на малий і середній бізнес, який прагне автоматизувати процеси продажів, клієнтського обслуговування та аналітики без залучення IT-команди. Вона може використовуватися компаніями, що працюють у сфері електронної комерції, освіти, логістики, медіа, а також у сфері послуг. Spike однаково ефективно підходить як для підприємців, які хочуть створити інтернет-магазин чи платформу з бронювання послуг, так і для агенцій, які займаються розробкою кастомізованих веб-продуктів для клієнтів.

Ключовою особливістю Spike є її кастомізованість. Кожен користувач має можливість сформувати власну бізнес-модель — визначити тип проєкту, структуру сторінок, набір інтегрованих сервісів і зовнішніх модулів. Завдяки цьому Spike не обмежується класичними сценаріями використання, як це роблять традиційні CMS або конструктори сайтів. На відміну від них, Spike виступає платформою для **побудови бізнес-процесів**, а не лише інтерфейсів. Система дозволяє управляти каталогами товарів, приймати онлайн-платежі,

вести клієнтські бази, створювати аналітичні панелі та налаштовувати автоматичні повідомлення або звіти.

Особливу увагу в Spike приділено інтерфейсу користувача, який створюється з урахуванням принципів інтуїтивності та модульності. Усі елементи інтерфейсу реалізовані у вигляді компонентів, які користувач може вільно комбінувати — переміщати, змінювати їхній вигляд, додавати або видаляти блоки без програмування. Це дозволяє створювати унікальні вебрішення із власною структурою та стилем, використовуючи візуальний конструктор на основі технології drag-and-drop. Таким чином, Spike поєднує простоту користувацького досвіду з гнучкістю професійних інструментів веброзробки.

Платформа також виконує функцію **інтеграційного середовища**. Вона підтримує підключення зовнішніх сервісів через REST або GraphQL API, що дає можливість використовувати популярні платіжні системи, CRM, маркетингові інструменти та аналітичні рішення. Наприклад, користувач може інтегрувати Stripe чи PayPal для обробки платежів, підключити Google Analytics для відстеження поведінки клієнтів або використовувати SendGrid для автоматичної розсилки. Завдяки відкритій структурі Spike легко доповнюється новими модулями, а сторонні розробники можуть створювати власні розширення для конкретних бізнес-сценаріїв.

З практичного погляду Spike виконує роль «платформи як сервісу» (PaaS), орієнтованої на кастомні бізнес-рішення.Усі ці бізнес-моделі можуть бути побудовані всередині Spike без ручного програмування, лише шляхом вибору шаблонів і налаштування модулів.

З технічного боку Spike розробляється як система з модульною архітектурою, побудована на стеку **MEAN**. Node.js і Express.js реалізують серверну логіку та маршрутизацію запитів, Angular відповідає за клієнтську частину з високою інтерактивністю, а MongoDB забезпечує гнучку структуру збереження даних, дозволяючи користувачам створювати унікальні схеми даних для власних проєктів. Комунікація між клієнтом і сервером відбувається

через REST API, а для забезпечення масштабованості використовується контейнеризація на основі Docker. Це дає змогу розгортати окремі екземпляри бізнесів у хмарних середовищах із власними базами даних та конфігураціями.

Технічне завдання проєкту Spike передбачає реалізацію декількох критичних вимог. По-перше, система повинна підтримувати динамічну кастомізацію, тобто можливість зміни структури сторінок і компонентів без втручання у код. По-друге, необхідно забезпечити інтеграцію із зовнішніми сервісами через стандартизовані API. По-третє, платформа має бути масштабованою, щоб підтримувати як десятки, так і тисячі одночасних користувачів. Крім того, пріоритетним завданням є захист даних: усі облікові записи користувачів і транзакції мають бути захищені за допомогою JWT-токенів, шифрування SSL/TLS та системи багаторівневих ролей доступу.

Spike має також реалізувати функції аналітичної панелі, яка дозволить користувачам відстежувати ключові показники бізнесу: обсяг продажів, кількість клієнтів, середній чек, джерела трафіку тощо. Усі аналітичні дані зберігаються у базі MongoDB та візуалізуються за допомогою Chart.js, що дає змогу створювати звіти у реальному часі.

Таким чином, платформа Spike виступає комплексним рішенням для підприємців, організацій та розробників, які прагнуть швидко створювати кастомізовані онлайн-бізнеси без технічних обмежень. Її головна цінність полягає в тому, що вона поєднує простоту використання з потужною архітектурою, забезпечуючи гнучкість, безпеку та стабільність у будь-яких умовах. У результаті Spike стає інструментом, який не просто спрощує процес запуску онлайн-бізнесу, а формує новий підхід до цифрового підприємництва — гнучкий, динамічний і масштабований.

3.2 Етапи розробки платформи та його функціональні можливості

Розробка платформи Spike відбувалася у кілька послідовних етапів, кожен з яких впливав на подальшу структуру системи та формування її функціональних можливостей.

На початку був підготовлений первинний макет інтерфейсу у Figma. У ньому було визначено базове розташування елементів, форму сторінок, вигляд панелі керування та основні сценарії роботи користувача. Цей варіант дозволив оцінити загальний вигляд майбутньої платформи та сформувати початкове уявлення про структуру Spike.

Після створення першого макету розпочалося початкове верстання клієнтської частини. За допомогою HTML та SCSS були зібрані базові сторінки, створено адаптивну сітку, стилі основних елементів та початкові шаблони. На цьому етапі були виявлені недоліки первинного дизайну, через що було прийнято рішення виконати повний редизайн.

У Figma було створено оновлену версію макету, яка враховувала майбутні інтерактивні модулі, систему конфігурацій та можливість гнучкого керування сторінками користувача. Після цього було проведено повторне верстання всіх сторінок, вже з орієнтацією на подальшу інтеграцію з Angular та можливість динамічного оновлення змісту.

