

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Відокремлений структурний підрозділ
«Тернопільський фаховий коледж Тернопільського національного
технічного університету імені Івана Пулюя»

Відділення телекомунікацій та електронних систем

(назва відділення)

Циклова комісія комп'ютерних наук

(повна назва циклової комісії)

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи
фахового молодшого бакалавра

на тему: Розробка вебсайту компанії дизайнерських послуг «Pixora»

Виконав: студент 4 курсу, групи 423

спеціальності 122 «Комп'ютерні науки»

(шифр і назва спеціальності)

Кузьма Р.М

(прізвище та ініціали)

Керівник Лісовий В.М

(прізвище та ініціали)

Рецензент

(прізвище та ініціали)

ВІДОКРЕМЛЕНИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ
«ТЕРНОПІЛЬСЬКИЙ ФАХОВИЙ КОЛЕДЖ
ТЕРНОПІЛЬСЬКОГО НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ
ІМЕНІ ІВАНА ПУЛЮЯ»

Відділення телекомунікацій та електронних систем
Циклова комісія комп'ютерних наук
Освітньо-професійний ступінь «фаховий молодший бакалавр»
Спеціальність 122 Комп'ютерні науки
Галузь знань 12 Інформаційні технології

ЗАТВЕРДЖУЮ

Голова циклової комісії
комп'ютерних наук

_____ Галина МАРЦІЯШ

« 02 » березня 2026 року

**З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ**

Кузьмі Роману Миколайовичу

(прізвище, ім'я, по батькові)

1. Тема роботи: Розробка вебсайту компанії дизайнерських послуг «Pihora»

керівник роботи _____ Лісовий Володимир Миколайович _____,
(прізвище, ім'я, по батькові)

затверджені наказом вищого навчального закладу № 4/9-132 від 27.02.2026 р.

2. Строк подання студентом роботи: 19.06.2026 р.

3. Вихідні дані до роботи: технічне завдання на розробку програмного забезпечення, мови програмування: TypeScript, JavaScript, Java 17; API: REST API Spring Boot, WebSocket (STOMP); бібліотеки та фреймворки: React 18, Vite, Axios, Zustand, TanStack React Query, Spring Boot 3.3, Spring Security, JWT, Flyway, PostgreSQL 16, стандарти IEEE 29148-2018, IEEE 29119, ДСТУ 8302:2015.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):

1 Загальний розділ

1.1 Аналітичний огляд існуючих рішень

1.2 Технічне завдання

1.2.1 Найменування та область застосування

1.2.2 Призначення розробки

1.2.3 Вимоги до функціоналу web-сайту

1.2.4 Вимоги до програмної документації

1.2.5 Техніко-економічні показники

1.2.6 Стадії та етапи розробки

1.2.7 Порядок тестування та прийому

2 Розробка технічного та робочого проєкту

2.1 Розробка структури сайту і web-сторінок

2.2 Створення та верстка сторінок сайту

2.3 Розробка структури бази даних сайту

2.4 Програмування сайту

2.4.1 Написання клієнтської частини

2.4.2 Написання admin-частини

2.5 Тестування web-сайту

3 Спеціальний розділ

3.1 Інструкція з розміщення сайту в Інтернеті

3.2 Інструкція з обслуговування та наповнення сайту

3.3 Інструкція з популяризації та підтримки сайту

4 Економічний розділ

4.1 Визначення стадій технологічного процесу та загальної тривалості

проведення НДР

4.2 Визначення витрат на оплату праці та відрахувань на соціальні заходи

4.3 Розрахунок витрат на електроенергію

4.4 Розрахунок суми амортизаційних відрахувань веб-сайту «Ріхога»

4.5 Обчислення накладних витрат

4.6 Складання кошторису витрат та визначення собівартості веб-сайту

«Ріхога»

4.7 Розрахунок ціни робіт

4.8 Визначення економічної ефективності і терміну окупності капітальних

вкладень

5 Охорона праці, техніка безпеки та екологічні вимоги

6 Висновки

Додаткові вказівки: виконання кваліфікаційної роботи із розробкою програмного продукту – вебсайту

5. Перелік графічного матеріалу:

1. Структурна схема програми вебсайту Ріхога
2. Діаграма варіантів використання
3. ER-діаграма бази даних
4. Таблиця техніко-економічних показників

6. Консультанти розділів роботи

Розділ	Ім'я, прізвище та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний розділ	Любов КАЛУШКА		
Охорона праці, техніка безпеки та екологічні вимоги	Генадій ГОРЯЧЕК		

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Отримання і аналіз технічного завдання	20.03.2026	
2	Збір і узагальнення інформації	01.05.2026	
3	Написання першого розділу	15.05.2026	
4	Розробка технічного та робочого проєкту	29.05.2026	
5	Написання спеціального розділу	05.06.2026	
6	Розрахунок економічної частини	08.06.2026	
7	Написання розділу охорони праці	09.06.2026	
8	Виконання графічної частини	10.06.2026	
9	Оформлення пояснювальної записки	11.06.2026	
10	Погодження нормоконтролю	12.06.2026	
11	Попередній захист кваліфікаційної роботи	09.06.2026	
12	Захист кваліфікаційної роботи	23.06.2026	

7. Дата видачі завдання: 05 березня 2026 р.

Студент

(підпис)

Роман КУЗЬМА

Керівник роботи

(підпис)

Володимир ЛІСОВИЙ

ЗМІСТ

АНОТАЦІЯ	4
ВСТУП.....	5
1 Загальний розділ.....	7
1.1. Аналітичний огляд існуючих рішень.....	7
1.2. Технічне завдання	13
1.2.1. Найменування та область застосування	13
1.2.2. Призначення розробки.....	14
1.2.3. Вимоги до функціоналу web-сайту	15
1.2.4. Вимоги до програмної документації.....	17
1.2.5. Техніко-економічні показники	19
1.2.6. Стадії та етапи розробки	19
1.2.7. Порядок тестування та прийому.....	20
2 Розробка технічного та робочого проєкту.....	22
2.2. Створення та верстка сторінок сайту.....	27
2.3. Розробка структури бази даних сайту.....	34
2.4. Програмування сайту.....	40
2.4.1. Написання клієнтської частини.....	42
2.4.2. Написання admin-частини.....	45
2.5. Тестування web-сайту.....	48
3 Спеціальний розділ	56
3.1. Інструкція з розміщення сайту в Інтернеті.....	56
3.2. Інструкція з обслуговування та наповнення сайту.....	61
3.3. Інструкція з популяризації та підтримки сайту	66
4 Економічний розділ	72
4.1. Визначення стадій технологічного процесу та загальної тривалості проведення НДР.....	72

					2026.КВР.122.423.10.00.00 ПЗ						
Зм.	Арк.	№ докум.	Підпис	Дата							
Розроб.		Кузьма Р.М.			Розробка вебсайту компанії дизайнерських послуг «Ріхога» Пояснювальна записка			Лім.	Арк.	Аркушів	
Перевір.		Прізвище І.П.								2	
Реценз.								ВСП ТФК ТНТУ КН-423 м. Тернопіль			
Н. Контр.		Приймак В.А.									
Затверд.											

4.2. Визначення витрат на оплату праці та відрахувань на соціальні заходи	73
4.3. Розрахунок витрат на електроенергію	75
4.4. Розрахунок суми амортизаційних відрахувань розробки вебсайту компанії дизайнерських послуг «Ріхора»	76
4.5. Обчислення накладних витрат.....	77
4.6. Складання кошторису витрат та визначення собівартості	77
4.7. Розрахунок ціни розробки вебсайту компанії дизайнерських послуг «Ріхора».....	78
4.8. Визначення економічної ефективності і терміну окупності капітальних вкладень	78
5 Охорона праці, техніка безпеки та екологічні вимоги	80
5.1. Нормативно-правова база та організація безпечної праці	80
5.2. Характеристика робочого процесу та виробничих факторів	83
5.3. Ергономіка та охорона здоров'я при роботі з персональним комп'ютером	86
5.4. Техніка безпеки при експлуатації обчислювальної техніки.....	88
5.5. Охорона праці та безпека користувачів вебсайту Ріхора	89
5.6. Екологічні вимоги та раціональне використання ресурсів.....	90
5.7. Заходи щодо підвищення безпеки та екологічності (рекомендації).....	92
ВИСНОВКИ	94
ПЕРЕЛІК ПОСИЛАНЬ	97
ДОДАТКИ	99
Додаток А. Лістинг файлу OrderController.java.....	99
Додаток Б. Лістинг файлу SecurityConfig.java.....	104
Додаток В. Лістинг файлу App.tsx.....	108
Додаток Г. Лістинг файлу stompClient.ts.....	111
Додаток Д. Лістинг файлу useChatWebSocket.ts	115

АНОТАЦІЯ

Тема кваліфікаційної роботи: Розробка вебсайту компанії дизайнерських послуг «Ріхога»

Метою кваліфікаційної роботи є розробка full-stack web-застосунку для каталогу дизайнерських послуг, оформлення та супроводу замовлень, виставлення чека на оплату, особистого кабінета клієнта та дизайнера, чату в режимі реального часу, відгуків, сповіщень і адміністративної панелі.

Пояснювальна записка складається з 5 розділів.

У загальній частині описуються аналітичний огляд існуючих рішень та аналіз технічного завдання на вебсайт компанії дизайнерських послуг «Ріхога», сформованого за результатами порівняння платформ Upwork, Fiverr і Freelancehunt.

У другому розділі представлено процес створення програмного продукту, опис та обґрунтування вибору структури та методу організації вхідних та вихідних даних, опис алгоритмів, інформаційних зв'язків, зовнішнє проектування програми, тестування та налагодження програмного забезпечення вебсайту компанії дизайнерських послуг «Ріхога»

У спеціальній частині описані процес розміщення програмного продукту в мережі Інтернет, інструкція з обслуговування та наповнення сайту, а також інструкція з популяризації та підтримки застосунку при локальному запуску та хмарному хостингу (Vercel, Render, Neon).

Розрахунок вартості розробки та економічної ефективності приведено в економічній частині четвертого розділу.

Основні питання охорони праці та техніки безпеки розглянуто в п'ятому розділі записки.

Обсяг пояснювальної записки 116 сторінок.

До складу кваліфікаційної роботи входить графічна частина, яка складається з структурної схеми програми, діаграми варіантів використання, техніко-економічних показників, ER-діаграми бази даних, що виконані на окремих аркушах формату A1

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
						4
Зм.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Сфера надання дизайн-послуг (графічний дизайн, брендинг, UI/UX, ілюстрація) активно розвивається в умовах цифровізації бізнесу та освіти. Замовники та виконавці часто взаємодіють через соціальні мережі та месенджери, що ускладнює облік замовлень, узгодження термінів, фіксацію суми оплати та збереження історії спілкування. Відсутність єдиної інформаційної системи призводить до втрати повідомлень, дублювання даних і зниження прозорості виконання робіт.

Актуальність теми зумовлена потребою в спеціалізованому web-ресурсі, який поєднує каталог послуг, оформлення замовлення, контроль статусів, виставлення чека на оплату, комунікацію між клієнтом і дизайнером та адміністрування довідників. Такий підхід відповідає напрямку підготовки за спеціальністю 122 «Комп'ютерні науки» та типовим завданням фахового молодшого бакалавра - аналізу, проєктуванню, реалізації та тестуванню програмного забезпечення на базі сучасних інформаційних технологій.

Метою кваліфікаційної роботи є розробка web-сайту Ріхога - платформи для замовлення дизайн-послуг, що автоматизує основні бізнес-процеси взаємодії замовника, виконавця та адміністратора.

Об'єкт дослідження - процес організації замовлення дизайн-послуг у веб-середовищі.

Предмет дослідження - методи та засоби проєктування і реалізації клієнт-серверної інформаційної системи з реляційною базою даних.

Для досягнення мети поставлено такі завдання:

- провести аналітичний огляд існуючих рішень (маркетплейси, CMS, канали комунікації);
- сформулювати технічне завдання на web-сайт відповідно до вимог до функціоналу, надійності та умов експлуатації;
- розробити технічний і робочий проєкт (структура сайту, база даних, клієнтська та серверна частини);

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
						5
Зм.	Арк.	№ докум.	Підпис	Дата		

- реалізувати програмний продукт (React [1], Spring Boot [2], PostgreSQL [3]);
- провести тестування та оформити програмну документацію;
- обґрунтувати техніко-економічні показники та вимоги з охорони праці.

Практична значущість полягає в отриманні працездатного web-застосунку, придатного для демонстрації на захисті, навчальної експлуатації в коледжі або як основи для малого дизайн-студію. Система підтримує ролі клієнта, дизайнера та адміністратора, каталог послуг, замовлення, імітацію оплати за чеком, чат і відгуки.

Методи дослідження: аналіз літератури та аналогів, об'єктно-орієнтоване проєктування (UML), реляційна модель даних (ER-діаграма), ітеративна розробка, функціональне та інтеграційне тестування.

Структура пояснювальної записки: вступ, загальний розділ (огляд і технічне завдання), розділ з розробки технічного та робочого проєкту, спеціальний розділ (інструкції), економічний розділ, розділ з охорони праці, висновки, перелік посилань та додатки.

Графічна частина містить структурні та UML-діаграми, ER-модель бази даних і схему головної сторінки сайту.

У розділі 1 наведено аналітичний огляд існуючих рішень і технічне завдання на web-сайт Ріхора. У розділі 2 описано проєктування та реалізацію. У розділі 3 - інструкції з розміщення, обслуговування та підтримки. Економічні показники та охорона праці розглянуті у розділах 4 і 5.

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		6

1 ЗАГАЛЬНИЙ РОЗДІЛ

1.1. Аналітичний огляд існуючих рішень

У підрозділі 1.1 виконано аналітичний огляд програмних продуктів і способів організації замовлень дизайнерських послуг. Огляд містить вступне пояснення проблеми предметної області, перелік областей застосування та задач, які вирішують системи цього класу, короткий аналіз основних груп існуючих рішень (міжнародні маркетплейси, вітчизняні біржі, конструктори сайтів, месенджери), порівняльну характеристику на рисунку. 1.1 та обґрунтування доцільності розробки власного вебсайту Ріхога.

Порівняльна характеристика підходів наведена на рисунку. 1.1.

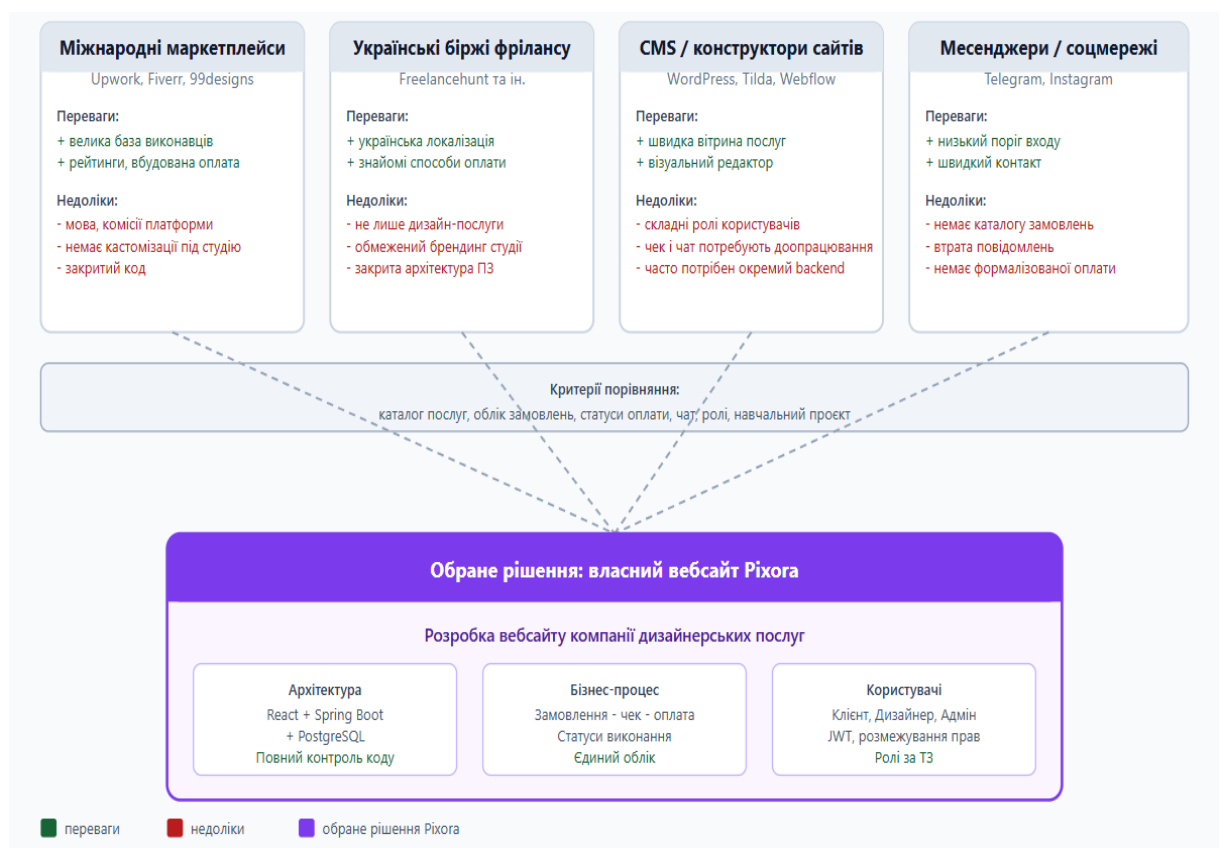


Рисунок 1.1 – Порівняльна схема способів організації замовлень дизайнерських послуг

Компанії, що надають дизайнерські послуги (графічний дизайн, брендинг, інтерфейси користувача, ілюстрація, поліграфія), потребують інструментів для представлення послуг, прийому замовлень, узгодження вимог із замовником та обліку етапів виконання робіт. Області застосування інформаційних систем цього типу охоплюють студії графічного дизайну, невеликі агенції, команди фрілансерів, навчальні проєкти з розробки програмного забезпечення, де необхідно продемонструвати повний цикл створення вебзастосунку.

Задачі, які має вирішувати програмний продукт, узгоджуються з майбутнім технічним завданням (п. 1.2.3): ведення каталогу категорій і послуг; реєстрація та вхід користувачів із ролями клієнт, дизайнер, адміністратор; оформлення замовлення; зміна статусів виконання; виставлення чека та підтвердження оплати; обмін повідомленнями в чаті; збір відгуків і рейтингу; адміністрування довідників і користувачів.

У більшості випадків невеликі студії та студентські проєкти використовують розрізнені канали комунікації - електронну пошту, месенджери, табличні редактори, публікації в соціальних мережах. Такі канали не утворюють єдиної інформаційної системи: статуси замовлень, історія оплат і файли розподілені між сервісами, що ускладнює контроль бізнес-процесу та перевірку відповідності реалізації вимогам кваліфікаційної роботи на захисті.

1.1.1. Міжнародні фріланс-платформи

До найпоширеніших світових рішень належать маркетплейси Upwork, Fiverr, 99designs, Toptal, що об'єднують замовників і виконавців на єдиному майданчику.

Upwork орієнтований на проєктну та погодинну співпрацю. Платформа забезпечує профілі спеціалістів, подання пропозицій, укладення контрактів, ескроу-оплату та систему відгуків. Для дизайнерських послуг доступні категорії графічного дизайну, брендингу, презентацій. Переваги: велика база виконавців, міжнародний ринок, рейтинги, арбітраж спорів. Недоліки для локальної студії та

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
						8
Зм.	Арк.	№ докум.	Підпис	Дата		

дипломного проєкту: мовний бар'єр; комісії платформи; залежність від правил стороннього сервісу; неможливість змінити закриту бізнес-логіку (власний сценарій «виставлення чека - оплата - завершення замовлення»); відсутність доступу до вихідного коду, схеми бази даних і серверного API для аналізу в пояснювальній записці.

Fiverr будується навколо фіксованих пакетів послуг («гігів») із заданою ціною та терміном. За структурою це наближається до каталогу послуг Pihora, однак маршрути замовлення, статуси та чат визначені платформою і недоступні для модифікації студентом-розробником. Недоліки: обмежений брендинг компанії; немає окремої ролі адміністратора студії в тому вигляді, як це передбачено технічним завданням; закритий код.

99designs спеціалізується на конкурсах дизайну: замовник публікує завдання, виконавці подають варіанти, переможець отримує винагороду. Модель відрізняється від прямого замовлення послуги з каталогу, реалізованого в Pihora, проте підтверджує важливість візуальної оцінки робіт і накопичення відгуків.

Міжнародні маркетплейси доцільно розглядати як прототипи за функціями залучення клієнтів, рейтингу та оплати. Вони не замінюють власну розробку, оскільки кваліфікаційна робота вимагає самостійного проєктування архітектури програмного забезпечення та реалізації предметно-орієнтованої моделі замовлення.

1.1.2. Українські біржі фрілансу

На вітчизняному ринку поширені Freelancehunt, Kabanchik.ua, Weblancer та суміжні платформи. Вони охоплюють широкий спектр завдань - від програмування до дизайну та копірайтингу, а не лише послуги дизайнерської студії.

Freelancehunt пропонує розміщення проєктів, конкурси, безпечну угоду, рейтинг фрілансерів, українську локалізацію та звичні для користувачів способи розрахунків. Переваги: мова інтерфейсу, географічна близькість, досвід роботи

					2026.KBP.122.423.10.00.00 ПЗ	Арк.
						9
Зм.	Арк.	№ докум.	Підпис	Дата		

вітчизняної аудиторії. Недоліки у контексті теми диплома: уніфікований каталог і процес замовлення під масовий ринок; обмежена можливість оформити окремий брендований сайт компанії; закрита архітектура - неможливо продемонструвати в записку проєктування бази даних, REST API, WebSocket-чату та ролей користувачів; адаптація під навчальне дослідження структури програмного продукту обмежена.

Kabanchik.ua та аналоги акцентують швидкий пошук виконавця для разових завдань. Для студії з повторюваними замовленнями, портфоліо послуг і внутрішнім обліком це радше канал залучення, аніж повноцінна інформаційна система компанії.

Українські біржі корисні як прототип ідей рейтингу, профілю виконавця та безпечної угоди, але не відповідають формулюванню теми кваліфікаційної роботи - розробки власного вебсайту компанії дизайнерських послуг.

1.1.3. Конструктори сайтів і системи керування вмістом

Клас CMS і конструкторів (WordPress, Tilda, Webflow, Wix) дозволяє швидко створити вітрину - лендінг, каталог, блог, форму зворотного зв'язку без розробки клієнтської частини з нуля.

WordPress із плагінами електронної комерції (WooCommerce) підтримує продаж послуг, кошик, базові статуси замовлень. Переваги: велика екосистема розширень, візуальні теми, адміністрування контенту через панель керування. Недоліки: реалізація трьох ролей з окремими панелями (клієнт, дизайнер, адміністратор), чату в режимі, близькому до реального часу, сценарію «дизайнер виставляє чек - клієнт підтверджує оплату - зміна статусу» потребує значної кількості плагінів або власної серверної надбудови; для диплома зі стеком React + Spring Boot + PostgreSQL архітектура конструктора не демонструє обраний навчальний стек.

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
						10
Зм.	Арк.	№ докум.	Підпис	Дата		

Tilda та Webflow сильні у візуальному оформленні та адаптивній верстці сторінок. Вони підходять для презентаційного сайту та портфоліо, однак слабо підтримують складний облік замовлень, JWT-автентифікацію, REST API і WebSocket без підключення зовнішнього backend.

1.1.4. Месенджери, соціальні мережі та портфоліо-платформи

Telegram, Viber, Instagram, Facebook широко використовуються для першого контакту з клієнтом, демонстрації робіт у стрічці та неформального обговорення деталей замовлення. Переваги: низький поріг входу, звичний для аудиторії канал зв'язку. Недоліки: відсутність централізованого каталогу послуг із фільтрами; замовлення не мають формалізованих статусів оплати та виконання; історія переписки та файлів розпорошена; немає ролей і адміністративної панелі; складно відтворити бізнес-процес для перевірки на відповідність технічному завданню.

На практиці невеликі студії часто поєднують месенджер із табличним обліком: клієнт звертається в Instagram або Telegram, менеджер фіксує суму й термін у Google Sheets або Excel, готові макети пересилає окремими повідомленнями або архівами. Такий гібридний підхід знижує витрати на старт, однак не забезпечує цілісності даних - при зміні статусу оплати, пошуку замовлення за минулий період або перевірці історії листування доводиться вручну зіставляти кілька сервісів. Крім того, персональні дані клієнтів і комерційні умови опиняються в особистих акаунтах співробітників, що ускладнює передачу справ іншому дизайнеру та контроль з боку адміністратора. Вебсайт Ріхога покликаний об'єднати вітрину послуг, облік замовлення, чек, оплату й переписку в межах одного застосунку, тоді як соціальні мережі залишаються лише каналом залучення аудиторії, а не основною системою обліку.

Behance та Dribbble виконують функцію портфоліо дизайнера, а не системи замовлень. Їх доцільно враховувати при проєктуванні блоку портфоліо виконавця

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		11

в Ріхога, однак вони не замінюють корпоративний вебсайт із сервісом оформлення та супроводу замовлення.

Окремо слід зазначити внутрішні засоби обліку (Trello, Notion, Bitrix24): вони підтримують організацію роботи команди, але не орієнтовані на публічний каталог дизайнерських послуг і самообслуговування клієнта через веб-інтерфейс без участі менеджера.

1.1.5. Порівняльна характеристика підходів та обґрунтування власної розробки

Для систематизації результатів огляду застосовано критерії, що впливають із майбутнього технічного завдання на Ріхога: наявність каталогу послуг, обліку замовлень, статусів оплати, чату, ролей користувачів, повного контролю вихідного коду (навчальний проєкт). Порівняльна характеристика розглянутих підходів наведена на рисунок 1.1.

На рисунку 1.1 показано чотири групи існуючих рішень (міжнародні маркетплейси, вітчизняні біржі, конструктори сайтів, месенджери та соціальні мережі) з основними перевагами й недоліками, а також виділено обраний напрям - власний вебсайт Ріхога з повним контролем архітектури, бізнес-процесу «замовлення - чек - оплата», ролями користувачів та JWT-автентифікацією.

За результатами порівняння міжнародні та вітчизняні маркетплейси забезпечують широкий вибір виконавців, рейтинги й вбудовані механізми оплати, але для локальної дизайн-студії та навчального проєкту обмежують контроль над бізнес-логікою, брендингом і доступом до програмної реалізації. Конструктори сайтів прискорюють створення вітрини, однак сценарій «ролі користувачів - чек - оплата - чат у реальному часі» потребує суттєвих доопрацювань. Месенджери зручні для початкового контакту, але не забезпечують структурованого обліку замовлень і формалізованих статусів.

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
						12
Зм.	Арк.	№ докум.	Підпис	Дата		

1.2. Технічне завдання

1.2.1. Найменування та область застосування

Повна назва об’єкта розробки: вебсайт компанії дизайнерських послуг «Ріхора».

