МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ ФАХОВИЙ БІЗНЕС-КОЛЕДЖ
Циклова комісія комп’ютерної інженерії та інформаційних технологій

КВАЛІФІКАЦІЙНА РОБОТА

На тему

ВЕБ-САЙТ ОНЛАЙН ЗАПИСУ КЛІЄНТІВ ДЛЯ САЛОНУ КРАСИ

Виконала: студентка групи 1П-22

Спеціальності

121 Інженерія програмного забезпечення

Аліна СИНЬОГУБ

Керівник:
Дмитро ПОДОРОШКО

Черкаси 2026

ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ ФАХОВИЙ БІЗНЕС-КОЛЕДЖ

Циклова комісія комп'ютерної інженерії та інформаційних технологій

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач циклової комісії КІ та ІТ

Дмитро ПОДОРОШКО
(підпис)
«____» _____ 2026 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Синьогуб Аліні Романівні

1. Тема кваліфікаційної роботи Веб-сайт онлайн запису клієнтів для салону краси

Керівник роботи Подорошко Дмитро Ігорович, викладач першої категорії

затверджені наказом закладу фахової передвищої освіти від «6» 10 2025 року № 84/1-у.

2. Строк подання студентом кваліфікаційної роботи 08.06.2026

3. **Вихідні дані до кваліфікаційної** роботи мова програмування Python, фреймворк Django, архітектурний шаблон MVT, база даних SQLite для локального тестування та PostgreSQL для production, кеш-сервер Redis, WSGI-сервер Gunicorn, засоби контейнеризації Docker та Docker Compose, механізм перевірки працездатності /health/.
4. **Зміст кваліфікаційної роботи** аналіз предметної області та постановка завдання; огляд і порівняльний аналіз існуючих програмних рішень; формування вимог до веб-орієнтованої системи; концептуальне моделювання системи; попередня архітектура та проєктування системи; реалізація програмного продукту; тестування, розгортання та супровід програмного продукту.
5. **Дата видачі завдання 15.09.2025 р.**

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Терміни виконання етапів	Примітка про виконання
1	Вступ	14.10.2025	
2	Розділ 1.	09.12.2025	
3	Розділ 2.	10.03.2026	
4	Розділ 3.	28.04.2026	
5	Висновки	12.05.2026	
6	Оформлення кваліфікаційної роботи (чистовий варіант)	26.05.2026	
7	Перевірка кваліфікаційної роботи на наявність ознак плагіату (за 10 днів до захисту)	02.06.2026	
8	Подання кваліфікаційної роботи на затвердження завідувачу ЦК (за 7 днів до захисту)	10.06.2026	

Студент _____ **Аліна СИНЬОГУБ**
(підпис)

Керівник роботи _____ **Дмитро ПОДОРОШКО**
(підпис)

АНОТАЦІЯ

Кваліфікаційна робота присвячена дослідженню, проєктуванню та розробці веборієнтованої інформаційної системи онлайн-бронювання послуг у б’юті-сфері.

У першому розділі проаналізовано сучасний стан цифровізації сфери послуг в Україні, визначено проблеми організації клієнтських записів у ручному режимі та проведено порівняльний аналіз існуючих систем автоматизації. На основі отриманих результатів сформовано функціональні та нефункціональні вимоги до розроблюваної системи.

У другому розділі виконано проєктування системи із застосуванням UML-діаграм, розроблено структуру бази даних та архітектуру вебзастосунку. Описано реалізацію системи «Beauty Time» на основі мови програмування Python, фреймворку Django та архітектурного шаблону MVT. Реалізовано механізми автоматичної перевірки доступності часових слотів і запобігання конфліктам під час бронювання.

У третьому розділі наведено результати функціонального, інтеграційного та навантажувального тестування розробленої системи. Розглянуто питання розгортання, супроводу та забезпечення стабільної роботи програмного продукту.

У висновках узагальнено результати дослідження, підтверджено практичну цінність створеної інформаційної системи та визначено перспективи її подальшого розвитку.

Ключові слова: Django, Python, веборієнтована система, онлайн-бронювання, автоматизація сфери послуг, база даних, MVT-архітектура, тестування, Docker.

ANNOTATION

The qualification work is devoted to the research, design, and development of a web-based information system for online booking services in the beauty industry.

The first chapter analyzes the current state of digitalization in the service sector in Ukraine, identifies the problems associated with manual client appointment management, and provides a comparative analysis of existing automation systems. Based on the results obtained, functional and non-functional requirements for the developed system are defined.

The second chapter presents the system design using UML diagrams, the database structure, and the architecture of the web application. The implementation of the “Beauty Time” system based on the Python programming language, the Django framework, and the MVT architectural pattern is described. Mechanisms for automatic availability checking of time slots and prevention of booking conflicts are implemented.

The third chapter contains the results of functional, integration, and load testing of the developed system. It also discusses deployment, maintenance, and approaches to ensuring stable system operation.

The conclusions summarize the results of the research, confirm the practical value of the developed information system, and outline prospects for its further development.

Keywords: Django, Python, web-based information system, online booking, service automation, database, MVT architecture, testing, Docker.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

БД	База даних
КСЗІ	Комплексна система захисту інформації
НСЗУ	Національна служба здоров'я України
API	Application Programming Interface
CI/CD	Continuous Integration / Continuous Deployment
CRM	Customer Relationship Management
CSRF	Cross-Site Request Forgery
HSTS	HTTP Strict Transport Security
HTTP/HTTPS	HyperText Transfer Protocol / HyperText Transfer Protocol Secure
JSON	JavaScript Object Notation
MVT	Model-View-Template
ORM	Object-Relational Mapping
PCI	Payment Card Industry
POS	Point of Sale
SaaS	Software as a Service
SPA	Single Page Application
UX	User Experience
VPS	Virtual Private Server
WSGI	Web Server Gateway Interface

ВСТУП

Актуальність теми. Процеси глобальної цифровізації охоплюють усі ланки сучасної економіки, висуваючи нові вимоги до ефективності управління бізнесом. У сегменті надання побутових послуг, зокрема в індустрії краси, рівень конкуренції вимагає впровадження високотехнологічних рішень для взаємодії з клієнтами. На сьогодні малий бізнес стикається з критичною потребою в автоматизації рутинних процесів, серед яких ключовим є онлайн-запис на послуги. Традиційні методи адміністрування, що базуються на паперових журналах або телефонній комунікації, не здатні забезпечити цілодобову доступність сервісу та оперативність обробки даних.

Розробка веб-орієнтованої інформаційної системи для салону краси «Beauty Time» дозволяє вирішити низку стратегічних завдань: мінімізувати вплив людського фактору, забезпечити прозорість формування графіку майстрів та підвищити лояльність клієнтів через надання зручного інтерфейсу для самостійного бронювання. Використання сучасних фреймворків для створення таких систем є актуальним з точки зору забезпечення кібербезпеки та масштабованості програмного продукту [11].

Ступінь дослідження теми. Теоретичні засади проектування інформаційних систем та архітектурних шаблонів досліджувалися у працях провідних фахівців з програмної інженерії, таких як М. Фаулер та М. Лутц [9]. Проте швидкий розвиток веб-технологій та поява нових версій інструментарію, зокрема фреймворку Django, створюють необхідність у дослідженні прикладних аспектів реалізації архітектури Model-View-Template (MVT) для специфічних потреб малого бізнесу. Питання інтеграції реляційних баз даних та забезпечення захисту персональних даних користувачів у веб-середовищі залишаються одними з найбільш дискусійних у професійній спільноті розробників.

Об’єкт дослідження — система надання послуг онлайн-запису та управління клієнтською базою у сфері побутового обслуговування з використанням сучасних засобів автоматизації.

Предмет дослідження — методи, технології та інструментарій розробки динамічних веб-додатків на основі архітектурного шаблону MVT з використанням мови програмування Python та фреймворку Django.

Мета дослідження — теоретичне обґрунтування, проектування та програмна реалізація веб-орієнтованої інформаційної системи для автоматизації процесів онлайн-бронювання на прикладі салону краси «Beauty Time», що спрямована на підвищення ефективності бізнес-процесів та покращення якості сервісу.

Завдання дослідження:

— проаналізувати існуючі технологічні підходи до автоматизації сфери послуг та обґрунтувати вибір стеку технологій (Python/Django);

— сформулювати технічні вимоги до системи, включаючи функціональні можливості користувачів та адміністраторів;

— спроектувати архітектурну модель бази даних для ефективного збереження та маніпулювання даними про записи, послуги та клієнтів;

— реалізувати серверний компонент системи (Backend) на основі шаблону MVT, забезпечивши логічне розділення даних та представлення;

— розробити адаптивний інтерфейс користувача (Frontend), використовуючи сучасні методи верстки та стилізації;

— впровадити модулі автентифікації та розмежування прав доступу для забезпечення конфіденційності інформації;

— провести комплексне тестування розробленого програмного забезпечення на предмет стійкості до помилок та безпеки.

Методи дослідження. У роботі використано методи системного аналізу для вивчення предметної області, методи об'єктно-орієнтованого проектування для створення архітектури системи, а також методи емпіричного тестування для перевірки працездатності програмного продукту.

Практична цінність. Отримані результати мають безпосереднє практичне спрямування та можуть бути впроваджені в операційну діяльність реального суб’єкта господарювання. Розроблена система дозволяє автоматизувати збір статистичних даних, зменшити навантаження на персонал та надати клієнтам

сучасний інструмент взаємодії. Програмний код системи спроектований з урахуванням можливості подальшого масштабування та додавання нових функціональних модулів (наприклад, системи онлайн-оплати або інтеграції з месенджерами).

Очікувані результати. Завершення дослідження передбачає створення повнофункціонального прототипу веб-орієнтованої системи, що відповідає сучасним стандартам безпеки (CSRF, SQL-injection protection) та забезпечує стабільну роботу в умовах багатокористувацького доступу.

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Аналіз предметної області

Сучасний стан ринку побутових послуг в Україні характеризується інтенсивним впровадженням інформаційних технологій у всі сфери взаємодії з клієнтами. Предметною областю дослідження є процеси операційного управління та клієнтського сервісу в салоні краси «Beauty Time». Специфіка даної галузі полягає у високій частоті транзакцій (записів) та необхідності точного синхронізування часових ресурсів майстрів із запитами споживачів.

Аналіз поточної моделі функціонування об’єкта автоматизації дозволяє виділити ряд критичних проблем, що гальмують розвиток бізнесу. Основним методом реєстрації клієнтів на сьогодні залишається вербальна комунікація через телефон або месенджери, що зумовлює наступні негативні чинники:

— Низька конверсія запитів у неробочий час. За даними дослідження Phorest Salon Software, проведеного на основі аналізу бронювань у понад 5 000 салонах і спа, лише 54% записів здійснюються в робочі години, тоді як 46% поза ними. Це свідчить про те, що відсутність онлайн-запису в неробочий час може призводити до втрати частини потенційних клієнтів [7].

— Ризик виникнення конфліктних ситуацій (Overbooking). У разі паралельного надходження запитів з різних каналів (телефон, Instagram, особистий візит) адміністратор не завжди встигає оперативно оновити паперовий журнал. Це призводить до накладання записів на один і той самий час до одного майстра, що негативно впливає на репутацію закладу.

— Відсутність інструментів для аналітики. Ведення записів у ручному режимі унеможливлює автоматичне формування звітів про завантаженість персоналу, популярність окремих послуг (наприклад, манікюр чи перукарські послуги) та частоту візитів постійних клієнтів.

З точки зору програмної інженерії, об'єкт автоматизації представляє собою складну динамічну систему з багатьма актантами. Процес надання послуги складається з декількох етапів: ідентифікація клієнта, вибір специфічної послуги з

каталогу, перевірка доступності майстра у базі даних та фіксація часового слоту. Кожен із цих етапів потребує чіткої алгоритмізації для виключення помилок, спричинених людським фактором. Кожен із цих етапів потребує чіткої алгоритмізації для виключення помилок, спричинених людським фактором (див. Рис.1.1).

Рисунок 1.1 – Процес реєстрації запису за поточною моделлю «AS-IS»

Важливим аспектом аналізу є поведінкові особливості цільової аудиторії салону «Beauty Time». Сучасний споживач надає перевагу самообслуговуванню (self-service) через вебінтерфейси, оскільки це економить час та дозволяє візуально оцінити доступні варіанти запису. Отже, розробка веб-орієнтованої системи є не просто технічним покращенням, а стратегічною необхідністю для виживання бізнесу в умовах діджиталізації.

Додатково слід врахувати вимоги до безпеки зберігання персональних даних. У процесі реєстрації клієнти надають контактну інформацію (номер телефону, ім'я), яка згідно з чинним законодавством України потребує належного рівня захисту. Впровадження системи на базі надійного фреймворку дозволить

реалізувати шифрування та безпечне зберігання цих даних у реляційній базі даних SQLite, що є значно надійнішим за фізичні носії інформації.

Впровадження веборієнтованої системи дозволяє автоматизувати процес реєстрації клієнтів та усунути основні недоліки існуючої моделі (див. рисунок 1.2).

Рисунок 1.2 – Процес реєстрації запису за цільовою моделлю «TO-BE»

Автоматична перевірка вільного слота в базі даних SQLite дозволяє миттєво підтверджувати запис, забезпечуючи повну відсутність овербукінгу. Такий підхід не лише розвантажує персонал, а й гарантує клієнтам актуальність наданої інформації в будь-який час доби.

Підсумовуючи аналіз предметної області, можна стверджувати, що створення інформаційної системи для «Beauty Time» дозволить централізувати всі потоки даних, автоматизувати рутинні операції адміністратора та створити комфортне цифрове середовище для клієнтів. Це закладає фундамент для подальшого проектування бази даних та розробки програмних модулів системи.

1.2 Огляд та порівняльний аналіз існуючих програмних рішень

Для розробки ефективної архітектури вебзастосунку «Beauty Time» було проведено порівняльний аналіз трьох програмних продуктів. Вибір аналогів зумовлений необхідністю дослідження різних підходів до реалізації систем онлайн-бронювання: глобального агрегатора, спеціалізованої медичної системи та локального брендованого рішення.

1.Fresha (глобальна SaaS-платформа)

Fresha є великою міжнародною платформою для онлайн-бронювання та управління бізнесом у сфері краси й wellness. На офіційних сторінках компанія вказує, що платформі довіряють понад 120 000 бізнесів, а в мобільному застосунку для бізнесу зазначено понад 18 млн записів щомісяця та понад 450 000 фахівців, які працюють через сервіс. Fresha підтримує календар записів, онлайн-бронювання, передоплату, POS-функції, клієнтські профілі, автоматизовані повідомлення та інтеграції з Google, Instagram і власним маркетплейсом [1] .

Водночас Fresha залишається платформним рішенням із наперед визначеною логікою сервісу. Для нових клієнтів, які приходять через маркетплейс, застосовується комісія, а розширена підтримка надається як окрема платна опція [2]. Тому для локального салону з потребою у виразній брендованій айдентиці така система є зручною, але не максимально гнучкою.

Інтерфейс вибору послуг у системі Fresha наведено на рис. 1.3.