Після завершення інтерфейсної частини розпочалася робота над структурою даних і серверною логікою. Було визначено моделі даних у MongoDB, створено окремі колекції для бізнес-проєктів користувачів, а також реалізовано базові маршрути для REST API. На цьому ж етапі була створена система авторизації, контролери для роботи з даними та middleware для їхньої перевірки.

Наступним етапом стала реалізація основного функціоналу платформи. Було створено редактор сторінок, механізми збереження налаштувань, систему блоків, управління проєктами та початкові інтеграційні можливості. Усі ключові операції були протестовані під час проміжних перевірок, що дозволяло швидко усувати помилки.

Після реалізації основних модулів було проведено первинне тестування всієї системи. Перевірялась робота API, коректність взаємодії клієнтської та серверної частин, стабільність відображення інтерфейсу та обробка даних. На цьому етапі були виправлені дрібні помилки й уточнені окремі функції.

Далі відбулося повне тестування платформи, під час якого перевірялися різні сценарії використання, стабільність роботи окремих модулів та поведінка системи під навантаженням. Після виправлення знайдених недоліків було виконано фінальну перевірку.

Заключний етап завершив розробку: всі модулі були узгоджені, інтерфейс перевірений, а платформа Spike досягла готової форми та могла бути розгорнута для подальшого використання.

3.3 Серверна архітектура та управління базами даних

Серверна частина платформи Spike є основою всієї системи, оскільки саме на рівні бекенду відбувається опрацювання бізнес-логіки, комунікація між модулями, обробка запитів користувачів та взаємодія з базою даних. Архітектуру було сформовано таким чином, щоб вона не лише забезпечувала швидку реакцію на запити, а й залишалася змінною для подальшого розширення. Саме через це в Spike застосовується багатошарова структура, де кожен шар виконує окрему роль і не залежить безпосередньо від інших компонентів.

У розробці серверного ядра було важливо уникнути монолітної логіки, оскільки платформа орієнтована на різні типи бізнесів, а їхні вимоги можуть суттєво відрізнятися. Тому сервер організовано за принципами розподіленої архітектури, де функціональні блоки існують як окремі мікросервісні модулі. Це забезпечує незалежність кожного сегмента — модуль обробки замовлень не впливає на модуль авторизації, а інтеграційний блок не заважає роботі аналітичного. Такий підхід дозволяє змінювати або вдосконалювати окремі частини платформи, щоб не порушити її загальну роботу, що є критичним фактором для систем.

Комунікація між серверами відбувається через REST API, що забезпечує стандартизованість взаємодії. Кожна операція має окремий маршрут, а логіка запитів працює з сервісними класами, адже це дозволяє розділити

відповідальність і створити систему, в якій легше відстежувати виклики методів, виконувати логування та керувати помилками. Центральне місце займає API-шлюз, який приймає всі запити користувачів, а потім передає їх у відповідні мікросервіси. Така схема робить систему зручнішою для маршрутизації, оскільки користувач взаємодіє з єдиним входом, а сервер у фоновому режимі розподіляє навантаження між внутрішніми компонентами.

З огляду на специфіку платформи, що дозволяє користувачам формувати власні конфігурації бізнесу, важливо забезпечити коректну роботу з даними. Для цього Spike використовує MongoDB. Кожен бізнес-проєкт користувача може мати власні моделі документів: товари, клієнти, каталоги, замовлення, таблиці конфігурацій. MongoDB дозволяє уникнути жорстких схем і будувати структуру даних динамічно, що є критично важливим для кастомізованої платформи, де користувач може створювати нестандартні модулі.

Щоб забезпечити цілісність та узгодженість даних, у Spike реалізовано механізм ізоляції бізнес-проєктів, де дані різних користувачів зберігаються у окремих “блоках” або через логічні простори імен. Це не лише підвищує рівень безпеки, а й дозволяє оптимізувати роботу індексів, що пришвидшує пошук і фільтрацію інформації.

Важливою частиною архітектури є механізм кешування, який дозволяє розвантажити базу даних та зменшити час відповіді при повторних запитах. У Spike застосовується Redis як окремий високошвидкісний кеш-сервер, який зберігає попередньо оброблені дані — токени авторизації, попередньо сформовані сторінки, сесійні дані чи результати аналітичних запитів. Такий підхід не лише прискорює роботу, а й знижує навантаження на MongoDB на високих піках активності.

Серверна частина платформи побудована з урахуванням високих вимог до безпеки. Усі етапи обробки запитів проходять через middleware-рівень, що включає перевірку форматів і даних у тілі запиту, прав доступу та правильності токенів. Авторизація реалізована через JWT, а передані дані шифруються за допомогою SSL/TLS.

Завдяки комбінації мікросервісного підходу, документно-орієнтованих даних, кешування та централізованого API-шлюзу серверна частина Spike формує єдине технологічне ядро, здатне витримувати змінні навантаження і зберігати стабільність при масштабуванні. Така архітектура дозволяє впроваджувати нові функції без зупинки роботи системи, адаптувати платформу під нетипові бізнес-сценарії та забезпечувати високий рівень взаємодії між користувачем і даними. У результаті серверна частина виступає фундаментом Spike як гнучкої, багатофункціональної та надійної веб-платформи для створення кастомізованих онлайн-бізнесів.

3.4 Інтерфейс платформи та забезпечення зручності користування

Інтерфейс платформи Spike побудований таким чином, щоб користувач мав швидкий доступ до всіх основних інструментів, необхідних для створення та керування онлайн-бізнесами. Уся структура інтерфейсу поділена на ключові робочі області, причому кожна з них відповідає за виконання окремих функцій платформи. Такий підхід дозволяє користувачу інтуїтивно орієнтуватися в середовищі, швидко перемикатися між завданнями та працювати з необхідними модулями без додаткової технічної підготовки.