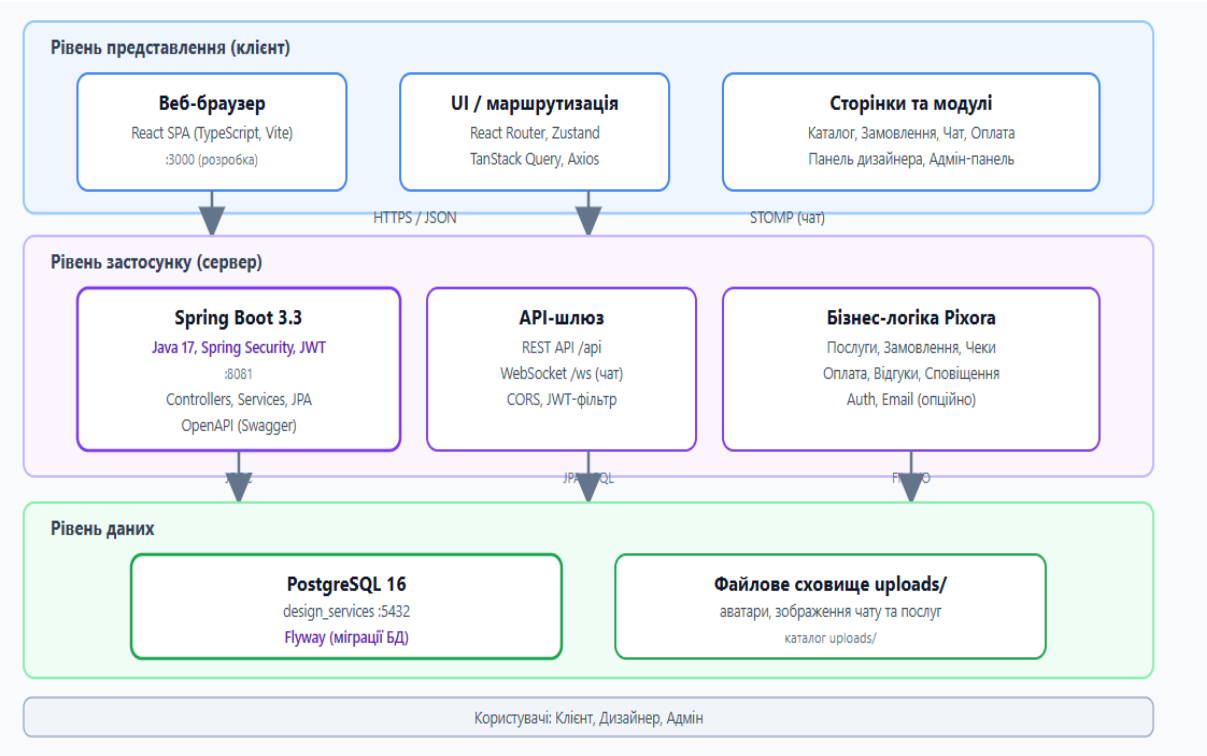
Скорочена назва: Ріхора.

Область застосування: діяльність компаній і студій, що надають дизайнерські послуги; демонстрація кваліфікаційного проєкту в навчальному закладі; основа для подальшого розгортання на хостингу малого бізнесу.

Об’єкт використання: взаємодія клієнта (замовника послуги), дизайнера (виконавця) та адміністратора (керування каталогом і даними системи).

Середовище функціонування: доступ через веб-браузер; сервер - ОС Windows або Linux, JDK 17+, СУБД PostgreSQL 16+.

Загальна архітектура системи показана на рисуюнок 1.2.



Рисуюнок 1.2 – Загальна архітектура вебсайту Ріхора

1.2.2. Призначення розробки

Експлуатаційне призначення - підвищення ефективності роботи компанії дизайнерських послуг за рахунок об'єднання в одному web-ресурсі каталогу послуг, оформлення замовлень, контролю статусів виконання, комунікації з клієнтом та обліку оплати.

Функціональне призначення - забезпечення користувачам засобів для:

- реєстрації, входу та роботи за ролями CLIENT, DESIGNER, ADMIN;
- перегляду та пошуку послуг у каталозі;
- створення та супроводу замовлень;
- виставлення чека на оплату та підтвердження оплати клієнтом;
- обміну повідомленнями в чаті;
- публікації відгуків, перегляду сповіщень, керування профілем;
- адміністрування категорій і контенту платформи.

Орієнтована взаємодія ролей із системою відображається діаграмою варіантів використання (див. рис. 1.3), яка подана на окремому плакаті 2026.KBP.122.423.10.00.00 ДВ в графічній частині дипломного проекту.

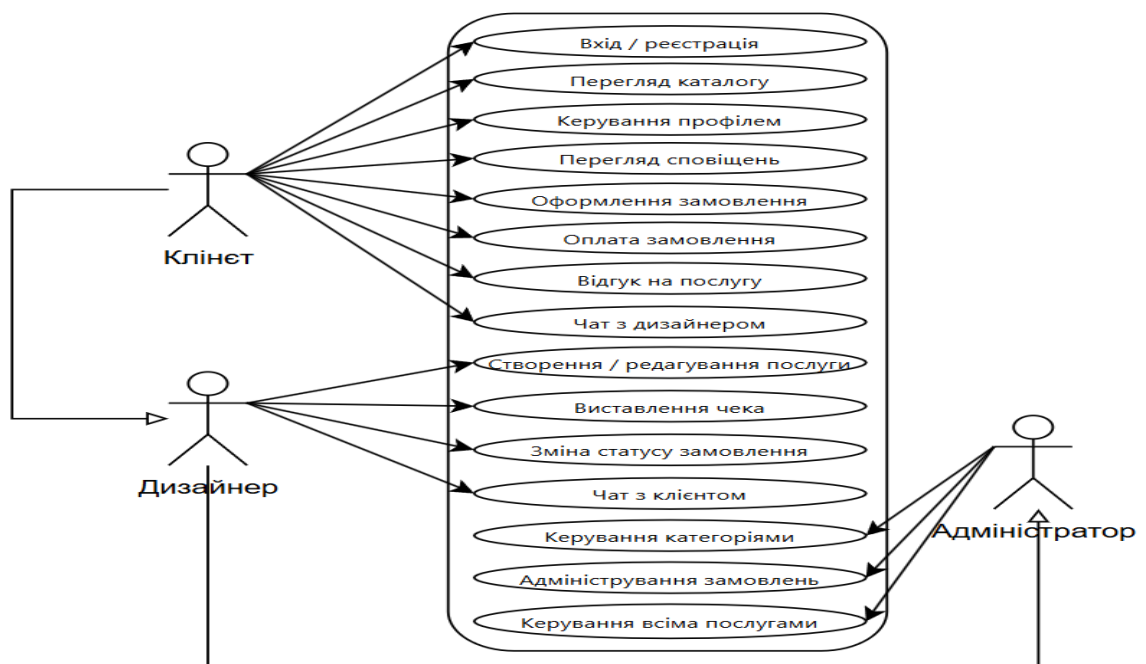


Рисунок 1.3 – UML-діаграма варіантів використання вебсайту Pihoga

1.2.3. Вимоги до функціоналу web-сайту

Вимоги до функціональних характеристик. Вебсайт повинен забезпечувати:

- автентифікація клієнтів, дизайнерів та адміністраторів на основі безпечних токенів (JWT) [4] з розмежуванням ролей; безпечне відновлення доступу до облікового запису за допомогою тимчасового одноразового email-коду, що надсилається на пошту користувача за умови попереднього налаштування SMTP-сервера;
- публічний перегляд каталогу послуг із фільтрацією за категорією та ціною;
- створення, редагування та видалення послуг дизайнером; керування всіма послугами - адміністратором;
- оформлення замовлення клієнтом (вибір послуги, вимоги, дедлайн);
- зміну статусу замовлення: «очікує», «в роботі», «завершено», «скасовано»;
- виставлення чека на оплату з указанням конкретної суми (наприклад, 5000 грн або формат «5к»);
- оплату замовлення клієнтом лише після виставлення чека (навчальна імітація платежу);
- забезпечення оперативного зв'язку між учасниками угоди (клієнтом та дизайнером) з підтримкою обміну текстовими повідомленнями та графічними зображеннями; архітектура модуля будується на комбінації REST API для завантаження історії та WebSocket [5] для миттєвої доставки повідомлень у режимі реального часу;
- створення відгуків на послуги;
- відображення сповіщень;
- панель дизайнера та адміністративну панель.

Життєвий цикл замовлення та оплати показано на рисунку 1.4.

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
						15
Зм.	Арк.	№ докум.	Підпис	Дата		

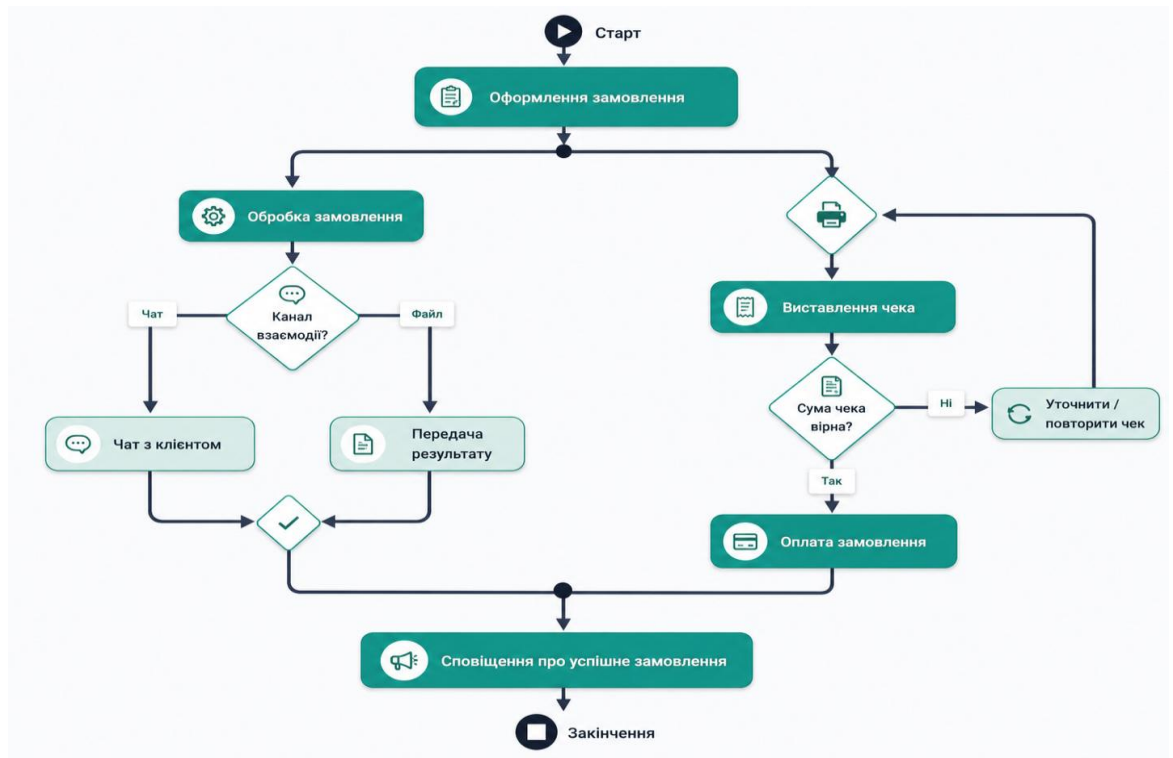


Рисунок 1.4 – Діаграма діяльності: замовлення, чек та оплата в Pihora

Вхідні дані. До основних вхідних даних належать: облікові дані та профіль користувача; параметри послуги (назва, опис, ціна, термін, категорія, зображення); дані замовлення (вимоги, дедлайн); сума чека; повідомлення чату; параметри відгуку; фільтри пошуку в каталозі. Джерело - форми web-інтерфейсу та завантажені файли; частина даних - з бази даних при повторних запитах.

Вихідні дані. Система повинна формувати: сторінки каталогу та картки послуг; списки і деталі замовлень із статусами виконання та оплати; підтвердження виставлення чека та оплати; стрічку повідомлень чату; список сповіщень; відповіді REST API [6] у форматі JSON для клієнтської частини.

Часові характеристики. У навчальному середовищі (локальний ПК або локальна мережа) встановлюється: час відповіді сервера на типовий запит - не більше 2 с; завантаження сторінки каталогу - не більше 3 с; відображення повідомлення чату після відправки - не більше 1 с. При одночасній роботі до 10 користувачів система повинна залишатися працездатною.

Вимоги до надійності. Передбачається: перевірка прав доступу за ролями; валідація вхідних даних на сервері; транзакційне збереження замовлень і статусів оплати; міграції схеми БД (Flyway [10]); зрозумілі повідомлення про помилки користувачу; заборона створення облікового запису адміністратора через публічну реєстрацію; захист API за допомогою JWT [4] та обмеження CORS.

Умови експлуатації. Клієнтська частина: сучасний браузер (Chrome, Firefox, Edge). Сервер: JDK 17+, PostgreSQL 16+ [3], не менше 4 ГБ оперативної пам'яті. Режим роботи - інтерактивний, багатосеансовий. Виведення інформації - на екран монітора; друк окремих сторінок - засобами браузера. Програмне середовище розробки: Node.js 18+ (frontend), Maven 3.8+ (backend).

1.2.4. Вимоги до програмної документації

Склад документації визначається методичними вказівками до кваліфікаційної роботи.

Обов'язковий склад:

- пояснювальна записка з темою «Розробка вебсайту компанії дизайнерських послуг „Ріхора“» (титул, завдання, анотація, зміст, розділи 1–5, висновки, перелік посилань, додатки);
- графічна частина (не менше 4 аркушів формату A1): архітектура (див. рис. 1.2), UML (див. рис. 1.3, див. рис.1.4), ER-діаграма БД (див. рис. 1.5), структурна схема головної сторінки (див. рис. 1.6), таблиця техніко-економічних показників (з розділу 4);
- опис API (Swagger UI);
- файл README із інструкцією запуску;
- електронний носій (вихідний код, презентація);
- інструкції з розміщення, обслуговування та підтримки - розділ 3.

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
						17
Зм.	Арк.	№ докум.	Підпис	Дата		

Логічну структуру бази даних та зв'язки між основними сутностями системи наведено на рисунку 1.5, яка подана на окремому плакаті 2026.КВР.122.423.10.00.00 СС в графічній частині дипломного проекту.

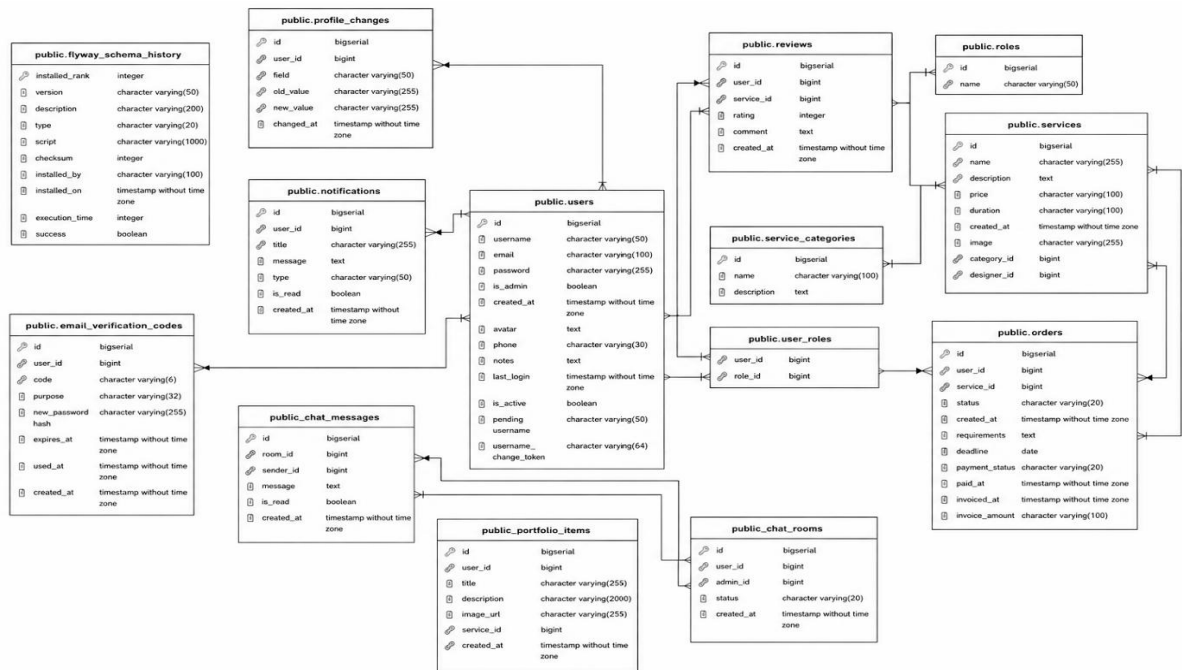


Рисунок 1.5 – ER-діаграма бази даних вебсайту Ріхора

Структуру головної сторінки та розташування її основних блоків наведено на рисунку 1.6.

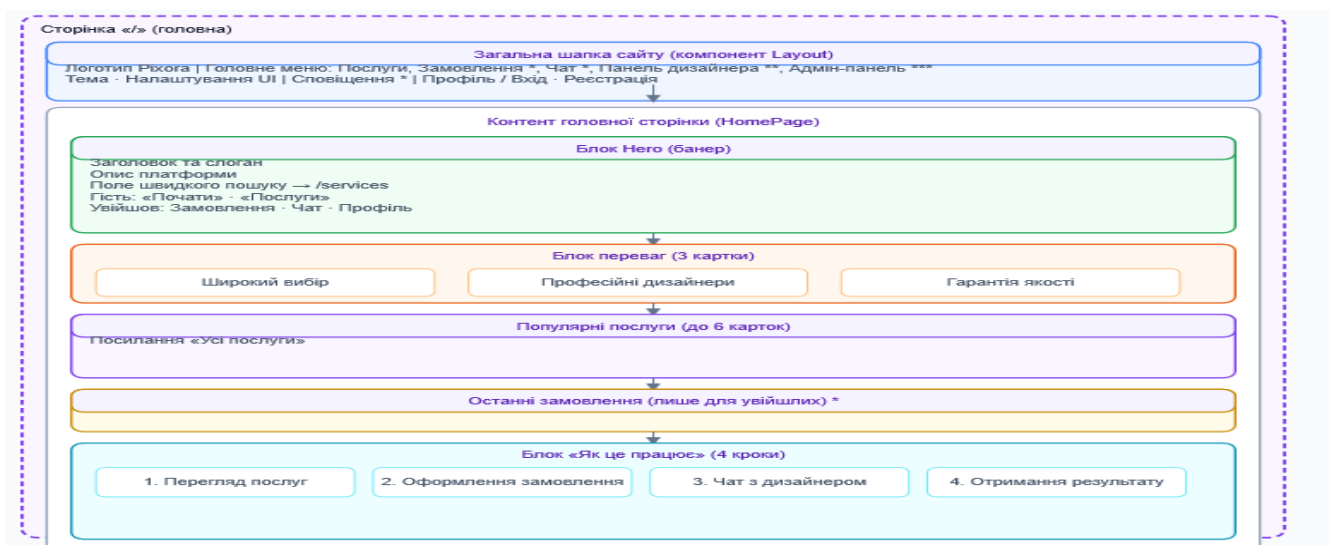


Рисунок 1.6 – Структурна схема головної сторінки вебсайту Ріхора

Вимоги до тексту програм: модулі супроводжуються коментарями щодо призначення; бізнес-логіка зосереджена в сервісному шарі backend; інтерфейс користувача - українською мовою; дотримання узгодженого стилю кодування Java та TypeScript.

1.2.5. Техніко-економічні показники

Розрахунок економічної ефективності та витрат на розробку вебсайту компанії дизайнерських послуг «Ріхога» повинен бути реалізований у відповідності до методичних вказівок для виконання економічного розділу дипломного проекту.

Ключовими параметрами для такого розрахунку повинні бути:

- час розробки веб-додатку – не більше 200 годин;
- кінцева загальна вартість розробки та впровадження веб-додатку не повинна перевищувати 150 000 грн;
- планований термін окупності має бути менше 3 років.

1.2.6. Стадії та етапи розробки

Розробку вебсайту компанії дизайнерських послуг «Ріхога» виконують відповідно до етапів життєвого циклу програмного забезпечення. Послідовність стадій, зміст робіт і очікувані результати наведено в таблиці 1.1.

Таблиця 1.1– Стадії та етапи розробки вебсайту Ріхога

№	Стадія (етап)	Зміст робіт	Результат етапу
1	2	3	4
1	Аналіз і постановка задачі	Огляд аналогів; формулювання вимог до функціоналу та документації; оформлення технічного завдання	Розділ 1 пояснювальної записки (ТЗ), рисунок 1.1

Продовження таблиці 1.1

1	2	3	4
2	Проектування	Архітектура системи; UML-діаграми; проєкт БД; структура головної сторінки; опис API	Рисунок 1.2–1.6; розділ 2.1–2.3
3	Реалізація	Клієнтська частина (React [1], TypeScript [8]); серверна частина (Spring Boot [2]); міграції Flyway [10]; інтеграція модулів	Програмний код; розділ 2.4
4	Тестування та налагодження	Функціональні сценарії за ролями (клієнт, дизайнер, адміністратор); перевірка REST API та чату	Розділ 2.5; протокол/опис перевірок
5	Впровадження	Розгортання БД (Docker); налаштування середовища виконання; інструкція запуску (README)	Розділ 3.1; працездатний вебсайт
6	Документування та захист	Пояснювальна записка; графічна частина; презентація; демонстрація на захисті	Комплект матеріалів КВР

1.2.7. Порядок тестування та прийому

Мета тестування - підтвердити відповідність вебсайту Ріхора вимогам п. 1.2.3 та працездатність у цільовому середовищі (JDK 17, PostgreSQL 16, браузер Chrome або Edge).

Види тестування:

- функціональне - перевірка функцій за ролями CLIENT, DESIGNER, ADMIN;

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
						20
Зм.	Арк.	№ докум.	Підпис	Дата		

- інтеграційне - взаємодія клієнтської частини з REST API [6] та WebSocket [5];
- перевірка збереження даних у PostgreSQL [3] після перезапуску сервера;
- базові перевірки безпеки (доступ без авторизації до захищених операцій, оплата без виставленого чека).

Контрольний сценарій прийому:

1. Реєстрація клієнта та дизайнера.
2. Дизайнер додає послугу в каталог.
3. Клієнт оформлює замовлення.
4. Дизайнер змінює статус на «в роботі» і виставляє чек із сумою 5к; у списку замовлень відображається 5 000 грн і позначка «Сума за чеком».
5. Клієнт виконує оплату; статус оплати - «Оплачено».
6. Перевірка обміну повідомленнями в чаті.
7. Перевірка функцій адміністратора (категорії, перегляд замовлень).

Критерії прийому: контрольний сценарій виконується без критичних помилок; дані коректно зберігаються в БД; пояснювальна записка та графічна частина відповідають методичним вимогам; отримано допуск до захисту.

Детальний чек-лист тестування та план тестування наведено в п. 2.5; результати зведено в таблиці 2.4.

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		21

2 РОЗРОБКА ТЕХНІЧНОГО ТА РОБОЧОГО ПРОЄКТУ

2.1. Розробка структури сайту і web-сторінок

Призначення майбутнього сайту та аналіз технічного завдання:

Першим кроком етапу «Розробка структури сайту і web-сторінок» є визначення призначення ресурсу. За темою кваліфікаційної роботи та технічним завданням (розділ 1.2) створюється корпоративний вебсайт організації - компанії дизайнерських послуг «Ріхога». Це не персональний сайт, не електронна бібліотека й не платформа дистанційного навчання, а комерційно-інформаційний вебресурс із сервісом замовлення послуг: публічна вітрина (каталог, опис компанії), особисті кабінети клієнтів (замовлення, оплата, чат, відгуки) та робочі місця співробітників (дизайнер, адміністратор).

Далі здійснюється аналіз технічного завдання. З ТЗ впливають: три ролі користувачів (рисунок 1.3); перелік функцій (реєстрація, каталог, замовлення, чек і оплата, чат, відгуки, сповіщення, адміністрування - п. 1.2.3); стек React + Spring Boot + PostgreSQL; бізнес-процес «замовлення - чек - оплата» (рисунок 1.4); вимоги до документації та графіки (рисунок 1.2, 1.5, 1.6). На цій основі складено карту сайту - усі розділи, URL, формат інформації на сторінках та правила навігації. Особлива увага в проєкті приділена функціоналу (відповідність прецедентам UML) та зручній навігації між публічною зоною, кабінетом клієнта та панелями персоналу.

Проектування інтерфейсу та дизайн-концепція:

Робота над дизайном виконується поетапно, як передбачено методичними рекомендаціями.

Етап 1 - концепція головної сторінки. На основі ТЗ і спроектованого інтерфейсу визначено: темна сучасна тема (асоціація з digital-дизайном), акцентні кольори (фіолетовий, блакитний), Него-блок з назвою Ріхога, поле пошуку, кнопки

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		22

заклику до дії. Логічна структура головної зафіксована на рисунок 1.6 (Layout, Hero, переваги, популярні послуги, блок «Як це працює»).

Етап 2 - концепції внутрішніх сторінок. Каталог - сітка карток із фільтрами; кабінет - списки замовлень зі статусами; панелі дизайнера та адміна - форми CRUD і таблиці даних. Узгоджено єдину систему компонентів (кнопки, поля, картки, модальні вікна).

Етап 3 - узгодження та затвердження. Враховано українську мову інтерфейсу, адаптивність (desktop / tablet / mobile), контраст і зрозумілі підписи елементів керування. Концепції проходили доопрацювання в процесі розробки React-компонентів.

Підсумок етапу 2.1: затверджені макети (логічні схеми) всіх сторінок - рисунок 1.6 (головна), таблиці 2.1 (повна карта маршрутів), рисунок 1.2–1.3 (архітектура та ролі).

2.1.1. Принципи побудови навігації

Вебсайт Ріхора повинен бути реалізований як односторінковий застосунок (Single Page Application, SPA [6]). Це означає, що після першого завантаження HTML-сторінки подальші переходи між розділами сайту (каталог, замовлення, чат) виконуються без повного перезавантаження документа в браузері: змінюється лише вміст основної області, а спільна шапка залишається на екрані. Такий підхід підвищує швидкодію інтерфейсу та наближає поведінку сайту до сучасних вебзастосунків (особисті кабінети, маркетплейси послуг).

Маршрутизація реалізована бібліотекою React Router v6 [7]. Кожному URL відповідає окремий React-компонент (сторінка). Загальний макет - компонент Layout, який містить навігаційну панель, логотип, меню за ролями, блок сповіщень і меню профілю; усередині Layout через компонент Outlet відображається активна сторінка.

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
						23
Зм.	Арк.	№ докум.	Підпис	Дата		

Архітектура взаємодії «браузер - API - база даних» наведена на рисунок 1.2. Розподіл функцій між ролями користувачів (клієнт, дизайнер, адміністратор) та варіанти використання - на рисунок 1.3. Логічна структура головної сторінки як центрального вузла входу - на рисунок 1.6.

2.1.2. Класифікація сторінок за доступом

Усі web-сторінки повинні бути поділено на три групи за рівнем доступу.

Публічні сторінки доступні без автентифікації. До них належать головна ('/'), каталог послуг ('/services'), картка окремої послуги ('/services/:id'), а також сторінки входу, реєстрації та відновлення пароля. Це забезпечує можливість ознайомлення потенційного клієнта з пропозиціями компанії до створення облікового запису, що відповідає маркетинговій функції корпоративного сайту.

Захищені сторінки потребують дійсного JWT-токена [4]. Компонент `ProtectedRoute` перевіряє наявність токена в сховищі стану (`Zustand` [15]); за його відсутності користувач перенаправляється на сторінку входу. До захищених належать: замовлення, відгуки, чат, профіль, налаштування безпеки, сповіщення, сторінка оплат.

Рольові сторінки додатково перевіряють роль користувача, збережену в JWT після входу. Маршрут `/designer` доступний ролям `DESIGNER` та `ADMIN`; маршрут `/admin` - лише `ADMIN`. На frontend це реалізовано компонентом `ProtectedRoute` з параметром `requiredRole`: при невідповідній ролі користувача перенаправляють на головну або сторінку входу. На backend ті самі обмеження дублюються анотаціями `@PreAuthorize` [14] у контролерах, тому обійти захист лише зміною URL у браузері неможливо. Це запобігає несанкціонованому доступу до панелей керування навіть при наявності валідного токена клієнта: клієнт може переглядати каталог і свої замовлення, але не керувати чужими послугами та глобальними налаштуваннями платформи.

