КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра технологій управління

Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Управління проектами»

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему:

«Дослідження технологій управління проєктом створення мобільного додатку для управління клієнтами салону б'юті послуг»

Студентки 2-го курсу групи УПз-21

Софії ПХАЙКО
(Ім'я, ПРІЗВИЩЕ)

_____ _____ _____
(підпис студента) *(дата)* *(підпис)*

Науковий керівник:

к.т.н., професор
(науковий ступінь, вчене звання)

Віктор МОРОЗОВ
(Ім'я, ПРІЗВИЩЕ)

Попередній захист:

_____
(Висновок: «До захисту в Екзаменаційній комісії»)

Завідувач кафедри
технологій управління _____ Віктор МОРОЗОВ _____
(підпис) *(Ім'я, ПРІЗВИЩЕ)* *(дата)*

Київ – 2026

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій

Кафедра технологій управління
Освітній рівень Магістр
Спеціальність 122 Комп'ютерні науки
Освітньо-професійна програма Управління проектами

ЗАТВЕРДЖУЮ
Завідувач кафедри
професор Віктор МОРОЗОВ

_____
«29» вересня 2025 року

ЗАВДАННЯ
НА ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Студентка: Пхайко Софія Дмитрівна
Група: УПз-21

1. Тема кваліфікаційної роботи: «Дослідження технологій управління проєктом створення мобільного додатку для управління клієнтами салону б'юті послуг»

Затверджена Протоколом № 15 від «16» червня 2025 року.

2. Строк подання студентом готової роботи – «21» травня 2026 року

3. Цільова установка та вихідні дані до роботи: Метою кваліфікаційної роботи є дослідження технологій управління проєктом створення мобільного додатку для управління клієнтами салону б'юті послуг. Вихідними даними є: результати аналізу ринку б'юті-послуг України, порівняльний аналіз існуючих CRM-систем (Fresha, Booksy, YCLIENTS), стандарти управління проєктами (PMI PMBOK, Agile Manifesto, Scrum Guide), стандарт якості програмного забезпечення ISO/IEC 25010, а також наукові праці у галузі управління IT-проєктами, проєктування інформаційних систем і мобільної розробки.

4. Зміст роботи: Розділ 1. Аналіз предметної області та обґрунтування проєкту: дослідження бізнес-процесів салону б'юті послуг; порівняльний аналіз

існуючих CRM-рішень; виявлення проблем управління клієнтами; формування концепції мобільного додатку; визначення зацікавлених сторін; розробка паспорту проєкту. Розділ 2. Моделювання системи та формування вимог: постановка задачі проєкту; аналіз користувачів і ролей системи; визначення функціональних і нефункціональних вимог; моделювання сценаріїв використання (Use Case); побудова концептуальної моделі інформаційної системи. Розділ 3. Планування та управління проєктом розробки: вибір методології управління проєктом (Scrum); імплементація фреймворку Scrum, формування Product Backlog і User Stories; організаційна структура команди; декомпозиція робіт (WBS); календарний план і діаграма Ганта; управління ризиками (реєстр із 30 ризиків); бюджетний план; ключові показники ефективності (KPI). Розділ 4. Розробка інформаційного та програмного забезпечення: концептуальна архітектура мобільного додатку; проєктування структури бази даних; розробка функціональних модулів; проєктування інтерфейсу користувача (UI/UX); інтеграція зовнішніх сервісів та ШІ-асистента адміністратора.

5. Перелік графічного матеріалу: Рис. 1.1 — Дерево проблем проєкту, Рис. 1.2 — Дерево цілей проєкту, Рис. 2.1 — Діаграма варіантів використання (Use Case), Рис. 2.2 — Концептуальна модель інформаційної системи, Рис. 3.1 — Схема організаційної структури команди проєкту, Рис. 3.2 — WBS проєкту (ієрархічна структура робіт), Рис. 3.3 — Діаграма Ганта , Рис. 4.1 — Концептуальна архітектура мобільного CRM-додатку, Рис. 4.2 — ER-діаграма бази даних, Рис. 4.3 — Клієнтська частина, Рис. 4.4 — Продовженя клієнтської частини , Рис. 4.5 — Адміністративна частина

6. Календарний план виконання роботи

№ з/п	Назва частин роботи	Виконання роботи
1	Вивчення літературних джерел з предмету дослідження	01.10.25–07.10.25

2	Збір і вивчення матеріалів	08.10.25–15.10.25
3	Складання розгорнутого плану кваліфікаційної роботи	16.10.25–20.10.25
4	Ознайомлення наукового керівника з розгорнутим планом кваліфікаційної роботи. Внесення змін	21.10.25–22.10.25
5	Підготовка розділу 1	23.10.25–06.11.25
6	Підготовка розділу 2	07.11.25–14.11.25
7	Підготовка розділу 3	15.11.25–23.11.25
8	Підготовка розділу 4	24.11.25–05.12.25
9	Оформлення кваліфікаційної роботи	28.11.25–05.12.25
10	Передача кваліфікаційної роботи науковому керівникові	05.12.25
11	Попередній захист кваліфікаційної роботи	04.05.26–11.05.26
12	Передача кваліфікаційної роботи рецензенту для рецензування	14.05.26
13	Захист кваліфікаційної роботи	25.05.25-26.05.26

Дата видачі завдання «29» вересня 2025 р.

Керівник роботи к.т.н., професор Віктор МОРОЗОВ _____
(підпис)

Завдання прийняла до виконання
студентка групи УПз-21 Софія ПХАЙКО

_____
(підпис)

АНОТАЦІЯ

кваліфікаційної роботи магістра на тему:

«Дослідження технологій управління проєктом створення мобільного додатку для управління клієнтами салону б'юті послуг»

Студентка: Пхайко Софія Дмитрівна

Науковий керівник: Морозов Віктор Володимирович

Рік захисту – 2026.

Метою кваліфікаційної роботи є дослідження технологій управління проєктом створення мобільного додатку для управління клієнтами салону б'юті послуг та розробка концепції програмного продукту, який автоматизує ключові бізнес-процеси підприємства.

Ціль проєкту — створення мобільного CRM-додатку, що автоматизує управління клієнтами, розкладом майстрів і комунікацією у б'юті-салоні шляхом застосування сучасних методологій управління ІТ-проєктом.

Практична цінність роботи полягає в тому, що розроблена концепція мобільного додатку, план управління проєктом на 52 тижні (24 спринти), реєстр ризиків із 30 позицій і проєктна документація можуть бути безпосередньо використані для реалізації реального програмного продукту власниками салонів краси та менеджерами ІТ-проєктів.

Кваліфікаційна робота складається з 4 розділів, висновків, списку використаних джерел (52 позиції) та 4 додатків. Загальний обсяг роботи — 129 сторінок.

Перший розділ присвячено аналізу предметної області: досліджено бізнес-процеси салону б'юті послуг, проведено порівняльний аналіз існуючих CRM-систем (Fresha, Booksy, YCLIENTS), виявлено ключові проблеми управління клієнтами, сформовано концепцію мобільного додатку, визначено зацікавлені сторони і розроблено паспорт проєкту.

У другому розділі здійснено моделювання системи та формування вимог: визначено ролі користувачів і принцип RBAC, сформовано 12 функціональних і

7 нефункціональних вимог із пріоритизацією MoSCoW, побудовано Use Case-модель і концептуальну архітектуру інформаційної системи.

Третій розділ містить повний план управління проєктом: обґрунтовано вибір методології Scrum, сформовано Product Backlog із 16 User Stories і критеріями приймання, розроблено WBS із декомпозицією на 6 фаз, побудовано календарний план на 24 спринти, реєстр ризиків із 30 позицій і триступеневими протиризиковими заходами, бюджетний план на 2 569 270 грн і 9 KPI проєкту.

Четвертий розділ присвячено розробці програмного забезпечення: спроєктовано трирівневу архітектуру (React Native + Node.js + PostgreSQL), базу даних із 8 сутностей, 8 функціональних модулів, інтерфейс користувача для клієнтської і адміністративної частин, а також інтеграцію 5 зовнішніх сервісів, включаючи ШІ-асистента адміністратора на основі LLM API.

Ключові слова: управління проєктом, мобільний додаток, CRM-система, б'юті-послуги, Scrum, Agile, React Native, ШІ-асистент, омніканальна комунікація, управління ризиками, WBS, Product Backlog, User Story.