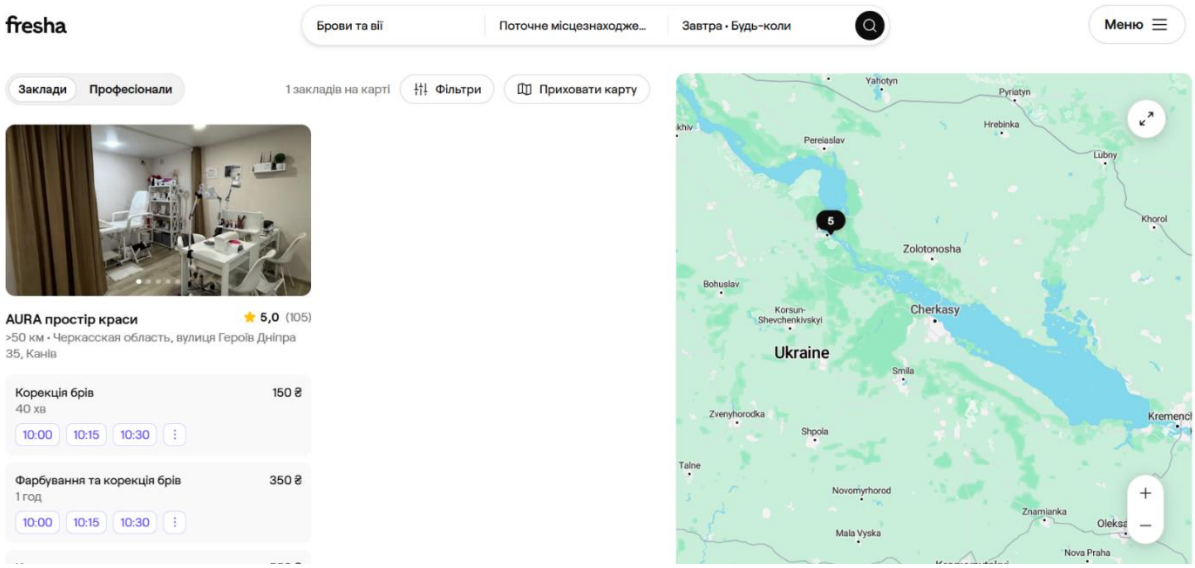

Рисунок 1.3 – Інтерфейс вибору послуг у системі Fresha

2. Helsi (медична інформаційна система України)

Helsi — це українська медична інформаційна система, орієнтована на пацієнтів, лікарів, державні та приватні медичні заклади. На офіційних сторінках сервісу зазначено близько 29 млн пацієнтів, 1 783 клініки та понад 229 тис. записів щодня. Основні функції системи включають пошук лікаря, онлайн-запис на прийом, доступ до електронної медичної картки, документів, рецептів, направлень і сервісів для всієї родини [3] .

Сильною стороною Helsi є надійність роботи з чутливими даними. Офіційно вказано, що дані зберігаються у дата-центрі із сертифікатом КСЗІ, а сама система має Атестат відповідності КСЗІ. Також наголошується, що обробка даних відбувається в правовому полі України, а закладам не потрібне окреме серверне обладнання для роботи із сервісом [4] .

Недоліком Helsi в контексті проєкту «Beauty Time» є її галузева спеціалізація. Офіційні описи системи зосереджені на медичній карті, електронних документах, eHealth та взаємодії з НСЗУ, тобто на сценаріях медичної сфери, а не на маркетингових чи брендованих інструментах для салонного бізнесу. Отже, з точки зору б’юті-сфери це рішення є корисним насамперед як приклад суворої організації запису й роботи з часовими слотами, але не як універсальна основа для клієнтського сервісу салону краси. Календарну сітку запису в системі Helsi наведено на рис. 1.4.

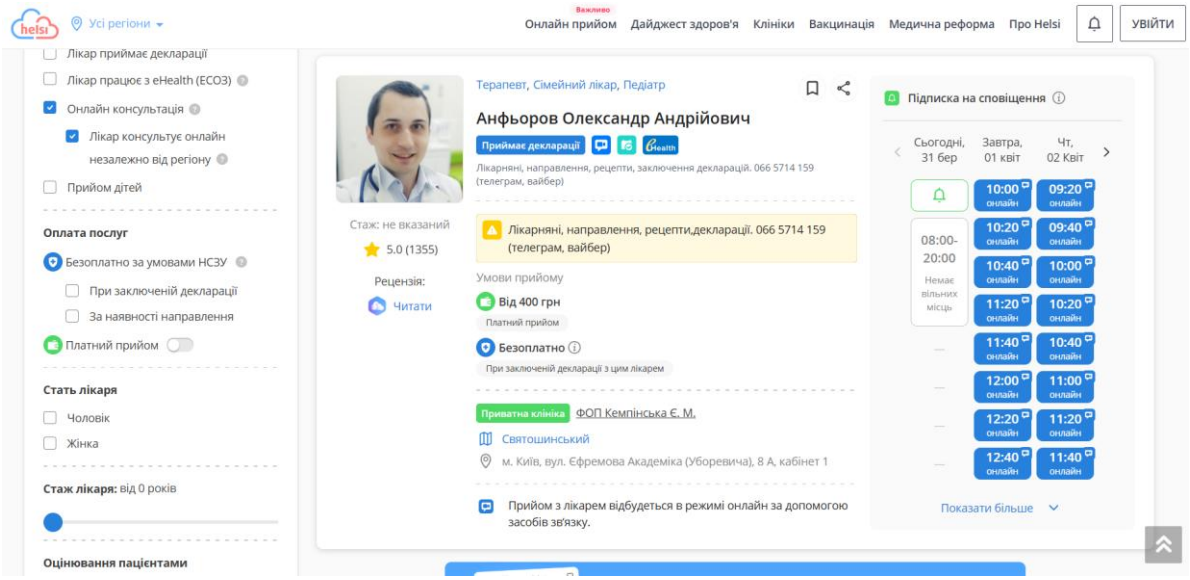

Рисунок 1.4 – Календарна сітка запису в системі Helsi

3. G.Bar (вебзастосунок мережі G×Bar)

G×Bar є прикладом брендованого цифрового рішення для власної мережі салонів. На офіційному сайті компанія вказує, що мережа охоплює 55 б'юті-барів у 13 країнах. Онлайн-запис реалізовано через офіційну сторінку запису та мобільний застосунок. У застосунку доступні вибір однієї або кількох послуг, пошук найближчого доступного часу, вибір майстра або улюбленого майстра, перегляд вартості послуг, історії попередніх візитів, рейтингів і відгуків. Також сервіс надсилає нагадування про запис [5] .

Головною перевагою G.Bar є сильна орієнтація на бренд і клієнтський досвід [6]. На відміну від універсальних платформ, це рішення від початку створене під конкретну мережу та її стиль взаємодії з клієнтом. Саме тому G.Bar є найбільш релевантним аналогом з точки зору візуальної подачі, мобільного UX та інтеграції сервісу запису в загальну комунікацію бренду.

Разом із тим у відкритих матеріалах акцент зроблено переважно на клієнтському застосунку та власному сервісі запису, а не на універсальних інструментах для зовнішньої інтеграції чи використання іншими салонами. Тому G.Bar доцільно розглядати як приклад сильного брендованого рішення, але не як готову модель для масштабованого універсального сервісу. Це важливо врахувати при проєктуванні «Beauty Time».

Процес вибору послуги у вебзастосунку G.Bar показано на рис. 1.5.

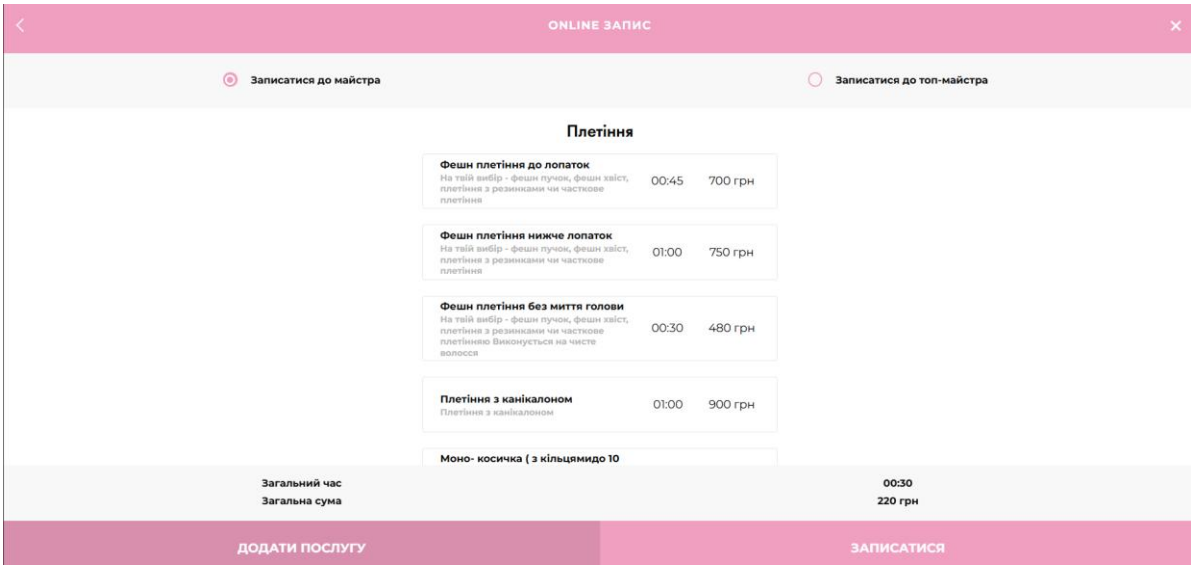

Рисунок 1.5 – Процес вибору послуги у вебзастосунку G.Bar

Порівняльна характеристика аналогічного ПЗ

Для прийняття обґрунтованого рішення щодо вибору технологій та функціонального наповнення власної розробки складено порівняльну таблицю.

Таблиця 1.1 – Порівняльний аналіз систем онлайн-запису

Функція	Fresha	Helsi	G.Bar
Онлайн-запис	Глобальний маркетплейс	Держ. система	Мобільний додаток
Кабінет / CRM	Історія та аналітика	Медкарта / Візити	Профіль клієнта
Сповіщення	SMS / Email	Додатки / Push	Push-повідомлення
Оплата онлайн	Так (карткою)	Ні	Обмежено
Вибір майстра	Фільтрація / Персонал	Спеціалізація	Рейтинги / Майстри
Тайм-слоти	Гнучкий календар	Жорстке планування	Календар мережі
Моб. адаптація	Висока (App + Web)	Висока (App)	Висока (Mobile UX)
API / Інтеграції	Платіжні системи	eHealth / НСЗУ	Власна CRM
Портфоліо	Відсутнє	Відсутнє	Частково (відгуки)

Таблиця 1.2 – Порівняльний аналіз систем онлайн-запису

Характеристика	Fresha	Helsi	G.Bar
Продуктивність	Висока (глобальна)	Висока (2.5 млн/міс)	Середня (локальна)
Масштабованість	Висока (хмарна)	Державна (на всю країну)	Обмежена (франчайзі)
Безпека	Захист платежів	Дуже висока (КСЗІ)	Базова (HTTPS)
Відмовостійкість	Висока (кешування)	Максимальна (SLA)	Середня
Зручність (UX)	Середня (шаблонна)	Середня (функціональна)	Висока (сучасний дизайн)
Підтримка	Базова (спільнота)	Державна гаряча лінія	Внутрішня мережева

Недоліки аналізованих рішень

Fresha є потужною універсальною платформою, однак працює в межах власної сервісної логіки, комісійної моделі та стандартного користувацького сценарію. Це знижує гнучкість при побудові унікального брендованого рішення.

Helsi вирізняється надійністю та високим рівнем захисту даних, проте її функціональність орієнтована на медичну сферу, а не на сервісну модель салону краси. У ній відсутній акцент на бренд, маркетинг і побудову клієнтського досвіду саме в beauty-сегменті.

G.Bar, навпаки, добре демонструє силу брендованого рішення, але є закритою екосистемою конкретної мережі. Тому цей аналог корисний насамперед як зразок візуальної подачі та клієнтського UX, а не як універсальний шаблон для масштабованого зовнішнього продукту.

У результаті аналізу існуючих рішень було визначено, що для салону «Beauty Time» доцільно розробити власний вебзастосунок із використанням сучасних вебтехнологій. Такий підхід дає змогу врахувати переваги розглянутих аналогів, а саме зручність і автоматизацію запису, характерні для Fresha, надійну роботу з часовими слотами, реалізовану в Helsi, а також виразну візуальну подачу й брендованість, притаманні G.Bar. Це створює основу для розроблення зручного, незалежного та адаптованого до потреб конкретного салону цифрового сервісу.

1.3 Формування вимог до веб-орієнтованої системи

Веб-орієнтована інформаційна система для салону краси «Beauty Time» розробляється з метою автоматизації онлайн-запису клієнтів на послуги салону. Необхідність створення такої системи пов’язана з потребою зробити процес бронювання зручнішим, скоротити кількість ручних дій адміністратора, упорядкувати ведення записів і забезпечити доступ до основної інформації через веб-інтерфейс.

На відміну від традиційного запису телефоном або через адміністратора, веб-орієнтована система дає змогу користувачеві самостійно переглянути інформацію про салон, ознайомитися з переліком послуг, актуальними акціями та пакетними пропозиціями, зареєструватися або авторизуватися, обрати дату й доступний

часовий слот для запису. Такий підхід підвищує зручність користування сервісом і знижує навантаження на адміністратора під час обробки записів.

Розроблювана система спрямована на вирішення таких практичних проблем, як відсутність єдиного цифрового середовища для запису клієнтів, ризик накладання записів на один і той самий час, складність ручного ведення обліку записів, потреба у швидкому перегляді й адмініструванні послуг, акцій, пакетних пропозицій і повідомлень, а також необхідність забезпечення зручного способу зворотного зв’язку між клієнтом і салоном.

Основне цільове призначення системи полягає в забезпеченні можливості перегляду інформаційних сторінок сайту, перегляду каталогу послуг із поділом за категоріями, відображення цін та описів послуг, перегляду актуальних акцій і пакетних пропозицій, реєстрації та авторизації користувачів, створення онлайн-запису на послугу, автоматичної перевірки доступності часових слотів, запобігання конфліктним і дубльованим записам, перегляду власних записів, їх перенесення або скасування, а також надсилання повідомлення через форму зворотного зв’язку зі збереженням у базі даних і можливістю email-сповіщення адміністратора.

Таблиця 1.3 – Функціональні вимоги до системи

ID	Категорія	Опис
FR-01	Контент	Відображення інформації про салон на головній сторінці, сторінках «Про нас» і «Контакти».
FR-02	Послуги	Перегляд каталогу послуг із поділом за категоріями, описом і ціною.
FR-03	Доступ	Реєстрація та авторизація користувачів за email і паролем.
FR-04	Запис	Онлайн-бронювання послуги авторизованим користувачем із вибором дати, часу, послуги та імені клієнта.
FR-05	Перевірка	Автоматична перевірка доступності часових слотів.
FR-06	Контроль	Запобігання дублюванню та перетину активних записів у межах одного часу.
FR-07	Розклад	Відображення доступних часових слотів і автоматичний підбір найближчої вільної дати за відсутності слотів.
FR-08	Кабінет	Перегляд користувачем власних активних і скасованих записів.
FR-09	Сервіс	Скасування та перенесення активних записів користувача.
FR-10	Фідбек	Надсилання повідомлення через контактну форму зі збереженням у базі даних.