Повний перелік маршрутів, компонентів і призначення сторінок наведено в таблиці 2.1

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		24

Таблиця 2.1 – Структура web-сторінок вебсайту Ріхора

Маршрут	Сторінка (компонент)	Призначення	Доступ
/	HomePage	Головна: банер, пошук, популярні послуги, блок «Як це працює»	Усі
/login, /register, /forgot-password	LoginPage, RegisterPage, ForgotPasswordPage	Вхід, реєстрація, відновлення пароля	Гості
/services	ServicesPage	Каталог послуг, фільтрація та пошук	Усі
/services/:id	ServiceDetailPage	Картка послуги, оформлення замовлення	Усі / авторизовані
/orders	OrdersPage	Замовлення клієнта, оплата	CLIENT+
/reviews	ReviewsPage	Відгуки	Авторизовані
/profile, /security, /notifications	ProfilePage, SecuritySettingsPage, NotificationsPage	Профіль, безпека, сповіщення	Авторизовані
/payments	PaymentsPage	Історія оплат	Авторизовані
/designer	DesignerDashboard	Послуги та замовлення дизайнера, виставлення чека	DESIGNER, ADMIN
/admin	AdminDashboard	Категорії, усі послуги та замовлення, модерація	ADMIN

2.1.3. Логіка переходів між розділами

Типовий сценарій клієнта на рівні структури сайту виглядає так: головна сторінка → каталог → деталі послуги → (після входу) оформлення замовлення → розділ «Мої замовлення» → оплата після виставлення чека → чат з виконавцем → відгук. Кожен крок відповідає окремому маршруту, що спрощує навчання користувача та тестування.

Сценарій дизайнера: вхід → панель дизайнера → перегляд замовлень по своїх послугах → зміна статусу на «в роботі» → виставлення чека → спілкування в чаті → завершення замовлення.

Сценарій адміністратора: вхід → адмін-панель → керування категоріями та глобальним каталогом → моніторинг усіх замовлень і відгуків → за потреби втручання в статус або чек.

Навігаційне меню в Layout динамічно формується: для неавторизованого користувача відображаються «Увійти» та «Реєстрація»; для авторизованого - пункти «Послуги», «Замовлення», «Чат», а для відповідних ролей - «Панель дизайнера» та «Адмін панель». Таким чином структура сайту відображає бізнес-модель компанії: публічна вітрина + персоналізований кабінет + операційні панелі персоналу.

2.1.4. Зв'язок структури сайту з бізнес-процесом

Структура web-сторінок узгоджена з діаграмою діяльності (рисунок 1.4) і відображає повний орієнтований життєвий цикл замовлення в Ріхора - від вибору послуги до завершення робіт і оплати. Перший крок процесу - ознайомлення з пропозицією: для цього призначені публічні сторінки головної та каталогу (/ , /services, /services/:id). Оформлення замовлення відповідає сторінці деталей послуги (форма вимог і дедлайну) та подальшому перегляду в розділі «Мої замовлення» (/orders), де клієнт бачить операційний статус (pending, in_progress, completed, cancelled).

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		26

Другий блок процесу - обробка замовлення виконавцем. Зміна статусу на «в роботі» та підготовка до розрахунку реалізовані в панелі дизайнера (/designer) або адміністратором (/admin). Виставлення чека - окрема дія в інтерфейсі (модальне вікно з указанням суми); після неї у замовленні з'являються поля invoice_amount та payment_status = invoiced, що відповідає ромбовим перевіркам на рисунок 1.4.

Третій блок - фінансовий. Оплата доступна клієнту лише на сторінках замовлень і оплат (/orders, /payments) після виставлення чека; до цього моменту кнопка «Оплатити» прихована, що узгоджує UI з guard-умовою «чек виставлено?». Підтвердження оплати переводить замовлення в стан paid і логічно завершує грошову частину процесу, після чого можливі чат (/chat) та відгук (/reviews) як супровідні, але важливі для якості сервісу етапи.

Тому технічний проєкт інтерфейсу не дублює бізнес-логіку випадково і не розносить один процес по непов'язаних розділах: кожна дія на діаграмі діяльності має зрозумілу «точку входу» на сайті, а критичні переходи (чек → оплата) контролюються і на сервері (OrderService), і в інтерфейсі. Затверджений процес «замовлення - чек - оплата» таким чином є основою карти сайту з таблиці 2.1, а не лише описом у розділі 1.

2.2. Створення та верстка сторінок сайту

Середовище створення web-сторінок:

Сторінки сайту створюватимуться у текстовому редакторі коду (Visual Studio Code / intellij IDEA. Розмітка записується у файлах TypeScript/Jsx (.tsx), стилі - класами Tailwind CSS і глобальними правилами в index.css. Збірка виконується засобом Vite; результат етапу верстки - готовий пакет web-сторінок у каталозі frontend/dist (HTML + JS + CSS після npm run build), що відповідає вимозі методички щодо зверстаних шаблонів у сучасній SPA-формі.

У класичному розумінні «верстка» - це HTML і CSS окремих статичних сторінок. У проєкті Ріхора інтерфейс побудований як компонентний UI на React 18

					2026.KBP.122.423.10.00.00 ПЗ	Арк.
						27
Зм.	Арк.	№ докум.	Підпис	Дата		

[1] і TypeScript [8], а стилізація виконана засобами Tailwind CSS [16] (утилітарні класи) з адаптивними префіксами `sm:`, `md:`, `lg:`. Збірка проєкту - Vite 5 [9], що забезпечує швидку розробку та оптимізований production-збірку в каталог `dist/`.

Такий підхід є стандартом для сучасних корпоративних сайтів із розвиненим функціоналом і не суперечить вимозі «створення web-сторінок», оскільки кожен маршрут з таблиці 2.1 відображає окрему логічну «сторінку» для користувача.

2.2.1. Загальний дизайн і доступність

Візуальна концепція Ріхога - темна тема з акцентами фіолетового та блакитного спектру, що асоціюється з креативністю та цифровими послугами.

Застосовано:

- єдину систему компонентів: кнопки (`btn`, `btn-primary`), поля вводу (`input`), картки (`card`, `service-tile`);
- іконки Lucide React для уніфікації піктограм;
- повідомлення користувачу через toast-сповіщення (`notify`);
- плавну появу блоків при прокрутці (ScrollReveal);
- перемикач світлої/темної теми (ThemeToggle);
- модальне вікно кастомізації кольорів і фону (UISettingsModal) - додаткова вимога до зручності інтерфейсу в навчальному проєкті.

Інтерфейс повністю українською мовою, що відповідає вимогам ТЗ (розділ 1.2.3) та цільовій аудиторії компанії в Україні.

2.2.2. Головна сторінка (HomePage)

Головна сторінка - вітрина бренду Ріхога. Структурно вона складається з таких секцій (відповідає рисунок 1.6):

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		28

1. Hero-блок - заголовок «Ріхога», слоган, короткий опис платформи, форма швидкого пошуку з переходом у каталог (`/services?search=...`). Для неавторизованих користувачів - кнопки «Почати зараз» (реєстрація) та «Переглянути послуги».

2. Блок переваг - три картки: широкий вибір послуг, професійні дизайнери, гарантія якості через відгуки.

3. Популярні послуги - динамічне завантаження до шести останніх послуг з API; кожна картка веде на детальну сторінку.

4. Останні замовлення - відображається лише для авторизованих (персоналізація кабінету).

5. «Як це працює» - чотири кроки з інтерактивними переходами (каталог, замовлення, чат, результат).

Така структура поєднує маркетингові та функціональні цілі: залучення, пояснення сервісу, швидкий вхід у каталог.

2.2.3. Каталог і картка послуги

ServicesPage реалізує пошук і фільтрацію: за текстом, категорією, діапазоном цін (параметри передаються в REST API). Результати відображаються сіткою карток із зображенням, назвою, ціною та тривалістю.

ServiceDetailPage показує повний опис послуги. Для авторизованого клієнта доступна форма замовлення: текст вимог, бажаний дедлайн. Після відправки створюється запис у таблиці `orders` зі статусом `pending`.

Верстка цих сторінок адаптивна: на мобільних пристроях сітка переходить у один стовпець, меню ховається в «гамбургер» (мобільне меню Layout).

2.2.4. Особистий кабінет клієнта

OrdersPage - центральна сторінка клієнта після оформлення замовлення.

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
						29
Зм.	Арк.	№ докум.	Підпис	Дата		

Відображається статус виконання ('pending', 'in_progress', 'completed', 'cancelled') та фінансовий стан ('unpaid', 'invoiced', 'paid'). Після виставлення чека дизайнером клієнт бачить суму ('invoice_amount') і може підтвердити оплату (навчальна імітація без банківського API).

ChatPage - інтерфейс чату з підтримкою тексту та зображень. Повідомлення оновлюються в реальному часі через WebSocket/STOMP [5] і [13] (підключення через проксі Vite у режимі розробки). У заголовку чату відображається стан з'єднання («Підключено» / «Очікування з'єднання»).

ProfilePage та SecuritySettingsPage - редагування профілю, зміна пароля з підтвердженням коду на email.

ReviewsPage - створення та перегляд відгуків на послуги.

NotificationsPage - історія сповіщень про зміну статусів замовлень та інші події.

2.2.5. Панелі дизайнера та адміністратора

Панелі дизайнера та адміністратора - це робочі інтерфейси для ролей DESIGNER і ADMIN. Вони доступні лише після автентифікації (маршрути /designer та /admin) і побудовані як окремі сторінки React з вертикальною композицією секцій: заголовок (Hero), блок зведеної статистики, далі - робочі зони з формами та списками даних.

Сторінка DesignerDashboard (компонент DesignerDashboard.tsx, маршрут /designer) - робоче місце виконавця. У верхній частині розміщено заголовок «Панель дизайнера» і короткий підзаголовок про керування послугами та замовленнями. Нижче - чотири інформаційні картки статистики: кількість власних послуг, усіх замовлень по цих послугах, активних (in_progress) та завершених (completed). Дані підвантажуються з REST API через TanStack React Query (запити my-services та my-orders), під час завантаження показується індикатор, при помилці сервера - повідомлення з підказкою перевірити backend.

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		30

Основний вміст плунується розбити на дві колонки (адаптивна сітка `grid md:grid-cols-2`). У секції «Мої послуги» дизайнер переглядає лише власні пропозиції (параметр `mine: true` у пошуку послуг), створює нову послугу або редагує існуючу. Форма містить поля назви, опису, ціни, тривалості, категорії та завантаження зображення (компонент `ServiceImageField`). Валідація перед відправкою виконується модулем `serviceForm.ts` (обов'язкові назва, опис, ціна). Після успішного збереження списки оновлюються автоматично (`invalidateQueries`), користувач отримує `toast`-повідомлення українською.

У другій колонці - «Замовлення по моїх послугах»: для кожного замовлення відображаються назва послуги, дата, статус виконання та фінансовий стан оплати. Дизайнер може перевести замовлення в роботу (`in_progress`), завершити (`completed`) або скасувати, якщо це передбачено бізнес-правилами. Кнопка виставлення чека з'являється лише коли дозволяє функція `canIssueInvoice` з `orderPayment.ts` (враховується роль, прив'язка послуги до дизайнера, відсутність оплати, не скасоване замовлення). Натискання відкриває модальне вікно `IssueInvoiceModal`, де вводиться сума чека; після підтвердження викликається API `issueInvoice`, і клієнт у своєму кабінеті бачить суму до оплати.

Сторінка `AdminDashboard` (`AdminDashboard.tsx`, маршрут `/admin`) - розширений інтерфейс адміністратора з повним контролем платформи. Структура також починається з `Hero` («Адмін панель») і блоку з чотирьох карток: категорії, послуги, замовлення, завершені замовлення. На відміну від панелі дизайнера, адмін бачить усі записи в системі, а не лише власні.

Нижче розташовані послідовні секції (вертикальний стек `space-y-8`), кожна в обгортці `card`:

- Категорії - CRUD довідника `service_categories` (створення, редагування, видалення з підтвердженням).
- Послуги - перегляд і керування всіма послугами всіх дизайнерів, ті самі форми, що й у дизайнера, але без обмеження `mine`.

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		31

- Замовлення - глобальний список замовлень, зміна статусу, виставлення чека, видалення помилкових записів.
- Відгуки - перегляд і модерація відгуків клієнтів.
- Статистика системи - зведені показники (кількість дизайнерів, огляд активності).

Адміністратор дублює частину функцій дизайнера на рівні всієї платформи (зміна статусу, чек), але додатково керує структурою каталогу та може втручатися в будь-яке замовлення.

Верстка обох панелей орієнтована на роботу з табличними та картковими даними, формами вбудованого типу (розгортання форми «Додати» в тій самій секції без переходу на іншу сторінку) та діями з іконками Lucide (Plus, Edit, Trash2, FileText тощо). Критичні операції захищені: видалення замовлення в адмін-панелі - через ConfirmDialog (підтвердження перед викликом API delete); видалення послуг і категорій - аналогічно з урахуванням ризику втрати даних. Після мутацій списки оновлюються без перезавантаження сторінки.

Для зручності користувача на панелях застосовано єдиний стиль із рештою сайту: кольори primary/accent, компоненти card, btn, input з index.css, градієнтні акценти на картках статистики, toast-повідомлення про успіх і помилки, плавні hover-ефекти. Тексти інтерфейсу - українською мовою. Таким чином панелі дизайнера та адміністратора є не окремим «дизайном», а продовженням загальної системи верстки Ріхора, але з функціоналом, орієнтованим на операційну роботу персоналу.

2.2.6. Організація frontend-коду

Клієнтська частина зібрана в каталозі frontend/src/ за принципом розділення відповідальності: кожна підпапка відповідає за свій шар застосунку, що спрощує супровід, пошук помилок і розширення функціоналу під час розробки дипломного проєкту.

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		32

Каталог `pages/` містить сторінки - компоненти верхнього рівня, які безпосередньо прив'язані до маршрутів у `App.tsx`. Серед них: `HomePage`, `ServicesPage`, `ServiceDetailPage`, `OrdersPage`, `ChatPage`, `ProfilePage`, `DesignerDashboard`, `AdminDashboard` та інші. Один файл - одна логічна web-сторінка для користувача, що відповідає записам у таблиці 2.1.

Каталог `components/` - повторно використовувані елементи інтерфейсу: `Layout` (шапка, меню, `Outlet` для вкладених маршрутів), `ProtectedRoute` (перевірка JWT і ролі), `IssueInvoiceModal`, `ConfirmDialog`, `ServiceImageField`, `UISettingsModal`, `ThemeToggle` тощо. Винесення їх у компоненти уникнуло дублювання розмітки між клієнтським кабінетом і панелями дизайнера/адміна.

Каталог `api/` - тонкий шар звернень до REST API [6] на базі `Axios` [12]: окремі модулі `auth.ts`, `services.ts`, `orders.ts`, `categories.ts`, `reviews.ts`, `chat.ts` тощо. Сторінки не формують URL вручну, а викликають типізовані функції; токен авторизації підставляється централізовано (`interceptor`), що узгоджується з JWT на backend.

Каталог `store/` - глобальний стан автентифікації (`authStore` на `Zustand` [15]): збереження токена, даних користувача, ролі. Від цього залежать умови в `App.tsx` (перенаправлення з `login/register`) і `ProtectedRoute`.

Каталог `lib/` - допоміжна бізнес-логіка інтерфейсу, не прив'язана до однієї сторінки: `orderPayment.ts` і `orderNormalize.ts` (умови оплати, чека, відображення ціни), `serviceForm.ts` (валідація та збір `payload` послуги), `serviceImage.ts` і `resolveMediaUrl` для URL зображень, `stompClient.ts` для WebSocket-чату [5], `toast.ts` для сповіщень.

Додатково використовуються: `hooks/` (наприклад, налаштування UI), `types/` - TypeScript-інтерфейси для DTO з API; `index.css` - глобальні класи `btn`, `card`, `input` і CSS-змінні теми.

Зв'язок шарів у типовому сценарії: сторінка з `pages/` → викликає `api/` через `React Query` [11] → отримує дані → відображає через `components/` → застосовує правила з `lib/` → стан користувача з `store/`. Така організація відповідає сучасним

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		33

практикам розробки SPA, полегшує тестування окремих модулів (наприклад, логіки оплати в orderPayment.ts без відкриття всієї OrdersPage) і узгоджується з модульною структурою backend (controller - service - repository).

2.3. Розробка структури бази даних сайту

Проектування бази даних - це відображення предметної області(клієнти, послуги, замовлення, чат, відгуки) у реляційну схему PostgreSQL. Процес виконувався послідовно за етапами методички: (1) визначення таблиць; (2) полів у кожній таблиці; (3) типів і обмежень полів; (4) зв'язків між таблицями (зовнішні ключі); (5) наповнення початковими даними, запити через JPA та REST API замість класичних «форм Access». Результат етапу: ER-діаграма наведена на рисунку 1.5; детальний опис - у таблиці 2.2.

2.3.1. Вибір СУБД і засобів доступу

Як сховище даних обрано PostgreSQL 16 [3] - надійну реляційну СУБД з підтримкою зовнішніх ключів, транзакцій і JSON-сумісних типів за потреби. Схема БД версіонується міграціями Flyway [10] (каталог `src/main/resources/db/migration`), що дозволяє відтворювати однакову структуру на машині розробника, сервері демонстрації та production.

На сервері доступ до даних має реалізовуватися через Spring Data JPA (сутності, репозиторії) та сервісний шар, де зосереджена бізнес-логіка (вимога ТЗ до структури програм). Такий підхід забезпечує чітке розділення відповідальності між компонентами системи, спрощує супровід коду та його подальше розширення. Режим Hibernate ddl-auto=validate гарантує, що структура БД відповідає міграціям, без самовільної зміни схеми при старті, що підвищує надійність роботи застосунку та зменшує ризик виникнення помилок під час розгортання.

Логічна ER-модель наведена на рисунок 1.5. Основні сутності та їх

					2026.KBP.122.423.10.00.00 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		34

призначення подано в таблиці 2.2.

Таблиця 2.2 – Основні таблиці бази даних Ріхога

Таблиця	Призначення	Зв'язки (скорочено)
users	Облікові записи користувачів	user_roles, замовлення, відгуки, чат
roles, user_roles	Ролі CLIENT, DESIGNER, ADMIN	Багато-до-багатьох з users
service_categories	Категорії послуг	3 services
services	Каталог послуг (ціна, опис, зображення)	Категорія, дизайнер, замовлення
orders	Замовлення (статус, вимоги, чек, оплата)	users, services
reviews	Відгуки та оцінки	users, services
notifications	Сповіщення користувачам	users
chat_rooms, chat_messages	Чат	Користувач, адмін/дизайнер
email_verification_codes	Коди підтвердження email	Безпека облікового запису
portfolio_items	Портфоліо робіт	Послуги
profile_changes	Журнал змін портфоліо	users

Для замовлень реалізовано життєвий цикл статусів (pending, in_progress, completed, cancelled) та фінансові поля: payment_status, invoiced_at, invoice_amount (виставлення чека та оплати). Така модель дозволяє відстежувати поточний стан виконання кожного замовлення на всіх етапах його обробки та забезпечує контроль

фінансових операцій. Крім того, зберігання інформації про оплату й дату формування рахунку спрощує аналіз замовлень і формування звітності в системі.

2.3.2. Ключові сутності предметної області

Користувачі та ролі. Таблиця `users` зберігає облікові дані. Ролі винесені в `roles` і `user\_roles` (значення CLIENT, DESIGNER, ADMIN), що гнучкіше за поле `is\_admin` і відповідає трьом типам акторів на рисунок 1.3.

Каталог. `service\_categories` і `services` описують пропозиції компанії. Поле `designer\_id` (міграція V4) прив'язує послугу до виконавця.

Замовлення. Таблиця `orders` - ядро бізнес-процесу. Поля:

- `status` - етап виконання робіт;
- `payment\_status` - фінансовий стан (`unpaid` → `invoiced` → `paid`);
- `invoice\_amount`, `invoiced\_at`, `paid\_at` - підтримка сценарію чека та

оплати (узгоджено з рисунок 1.4).

Комунікація. `chat\_rooms` і `chat\_messages` зберігають історію діалогів. Повідомлення можуть містити посилання на зображення у файловому сховищі (`uploads/chat/`).

Відгуки та сповіщення. `reviews` - оцінка 1–5 і коментар; `notifications` - інформування користувача про події в системі.

Безпека. `email\_verification\_codes` - коди для підтвердження email при зміні пароля; `profile\_changes` - журнал змін профілю для аудиту.

2.3.3. Міграції та цілісність даних

У проєкті Ріхога структура бази даних не створюється вручну один раз, а еволюціонує контрольовано за допомогою інструменту Flyway. Усі зміни схеми зберігаються у вигляді окремих SQL-файлів у каталозі src/main/resources/db/migration. При запуску Spring Boot міграції застосовуються

					2026.KBP.122.423.10.00.00 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		36

автоматично в порядку версій (V1, V2, ... V10); у таблиці flyway_schema_history фіксується, які файли вже виконані. Це гарантує однакову структуру БД на комп'ютері розробника, демонстраційному сервері та в production і відповідає сучасній практиці DevOps для вебзастосунків.

Початкова міграція V1__init_schema.sql формує ядро предметної області: таблиці users, service_categories, services, orders, reviews, notifications, chat_rooms, chat_messages, profile_changes, а також довідник ролей roles і зв'язок many-to-many user_roles. У V1 одразу визначені первинні та зовнішні ключі, унікальність username і email, обмеження рейтингу відгуку (1–5), початкові записи ролей ADMIN, DESIGNER, CLIENT. Це базовий «каркас», на який накладаються подальші зміни без переписування всієї схеми.

Наступні міграції розширюють функціонал сайту відповідно до уточнень ТЗ і результатів тестування:

- V2 - таблиця portfolio_items (портфоліо робіт, прив'язка до послуги);
- V3 - поле users.avatar змінено на тип TEXT (тимчасова підтримка legacy base64; пізніше основний шлях - файли в uploads);
- V4 - у services додано designer_id (зв'язок послуги з виконавцем, відповідає ролі DESIGNER на рисунок 1.3);
- V5 - таблиця email_verification_codes для підтвердження email при зміні/відновленні пароля (індекси за user_id, purpose, expires_at);
- V6 - у orders поля payment_status (за замовчуванням unpaid) та paid_at, індекс idx_orders_payment_status для швидкого фільтрування замовлень за оплатою;
- V7 - invoiced_at (момент виставлення чека);
- V8 - виправлення порожніх created_at у послугах;
- V9 - invoice_amount (сума чека для відображення клієнту);
- V10 - службовий UPDATE (backfill): для старих записів зі статусом invoiced без суми підставляється ціна з таблиці services.

Такий ланцюг міграцій ілюструє реальний цикл розробки дипломного

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		37

проекту: спочатку базова модель, потім оплата й чек (узгодження з рисунок 1.4), потім безпека облікового запису та технічні виправлення даних.

Цілісність даних забезпечується на рівні СУБД і на рівні застосунку. У PostgreSQL зовнішні ключі (REFERENCES) не дозволяють, наприклад, створити замовлення на неіснуючу послугу або повідомлення чату без існуючої кімнати. Для залежних записів від користувача застосовано ON DELETE CASCADE (відгуки, сповіщення, коди верифікації), щоб при видаленні облікового запису не залишались «осиротілі» рядки. Для посилань на категорії та дизайнера частіше використано ON DELETE SET NULL, щоб видалення довідникового запису не блокувало історію замовлень.

На рівні Spring Boot увімкнено spring.jpa.hibernate.ddl-auto=validate: Hibernate лише перевіряє відповідність JPA-сутностей (Order, User, DesignService тощо) уже застосованій схемі Flyway і не змінює таблиці при старті. Бізнес-правила (хто може виставити чек, коли дозволена оплата) дублюються в OrderService, що запобігає некоректним переходам payment_status навіть при прямому зверненні до API.

Транзакційність операцій (створення замовлення, виставлення чека, підтвердження оплати) виконується в межах @Transactional у сервісному шарі: або зберігається повний набір змін, або відкочується помилка - це важливо для узгодженості фінансових полів invoice_amount, invoiced_at, paid_at. Підсумок підрозділу: міграції Flyway забезпечують відтворювану еволюцію схеми, а зовнішні ключі, індекси, validate і сервісна логіка - цілісність і коректність даних під час експлуатації сайту Píxora.

2.3.4. Файлове сховище

Не всі дані вебсайту доцільно зберігати в реляційній базі. Двійковий вміст (зображення аватарів, прев'ю послуг, вкладення в чаті) у проєкті Píxora винесено у файлове сховище на диску сервера, а в таблицях PostgreSQL зберігаються лише публічні URL - рядки на кшталт /uploads/avatars/uuid.jpg, /uploads/services/...,

					2026.KBP.122.423.10.00.00 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		38

/uploads/chat/.... Такий підхід зменшує обсяг БД, прискорює вибірки списків замовлень і послуг і спрощує кешування статички браузером.

Кореневий каталог задається в конфігурації (StorageProperties, за замовчуванням uploads відносно робочої директорії backend). При старті FileStorageService автоматично створює підкаталоги avatars, chat, services. Завантаження виконується через REST (FileController) з прийманням MultipartFile; імена файлів генеруються як UUID з розширенням, що уникнуло колізій і передбачуваних URL.

Перед збереженням виконується валідація: файл не порожній, розмір не більше 5 МБ, тип вмісту - лише зображення (JPEG, PNG, GIF, WebP). Після копіювання на диск у відповідь клієнту повертається відносний шлях /uploads/..., який записується в поле сутності (avatar користувача, image послуги, посилання в повідомленні чату). Frontend через resolveMediaUrl і проксі Vite у режимі розробки звертається до того ж шляху на backend .

Віддача файлів налаштована в WebMvcConfig: обробник ресурсів /uploads/ мапиться на фізичний каталог uploadRoot. У SecurityConfig шлях /uploads/ дозволений для читання (перегляд зображень без повторної автентифікації на кожен файл), тоді як зміна даних - лише через захищені API з JWT.

Видалення реалізовано за принципом best-effort: метод deleteIfStored у FileStorageService видаляє файл з диску лише якщо URL починається з /uploads/ і шлях не виходить за межі кореня (захист від path traversal). Це доповнює видалення записів у БД при оновленні аватара або послуги.

Історично міграція V3 розширила поле avatar до TEXT для підтримки base64 у старих записах; у поточній архітектурі основний спосіб - файли в uploads/, а base64 залишається лише для зворотної сумісності на frontend. Для нових завантажень рекомендований саме файловий шлях.

Під час розгортання (розділ 3) каталог uploads має зберігатися на постійному томі Docker або резервуватися разом із БД: втрата каталогу призведе до «битих» посилань у таблицях, навіть якщо PostgreSQL відновлена з бекапу. У дипломній

					2026.KBP.122.423.10.00.00 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		39

демонстрації достатньо локального каталогу на машині з backend.

Підсумок: файлове сховище в Pihora - окремий шар сховища поруч із PostgreSQL; БД зберігає метадані та зв'язки, диск - медіаконтент. Це узгоджується з клієнт-серверною архітектурою (рисунок 1.2) і вимогами до швидкодії інтерфейсу при роботі з зображеннями в каталозі, профілі та чаті.

2.4. Програмування сайту

Завдання програмування полягає в тому, щоб об'єднати всі обрані технології (React [1], Spring Boot [2], PostgreSQL [3], JWT [4], WebSocket [5]) в єдиний працездатний вебресурс Pihora і надати користувачеві зручний інтерфейс для реалізації функціональних можливостей, визначених у технічному завданні (розділ 1.2.3). На відміну від розділів 2.1–2.3, де описано структуру сторінок і бази даних, у п. 2.4 розкрито програмну реалізацію: серверний API, клієнтський застосунок, панелі дизайнера та адміністратора, механізми безпеки та взаємодії компонентів (рисунок 1.2).