ЗМІСТ

ВСТУП

Б'юті-індустрія є одним із найстійкіших сегментів українського ринку послуг: за даними Державної служби статистики України, обсяг ринку послуг краси зріс на 15% у 2023 році порівняно з попереднім роком [20]. Станом на кінець 2024 року на платформі Barb.ua зареєстровано понад 12 000 салонів та майже 35 000 майстрів краси [36], що свідчить про масштаб і розвиненість галузі. При цьому значна частина цих підприємств, особливо малі та середні салони, досі покладається на ручні або напівцифрові підходи до роботи з клієнтами: паперові журнали, переписку у месенджерах, електронні таблиці. Така система перестає справлятися, щойно кількість записів зростає: з'являються накладки у розкладі, пропущені звернення, а адміністратор більше не може тримати всі процеси під контролем вручну [33].

Вирішенням цих проблем є спеціалізовані мобільні CRM-системи для автоматизації запису клієнтів, управління розкладом та ведення клієнтської бази. Потреба в таких рішеннях підтверджується розвитком вітчизняного ринку програмного забезпечення для б'юті-бізнесу: станом на 2024 рік в Україні працюють щонайменше 7 українських CRM для салонів краси [32], однак більшість із них або орієнтовані на великі мережі, або не задовольняють потреб малих підприємств у простоті та доступності. Дослідження 2023 року показало, що компанії, які впровадили системи онлайн-запису, фіксують збільшення доходу на 20% у перший рік використання [30].

Проте ефективне створення такого програмного продукту неможливе без застосування сучасних технологій управління IT-проєктом: структурованого планування робіт, формування команди, управління ризиками та визначення показників ефективності. Саме поєднання цих двох складових — проєктування CRM-рішення та управління проєктом його розробки — визначає предмет і спрямованість цієї роботи.

Метою роботи є дослідження технологій управління проєктом створення мобільного додатку для управління клієнтами салону б'юті послуг та розробка

концепції програмного продукту, який автоматизує ключові бізнес-процеси підприємства.

Об'єктом дослідження є процеси управління IT-проєктами зі створення мобільних програмних продуктів для автоматизації сервісного бізнесу.

Предметом дослідження є методи, моделі та інструменти управління проєктом створення мобільного CRM-додатку, включаючи процеси планування, організації команди, управління ризиками та оцінки ефективності.

Для досягнення поставленої мети вирішуються такі завдання:

- проаналізувати особливості бізнес-процесів салону б'юті послуг та виявити ключові проблеми управління клієнтами;
- дослідити існуючі CRM-рішення для б'юті-індустрії та обґрунтувати доцільність власної розробки;
- визначити зацікавлені сторони та сформувати паспорт проєкту;
- розробити функціональні та нефункціональні вимоги, Use Case-модель і структуру бази даних системи;
- розробити концептуальну архітектуру мобільного додатку та обґрунтувати вибір технічного стеку;
- сформувати організаційну структуру команди проєкту та побудувати WBS;
- розробити календарний план і діаграму Ганта для проєкту розробки;
- скласти реєстр ризиків із стратегіями їх мінімізації та визначити KPI проєкту.

У процесі виконання кваліфікаційної роботи вирішено всі поставлені завдання: проаналізовано особливості бізнес-процесів салону б'юті послуг і виявлено ключові проблеми управління клієнтами; досліджено існуючі CRM-рішення для б'юті-індустрії та обґрунтовано доцільність власної розробки; визначено зацікавлені сторони та сформовано паспорт проєкту; розроблено функціональні та нефункціональні вимоги, Use Case-модель і структуру бази даних системи; спроєктовано концептуальну архітектуру мобільного додатку;

сформовано організаційну структуру команди проєкту та побудовано WBS; розроблено календарний план і діаграму Ганта на 24 спринти; складено реєстр ризиків із 30 позицій та визначено 9 KPI проєкту.

Практичне значення результатів роботи полягає в тому, що розроблена концепція мобільного CRM-додатку та план управління проєктом можуть бути використані для реалізації реального програмного продукту. Власники салонів краси отримують обґрунтовану основу для ухвалення рішення про автоматизацію, а менеджери проєктів — структурований приклад планування ІТ-проєкту у сфері б'юті-послуг.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОБҐРУНТУВАННЯ ПРОЄКТУ

1.1 Особливості бізнес-процесів салону б’юті послуг

Салон б'юті послуг є підприємством сервісної економіки, де конкурентоспроможність визначається не лише якістю виконання процедур, а й тим, наскільки злагоджено організована внутрішня операційна діяльність. Клієнт оцінює салон у сукупності: зручність запису, точність дотримання часу, якість обслуговування та комунікація після візиту формують цілісний досвід, який безпосередньо впливає на повторні відвідування та рекомендації. Laudon та Laudon зазначають, що ефективне управління бізнес-процесами в сервісних організаціях неможливе без інформаційних систем, здатних автоматизувати рутинні операції та мінімізувати вплив людського фактора [1]. Саме тому розуміння структури і логіки бізнес-процесів салону є відправною точкою для проєктування будь-якого цифрового рішення в цій сфері.

Типовий салон краси має відносно просту, але функціонально насичену організаційну структуру. Власник або керівник відповідає за стратегічний розвиток, фінансову політику та маркетинг і приймає ключові рішення щодо асортименту послуг, цінової політики та розвитку персоналу. Адміністратор є центральною операційною фігурою: забезпечує комунікацію з клієнтами, веде записи, координує роботу майстрів і вирішує поточні організаційні питання. Майстри безпосередньо надають послуги та є носіями основної цінності для клієнта. За потреби залучається допоміжний персонал для підтримки основних процесів. Kotler і Keller підкреслюють, що в сервісному бізнесі орієнтація на клієнта та узгодженість внутрішніх процесів є взаємозалежними факторами успіху [2]. У контексті салону це означає, що організаційна структура повинна бути побудована так, щоб кожен учасник процесу міг ефективно виконувати свою роль без надлишкових узгоджень і ручної координації.

Запис клієнтів є вхідною точкою для всіх інших процесів салону і безпосередньо визначає завантаженість майстрів та фінансовий результат

підприємства. Він включає кілька послідовних кроків: вибір послуги та майстра, узгодження зручної дати й часу, підтвердження запису та надсилання нагадування напередодні візиту. Традиційна модель, за якої всі ці кроки виконує адміністратор вручну через телефон або месенджери, має кілька системних вад.

По-перше, вона обмежена робочим часом персоналу: клієнт не може самостійно записатися у вечірній час або вихідний день.

По-друге, людський фактор неминуче породжує помилки — неправильно зафіксований час, накладання записів або пропущене повідомлення.

По-третє, такий підхід не дозволяє клієнту самостійно оцінити доступність слотів і швидко прийняти рішення без очікування відповіді персоналу. Chaffey зазначає, що цифрові канали взаємодії з клієнтами суттєво підвищують ефективність бізнесу та знижують операційне навантаження на персонал [3]. За даними досліджень ринку, впровадження систем онлайн-запису дозволяє збільшити дохід салону на 20% вже у перший рік використання [30].

Формування ефективного розкладу майстрів є однією з найскладніших операційних задач у салоні краси. Вона потребує одночасного врахування тривалості конкретних процедур, робочого часу кожного майстра, перерв, вихідних, а також змін і скасувань з боку клієнтів. За відсутності автоматизованого інструменту адміністратор змушений утримувати весь цей масив інформації у фрагментованих записах, що неминуче призводить до помилок. Наслідком є або простій майстра через порожній слот, або перевантаження через накладання записів — обидва сценарії негативно впливають як на якість обслуговування, так і на фінансові показники підприємства. Автоматизовані системи планування вирішують цю проблему, відображаючи доступні часові слоти в реальному часі, запобігаючи конфліктам у розкладі та дозволяючи швидко переносити або скасовувати записи. Крім того, такі системи дають змогу аналізувати пікові й слабкі періоди завантаженості, що є корисним для обґрунтованого планування роботи персоналу [8].

Систематичний облік послуг і оплат є основою фінансової прозорості підприємства. Він включає фіксацію наданих процедур та їх вартості, форми розрахунку і формування зведеної звітності. Ця інформація дозволяє власнику оцінювати реальну прибутковість бізнесу, виявляти найбільш затребувані послуги та обґрунтовано коригувати цінову політику. Ведення клієнтської бази з повною історією відвідувань додає до цього аналітичного потенціалу вимір персоналізації: знаючи, які процедури і в якій послідовності отримував конкретний клієнт, майстер може запропонувати доречні послуги та уникнути несумісних процедур. Laudon та Laudon підкреслюють, що аналітика клієнтських даних дозволяє компаніям швидше адаптуватися до змін попиту та ухвалювати більш обґрунтовані управлінські рішення [1].