Продовження таблиці 1.3

FR-11	Пропозиції	Відображення актуальних акцій, пакетних пропозицій і звільнених слотів.
FR-12	Адмін	Керування основними даними системи через Django Admin.

Окремо система забезпечує адміністрування послуг, записів, повідомлень, акцій, пакетних пропозицій і звільнених слотів через Django Admin.

Метою розробки є створення веб-орієнтованої інформаційної системи онлайн-запису для салону краси, яка забезпечує зручне бронювання послуг, базову автоматизацію взаємодії з клієнтами, контроль доступності часових слотів і підтримку адміністрування даних. Для реалізації системи використовуються мова програмування Python, фреймворк Django та архітектурний шаблон MV.

Крім функціональних вимог, для системи визначено нефункціональні вимоги, що характеризують якість її роботи. Вони охоплюють продуктивність, надійність, безпеку, зручність використання, супроводжуваність і можливість подальшого розширення. Система повинна забезпечувати швидке завантаження типових сторінок і форм, коректне збереження записів без конфліктів, використання базових механізмів захисту Django, зрозумілий адаптивний інтерфейс для різних пристроїв, а також структуровану організацію програмного коду для подальшого розвитку.

До основних нефункціональних вимог системи належать:

– швидке завантаження типових сторінок і форм;

– коректне збереження записів без дублювання та часових конфліктів;

– використання базових механізмів захисту веб-застосунку, зокрема автентифікації, CSRF-захисту, валідації форм і захищених production-налаштувань;

– зрозумілий і адаптивний інтерфейс для користувачів на різних пристроях;

– структурована організація коду за моделями, формами, представленнями, сервісами та селекторами;

– можливість подальшого розширення функціональності;

– підтримка кешування публічних даних для підвищення продуктивності.

Під час постановки завдання також ураховано основні технічні обмеження та припущення. Система реалізується як веб-застосунок, що працює у браузері. Для її розробки використовується Django. У середовищі розробки застосовується база даних SQLite, а для production-розгортання передбачено використання PostgreSQL, Docker і захищених налаштувань. Система орієнтована на один салон краси. Запис виконується в межах єдиного розкладу роботи салону без окремого моделювання графіків різних майстрів. Поточна реалізація не передбачає онлайн-оплату, SMS-сповіщення або інтеграцію зі сторонніми календарями. Для коректної роботи системи необхідні доступ до мережі Інтернет і сучасний веб-браузер.

Отже, для веб-орієнтованої інформаційної системи салону краси «Beauty Time» визначено цільове призначення, основні можливості, функціональні та нефункціональні вимоги, а також ключові технічні обмеження. Актуальна реалізація системи не лише відповідає базовим вимогам онлайн-запису, а й розширює їх за рахунок акцій, пакетних пропозицій, звільнених слотів, email-сповіщень і production-готової конфігурації.

РОЗДІЛ 2 МОДЕЛЬ ТА СПЕЦИФІКАЦІЯ СТРУКТУРИ СИСТЕМИ ОНЛАЙН-ЗАПИСУ

2.1 Концептуальне моделювання системи

Концептуальне моделювання системи є важливим етапом проєктування, оскільки саме на цьому рівні формується узагальнене уявлення про майбутню інформаційну систему [12] , її основні сутності, інформаційні потоки та логіку функціонування. На відміну від етапів технічного проєктування і програмної реалізації, концептуальна модель не відображає структуру програмного коду, внутрішню організацію модулів або конкретні технології розробки. Її основне призначення полягає у формалізації результатів аналізу предметної області, перевірці логічної узгодженості вимог та створенні основи для подальшого прийняття архітектурних рішень.

У роботі досліджується веб-орієнтована інформаційна система салону краси «Beauty Time», основним призначенням якої є автоматизація процесу онлайн-запису клієнтів, надання відомостей про діяльність салону, відображення каталогу послуг, актуальних акцій і пакетних пропозицій, а також підтримка базової взаємодії з користувачами. Концептуальна модель системи повинна відобразити її ключове призначення: забезпечення зручного механізму бронювання послуг, збереження інформації про записи, контроль доступності часових слотів, підтримку керування записами та обробку повідомлень від клієнтів.

На концептуальному рівні систему доцільно розглядати як єдине цифрове середовище, у межах якого взаємодіють кілька типів учасників. Основними акторами системи є:

– неавторизований відвідувач, який може переглядати інформаційні сторінки, каталог послуг, акції, пакетні пропозиції, контакти, зареєструватися, увійти в систему або надіслати повідомлення через форму зворотного зв’язку;

– зареєстрований користувач, який, крім базових інформаційних функцій, може створювати онлайн-записи, переглядати доступні часові слоти, працювати з особистим кабінетом, переглядати власні записи, переносити або скасовувати активні записи;

– адміністратор, який керує основними даними системи через адміністративну панель Django, зокрема послугами, записами, повідомленнями, акціями, пакетними пропозиціями та звільненими слотами.

Взаємодію основних акторів із функціями інформаційної системи наведено на рис. 2.1.

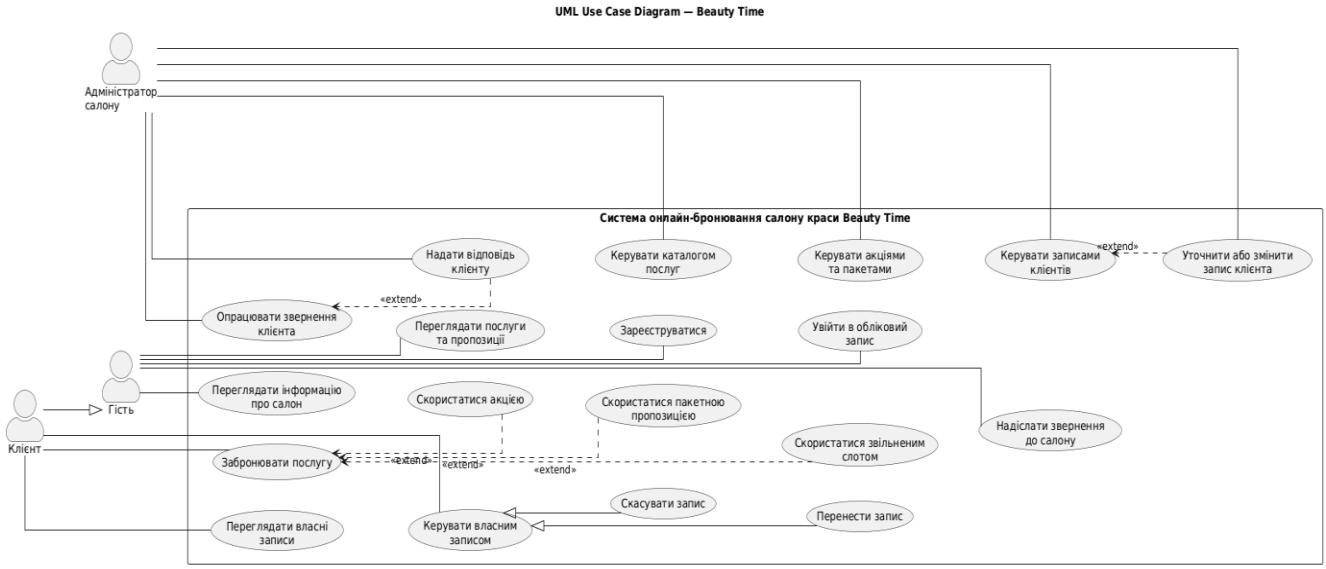

Рисунок 2.1 – Діаграма прецедентів.

З погляду предметної області центральне місце в системі займає процес запису на послугу. Саме він поєднує інформацію про клієнта, обрану послугу, дату, час, тривалість і стан бронювання. Для забезпечення цього процесу в концептуальній моделі виокремлюються такі основні сутності:

– користувач - зареєстрований клієнт, який взаємодіє із системою через особистий обліковий запис;

– послуга - елемент каталогу салону, що характеризується назвою, категорією, описом і вартістю;

– запис - факт бронювання послуги на конкретну дату та час із визначеним статусом і тривалістю;

– повідомлення - звернення користувача або відвідувача, надіслане через контактну форму;

– акція - спеціальна пропозиція, яка може бути пов’язана з окремою послугою та має період дії;

– пакетна пропозиція - комплексна пропозиція, що може містити кілька послуг і впливати на тривалість запису;

– звільнений слот - часовий проміжок, який став доступним після скасування або перенесення запису.

Концептуально між зазначеними сутностями існують такі зв’язки. Один користувач може мати кілька записів, тобто між сутностями «користувач» і «запис» існує зв’язок типу «один до багатьох». Одна послуга також може бути обрана у багатьох записах. Акція може бути пов’язана з певною послугою та використовуватися під час створення запису. Пакетна пропозиція може включати кілька послуг, а запис, створений на основі пакета, блокує часовий проміжок відповідно до тривалості такого пакета. Сутність «повідомлення» є логічно окремою від процесу бронювання, але входить до загального інформаційного простору системи, оскільки забезпечує канал комунікації клієнта із салоном. Звільнений слот формується після скасування або перенесення активного запису та може бути показаний користувачам як доступна пропозиція для бронювання.

Важливою частиною концептуального моделювання є узагальнений опис інформаційних потоків у системі. На вхід системи надходять дані від користувача: реєстраційні дані, email і пароль для входу, обрана послуга або пропозиція, дата, час, ім’я клієнта або текст повідомлення зворотного зв’язку. Система обробляє ці дані, виконує перевірку коректності, зіставляє обраний часовий проміжок із наявними активними записами та формує результат у вигляді підтвердження створення запису, повідомлення про недоступність часу, списку доступних слотів, найближчої вільної дати, відображення власних записів або збереження звернення в базі даних.

Таким чином, інформаційні потоки системи можна умовно поділити на такі групи:

– потоки інформаційного доступу, пов’язані з переглядом сторінок сайту, каталогу послуг, акцій, пакетних пропозицій і контактної інформації;

– потоки автентифікації, пов’язані з реєстрацією користувача, входом до системи та виходом з облікового запису;

– потоки транзакційної взаємодії, пов’язані зі створенням, перенесенням і скасуванням записів;

– потоки комунікації, пов’язані з надсиланням повідомлень через форму зворотного зв’язку;

– адміністративні потоки, пов’язані з керуванням послугами, записами, повідомленнями, акціями, пакетними пропозиціями та звільненими слотами.

Логіка функціонування системи на концептуальному рівні базується на послідовності типових сценаріїв взаємодії. Базовий сценарій починається з перегляду користувачем інформації про салон, каталогу послуг, акцій або пакетних пропозицій. Якщо користувач бажає створити запис, система перевіряє, чи він авторизований. Неавторизований відвідувач перенаправляється на сторінку входу або реєстрації. Після авторизації користувач переходить до вибору послуги, дати та доступного часу. Система отримує наявні активні записи, формує список доступних часових слотів з урахуванням тривалості запису та робочого часу салону. Якщо на вибрану дату немає вільного часу, система може запропонувати найближчу доступну дату.

Після вибору часу користувач надсилає форму запису. Система повторно виконує серверну перевірку, щоб запобігти конфлікту з іншими активними записами. Якщо конфлікту немає, створюється новий запис, а відповідний звільнений слот, якщо він існував, деактивується. Якщо обраний часовий проміжок уже недоступний, система повідомляє користувача про помилку та пропонує обрати інший доступний час. У подальшому зареєстрований користувач може перейти до особистого кабінету, переглянути власні записи, перенести активний запис на інший доступний час або скасувати його. У разі скасування чи перенесення запису попередній часовий слот може бути позначений як звільнений і доступний для інших клієнтів.

Послідовність виконання процесу онлайн-запису в інформаційній системі «Beauty Time» наведено на рис. 2.2.

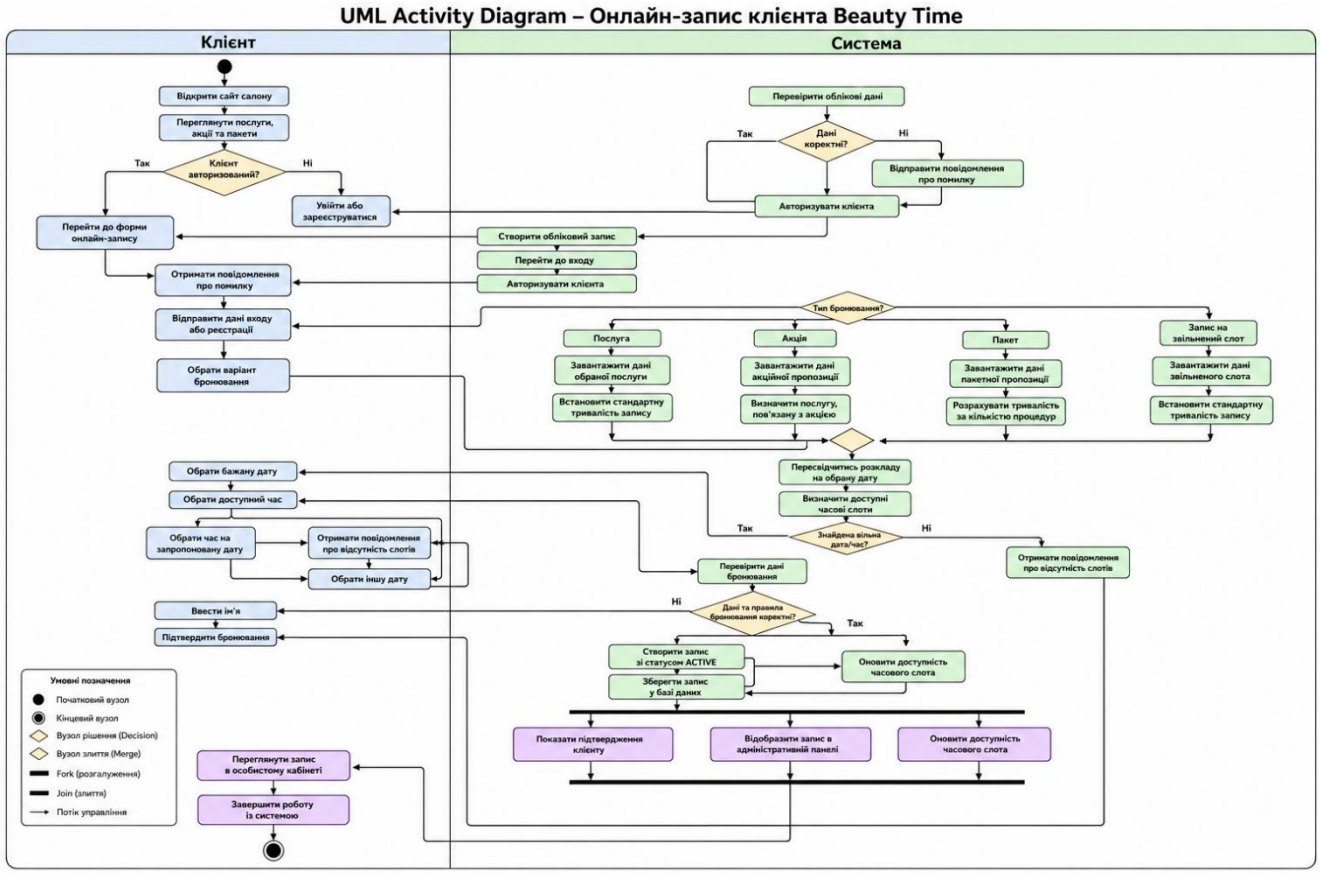

Рисунок 2.2 – Діаграма діяльності процесу онлайн-запису в інформаційній системі «Beauty Time»

Окрему роль у концептуальній моделі відіграють обмеження предметної області. Насамперед система повинна забезпечувати відсутність конфліктів у розкладі, тобто унеможливлювати бронювання одного й того самого або перетинного часового проміжку для різних клієнтів. Крім того, запис має створюватися лише в межах робочого часу салону. Запис на послугу доступний лише авторизованому користувачу, що забезпечує зв’язок створеного запису з конкретним обліковим записом.