Обрано підхід на основі фреймворків: backend - Spring Boot 3.3 [2] (Java 17), frontend - React 18 [1] з TypeScript [8], збірка клієнта - Vite 5 [9]. Таке рішення обґрунтоване необхідністю кастомного бізнес-процесу «замовлення - чек - оплата» (рисунок 1.4), рольовою моделлю з трьома акторами (рисунок 1.3) та інтеграцією чату в реальному часі.

Архітектурно система має вигляд клієнт-серверної моделі з REST API [6]. Браузер завантажує SPA (Single Page Application); подальші дії користувача виконуються через HTTP-запити до `/api/...` та WebSocket [5] `/ws`. Серверна логіка зосереджена в пакеті `com.designdev.designservicesapi`, клієнтська - у каталозі `frontend/src`. Дані зберігаються в PostgreSQL [3]; медіафайли - у файловому сховищі `uploads/` (див. п. 2.3.4).

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
						40
Зм.	Арк.	№ докум.	Підпис	Дата		

Середовище розробки та запуску:

- JDK 17, Apache Maven - збірка та запуск backend (`mvn spring-boot:run`);
- Node.js LTS, npm - залежності та dev-сервер frontend (`npm run dev`);
- Docker Compose - контейнер PostgreSQL 16;
- файл `.env` - параметри JWT, підключення до БД, за потреби SMTP;
- редактор коду Visual Studio Code / intellij IDEA.

У режимі розробки Vite проксує запити `/api`, `/uploads` і `/ws` на backend, що усуває проблеми CORS під час локального тестування. Документація REST API генерується Swagger/OpenAPI і доступна за адресою `/swagger-ui.html`.

Структура програмних модулів

Серверна частина організована за шаровою схемою (layered architecture):

- controller - REST-ендпоінти, прийом JSON, валідація `@Valid`, делегування в сервіси;
- service - бізнес-правила (замовлення, оплата, чек, права дизайнера);
- repository - доступ до БД через Spring Data JPA;
- model - JPA-сутності (User, Order, DesignService тощо);
- dto - об'єкти обміну з frontend (зокрема OrderDtos з підтримкою JSON-аліасів invoice_amount);
- config - Security, CORS, WebSocket, Flyway, статичні ресурси `/uploads/`;
- security - JWT-фільтр, UserDetailsService.

Клієнтська частина поділена на pages (сторінки за маршрутами), components (Layout, ProtectedRoute, модальні вікна), api (модулі Axios), store (Zustand authStore), lib (orderPayment, stompClient, serviceForm).

Відповідність основних модулів backend і frontend, REST-шляхів і сторінок наведено в таблиці 2.3. Таблиця є зведеним «паспортом» програмної реалізації для зв'язку з таблицею 2.1 (маршрути UI) і таблицею 2.2 (таблиці БД).

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		41

Зіставлення програмних модулів backend і frontend наведено в таблиці 2.3.

Таблиця 2.3 – Основні програмні модулі вебсайту Pixora

Модуль	Реалізація (backend)	Реалізація (frontend)
Авторизація	AuthController, JWT [4]	LoginPage, RegisterPage, authStore (Zustand [15])
Каталог	ServiceController, CategoryController	ServicesPage, ServiceDetailPage
Замовлення	OrderController	OrdersPage, PaymentsPage
Чат	ChatController, WebSocket [5]	ChatPage, stompClient
Відгуки	ReviewController	ReviewsPage
Сповіщення	NotificationController	NotificationsPage, NotificationsDropdown
Профіль	UserController, ProfileChangeController	ProfilePage, SecuritySettingsPage
Файли	FileController	Завантаження аватарів, зображень
Панель дизайнера	Ті самі API з перевіркою ролі	DesignerDashboard
Адміністрування	API + роль ADMIN	AdminDashboard

2.4.1. Написання клієнтської частини

Клієнтська частина призначена для гостей (перегляд каталогу без входу) та авторизованих користувачів із роллю CLIENT (а також для спільних сторінок профілю, які доступні й іншим ролям). Програмування виконувалось у середовищі Node.js з використанням TypeScript для статичної перевірки типів DTO.

Технологічний стек і точка входу:

Головний файл застосунку - `main.tsx`, який монтує дерево React у DOM. Маршрутизація визначена в `App.tsx` (React Router v6 [7]): публічні шляхи `/login`,

`/register`, `/forgot-password`; вкладені маршрути в Layout для `/`, `/services`, `/orders`, `/chat` тощо. Захищені розділи обгорнуті компонентом ProtectedRoute: за відсутності JWT виконується `

Стан автентифікації зберігається в `authStore` (Zustand [15]): токен у `localStorage`, користувач відновлюється декодуванням payload JWT (поля sub, role, userId). Це дозволяє після перезавантаження сторінки не вимагати повторного входу, поки токен дійсний.

HTTP-клієнт і авторизація

Центральний HTTP-клієнт - `api/client.ts` на базі Axios [12]. Interceptor запитів додає заголовок `Authorization: Bearer <token>`. Interceptor відповідей при статусі 401 очищає токен і перенаправляє на `/login`, що узгоджує поведінку UI з серверною відмовою в доступі.

Сторінки LoginPage та RegisterPage викликають `POST /api/auth/login` та `POST /api/auth/register`; у відповідь отримують `{ token }`, який передається в `setToken`. ForgotPasswordPage реалізує двокроковий сценарій: запит коду на email (`/api/auth/forgot-password/request`) і встановлення нового пароля (`/api/auth/forgot-password/reset`) - відповідає вимогам безпеки облікового запису.

Функції рядків таблиці 2.3 реалізовані у відповідних компонентах pages/ та модулях api/.

Робота з даними (TanStack Query)

Для зменшення «ручного» керування завантаженням застосовано TanStack React Query [11]. Приклад: HomePage та ServicesPage використовують `useQuery` для каталогу послуг; OrdersPage - `useQuery` з ключем `orders` і `useMutation` для оплати. Після успішної мутації викликається `queryClient.invalidateQueries`, і списки оновлюються без повного перезавантаження SPA - важливо для UX при зміні статусу замовлення або оплаті.

Доменні API винесені в окремі файли: `api/services.ts`, `api/orders.ts`, `api/reviews.ts`, `api/chat.ts`, `api/categories.ts`, `api/user.ts`. Кожен модуль інкапсулює шляхи та типи відповідей, що спрощує супровід при зміні контракту

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		43

Swagger.

Оформлення замовлення та оплата

Клієнт оформлює замовлення на `ServiceDetailPage`: форма з полями `requirements` і `deadline` відправляє `POST /api/orders`. На сервері `OrderService.create` встановлює `status = pending`, `payment_status = unpaid` (за замовчуванням з БД).

На `OrdersPage` і `PaymentsPage` відображаються статуси виконання та оплати. Логіка видимості кнопок винесена в `lib/orderPayment.ts`:

- `canClientPay` - `true` лише при `payment_status = invoiced` і не скасованому замовленні;
- `getOrderPayablePrice` - сума з `invoice_amount` або, для `legacy`-записів, ціна послуги;
- `getPaymentStatusLabel` - українські підписи для UI.

Підтвердження оплати викликає `POST /api/orders/{id}/pay`. На `backend processPayment` перевіряє, що платник - власник замовлення, і що чек уже виставлено; інакше повертається помилка з текстом «Спочатку дизайнер має виставити чек на оплату». Це програмна реалізація `guard-умов` з рисунок 1.4.

Нормалізація відповіді сервера (різні варіанти імен полів `invoiceAmount` / `invoice_amount`) виконується в `lib/orderNormalize.ts`, що усунуло розбіжності відображення суми чека на картці замовлення.

Чат (WebSocket [5] / STOMP [13])

`ChatPage` підключається до брокера через власну реалізацію `lib/stompClient.ts` (клас `StompClient`): встановлення `WebSocket`, кадри `CONNECT/SUBSCRIBE/SEND` за протоколом `STOMP 1.2`, підписка на `destination` кімнати. Повідомлення з текстом і зображенням надсилаються на сервер; `ChatWebSocketController` та `ChatController` зберігають історію в таблицях `chat_rooms`, `chat_messages`. У заголовку чату відображається стан з'єднання; при розриві з'єднання залишається можливість надсилання через `REST` (резервний канал).

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
						44
Зм.	Арк.	№ докум.	Підпис	Дата		

2.4.2. Написання admin-частини

Під admin-частиною в межах кваліфікаційної роботи розуміється сукупність функціоналу для ролей DESIGNER та ADMIN: панель дизайнера (`/designer`, DesignerDashboard) та адмін-панель (`/admin`, AdminDashboard). Обидві реалізовані тими самими технологіями, що й клієнтська частина, але з додатковими перевітками прав.

У App.tsx для `/designer` задано `user.role === 'ADMIN' проходить для обох маршрутів), але клієнт не потрапить на `/admin`.

На backend аналогічні обмеження: `@PreAuthorize [14] ("hasAnyRole('ADMIN','DESIGNER'))"` на зміну статусу замовлення та виставлення чека; `@PreAuthorize("hasRole('ADMIN'))` для шляхів `/api/admin/` (якщо виділені окремі admin-endpoints); SecurityConfig відкриває публічно лише GET каталогу та auth, решта `/api/orders/` - для автентифікованих.

DesignerDashboard завантажує:

- список власних послуг (`servicesApi.search` з параметром `mine: true`);
- замовлення, де `designerId` збігається з поточним користувачем, або де він є замовником (логіка фільтра в `OrderController.search` для ролі DESIGNER).

Дизайнер створює/редагує/видаляє послуги через мутації React Query; форма використовує `lib/serviceForm.ts` (`buildServicePayload`, `isServiceFormValid`). Зображення послуги завантажується через `FileController` → `FileStorageService` → каталог `uploads/services`.

Для замовлення доступні зміна статусу (`PATCH /api/orders/{id}/status`) та виставлення чека через `IssueInvoiceModal` → `POST /api/orders/{orderId}/invoice`. На frontend кнопка «Виставити чек» контролюється функцією `canIssueInvoice`; на backend - метод `issueInvoice` у `OrderService`: перевірка `assertCanManageOrderAsDesigner`, нормалізація суми (зокрема підтримка вводу

					2026.KBP.122.423.10.00.00 ПЗ	Арк.
						45
Зм.	Арк.	№ докум.	Підпис	Дата		

«5к» → «5000 грн»), встановлення `payment_status = invoiced`, `saveAndFlush` і `refresh` сутності для актуального DTO.

AdminDashboard розширює можливості дизайнера:

- CRUD категорій (CategoryController);
- перегляд і керування всіма послугами та замовленнями в системі;
- перегляд відгуків (ReviewController);
- видалення замовлення з ConfirmDialog → DELETE `/api/orders/{id}`.

Адміністратор не обмежений `designer_id` при пошуку замовлень (у OrderController для `isAdmin` фільтри `userId/designerId` не застосовуються). Це реалізує прецедент «керування платформою» з рисунок 1.3.

Приклад логіки оплати в OrderService.processPayment: перевірка власника замовлення, стану `invoiced`, встановлення `paid` та `paid_at` - імітація оплати без зовнішнього платіжного шлюзу (навчальний проєкт).

Після успішного login AuthService формує JWT, у payload якого передаються ідентифікатор користувача, ім'я (sub) та ролі (authorities виду ROLE_CLIENT, ROLE_DESIGNER, ROLE_ADMIN). JwtAuthenticationFilter зчитує заголовок Authorization на кожному захищеному запиті, валідує підпис і термін дії токена, заповнює SecurityContext. Далі спрацьовують правила SecurityConfig: публічний доступ до GET `/api/services`, `/api/categories`, до `/api/auth/`, Swagger, actuator health, статика `/uploads/` і WebSocket `/ws/`; решта `/api/orders/` і `/api/chat/` - лише для автентифікованих.

Методна безпека `@EnableMethodSecurity` дозволяє тонко обмежувати окремі методи контролерів (наприклад, `issueInvoice` - лише ADMIN і DESIGNER). Це другий рівень захисту після маршрутизації URL і доповнює перевірки в OrderService.assertCanManageOrderAsDesigner (прив'язка замовлення до `designer_id` послуги).

Обмін даними виконується JSON-об'єктами DTO, а не «голими» JPA-сутностями - це запобігає витoku паролів і циклічним JSON при двобічних зв'язках. OrderDtos.OrderResponse містить поля для UI: `status`, `paymentStatus`, `invoiceAmount`

					2026.KBP.122.423.10.00.00 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		46

(з JsonAlias invoice_amount для сумісності), вкладений об'єкт service з ціною та designerId. Такий контракт узгоджений із TypeScript-типами у `frontend/src/types` і нормалізаторами в lib.

Валідація вхідних даних - анотації Jakarta Validation на DTO (@NotBlank, @Min/@Max для рейтингу). Помилки валідації та IllegalArgumentException з сервісів перетворюються RestExceptionHandler на відповіді з HTTP 400/404 і текстом українською, які frontend показує через notify.error (toast).

Паролі зберігаються з хешуванням BCrypt [14] (PasswordEncoder у SecurityConfig). JWT [4] генерується при login/register і перевіряється JwtAuthenticationFilter на кожному запиті. Сесія без стану (STATELESS). Глобальні винятки обробляє RestExceptionHandler - клієнт отримує зрозумілі HTTP-статуси та повідомлення українською.

Профілі Spring: dev - тестові користувачі (DataInitializer), розширене логування; prod - експлуатація без зайвих деталей. Actuator health використовується при тестуванні (п. 2.5).

2.4.3. Збірка та інтеграція компонентів

Після завершення програмування окремих модулів виконано інтеграційну збірку:

- backend: `mvn clean package` → JAR, запуск з підключенням до PostgreSQL і застосуванням міграцій Flyway [10];
- frontend: `npm run build` → статичні файли в `frontend/dist`, які можна розмістити на Nginx або віддавати через Spring (за конфігурацією розділу 3);
- перевірка end-to-end: реєстрація → замовлення → чек → оплата → чат (узгодження з таблицею 2.4).

Програмний код, структура сценаріїв і зв'язки між модулями наведені в таблиці 2.3; Детальні лістинги основних модулів наведено в додатках А–Д та (с. 99–115): OrderController - додаток А (с. 99); SecurityConfig - додаток Б (с. 104);

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		47

App.tsx - додаток В (с. 108); App.tsx - додаток Д (с. 138); stompClient.ts - додаток Г (с. 111); useChatWebSocket.ts - додаток Д (с. 115).

Програмована реалізація сайту Ріхора об'єднує React SPA [1] та Spring Boot API в єдину систему з рольовою безпекою, REST [6] і WebSocket [5], повністю підтримуючи функціонал ТЗ. Модулі backend і frontend, сторінки та API зведені в таблиці 2.3; бізнес-процес замовлення реалізовано в OrderService і на сторінках OrdersPage, DesignerDashboard, AdminDashboard. Розділ 3 описує експлуатацію та розміщення зібраного продукту.

2.5. Тестування web-сайту

Після завершення програмування (п. 2.4) виконано тестування вебсайту компанії дизайнерських послуг Ріхора з метою перевірки відповідності реалізованого програмного забезпечення вимогам технічного завдання (розділ 1.2.3) та коректності ключового бізнес-процесу «замовлення - чек - оплата» (рисунки 1.4). У підрозділі описано процес і результати тестування, методи та види перевірок; критерії та сценарії перевірки зведено в таблиці 2.4.

2.5.1. Мета, об'єкт і завдання тестування

Мета тестування - підтвердити працездатність вебресурсу в цілому (frontend + backend + база даних), коректність розподілу функцій між ролями клієнт, дизайнер, адміністратор (рисунки 1.3) та відсутність критичних дефектів, що блокують основні сценарії користування.

Об'єкт тестування: вебсайт Ріхора - SPA на React [1], REST API на Spring Boot [2], СУБД PostgreSQL 16 [3] (Docker), файлове сховище uploads/.

Завдання тестування:

- перевірити публічні та захищені розділи сайту (таблиці 2.1);

					2026.KBP.122.423.10.00.00 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		48

– перевірити повний цикл замовлення:

оформлення → обробка → виставлення чека → оплата;

- перевірити чат, відгуки, профіль, панелі дизайнера та адміна;
- перевірити авторизацію (JWT), обмеження доступу за ролями;
- перевірити доступність API та стан сервера (health, Swagger);
- зафіксувати результати у таблиці 2.4.

2.5.2. Види та методи тестування

У межах кваліфікаційної роботи застосовано такі види тестування:

Функціональне тестування - основний вид. Перевіряється відповідність функцій сайту вимогам ТЗ: реєстрація, каталог, замовлення, чек, оплата, чат, відгуки, адміністрування. Виконувалось вручну за сценаріями користувача для кожної ролі (гість, CLIENT, DESIGNER, ADMIN).

Інтеграційне тестування - перевірка взаємодії компонентів: браузер ↔ REST API ↔ PostgreSQL; frontend ↔ WebSocket; завантаження файлів у uploads/ і відображення URL на сторінках. Частково підтримується автоматизованими тестами backend (запуск `mvn test`, за наявності - Testcontainers для PostgreSQL).

Тестування інтерфейсу користувача (UI) - перевірка відображення сторінок, української мови підписів, повідомлень про помилки (toast), адаптивності основних розділів (desktop).

Тестування безпеки (обмежене) - перевірка недоступності `/orders` і `/admin` без токена або з чужою роллю; відмова при невірному паролі; передача JWT у заголовок Authorization.

Регресійне тестування (вибіркове) - повторне виконання сценарію замовлення - чек - оплата після виправлення дефектів (відображення суми чека, health actuator).

Методи виконання: ручне тестування за чек-листом і сценаріями (таблиця 2.4); перевірка API через Swagger UI; моніторинг стану - `GET /actuator/health`;

					2026.KBP.122.423.10.00.00 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		49

перегляд даних у PostgreSQL (pgAdmin) за потреби; аналіз мережевих запитів у інструментах розробника браузера (DevTools).

2.5.3. Середовище та критерії початку і завершення тестування

Середовище тестування:

- ОС: Microsoft Windows 10/11;
- браузер: Google Chrome (актуальна версія);
- backend: Spring Boot;
- frontend: Vite dev-server або production-збірка;
- БД: PostgreSQL у Docker (``docker compose up -d``);
- облікові записи: тестові користувачі ролей CLIENT, DESIGNER, ADMIN (профіль dev або створені при реєстрації).

Критерії початку тестування: завершено п. 2.4; backend і frontend запускаються без помилок; міграції Flyway [10] застосовані; ``/actuator/health`` повертає статус UP.

Критерії завершення: виконано всі сценарії таблиці 2.4 з позитивним результатом (+); критичні дефекти усунені або задокументовані.

2.5.4. Процес тестування

Процес тестування проводився поетапно відповідно до Плану тестування (п. 2.5.2):

- Етап 1. Підготовка - складання сценаріїв за ролями, перевірка доступності серверів, входу тестових користувачів;
- Етап 2. Smoke-тест - відкриття головної сторінки, каталогу, входу в систему; перевірка health і Swagger;

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
						50
Зм.	Арк.	№ докум.	Підпис	Дата		

- Етап 3. Функціональне тестування за ролями - послідовне виконання сценаріїв гостя, клієнта, дизайнера, адміністратора (таблиця 2.4);
- Етап 4. Тестування бізнес-процесу - сквозний сценарій: клієнт створює замовлення → дизайнер змінює статус і виставляє чек → клієнт бачить суму і підтверджує оплату → перевірка payment_status = paid у інтерфейсі та БД;
- Етап 5. Негативні та граничні випадки - п. 2.5.7;
- Етап 6. Оформлення результатів - заповнення таблиці 2.4.

2.5.5. Результати функціонального тестування

Для відображення критеріїв перевірки вебсайту в пояснювальній записці використано таблицю сценаріїв функціонального тестування. Кожний рядок відповідає перевірці функції за роллю користувача; статус «+» означає успішне проходження сценарію, результат наведено в таблиці 2.4.

Таблиця 2.4 – Сценарії функціонального тестування вебсайту Ріхора

№	Роль	Сценарій	Очікуваний результат	Статус
1	2	3	4	5
1	Гість	Відкриття головної, перехід у каталог	Сторінки відображаються, пошук веде до /services	+
2	Гість	Реєстрація нового клієнта	Обліковий запис створено, вхід виконано	+
3	CLIENT	Оформлення замовлення на послугу	Замовлення зі статусом pending	+
4	DESIGNER	Прийняття замовлення в роботу	Статус in_progress	+
5	DESIGNER	Виставлення чека з сумою	invoiced, збережено invoice_amount	+

Продовження таблиці 2.4

1	2	3	4	5
6	CLIENT	Підтвердження оплати	paid, оновлено payment_status	+
7	CLIENT	Відправлення повідомлення в чаті	Повідомлення з'являється в кімнаті	+
8	CLIENT	Додавання відгуку на послугу	Відгук збережено в БД	+
9	ADMIN	Створення категорії, редагування послуги	Зміни відображаються в каталозі	+
10	ADMIN	Перегляд усіх замовлень	Список замовлень користувачів	+

Усі 10 сценаріїв виконано з позитивним результатом. Сценарії № 3–6 підтверджують бізнес-процес «замовлення - чек - оплата».

2.5.6. Результати тестування за ролями

Гість. Перевірено відкриття головної сторінки, перехід у каталог, пошук послуг, перегляд картки послуги без реєстрації. Спроба відкрити /orders без автентифікації - перенаправлення на сторінку входу. Результат: відповідає вимогам публічної зони сайту.

Клієнт (CLIENT). Виконано реєстрацію та вхід; оформлено замовлення на послугу (поля вимог і дедлайн); у розділі «Мої замовлення» відображено статус pending. Після дій дизайнера з'явився чек із числовою сумою (invoice_amount). Підтверджено оплату - payment_status = paid. Перевірено відправку повідомлення в чат і створення відгуку. Результат: позитивний (таблиця 2.4).

Дизайнер (DESIGNER). Створено послугу з зображенням; отримано замовлення по цій послугі; статус змінено на in_progress; через IssueInvoiceModal виставлено чек; перевірено, що клієнт бачить суму; замовлення завершено.

					2026.KBP.122.423.10.00.00 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		52

Результат: позитивний.

Адміністратор (ADMIN). Створено/редаговано категорію; переглянуто всі послуги та замовлення; змінено статус замовлення; перевірено доступ до /admin для CLIENT - доступ заборонено (ProtectedRoute і backend). Результат: позитивний.

Сквозний сценарій (див. рис. 1.4) пройдено без критичних збоїв; узгодженість UI і API підтверджена.

2.5.7. Негативні та граничні випадки

Додатково перевірено:

- вхід з невірним паролем - відмова, JWT не видається;
- оплата до виставлення чека - кнопка «Оплатити» недоступна (canClientPay = false);
- виставлення чека без суми / повторне введення суми для invoiced без invoice_amount - обробка на backend (OrderService.issueInvoice);
- завантаження зображення понад 5 МБ - повідомлення про помилку (FileStorageService);
- недоступність WebSocket - індикатор очікування з'єднання в чаті, можливість роботи через REST;
- запит до захищеного API без токена - HTTP 401.

Результат: система коректно обробляє помилкові дії користувача та не порушує цілісність даних.

2.5.8. Перевірка API та автоматизація

Перевірено REST API за допомогою Swagger UI: наявність endpoints auth, services, orders, chat, reviews; авторизація через Bearer JWT. Запит GET /actuator/health - статус UP, компонент БД у стані UP.

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		53

Запуск `mvn test` використовувався для перевірки стабільності backend-проєкту (інтеграційні тести за наявності). Повна автоматизація UI (Playwright, Cypress) у межах дипломного проєкту не реалізована; рекомендована для подальшого розвитку.

2.5.9. Виявлені дефекти та їх усунення

У процесі розробки та тестування виявлено й усунено такі проблеми:

- некоректне відображення суми чека на картці замовлення (узгодження DTO, orderNormalize, saveAndFlush на backend);
- відображення часу в чаті - виправлено форматування на frontend;
- недоступність або некоректний стан health actuator - налаштовано для демонстрації.

На момент завершення тестування критичних дефектів, що блокують основний сценарій, не виявлено.

2.5.10. Висновок з тестування

За результатами тестування вебсайт Ріхора задовольняє основні функціональні вимоги технічного завдання (розділ 1.2.3). Підтверджено працездатність каталогу послуг і категорій, оформлення та супроводу замовлень, формування чека й оплати, обміну повідомленнями в чаті, системи відгуків, а також панелей дизайнера та адміністратора. Окремо перевірено механізм автентифікації JWT, рольовий доступ (клієнт, дизайнер, адміністратор), відновлення пароля через e-mail і публічний профіль дизайнера.

Усі сценарії з таблиці 2.4 виконано з позитивним результатом (статус «+»): перевірено публічні та захищені розділи сайту, повний цикл «замовлення - чек - оплата», роботу чату, відгуків, профілю та панелей дизайнера й адміністратора. Негативні перевірки (невалідний логін, доступ

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
						54
Зм.	Арк.	№ докум.	Підпис	Дата		

до /orders і /admin без токена або з чужою роллю) дали очікувану відмову, що підтверджує коректність JWT-автентифікації та розмежування прав.

Виявлені під час розробки недоліки - відображення суми чека на картці замовлення, формат часу в чаті, налаштування ендпоінта actuator/health - усунено до завершення циклу тестування; після виправлень регресійно повторено сценарії замовлення, виставлення чека та підтвердження оплати. Тестування проводилось у середовищі розробника (localhost, PostgreSQL у Docker) та підтверджено можливість роботи розгорнутої версії в мережі Інтернет (frontend на Vercel, backend на Render, база даних Neon).

Система в цілому готова до демонстрації на захисті кваліфікаційної роботи: ключові бізнес-процеси відповідають технічному завданню (розділ 1.2.3), критичних дефектів, що блокують основні сценарії, не виявлено. Для промислової експлуатації доцільно розширити автоматизовані тести (UI, API), впровадити регламент резервного копіювання та періодичний моніторинг стану сервісів.

Рекомендації для експлуатації: періодичний моніторинг GET /actuator/health, резервне копіювання БД і каталогу uploads/, оновлення залежностей Maven і npm для усунення вразливостей, розширення автоматизованих тестів перед production-розгортанням згідно з інструкціями розділу 3.

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		55

3 СПЕЦІАЛЬНИЙ РОЗДІЛ

Спеціальний розділ пояснювальної записки кваліфікаційної роботи на тему «Розробка вебсайту компанії дизайнерських послуг „Ріхога“» містить експлуатаційну документацію розробленого програмного продукту. У ньому послідовно розглянуто питання публікації сайту в мережі Інтернет, щоденної експлуатації та наповнення ресурсу контентом, а також заходів із його популяризації та підтримки користувачів.

Розділ поділено на три підрозділи, кожний з яких відповідає вимогам методичних рекомендацій щодо спеціального розділу кваліфікаційної роботи. У підрозділі 3.1 описано процедуру реєстрації та розміщення вебсайту на хостингу й перевірку його роботи за новою публічною адресою. У підрозділі 3.2 наведено інструкцію з експлуатації адміністративної частини сайту, основні функції та структуру меню, порядок створення й редагування контенту, а також мінімальні вимоги до конфігурації серверного середовища. У підрозділі 3.3 розкрито можливості та механізми просування популярності Ріхога в мережі Інтернет.

3.1. Інструкція з розміщення сайту в Інтернеті

У підрозділі 3.1 описано процедуру розміщення розробленого вебсайту Ріхога на хостингу: від підготовки репозиторію до отримання постійної адреси в Інтернеті та перевірки основних сценаріїв роботи (вхід, каталог, замовлення, чат) за новим доменним ім'ям. Матеріал відповідає фактичній схемі деплою на сервісах Neon, Render, Vercel і GitHub.

3.1.1. Призначення та вибір способу розміщення

Після завершення розробки та тестування (розділ 2) вебсайт Ріхога необхідно опублікувати в мережі Інтернет, щоб користувачі зверталися до нього за постійною адресою, незалежно від комп'ютера розробника. Для курсового проєкту обрано

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		56

хмарний хостинг: клієнт (SPA) - Vercel, сервер (REST API, WebSocket, e-mail) - Render (Docker), база даних - Neon (PostgreSQL EU). Код - на GitHub; після `git push` у `main` виконується автодеплой.