Взаємодія між салоном і клієнтом не обмежується моментом надання послуги. Підтвердження запису, нагадування про візит, інформування про акції та збір зворотного зв'язку після відвідування є невід'ємними елементами якісного сервісу. За відсутності централізованої системи ця комунікація здійснюється нерегулярно і залежить від ініціативи окремих співробітників, що знижує її ефективність і створює ризик втрати клієнта. Автоматизація нагадувань зменшує кількість пропущених записів і водночас формує у клієнта відчуття структурованого та уважного підходу з боку салону [3].

Усі описані процеси є не ізольованими, а взаємозалежними. Якість запису клієнта безпосередньо визначає точність розкладу, розклад — завантаженість і доходи майстрів, а повнота клієнтської бази — можливості аналітики, на основі якої власник приймає управлінські рішення. Саме тому ефективне управління салоном вимагає не набору окремих інструментів, а інтегрованої системи, що об'єднує всі процеси в єдиному середовищі. За даними вітчизняного ринку, станом на 2024 рік понад 12 000 салонів зареєстровано лише на одній платформі Barb.ua [36], однак більшість малих підприємств досі не використовують спеціалізованих цифрових рішень для управління своєю діяльністю [33]. Це підтверджує як масштаб проблеми, так і практичну затребуваність мобільних CRM-рішень для б'юті-індустрії.

Таким чином, бізнес-процеси салону б'юті послуг охоплюють організацію роботи персоналу, управління записами клієнтів, планування розкладу, ведення фінансового обліку та комунікацію з відвідувачами. Їхня ефективна реалізація є безпосередньою передумовою конкурентоспроможності підприємства, а впровадження інтегрованого мобільного рішення — обґрунтованим і практично необхідним кроком у напрямку цифровізації управління.

1.2 Аналіз існуючих CRM та мобільних додатків для б’юті індустрії

Аналіз існуючих програмних рішень для управління салоном краси є необхідним етапом обґрунтування власної розробки, оскільки дозволяє визначити, які потреби ринку вже задоволені, а які залишаються без відповіді. Розгляд конкурентних продуктів доцільно проводити не лише з точки зору функціонального наповнення, а й з урахуванням реального контексту використання: масштабу бізнесу, для якого орієнтована система, зручності впровадження, вартості та ступеня залежності від зовнішньої платформи. Саме ці параметри у підсумку визначають, наскільки рішення є прийнятним для конкретного підприємства. Ctair і Reynolds зазначають, що ефективність інформаційної системи оцінюється не лише за переліком її функцій, а й за тим, наскільки вона відповідає реальним потребам і можливостям організації, що її використовує [8].

Серед найбільш поширених на міжнародному ринку платформ для б'юті-індустрії виділяються Fresha, Booksy та YCLIENTS [4–6]. Кожна з них реалізує ідею цифровізації б'юті-бізнесу по-різному, що зумовлює характерні переваги та обмеження кожного рішення. Разом із тим усі три системи вирішують спільне коло завдань: усунення ручного запису клієнтів, централізація клієнтських даних, автоматизація планування та забезпечення комунікації між салоном і відвідувачем. Саме ці завдання є ключовими проблемами галузі, на вирішення яких спрямована більшість сучасних CRM-продуктів для сфери послуг [7].

Fresha є прикладом найбільш комплексного підходу до автоматизації салону. Платформа об'єднує онлайн-запис, управління розкладом, клієнтську

базу, фінансову аналітику, маркетингові інструменти та онлайн-оплату в межах єдиного середовища [4]. Базовий функціонал є безкоштовним, однак транзакційні операції, зокрема онлайн-платежі, передбачають комісію з кожного розрахунку. Це робить модель монетизації Fresha фінансово непрозорою для малого бізнесу, де обсяг транзакцій може суттєво вплинути на реальну вартість використання. З іншого боку, комплексність платформи є безперечною перевагою для великого салону або мережі: власник отримує єдиний інструмент для управління операційною діяльністю, аналітикою та маркетингом без потреби інтегрувати кілька різних сервісів. Для невеликого підприємства або приватного майстра, однак, така кількість функцій є надлишковою і ускладнює початок роботи з системою.

Booksy реалізує принципово інший підхід, поєднуючи функції CRM з елементами маркетингової платформи для залучення нових клієнтів [5]. Майстер або салон, зареєстрований у Booksy, отримує не лише інструмент управління записами, а й можливість бути знайденим потенційними клієнтами безпосередньо через каталог сервісу. Це робить платформу особливо привабливою для тих, хто тільки починає діяльність або прагне розширити клієнтську базу без суттєвих маркетингових витрат. Мобільний інтерфейс для клієнтів є інтуїтивно зрозумілим і забезпечує швидкий процес бронювання. Разом із тим така модель породжує залежність від платформи як основного каналу комунікації з клієнтами: бізнес частково втрачає контроль над власною аудиторією, а внутрішня конкуренція між майстрами у межах платформи може негативно впливати на рівень завантаженості окремих фахівців.

YCLIENTS є рішенням, орієнтованим насамперед на середній і великий бізнес із розвиненою організаційною структурою. Система пропонує широкий набір інструментів: фінансовий облік, складський контроль, розрахунок заробітної плати, програми лояльності, детальну аналітику ефективності персоналу та підтримку мережевої структури з кількома локаціями [6]. Глибина функціоналу робить YCLIENTS потужним інструментом для управління складним підприємством, однак вона ж є джерелом основного обмеження:

система вимагає значних зусиль для початкового налаштування та адаптації персоналу, що є суттєвим бар'єром для малого бізнесу з обмеженими ресурсами. Крім того, після 2022 року YCLIENTS втратила позиції на українському ринку через зміну власника на структуру, пов'язану з Росією, що спонукало значну частину українських салонів до переходу на вітчизняні альтернативи [32]. Станом на 2024 рік в Україні функціонує щонайменше сім власних CRM-систем для б'юті-бізнесу, що свідчить про активний розвиток цього сегмента вітчизняного програмного забезпечення [32].

Для систематизації результатів порівняльного аналізу розглянуті системи представлено у таблиці 1.

Таблиця 1.1

Порівняльний аналіз переваг та недоліків CRM-систем для б’юті індустрії

Система	Основні функції	Переваги	Недоліки
Fresha	онлайн-запис, календар, база клієнтів, онлайн-оплата, аналітика, маркетинг, управління персоналом	комплексне рішення «все в одному»; автоматизація більшості процесів; сучасний інтерфейс; безкоштовний базовий функціонал	комісії за платежі; складність для нових користувачів; надлишковий функціонал для малого бізнесу
Booksy	онлайн-запис, календар, база клієнтів, маркетинг, мобільний додаток, аналітика	залучення клієнтів через платформу; зручний мобільний інтерфейс; простота використання; швидкий запис	платні функції; залежність від платформи; внутрішня конкуренція між майстрами
YCLIENTS	онлайн-запис, база клієнтів, фінанси, аналітика, складський облік, зарплати, програми лояльності	потужна система управління бізнесом; детальна аналітика; підходить для мереж салонів; інтеграції	складність впровадження; платна підписка; надлишкова функціональність для малого бізнесу

Узагальнення результатів аналізу дозволяє виявити кілька ключових тенденцій розвитку CRM-рішень для б'юті-індустрії. По-перше, всі розглянуті системи орієнтовані на максимальну автоматизацію рутинних операцій, що дозволяє зменшити навантаження на адміністратора та підвищити точність планування. По-друге, зростає роль аналітики та даних: сучасні платформи дозволяють не лише вести записи, а й формувати управлінські звіти, аналізувати поведінку клієнтів і вимірювати ефективність персоналу. По-третє, клієнтський досвід стає самостійним конкурентним фактором: зручність мобільного інтерфейсу, швидкість бронювання та якість автоматичних нагадувань безпосередньо впливають на лояльність відвідувачів [7].

Водночас проведений аналіз виявляє і спільне обмеження існуючих рішень: жодне з них не є оптимальним для малого українського салону, який потребує простого, доступного та незалежного від зовнішньої платформи інструменту управління. Fresha та Booksy передбачають транзакційні витрати або маркетингову залежність, YCLIENTS є надмірно складною для невеликого бізнесу, а вітчизняні рішення, попри їх зростаючу кількість, поки що поступаються міжнародним платформам за рівнем функціональності та зручності. Buttle і Maklan підкреслюють, що ефективна CRM-система повинна відповідати реальному масштабу та структурі бізнесу, для якого вона призначена, а не нав'язувати власну логіку роботи [7]. Саме невідповідність між функціональними можливостями наявних рішень і реальними потребами малих салонів обґрунтовує доцільність розробки власного мобільного додатку, адаптованого до специфіки невеликого б'юті-підприємства.