Отже, концептуальне моделювання системи «Beauty Time» дозволяє узагальнено описати її призначення, основні сутності, інформаційні потоки та сценарії функціонування. Результати цього етапу створюють підґрунтя для подальшого аналітико-модельного опрацювання, проєктування структури даних, визначення функціональної архітектури та реалізації веб-орієнтованої інформаційної системи.

2.2 Попередня архітектура та проєктування системи

Попередню архітектуру системи «Beauty Time» сформовано як монолітний веб-застосунок клієнт-серверного типу з багатосторінковим інтерфейсом [13]. Усі вони взаємодіють із реляційною базою даних через рівень моделей Django ORM [14].Такий підхід відповідає характеру предметної області, у якій основні сценарії взаємодії зосереджені навколо перегляду послуг, акцій і пакетних пропозицій, створення онлайн-запису авторизованим користувачем, керування власними бронюваннями та надсилання повідомлень через контактну форму. Єдина серверна частина дає змогу зосередити критичну бізнес-логіку в одному контурі, забезпечити узгоджене застосування правил бронювання та уникнути дублювання перевірок у різних частинах системи.

Архітектурна концепція системи узгоджується з MV-шаблоном, що використовується у фреймворку Django. У межах такого підходу представлення даних для користувача реалізується через шаблони сторінок, обробка HTTP-запитів і координація сценаріїв взаємодії зосереджуються у views, а збереження предметних сутностей забезпечується моделями. Крім того, частина прикладної логіки винесена в окремі сервіси та селектори, що спрощує повторне використання правил бронювання, отримання даних і підтримку коду. Це дозволяє логічно відокремити інтерфейс, прикладну логіку та рівень даних без надмірного ускладнення загальної побудови системи.

Функціональна структура системи поділяється на кілька контурів. Публічний контур охоплює інформаційні сторінки сайту, каталог послуг, перегляд акцій і пакетних пропозицій, сторінку контактів, форму зворотного зв’язку, а також сторінки реєстрації та входу. Автентифікований контур охоплює форму онлайн-запису, динамічне отримання доступних часових слотів, створення бронювання та особистий кабінет користувача, де доступні перегляд, перенесення та скасування власних записів. Службовий контур реалізовано через стандартний інтерфейс Django Admin, за допомогою якого здійснюється керування послугами, записами, повідомленнями клієнтів, акціями, пакетними пропозиціями та звільненими

слотами. Для поточного масштабу системи таке рішення є достатнім і не потребує створення окремого спеціалізованого адміністративного інтерфейсу.

На рівні попереднього проєктування система розглядається як сукупність взаємопов’язаних програмних компонентів. Основними архітектурними елементами є веб-інтерфейс користувача, підсистема маршрутизації та представлень (views), компоненти валідації форм, а також окремі сервіси прикладної логіки (сервіс запису, каталог послуг та вбудована адмін-панель Django). Усі вони взаємодіють із реляційною базою даних через рівень моделей Django ORM, що забезпечує збереження та обробку інформації про користувачів, послуги, записи та повідомлення.

Узагальнену компонентну структуру системи наведено на рис. 2.3.

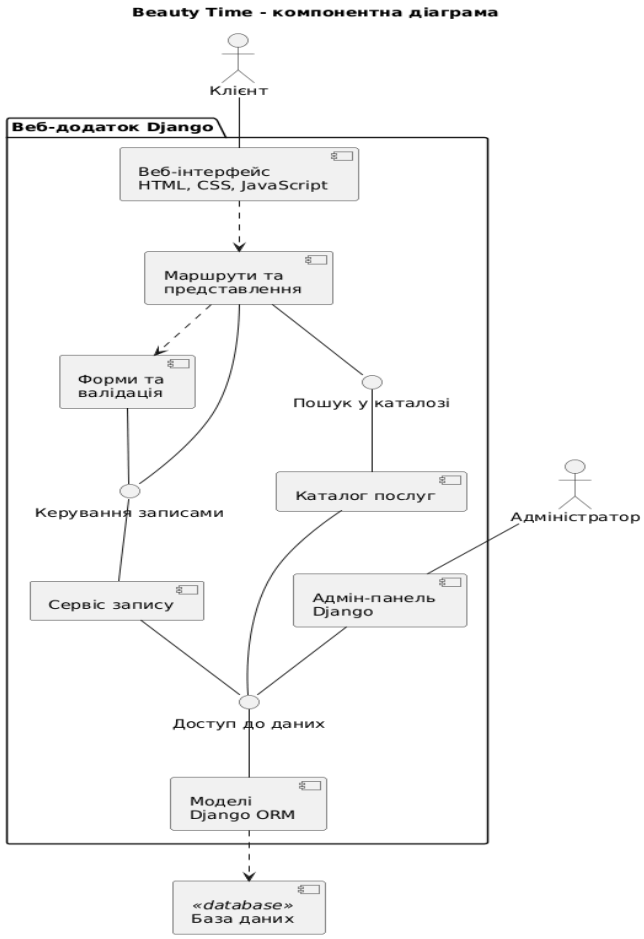

Рисунок 2.3 – Узагальнена компонентна структура системи

Ключовим архітектурним рішенням стало поєднання серверного формування сторінок із локальною динамічною взаємодією в тих фрагментах інтерфейсу, де вона дійсно потрібна. Основна частина системи працює як традиційний веб-застосунок із серверним рендерингом сторінок, що позитивно впливає на простоту реалізації та супроводу. Водночас для сторінки онлайн-запису передбачено механізм динамічного отримання доступних часових слотів після вибору дати. Такий підхід дає змогу зменшити кількість зайвих дій користувача, не перетворюючи систему на окремий SPA-застосунок.

Центральним сценарієм, навколо якого побудовано попередню архітектуру, є процес запису на послугу. Користувач переходить на сторінку запису після авторизації, обирає послугу, акцію або пакетну пропозицію, вказує дату, після чого система визначає наявність доступного часу. Якщо на вибрану дату вільні слоти відсутні, користувачу пропонується найближча доступна дата з вільним часом. Після надсилання форми здійснюється серверна перевірка коректності введених даних і повторна перевірка відсутності конфлікту з активними записами. Лише після цього створюється новий запис. Якщо бронювання здійснюється на основі пакетної пропозиції, система враховує збільшену тривалість запису відповідно до кількості процедур у пакеті. Така побудова дозволяє знизити ризик накладання бронювань і підвищує надійність роботи системи в разі одночасних дій кількох користувачів.

Попереднє проєктування інтерфейсу орієнтоване на короткий і зрозумілий маршрут користувача. Основними кроками взаємодії є ознайомлення з інформацією про салон, перегляд послуг, акцій або пакетних пропозицій, авторизація за потреби, перехід до форми запису, вибір дати й часу, підтвердження дії та подальше керування створеним записом. Особистий кабінет зосереджений лише на тих операціях, що є справді необхідними після бронювання: перегляді, перенесенні та скасуванні власних записів. Під час попереднього проєктування зручність інтерфейсу оцінювалася через проходження основних користувацьких сценаріїв без проведення окремого емпіричного UX-дослідження.

Попереднє проєктування основних користувацьких екранів системи представлено на рис. 2.4–2.5.

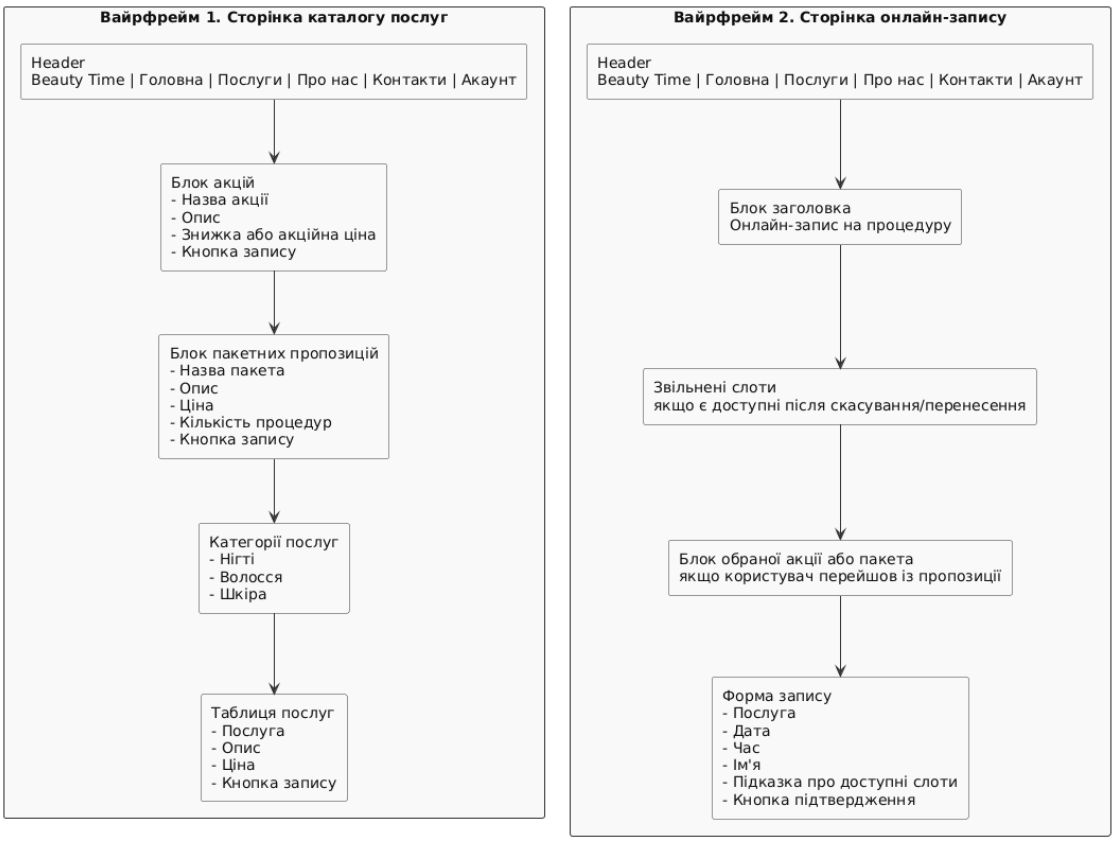

Рисунок 2.4 – Вайрфрейми основних користувацьких екранів системи

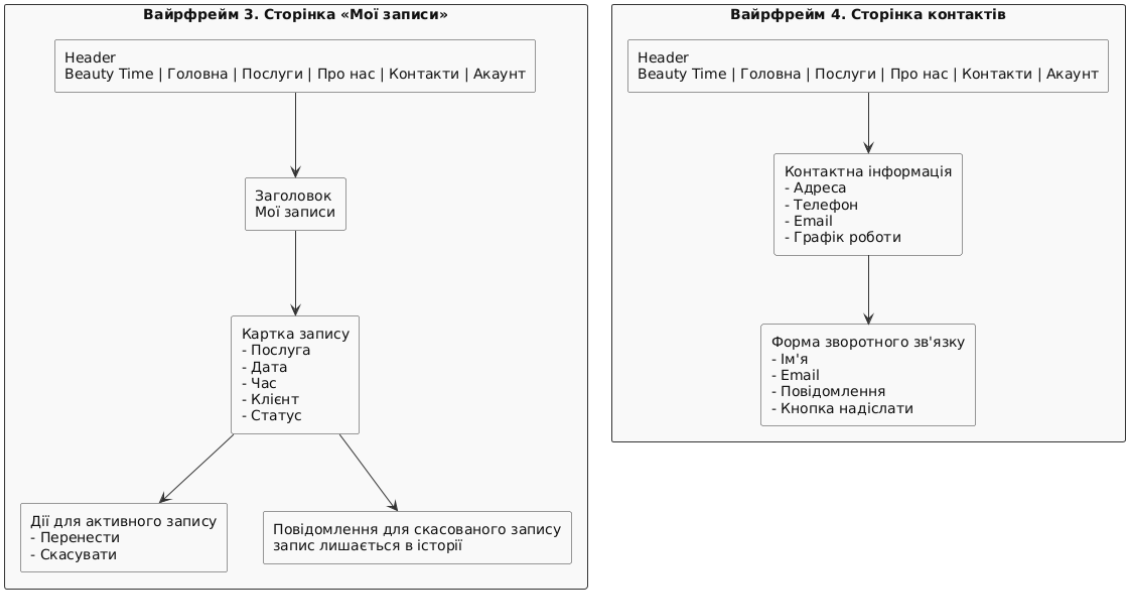

Рисунок 2.5– Вайрфрейми основних користувацьких екранів системи

База даних системи «Beauty Time» побудована як реляційне сховище, що відображає основні сутності предметної області та зв’язки між ними [15]. Центральною таблицею є таблиця записів, у якій зберігаються дані про клієнта, обрану послугу, дату, час, тривалість і статус бронювання. Запис пов’язується з користувачем і послугою, а також за потреби може містити посилання на акцію або пакетну пропозицію. Окремими таблицями зберігаються послуги салону, акції, пакетні пропозиції, звільнені часові слоти та повідомлення з контактної форми. Для зв’язку пакетних пропозицій із послугами використовується проміжна таблиця, оскільки один пакет може включати кілька послуг, а одна послуга може входити до різних пакетів. Повідомлення зворотного зв’язку логічно пов’язані з користувачем або відвідувачем, який їх надсилає, однак у базі даних вони зберігаються окремо без обов’язкового зовнішнього ключа на користувача, оскільки контактна форма доступна також неавторизованим відвідувачам. Така структура бази даних забезпечує збереження основних даних системи, підтримує процес онлайн-запису та дозволяє адмініструвати послуги, бронювання, повідомлення й спеціальні пропозиції через Django Admin. Модель даних інформаційної системи «Beauty Time» наведено на рис. 2.6.