Новим доменним ім'ям у сенсі методички є безкоштовний піддомен Vercel, наприклад `https://pixora-orpin.vercel.app` - постійна URL для викладача та користувачів. Окремий домен `.ua` / `.com` не обов'язковий. Склад компонентів - таблиці 3.1.

Таблиця 3.1 – Компоненти розміщення вебсайту Pixora в мережі Інтернет

№	Компонент	Сервіс	Призначення
1	Frontend (SPA)	Vercel	React [1]/Vite [9], CDN, HTTPS
2	Backend (API)	Render (Docker)	Spring Boot 3.3 [2], Java 17
3	База даних	Neon	PostgreSQL [3], SSL
4	Код	GitHub	Репозиторій Pixora
5	Пошта	Gmail SMTP	коди, відновлення пароля

3.1.2. Підготовка проєкту до публікації

Перед реєстрацією на хостингах виконують підготовку репозиторію. Вихідний код розміщують на GitHub у гілці `main`; файли з паролями та секретами (`.env`, `render-env.local`) виключають із репозиторію за допомогою `.gitignore`. У корені проєкту має бути `Dockerfile` для багатоетапної збірки backend (Maven і JRE 17), оскільки Render не надає готового Java-runtime. Для frontend налаштовують `frontend/vercel.json` і збірку з копією `index.html` у `404.html`, щоб при оновленні сторінки в браузері на маршрутах `/orders`, `/chat` не виникала платформна помилка 404. Схема бази даних застосовується автоматично через Flyway [10] при старті backend у профілі `prod`. За потреби для демонстрації на захисті виконують одноразовий імпорт демонстраційних даних у Neon через `pg\_dump` і `psql`.

Локально доцільно перевірити збірку командою `mvn clean package` для серверної частини та `npm run build` у каталозі `frontend`. Усі файли, на які є

посилання в коді (імпорти модулів), повинні бути закомічені в Git - інакше хмарна збірка на Vercel завершується помилкою на кшталт «Could not resolve module».

3.1.3. Реєстрація та розміщення бази даних

Першим кроком публікації є розміщення структури та даних у керованій СУБД. На офіційному сайті neon.tech виконують реєстрацію (e-mail або через GitHub). У консолі обирають Create project, задають ім'я (наприклад, `pixora`) і регіон EU, щоб мінімізувати затримку при зверненнях з Render (Frankfurt).

У розділі Connection details фіксують параметри: ім'я хоста виду `er-...eu-central-1.aws.neon.tech`, база `neondb`, користувач `neondb\_owner`, пароль (кнопка Show password). Для підключення Spring Boot у production обов'язково формують JDBC-рядок, а не URI `postgresql://`:
`jdbc:postgresql://<host>/neondb?sslmode=require`

Параметр `sslmode=require` забезпечує шифроване з'єднання між Render і Neon. У SQL Editor виконують `SELECT 1;` - успішний результат підтверджує доступність БД. Якщо на етапі розробки вже наповнено локальну PostgreSQL, виконують експорт лише даних (`pg\_dump --data-only`) і імпорт у Neon через `psql` або pgAdmin, після чого перевіряють вхід тестовими обліковими записами.

3.1.4. Реєстрація та розміщення backend

На render.com підключають GitHub → New + → Web Service → репозиторій Pixora, гілка `main`. Параметри: Name `pixora-api`, Region Frankfurt, Runtime Docker, Dockerfile `./Dockerfile`, Build/Start - порожньо, Plan Free.

У Environment додають змінні з таблиці 3.2. Критичні: `JWT\_SECRET` (Base64 $\geq$ 256 біт), `CORS\_ORIGINS` (після Vercel - точний URL сайту), `MAIL\_` для Gmail (порт 465). Наведено у таблиці 3.2.

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		58

Таблиця 3.2 – Змінні середовища backend (Render) та параметри frontend (Vercel)

Змінна / параметр	Значення / примітка
SPRING_PROFILES_ACTIVE	prod
DATABASE_URL	JDBC Neon (jdbc:postgresql://...?sslmode=require)
DATABASE_USERNAME	neondb_owner
DATABASE_PASSWORD	пароль з Neon
JWT_SECRET	Base64-рядок не коротший за 256 біт
CORS_ORIGINS	URL Vercel без «/» в кінці
MAIL_ENABLED, MAIL_HOST, MAIL_PORT, MAIL_USERNAME, MAIL_PASSWORD, MAIL_FROM	Gmail SMTP, порт 465
Vercel: Root Directory	frontend
Vercel: VITE_API_URL	https://pixora-api.onrender.com/api

Create Web Service - очікування 10–20 хв. Перевірка: `.../actuator/health` → `"status": "UP"`. Free-тариф: sleep ~15 хв, перший запит до 60 с (cold start).

3.1.5. Реєстрація та розміщення frontend (Vercel)

На vercel.com → Import з GitHub → Root Directory `frontend`, Framework Vite, `VITE\_API\_URL` з таблиці 3.2 → Deploy. Production URL - нове доменне ім'я сайту (наприклад `pixora-orgin.vercel.app`). У Settings → Domains можна додати власний домен (опційно).

3.1.6. Зв'язування frontend і backend

На Render: `CORS\_ORIGINS` = точний URL Vercel → Manual Deploy. При

зміні API - Redeploy Vercel з новим `VITE\_API\_URL`. Frontend при завантаженні «прогріває» API запитом до health.

3.1.7. Перевірка роботи за новим доменним ім'ям

Перевірка виконується за публічною адресою Vercel, не localhost. Контрольний перелік - таблиці 3.3.

Таблиця 3.3 – Перевірка працездатності вебсайту після розміщення в Інтернеті

№	Дія	Очікуваний результат
1	Відкрити URL Vercel	Завантаження інтерфейсу, каталог
2	Оновити сторінку (F5) на /login, /orders	Немає платформного 404
3	Вхід або реєстрація	JWT, поява пунктів меню кабінету
4	Оформлення замовлення послуги	Запис у Neon, статус pending
5	Виставлення рахунку та оплата	Оновлення payment_status
6	Відкриття чату, відправка повідомлення	WebSocket, історія в чаті
7	«Забули пароль»	Лист на e-mail
8	Додавання відгуку	Запис у розділі відгуків

Бажано перевірити з ролей CLIENT і DESIGNER/ADMIN та з телефона.

Перевірка за новим доменним ім'ям є обов'язковим завершальним етапом розміщення: до цього моменту система могла працювати лише на 'localhost', недоступному викладачу. Після отримання Production URL на Vercel усі сценарії повторюють у браузері з інтернету (домашня мережа, мобільний 4G). Особливу увагу приділяють крос-доменній взаємодії: якщо frontend відкривається, а вхід повертає помилку мережі - причина зазвичай у `CORS\_ORIGINS` або невірному `VITE\_API\_URL`. Якщо каталог порожній - перевіряють підключення

					2026.KBP.122.423.10.00.00 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		60

Render→Neon і наявність даних після Flyway/імпорту. Успішне проходження таблиці 3.3 фіксують у пояснювальній записці як підтвердження готовності програмного продукту до експлуатації в мережі Інтернет.

3.1.8. Оновлення версії в мережі

`git commit` → `git push` → автодеплой → повтор таблиці 3.3. При помилці Vercel - перевірити, що всі файли в Git. Відкат - Promote to Production попереднього деплою на Vercel.

3.2. Інструкція з обслуговування та наповнення сайту

Підрозділ 3.2 присвячено щоденній експлуатації вебсайту Ріхога після його розміщення в мережі Інтернет: роботі з адміністративною та дизайнерською частинами інтерфейсу, наповненню каталогу контентом, супроводу замовлень і технічному обслуговуванню. Тут описано ролі користувачів, структуру меню, порядок створення та редагування матеріалів на сайті, а також мінімальні вимоги до конфігурації серверного середовища для локальної розробки та хмарної публікації.

3.2.1. Призначення та ролі

Після публікації вебсайт переходить у стадію експлуатації, що включає наповнення каталогу послуг і категорій, супровід замовлень на всіх етапах життєвого циклу (оформлення, виставлення чека, підтвердження оплати, завершення), ведення чату з клієнтами та збереження даних у хмарній базі PostgreSQL на платформі Neon. Адміністрування системи здійснюється через той самий веб-інтерфейс за маршрутом /admin, що спрощує керування контентом, користувачами та замовленнями без звернення до SQL-редактора. Дизайнери працюють у власній панелі /designer, а клієнти — через особистий кабінет і

					2026.KBP.122.423.10.00.00 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		61

розділи /orders, /chat, /profile.

Такий підхід не потребує встановлення додаткового програмного забезпечення на робочих місцях адміністратора: достатньо сучасного браузера та облікового запису з відповідною роллю. Централізований доступ до функцій керування забезпечує єдину точку входу, узгодженість даних між frontend, backend (Render) і базою даних, а також зручність щоденної експлуатації після розміщення сайту в мережі Інтернет (розділ 3.1). Ролі користувачів (клієнт, дизайнер, адміністратор) і їхні повноваження щодо розділів вебсайту наведено в таблиці 3.4.

Таблиця 3.4 – Ролі користувачів та основні розділи вебсайту Píxora

Роль	Код	Основні розділи	Ключові операції
Клієнт	CLIENT	/services, /orders, /chat, /reviews, /favorites, /payments, /profile	замовлення, оплата, чат, відгуки
Дизайнер	DESIGNER	/designer, /designers/:id	CRUD своїх послуг, рахунки, портфоліо
Адміністратор	ADMIN	/admin	категорії, усі послуги та замовлення, статистика

3.2.2. Основні функції та меню

Експлуатація сайту передбачає розуміння навігації для кожної ролі. Загальна шапка (Layout) містить логотип Píxora, перемикач світлої та темної теми, іконку сповіщень для авторизованих користувачів, аватар і випадаюче меню профілю; на мобільних пристроях доступне компактне меню.

Публічна частина включає головну сторінку ('/'), каталог послуг ('/services') з фільтрами, детальну картку послуги, публічний профіль дизайнера, а також сторінки входу, реєстрації та відновлення пароля. Після авторизації клієнт отримує доступ до розділів «Мої замовлення» (статуси, чек, оплата, файли), чату з

WebSocket і вкладеннями зображень, відгуків, обраного, платежів, профілю, налаштувань безпеки з підтвердженням e-mail-кодом і сповіщень.

Дизайнер працює в панелі `/designer`: переглядає статистику, створює та редагує лише власні послуги, обробляє замовлення за своїми послугами, змінює статуси, виставляє рахунок (Issue Invoice) і веде чат із замовником. Адміністратор у `/admin` бачить зведену статистику за категоріями, послугами, замовленнями, рейтингом і користувачами, а також керує всіма блоками адміністративної панелі, описаними в п. 3.2.3.

Навігація узгоджена з бізнес-процесом з рисунок 1.4: каталог → замовлення → чек → оплата → чат → відгук.

3.2.3. Експлуатація адміністративної частини

Категорії: перегляд таблиці, «Додати» (назва, опис), редагування, видалення (якщо немає залежних послуг).

Послуги (адмін): усі послуги каталогу; створення з прив'язкою до дизайнера та категорії; ціна, термін, зображення в `uploads/`; редагування/видалення будь-якої.

Замовлення (адмін): повний список; зміна статусу; виставлення рахунку; видалення; деталі в модальному вікні (файли, оплата).

Відгуки: перегляд для контролю якості.

Дизайнер керує лише своїми послугами та замовленнями; категорії глобально - лише ADMIN.

Адміністративна частина реалізує вимоги технічного завдання (розділ 1.2.3) щодо керування каталогом і замовленнями без залучення SQL-редактора для щоденних операцій. Адміністратор діє як контент-менеджер платформи: створює структуру категорій, контролює якість описів послуг, вирішує спірні ситуації з замовленнями. Дизайнер діє як виконавець: наповнює власну вітрину послуг і комунікує з клієнтом у чаті. Розділення ролей зменшує ризик випадкового

					2026.KBP.122.423.10.00.00 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		63

видалення чужих даних і відповідає принципу найменших привілеїв у інформаційній безпеці.

3.2.4. Створення та редагування контенту

Наповнення сайту виконується через веб-форми, без редагування HTML. Адміністратор у розділі «Категорії» адмін-панелі створює нову категорію (назва та опис) і зберігає запис. Дизайнер або адміністратор додає послугу: заповнює назву, опис, ціну, термін, обирає категорію, завантажує зображення - після збереження картка з'являється в каталозі. Клієнт на сторінці послуги натискає «Замовити» і за потреби додає вимоги, дедлайн і файл-приклад. Дизайнер або адміністратор змінює статус замовлення, за потреби виставляє рахунок; клієнт підтверджує оплату в інтерфейсі платежів. Дизайнер наповнює портфоліо в профілі - роботи відображаються на `/designers/:id`. Після завершення замовлення клієнт може залишити відгук у відповідному розділі.

Редагування категорій і послуг виконується через кнопку редагування в списку; зміни зберігаються в Neon, інтерфейс оновлюється через React Query. Замовлення клієнт не змінює з правами адміністратора - лише веде переписку в чаті та додає файли. У профілі змінюють ім'я, телефон, аватар і тему оформлення. Ролі DESIGNER та ADMIN призначаються через таблицю `user_roles` у базі даних або шляхом імпорту підготовленого дампу.

Створення категорії детальніше: після входу як ADMIN відкривають `/admin`, у блоці категорій натискають додавання, вводять назву українською (наприклад, «Брендинг») та короткий опис для SEO і фільтрації. Збереження відправляє POST-запит на API; при успіху категорія з'являється у випадяючому списку форми послуги.

Створення послуги: дизайнер у `/designer` або адмін у `/admin` заповнюють обов'язкові поля. Ціна зберігається як число; термін - у днях. Зображення завантажується окремим запитом; сервер повертає URL у `uploads/`. Після

					2026.KBP.122.423.10.00.00 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		64

збереження клієнт бачить картку в `/services` і може оформити замовлення.

Редагування існуючого контенту не відрізняється від створення: відкривають форму з заповненими полями, змінюють текст або файл, зберігають. Видалення послуги або категорії супроводжується підтвердженням у UI, щоб уникнути випадкових втрат. Історія замовлень при цьому може зберігатися згідно з налаштуваннями FK у БД (розділ 2).

3.2.5. Технічне обслуговування

Експлуатаційне обслуговування включає щотижневу перевірку доступності API запитом до `/actuator/health` на Render, щомісячне резервне копіювання бази даних Neon (вбудований Backup або `pg_dump`), перегляд логів Render і Vercel при збоях, оновлення коду після локального тестування та надсилання змін через `git push`. Після зміни публічної адреси сайту на Vercel обов'язково оновлюють `CORS_ORIGINS` на Render і `VITE_API_URL` на Vercel, після чого повторюють перевірку з таблиці 3.3. Раз на рік або при підозрі на компрометацію доцільно оновити `JWT_SECRET` і пароль SMTP.

Файли зображень у каталозі `uploads/` на безкоштовному Render зберігаються на епізодичному диску і можуть зникати після redeploy, тоді як посилання в базі даних залишаються. У такому разі критичні зображення завантажують повторно; для промислової експлуатації рекомендують постійне хмарне сховище (S3, Cloudinary).

3.2.6. Мінімальна конфігурація сервера для хостингу

Локально: Windows 10/11 (або Linux/macOS); RAM ≥ 8 ГБ (реком. 16); CPU 2+ ядра; диск 10+ ГБ; JDK 17, Node 18+, Maven 3.9+, опційно Docker + PostgreSQL 16; порти UI `3000`, API `8081`.

Хмара (фактично): Vercel Free - статика + CDN + HTTPS; Render Free -

					2026.KBP.122.423.10.00.00 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		65

Docker, ~512 МБ RAM, sleep 15 хв; Neon Free - PostgreSQL EU, ~0,5 ГБ. Це мінімум для публікації дипломного проєкту без VPS.

VPS (альтернатива): Linux x64, 2 ГБ RAM (реком. 4), 20 ГБ SSD, Java 17, PostgreSQL 14+ або Neon, nginx 80/443, Let's Encrypt. Для дипломної роботи обрано хмару через простоту.

Визначення мінімальної конфігурації необхідне методичкою для обґрунтування можливості експлуатації на доступному обладнанні. Локальні вимоги відповідають ПК розробника; хмарні - безкоштовному tier, достатньому для захисту дипломної роботи; VPS - запасному варіанту для організації, яка не використовує публічну хмару. У пояснювальній записці доцільно зазначити, що при зростанні навантаження (одночасні користувачі, великий обсяг `uploads/`) потрібне підвищення тарифу Render (більше RAM, без sleep) та постійне файлове сховище.

3.3. Інструкція з популяризації та підтримки сайту

Підрозділ 3.3 розкриває заходи з підвищення популярності вебсайту Ріхора в мережі Інтернет та організацію підтримки його користувачів. Описано вбудовані в застосунок механізми залучення та утримання аудиторії, зовнішнє просування (пошукова оптимізація, соціальні мережі), типові звернення користувачів і орієнтовний план дій на перші місяці експлуатації після публікації ресурсу.

3.3.1. Мета популяризації

Популяризація вебсайту Ріхора спрямована на залучення двох груп користувачів: замовників дизайнерських послуг (клієнтів) та виконавців (дизайнерів), які розміщують власні послуги в каталозі. Успішна популяризація підвищує довіру до платформи, стимулює повторні замовлення та формує позитивний імідж компанії в цифровому середовищі.

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
						66
Зм.	Арк.	№ докум.	Підпис	Дата		

Важливо забезпечити наповнений каталог, працездатність усіх сценаріїв і наочну демонстрацію циклу «каталог - замовлення - оплата - відгук». У подальшій комерційній експлуатації цільовою аудиторією є малі підприємства, SMM-фахівці, фріланс-дизайнери та приватні замовники в Україні, які шукають зручний сервіс замовлення дизайну без звернення до розрізнених каналів.

Популяризація не обмежується одноразовою рекламою: вона поєднує постійне наповнення контентом (п. 3.2), технічну стабільність хостингу та комунікацію з користувачами через чат і e-mail.

3.3.2. Вбудовані механізми

Ефективність просування сучасного вебзастосунку значно зростає, якщо сам продукт містить функції, що стимулюють повернення користувачів і поширення посилань. У Ріхога такі можливості реалізовані на етапі проєктування (розділи 1–2) і не вимагають підключення окремих платних рекламних модулів.

Система відгуків і рейтингу. Після завершення замовлення (статус `completed`) клієнт може залишити оцінку та текстовий коментар. Середній рейтинг відображається в каталозі та в адміністративній статистиці. Це створює соціальний доказ якості: нові відвідувачі охотіше оформлюють замовлення, бачачи позитивні відгуки на картках послуг.

Публічне портфоліо дизайнера. Кожен виконавець може наповнити портфоліо в особистому профілі; сторінка `/designers/:id` доступна без авторизації. Дизайнер може розміщувати це посилання в Instagram, Telegram або Facebook - так на платформу надходить реферальний трафік без окремого лендингу.

Обране (favorites). Клієнт зберігає цікаві послуги в список обраного і повертається до них пізніше, що збільшує кількість повторних сесій і ймовірність замовлення.

Е-mail-комунікація. Коди підтвердження при зміні пароля та листи відновлення доступу підсилюють довіру (офіційний канал зв'язку) і містять згадку про сервіс. Це підтримує імідж платформи навіть поза браузером.

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		67

Чат у реальному часі (WebSocket). Оперативна відповідь дизайнера або адміністратора знижує відтік клієнта до конкурентів; позитивний досвід спілкування підвищує ймовірність рекомендацій «на вустах».

Темна та світла тема, адаптивна верстка.Зручна робота зі смартфона розширює аудиторію, зокрема користувачів, які переходять із соціальних мереж з мобільного пристрою. На мобільній версії чату при відкритті розділу автоматично показується остання активна розмова, що зменшує бар'єр входу в комунікацію.

3.3.3. Просування в Інтернеті

SEO: повні описи послуг («логотип», «брендинг»), семантичні URL, Google Search Console, CDN Vercel (враховувати cold start API).

Соцмережі: Instagram/Telegram з URL у bio; кейси портфолію; відео «3 кроки до замовлення»; студентські групи.

Партнерства: курси дизайну; знижка в описі послуги.

Пошукові системи індексують публічні сторінки Pixa, якщо вони доступні без авторизації та містять текстові описи. Тому адміністратор несе відповідальність за якість контенту категорій і послуг: унікальні заголовки, ключові слова, відсутність «порожніх» карток. Семантичні URL полегшують сприйняття посилання в месенджерах. Швидкість frontend на Vercel позитивно впливає на поведінкові фактори (менший відтік через довге завантаження UI); затримка API на Free Render компенсується індикаторами завантаження в інтерфейсі та попереднім «прогрівом» health-ендпоінту.

У соціальних мережах на етапі захисту дипломної роботи важливою є кількість переходів за посиланням на Production URL. Рекомендується зафіксувати дату публікації в академічних чатах і за можливості зберегти скріншоти статистики Vercel Analytics для додатку. Дизайнери платформи поширюють персональні сторінки `/designers/:id`, що доповнює централізоване просування без великого рекламного бюджету.

Практичні заходи зовнішнього просування включають наповнення каталогу

					2026.KBP.122.423.10.00.00 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		68

якісними описами та реєстрацію сайту в Google Search Console, ведення сторінок бренду в Instagram і Telegram з посиланням на Vercel у профілі, публікацію кейсів із портфоліо, розміщення URL у матеріалах захисту дипломної роботи та зазначення акції «перше замовлення» в описі обраної послуги.

3.3.4. E-mail, аналітика

Транзакційні листи (коди підтвердження, відновлення пароля) у поточній версії вже налаштовані через Gmail SMTP на Render (п. 3.1). Вони підтримують безпеку та контакт із брендом. Масові розсилки новин платформи (дайджест нових послуг, акції) у майбутніх версіях можна інтегрувати через сервіси Mailchimp, SendPulse або аналоги з дотриманням згоди користувача на розсилку.

Для обліку відвідуваності опційно підключають Vercel Analytics або Google Analytics 4: відстежують кількість сесій, джерела переходів, Core Web Vitals. Внутрішня статистика в адмін-панелі (`/admin`) показує кількість категорій, послуг, замовлень, середній рейтинг і активних користувачів - це доповнює зовнішню аналітику без додаткових витрат. На захисті доцільно продемонструвати хоча б один показник (наприклад, кількість замовлень або відгуків) з адмін-панелі.

3.3.5. Підтримка користувачів

Підтримка користувачів - необхідна умова утримання популярності: незадоволений клієнт швидше поширює негативний досвід, ніж задоволений - позитивний.

Первинні канали підтримки: вбудований чат між клієнтом і дизайнером або адміністратором у контексті замовлення; сторінка «Забули пароль» для відновлення доступу; втручання адміністратора через адмін-панель (зміна статусів замовлень, контроль відгуків). При масових скаргах на недоступність сайту перевіряють `GET /actuator/health` на Render і переглядають логи на Render та Vercel.

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
						69
Зм.	Арк.	№ докум.	Підпис	Дата		

Типові звернення та рекомендовані дії. Якщо користувач скаржиться на довге завантаження, пояснюють особливість cold start на безкоштовному тарифі Render (перший запит після паузи до 60 секунд) і рекомендують повторити дію через одну хвилину або попередньо відкрити головну сторінку. Якщо не надходить лист на e-mail, перевіряють коректність змінних `MAIL\_\*` на Render, наявність пароля застосунку Gmail і папку «Спам» у поштовій скриньці. Помилка 404 при оновленні сторінки в браузері (F5 на `/orders`, `/chat`) зазвичай означає відсутність або застарілий деплой `frontend/vercel.json` - потрібен redeploy з актуальним репозиторієм. Зникнення фотографій (аватар, прев'ю послуги) після redeploy backend пов'язане з епізодичним диском на Render Free - файл завантажують повторно через інтерфейс. Проблеми з чатом (повідомлення не надходять) перевіряють підключення WebSocket, значення `CORS\_ORIGINS`, а також чи не перебуває API в режимі sleep.

3.3.6. План заходів на перші три місяці

Для систематичного просування доцільно дотримуватися поетапного плану, що поєднує зовнішню видимість, наповнення контентом і технічну стабільність.

У перший місяць після публікації основний акцент - на видимості та наповненні платформи. Публікують Production URL у матеріалах захисту дипломної роботи, на слайдах презентації та в соціальних мережах. Наповнюють каталог не менше ніж п'ятьма послугами в різних категоріях, перевіряють проходження контрольних сценаріїв з таблиці 3.3, залучають 2–3 тестових користувачів для оформлення демонстраційних замовлень.

У другий місяць акцент зміщують на індексацію та зворотний зв'язок. Публікують 2–3 пости з кейсами портфоліо, підключають Google Search Console, збирають короткі опитування від 5 користувачів щодо зручності інтерфейсу (навігація, мобільна версія, чат). За результатами формують перелік можливих покращень для розділу 2 або для майбутніх релізів.

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
						70
Зм.	Арк.	№ докум.	Підпис	Дата		

У третій місяць пріоритет - стабільність та легка стимуляція замовлень. Виконують резервну копію бази даних Neon, оновлюють залежності npm/Maven після локального тестування, додають в опис однієї з послуг згадку про акцію «перше замовлення». Аналізують статистику адмін-панелі (кількість завершених замовлень, середній рейтинг).

3.3.7. Правові аспекти

Популяризація та збір персональних даних (e-mail, телефон, файли замовлень, повідомлення в чаті) повинні відповідати законодавству України про захист персональних даних. На сайті доцільно розмістити політику конфіденційності та інформацію про мету обробки даних, строк зберігання та права користувача.

Модерація відгуків і повідомлень у чаті адміністратором допомагає запобігти спаму, образам і недобросовісному контенту. При використанні зображень у портфоліо слід мати права на їх публікацію або згоду замовника.

Секрети (JWT, паролі SMTP, доступ до Neon) не публікують у відкритому репозиторії GitHub - лише в змінних середовища на Render. Це також стосується популяризації: у рекламних матеріалах не демонструють внутрішні ключі та паролі.

У підрозділі 3.3 визначено, що популяризація Pixora поєднує вбудовані можливості застосунку (відгуки, портфоліо, чат, сповіщення) із зовнішнім просуванням через SEO та соціальні мережі, а стабільна робота сайту підтримується зворотним зв'язком із користувачами та поетапним планом дій на перші три місяці. Для успішної презентації дипломної роботи достатньо публічної адреси Vercel, наповненого каталогу та демонстрації повного циклу замовлення; подальше масштабування можливе за рахунок власного домену, платного хостингу та постійного сховища файлів.

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		71

4 ЕКОНОМІЧНИЙ РОЗДІЛ

Метою економічної частини даного дипломного проєкту є проведення економічних розрахунків, спрямованих на визначення економічної ефективності розробки вебсайту компанії дизайнерських послуг «Ріхога» прийняття рішення про подальший розвиток і впровадження або ж недоцільність проведення відповідної розробки.

Об'єктом розробки є вебсайт компанії дизайнерських послуг «Ріхога».

Розрахунок вартості розробки виконується в декілька етапів:

- описати технологічний процес розробки із зазначенням трудомісткості кожної операції;
- визначити суму витрат на оплату праці основного і допоміжного персоналу, включаючи відрахування на соціальні заходи;
- обчислити витрати на електроенергію;
- нарахувати суму амортизаційних відрахувань;
- визначити суму накладних витрат;
- скласти кошторис та визначити собівартість робіт;
- розрахувати ціну робіт;
- визначити економічну ефективність та термін окупності.

4.1. Визначення стадій технологічного процесу та загальної тривалості проведення НДР

В цьому підрозділі розглянемо основні етапи технологічного процесу для розробки вебсайту компанії дизайнерських послуг «Ріхога». Для визначення загальної тривалості робіт дані витрат часу по операціях зведено в таблицю 4.1.