Таким чином, проведений аналіз підтверджує, що ринок CRM-систем для б'юті-індустрії є зрілим і технологічно розвиненим, однак не позбавленим прогалин. Орієнтація на баланс між функціональністю та простотою використання, незалежність від зовнішніх платформ і адаптованість до потреб малого бізнесу є тими характеристиками, які мають визначати концепцію власного мобільного рішення.

1.3 Проблеми управління клієнтами в салоні краси

Незважаючи на зростання б'юті-ринку в Україні та розвиток доступних цифрових інструментів, значна частина салонів краси продовжує стикатися з системними проблемами в організації роботи з клієнтами. Ці проблеми мають як організаційний, так і технологічний характер, і в сукупності безпосередньо впливають на якість обслуговування, завантаженість персоналу та фінансові результати підприємства. Важливо, що кожна з цих проблем не є ізольованою: вони взаємно посилюють одна одну, формуючи системний дефіцит ефективності, який неможливо усунути точковими заходами без комплексної цифровізації процесів управління.

Найбільш поширеною і водночас найбільш відчутною для бізнесу є проблема ручного запису клієнтів. У більшості малих салонів запис здійснюється через телефонні дзвінки або переписку в месенджерах, де адміністратор вручну фіксує час, майстра і послугу. Такий підхід є принципово обмеженим з кількох причин. По-перше, він прив'язаний до робочого часу персоналу: клієнт, який хоче записатися ввечері або у вихідний день, або не може цього зробити, або змушений чекати відповіді до наступного ранку. По-друге, ручний запис є джерелом систематичних помилок — накладання записів, неправильно зафіксованого часу або майстра, втраченого повідомлення в потоці переписки. По-третє, процес є трудомістким для самого адміністратора, який змушений витрачати значну частину робочого часу на рутинну координацію замість роботи з клієнтами в салоні. Laudon та Laudon зазначають, що автоматизація операційних процесів дозволяє суттєво зменшити кількість помилок і підвищити продуктивність персоналу в сервісних організаціях [1]. За даними досліджень ринку, компанії, що впровадили системи онлайн-запису, фіксують збільшення доходу на 20% вже у перший рік використання [30], що свідчить про пряму фінансову значущість вирішення цієї проблеми.

Тісно пов'язаною з попередньою є проблема відсутності єдиної централізованої бази клієнтів. У більшості салонів, що не використовують спеціалізованого програмного забезпечення, інформація про відвідувачів

зберігається фрагментовано: частково у паперових журналах, частково в телефонах окремих майстрів або адміністратора, частково в архівах переписок у месенджерах. Така фрагментація даних унеможливлює швидкий доступ до інформації про конкретного клієнта та її ефективне використання. Майстер не може оперативно переглянути, які процедури і коли отримував відвідувач, які матеріали використовувалися або які індивідуальні особливості слід враховувати. Це знижує рівень персоналізації обслуговування, яке є одним із ключових факторів формування клієнтської лояльності у б'юті-сфері. Buttle і Maklan підкреслюють, що впровадження CRM-систем дозволяє централізувати клієнтські дані, забезпечити їх актуальність і використовувати для побудови довгострокових відносин із клієнтами [7].

Складність управління розкладом майстрів породжує окремий клас операційних проблем. Ефективний розклад у салоні краси — це не просто перелік записів, а динамічна система, що постійно змінюється під впливом нових бронювань, скасувань, переносів і змін у графіку персоналу. Без автоматизованого інструменту адміністратор змушений утримувати всі ці зміни вручну, що підвищує ймовірність конфліктів у розкладі. Накладання двох записів на одного майстра в той самий час, помилкове підтвердження вже зайнятого слота або неврахування часу на підготовку між процедурами — все це реальні ризики ручного планування, кожен із яких безпосередньо позначається на репутації салону та задоволеності клієнтів. Stair і Reynolds зазначають, що автоматизовані системи управління розкладом дозволяють оптимізувати використання часу персоналу та суттєво знизити операційні помилки в сервісному бізнесі [8].

Відсутність автоматичних нагадувань про заплановані візити є ще одним джерелом прямих фінансових втрат для салону. У б'юті-сфері досить поширеними є ситуації, коли клієнт забуває про запис і не з'являється без попереднього повідомлення, залишаючи майстра без роботи на відведений час. Такий пропущений слот, як правило, неможливо заповнити в короткий термін, що означає пряму втрату доходу. Використання автоматичних push-

повідомлень, SMS або email-нагадувань дозволяє значно зменшити кількість подібних випадків, одночасно формуючи у клієнта сприйняття салону як організованого та уважного до деталей підприємства [3]. Chaffey зазначає, що цифрові канали комунікації з клієнтами є не лише інструментом зручності, а й засобом підвищення фінансової стабільності сервісного бізнесу [3].

Обмеженість аналітичних можливостей позбавляє власника салону інструментів для обґрунтованого розвитку бізнесу. За відсутності систематизованих даних про популярність послуг, ефективність окремих майстрів, частоту повторних візитів і сезонні коливання попиту управлінські рішення ухвалюються переважно інтуїтивно. Це ускладнює планування асортименту, формування акційних пропозицій, оцінку ефективності маркетингових заходів та прогнозування доходів. Laudon та Laudon підкреслюють, що сучасні інформаційні системи дозволяють збирати та аналізувати операційні дані в режимі реального часу, перетворюючи їх на основу для прийняття обґрунтованих управлінських рішень [1]. У контексті б'юті-бізнесу це означає можливість виявити, які послуги є найбільш прибутковими, які клієнти відвідують салон найчастіше і в який час спостерігається найбільший попит.

Окремою проблемою є обмежена та несистематична комунікація з клієнтами за межами безпосереднього обслуговування. Без централізованого інструменту управління комунікацією салон не може ефективно інформувати клієнтів про нові послуги, сезонні акції або зміни в графіку роботи. Це знижує рівень залученості аудиторії та обмежує можливості для повторного продажу. Сучасні CRM-системи вирішують цю проблему шляхом автоматизації комунікаційних сценаріїв — від підтвердження запису до персоналізованих пропозицій на основі історії відвідувань клієнта [7].

Нарешті, відсутність цифрової системи управління стає критичним обмеженням при масштабуванні бізнесу. Якщо власник салону планує відкриття нових локацій або суттєве збільшення кількості майстрів, ручні процеси управління просто не здатні забезпечити необхідний рівень

координації. Кожна нова точка або новий співробітник у такій моделі додає не лише обсяг роботи, а й непропорційно зростаючу складність координації, що робить розширення ризикованим без попереднього впровадження автоматизованої системи [33].

Таким чином, проблеми управління клієнтами в салоні краси мають комплексний і взаємопов'язаний характер. Ручний запис, фрагментованість клієнтських даних, складність управління розкладом, відсутність автоматичних нагадувань, обмеженість аналітики та несистематична комунікація формують сукупність операційних дефіцитів, які неможливо усунути точковими заходами. Їх вирішення вимагає впровадження єдиної інтегрованої інформаційної системи — мобільного CRM-додатку, здатного об'єднати всі ці процеси та забезпечити ефективне управління бізнесом на якісно новому рівні.

Для наочного узагальнення виявлених проблем і обґрунтування запропонованого рішення побудовано дерево проблем і дерево цілей проєкту (рис. 1.1, рис. 1.2). Дерево проблем відображає причинно-наслідкові зв'язки між виявленими операційними дефіцитами салону краси: центральною проблемою є фрагментованість процесів управління клієнтами та відсутність єдиного цифрового інструменту, що породжує три групи наслідків — фінансові втрати від пропущених візитів, зниження якості клієнтського досвіду та неможливість масштабування бізнесу (рис. 1.1). Причинами центральної проблеми є ручний запис клієнтів через месенджери, відсутність централізованої бази клієнтів, відсутність автоматичних нагадувань, обмежена аналітика бізнесу та розрізнені канали комунікації. Дерево цілей є дзеркальним відображенням дерева проблем: кожна проблема трансформується у конкретну ціль, досягнення якої забезпечується відповідним функціональним модулем розроблюваної системи [1]. Так, центральною ціллю проєкту є створення мобільного CRM-додатку для автоматизації управління клієнтами салону б'юті послуг, а підцілями — автоматизація онлайн-запису клієнтів, централізація клієнтської бази з історією, впровадження автоматичних нагадувань, аналітика та фінансова звітність, а також омніканальний чат зі ШІ-асистентом (рис. 1.2). Реалізація сукупності цих

підцілей забезпечує досягнення очікуваних результатів проєкту: зменшення фінансових втрат від пропусків, підвищення якості клієнтського досвіду та забезпечення масштабованості бізнесу.