Головна сторінка платформи Spike (рис. 3.1-3.2) призначена для ознайомлення користувачів з основними можливостями системи. Вона містить інформаційні блоки, які коротко описують ключові переваги платформи та

дозволяють зрозуміти, які бізнес-задачі Spike може вирішувати.

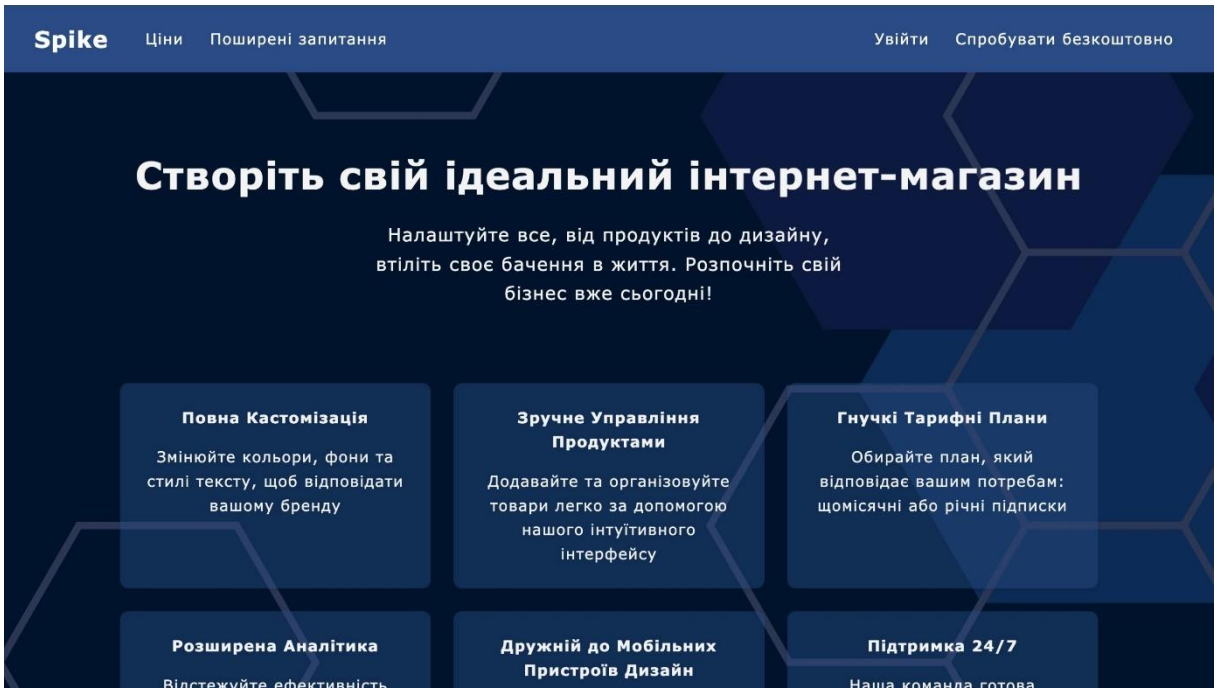


Рис. 3.1 – Головна сторінка

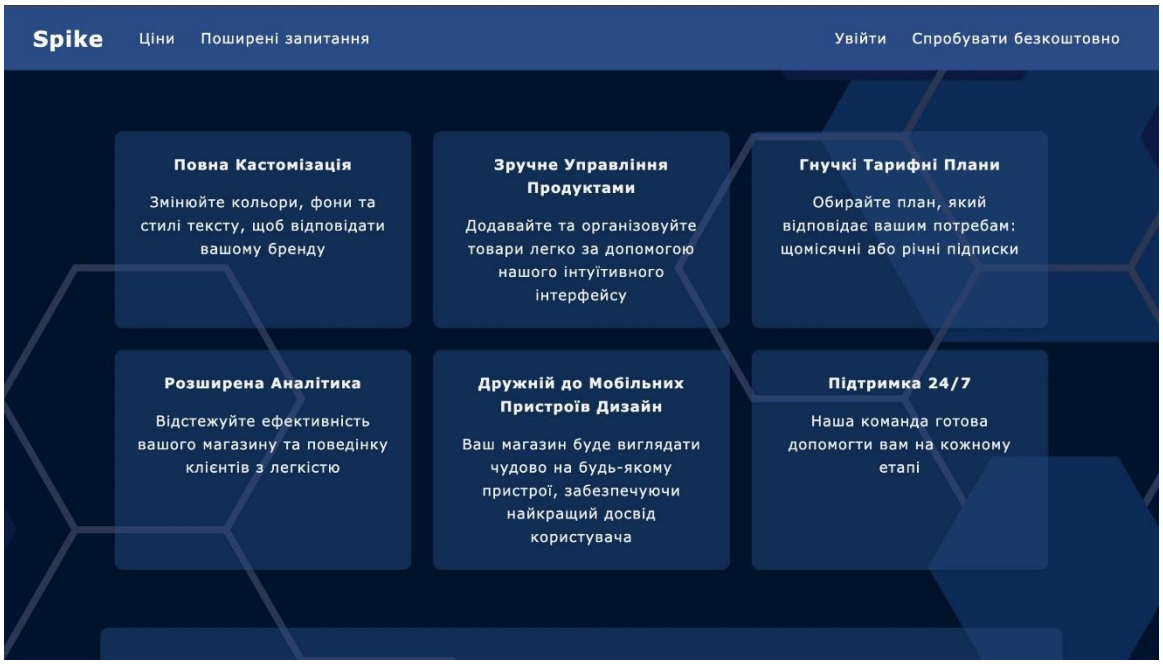


Рис 3.2 – Головна сторінка

На сторінці відображаються такі елементи:

- заголовок з основною пропозицією платформи;
- короткий опис можливостей Spike;

- інформаційні картки з переліком основних функцій, серед яких:
  – повна кастомізація зовнішнього вигляду,
  – управління продуктами,
  – гнучкі тарифні плани,
  – розширена аналітика,
  – адаптивний дизайн,
  – цілодобова підтримка;
- кнопки «Увійти» та «Спробувати безкоштовно»;
- навігаційне меню (посилання на ціни та розділ «Поширені запитання»).