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
						72
Зм.	Арк.	№ докум.	Підпис	Дата		

Таблиця 4.1 - Середній час виконання робіт по обслуговуванню та стадії (операції) технологічного процесу

№ п/п	Назва операції (стадії)	Виконавець	Середній час виконання операції, год.
1	Планування та аналіз	Кер. проєкту (Pm)	10
		Інженер (ІІ)	10
2	Розробка технічного завдання	Кер. проєкту (Pm)	8
		Інженер (ІІ)	6
3	Дизайн інтерфейсу	Інженер (ІІ)	18
		Інженер (І2)	28
4	Розробка функціоналу	Інженер (ІІ)	48
5	Тестування та відладка	Тестувальник	12
6	Документування	Інженер (ІІ)	4
7	Розгортання та підтримка	Інженер (І2)	18
8	Управління проєктом	Кер. проєкту (Pm)	18
Разом			180

Сумарний час виконання операцій технологічного процесу становить 180 години.

4.2. Визначення витрат на оплату праці та відрахувань на соціальні заходи

У даному підрозділі проводиться аналіз і розрахунок витрат, пов'язаних з оплатою праці та відрахуваннями на соціальні заходи, що необхідні для розробки вебсайту компанії дизайнерських послуг «Ріхора».

Розмір заробітної плати залежить від складності та умов виконуваної роботи,

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		73

професійно-ділових якостей працівника, результатів його праці та діяльності підприємства.

Основна заробітна плата розраховується за формулою:

$$З_{\text{осн.}} = T_c \cdot K_r \quad (4.1)$$

де: T_c – тарифна ставка, грн. (приймаємо для керівника проекту (Рм) – 450 грн./год, інженера (І2) – 272 грн./год.), інженера (І1) – 113 грн./год., тестувальник – 100 грн./год.; K_r – кількість відпрацьованих годин.

Отже, основна заробітна плата для:

Керівника проекту (Рм) $З_{\text{осн}2} = 36 \cdot 450 = 16\,200$ грн.

Інженера (І2) $З_{\text{осн}4} = 46 \cdot 272 = 12\,512$ грн.

Інженера (І1) $З_{\text{осн}3} = 86 \cdot 113 = 9\,718$ грн.

Тестувальник $З_{\text{осн}4} = 12 \cdot 100 = 1\,200$ грн.

Сумарна основна заробітна плата становить

$$З_{\text{осн}} = 16\,200 + 12\,512 + 9\,718 + 1\,200 = 39\,630 \text{ грн.}$$

Додаткова заробітна плата становить 10 – 15 % від суми основної заробітної плати.

$$З_{\text{дод.}} = З_{\text{осн.}} \cdot K_{\text{допл.}} \quad (4.2)$$

де: $K_{\text{допл.}}$ – коефіцієнт додаткових виплат працівникам.

Отже додаткова заробітна плата по категоріях працівників становить:

Керівника проекту $З_{\text{дод}2} = 16\,200 \cdot 0,13 = 2\,106$ грн.

Інженера (І2) $З_{\text{дод}3} = 12\,512 \cdot 0,13 = 1\,627$ грн.

Інженера (І1) $З_{\text{дод}4} = 9\,718 \cdot 0,13 = 1\,263$ грн.

Тестувальник $З_{\text{дод}4} = 1\,200 \cdot 0,13 = 156$ грн.

Загальна додаткова заробітна плата становить:

$$З_{\text{дод}} = 2\,106 + 1\,627 + 1\,263 + 156 = 5\,152 \text{ грн.}$$

Звідси загальні витрати на оплату праці ($B_{\text{о.п.}}$) визначаються за формулою:

$$B_{\text{о.п.}} = З_{\text{осн.}} + З_{\text{дод.}} \quad (4.3)$$

$$B_{\text{о.п.}} = 39\,630 + 5\,152 = 44\,782 \text{ грн.}$$

Єдиний соціальний внесок (ЄСВ – 22%) визначається за формулою:

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
						74
Зм.	Арк.	№ докум.	Підпис	Дата		

$$B_{\text{ECB}} = B_{\text{оп}} * 0,22 \quad (4.4)$$

$$B_{\text{ECB}} = 44\,782 * 0,22 = 9\,852 \text{ грн.}$$

Проведені розрахунки витрат на оплату праці наведено у таблиці 4.2.

Таблиця 4.2 – Зведені розрахунки витрат на оплату праці

№ п/ п	Категорія працівників	Основна заробітна плата, грн.			Додаткова заробітна плата, грн.	ЄСВ, грн.	Всього витрати на оплату праці, грн. $6 = 3+4+5$
		Тарифна ставка, грн.	К-сть годин	Фактично нарах. зарплати, грн.			
		1	2	3	4	5	6
1	Кер. проекту (Рм)	450	36	16 200	2 106		
2	Інженера (І2)	272	46	12 512	1 627		
3	Інженера (І1)	113	86	9 718	1 263		
4	Тестувальник	100	12	1 200	156		
Разом				39 630	5 152	9852	54 634

Отже, загальні витрати на оплату праці становлять 54 634 грн.

4.3. Розрахунок витрат на електроенергію

Розрахуємо вартість електроенергії. Затрати на електроенергію 1-ці обладнання визначаються за формулою:

$$Z_{\text{в}} = W * T * S \quad (4.5)$$

де: W – необхідна потужність, кВт; T – кількість годин роботи обладнання;
 S – вартість кіловат-години електроенергії (приймаємо 15, 94 грн).

В нашій системі є 1 ПК. Витрати на електроенергію для цього комп'ютера обчислимо окремо, взявши за основу, що час роботи обладнання обчислюється в залежності від виконуваних робіт (згідно таблиці 4.1) і споживані потужності

наступні: комп'ютер – 0,82 кВт/год.

$$З_{ек} = 0,82 * 180 * 15,94 = 2353 \text{ грн.}$$

Витрати на електроенергію становлять 2353 грн.

4.4. Розрахунок суми амортизаційних відрахувань розробки вебсайту компанії дизайнерських послуг «Pixora»

Характерною рисою основних фондів (засобів праці) є те, що вони функціонують у процесі виробництва протягом тривалого часу, зберігаючи при цьому свою початкову натуральну форму. Проте під впливом часу, навантажень та науково-технічного прогресу вони неминуче зношуються.

Оскільки підприємство не може миттєво замінити зношений верстат чи будівлю без значних фінансових втрат, економічна система передбачає їх поступове відшкодування у вартісній формі. Цей процес забезпечується через амортизацію.

Амортизація – це процес перенесення вартості основних фондів на вартість новоствореної продукції з метою їх повного відновлення

Комп'ютери та оргтехніка належать до четвертої групи основних фондів.

Амортизація на них нараховується лише в випадку, якщо мінімально допустимі строки їх корисного використання 2 роки. Для визначення амортизаційних відрахувань застосовуємо формулу:

$$A = \frac{Б_В * Н_А}{100\%} * T, \quad (4.6)$$

де: А – амортизаційні відрахування за звітний період, грн.;

Б_В – балансова вартість групи основних фондів на початок звітного періоду, грн.;

Н_А – норма амортизації, 0,04 %.

Оскільки для написання програми та її тестування використовується один ПК, вартістю 30000,00 грн., то сума амортизаційних відрахувань становитиме:

$$A = \frac{30\,000,00 * 0,04}{150} * 180 = 1\,440 \text{ грн.}$$

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
						76
Зм.	Арк.	№ докум.	Підпис	Дата		

4.5. Обчислення накладних витрат

Накладні витрати пов'язані з обслуговуванням виробництва, утриманням апарату управління підприємства (фірми) та створення необхідних умов праці. В залежності від організаційно-правової форми діяльності господарюючого суб'єкта, накладні витрати можуть становити 20–60 % від суми основної та додаткової заробітної плати працівників.

$$H_B = B_{o.п.} * 0,4 \quad (4.7)$$

де: H_B – накладні витрати.

$$H_B = 44\,782 * 0,4 = 17\,913 \text{ грн.}$$

4.6. Складання кошторису витрат та визначення собівартості

Для складання кошторису витрат та визначення собівартості, результати проведених вище розрахунків зведемо у таблиці 4.4.

Таблиця 4.4 - Кошторис витрат розробки вебсайту компанії дизайнерських послуг «Ріхора»

№	Зміст витрат	Сума, грн.	В % до загальної суми
1.	Витрати на оплату праці	54 634	72
2.	Витрати на електроенергію	2 353	3
3.	Амортизаційні відрахування	1 440	2
4.	Накладні витрати	17 913	23
5.	Собівартість	76 340	100

Собівартість (C_B) НДР розрахуємо за формулою:

$$C_B = B_{o.п.} + B_{c.з.} + 3e + A + H_B \quad (4.8)$$

Отже, собівартість дорівнює $C_B = 76\,340$ грн.

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
						77
Зм.	Арк.	№ докум.	Підпис	Дата		

4.7. Розрахунок ціни розробки вебсайту компанії дизайнерських послуг «Ріхора»

Розрахунок ціни науково-дослідної роботи включає в себе урахування різноманітних факторів, таких як рівень рентабельності, собівартість та податкова ставка.

Ціну робіт можна визначити за формулою:

$$Ц = C_{\text{в}} * (1 + P_{\text{рен}}) * (1 + \text{ПДВ}), \quad (4.9)$$

де: $C_{\text{в}}$ – собівартість; $P_{\text{рен}}$ – рівень рентабельності; ПДВ – ставка податку на додану вартість.

$$Ц = 76\,342 * (1 + 0,3) * (1 + 0,2) = 119\,090 \text{ грн.}$$

4.8. Визначення економічної ефективності і терміну окупності капітальних вкладень

Ефективність виробництва – це узагальнене і повне відображення кінцевих результатів використання робочої сили, засобів та предметів праці на підприємстві за певний проміжок часу.

Для визначення ефективності продукту розраховують чисту теперішню вартість (ЧТВ) і термін окупності (Ток).

$$\text{ЧТВ} = -C_{\text{в}} + \sum_{i=1}^t \frac{\Gamma_{\text{п}}}{(1 + i)^t}, \quad (4.10)$$

де: $C_{\text{в}}$ – собівартість розробки; $\Gamma_{\text{п}}$ – грошовий потік за t – ий рік; t – відповідний рік проекту; i – величина дисконтної ставки (12%).

$$\text{ЧТВ} = -76\,342 + \frac{42\,750}{(1 + 0,12)^1} + \frac{42\,750}{(1 + 0,12)^2} + \frac{42\,750}{(1 + 0,12)^3} = 23\,433 \text{ грн}$$

Якщо $\text{ЧТВ} \geq 0$, то проект може бути рекомендований до впровадження.

Термін окупності визначається за формулою:

$$T_{\text{ок}} = T_{\text{пв}} + \frac{H_{\text{в}}}{\Gamma_{\text{пп}}} \quad (4.11)$$

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
						78
Зм.	Арк.	№ докум.	Підпис	Дата		

де: ТПВ – період до повного відшкодування витрат, років; НВ – невідшкодовані витрати на початок року, грн.; ГПР – грошовий потік на початок року, грн.

$$T_{ок} = 2 + \frac{7\,102}{42\,750} = 2,2 \text{ р.}$$

Всі дані внесемо в зведену таблицю 4.5.

Таблиця 4.5 – Техніко-економічні показники розробки вебсайту компанії дизайнерських послуг «Ріхога»

№ п/п	Показник	Значення
1.	Собівартість, грн.	76 340
2.	Плановий прибуток або грошовий потік, грн.	42 750
3.	Ціна, грн.	119 090
4.	Чиста теперішня вартість, грн.	23 433
5.	Термін окупності, рік	2,2

Прибутковість проекту та термін окупності свідчать про його фінансову ефективність та здатність повернути капітальні вкладення протягом 1,8 року. Отже, на основі отриманих показників можна зробити висновок, що розробка вебсайту компанії дизайнерських послуг «Ріхога» є доцільною з економічної точки зору.

5 ОХОРОНА ПРАЦІ, ТЕХНІКА БЕЗПЕКИ ТА ЕКОЛОГІЧНІ ВИМОГИ

Розділ 5 пояснювальної записки дипломної роботи на тему «Розробка вебсайту компанії дизайнерських послуг „Ріхога“» розкриває питання безпечної організації праці під час створення, тестування та експлуатації програмного продукту, техніки безпеки при роботі з обчислювальною технікою та мережевою інфраструктурою, а також екологічні наслідки обраної технологічної схеми (локальна розробка, хмарний хостинг Neon, Render, Vercel). Зміст узгоджений із основним напрямом кваліфікаційної роботи (розділи 1–3) і відповідає типовому завданню консультанта з охорони праці для студентів спеціальностей, пов’язаних із розробкою програмного забезпечення та вебтехнологій.

У розділі послідовно розглянуто: нормативно-правову базу; характеристику робочого процесу та небезпечних і шкідливих виробничих факторів; вимоги ергономіки робочого місця програміста; техніку безпеки при експлуатації персональних комп’ютерів і периферії; організацію безпечної експлуатації вебсайту Ріхога для користувачів різних ролей; екологічні вимоги та оцінку енергоспоживання. Для підтвердження відповідності робочого місця нормам наведено розрахунок освітленості та оцінку річного споживання електроенергії робочим місцем розробника; окремі положення зведено в таблиці 10–12.

5.1. Нормативно-правова база та організація безпечної праці

Безпека праці при виконанні кваліфікаційної роботи регулюється Законом України «Про охорону праці», Кодексом законів про працю України, Законом України «Про основні засади державної політики у сфері охорони навколишнього природного середовища», а також галузевими нормативними документами з охорони праці (Державні нормативні акти охорони праці - ДНАОП), будівельними нормами (ДБН) та державними стандартами (ДСТУ) у частині, що стосується освітлення, електробезпеки та ергономіки.

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
						80
Зм.	Арк.	№ докум.	Підпис	Дата		

Для робіт, пов'язаних із використанням обчислювальної техніки, застосовують положення ДНАОП 0.03-3.17-88 «Безпека при роботі з обчислювальною технікою» (у діючих методичних матеріалах університету часто посилаються на цей акт разом із оновленими рекомендаціями Міністерства охорони здоров'я щодо медичних вимог до організації роботи операторів і користувачів відеодисплейними терміналами). Ергономічні вимоги до програмних засобів і робочих місць узгоджують із міжнародним стандартом ISO 9241 (ергономіка взаємодії людини з системами) та ДСТУ ISO 9241-5:2006 (ергономічні вимоги до роботи з дисплеями).

Освітлення приміщень, де розташовано робочі місця з відеодисплейними терміналами, нормується ДБН В.2.5-28:2018 «Освітлення. Основні положення» (категорія візуальної роботи - уточнюється за характером завдань; для роботи з екраном переважно застосовують комбіноване освітлення з обмеженням відблисків на моніторі).

Електробезпека побутових і офісних електроустановок - за Правилами технічної експлуатації електроустановок споживачів та Правилами улаштування електроустановок, затвердженими наказами Міністерства енергетики. Пожежна безпека - за Правилами пожежної безпеки в Україні, затвердженими наказом Міністерства внутрішніх справ.

У сфері обробки персональних даних користувачів вебсайту Ріхога (реєстрація, профіль, замовлення, чат) додатково враховують Закон України «Про захист персональних даних» та вимоги Закону України «Про основні засади забезпечення кібербезпеки України» - це не замінює охорону праці, але визначає безпечну поведінку при експлуатації інформаційної системи.

Організація праці під час виконання дипломного проєкту включає: робоче місце студента-розробника (домашнє або навчальне), періодичну роботу в комп'ютерному класі університету, віддалену взаємодію з хмарними сервісами без фізичного доступу до серверної. Відповідальність за безпечний стан обладнання, заземлення розеток, вільні проходи та наявність засобів пожежогасіння несе

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		81

роботодавець (заклад освіти) або власник приміщення при домашній роботі; студент зобов'язаний дотримуватися інструкцій з охорони праці, проходити інструктаж і використовувати обладнання за призначенням. Наведено у таблиці 5.1.

Таблиця 5.1 – Основні нормативні документи, що використані при розробленні розділу 5

№	Назва документа	Номер / дата затвердження	Сфера застосування в роботі над Ріхора
1	2	3	4
1	Закон України «Про охорону праці»	2694-ХІІ від 14.10.1992	загальні вимоги до безпечних умов праці студента-розробника
2	Кодекс законів про працю України	від 10.12.1971	режим праці, відпочинок, відповідальність
3	ДНАОП 0.03-3.17-88 «Безпека при роботі з обчислювальною технікою»	Наказ Держнаглядохоронпраці	робота з персональним комп'ютером, монітором, периферією
4	ДБН В.2.5-28:2018 «Освітлення. Основні положення»	наказ Мінрегіонбуду	норми освітлення робочого місця програміста (розрахунок 1)
5	ДСТУ ISO 9241-5:2006 «Ергономічні вимоги до роботи з дисплеями»	-	відстань до екрана, положення тіла, перерви
6	Правила технічної експлуатації електроустановок споживачів	наказ Міненерговугілля	електробезпека 220 В, мережевий фільтр, заземлення

Продовження таблиці 5.1

1	2	3	4
7	Правила пожежної безпеки в Україні	наказ МВС № 1417 від 30.12.2014	пожежна безпека в приміщенні комп'ютерами
8	Закон України «Про захист персональних даних»	2297-VI від 01.06.2010	обробка даних користувачів сайту (профіль, замовлення, чат)
9	Закон України «Про основні засади державної політики у сфері охорони навколишнього природного середовища»	2813-XII від 25.06.1991	екологічні вимоги, раціональне використання ресурсів
10	Закон України «Про основні засади забезпечення кібербезпеки України»	2163-VIII від 05.10.2017	безпечна експлуатація вебсайту, захист від несанкціонованого доступу

5.2. Характеристика робочого процесу та виробничих факторів

Об'єктом дослідження є умови та безпека праці ІТ-спеціалістів у процесі повного життєвого циклу (створення, тестування, публікації та супроводу) вебсайту Ріхора. Оцінці підлягають небезпечні та шкідливі виробничі фактори під час кодування (Java, TypeScript), роботи в IDE, локального запуску Spring Boot (порт 8081), Vite (порт 3000), Docker (PostgreSQL), деплою на Render/Vercel та адміністрування через вебінтерфейси Neon, GitHub, Gmail.

Види діяльності згідно з розділами 2–3 кваліфікаційної роботи:

– Проєктування та програмування - тривала робота за дисплеєм, висока зорова та розумова напруга, статичне положення тіла, повторювані рухи кистей.

- Тестування - багатократне проходження сценаріїв, перевірка адаптивності на різних розмірах екрана, можлива тривала робота в вечірній час.
- Публікація та експлуатація - налаштування змінних середовища, робота з секретами (паролі бази даних, ключ JSON Web Token), контроль логів; фізичне навантаження нижче, ніж при програмуванні, але зростає інформаційний ризик при помилковому розкритті секретів.
- Експлуатація користувачами (клієнт, дизайнер, адміністратор) - робота через браузер: каталог, замовлення, чат WebSocket, завантаження зображень; для дизайнерів - перегляд портфоліо, тривала робота з кольором на моніторі.
- Небезпечні виробничі фактори (можуть призвести до травми):
 - електричний струм - при пошкодженні кабелів живлення, неправильному підключенні зарядних пристроїв, відсутності заземлення в старих електромережах;
 - пожежа - перегрів блоків живлення, пил у системному блоці, неправильне використання мережевих фільтрів і подовжувачів;
 - падіння - заплутані кабелі під столом у навчальному класі;
 - психофізіологічне перевантаження - при аварійному дедлайні, тривалій роботі без перерв (ризик помилок у коді та конфігурації деплою).
- Шкідливі виробничі фактори (вплив на здоров'я при тривалій експозиції):
 - напруження зору - дрібний шрифт коду, низький контраст темної теми редактора, багатогодинна робота без відпочинку для очей;
 - низька рухливість - сидяче положення понад 2–3 години без перерви;
 - електромагнітні пол - монітор, системний блок, маршрутизатор Wi-Fi (рівні, як правило, нижчі за гранично допустимі при дотриманні відстаней);

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
						84
Зм.	Арк.	№ докум.	Підпис	Дата		

- шум - вентилятори комп'ютера, кондиціонер, шум у комп'ютерному класі (зазвичай нижче допустимих норм у бібліотеці, але при домашній роботі біля вікна з вуличним шумом варто враховувати фон);
- нервово-психічне напруження - відповідальність за здачу диплома, діагностика помилок збірки на Vercel, cold start API на Render;
- інформаційне перевантаження - одночасна робота з документацією, месенджерами, електронною поштою. Наведено у таблиці 5.2

Таблиця 5.2 – Небезпечні та шкідливі фактори на етапах життєвого циклу Píxora

Етап життєвого циклу	Тип фактору	Фактор	Можливі наслідки	Заходи зниження ризику
1	2	3	4	5
Розробка (розділ 2)	шкідливий	напруження зору	втома очей, головний біль	перерви 10–15 хв кожні 2 год, освітлення 300–500 лкс
Розробка (розділ 2)	шкідливий	низька рухливість	болі в спині, шиї	ергономічне крісло, гімнастика
Розробка (розділ 2)	небезпечний	електричний струм 220 В	ураження струмом	справні кабелі, мережевий фільтр, не розбирати блок живлення під напругою

Продовження таблиці 5.2

1	2	3	4	5
Тестування (п. 2.5)	шкідливий	зорове та психоемоційн е навантаження	пропуск дефектів	сценарії порціями по 45–60 хв
Деплой (п. 3.1)	небезпечни й (інформаці йний)	витік секретів	компромет ація бази даних	паролі лише в змінних середовища Render, не в Git
Експлуатація (п. 3.2–3.3)	шкідливий	робота за дисплеєм	втома зору	адаптивний інтерфейс, темна і світла тема

Хмарна схема розміщення зменшує потребу студента в обслуговуванні власної серверної (шум, клімат, підвищена електробезпека серверних стійок), але не усуває фактори на робочому місці розробника та фактори для кінцевих користувачів, які працюють з сайтом через браузер.

5.3. Ергономіка та охорона здоров'я при роботі з персональним комп'ютером

Робоче місце для розробки вебсайту Ріхога повинно відповідати вимогам ДНАОП 0.03-3.17-88 та рекомендаціям щодо роботи з відеодисплейними терміналами: стіл з регульованою висотою або фіксованою за зростом, крісло з опорою для попереку, монітор на відстані 50–70 см від очей, верхній край екрана на рівні очей або трохи нижче, клавіатура на висоті ліктів, достатнє освітлення без відблисків на матриці.

Режим праці та відпочинку. При роботі з дисплеєм більше 2 годин безперервно необхідні перерви 10–15 хвилин (гімнастика для очей і тіла, зміна

фокусу погляду у вікно). Загальна тривалість роботи за комп'ютером у навчальний день не повинна перевищувати норм, установлених для студентів закладом вищої освіти (зазвичай узгоджується з навчальним планом і санітарними правилами). Під час інтенсивного тестування Ріхога доцільно чергувати сценарії: після 45–60 хвилин перевірки інтерфейсу - коротка перерва, щоб знизити втому зору та кількість пропущених дефектів.

Організація програмного середовища. У проєкті використовують світлу та темну тему інтерфейсу сайту; для редактора коду рекомендується контрастна тема без надмірно яскравих кольорів, розмір шрифту не менше 12–14 пунктів, міжрядковий інтервал достатній для розрізнення символів. Два монітори (код + браузер) зменшують втому від постійного перемикання вікон, але вимагають повороту голови - обидва екрани розташовують під однаковим кутом огляду.

Освітлення робочого місця . Перевірка відповідності нормам ДБН В. 2.5-28:2018 для робочого місця програміста в навчальній кімнаті або домашньому кабінеті виконується методом розрахунку освітленості (E) , лкс.

Для роботи з відеодисплейними терміналами (категорія змісту робіт - висока зорова напруга, розрізнення дрібних деталей на екрані) норми комбінованого освітлення зазвичай вимагають не менше 300–500 лкс на горизонтальній поверхні столу разом із обмеженням прямого відблиску на екрані. Отримане значення 231 лкс нижче рекомендованого для постійної професійної роботи оператора, але може бути підвищене додатковою настільною лампою з розсіювачем (локальне освітлення документів, не спрямоване на монітор) та природним освітленням від вікна. З урахуванням денного світла (додатково 100–150 лкс) сумарна освітленість наближається до 350–400 лкс, що відповідає вимогам.

Середню освітленість від штучного загального освітлення визначають за формулою:

$$E = \frac{(N * \Phi * \eta)}{(S \cdot K_3)} \quad (5.1)$$

Підставляємо вихідні дані: площа приміщення $S = 12 \text{ м}^2$, кількість світильників $N = 2$ шт., світловий потік одного світильника $\Phi = 3200 \text{ лм}$, коефіцієнт

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		87

використання $\eta = 0,65$, коефіцієнт запасу $K_3 = 1,5$.

$$E(\text{штуч}) = \frac{(2 * 3200 * 0,65)}{(12 * 1,5)} = \frac{4160}{18} \\ = 231 \text{ лкс.}$$

З урахуванням природного освітлення з вікна (оцінка $E_{\text{прир}} = 120$ лкс):

$$E_{\text{сума}} = 231 + 120 = 351 \text{ лкс.}$$

Норма для роботи з відеодисплейним терміналом за ДБН В.2.5-28:2018 - 300–500 лкс, отже в денний час умови відповідають вимогам. Для вечірньої роботи доцільна настільна лампа (~80 лкс):

$$E_{\text{вечір}} = 231 + 80 = 311 \text{ лкс.}$$

Висновок за розрахунком освітленості: робоче місце розробника Ріхога при наявності двох світильників загального освітлення та денного світла відповідає нормам за величиною (E) ; необхідно контролювати блики на екрані (розташування монітора перпендикулярно до вікна, регулювання яскравості).

Мікроклімат. Температура в приміщенні з комп'ютерами 18–25 °С, відносна вологість 40–60 %. Системний блок і ноутбук віддають тепло - забезпечують провітрювання; на Render і Vercel обчислення виконуються віддалено, тому локальне тепловиділення знижене порівняно з власною серверною.

5.4. Техніка безпеки при експлуатації обчислювальної техніки

Електробезпека. Персональний комп'ютер і монітор живляться від мережі 220 В змінного струму. Користувач не має права розбирати блок живлення під напругою, ремонтувати кабелі без кваліфікації електрика, перевантажувати подовжувачі потужністю обігрівачів і чайників у тій самій розетці. Обладнання підключають через мережевий фільтр із захистом від перенапруги; кабелі прокладають так, щоб не створювати ризику спотикання. При переносі ноутбука для демонстрації Ріхога на захисті - відключення зарядного пристрою від розетки перед перенесенням кабелю.

Пожежна безпека. Заборонено закривати вентиляційні отвори системного

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		88

блока, курити в приміщенні з папером і технікою, залишати увімкнене обладнання без нагляду на тривалий час (особливо при навантаженні збіркою Docker). У навчальному класі знають розташування вогнегасника та шлях евакуації. Пил на вентиляторах підвищує ризик перегріву - періодичне очищення обладнання.

Безпека при роботі з мережею Інтернет. Під час деплою (п. 3.1) студент працює з обліковими записами Neon, Render, Vercel, GitHub. Техніка безпеки інформаційного характеру: не передавати паролі третім особам, не публікувати `JWT\_SECRET` і паролі бази даних у репозиторії, використовувати HTTPS (на продакшені Vercel і Render забезпечують сертифікати), складні паролі, двофакторну автентифікацію на GitHub за можливості. Фішингові листи, що імітують хмарні сервіси, можуть призвести до втрати даних користувачів Pixora - перевіряти домен відправника.

Безпека при локальному запуску. У режимі розробки API доступний на `localhost:8081`; не відкривати порти в локальну мережу без потреби, щоб уникнути несанкціонованого доступу до тестової бази з персональними даними демокористувачів.