Рис. 1.1 Дерево проблем проєкту

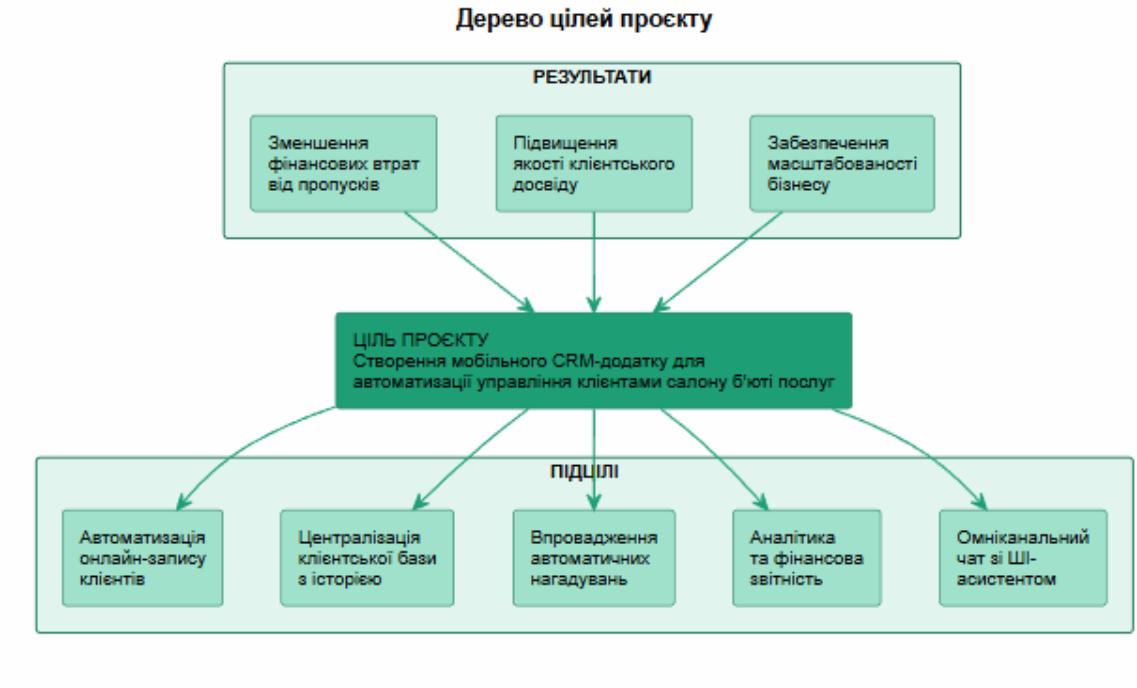


Рис. 1.2 Дерево цілей проєкту

Таким чином, обидві діаграми у сукупності формують логічний каркас проєкту: від діагностики поточного стану — через визначення цілей — до

обґрунтування складу функціональних модулів майбутньої системи. Побудова дерева проблем і дерева цілей є визнаною практикою проєктного аналізу, рекомендованою стандартами управління проєктами розвитку, зокрема методологією ZOOP/PCM, і дозволяє забезпечити узгодженість між виявленими потребами стейкхолдерів та архітектурними рішеннями системи [1]. Саме ця узгодженість є передумовою того, щоб розроблюваний продукт вирішував реальні проблеми бізнесу, а не лише відтворював функціонал наявних на ринку рішень.

1.4 Формування концепції мобільного додатку

Формування концепції мобільного додатку є ключовим етапом ініціювання проєкту, на якому визначається логіка майбутнього продукту, його цінність для користувача та відповідність реальним бізнес-потребам. Концепція не є технічним документом — вона відповідає на питання про те, який продукт створюється, для кого і чому саме в такому форматі. Laudon та Laudon зазначають, що успішне впровадження інформаційної системи починається з чіткого розуміння проблеми, яку вона має вирішити, та цільової аудиторії, для якої вона призначена [1]. Проведений аналіз бізнес-процесів салону та існуючих CRM-рішень створює необхідне підґрунтя для формування такої концепції.

Основна ідея продукту полягає у створенні єдиного цифрового середовища, яке об'єднує всі процеси взаємодії між клієнтом і салоном — від першого запису до аналізу повторних відвідувань. У традиційній моделі роботи ці процеси розподілені між різними каналами та інструментами: запис здійснюється через месенджери, інформація про клієнтів зберігається фрагментовано, а контроль розкладу залежить від уважності адміністратора. Впровадження мобільного додатку дозволяє централізувати всі ці процеси в одній системі, забезпечити їх автоматизацію та зробити доступними для всіх учасників у будь-який час. Такий підхід відповідає сучасним принципам побудови інформаційних систем для сервісного бізнесу, де інтеграція процесів є основою операційної ефективності [8].

Вибір мобільного формату як основного каналу реалізації є принциповим і обґрунтованим рішенням. Мобільні пристрої є основним засобом доступу до цифрових сервісів як для клієнтів, так і для персоналу салону. Клієнт записується на послугу тоді, коли це зручно йому — ввечері, у транспорті або в перерві між справами, — а не тоді, коли салон відкритий і адміністратор вільний. Майстер переглядає свій розклад на смартфоні перед початком робочого дня. Адміністратор оперативно вносить зміни, не прив'язуючись до стаціонарного комп'ютера. Wroblewski зазначає, що підхід Mobile First є не просто технічним рішенням, а стратегічним вибором на користь тієї платформи, де користувач проводить найбільше часу та де його взаємодія з сервісом є найбільш природною [24]. Chaffey також підкреслює, що мобільні канали взаємодії з клієнтами забезпечують вищий рівень залученості порівняно з веб-інтерфейсами у сфері послуг [3].

Цільова аудиторія додатку охоплює кілька груп користувачів із різними сценаріями використання системи. Власники салонів зацікавлені насамперед у контролі бізнес-показників: завантаженості майстрів, кількості записів, доходів і популярності послуг. Для них додаток є інструментом управлінської аналітики, який дозволяє приймати рішення на основі актуальних даних, а не інтуїції. Адміністратори використовують систему як основний робочий інструмент щоденно: створення та редагування записів, управління розкладом, ведення клієнтської бази та контроль оплат. Наявність централізованої системи суттєво спрощує їхню роботу та зменшує кількість помилок. Майстри отримують доступ до власного графіку, інформації про заплановані візити та клієнтську історію, що дозволяє їм краще підготуватися до кожного прийому та підвищити якість обслуговування. Клієнти взаємодіють із системою через мобільний інтерфейс для самостійного запису: вони обирають послугу, майстра і зручний час, отримують підтвердження та нагадування. Такий підхід відповідає сучасним очікуванням споживачів, орієнтованих на швидкість та автономність у взаємодії з сервісами [2]. Sommerville зазначає, що визначення всіх категорій користувачів та їхніх сценаріїв використання є обов'язковим

кроком перед формуванням вимог до системи, оскільки дозволяє уникнути ситуації, коли продукт добре вирішує задачі однієї групи, але залишає незадоволеними потреби інших [9].

Ключові функціональні можливості системи формуються на основі аналізу виявлених проблем та потреб цільової аудиторії. Реєстрація та авторизація користувачів із розподілом ролей забезпечує диференційований доступ до функцій системи залежно від типу користувача. Онлайн-запис на послуги дозволяє клієнтам самостійно бронювати час без участі адміністратора, що знімає обмеження, пов'язані з робочим графіком персоналу. Управління розкладом майстрів у реальному часі запобігає накладанням записів та дозволяє гнучко реагувати на зміни. Єдина база клієнтів із повною історією відвідувань забезпечує персоналізований підхід та є основою для аналітики. Автоматичні нагадування про записи — через push-повідомлення або SMS — зменшують кількість пропущених візитів і стабілізують завантаженість майстрів. Облік оплат і базова фінансова звітність дають власнику актуальну інформацію про доходи підприємства. Можливість онлайн-оплати підвищує зручність для клієнта та відповідає сучасним тенденціям розвитку цифрових платіжних інструментів у сфері послуг [3].

При визначенні функціонального наповнення принципово важливим є питання балансу між повнотою можливостей та простотою використання. Аналіз існуючих рішень показав, що перевантаженість функціями є одним із головних бар'єрів для впровадження CRM-систем у малому бізнесі. Nielsen зазначає, що зручність використання є одним із ключових факторів успіху програмного продукту, і будь-яка функція, яка ускладнює інтерфейс без пропорційної користі для користувача, знижує загальну ефективність системи [12]. Саме тому концепція пропонованого додатку передбачає реалізацію у форматі MVP — мінімально життєздатного продукту, що містить базовий функціонал, достатній для вирішення ключових проблем бізнесу, з можливістю поступового розширення у міру зростання потреб [8]. Такий підхід дозволяє швидше вивести продукт на ринок, перевірити його ефективність у реальних

умовах та мінімізувати ризики надмірних інвестицій у функції, які можуть виявитися незатребуваними.