Сторінка виконує ознайомчу функцію та допомагає користувачу дізнатися про можливості платформи до реєстрації чи входу в систему.

Сторінка перегляду товарів доступна кінцевим користувачам, які взаємодіють з онлайн-магазином, створеним у платформі Spike (рис. 3.3-3.4). Вона виконує роль каталогу та дозволяє покупцю ознайомитись з усім асортиментом. На сторінці розміщені такі елементи:

- список товарів у вигляді карточок;
- зображення кожного товару;
- назва та короткий опис;
- актуальна ціна;
- навігація або фільтри (за потреби конкретної реалізації магазину).

Сторінка забезпечує користувачу можливість швидко переглядати доступні товари та взаємодіяти з ними у форматі стандартного інтернет-магазину.

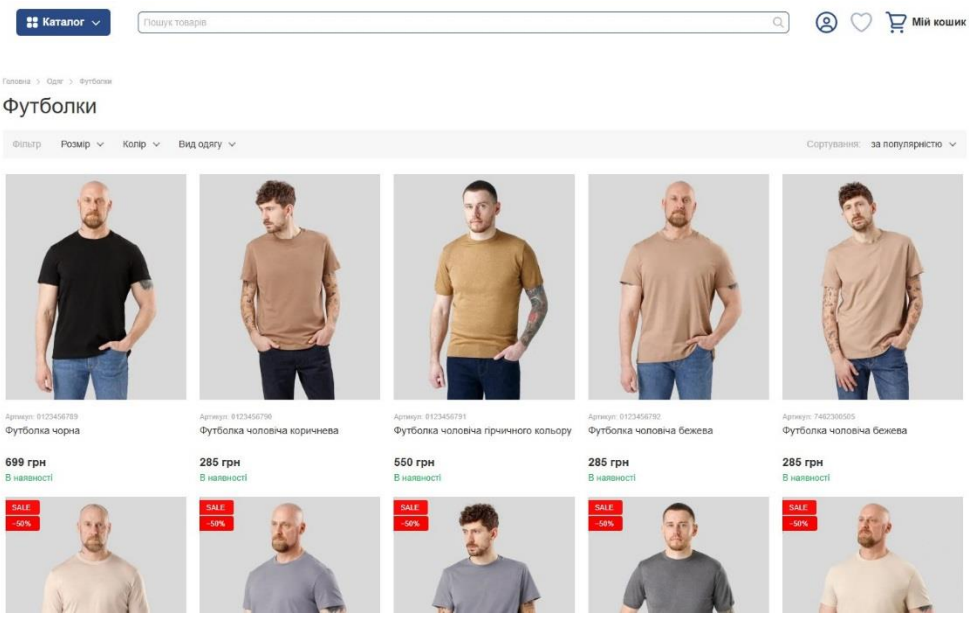
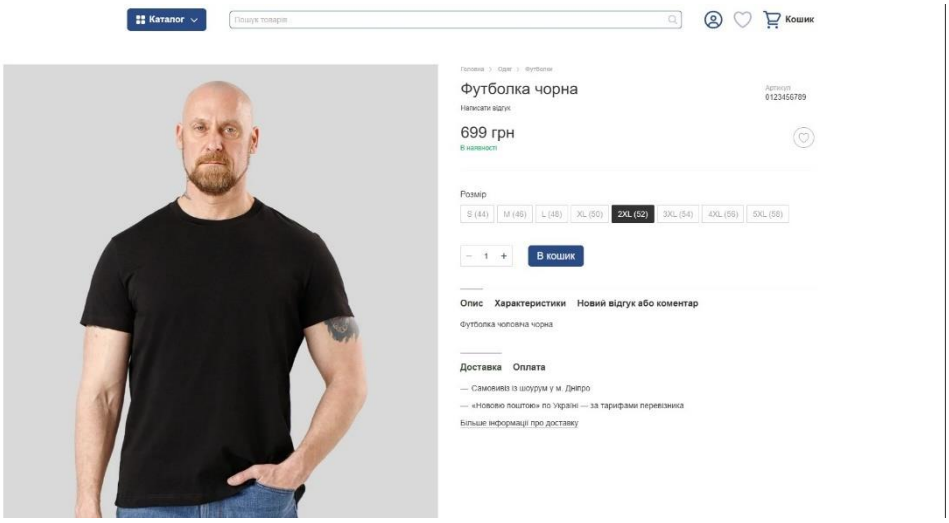


Рис 3.3-3.4 – Каталог



Вкладка «Замовлення» в адміністративній панелі (рис. 3.5) призначена для перегляду та керування всіма оформленими замовленнями у створеному магазині. Розділ дозволяє власнику бізнесу контролювати процес продажів та відстежувати дії клієнтів. Сторінка містить такі елементи:

- список усіх отриманих замовлень;
- ім’я та e-mail клієнта, який здійснив покупку;
- загальна сума;
- статус замовлення (наприклад: «нове», «в обробці», «виконано»)