WebSocket і файлове сховище. Живий чат і каталог `uploads/` для аватарів не створюють додаткових фізичних ризиків, але вимагають контролю розміру файлів і типів зображень, щоб уникнути перевантаження диска сервера та шкідливих вкладень - це технічна безпека програмного продукту, узгоджена з п. 2.4–2.5.

5.5. Охорона праці та безпека користувачів вебсайту Pixora

Після публікації вебсайт експлуатують клієнти, дизайнери та адміністратор (таблиця 3.4, розділ 3). З точки зору охорони праці це користувачі відеодисплейних терміналів, які не обслуговують серверну апаратуру.

Клієнт оформлює замовлення, спілкується в чаті, завантажує файли, оплачує послугу. Ризики: тривала робота з телефону (напруження ший при нахилі голови), ослаблення зору в темній кімнаті при яскравому екрані. Інтерфейс Pixora

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		89

адаптивний (п. 3.3), що зменшує незручності на мобільному пристрої.

Дизайнер веде каталог послуг, портфолію, відповідає в чаті, переглядає зображення. Додатковий фактор - колірне навантаження при оцінці макетів; рекомендовано калібрування монітора або перегляд на кількох пристроях перед здачею макета клієнту.

Адміністратор керує користувачами, категоріями, статистикою. Відповідальність за коректність даних і конфіденційність (не розголошувати персональні дані клієнтів поза системою) є частиною безпечної експлуатації.

Захист персональних даних реалізовано на рівні програмного забезпечення: автентифікація JSON Web Token [4], ролі, підтвердження зміни пароля кодом на електронну пошту (розділ 2). Користувачеві нагадують про надійний пароль і не використання одного пароля на сторонніх сайтах - це знижує ризик компрометації облікового запису.

Психологічне навантаження в чаті замовлень може зростати при конфліктах - адміністратор і дизайнер повинні дотримуватися етики спілкування; платформа фіксує історію повідомлень для прозорості, але не замінює професійну комунікацію.

5.6. Екологічні вимоги та раціональне використання ресурсів

Розробка та експлуатація вебсайту Ріхога мають помірний екологічний слід порівняно з друкованими каталогами послуг і офісною документацією на папері.

Зменшення споживання паперу. Замовлення, чеки, статуси, чат і файли зберігаються в електронному вигляді в PostgreSQL і файловому сховищі; клієнт не зобов'язаний друкувати договір для типового замовлення дизайну. Це відповідає тенденції «зеленого офісу» та зменшує викиди, пов'язані з виробництвом і утилізацією паперу.

Енергоспоживання робочого місця розробника. Оцінка річного споживання електроенергії при роботі над дипломним проектом.

					2026.KBP.122.423.10.00.00 ПЗ	Арк.
						90
Зм.	Арк.	№ докум.	Підпис	Дата		

Річне споживання кожного приладу:

$$W = \frac{N * t}{1000 \left(\frac{\text{кВт}}{\text{год}} \right)} \quad (5.2)$$

$W_1 = (120 * 1200) / 1000 = 144$ кВт·год - ноутбук / системний блок;

$W_2 = (35 * 1200) / 1000 = 42$ кВт·год - монітор 24";

$W_3 = (10 * 2000) / 1000 = 20$ кВт·год - маршрутизер;

$W_4 = (5 * 1200) / 1000 = 6$ кВт·год - периферія.

$W_{\Sigma} = 144 + 42 + 20 + 6 = 212$ кВт·год/рік.

Орієнтовні викиди вуглекислого газу (коефіцієнт 0,38 кг/кВт·год)

$M_{CO_2} = 212 * 0,38 = 81$ кг/рік.

При тарифі побутового споживача ~4 грн/кВт·год вартість складає порядку 950 грн/рік - для одного робочого місця. Порівняно з окремою серверною в приміщенні (постійне охолодження, стійки 500–2000 Вт) хмарна схема переносить основне енергоспоживання на дата-центри провайдерів (Neon EU, Render Frankfurt, CDN Vercel), де вищий ККД через консолідацію навантаження.

Хмарні сервіси. Neon, Render і Vercel декларують використання сучасних дата-центрів з енергоефективним охолодженням; для дипломного проєкту обрано Free-тариф з автоматичним «засинанням» backend на Render після бездіяльності, що знижує безцільне споживання електроенергії порівняно з сервером, увімкненим 24/7 у кімнаті. Недолік - cold start (затримка до 60 с), що є компромісом між екологічністю тарифу та зручністю користувача (компенсується прогрівом API, п. 3.1).

Електронні відходи. Проєкт не вимагає закупівлі нового обладнання; розробка можлива на наявному ноутбуці, що продовжує термін служби техніки та зменшує внесок у утилізацію відходів від електричного та електронного обладнання. При модернізації - передача придатної техніки на повторне використання або утилізація через офіційні пункти прийому.

Викиди парникових газів. Прямі викиди від програмного продукту відсутні; непрямий внесок - виробництво електроенергії для дата-центрів і робочого місця.

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		91

Використання європейського регіону Neon (EU) і Render (Frankfurt) зменшує транзитні втрати та узгоджується з вимогами до розміщення даних європейських користувачів.

Раціональне використання дискового простору. Локальна папка `uploads/` і база даних не повинні накопичувати дублікати великих зображень без потреби - адміністратор періодично перевіряє обсяг (п. 3.2), що знижує потребу в розширенні дисків і повторному виробництві носіїв. Наведено у таблиці 5.3

Таблиця 5.3 – Параметри ергономіки робочого місця розробник

Параметр	Нормативна / рекомендована величина	Фактичне забезпечення на робочому місці
Відстань від очей до екрана	на рівні очей або нижче на 5–10 см	на рівні очей
Кут нахилу екрана	10–20° назад	15°
Висота сидіння	стопи повністю на підлозі	регульоване крісло
Перерва при роботі з дисплеєм	10–15 хв після 2 год безперервної роботи	дотримується
Освітленість на столі (комбіноване освітлення)	300–500 лкс (ДБН В.2.5- 28:2018)	351 лкс (таблиця 5.3)
Температура повітря	18–25 °C	22 °C
Відносна вологість	40–60 %	50 %

5.7. Заходи щодо підвищення безпеки та екологічності (рекомендації)

На основі аналізу розділу 5 для подальшої роботи над Ріхога та аналогічними проєктами рекомендовано:

Робоче місце: регулювання висоти стільця, перерви кожні 45–60 хв, настільна лампа для доведення освітленості до 300–500 лкс, якщо загальне освітлення недостатнє.

Електробезпека: мережевий фільтр, справні кабелі, не перевантажувати розетки; у класі - знати місце вогнегасника.

Експлуатація: не зберігати секрети в Git; оновлювати залежності Maven і npm для усунення вразливостей; резервне копіювання Neon за інструкцією провайдера.

Користувачі: інформування про перерви при тривалій роботі в чаті; етичне спілкування; захист персональних даних.

Екологія: залишати sleep на тестовому середовищі; для промислової експлуатації після диплома - обрати тариф з передбачуваним навантаженням, щоб не тримати зайві інстанси увімкненими; за можливості - власний домен і стабільний API без надмірних повторних збірок (менше хвилин збірки на Vercel).

У розділі 5 розглянуто охорону праці, техніку безпеки та екологічні вимоги у зв'язку з розробкою та експлуатацією вебсайту компанії дизайнерських послуг Ріхога. Наведено нормативну базу (закони України, ДНАОП, ДБН, ДСТУ ISO 9241, правила пожежної та електробезпеки), характеристику небезпечних і шкідливих факторів на етапах програмування, тестування, деплою та роботи користувачів. Виконано розрахунок освітленості робочого місця (з урахуванням додаткового денного світла - відповідність нормам) та оцінку річного енергоспоживання (~238 кВт·год) робочого місця розробника. Показано, що обрана хмарна архітектура (Neon, Render, Vercel) зменшує локальне навантаження на середовище порівняно з власною серверною, а цифровий документообіг замовлень знижує споживання паперу. Рекомендації з ергономіки, електробезпеки, захисту даних і раціонального використання ресурсів узгоджені з матеріалом розділів 2–3 і можуть бути використані при складанні завдання на кваліфікаційну роботу (підпис консультанта з охорони праці) та при захисті диплома.

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		93

ВИСНОВКИ

У кваліфікаційній роботі на тему «Розробка вебсайту компанії дизайнерських послуг „Ріхога“» виконано повний цикл виробничого проєктування програмного продукту: від аналітичного огляду існуючих рішень і формулювання технічного завдання до програмної реалізації, тестування, публікації в мережі Інтернет та розробки експлуатаційної документації. Мета роботи - створення працездатного вебзастосунку для представлення дизайнерських послуг, оформлення замовлень, виставлення чека, підтвердження оплати, спілкування в чаті та адміністрування контенту - досягнута.

Відповідність завданню на кваліфікаційну роботу. Розроблений програмний продукт відповідає вимогам, сформульованим у завданні та технічному завданні (розділ 1.2): реалізовано публічний каталог послуг і категорій; реєстрацію та вхід користувачів із розмежуванням ролей клієнт, дизайнер, адміністратор; бізнес-процес «замовлення - чек - оплата - завершення»; чат із підтримкою WebSocket; відгуки та рейтинг; сповіщення; адміністративну панель керування; українськомовний інтерфейс. Сайт опубліковано за постійною адресою в мережі Інтернет (Vercel, Render, Neon) і перевірено за контрольними сценаріями (таблиця 3.3, розділ 3). Розділ з охорони праці, техніки безпеки та екологічних вимог (розділ 5) узгоджений із темою роботи та містить розрахунки освітленості робочого місця і оцінку енергоспоживання.

Основні результати та показники роботи.

- Проведено аналітичний огляд міжнародних маркетплейсів, вітчизняних бірж фрілансу, конструкторів сайтів і каналів месенджерів; обґрунтовано доцільність власної розробки (п. 1.1, рисунок 1.1).
- Спроектовано структуру вебсайту, базу даних PostgreSQL, архітектуру клієнтської та серверної частин (рисунок 1.2–1.6, таблиця 2.1–2.2, розділ 2).
- Реалізовано односторінковий застосунок на React і Vite, серверний REST API на Spring Boot 3.3 [2] (Java 17), автентифікацію JWT [4], живий чат WebSocket [5], файлове сховище зображень у каталозі uploads,

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
						94
Зм.	Арк.	№ докум.	Підпис	Дата		

підтвердження зміни пароля кодом на електронну пошту (розділ 2.4).

– Проведено функціональне тестування за ролями гостя, клієнта, дизайнера та адміністратора; 15 ключових сценаріїв із таблиця 2.4 виконано успішно; ендпоінт /actuator/health повертає статус UP (розділ 2.5).

– Виконано деплой на хмарні сервіси GitHub, Neon, Render, Vercel; отримано публічну адресу сайту та перевірено крос-доменну взаємодію frontend і backend (п. 3.1, таблиця 3.1–3.3).

– Підготовлено експлуатаційну документацію: інструкції з розміщення, експлуатації адміністративної частини, наповнення контентом і заходів з популяризації (розділ 3).

Самостійне виконання та новизна. Кваліфікаційну роботу виконано самостійно: аналіз предметної області, проектування структури сайту й бази даних, програмування клієнтської та серверної частин, налаштування деплою, тестування, оформлення пояснювальної записки та ілюстративного матеріалу. Консультації наукового керівника та консультанта з охорони праці використано для уточнення структури записки та нормативних розрахунків. У процесі виконання вперше для автора розв’язано комплекс завдань: побудова предметно-орієнтованої моделі «послуга - замовлення - чек - оплата - чат - відгук» у повному стеку React + Spring Boot + PostgreSQL; інтеграція WebSocket-чату з прив’язкою до замовлення; публікація дипломного проєкту на безкоштовному хмарному хостингу з керованою базою даних Neon; реалізація рольової безпеки та підтвердження зміни пароля через електронну пошту.

Публікації та наукова активність. У процесі роботи над дипломним проєктом статей, авторських свідоцтв (заявок), доповідей на конференціях, опублікованих або підготовлених до друку матеріалів не передбачалося. Презентація роботи передбачена на захисті кваліфікаційної роботи; за бажанням студента посилення на демонстраційний сайт може бути розміщено в академічних чатах (п. 3.3).

Напрями подальшої роботи над темою. Подальша розробка Ріхога доцільна

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
						95
Зм.	Арк.	№ докум.	Підпис	Дата		

як розвиток навчального проєкту до комерційної або стартап-платформи. Перспективні напрями: підключення власного доменного імені та платного тарифу Render без режиму sleep; інтеграція реальної платіжної системи (LiqPay, Monobank тощо) замість імітації оплати; постійне файлове сховище (S3, MinIO); розширення автоматизованого тестування (Playwright, Cypress); e-mail-розсилки, аналітика відвідуваності, мобільний застосунок або PWA; локалізація англійською мовою.

Повна зупинка розвитку теми не мотивується: створена архітектура та опублікований продукт дозволяють нарощувати функціонал без зміни загальної структури системи.

Загальний висновок. У результаті виконання кваліфікаційної роботи створено та опубліковано вебсайт компанії дизайнерських послуг Pihora, який відповідає завданню на кваліфікаційну роботу, демонструє застосування сучасних вебтехнологій і може використовуватися як основа реальної вітрини студії дизайну. Отримані результати підтверджують правильність обраного підходу - власна розробка замість готових маркетплейсів і конструкторів - і готовність програмного продукту до демонстрації на захисті.

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		96

ПЕРЕЛІК ПОСИЛАНЬ

1. React Documentation [Електронний ресурс]. – Режим доступу: <https://react.dev>
2. Spring Boot Reference Documentation [Електронний ресурс]. – Режим доступу: <https://docs.spring.io/spring-boot/docs/current/reference/html/>
3. PostgreSQL Documentation [Електронний ресурс]. – Режим доступу: <https://www.postgresql.org/docs/>
4. Jones M., Bradley J. JSON Web Token (JWT) [Електронний ресурс]. – RFC 7519. – Режим доступу: <https://datatracker.ietf.org/doc/html/rfc7519>
5. Fette I., Melnikov A. The WebSocket Protocol [Електронний ресурс]. – RFC 6455. – Режим доступу: <https://datatracker.ietf.org/doc/html/rfc6455>
6. Fielding R. Architectural Styles and the Design of Network-based Software Architectures: Dissertation. – 2000. – (REST).
7. React Router Documentation [Електронний ресурс]. – Режим доступу: <https://reactrouter.com>
8. TypeScript Documentation [Електронний ресурс]. – Режим доступу: <https://www.typescriptlang.org/docs/>
9. Vite Documentation [Електронний ресурс]. – Режим доступу: <https://vitejs.dev>
10. Flyway Documentation [Електронний ресурс]. – Режим доступу: <https://documentation.red-gate.com/fd>
11. TanStack Query Documentation [Електронний ресурс]. – Режим доступу: <https://tanstack.com/query>
12. Axios Documentation [Електронний ресурс]. – Режим доступу: <https://axios-http.com>

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
						97
Зм.	Арк.	№ докум.	Підпис	Дата		

13. STOMP Protocol Specification [Електронний ресурс]. – Режим доступу: <https://stomp.github.io>

14. Spring Security Reference [Електронний ресурс]. – Режим доступу: <https://docs.spring.io/spring-security/reference/>

15. Zustand Documentation [Електронний ресурс]. – Режим доступу: <https://zustand.docs.pmnd.rs>

16. Tailwind CSS Documentation [Електронний ресурс]. – Режим доступу: <https://tailwindcss.com/docs>

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		98

ДОДАТКИ

Додаток А. Лістинг файлу OrderController.java

```
package com.designdev.designservicesapi.controller;

import com.designdev.designservicesapi.dto.FeatureDtos;
import com.designdev.designservicesapi.dto.OrderDtos;
import com.designdev.designservicesapi.service.OrderExtrasService;
import com.designdev.designservicesapi.service.OrderService;
import jakarta.validation.Valid;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.http.MediaType;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.multipart.MultipartFile;

import java.time.LocalDate;
import java.util.List;

@RestController
@RequestMapping("/api/orders")
public class OrderController {

    private final OrderService orderService;
    private final OrderExtrasService orderExtrasService;

    public OrderController(OrderService orderService, OrderExtrasService
orderExtrasService) {
        this.orderService = orderService;
    }
}
```

					2026.KBP.122.423.10.00.00 ПЗ	Арк.
						99
Зм.	Арк.	№ докум.	Підпис	Дата		

```

        this.orderExtrasService = orderExtrasService;
    }

    @PostMapping(value = "", consumes = "application/json")
    @PreAuthorize("isAuthenticated()")
    public OrderDtos.OrderResponse create(@Valid @RequestBody
    OrderDtos.CreateOrderRequest request) {
        Authentication authentication =
        SecurityContextHolder.getContext().getAuthentication();
        String username = authentication.getName();
        return orderService.create(username, request);
    }

    @GetMapping
    @PreAuthorize("isAuthenticated()")
    public Page<OrderDtos.OrderResponse> search(
        @RequestParam(required = false) String status,
        @RequestParam(required = false) LocalDate createdFrom,
        @RequestParam(required = false) LocalDate createdTo,
        Pageable pageable
    ) {
        Authentication authentication =
        SecurityContextHolder.getContext().getAuthentication();
        String username = authentication.getName();

        boolean isAdmin = authentication.getAuthorities().stream()
            .anyMatch(auth ->
auth.getAuthority().equals("ROLE_ADMIN"));
        boolean isDesigner = authentication.getAuthorities().stream()
            .anyMatch(auth ->
auth.getAuthority().equals("ROLE_DESIGNER"));

        Long userId = null;
        Long designerId = null;
    }

```

					2026.KBP.122.423.10.00.00 ПЗ	Арк.
						100
Зм.	Арк.	№ докум.	Підпис	Дата		

```

        if (!isAdmin) {
            Long currentUserId = orderService.getCurrentUserId(username);
            if (isDesigner) {
                designerId = currentUserId;
                userId = currentUserId;
            } else {
                userId = currentUserId;
            }
        }

        OrderDtos.OrderFilter filter = new OrderDtos.OrderFilter(status,
        createdFrom, createdTo, userId, designerId);
        return orderService.search(filter, pageable);
    }

    @PostMapping("/{orderId}/invoice")
    @PreAuthorize("hasAnyRole('ADMIN','DESIGNER')")
    public OrderDtos.OrderResponse issueInvoice(
        @PathVariable Long orderId,
        @Valid @RequestBody OrderDtos.IssueInvoiceRequest request
    ) {
        Authentication authentication =
        SecurityContextHolder.getContext().getAuthentication();
        return orderService.issueInvoice(authentication.getName(), orderId,
        request);
    }

    @PostMapping("/{orderId}/pay")
    @PreAuthorize("isAuthenticated()")
    public OrderDtos.PaymentResponse pay(@PathVariable Long orderId) {
        Authentication authentication =
        SecurityContextHolder.getContext().getAuthentication();
        return orderService.processPayment(authentication.getName(),
        orderId);
    }

```

					2026.KBP.122.423.10.00.00 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		101

```

    }

    @PatchMapping("/{orderId}/status")
    @PreAuthorize("hasAnyRole('ADMIN','DESIGNER')")
    public OrderDtos.OrderResponse updateStatus(@PathVariable Long orderId,
    @RequestParam String status) {
        Authentication authentication =
SecurityContextHolder.getContext().getAuthentication();
        return orderService.updateStatus(orderId, status,
authentication.getName());
    }

    @DeleteMapping("/{orderId}")
    @PreAuthorize("hasAnyRole('ADMIN','DESIGNER')")
    public void delete(@PathVariable Long orderId) {
        Authentication authentication =
SecurityContextHolder.getContext().getAuthentication();
        orderService.delete(orderId, authentication.getName());
    }

    @GetMapping("/{orderId}/history")
    @PreAuthorize("isAuthenticated()")
    public List<FeatureDtos.OrderStatusHistoryResponse>
statusHistory(@PathVariable Long orderId) {
        Authentication authentication =
SecurityContextHolder.getContext().getAuthentication();
        return orderExtrasService.getStatusHistory(orderId,
authentication.getName());
    }

    @GetMapping("/{orderId}/files")
    @PreAuthorize("isAuthenticated()")
    public List<FeatureDtos.OrderFileResponse> listFiles(@PathVariable Long
orderId) {

```

					2026.KBP.122.423.10.00.00 ПЗ	Арк.
						102
Зм.	Арк.	№ докум.	Підпис	Дата		

```

        Authentication authentication =
SecurityContextHolder.getContext().getAuthentication();

        return orderExtrasService.listFiles(orderId,
authentication.getName());
    }

    @PostMapping(value = "/{orderId}/files", consumes =
MediaType.MULTIPART_FORM_DATA_VALUE)
    @PreAuthorize("isAuthenticated()")
    public FeatureDtos.OrderFileResponse uploadFile(
        @PathVariable Long orderId,
        @RequestParam("file") MultipartFile file
    ) {
        Authentication authentication =
SecurityContextHolder.getContext().getAuthentication();

        return orderExtrasService.uploadFile(orderId,
authentication.getName(), file);
    }
}

```

					2026.KBP.122.423.10.00.00 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		103

Додаток Б. Лістинг файлу SecurityConfig.java

```
package com.designdev.designservicesapi.config;

import com.designdev.designservicesapi.security.JwtAuthenticationFilter;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.authentication.AuthenticationProvider;
import
org.springframework.security.authentication.dao.DaoAuthenticationProvider;
import
org.springframework.security.config.annotation.authentication.configuration
.AuthenticationConfiguration;
import
org.springframework.security.config.annotation.method.configuration.EnableM
ethodSecurity;
import
org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.http.SessionCreationPolicy;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.http.HttpMethod;
import org.springframework.security.web.SecurityFilterChain;
import
org.springframework.security.web.authentication.UsernamePasswordAuthenticat
ionFilter;
import org.springframework.web.cors.CorsConfigurationSource;

@Configuration
@EnableMethodSecurity
public class SecurityConfig {
```

					2026.KBP.122.423.10.00.00 ПЗ	Арк.
						104
Зм.	Арк.	№ докум.	Підпис	Дата		

```

private final JwtAuthenticationFilter jwtAuthenticationFilter;
private final UserDetailsService userDetailsService;
private final CorsConfigurationSource corsConfigurationSource;

public SecurityConfig(JwtAuthenticationFilter jwtAuthenticationFilter,
                      UserDetailsService userDetailsService,
                      CorsConfigurationSource corsConfigurationSource)
{
    this.jwtAuthenticationFilter = jwtAuthenticationFilter;
    this.userDetailsService = userDetailsService;
    this.corsConfigurationSource = corsConfigurationSource;
}

@Bean
public SecurityFilterChain securityFilterChain(HttpSecurity http) throws
Exception {
    http.csrf(csrf -> csrf.disable())
        .cors(cors ->
cors.configurationSource(corsConfigurationSource))
        .authorizeHttpRequests(auth -> auth
            .requestMatchers(HttpMethod.GET, "/api/services",
"/api/services/**").permitAll()
            .requestMatchers(HttpMethod.GET, "/api/categories",
"/api/categories/**").permitAll()
            .requestMatchers(HttpMethod.GET,
"/api/portfolio/designer/**").permitAll()
            .requestMatchers(HttpMethod.GET,
"/api/users/public/**").permitAll()
            .requestMatchers(HttpMethod.GET,
"/api/files/**").permitAll()
            .requestMatchers(
                "/api/auth/**",
                "/api/reviews",

```

					2026.KBP.122.423.10.00.00 ПЗ	Арк.
						105
Зм.	Арк.	№ докум.	Підпис	Дата		

```

        "/api/reviews/**",
        "/swagger-ui/**",
        "/swagger-ui.html",
        "/v3/api-docs/**",
        "/swagger-resources/**",
        "/webjars/**",
        "/",
        "/error",
        "/actuator/health",
        "/actuator/info",
        "/uploads/**",
        "/ws/**"
    ).permitAll()
    .requestMatchers("/api/admin/**").hasRole("ADMIN")
        .requestMatchers("/api/orders",
"/api/orders/**").authenticated()
        .requestMatchers("/api/chat/**").authenticated()
        .anyRequest().authenticated()
    )

    .sessionManagement(session ->
session.sessionCreationPolicy(SessionCreationPolicy.STATELESS))
        .authenticationProvider(authenticationProvider())
            .addFilterBefore(jwtAuthenticationFilter,
UsernamePasswordAuthenticationFilter.class);

    return http.build();
}

@Bean
public PasswordEncoder passwordEncoder() {
    return new BCryptPasswordEncoder();
}

@Bean

```

```

    public AuthenticationProvider authenticationProvider() {
        DaoAuthenticationProvider provider = new
        DaoAuthenticationProvider();
        provider.setUserDetailsService(userDetailsService);
        provider.setPasswordEncoder(passwordEncoder());
        return provider;
    }

    @Bean
    public AuthenticationManager
    authenticationManager(AuthenticationConfiguration configuration) throws
    Exception {
        return configuration.getAuthenticationManager();
    }
}

```

					2026.KBP.122.423.10.00.00 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		107

Додаток В. Лістинг файлу App.tsx

```
import { useEffect } from 'react';
import { BrowserRouter, Routes, Route, Navigate } from 'react-router-dom';
import { useAuthStore } from './store/authStore';
import ApiWarmupBanner from './components/ApiWarmupBanner';
import { warmupBackend } from './lib/apiWarmup';
import Layout from './components/Layout';
import HomePage from './pages/HomePage';
import LoginPage from './pages/LoginPage';
import RegisterPage from './pages/RegisterPage';
import ServicesPage from './pages/ServicesPage';
import ServiceDetailPage from './pages/ServiceDetailPage';
import OrdersPage from './pages/OrdersPage';
import ReviewsPage from './pages/ReviewsPage';
import ChatPage from './pages/ChatPage';
import DesignerDashboard from './pages/DesignerDashboard';
import AdminDashboard from './pages/AdminDashboard';
import ProfilePage from './pages/ProfilePage';
import NotificationsPage from './pages/NotificationsPage';
import NotificationSettingsPage from './pages/NotificationSettingsPage';
import SecuritySettingsPage from './pages/SecuritySettingsPage';
import ForgotPasswordPage from './pages/ForgotPasswordPage';
import PaymentsPage from './pages/PaymentsPage';
import FavoritesPage from './pages/FavoritesPage';
import DesignerPublicPage from './pages/DesignerPublicPage';
import ProtectedRoute from './components/ProtectedRoute';

function App() {
  const token = useAuthStore((state) => state.token);
  const isAuth = !!token;

  useEffect(() => {
```

					2026.KBP.122.423.10.00.00 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		108

```

        warmupBackend();
    }, []);

    return (
        <BrowserRouter>
            <ApiWarmupBanner />
            <Routes>
                <Route path="/login" element={!isAuth ? <LoginPage /> : <Navigate
to="/" />} />
                <Route path="/register" element={!isAuth ? <RegisterPage /> :
<Navigate to="/" />} />
                <Route path="/forgot-password" element={!isAuth ?
<ForgotPasswordPage /> : <Navigate to="/" />} />

                <Route element={<Layout />}>
                    <Route path="/" element={<HomePage />} />
                    <Route path="/services" element={<ServicesPage />} />
                    <Route path="/services/:id" element={<ServiceDetailPage />} />
                    <Route path="/designers/:designerId" element={<DesignerPublicPage
/>} />

                    <Route element={<ProtectedRoute />}>
                        <Route path="/orders" element={<OrdersPage />} />
                        <Route path="/reviews" element={<ReviewsPage />} />
                        <Route path="/chat" element={<ChatPage />} />
                        <Route path="/profile" element={<ProfilePage />} />
                        <Route path="/notifications" element={<NotificationsPage />} />
                        <Route path="/notifications/settings"
element={<NotificationSettingsPage />} />
                        <Route path="/security" element={<SecuritySettingsPage />} />
                        <Route path="/payments" element={<PaymentsPage />} />
                        <Route path="/favorites" element={<FavoritesPage />} />
                    </Route>
                </Route>
            </Routes>
        </BrowserRouter>
    );
}