Окремим функціональним пріоритетом є підвищення довіри клієнта до майстра ще до першого візиту. З цією метою система передбачає публічні профілі майстрів із галереєю портфоліо, структурованою за категоріями послуг. Клієнт може оцінити реальні результати роботи і прийняти усвідомлене рішення про запис, що знижує бар'єр першого звернення і підвищує конверсію. Додатково система інтегрує омніканальний чат із підтримкою ШІ-асистента, що дозволяє адміністратору обробляти звернення з усіх месенджерів в єдиному інтерфейсі з мінімальними витратами часу.

Окремої уваги заслуговує питання безпеки даних. Система обробляє персональну інформацію клієнтів — контактні дані, історію відвідувань, відомості про оплати, — тому забезпечення її захисту є не лише технічною вимогою, а й юридичним зобов'язанням. Відповідно до стандарту ISO/IEC 25010, безпека є однією з ключових характеристик якості програмного забезпечення, що включає захист від несанкціонованого доступу, цілісність даних та конфіденційність інформації користувачів [13]. Ці вимоги мають бути враховані вже на етапі формування концепції, а не додані як доповнення на пізніших стадіях розробки.

Таким чином, сформована концепція мобільного додатку спирається на чітко визначену проблему, конкретні групи користувачів та функціональні пріоритети, що відповідають реальним потребам б'юті-бізнесу. Орієнтація на простоту використання, мобільний формат і поетапне нарощування функціоналу визначає її як практично реалізовану та ринково обґрунтовану основу для подальшого проєктування системи.

1.5 Визначення зацікавлених сторін

Визначення зацікавлених сторін є обов'язковим етапом ініціювання будь-якого ІТ-проєкту, оскільки дозволяє врахувати інтереси всіх учасників процесу ще до початку технічної реалізації. Зацікавлена сторона — це будь-яка особа або група осіб, які можуть впливати на проєкт або зазнавати впливу з його

боку: як у процесі розробки, так і після впровадження системи. Ігнорування інтересів навіть одної групи stakeholders здатне призвести до того, що технічно коректний продукт виявиться незручним у використанні або не прийнятим цільовою аудиторією. Sommerville зазначає, що повне та точне визначення зацікавлених сторін є одним із найважливіших чинників успішності проєкту розробки програмного забезпечення, оскільки саме їхні потреби формують вимоги до системи [9]. У контексті розробки мобільного CRM-додатку для салону б'юті послуг виділяється п'ять основних груп зацікавлених сторін: власник бізнесу, адміністратор, майстри, клієнти та команда розробки.

Власник бізнесу є ініціатором впровадження системи та ключовою особою, яка визначає стратегічні пріоритети проєкту. Його основний інтерес полягає у підвищенні ефективності управління салоном, зниженні операційних витрат і збільшенні доходів. Для досягнення цих цілей власнику необхідний доступ до аналітичних даних: статистики записів, завантаженості майстрів, популярності послуг і динаміки фінансових показників. Laudon та Laudon підкреслюють, що саме використання аналітичних інструментів у CRM-системах дозволяє власникам бізнесу переходити від інтуїтивного управління до прийняття рішень на основі даних, що суттєво підвищує конкурентоспроможність підприємства [1]. Крім аналітики, власник зацікавлений у масштабованості системи: рішення має залишатися ефективним навіть у разі зростання кількості майстрів або відкриття нових локацій. Таким чином, для власника бізнесу додаток є насамперед інструментом стратегічного контролю та розвитку, а не лише засобом автоматизації операцій.

Адміністратор є тим учасником, для якого впровадження системи матиме найбільш відчутний вплив на щоденну роботу. Саме адміністратор взаємодіє з системою найінтенсивніше: створює та редагує записи, управляє розкладом майстрів, веде клієнтську базу, контролює оплати та забезпечує комунікацію з відвідувачами. У традиційній моделі більша частина цих операцій виконується вручну і потребує постійної уваги та координації. Впровадження мобільного додатку суттєво змінює характер роботи адміністратора: рутинні операції

автоматизуються, ризик помилок зменшується, а вивільнений час може бути спрямований на безпосередню роботу з клієнтами в салоні. Для цієї групи користувачів критично важливою є зручність інтерфейсу та швидкість виконання типових дій. Nielsen зазначає, що зручність використання є одним із ключових факторів успішності програмного продукту, і особливо це стосується систем, з якими користувач працює щоденно протягом тривалого часу [12]. Складний або неінтуїтивний інтерфейс у таких умовах не лише знижує продуктивність, а й формує стійкий негативний досвід, що може призвести до відмови від системи.

Майстри салону є важливою групою зацікавлених сторін, хоча їхня взаємодія із системою є більш обмеженою порівняно з адміністратором. Основна потреба майстра полягає у доступі до актуальної інформації про власний графік роботи та заплановані візити. Можливість переглянути розклад на наступний день, отримати деталі щодо послуги та клієнта, а також оперативно дізнатися про зміни або скасування записів дозволяє майстру краще організувати свій робочий час і підвищити якість підготовки до кожного прийому. Доступ до клієнтської історії дає можливість враховувати попередні процедури та індивідуальні особливості відвідувача, що безпосередньо впливає на рівень персоналізації обслуговування. Norman підкреслює, що ефективна взаємодія користувача з системою можлива лише тоді, коли інтерфейс відповідає реальним робочим сценаріям і не вимагає від користувача зайвих розумових зусиль [17]. Для майстрів це означає, що система повинна надавати саме ту інформацію, яка потрібна в конкретний момент, без перевантаження зайвими функціями, призначеними для інших ролей.

Клієнти є кінцевими користувачами мобільного інтерфейсу системи і водночас тією групою, задоволеність якої визначає комерційний успіх продукту. Сучасний клієнт б'юті-салону очікує від сервісу швидкості, зручності та доступності у будь-який час. Можливість самостійно записатися на послугу, обрати зручний час і конкретного майстра без необхідності телефонувати або чекати відповіді в месенджері є для клієнта суттєвою конкурентною перевагою

салону. Додаткову цінність створюють автоматичні нагадування про майбутній візит, можливість переглянути або змінити запис і доступ до історії відвідувань. Такий рівень цифрового сервісу відповідає сучасним очікуванням споживачів і безпосередньо впливає на їхню лояльність до бізнесу [2]. Chaffey зазначає, що зручність цифрової взаємодії є одним із ключових факторів, що визначають вибір клієнта між конкурентами у сфері послуг [3]. Саме тому якість клієнтського досвіду у мобільному додатку є не менш важливою, ніж функціональність адміністративної частини системи.

Команда розробки є зацікавленою стороною, яка безпосередньо відповідає за технічну реалізацію продукту та його відповідність вимогам усіх інших груп. До її складу входять менеджер проєкту, розробники, UX/UI-дизайнер, тестувальник і бізнес-аналітик. Кожен учасник команди має власні інтереси в проєкті: чіткі та несуперечливі вимоги, реалістичні терміни, зрозуміла пріоритизація задач і конструктивний зворотний зв'язок від замовника. Pressman зазначає, що ефективна взаємодія між командою розробки та іншими зацікавленими сторонами є однією з ключових умов успішного завершення проєкту, і її забезпечення є прямою відповідальністю менеджера проєкту [10]. Недостатня комунікація або нечітке розуміння вимог на ранніх етапах неминуче призводить до переробок, перевитрат бюджету та затримок у термінах.

Для систематизації інформації про зацікавлені сторони, їхні основні інтереси та очікування від системи доцільно представити узагальнені дані таблиці 1.2.

Таблиця 1.2

Зацікавлені сторони проєкту та їхні основні інтереси

Зацікавлена сторона	Роль у проєкті	Основні інтереси та очікування
1	2	3
Власник бізнесу	Замовник, ініціатор проєкту	Аналітика діяльності салону; контроль завантаженості майстрів і доходів;

1	2	3
Майстри	Внутрішні користувачі системи	Доступ до власного графіку; інформація про заплановані візити та клієнтів; своєчасні сповіщення про зміни в розкладі
Клієнти	Зовнішні користувачі мобільного додатку	Самостійний онлайн-запис у будь-який час; вибір послуги, майстра і часу; автоматичні нагадування; перегляд і зміна запису
Команда	Виконавець	Чіткі та несуперечливі вимоги; реалістичні
		масштабованість системи; підвищення прибутковості
Адміністратор	Основний операційний користувач	Зручний і швидкий інтерфейс; автоматизація запису та розкладу; централізована клієнтська база; зменшення ручної роботи

Закінчення таблиці 1.2

розробки	проєкту	терміни; зрозуміла пріоритизація задач; конструктивний зворотний зв'язок від замовника

Врахування інтересів усіх визначених груп зацікавлених сторін є необхідною умовою для формування збалансованих вимог до системи. Кожна група висуває до продукту власні пріоритети, і завданням проєктної команди є знаходження рішень, що задовольняють ці пріоритети без суперечностей між собою. Зокрема, потреба адміністратора у функціональній повноті системи має бути збалансована з потребою клієнта у простоті інтерфейсу, а аналітичні запити власника — з технічними можливостями та обмеженнями бюджету проєкту. Такий підхід відповідає сучасним принципам управління вимогами в IT-проєктах і забезпечує основу для подальшого моделювання системи та визначення її функціональних і нефункціональних характеристик [9].