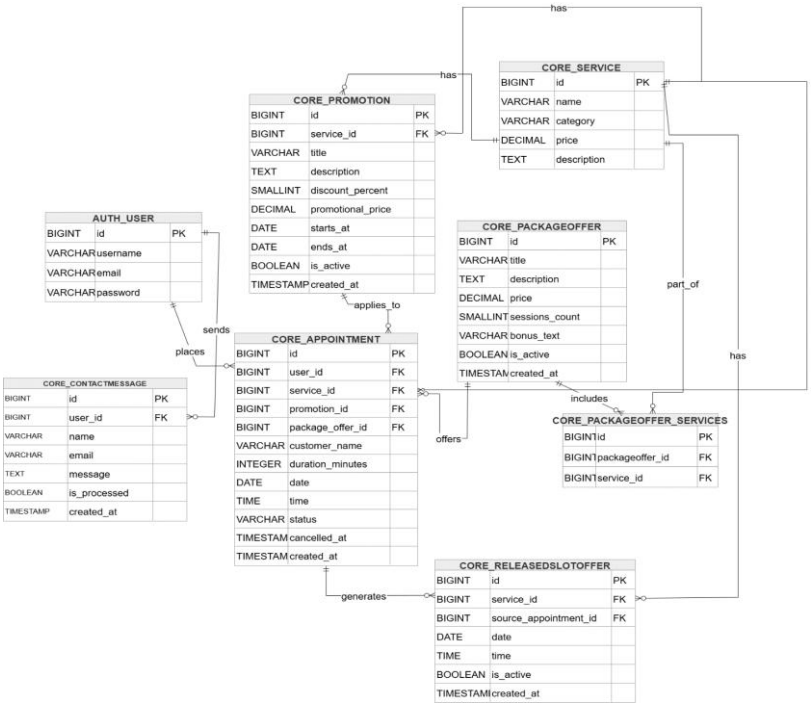

Рисунок 2.6– Модель даних системи «Beauty Time»

Важливим аспектом попередньої архітектури є розмежування доступу та визначення довірчих меж. Модель доступу побудовано на трьох ролях: неавторизований відвідувач, зареєстрований користувач і адміністратор. Неавторизований відвідувач може переглядати публічні сторінки, каталог послуг, акції, пакетні пропозиції, контакти, зареєструватися, увійти в систему та надіслати повідомлення через контактну форму. Зареєстрований користувач додатково отримує доступ до створення онлайн-запису, перегляду доступних часових слотів, власних записів і можливості їх перенесення або скасування. Адміністратор має службовий доступ до керування даними через стандартний інтерфейс Django Admin, зокрема до послуг, записів, повідомлень, акцій, пакетних пропозицій і звільнених слотів. Межа довіри проходить між браузером користувача та серверною частиною системи: усі вхідні дані з боку клієнта вважаються потенційно недовіреними та повинні проходити серверну перевірку. Такий підхід забезпечує контроль доступу до функцій системи та створює основу для захисту даних під час обробки користувацьких запитів.

Безпека системи на рівні попереднього проєктування забезпечується через автентифікацію користувачів, розмежування ролей і прав доступу, серверну валідацію даних та відсутність прямого доступу клієнта до бази даних. Критичні дії, пов’язані зі створенням, перенесенням і скасуванням бронювань, виконуються лише на сервері та доступні для авторизованого користувача. Така модель дозволяє підтримувати логічну цілісність даних і знижує ризик некоректної обробки записів.

Прийняті архітектурні рішення узгоджуються з основними атрибутами якості програмного забезпечення. Надійність забезпечується централізацією критичної логіки бронювання на сервері та повторною перевіркою доступності слотів. Зручність використання підтримується коротким сценарієм взаємодії та динамічним отриманням доступного часу. Супроводжуваність досягається логічним розподілом відповідальності між рівнем представлення, прикладною логікою та рівнем даних, а також використанням окремих сервісів і селекторів для бізнес-правил та отримання даних. Масштабованість у межах поточної концепції

полягає у можливості подальшого розширення функціоналу без зміни базового архітектурного підходу.

Отже, попередня архітектура системи «Beauty Time» формує узгоджену основу для подальшої реалізації програмного продукту. Вона враховує специфіку предметної області, ролі користувачів, центральний сценарій онлайн-запису, вимоги до безпеки та зручності використання, а також забезпечує зв’язок між функціональною структурою системи, її поведінкою та основними інформаційними сутностями.

2.3 Реалізація програмного продукту

Реалізацію програмного продукту «Beauty Time» виконано на основі фреймворку Django мовою Python [16] . Застосунок побудовано як монолітну веб-систему клієнт-серверного типу з багатосторінковим інтерфейсом, у якій серверна частина відповідає за маршрутизацію, обробку HTTP-запитів, валідацію введених даних, виконання бізнес-правил, автентифікацію користувачів і роботу з базою даних. Клієнтська частина забезпечує відображення сторінок, роботу веб-форм і локальну динамічну взаємодію в окремих сценаріях, зокрема під час отримання доступних часових слотів на сторінці онлайн-запису.

Проєкт організовано у Git-репозиторії з логічною структурою директорій [17], узгодженою з прийнятою архітектурою системи. У репозиторії виділено окремі каталоги для глобальної конфігурації проєкту, прикладної логіки, шаблонів, статичних ресурсів, тестів, документації та контейнеризованого запуску. У корені репозиторію розміщено ключові файли, необхідні для запуску, конфігурації та розгортання системи, зокрема manage.py, requirements.txt, Dockerfile, docker-compose.yml, .env.example, pytest.ini та README.md.

Структуру Git-репозиторію програмного продукту «Beauty Time» наведено на рис. 2.7.

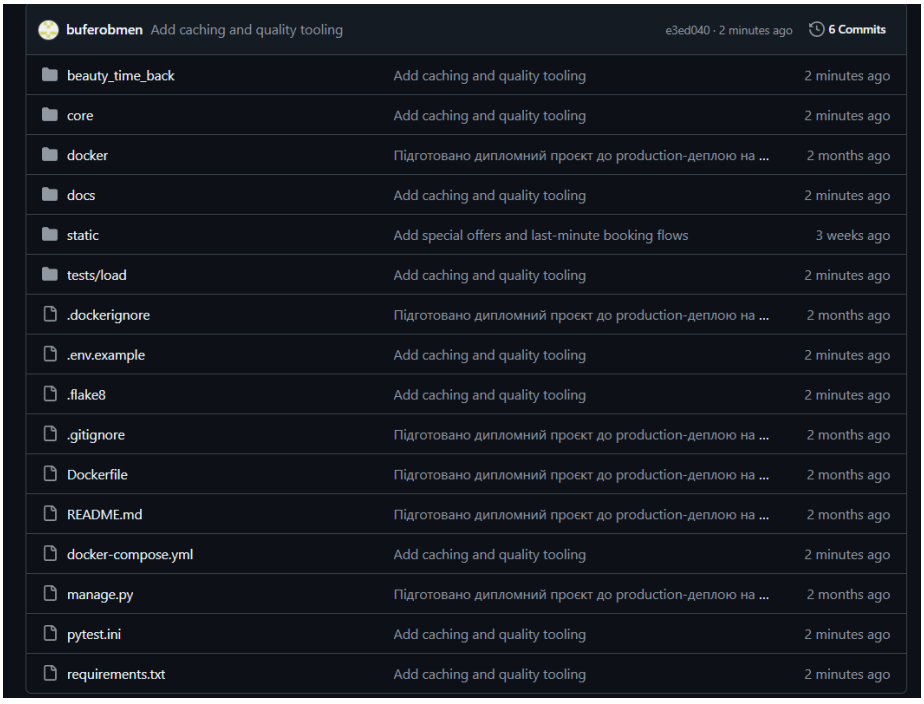

Рисунок 2.7 – Структура Git-репозиторію програмного продукту «Beauty Time».

Розробка продукту велася поетапно з фіксацією ключових змін у системі контролю версій Git. Історія комітів відображає основні етапи розвитку проєкту, серед яких ініціалізація репозиторію, реалізація реєстрації та авторизації користувачів, створення механізму онлайн-запису, додавання особистого кабінету, перенесення та скасування записів, реалізація контактної форми, підготовка production-конфігурації, вдосконалення логіки доступності часових слотів, а також додавання акцій, пакетних пропозицій і звільнених слотів. Наявність осмисленої історії змін дає змогу простежити послідовність реалізації функціоналу та підтверджує поетапний характер розробки програмного продукту.

Історію розробки та основні коміти проєкту «Beauty Time» наведено на рис.2.8.

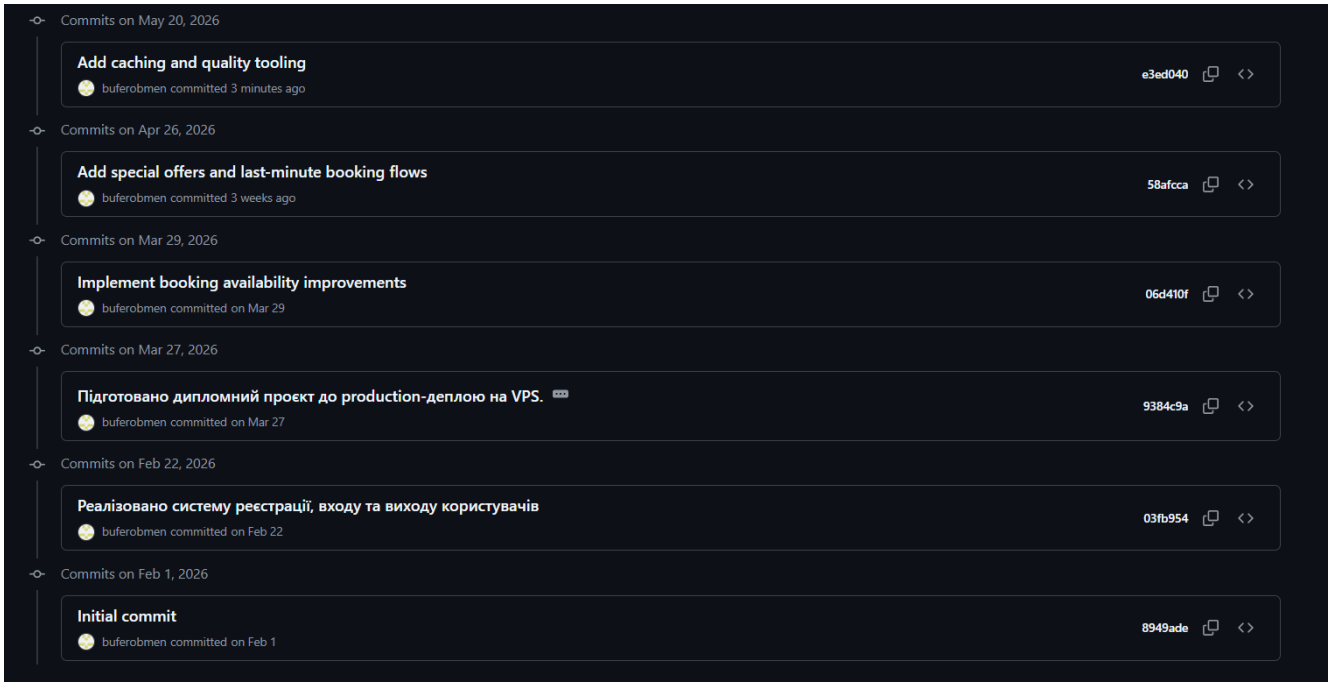

Рисунок 2.8 – Історія розробки та комітів проєкту «Beauty Time».

Окрему роль у технічній реалізації відіграє README-файл, який виконує функцію основної інструкції з запуску, перевірки та демонстрації системи. У ньому подано відомості про локальний запуск, контейнеризоване розгортання через Docker, базову архітектурну структуру проєкту, використання службового health-endpoint, а також рекомендації щодо production-конфігурації на VPS. Такий

супровідний матеріал підвищує відтворюваність програмного продукту та полегшує його демонстрацію.

Фрагмент README-файлу з інструкцією запуску та розгортання програмного продукту наведено на рис. 2.9.

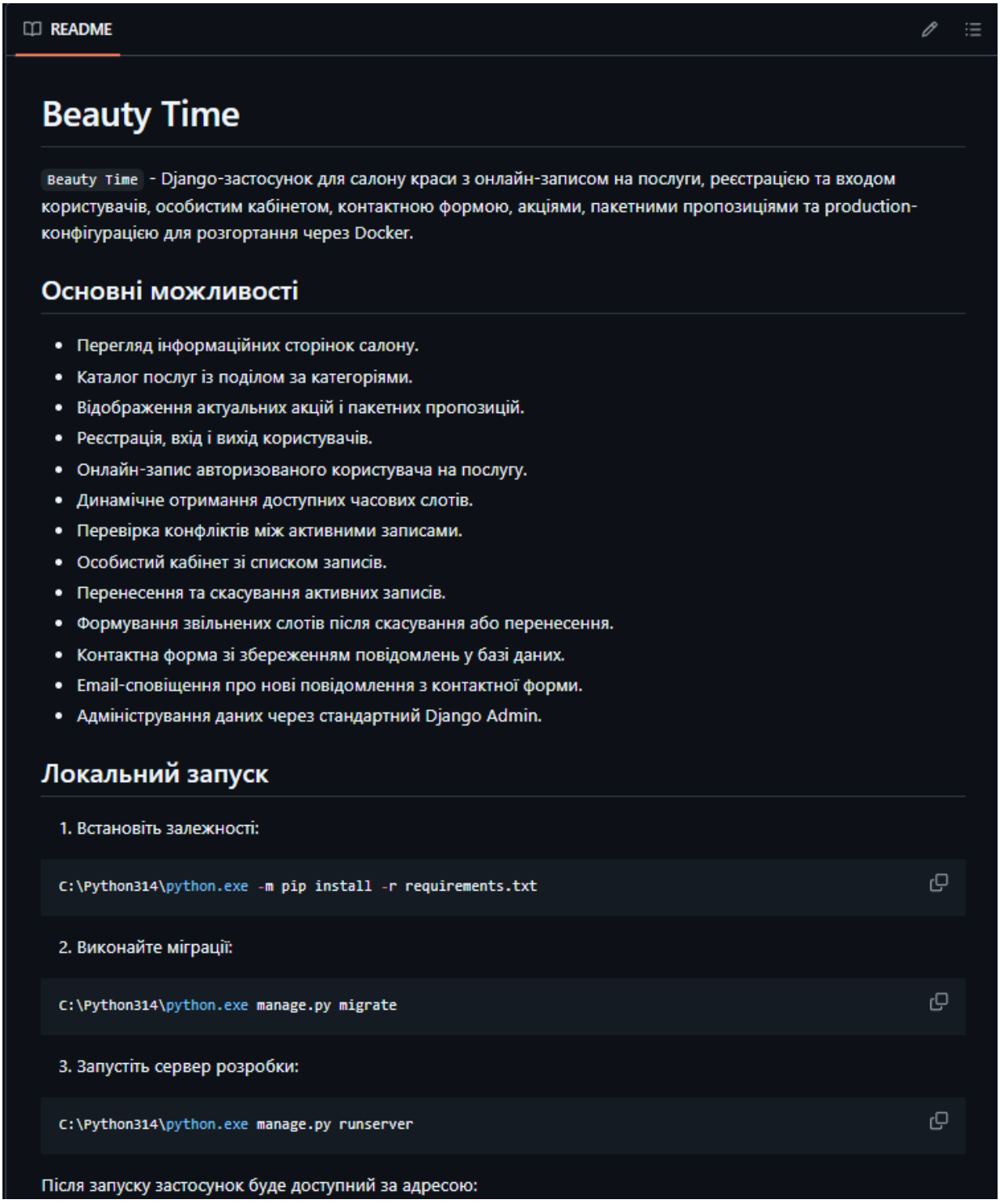

Рисунок 2.9 – Фрагмент README-файлу з інструкцією запуску та розгортання програмного продукту «Beauty Time».