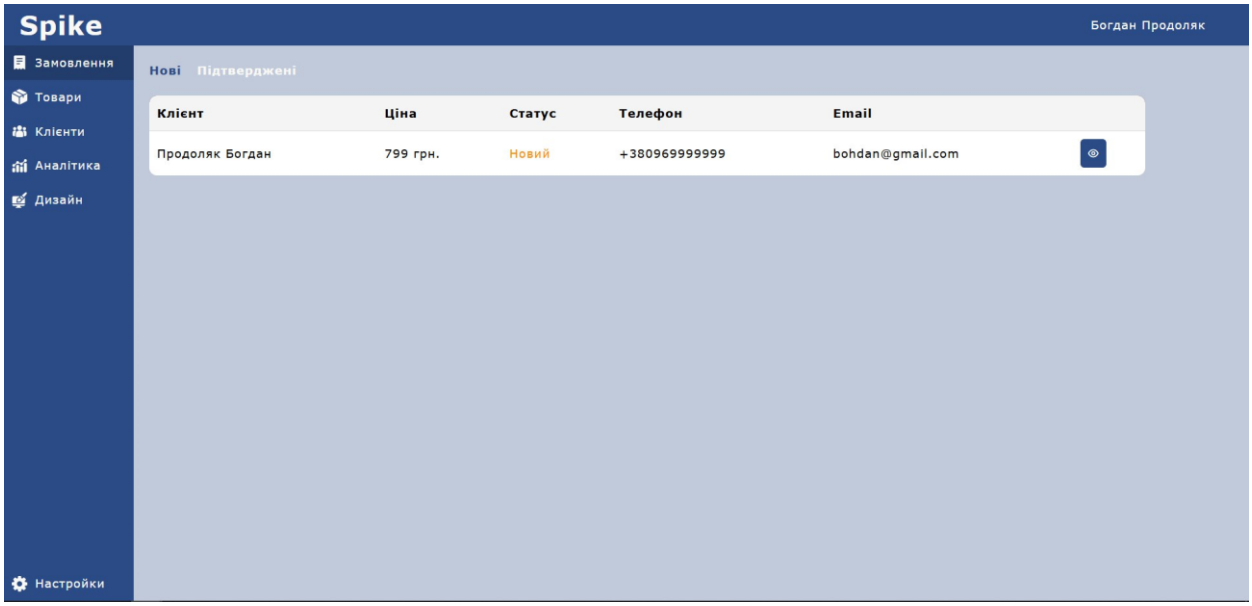


Рис 3.5 - Замовлення

Вкладка «Товари» в адміністративній панелі (рис. 3.6-3.7) призначена для керування асортиментом магазину, створеного у платформі Spike. Тут власник бізнесу може додавати нові позиції, редагувати існуючі та контролювати їх відображення на клієнтській сторінці. Розділ містить такі елементи:

- список усіх наявних товарів;
- назва кожної позиції;
- встановлена ціна;
- кнопка редагування товару;
- форма або кнопка для додавання нового товару;