Таким чином, визначення зацікавлених сторін дозволяє сформувати повне уявлення про коло учасників проєкту, їхні потреби та рівень впливу на кінцевий продукт. Отримані результати є вихідною точкою для наступного етапу — постановки задачі проєкту та формування паспорту, де визначені інтереси stakeholders трансформуються у конкретні цілі, функції та обмеження системи.

1.6 Формування паспорту проєкту

Паспорт проєкту є базовим документом ініціювання, який узагальнює ключові характеристики майбутнього продукту ще до початку його технічної реалізації. Він фіксує мету, цілі, функціональні можливості, обмеження та очікувані результати проєкту у структурованому вигляді, забезпечуючи єдине розуміння задачі між усіма зацікавленими сторонами. Відповідно до сучасних підходів до управління IT-проєктами, формування паспорту є обов'язковим кроком на етапі ініціювання, оскільки дозволяє офіційно авторизувати проєкт, визначити його межі та призначити відповідальних осіб [10]. За відсутності такого документа проєкт ризикує розвиватися без чітких орієнтирів, що

призводить до розширення обсягу робіт за межі початкових домовленостей, перевитрат бюджету та конфліктів між учасниками щодо пріоритетів.

Необхідність формування паспорту для даного проєкту зумовлена кількома факторами. По-перше, система охоплює кілька різних груп користувачів із різними потребами, і без чіткого визначення меж функціоналу існує ризик надмірного розширення вимог у процесі розробки. По-друге, проєкт реалізується в умовах обмеженого бюджету та часу, що вимагає чіткої пріоритизації цілей і свідомого вибору підходу MVP. По-третє, наявність паспорту забезпечує узгодженість між замовником і командою розробки щодо того, що входить до обсягу проєкту, а що виходить за його межі. Stair і Reynolds зазначають, що чітке визначення обсягу та цілей є одним із ключових факторів успішного завершення проєктів з розробки інформаційних систем, і будь-яка невизначеність на цьому рівні багатократно зростає у вартості виправлення на пізніших етапах [8].

Назва проєкту відображає його практичну спрямованість: розробка мобільного CRM-додатку для управління клієнтами салону б'юті послуг. Основною метою є створення програмного продукту, що автоматизує ключові операційні процеси салону, підвищує якість обслуговування клієнтів і забезпечує власника інструментами для управління бізнесом на основі даних. Досягнення цієї мети передбачає вирішення конкретних задач: усунення ручного запису клієнтів шляхом впровадження онлайн-бронювання, централізацію клієнтської бази та історії відвідувань, автоматизацію управління розкладом майстрів, впровадження системи автоматичних нагадувань і забезпечення базового фінансового обліку.

Функціональні можливості системи у межах паспорту визначаються на рівні модулів, без деталізації технічної реалізації. Система включає модуль реєстрації та авторизації користувачів із розподілом ролей, модуль онлайн-запису на послуги, модуль управління розкладом майстрів, модуль клієнтської бази з історією відвідувань, модуль послуг і цін, модуль автоматичних нагадувань, модуль обліку оплат та панель адміністратора з базовою

аналітикою. Такий рівень опису функціоналу є достатнім для ініціювання проєкту і формує основу для подальшого детального визначення вимог у наступних розділах роботи.

Обмеження проєкту є не менш важливою складовою паспорту, ніж його цілі, оскільки дозволяють реалістично оцінити можливості реалізації. До основних обмежень належать: обмежений бюджет, що зумовлює вибір підходу MVP і відмову від ряду другорядних функцій на першому етапі; обмежений термін реалізації, який вимагає ефективного планування та пріоритизації задач; залежність від стабільного інтернет-з'єднання для роботи більшості функцій системи; необхідність забезпечення захисту персональних даних клієнтів відповідно до чинного законодавства; різний рівень цифрової грамотності користувачів, що висуває підвищені вимоги до простоти інтерфейсу. Pressman зазначає, що свідоме визначення обмежень на етапі ініціювання є ознакою зрілого підходу до управління проєктом, оскільки дозволяє команді зосередитися на пріоритетних задачах і уникнути неконтрольованого розширення обсягу робіт [10].

Очікувані результати проєкту охоплюють два рівні: продуктовий і бізнесовий. На продуктовому рівні результатом є функціональний прототип мобільного CRM-додатку з реалізованим базовим функціоналом та задокументованою архітектурою, придатною для подальшого розвитку. На бізнесовому рівні очікується оптимізація процесу запису клієнтів, зменшення кількості помилок у розкладі, зниження навантаження на адміністратора та підвищення рівня задоволеності клієнтів завдяки зручному цифровому каналу взаємодії із салоном. Buttle і Maklan підкреслюють, що оцінка результатів CRM-проєкту має охоплювати як технічні, так і бізнесові показники, оскільки саме останні визначають реальну цінність впровадженої системи [7].

Основні характеристики проєкту узагальнено у таблиці 1.3.

Таблиця 1.3

Паспорт проєкту

Розділ	Опис

1	2
Назва проєкту	Розробка мобільного CRM-додатку для управління клієнтами салону б'юті послуг
Мета проєкту	Створення мобільного додатку для автоматизації управління клієнтами, розкладом майстрів та операційними процесами салону б'юті послуг
Цілі проєкту	Автоматизація онлайн-запису клієнтів; централізація клієнтської бази та історії відвідувань; оптимізація управління розкладом майстрів; впровадження автоматичних нагадувань; забезпечення базового фінансового обліку; підвищення якості обслуговування клієнтів
Основні функції системи	Реєстрація та авторизація з розподілом ролей; онлайн-запис на послуги; управління розкладом майстрів; клієнтська база з історією відвідувань; управління послугами та цінами; автоматичні push/SMS-нагадування; облік оплат; панель адміністратора з аналітикою
Обмеження проєкту	Обмежений бюджет і терміни реалізації; підхід MVP на першому етапі; залежність від інтернет-з'єднання; вимоги до захисту персональних даних; різний рівень цифрової грамотності користувачів

Закінчення таблиці 1.3

1	2
Очікувані результати	Функціональний прототип мобільного додатку; оптимізація процесу запису клієнтів; зменшення помилок у розкладі; зниження навантаження на адміністратора; підвищення рівня задоволеності клієнтів; архітектура, придатна для масштабування
Зацікавлені сторони	Замовник: власник салону; команда: Project Manager, розробники, дизайнер, QA, аналітик; користувачі: адміністратор,

	майстри, клієнти

Сформований паспорт проєкту забезпечує узгодженість між цілями замовника, потребами користувачів і технічними можливостями команди розробки. Він визначає межі системи, фіксує пріоритети та створює документальну основу для переходу до наступного етапу — моделювання системи та формування детальних вимог, що є предметом другого розділу роботи.

РОЗДІЛ 2. МОДЕЛЮВАННЯ СИСТЕМИ ТА ФОРМУВАННЯ ВИМОГ

2.1 Постановка задачі проєкту

Постановка задачі є відправним етапом будь-якого проєкту з розробки інформаційної системи, оскільки саме тут формується чітке і формалізоване розуміння того, яку проблему має вирішити майбутній продукт, в яких умовах він створюватиметься і які результати вважатимуться успішними. Без такого фундаменту проєкт ризикує розвиватися без орієнтирів: вимоги розширюватимуться стихійно, пріоритети конфліктуватимуть, а фінальний продукт може виявитися технічно коректним, але практично неефективним. Laudon та Laudon зазначають, що якість постановки задачі безпосередньо визначає якість кінцевої системи, оскільки саме на цьому етапі встановлюється відповідність між потребами бізнесу, очікуваннями користувачів і технічними можливостями реалізації [1].