```

```

    <Route element={<ProtectedRoute requiredRole="DESIGNER" />}>
      <Route path="/designer" element={<DesignerDashboard />} />
    </Route>

    <Route element={<ProtectedRoute requiredRole="ADMIN" />}>
      <Route path="/admin" element={<AdminDashboard />} />
    </Route>
  </Route>
</Routes>
</BrowserRouter>
);
}

export default App;

```

					2026.КВР.122.423.10.00.00 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		110

Додаток Г. Лістинг файлу `stompClient.ts`

```
type FrameHandler = (body: string) => void;

interface Subscription {
  id: string;
  destination: string;
  handler: FrameHandler;
}

export class StompClient {
  private ws: WebSocket | null = null;
  private subscriptions = new Map<string, Subscription>();
  private connected = false;
  private onConnectCallback: (() => void) | null = null;

  connect(
    wsUrl: string,
    connectHeaders: Record<string, string>,
    onConnect: () => void,
    onDisconnect?: () => void
  ): void {
    this.onConnectCallback = onConnect;
    this.ws = new WebSocket(wsUrl);

    this.ws.onopen = () => {
      this.sendFrame('CONNECT', {
        'accept-version': '1.2',
        'heart-beat': '0,0',
        ...connectHeaders,
      });
    };
  }
}
```

					2026.KBP.122.423.10.00.00 ПЗ	Арк.
						111
Зм.	Арк.	№ докум.	Підпис	Дата		

```

this.ws.onmessage = (event) => {
    const frames = String(event.data).split('\0').filter(Boolean);
    for (const raw of frames) {
        this.handleFrame(raw);
    }
};

this.ws.onclose = () => {
    this.connected = false;
    onDisconnect?.();
};
}

```

```

subscribe(destination: string, handler: FrameHandler): string {
    const id = `sub-${Math.random().toString(36).slice(2)}`;
    this.subscriptions.set(id, { id, destination, handler });
    if (this.connected) {
        this.sendFrame('SUBSCRIBE', { id, destination });
    }
    return id;
}

```

```

publish(destination: string, body: string): void {
    if (!this.connected) return;
    this.sendFrame('SEND', { destination, 'content-type': 'application/json'
}, body);
}

```

```

disconnect(): void {
    if (this.ws && this.connected) {
        this.sendFrame('DISCONNECT');
    }
    this.ws?.close();
    this.ws = null;
}

```

					2026.KBP.122.423.10.00.00 ПЗ	Арк.
						112
Зм.	Арк.	№ докум.	Підпис	Дата		

```

    this.connected = false;
    this.subscriptions.clear();
}

isConnected(): boolean {
    return this.connected;
}

private handleFrame(raw: string): void {
    const newline = raw.indexOf('\n');
    if (newline === -1) return;

    const command = raw.substring(0, newline);
    const rest = raw.substring(newline + 1);
    const headerEnd = rest.indexOf('\n\n');
    const headerBlock = headerEnd >= 0 ? rest.substring(0, headerEnd) : rest;
    const body = headerEnd >= 0 ? rest.substring(headerEnd + 2) : '';

    const headers: Record<string, string> = {};
    for (const line of headerBlock.split('\n')) {
        const colon = line.indexOf(':');
        if (colon > 0) {
            headers[line.substring(0, colon)] = line.substring(colon + 1);
        }
    }

    if (command === 'CONNECTED') {
        this.connected = true;
        for (const sub of this.subscriptions.values()) {
            this.sendFrame('SUBSCRIBE', { id: sub.id, destination:
sub.destination });
        }
        this.onConnectCallback?.();
        return;
    }
}

```

					2026.KBP.122.423.10.00.00 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		113

```

    }

    if (command === 'MESSAGE') {
        const subId = headers.subscription;
        const sub = subId ? this.subscriptions.get(subId) : undefined;
        sub?.handler(body);
    }
}

private sendFrame(command: string, headers: Record<string, string> = {},
body = ''): void {
    if (!this.ws || this.ws.readyState !== WebSocket.OPEN) return;

    let frame = `${command}\n`;
    for (const [key, value] of Object.entries(headers)) {
        frame += `${key}:${value}\n`;
    }
    frame += `\n${body}\0`;
    this.ws.send(frame);
}
}

```

					2026.KBP.122.423.10.00.00 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		114

Додаток Д. Лістинг файлу useChatWebSocket.ts

```
import { useEffect, useRef, useState } from 'react';
import { StompClient } from '../lib/stompClient';
import { getWsUrl } from '../lib/mediaUrl';

export interface ChatMessageEvent {
  id: number;
  senderId: number;
  senderUsername?: string;
  senderAvatar?: string;
  message: string;
  isRead: boolean;
  createdAt: string;
}

function buildWebSocketUrl(): string {
  const base = getWsUrl();
  const wsBase = base.replace(/^http/, 'ws');
  return `${wsBase}/ws`;
}

export function useChatWebSocket(
  roomId: number | null,
  onMessage: (message: ChatMessageEvent) => void
) {
  const clientRef = useRef<StompClient | null>(null);
  const onMessageRef = useRef(onMessage);
  const [isConnected, setIsConnected] = useState(false);
  onMessageRef.current = onMessage;

  useEffect(() => {
    const token = localStorage.getItem('token');
    if (!roomId || !token) {
      setIsConnected(false);
    }
  });
}
```

```

        return;
    }

    const client = new StompClient();
    clientRef.current = client;
    setIsConnected(false);

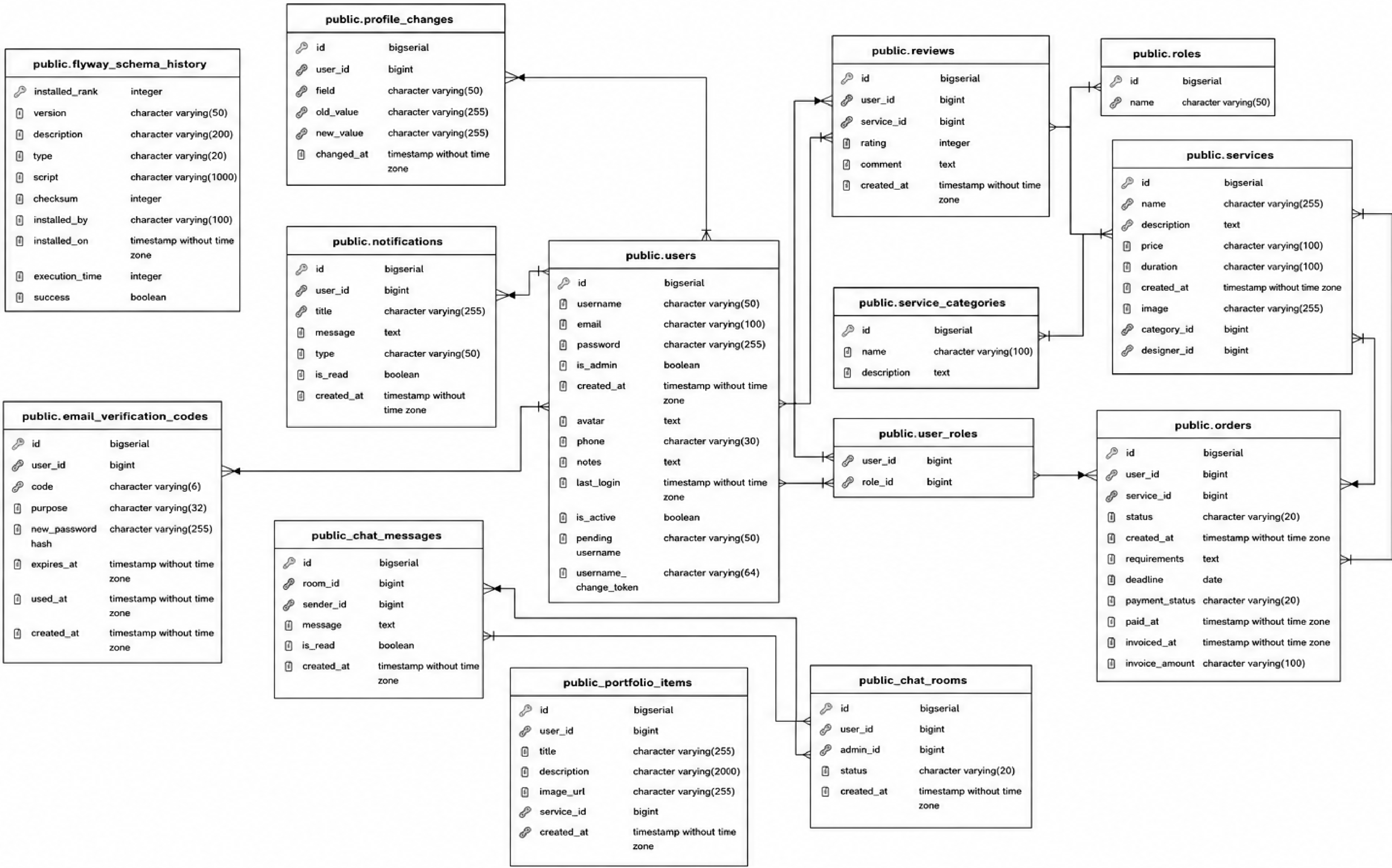
    client.connect(
        buildWebSocketUrl(),
        { Authorization: `Bearer ${token}` },
        () => {
            setIsConnected(true);
            client.subscribe(`/topic/chat/${roomId}`, (body) => {
                const payload = JSON.parse(body) as ChatMessageEvent;
                onMessageRef.current(payload);
            });
        },
        () => setIsConnected(false)
    );

    return () => {
        client.disconnect();
        clientRef.current = null;
        setIsConnected(false);
    };
}, [roomId]);

const sendMessage = (payload: { roomId: number; senderId: number; message:
string }) => {
    const client = clientRef.current;
    if (!client?.isConnected()) return false;
    client.publish('/app/chat.send', JSON.stringify(payload));
    return true;
};

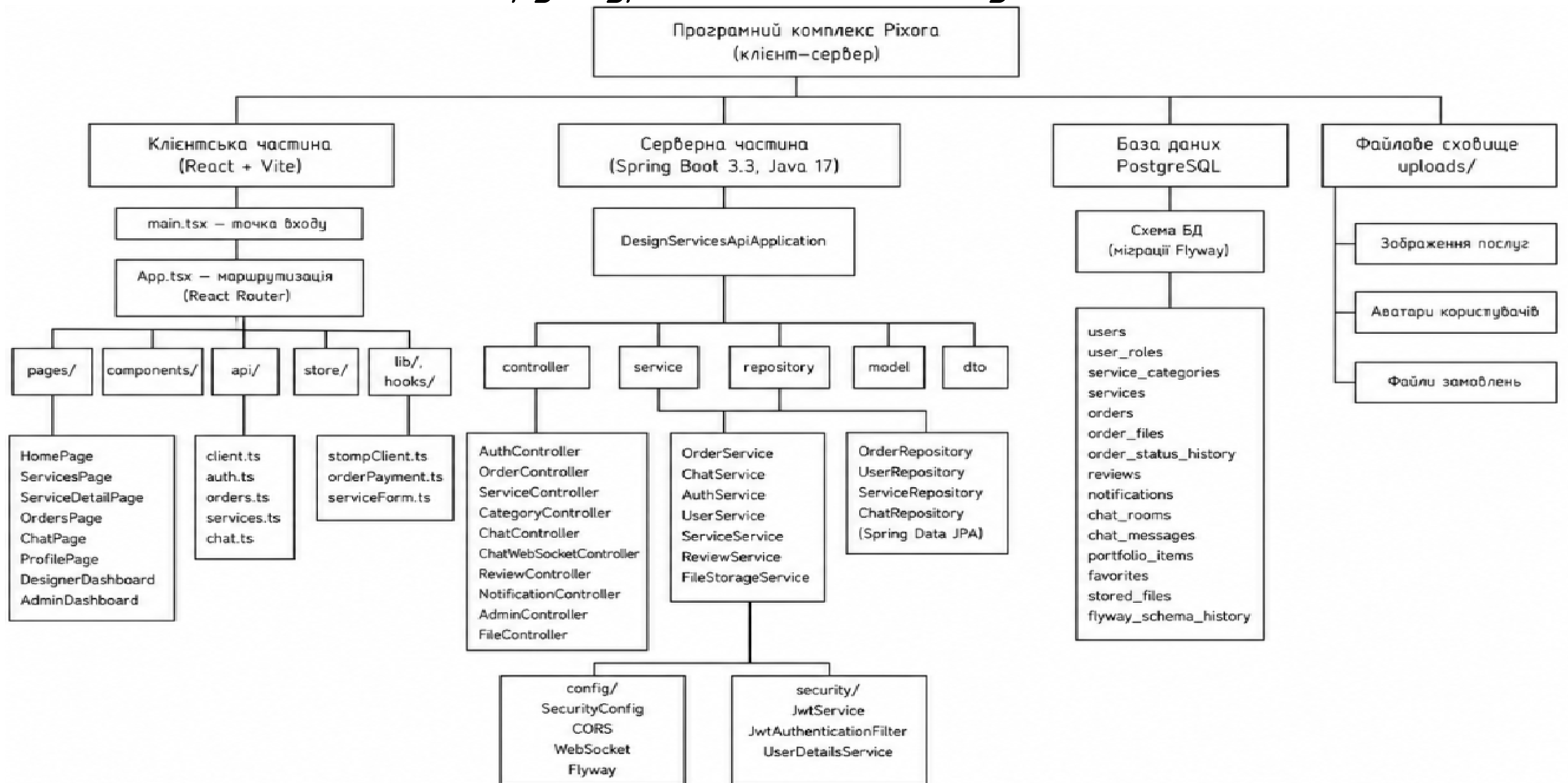
```

ER-діаграма бази даних

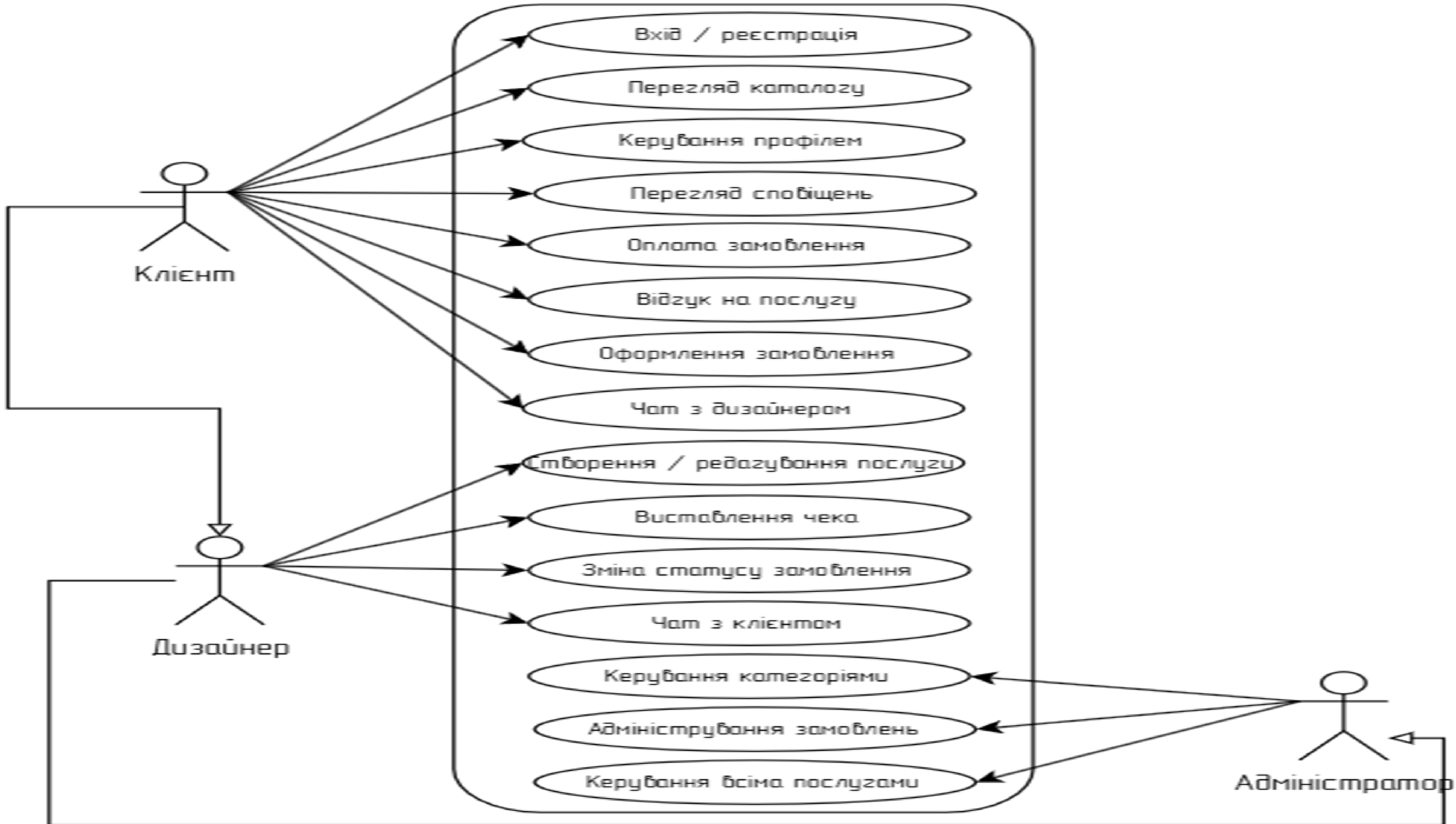


						2026.КВР.122.4.23.10.00.00-Т5			
Зм.	Арк.	№ док.	Підпис	Дата	Розробка вебсайту компанії дизайнерський послуг «Ріхога»		Літ	Маса	Масштаб
Розроб.		Кізяга ГМ			Таблиця техніко-економічних показників				
Перев.		Лисовий ВМ							
Г.контр.							Арк.	Арк.збів	1
Рецензент							ВП ТФК ТНТУ КН-4.23		
Н.контр.		Ліришак В.А.					м. Тернопіль		
Затв.									

Структурна схема вебсайту Ріхора,



Діаграма варіантів використання



Авторська довідка

(кваліфікаційної роботи фахового молодшого бакалавра)

Назва кваліфікаційної роботи фахового молодшого бакалавра: розробка вебсайту компанії дизайнерських послуг «Pixora»

Назва (англ.): Development of website "Pixora" design services company

Освітньо-професійний ступінь : фаховий молодший бакалавр

Шифр та назва спеціальності: 122 Комп'ютерні науки

Екзаменаційна комісія: Екзаменаційна комісія № 2

Установа захисту: ВСП «Тернопільський фаховий коледж Тернопільського національного технічного університету імені Івана Пулюя»

Дата захисту: 23.06.2026

Місто: Тернопіль

Сторінки:

Кількість сторінок роботи: 116

УДК: 004.4

Автор роботи

Прізвище, ім'я, по батькові (укр.): Кузьма Роман Миколайович

Прізвище, ім'я (англ.): Kuzma Roman

Місце навчання (установа, факультет, місто, країна): ВСП «Тернопільський фаховий коледж Тернопільського національного технічного університету імені Івана Пулюя»

Керівник

Прізвище, ім'я, по батькові (укр.): Лісовий Володимир Миколайович

Прізвище, ім'я (англ.): Lisovyy Volodymyr

Місце праці (установа, підрозділ, місто, країна): ВСП «Тернопільський фаховий коледж Тернопільського національного технічного університету імені Івана Пулюя»

Вчене звання, науковий ступінь, посада: категорія «вищий спеціаліст», викладач комп'ютерних дисциплін

Рецензент

Прізвище, ім'я, по батькові (укр.): Млинко Богдана Богданівна

Прізвище, ім'я (англ.): Mlynko Bohdana

Місце праці (установа, підрозділ, місто, країна): Тернопільський національний технічний університет імені Івана Пулюя

Вчене звання, науковий ступінь, посада: кандидат технічних наук, доцент, доцент кафедри комп'ютерних наук

Бібліографічний опис :

Кузьма Р.М. Інформаційні технології керування міським середовищем: кваліфікаційна робота на здобуття освітньо-професійного ступеня фаховий молодший бакалавр за спеціальністю «122 – Комп'ютерні науки». Тернопіль: ВСП «ТФК ТНТУ», 2026. 116 с.

Ключові слова

українською:

вебсайт, дизайнерські послуги, клієнт-серверна інформаційна система, React, Spring Boot, PostgreSQL, REST API, JWT-автентифікація, WebSocket-чат, реляційна база даних

англійською: *website, design services, client-server information system, React, Spring Boot, PostgreSQL, REST API, JWT authentication, WebSocket chat, relational database*

Анотація

українською: Тема кваліфікаційної роботи: Розробка вебсайту компанії дизайнерських послуг «Pixora».

Метою кваліфікаційної роботи є розробка full-stack web-застосунку для каталогу дизайнерських послуг, оформлення та супроводу замовлень, виставлення чека на оплату, особистого кабінета клієнта та дизайнера, чату в режимі реального часу, відгуків, сповіщень і адміністративної панелі.

Пояснювальна записка складається з 5 розділів.

У загальній частині описуються аналітичний огляд існуючих рішень та аналіз технічного завдання на вебсайт компанії дизайнерських послуг «Pixora», сформованого за результатами порівняння платформ Upwork, Fiverr і Freelancehunt.

У другому розділі представлено процес створення програмного продукту, опис та обґрунтування вибору структури та методу організації вхідних та вихідних даних, опис алгоритмів, інформаційних зв'язків, зовнішнє проектування програми, тестування та налагодження програмного забезпечення вебсайту компанії дизайнерських послуг «Pixora».

У спеціальній частині описані процес розміщення програмного продукту в мережі Інтернет, інструкція з обслуговування та наповнення сайту, а також інструкція з популяризації та підтримки застосунку при локальному запуску та хмарному хостингу (Vercel, Render, Neon).

Розрахунок вартості розробки та економічної ефективності приведено в економічній частині четвертого розділу.

Основні питання охорони праці та техніки безпеки розглянуто в п'ятому розділі записки.

Обсяг пояснювальної записки 116 сторінок.

До складу кваліфікаційної роботи входить графічна частина, яка складається з структурної схеми програми, діаграми варіантів використання, техніко-економічних показників, ER-діаграми бази даних, що виконані на окремих аркушах формату A1.

англійською: Topic of the qualification work: Website development for the design services company "Pixora".

The purpose of the qualification work is to develop a full-stack web application for a catalog of design services, order processing and support, issuing a check for payment, a personal account for the client and designer, real-time chat, reviews, notifications and an administrative panel.

The explanatory note consists of 5 sections.

The general part describes an analytical review of existing solutions and an analysis of the technical specifications for the website of the design services company "Pixora", formed based on the results of a comparison of the Upwork, Fiverr and Freelancehunt platforms.

The second section presents the process of creating a software product, a description and justification of the choice of the structure and method of organizing input and output data, a description of algorithms, information connections, external design of the program, testing and debugging of the software of the website of the design services company "Pixora".

The special part describes the process of placing the software product on the Internet, instructions for maintaining and filling the site, as well as instructions for popularizing and supporting the application during local launch and cloud hosting (Vercel, Render, Neon).

The calculation of the cost of development and economic efficiency is given in the economic part of the fourth section.

The main issues of labor protection and safety are considered in the fifth section of the note.

The volume of the explanatory note is 116 pages.

The qualification work includes a graphic part, which consists of a structural diagram of the program, a diagram of use cases, technical and economic indicators, an ER-diagram of the database, which are made on separate sheets of A1 format.

Зміст:

АНОТАЦІЯ

ВСТУП

1 Загальний розділ

1.1. Аналітичний огляд існуючих рішень

1.2. Технічне завдання

1.2.1. Найменування та область застосування

1.2.2. Призначення розробки

1.2.3. Вимоги до функціоналу web-сайту

1.2.4. Вимоги до програмної документації

1.2.5. Техніко-економічні показники

1.2.6. Стадії та етапи розробки

1.2.7. Порядок тестування та прийому

2 Розробка технічного та робочого проєкту

2.2. Створення та верстка сторінок сайту

2.3. Розробка структури бази даних сайту

2.4. Програмування сайту

2.4.1. Написання клієнтської частини

2.4.2. Написання admin-частини

2.5. Тестування web-сайту

3 Спеціальний розділ

3.1. Інструкція з розміщення сайту в Інтернеті

3.2. Інструкція з обслуговування та наповнення сайту

3.3. Інструкція з популяризації та підтримки сайту

4 Економічний розділ

4.1. Визначення стадій технологічного процесу та загальної тривалості проведення НДР

4.2. Визначення витрат на оплату праці та відрахувань на соціальні заходи

4.3. Розрахунок витрат на електроенергію

4.4. Розрахунок суми амортизаційних відрахувань розробки вебсайту компанії дизайнерських послуг «Ріхора»

4.5. Обчислення накладних витрат

4.6. Складання кошторису витрат та визначення собівартості

4.7. Розрахунок ціни розробки вебсайту компанії дизайнерських послуг «Ріхора»

4.8. Визначення економічної ефективності і терміну окупності капітальних вкладень

5 Охорона праці, техніка безпеки та екологічні вимоги

5.1. Нормативно-правова база та організація безпечної праці

5.2. Характеристика робочого процесу та виробничих факторів

5.3. Ергономіка та охорона здоров'я при роботі з персональним комп'ютером

5.4. Техніка безпеки при експлуатації обчислювальної техніки

5.5. Охорона праці та безпека користувачів вебсайту Ріхора

5.6. Екологічні вимоги та раціональне використання ресурсів

5.7. Заходи щодо підвищення безпеки та екологічності (рекомендації)

ВИСНОВКИ

ПЕРЕЛІК ПОСИЛАНЬ

ДОДАТКИ

Додаток А. Лістинг файлу OrderController.java

Додаток Б. Лістинг файлу SecurityConfig.java

Додаток В. Лістинг файлу App.tsx

Додаток Г. Лістинг файлу stompClient.ts

Додаток Д. Лістинг файлу useChatWebSocket.ts

Список літератури:

- React Documentation [Електронний ресурс]. – Режим доступу: <https://react.dev>
- Spring Boot Reference Documentation [Електронний ресурс]. – Режим доступу: <https://docs.spring.io/spring-boot/docs/current/reference/html/>
- PostgreSQL Documentation [Електронний ресурс]. – Режим доступу: <https://www.postgresql.org/docs/>
- Jones M., Bradley J. JSON Web Token (JWT) [Електронний ресурс]. – RFC 7519. – Режим доступу: <https://datatracker.ietf.org/doc/html/rfc7519>
- Fette I., Melnikov A. The WebSocket Protocol [Електронний ресурс]. – RFC 6455. – Режим доступу: <https://datatracker.ietf.org/doc/html/rfc6455>
- Fielding R. Architectural Styles and the Design of Network-based Software Architectures: Dissertation. – 2000. – (REST).
- React Router Documentation [Електронний ресурс]. – Режим доступу: <https://reactrouter.com>
- TypeScript Documentation [Електронний ресурс]. – Режим доступу: <https://www.typescriptlang.org/docs/>
- Vite Documentation [Електронний ресурс]. – Режим доступу: <https://vitejs.dev>
- Flyway Documentation [Електронний ресурс]. – Режим доступу: <https://documentation.red-gate.com/fd>
- TanStack Query Documentation [Електронний ресурс]. – Режим доступу: <https://tanstack.com/query>
- Axios Documentation [Електронний ресурс]. – Режим доступу: <https://axios-http.com>
- STOMP Protocol Specification [Електронний ресурс]. – Режим доступу: <https://stomp.github.io>
- Spring Security Reference [Електронний ресурс]. – Режим доступу: <https://docs.spring.io/spring-security/reference/>
- Zustand Documentation [Електронний ресурс]. – Режим доступу: <https://zustand.docs.pmnd.rs>
- Tailwind CSS Documentation [Електронний ресурс]. – Режим доступу: <https://tailwindcss.com/docs>