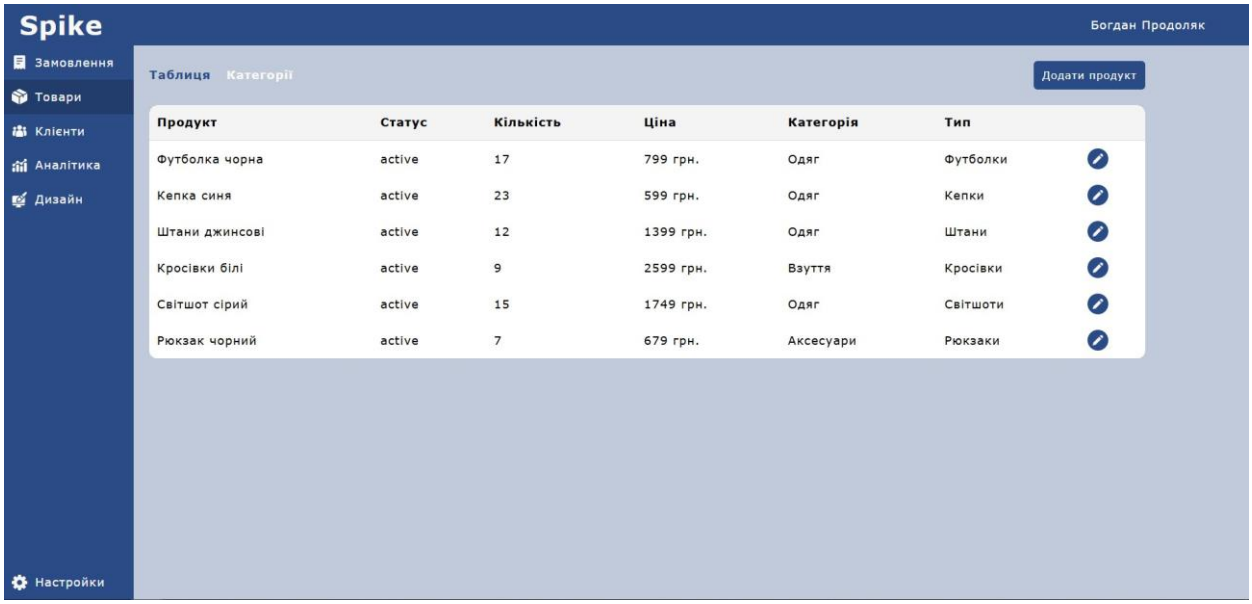


Рис 3.6 – Товари

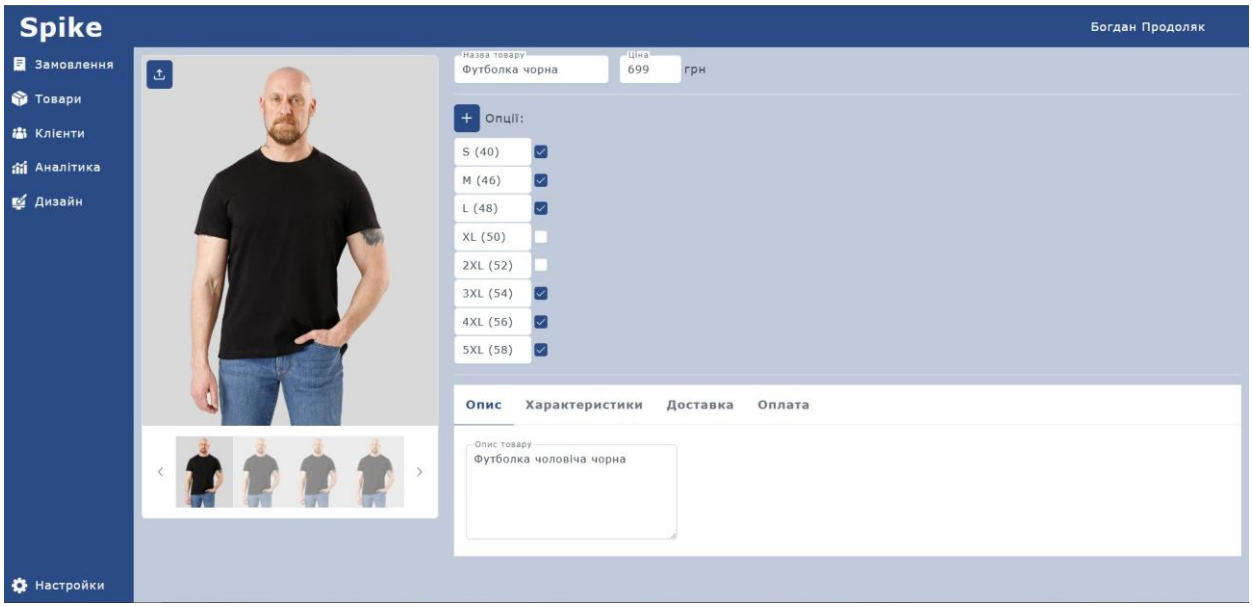


Рис 3.7 – Редагування товару

Вкладка «Клієнти» в адміністративній панелі (рис. 3.8) відображає інформацію про всіх користувачів, які зареєструвалися у створеному онлайн-магазині або здійснювали замовлення. Розділ дозволяє власнику бізнесу переглядати активність покупців та отримувати коротку статистику щодо їхньої взаємодії з магазином. На сторінці присутні такі елементи:

- список зареєстрованих клієнтів;
- ім’я або e-mail користувача;

- кількість оформлених замовлень;

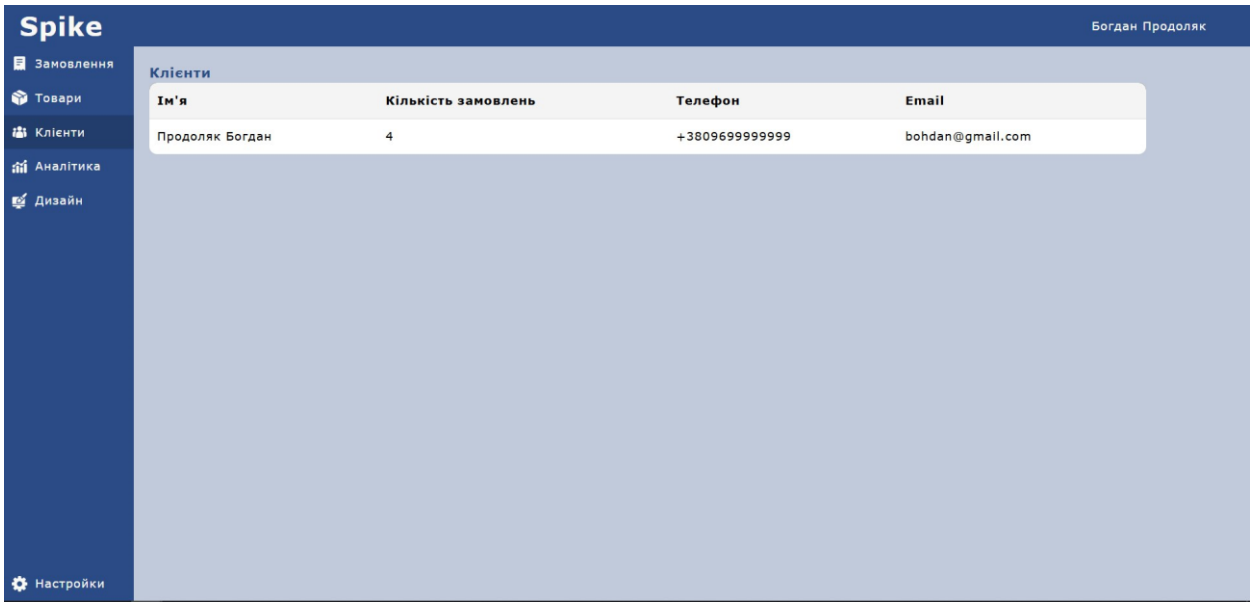


Рис 3.8 - Клієнти

Вкладка «Аналітика» в адміністративній панелі (рис. 3.9) надає власнику бізнесу узагальнену статистику щодо активності користувачів та здійснених замовлень. На цій сторінці відображаються ключові графіки та аналітичні дані, що допомагають оцінити ефективність роботи магазину.

Рис 3.9 - Аналітика

3.5 Методи тестування системи та аналіз її продуктивності

Після завершення основних етапів розробки було проведено комплексне тестування платформи Spike з метою перевірки коректності роботи функціоналу, стабільності серверної частини та продуктивності системи під різним навантаженням. Тестування охоплювало як функціональні, так і нефункціональні аспекти, що дозволило оцінити роботу платформи в умовах, максимально наближених до реальної експлуатації.

Функціональне тестування було спрямоване на перевірку основних можливостей платформи: створення проєктів користувачем, управління товарами та клієнтами, оформлення замовлень, роботу адміністративної панелі, а також коректність відображення інтерфейсу. Перевірялась взаємодія між Angular-компонентами та серверною частиною, правильність виконання REST API-запитів та відповідність отриманих даних заданим моделям.

Нефункціональне тестування охоплювало перевірку продуктивності, безпеки, стабільності роботи, а також зручності користування. Особливу увагу було приділено поведінці системи під навантаженням, зокрема часу відповіді API, швидкості рендерингу Angular-інтерфейсу та роботі MongoDB при обробці великої кількості запитів. Це допомогло виявити можливі “вузькі місця”, пов’язані зі швидкістю доступу до даних та кешуванням.