Структуру первинного коду побудовано за принципом розмежування відповідальності. Глобальна конфігурація, маршрути верхнього рівня та налаштування середовищ зосереджені в каталозі beauty_time_back. Основна прикладна логіка розміщена в застосунку core, де реалізовано моделі даних, форми, представлення, службову логіку, селектори, кешування, сигнали, email-сповіщення та адміністративні налаштування. Каталог static містить таблиці стилів, JavaScript-файли та інші статичні ресурси інтерфейсу, каталог tests використовується для додаткових перевірок, а каталог docker призначено для контейнеризованого запуску. Така організація коду відповідає архітектурі системи та сприяє підтримуваності реалізації.

Рівень моделей описує основні сутності предметної області: послугу, запис, повідомлення зворотного зв’язку, акцію, пакетну пропозицію та звільнений часовий слот. Для користувачів використовується стандартна модель Django User, яка застосовується для реєстрації, входу та прив’язки записів до конкретного облікового запису. Форми відповідають за серверну валідацію вхідних даних і підтримують сценарії реєстрації, входу, створення запису, перенесення бронювання та надсилання повідомлень. Представлення координують взаємодію між формами, шаблонами та бізнес-логікою. Окремий сервісний шар реалізує правила бронювання, розрахунок доступних часових слотів, створення, перенесення та скасування записів. Для читання даних використано окремі селектори, що зменшує навантаження на представлення та забезпечує більш прозору структуру коду.

Функціонально система охоплює кілька взаємопов’язаних сценаріїв. Публічна частина надає доступ до головної сторінки, сторінки про салон, каталогу послуг, акцій, пакетних пропозицій і контактної форми. Користувач може зареєструвати обліковий запис, увійти в систему та після авторизації створити новий запис на послугу. Після входу відкривається доступ до особистого кабінету, де користувач може переглядати власні записи, переносити їх або скасовувати. Повідомлення, надіслані через контактну форму, зберігаються в базі даних як окремі інформаційні об’єкти та можуть супроводжуватися email-сповіщенням

адміністратора. Службове керування послугами, записами, повідомленнями, акціями, пакетними пропозиціями та звільненими слотами реалізовано через стандартний інтерфейс Django Admin.

Центральною частиною реалізації став модуль онлайн-запису. Його логіка побудована навколо формування доступних часових слотів у межах робочого часу салону. У поточній реалізації день запису поділено на півгодинні стартові слоти, стандартна тривалість звичайного запису становить одну годину, а для пакетної пропозиції тривалість визначається кількістю процедур у пакеті. Під час вибору дати система визначає доступний час з урахуванням уже створених активних записів, виключає конфліктні інтервали та не пропонує минулий час для поточного дня. Якщо на вибрану дату вільних слотів немає, користувачеві автоматично пропонується найближча дата, на якій доступний запис.

Після надсилання форми система повторно перевіряє коректність дати й часу, відповідність робочому графіку та відсутність конфлікту з іншими активними бронюваннями. Така перевірка виконується на сервері й забезпечує узгодженість даних навіть у випадку одночасних дій кількох користувачів. Перенесення запису використовує ті самі правила перевірки, що й первинне бронювання, а скасування реалізовано через зміну статусу запису без фізичного видалення даних. Після скасування або перенесення попередній часовий слот може бути збережений як звільнений слот і показаний іншим користувачам як доступна пропозиція для бронювання.

Програмний інтерфейс взаємодії з функціями системи реалізовано у вигляді набору HTTP-маршрутів. Основна частина функціональності працює через серверно згенеровані HTML-сторінки та веб-форми. Окремий JSON-інтерфейс використовується для отримання доступних часових слотів на сторінці онлайн-запису. Додатково реалізовано службовий endpoint перевірки стану застосунку, який використовується для технічного моніторингу працездатності.

Таблиця 2.1 – Основні маршрути та точки взаємодії системи «Beauty Time»

Метод	Маршрут	Призначення	Формат відповіді	Рівень доступу
GET	/	Відображення головної сторінки	HTML	Публічний
GET	/about/	Відображення сторінки про салон	HTML	Публічний
GET	/services/	Відображення сторінки послуг	HTML	Публічний
GET, POST	/booking/	Відображення форми запису та створення нового запису	HTML / redirect	Публічний
GET	/booking/available-slots/?date=YYYY-MM-DD	Отримання доступних часових слотів для вибраної дати	JSON	Публічний
GET, POST	/contacts/	Відображення контактної форми та надсилання повідомлення	HTML / redirect	Публічний
GET, POST	/register/	Реєстрація нового користувача	HTML / redirect	Публічний
GET, POST	/login/	Авторизація користувача	HTML / redirect	Публічний
GET	/my-appointments/	Перегляд власних записів користувача	HTML	Авторизований користувач
POST	/my-appointments/<appointment_id>/cancel/	Скасування активного запису	redirect	Авторизований користувач
GET, POST	/my-appointments/<appointment_id>/reschedule/	Перегляд форми перенесення та перенесення запису	HTML / redirect	Авторизований користувач
GET	/logout/	Вихід із системи	redirect	Авторизований користувач

Продовження таблиці 2.1

GET	/health/	Перевірка стану застосунку та доступності бази даних	JSON	Службовий
GET, POST	/admin/	Стандартний службовий інтерфейс Django admin	HTML	Адміністратор

Для динамічного оновлення часу запису реалізовано окремий JSON-інтерфейс, який повертає список доступних слотів на обрану дату або, за потреби, найближчу дату з вільним часом. Такий підхід дає змогу підвищити зручність користувацького сценарію та скоротити кількість повних перезавантажень сторінки. Приклад JSON-відповіді endpoint отримання доступних часових слотів наведено в лістингу 2.1.

Лістинг 2.1 – Приклад JSON-відповіді для отримання доступних часових слотів

```
{
 "requested_date": "2026-04-02",
 "date": "2026-04-03",
 "available_times": ["09:00", "09:30", "10:30", "11:00"],
 "message": "На 02.04.2026 вже немає вільного часу. Підібрали найближчу вільну дату: 03.04.2026.",
 "redirected_to_next_date": true
}
```

Відтворюваність запуску забезпечується кількома рівнями конфігурації. Для локальної розробки використовується requirements.txt, файл manage.py та окремі налаштування середовища development, у яких базою даних виступає SQLite. Для контейнеризованого запуску створено Dockerfile, docker-compose.yml і сценарій старту контейнера, який автоматично виконує міграції, збір статичних ресурсів і запуск застосунку через Gunicorn. Файл .env.example містить шаблон змінних середовища, необхідних для конфігурації під час розгортання. У production-середовищі передбачено використання PostgreSQL, trusted origins, захищених cookie та інших налаштувань безпеки.

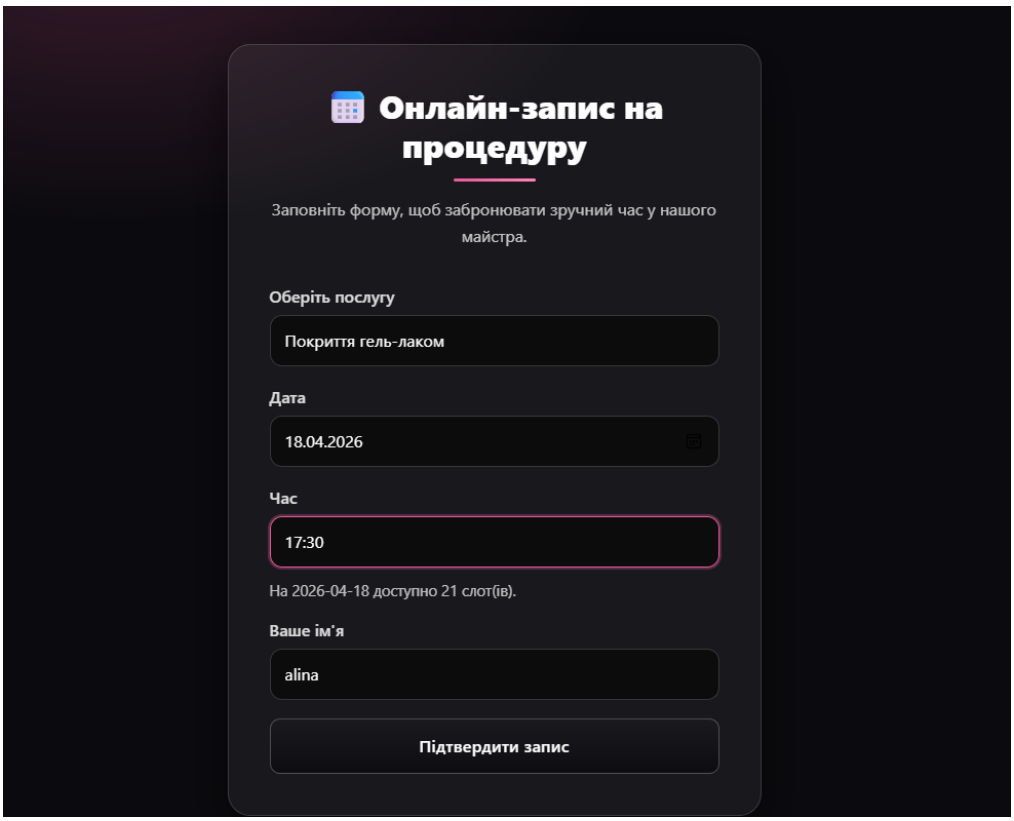

Рисунок 2.9 – Сторінка онлайн-запису реалізованого веб-застосунку «Beauty Time».

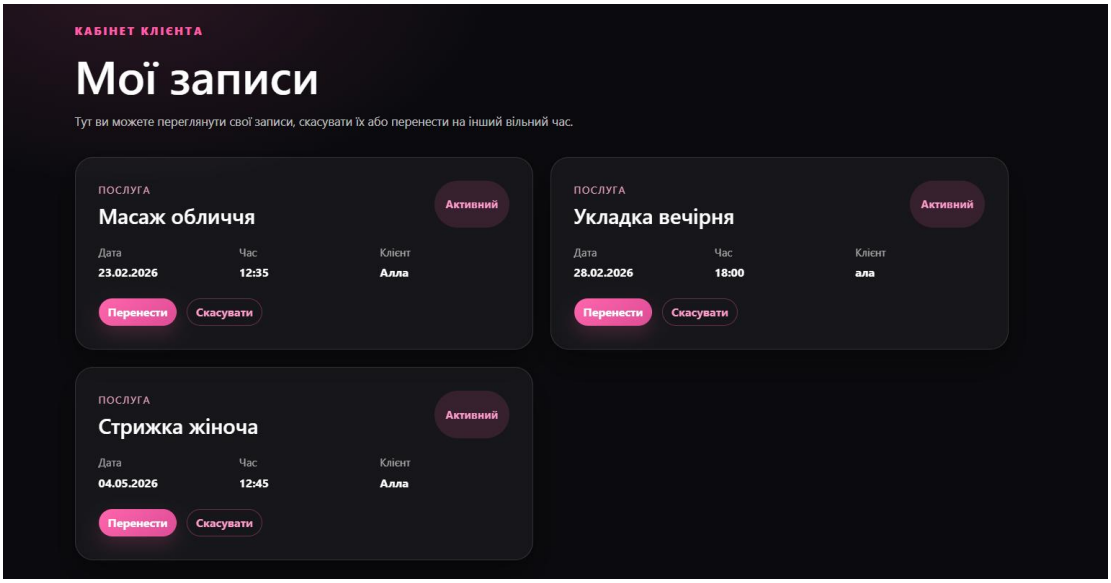

Рисунок 2.10 – Сторінка «Мої записи» реалізованого веб-застосунку «Beauty Time»

Коректність реалізації перевірено на рівні основних користувацьких сценаріїв, зокрема бронювання, виявлення конфліктів часу, перенесення і скасування записів, роботи контактної форми та базових правил валідації.

Поєднання структурованого репозиторію, відтворюваного запуску, зрозумілої організації первинного коду та реалізованої прикладної логіки дало змогу отримати працездатний програмний продукт, який відповідає поставленим функціональним вимогам і готовий до демонстрації.

РОЗДІЛ 3 ТЕСТУВАННЯ, РОЗГОРТАННЯ ТА СУПРОВІД ПРОГРАМНОГО ПРОДУКТУ

3.1 Тестування програмного забезпечення

Тестування вебзастосунку Beauty Time виконувалося з метою перевірки стабільності основних функціональних сценаріїв, коректності бізнес-логіки запису клієнтів до салону краси, якості програмного коду та здатності системи працювати в умовах підвищеного навантаження. Оскільки застосунок реалізовано на базі фреймворку Django, основна увага була приділена перевірці серверної логіки, роботи ORM-рівня, форм, представлень, маршрутизації, кешування та доступності основних сторінок [18].

Об’єктом тестування є вебзастосунок салону краси, який містить функції реєстрації та авторизації користувачів, перегляду послуг, створення запису, перегляду власних записів, скасування та перенесення запису, роботи з акційними пропозиціями, пакетними послугами та контактною формою.

Для забезпечення якості програмного продукту було використано такі інструменти:

Таблиця 3.1– Інструментарій для автоматизації тестування та контролю якості коду

Інструмент	Призначення
Django TestCase	Перевірка функціональності застосунку на рівні моделей, форм, представлень і маршрутів
Pytest + pytest-django	Автоматизований запуск тестів Django у зручному форматі
Flake8	Статична перевірка стилю та якості Python-коду
Locust	Підготовка сценаріїв навантажувального тестування
Django Cache Framework	Перевірка кешування даних і стійкості до тимчасової недоступності бази даних

План тестування передбачав перевірку функціональних і нефункціональних вимог. До функціональних вимог належать реєстрація користувача, вхід у систему, створення запису, перевірка доступних часових слотів, скасування та перенесення запису, робота з акціями й пакетними пропозиціями, а також надсилання

повідомлення через контактну форму. До нефункціональних вимог належать стабільність роботи, базова продуктивність, контроль якості коду та стійкість до тимчасових проблем із доступом до бази даних.

Критеріями початку тестування були: наявність реалізованої основної функціональності, виконані міграції бази даних, налаштоване тестове середовище та підготовлені тестові дані. Критеріями завершення тестування вважалися успішне проходження автоматизованих тестів, відсутність критичних помилок у Flake8, коректна робота health endpoint та успішна перевірка кешування.

Тестове середовище включало Python 3.14, Django 6, SQLite для локального тестування, PostgreSQL для production-конфігурації, Redis як production-кеш за наявності змінної REDIS_URL, а також локальний кеш LocMemCache для середовища розробки.

Таблиця 3.2 – Результати виконання тестових сценаріїв системи

ID	Тип тесту	Сценарій	Очікуваний результат	Статус
TC-01	Функціональний	Відкрити публічні сторінки сайту	Сторінки доступні, код відповіді 200	Виконано
TC-02	Функціональний	Зареєструвати нового користувача	Користувач створюється, виконується перенаправлення на сторінку входу	Виконано
TC-03	Функціональний	Спроба реєстрації з повторним email	Система показує помилку валідації	Виконано
TC-04	Функціональний	Вхід користувача за email і паролем	Користувач авторизується	Виконано
TC-05	Інтеграційний	Створення запису авторизованим користувачем	Запис зберігається в базі даних	Виконано
TC-06	Негативний	Запис на минулу дату	Система блокує створення запису	Виконано
TC-07	Негативний	Запис на зайнятий часовий слот	Система показує помилку про недоступний час	Виконано

Продовження таблиці 3.2

TC-08	Функціональний	Перегляд власних записів	Користувач бачить тільки свої записи	Виконано
TC-09	Функціональний	Скасування запису	Статус запису змінюється на скасований	Виконано
TC-10	Функціональний	Перенесення запису	Дата або час запису оновлюються	Виконано
TC-11	Інтеграційний	Перевірка пакетної пропозиції	Тривалість запису збільшується відповідно до пакета	Виконано
TC-12	Інтеграційний	Контактна форма	Повідомлення зберігається та обробляється системою	Виконано
TC-13	Нефункціональний	Перевірка кешування списку послуг	Повторний запит виконується без звернення до БД	Виконано
TC-14	Нефункціональний	Відмова БД при наявності stale cache	Система повертає кешовані публічні дані	Виконано
TC-15	Статична якість	Перевірка Flake8	Критичних порушень стилю не виявлено	Виконано

Для візуалізації, систематизації та трекінгу результатів тестування було використано систему управління тестуванням TestRail.

Результати успішного виконання створеного тестового набору в середовищі TestRail представлено на рисунку 3.1.

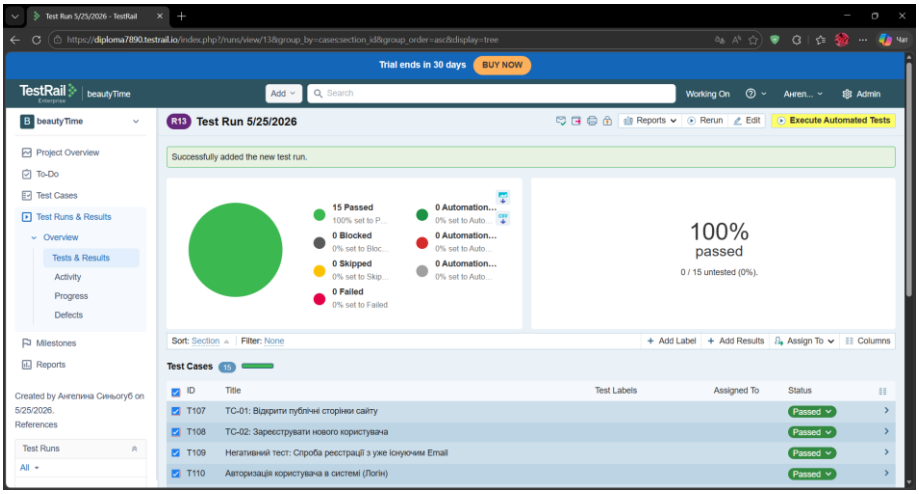

Рисунок 3.1 – Інтерфейс системи TestRail із результатами виконання тестового набору для вебзастосунку «Beauty Time»

Як видно з рисунка 3.1, тестовий набір містить 15 тест-кейсів, які покривають основні функціональні, інтеграційні та нефункциональні вимоги. Усі 15 тестів (100%) мають статус Passed (успішно пройдено), що свідчить про коректність роботи бізнес-логіки, відсутність критичних дефектів у реалізованих модулях та готовність системи до демонстрації.

Автоматизовані тести розміщено у файлі core/tests.py. Після додавання тестів кешування загальна кількість тестів становить 32.

Окремо було налаштовано статичну перевірку якості коду за допомогою Flake8. Для цього створено файл .flake8, у якому виключено службові директорії, залежності, міграції та статичні файли. Такий підхід дозволяє перевіряти саме програмний код застосунку, не змішуючи його з автоматично згенерованими або сторонніми файлами. Після налаштування Flake8 перевірка завершилась без помилок.

Для навантажувального тестування підготовлено сценарій Locust у файлі tests/load/locustfile.py. Він моделює поведінку звичайного користувача: відкриття головної сторінки, сторінки послуг, контактної сторінки та health endpoint. Такий сценарій дозволяє перевірити, як застосунок реагує на одночасні запити до найчастіше використовуваних публічних сторінок.

Окрему увагу приділено кешуванню. У застосунку реалізовано кешування публічних даних: списку послуг, активних акцій, пакетних пропозицій, звільнених слотів та last-minute пропозицій. Кешування виконано через Django Cache Framework. У локальному середовищі використовується LocMemCache, а в production-конфігурації передбачено Redis через змінну REDIS_URL.

Кешування розміщене на рівні доступу до даних у core/selectors.py, тобто між представленнями та ORM-запитами. Це дозволяє зменшити кількість звернень до бази даних для публічних сторінок. Крім того, реалізовано механізм stale cache: якщо база даних тимчасово недоступна, але в кеші є попередньо збережені дані, система може повернути ці дані користувачу замість повного збою сторінки. Водночас критичні операції, такі як створення або перенесення запису, не виконуються лише з кешу, оскільки потребують актуального стану бази даних.

Таким чином, проведене тестування підтвердило стабільну роботу основних сценаріїв вебзастосунку Beauty Time. Автоматизовані тести забезпечують повторюваність перевірки, Flake8 контролює якість коду, Locust дозволяє виконувати навантажувальні перевірки, а кешування підвищує стійкість системи до тимчасових проблем із доступом до бази даних. Отримані результати свідчать, що програмний продукт відповідає базовим функціональним і нефункціональним вимогам.

3.2 Побудова інфраструктури постачання програмного забезпечення

У межах розроблення вебзастосунку Beauty Time було побудовано базову інфраструктуру постачання програмного забезпечення, яка забезпечує відтворюваний запуск системи, автоматизовану підготовку середовища, контроль залежностей та можливість розгортання застосунку в production-середовищі. Оскільки система реалізована на базі Django, інфраструктура постачання орієнтована на контейнеризований запуск Python-застосунку разом із базою даних PostgreSQL та кеш-сервером Redis.

Основною метою побудови інфраструктури було забезпечення однакового запуску програмного продукту в різних середовищах: локальному середовищі розробки, тестовому середовищі та production-середовищі на VPS-сервері. Для цього в проєкті використано Docker та Docker Compose [19], які дозволяють описати компоненти системи у вигляді конфігураційних файлів та забезпечити відтворюваність процесу розгортання.

До складу інфраструктури входять такі компоненти:

- Django web application – основний серверний застосунок;
- Gunicorn – WSGI-сервер для запуску Django у production-середовищі;
- PostgreSQL – основна реляційна база даних;
- Redis – кеш-сервер для збереження публічних даних та зменшення навантаження на базу даних;
- WhiteNoise – засіб обслуговування статичних файлів;
- Docker – інструмент контейнеризації;

— Docker Compose – засіб оркестрації багатокомпонентної системи;
— .env – файл конфігурації змінних середовища;
— Healthcheck – механізм автоматичної перевірки працездатності контейнерів.

Для контейнеризації застосунку використано Dockerfile на базі образу python:3.12-slim, що дозволяє зменшити розмір контейнера та забезпечити необхідне середовище для запуску Django-проєкту. У процесі збірки контейнера інсталюються системні залежності та Python-бібліотеки, після чого копіюється код проєкту та формується готове середовище виконання.

Процес запуску застосунку винесено в окремий стартовий сценарій контейнера. Під час запуску автоматично виконуються міграції бази даних, підготовка статичних файлів та запуск застосунку через Gunicorn. Такий підхід забезпечує повторюваність запуску та автоматичну підготовку середовища після створення контейнера.

Оркестрація компонентів реалізована у файлі docker-compose.yml. У ньому описано сервіси вебзастосунку, бази даних та кеш-сервера. Для сервісу вебзастосунку налаштовано залежності від PostgreSQL та Redis. Запуск контейнера застосунку виконується лише після успішного проходження healthcheck залежними сервісами, що зменшує ризик помилок під час старту системи.

Для PostgreSQL використовується окремий volume, який забезпечує збереження даних між перезапусками контейнерів. Це дозволяє уникнути втрати даних користувачів, записів, послуг та інших сутностей системи під час оновлення або повторного запуску контейнерів [20].

Конфігурація застосунку винесена у файл .env, у якому задаються параметри підключення до бази даних, Redis, Gunicorn та production-налаштування Django. Такий підхід відповідає принципу розділення коду та конфігурації, що спрощує перенесення застосунку між різними середовищами.

Окремо реалізовано endpoint /health/, який використовується для перевірки працездатності застосунку. Він перевіряє доступність бази даних та повертає

інформацію про стан системи. Для вебконтейнера налаштовано автоматичний healthcheck, який періодично перевіряє доступність цього endpoint.

У межах проєкту підготовлено базові елементи, які можуть бути використані в CI/CD-процесі: автоматизовані тести, перевірка стилю коду, сценарій навантажувального тестування, Docker-збірка та healthcheck застосунку.

Таким чином, pipeline постачання програмного забезпечення включає перевірку коду, запуск тестів, збірку контейнера, запуск залежних сервісів, виконання міграцій бази даних, підготовку статичних файлів, запуск Gunicorn та перевірку працездатності системи.

Узагальнену послідовність процесів постачання та розгортання вебзастосунку «Beauty Time» наведено на рис. 3.2.

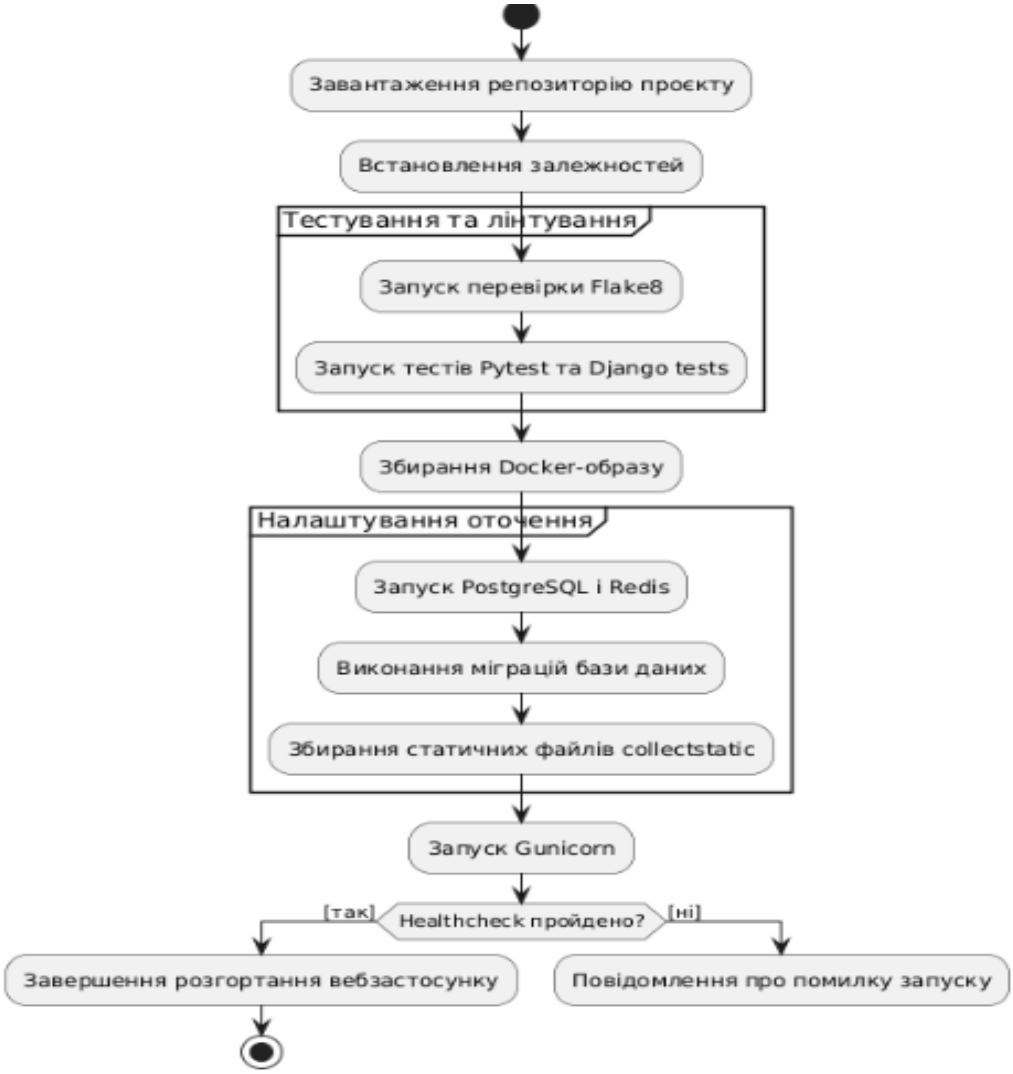

Рисунок 3.2 – Workflow процесу постачання та розгортання вебзастосунку «Beauty Time»

Для розгортання системи у production-середовищі рекомендовано використовувати Docker Compose разом із Nginx у ролі reverse proxy. У такій схемі Nginx приймає HTTPS-запити, обробляє TLS-сертифікати та передає запити до контейнера вебзастосунку. Сам застосунок працює за Gunicorn, а статичні файли обслуговуються через WhiteNoise.

У production-конфігурації також передбачені механізми безпеки Django, зокрема налаштування ALLOWED_HOSTS, CSRF_TRUSTED_ORIGINS, secure cookies, HTTPS redirect та HSTS. Це дозволяє адаптувати застосунок до реального середовища розгортання та підвищити безпеку передачі даних.

У даній роботі не використовувалася Kubernetes-інфраструктура, оскільки для масштабу вебзастосунку Beauty Time достатнім є використання Docker Compose. Водночас архітектура контейнеризації дозволяє в майбутньому перенести систему до Kubernetes та масштабувати окремі сервіси незалежно один від одного.

Подальшим напрямом розвитку інфраструктури може стати впровадження повноцінного CI/CD pipeline, наприклад на базі GitHub Actions, із автоматичним запуском тестів, перевіркою якості коду та публікацією Docker-образів після оновлення репозиторію.

Отже, побудована інфраструктура постачання програмного забезпечення забезпечує відтворюваний запуск вебзастосунку, автоматичну підготовку середовища, контроль працездатності сервісів та базову готовність до production-розгортання. Використання Docker, Docker Compose, PostgreSQL, Redis, Gunicorn та механізмів healthcheck дозволяє розглядати проєкт Beauty Time як інженерно підготовлений програмний продукт, придатний для стабільної експлуатації та подальшого розвитку.