Для тестування продуктивності застосовувались сценарії з різним рівнем навантаження, включаючи одночасне опрацювання запитів на отримання товарів, оформлення замовлень та оновлення даних у адміністративній панелі.

Інтеграційне тестування перевіряло коректність взаємодії між фронтендом та бекендом, а також роботу модулів Spike у поєднанні з платіжними сервісами, API та аналітичними інструментами. Усі модулі були протестовані на правильність передачі даних, обробку винятків та відповідність структури відповідей очікуваним форматам.

Системне тестування включало повну перевірку роботи платформи: створення магазину, роботу каталогу, оформлення замовлень, управління товарами та клієнтами, формування аналітичних даних і роботу адміністративної панелі.

Окремим етапом було проведення користувацького тестування. Ряд користувачів перевіряли інтерфейс з точки зору зручності, простоти виконання базових операцій та логічності навігації. Отримані відгуки дозволили вдосконалити окремі елементи інтерфейсу та покращити взаємодію з адміністративною панеллю.

Після тестування було виконано серію оптимізацій, що включали корекцію запитів до бази даних, покращення кешування серверних відповідей, оптимізацію завантаження Angular-компонентів та усунення дрібних помилок. Моніторинг системи після впровадження дозволив відстежити стабільність її роботи та забезпечити своєчасне виправлення нових недоліків.

Проведені тести підтвердили, що платформа Spike демонструє стабільність роботи, коректність усіх основних функцій та достатній рівень продуктивності для подальшого масштабування й реального використання.

ВИСНОВКИ

У процесі підготовки магістерської кваліфікаційної роботи була досягнута поставлена мета та виконані всі завдання, визначені на початку дослідження. У рамках роботи розроблено веб-платформу Spike, призначену для створення кастомізованих онлайн-бізнесів на основі технологічного стеку MEAN. Платформа забезпечує можливість формування інтернет-магазинів, сервісів замовлень та інших цифрових рішень без необхідності глибокої технічної підготовки користувачів.

На початковому етапі було проведено аналіз ринку існуючих платформ та сервісів для створення онлайн-бізнесів. Це дозволило визначити основні недоліки популярних рішень, зокрема обмежену кастомізацію, високу вартість користування та залежність від закритих екосистем. Отримані результати стали підґрунтям для формування вимог до платформи Spike та визначення напрямів її функціонального розвитку.

Для реалізації програмного продукту був обраний стек технологій MEAN (MongoDB, Express.js, Angular, Node.js), який забезпечив єдність середовища розробки, гнучкість побудови архітектури та можливість масштабування системи. У ході роботи було створено структуру бази даних, розроблено серверну частину з використанням REST API, а також клієнтський інтерфейс із можливістю динамічного відображення даних та кастомізації сторінок.

Розробка платформи складалася з кількох послідовних етапів: створення макетів інтерфейсу, проєктування структури даних, формування клієнтської частини, реалізація основного функціоналу, інтеграція модулів та комплексне тестування. Особливу увагу було приділено стабільності роботи, коректності взаємодії між компонентами та зручності користування. Для перевірки якості було проведено функціональне та інтеграційне тестування, що дозволило усунути виявлені недоліки й оптимізувати продуктивність.

На основі проведеного дослідження можна зробити висновок, що платформа Spike є сучасним та перспективним інструментом для створення онлайн-бізнесів різної складності. Вона поєднує гнучкість у налаштуванні, зручність інтерфейсу та стабільність роботи, що робить її корисною як для початківців, так і для досвідчених користувачів. Подальший розвиток Spike може включати розширення функціональних можливостей, впровадження додаткових бізнес-модулів, інтеграцію з новими сервісами та оптимізацію інтерфейсу для різних категорій користувачів.

Результати роботи можуть бути використані у подальших дослідженнях, пов’язаних з розробкою веб-платформ, автоматизацією бізнес-процесів та впровадженням систем швидкого запуску цифрових продуктів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Statista. (2023). "Global e-commerce market size 2025." [Онлайн]. Доступно: https://www.statista.com

2. eMarketer. (2023). "B2C e-commerce sales worldwide." [Онлайн]. Доступно: https://www.emarketer.com

3. Forrester Research. (2023). "B2B e-commerce market forecast." [Онлайн]. Доступно: https://www.forrester.com

4. Harvard Business Review. (2021). "Why startups fail." [Онлайн]. Доступно: https://hbr.org

5. HubSpot. (2023). "Impact of integrated marketing strategies." [Онлайн]. Доступно: https://www.hubspot.com

6. McKinsey. (2022). "The value of financial analytics." [Онлайн]. Доступно: https://www.mckinsey.com

7. Accenture. (2023). "Cloud computing trends." [Онлайн]. Доступно: https://www.gartner.com

8. Kubernetes Documentation. *Production-Grade Container Orchestration*. – https://kubernetes.io/docs/home/

9. Gartner Research. *Low-Code Development Technologies Report 2023*. – https://www.gartner.com/en/documents/

10. Richards M., Ford N. *Software Architecture: The Hard Parts.* O'Reilly Media, 2021.

11. Davis S. *Extensible Web Systems and Modular Integration Patterns.* IEEE Access, 2021. – https://ieeexplore.ieee.org/document/9511014/

ДОДАТОК А

Важливі техніко-економічні показники платформи

Розробка веб-платформи Spike потребує оцінки її економічної доцільності та визначення того, який практичний ефект вона здатна забезпечити користувачам. Система створюється як інструмент для швидкого формування онлайн-бізнесів, що дозволяє зменшити витрати на розробку, прискорити запуск проєктів та підвищити ефективність роботи підприємців і малих компаній.