3.3 Супровід, моніторинг та експлуатація програмного продукту

Після реалізації та розгортання вебзастосунок Beauty Time потребує супроводу, моніторингу та планової експлуатації. Цей етап є важливою частиною

життєвого циклу програмного забезпечення, оскільки дає змогу контролювати працездатність системи, своєчасно виявляти помилки, аналізувати навантаження та забезпечувати стабільну роботу сервісу для кінцевих користувачів.

Вебзастосунок Beauty Time призначений для запису клієнтів до салону краси, перегляду послуг, роботи з акціями, пакетними пропозиціями, контактною формою та особистими записами користувача. Тому під час експлуатації особливу увагу необхідно приділяти доступності системи, коректній роботі бази даних, стабільності механізму запису, швидкості відповіді сторінок і збереженню даних користувачів.

У проєкті реалізовано базові механізми контролю працездатності. Основним технічним засобом перевірки стану системи є endpoint /health/. Він виконує перевірку доступності бази даних і повертає інформацію про стан застосунку. Якщо підключення до бази даних працює коректно, endpoint повертає успішну відповідь. Якщо база даних недоступна, система повертає помилку з відповідним HTTP-статусом.

У Docker Compose для контейнера вебзастосунку налаштовано автоматичний healthcheck, який періодично звертається до /health/. Також healthcheck налаштовано для PostgreSQL та Redis. Це дає змогу контролювати не лише сам Django-застосунок, а й залежні сервіси, без яких система не може повноцінно працювати.

Таблиця 3.3 - Специфікація компонентів інфраструктури для контролю працездатності

Компонент	Що контролюється	Значення для системи
Django web application	Доступність сторінок, HTTP-статуси, помилки	Контроль працездатності основного застосунку
PostgreSQL	Доступність БД, збереження даних, підключення	Стабільність роботи записів, користувачів і послуг
Redis	Доступність кешу, коректність кешування	Зменшення навантаження на базу даних
Gunicorn	Кількість worker-процесів, таймаути, помилки запуску	Стабільна обробка HTTP-запитів
Docker containers	Стан контейнерів, перезапуски, healthcheck	Контроль інфраструктури розгортання
Дисковий простір	Обсяг volume PostgreSQL, статичні файли	Запобігання втраті працездатності через нестачу місця

Для моніторингу системи у промисловому середовищі доцільно використовувати Grafana або аналогічний інструмент візуалізації метрик. У межах кваліфікаційної роботи достатньо визначити ключові метрики, які відображають стан застосунку та відповідають його функціональному призначенню. У поточній реалізації вже підготовлено базові технічні передумови для такого моніторингу: healthcheck endpoint, контейнеризацію, production-конфігурацію, логування та кешування.

Таблиця 3.4 - Перелік та призначення метрик моніторингу системи

Метрика	Призначення
Стан /health/	Перевірка доступності застосунку та бази даних
Кількість HTTP-запитів	Аналіз активності користувачів
Частка відповідей 4xx і 5xx	Виявлення клієнтських і серверних помилок
Середній час відповіді	Оцінка продуктивності сторінок
Кількість активних записів	Бізнес-метрика роботи системи
Кількість скасованих записів	Аналіз поведінки користувачів
Стан PostgreSQL	Контроль основного сховища даних
Стан Redis	Контроль кешування
Використання CPU та RAM	Оцінка навантаження на сервер
Використання дискового простору	Контроль volume бази даних

Приклад логіки дашборду для Beauty Time може включати блоки: "Стан застосунку", "Активність користувачів", "Помилки", "База даних", "Кешування", "Ресурси сервера". Такий поділ дає змогу швидко оцінити, чи проблема пов'язана з кодом застосунку, базою даних, кешем або інфраструктурою.

Окремим напрямом супроводу є журналювання подій. У production-конфігурації Django передбачено логування через стандартний механізм LOGGING, де рівень журналювання задається змінною середовища DJANGO_LOG_LEVEL. Логи виводяться в консоль контейнера, що є типовим підходом для Docker-середовища. Надалі ці логи можуть збиратися централізовано за допомогою Loki, ELK Stack або іншого інструменту збору журналів.

Таблиця 3.5 - Перелік та призначення подій для журналювання

Тип події	Приклад	Призначення
Помилки сервера	Винятки Django, HTTP 500	Діагностика критичних проблем
Помилки бази даних	Недоступність PostgreSQL	Виявлення проблем з інфраструктурою
Помилки кешу	Недоступність Redis	Контроль допоміжних сервісів
Події авторизації	Вхід користувача, помилки входу	Аналіз доступу до системи
Події запису	Створення, скасування, перенесення запису	Контроль ключового бізнес-процесу
Помилки контактної форми	Неможливість відправити повідомлення	Діагностика комунікаційних проблем
Healthcheck-помилки	Невдалий запит до /health/	Контроль працездатності системи

Для стабільної експлуатації також важливо передбачити резервне копіювання бази даних. Оскільки основні дані Beauty Time зберігаються в PostgreSQL, у production-середовищі доцільно регулярно створювати резервні копії volume або виконувати дамп бази даних. Це дасть змогу відновити інформацію про користувачів, записи, послуги, акції та контактні повідомлення у випадку збою сервера або пошкодження даних.

Кешування в системі виконує допоміжну роль. Redis може використовуватися в production-середовищі для зменшення кількості звернень до бази даних при отриманні публічних даних, зокрема списку послуг, акцій, пакетних пропозицій та last-minute слотів. У проєкті також реалізовано механізм stale cache: якщо база даних тимчасово недоступна, система може використати попередньо збережені публічні дані. Це підвищує стійкість застосунку та покращує користувацький досвід.

У процесі супроводу важливо контролювати оновлення програмного продукту. Перед розгортанням нової версії рекомендовано виконувати автоматизовані перевірки, які підтверджують коректність основної функціональності та готовність системи до запуску.

Таблиця 3.6 - Етапи автоматизованого тестування та збірки системи

Етап	Призначення
Запуск тестів	Перевірка основної функціональності
Flake8-перевірка	Контроль якості Python-коду
Перевірка міграцій	Запобігання помилкам структури БД
Docker build	Перевірка збірки контейнера
Healthcheck після запуску	Підтвердження працездатності
Перевірка логів	Виявлення прихованих помилок після старту

Під час подальшого розвитку системи доцільно вести release notes, у яких фіксуються зміни між версіями: додані функції, виправлені помилки, зміни в базі даних, оновлення залежностей та відомі обмеження. Це спрощує супровід системи та дає змогу відстежувати історію розвитку програмного продукту.

Таким чином, супровід, моніторинг та експлуатація Beauty Time базуються на поєднанні healthcheck-механізмів, журналювання, контролю контейнерів, моніторингу бази даних і кешу, а також регулярного тестування перед оновленнями. Хоча в межах роботи не реалізовано повноцінну промислову систему моніторингу з Grafana або Kibana, у проєкті передбачено технічні основи для її подальшого впровадження. Це дає змогу розглядати Beauty Time не лише як реалізований вебзастосунок, а як систему, підготовлену до експлуатації, супроводу та подальшого розвитку в реальному середовищі.

ВИСНОВКИ

У межах кваліфікаційної роботи було розроблено веб-орієнтовану інформаційну систему «Beauty Time», призначену для автоматизації онлайн-запису клієнтів салону краси на послуги. Під час виконання роботи було проаналізовано предметну область, визначено основні проблеми традиційного запису, сформовано функціональні та нефункціональні вимоги, побудовано концептуальну модель системи, спроєктовано попередню архітектуру та реалізовано працездатний програмний продукт.

Результатом роботи став монолітний Django-застосунок клієнт-серверного типу з багатосторінковим інтерфейсом. Система забезпечує перегляд інформаційних сторінок салону, каталогу послуг, актуальних акцій і пакетних пропозицій, реєстрацію та авторизацію користувачів, створення онлайн-запису, динамічне отримання доступних часових слотів, перегляд власних записів, перенесення та скасування бронювань, а також надсилання повідомлень через контактну форму. Адміністрування даних реалізовано через стандартний інтерфейс Django Admin, що дозволяє керувати послугами, записами, повідомленнями, акціями, пакетними пропозиціями та звільненими слотами без створення окремої спеціалізованої адміністративної панелі.

Особливу увагу під час реалізації було приділено центральному сценарію системи — онлайн-запису на послугу. Було реалізовано механізм формування доступних часових слотів у межах робочого часу салону, перевірку конфліктів з активними записами, заборону бронювання минулого часу для поточного дня та повторну серверну перевірку доступності слоту після надсилання форми. Для звичайного запису використовується стандартна тривалість, а для пакетних пропозицій тривалість визначається кількістю процедур у пакеті. У разі відсутності вільного часу на вибрану дату система може запропонувати найближчу доступну дату. Така логіка підвищує надійність бронювання та зменшує ризик накладання записів.

Архітектура програмного продукту побудована відповідно до шаблону MVT, характерного для Django. Моделі описують основні сутності предметної області,

представлення координують обробку HTTP-запитів, шаблони відповідають за відображення сторінок, а окремі сервіси та селектори використовуються для винесення бізнес-логіки й читання даних із представлень. Такий підхід покращує супроводжуваність коду та спрощує подальше розширення системи. Додатково реалізовано кешування публічних даних, підтримку Redis у production-середовищі, сигнали очищення кешу після зміни даних, службовий health-endpoint і контейнеризоване розгортання через Docker.

Поставлені функціональні вимоги загалом було виконано. Система підтримує основні сценарії взаємодії користувача із салоном: перегляд інформації, вибір послуги або пропозиції, створення запису після авторизації, керування власними записами та надсилання звернення. Нефункціональні вимоги також враховано на рівні реалізації: серверна валідація та обмеження доступу підвищують безпеку, централізована логіка бронювання забезпечує надійність, кешування публічних даних покращує продуктивність, а структурована організація коду підвищує придатність системи до подальшого розвитку.

Ефективність реалізованого рішення підтверджується проходженням основних тестових сценаріїв. Було перевірено доступність публічних сторінок, реєстрацію та авторизацію користувачів, створення запису, заборону запису на минулі дати, виявлення конфліктів часових слотів, перенесення й скасування бронювань, створення звільнених слотів, роботу акцій і пакетних пропозицій, контактної форми, кешування та службового endpoint перевірки стану застосунку. Наявність Git-репозиторію, README-файлу українською мовою, тестового звіту та контейнеризованої конфігурації забезпечує відтворюваність і можливість демонстрації програмного продукту.

Разом із тим система має напрями для подальшого розвитку. У майбутньому доцільно реалізувати окремі графіки роботи майстрів, оскільки поточна версія використовує єдиний розклад салону. Також перспективним є додавання ролі майстра, розширеної адміністративної аналітики, email- або SMS-нагадувань про запис, інтеграції зі сторонніми календарями, онлайн-оплати, більш деталізованої системи статусів бронювання та повноцінного механізму керування робочими

днями й вихідними. Окремим напрямом може бути проведення UX-дослідження з реальними користувачами та вдосконалення інтерфейсу на основі отриманого зворотного зв’язку.

Отже, у результаті виконання кваліфікаційної роботи було створено працездатну веб-орієнтовану інформаційну систему для салону краси «Beauty Time», яка автоматизує ключові процеси онлайн-запису та взаємодії з клієнтами. Розроблений програмний продукт відповідає поставленій меті, реалізує визначені вимоги, має зрозумілу архітектуру, підтверджену тестами функціональність і може бути використаний як основа для подальшого розвитку повноцінної системи керування записами салону краси.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Лутц М. Вивчаємо Python. 5-те вид. Київ : Форс Україна, 2023. 1168 с. URL: https://www.oreilly.com/library/view/learning-python-5th/9781449355722/
2. Синьогуб А. Р. Особливості проєктування та реалізації веб-орієнтованих інформаційних систем на основі шаблону MVT. Інформаційні технології та програмна інженерія : зб. матеріалів доп. учасн. наук.-практ. конф. Черкаси : ЧДБК, 2026.
3. Фаулер М. Архітектура корпоративних програмних додатків. Київ : Діалектика, 2020. 544 с. URL: https://martinfowler.com/books/eaa.html
4. Fresha. Salon Software and Booking System | Pricing. URL: https://www.fresha.com/pricing?lang=ru (дата звернення: 04.04.2026).
5. Fresha. Payments. URL: https://www.fresha.com/ru/for-business/features/payments (дата звернення: 04.04.2026).
6. Helsi. Helsi. URL: https://app.helsi.me/ (дата звернення: 04.04.2026).
7. Helsi. Про нас. URL: https://reform.helsi.me/about-us (дата звернення: 04.04.2026).
8. G×Bar. G×Bar — це як салон краси, тільки набагато краще! URL: https://gbar.com.ua/ua (дата звернення: 04.04.2026).
9. G×Bar. Онлайн запис в G×Bar | Обирай свій G×Bar або ж знайди улюбленого майстра. URL: https://gbar.com.ua/online-order (дата звернення: 04.04.2026).
10. Salon Today. Data Confirms Demand for Online Booking. URL: https://www.salontoday.com/603348/data-confirms-demand-for-online-booking (дата звернення: 04.04.2026).
11. Django documentation. Django documentation | Django. URL: https://docs.djangoproject.com/ (дата звернення: 05.04.2026).
12. Mobile Verification Toolkit. MVT Documentation. URL: https://mvt.readthedocs.io/ (дата звернення: 05.04.2026).
13. Mobile Verification Toolkit (MVT). Documentation. URL: https://docs.mvt.re/ (дата звернення: 05.04.2026).

14. Буч Г., Рамбо Д., Джекобсон А. UML. Класичний посібник користувача. Київ : Діалектика, 2019. 496 с. URL: https://www.pearson.com/en-us/subject-catalog/p/uml-user-guide-the/P200000000922/9780321267979
15. Ларман К. Застосування UML і шаблонів проєктування. Київ : Діалектика, 2021. 736 с. URL: https://www.craiglarman.com/wiki/index.php?title=Book_Applying_UML_and_Patterns
16. Річардсон К. Шаблони мікросервісної архітектури. Київ : Фабула, 2022. 544 с. URL: https://microservices.io/book
17. Silberschatz A., Korth H., Sudarshan S. Database System Concepts. 7th ed. New York : McGraw-Hill Education, 2019. 1376 p. URL: https://db-book.com/
18. Date C. J. An Introduction to Database Systems. 8th ed. Boston : Pearson, 2019. 1024 p. URL: https://www.pearson.com/en-us/subject-catalog/p/introduction-to-database-systems-an/P200000003310/9780321197849
19. Chacon S., Straub B. Pro Git. 2nd ed. New York : Apress, 2014. 520 p. URL: https://git-scm.com/book/en/v2
20. Myers G. J., Sandler C., Badgett T. The Art of Software Testing. 3rd ed. Indianapolis : Wiley, 2011. 256 p. URL: https://www.wiley.com/en-us/The+Art+of+Software+Testing%2C+3rd+Edition-p-9781118031964
21. Merkel D. Docker: Lightweight Linux Containers for Consistent Development and Deployment. Linux Journal. 2014. № 239. P. 2–7.
22. PostgreSQL Documentation. URL: https://www.postgresql.org/docs/ (дата звернення: 08.06.2026).
23. Redis Documentation. URL: https://redis.io/docs/latest/ (дата звернення: 08.06.2026).
24. Locust Documentation. URL: https://docs.locust.io/ (дата звернення: 08.06.2026).