Однією з ключових переваг Spike є зниження витрат на створення веб-рішень. Використання готових модулів і вбудованих бізнес-функцій дозволяє уникнути повного циклу індивідуальної розробки, що скорочує фінансові витрати та час впровадження в декілька разів. Платформа дає можливість будувати інтернет-магазини, сервіси замовлень чи освітні платформи без додаткової технічної підтримки, що робить її економічно вигідною для малого та середнього бізнесу.

Spike також забезпечує автоматизацію ключових операцій, таких як управління контентом, інтеграція з платіжними системами, обробка замовлень та збір аналітики. Це зменшує операційні витрати та скорочує потребу в додатковому персоналі. Завдяки модульності та інтеграціям користувач отримує можливість ефективно керувати бізнес-процесами у межах однієї системи.

Важливим економічним аспектом є розширення ринку користувачів. Платформа орієнтована не лише на розробників, але й на підприємців без технічної підготовки. Завдяки цьому Spike може використовуватися у різних видах діяльності, що дає потенціал для монетизації через підписки, партнерські інтеграції та додаткові модулі.

Для визначення економічного ефекту в рамках дипломного проєкту необхідно розрахувати:

– капітальні та інвестиційні витрати;

- собівартість програмного продукту;
- можливі доходи від впровадження;
- чистий прибуток та термін окупності;
- коефіцієнт економічної ефективності.

Головна мета планування процесу розробки – визначення необхідних ресурсів на всіх його етапах, їх рівень залежить від обраного типу застосунку, його складності, особливостей дизайну тощо. Такі етапи наведено у таблиці Б.1.

Таблиця Б.1

Перелік етапів та робіт по розробці платформи

Найменування		**Вид роботи**		**Виконавець, посада**
стадії	**етапи**	**шифр**	**зміст роботи**	
1	**2**	**3**	**4**	**5**
1 Підготовча стадія	1.1 Вивчення стану питання	1.1.1	Дослідження проблеми	
		1.1.2	Вивчення та аналіз аналогічних розробок	
		1.1.3	Економічне обґрунтування доцільності виконання проекту	
	1.2 Розробка технічного завдання (ТЗ)	1.2.1	Складання плану та розрахунок розробки	

1	2	3	4	5
2 Технічна пропозиція	2.1 Аналіз ТЗ та техніко-економічне обгрунтування проекту	2.1.1	Доведення техніко-економічного обгрунтування	
3 Теоретична розробка	3.1 Теоретичне вивчення задачі	3.1.1	Визначення переліку технологій, які використовуватимуться при розробці та мови програмування	
		3.1.2	Розробка алгоритмів роботи програми на високому рівні	
		3.1.3	Розробка структури програмного забезпечення та схеми взаємодії її компонентів	
4 Практична реалізація	4.1 Розробка дизайну 4.2 Розробка структури сайту	4.1.2	Визначення шаблону дизайну	
		4.2.1	Розробка ядра сайту	
	4.3 Розробка бази даних	4.2.2	Система управління сайтом	
		4.3.1	Визначення системи управління базами даних	
		4.3.2	Розробка структури бази даних	
5. Заключна стадія	5.1 Ознайомлення зацікавлених осіб з проектом	5.1.1	Підготовка презентації	
		5.1.2	Демонстрація системи	
		5.1.3	Навчання роботи з системою	

На основі даного переліку визначається кількість виконавців, тривалість виконання робіт в днях, та рівень оплати праці виконавців по видам робіт, що виконуються. Такі дані приведені у таблиці 5.2.

Таблиця Б.2

Зведені результати тривалості та трудомісткості розробки та реалізації проекту

Шифр роботи	Найменування роботи	Виконавець, посада, спеціальність	Кількість виконавців	Тривалість виконання роботи, дні	Денна тарифна ставка, грн	Оплата праці, грн
1.1.1 - 1.2.1	Дослідження проблеми	Інженер-програміст	1	2	320	640
4.1.1 - 4.3.2	Практична реалізація	Інженер-програміст	1	6	320	1920
5.1.1 - 5.1.3	Заключна стадія	Інженер-програміст	1	1	320	320

До розробки застосунку залучено одного інженера-програміста, зайнятість якого становить 18 днів та керівника практики від підприємства. Розрахунок місячної заробітної плати програміста та керівника виконується згідно формули:

Мз=(Мпз* Крг)*Рд , де

Мпз-погодинна заробітна плата ;

Крг – кількість робочих годин в день (8 год);

Рд – кількість робочих днів;

Мз – заробітна плата в місяць.

Розраховуємо мінімальну місячну зарплату програміста та керівника.

Дані розрахунків заносимо в таблицю:

Таблиця Б.3

Витрати на заробітну плату

Найменування посади	**Місячний посадовий оклад, грн**	**Оплата за робочий день, грн**	**Кількість днів роботи**	**Витрати на заробітну плату, грн**
Керівник	4800	240	9	2160
Інженер-програміст	6400	320	18	5760

Разом: 7920.

Витрати на науково-дослідницьку роботу по розробці програмних засобів та апаратури, щодо даного дипломного проекту, включають наступні елементи:

— витрати на основну заробітну плату виконавців.

Це сума заробітної плати за кожний етап робіт і вона наведена у таблиці Б.2

ЗПосновна= 7920

— додаткова заробітна плата. Її можна обчислити за формулою:

ЗПдодаткова=0,12*ЗПосновна (Б.1)

ЗПдодаткова= 950,4

Таким чином, загальний фонд оплати праці, що обчислюється за формулою:

ФОП=ЗПосновна+ЗПдодаткова (Б.2)

ФОП=8870,4

обов’язкові відрахування на заробітну плату (ЄСВ – єдиний соціальний внесок – 22%). Таким чином обов’язкові відрахування складають:

ЄСВ = ФОП * 0,22 = 1951,5

накладні витрати розраховуємо за формулою:

НВ=0,2*ЗПосновна. (Б.3)

НВ=1584