Аналіз предметної області, проведений у першому розділі, дозволяє сформулювати центральну проблему проєкту: більшість малих і середніх салонів б'юті-послуг в Україні не мають інтегрованого цифрового інструменту для управління клієнтами, розкладом і фінансовим обліком. Наслідком є системна операційна неефективність: ручний запис клієнтів, фрагментовані клієнтські дані, накладки у розкладі, фінансові втрати від пропущених візитів та відсутність аналітичної підтримки управлінських рішень. Кожна з цих проблем у відриві від інших є усувною локальними заходами, однак у сукупності вони утворюють взаємопов'язану систему дефіцитів, яка потребує комплексного технологічного рішення. Важливо також, що ці проблеми не є специфічними для одного типу підприємств: вони характерні для переважної більшості малих салонів незалежно від їхнього асортименту послуг або географії, що підтверджує системний, а не ситуативний характер проблеми [33].

Необхідно розрізняти два рівні постановки задачі: бізнесовий і системний. На бізнесовому рівні задача формулюється як підвищення

операційної ефективності салону, зменшення фінансових втрат від неефективного управління часом майстрів і підвищення рівня задоволеності клієнтів. На системному рівні та сама задача трансформується у конкретні вимоги до функціоналу, архітектури і характеристик якості програмного продукту. Sommerville підкреслює, що перехід від бізнесової задачі до системних вимог є одним із найскладніших і найбільш критичних кроків у розробці програмного забезпечення, оскільки саме тут найчастіше виникають непорозуміння між замовником і командою [9]. Для уникнення таких непорозумінь постановка задачі у даній роботі охоплює обидва рівні і забезпечує прозорий зв'язок між ними.

Запропонованим рішенням є мобільний CRM-додаток, що інтегрує ключові операційні процеси салону в єдину систему. Вибір мобільного формату зумовлений тим, що смартфон є основним пристроєм взаємодії як для клієнтів, так і для персоналу. Клієнт записується тоді, коли йому зручно, незалежно від робочого часу адміністратора. Майстер переглядає графік у будь-який момент. Адміністратор вносить зміни без прив'язки до стаціонарного робочого місця. Такий рівень доступності є необхідною умовою для реальної автоматизації, а не лише формального переведення процесів у цифровий формат [3]. Wroblewski зазначає, що підхід Mobile First — це не просто технічний вибір, а визнання того факту, що мобільний пристрій є основним контекстом взаємодії сучасного користувача з цифровими сервісами [24].

Метою системи є автоматизація управління клієнтами салону б'юті послуг шляхом створення мобільного додатку, що забезпечує онлайн-запис, управління розкладом майстрів, централізоване зберігання клієнтських даних, автоматичні нагадування та базову аналітику діяльності підприємства. Для досягнення цієї мети система має вирішити сім конкретних задач. По-перше, замінити ручний запис клієнтів функцією онлайн-бронювання, доступною цілодобово. По-друге, централізувати клієнтську базу з повною історією відвідувань і наданих послуг. По-третє, автоматизувати управління розкладом майстрів із відображенням доступності в реальному часі та запобіганням

конфліктам. По-четверте, впровадити автоматичні push- і SMS-нагадування для зменшення кількості пропущених візитів. По-п'яте, забезпечити облік оплат і базову фінансову звітність. По-шосте, надати власнику бізнесу інструменти аналітики для прийняття обґрунтованих управлінських рішень. По-сьоме, забезпечити масштабованість системи для її подальшого розвитку без необхідності повного перепроєктування.

Визначення цілей системи нерозривно пов'язане з визначенням критеріїв їхнього досягнення. Мета сформульована якісно є лише орієнтиром; для оцінки успішності проєкту необхідні вимірювані показники. Зокрема, впровадження онлайн-запису вважатиметься успішним, якщо частка записів через мобільний додаток перевищить 70% від загального обсягу після місяця використання. Ефективність системи нагадувань — якщо кількість пропущених без повідомлення візитів скоротиться не менш ніж на 30% порівняно з доцифровим періодом. Ці критерії є орієнтовними і мають бути уточнені спільно із замовником у процесі деталізації вимог, однак їх формулювання вже на етапі постановки задачі є ознакою зрілого підходу до управління проєктом [10].

Разом із цілями необхідно чітко визначити обмеження проєкту, оскільки саме вони встановлюють реалістичні межі реалізації. Першим і найбільш значущим є підхід MVP — мінімально життєздатного продукту. Це означає, що у першій версії реалізується базовий функціонал, достатній для вирішення ключових проблем, тоді як другорядні або складні у реалізації функції відкладаються на наступні ітерації. Pressman зазначає, що ітеративний підхід до розробки дозволяє суттєво знизити ризики проєкту і швидше отримати зворотний зв'язок від реальних користувачів [10]. Другим обмеженням є залежність від стабільного інтернет-з'єднання: більшість функцій системи вимагають онлайн-доступу, що є прийнятним для міського середовища. Третім є необхідність відповідності вимогам захисту персональних даних відповідно до чинного законодавства України. Четвертим — різний рівень цифрової грамотності користувачів, що ставить підвищені вимоги до простоти та

інтуїтивності інтерфейсу. П'ятим є обмежений бюджет і терміни реалізації, які визначають вибір технологічного стеку та обсяг залученої команди.

Відповідно до стандарту ISO/IEC 25010, якісна інформаційна система повинна відповідати не лише функціональним вимогам, а й характеристикам якості: зручності використання, продуктивності, надійності, безпеки та масштабованості [13]. Ці характеристики мають враховуватися вже на етапі постановки задачі, оскільки їх ігнорування на ранніх стадіях неминуче призводить до дорогих переробок на пізніших. Sommerville підкреслює, що нефункціональні вимоги нерідко є більш критичними для успіху системи, ніж функціональні, оскільки визначають умови її реального використання [9]. Для мобільного додатку у сфері послуг це означає передусім швидкість відгуку, стабільність роботи і захист персональних даних клієнтів — саме ці характеристики формують базовий рівень довіри користувача до системи.

Слід також зазначити, що постановка задачі у контексті цього проєкту має подвійний вимір: продуктовий і проєктний. Продуктовий вимір описує, що має робити система і якими характеристиками якості вона повинна відповідати. Проєктний вимір визначає, як буде організований процес її створення: які методології управління проєктом застосовуватимуться, як формуватиметься команда, які ризики є ймовірними і як ними управляти. Саме проєктний вимір є предметом четвертого розділу роботи, тоді як поточний розділ зосереджується на системному вимірі — моделюванні системи, визначенні вимог і проєктуванні її архітектури. Такий розподіл дозволяє забезпечити структурованість дослідження і уникнути змішування різнорідних аспектів проєкту в межах одного розділу.

Таким чином, постановка задачі проєкту охоплює чітко сформульовану проблему, конкретну мету і перелік задач системи, вимірювані критерії успіху, визначені обмеження і характеристики якості. Ці елементи разом формують проєктний контекст, у межах якого здійснюватиметься подальше моделювання системи, визначення вимог і проєктування архітектурних рішень.

2.2 Аналіз користувачів та ролей системи

Визначення ролей користувачів є необхідною передумовою як для формування вимог до системи, так і для проєктування її архітектури. Роль у контексті інформаційної системи — це не просто категорія людей, які взаємодіють із продуктом, а функціональна позиція з конкретним набором дозволених операцій, сценаріїв використання та обмежень доступу. Sommerville зазначає, що розуміння ролей і відповідних їм сценаріїв є основою для коректного формування як функціональних, так і нефункціональних вимог до системи [9]. Помилки на цьому етапі мають непропорційно великі наслідки: якщо роль визначена неповно, система отримує функціональні прогалини; якщо надмірно — інтерфейс перевантажується зайвими функціями, що знижує ефективність роботи реальних користувачів.

В основі розподілу ролей лежить принцип рольового контролю доступу (Role-Based Access Control, RBAC), який є стандартним підходом в інформаційних системах, що обробляють різнорідні категорії даних із різним рівнем конфіденційності. Відповідно до цього принципу, кожен користувач отримує доступ лише до тих функцій і даних, які необхідні для виконання його ролі — не більше і не менше [10]. Це одночасно забезпечує безпеку системи, спрощує інтерфейс для кожної категорії користувачів і мінімізує ризик несанкціонованих дій. У межах розроблюваного мобільного CRM-додатку виділяються чотири основні ролі — клієнт, адміністратор, майстер і власник бізнесу, — а також допоміжна технічна роль системного адміністратора.

Перш ніж переходити до детального опису кожної ролі, доцільно розглянути ключові виміри, за якими вони відрізняються між собою. Перший вимір — частота використання системи: адміністратор взаємодіє з нею безперервно протягом робочого дня, майстер — кілька разів на день, власник — кілька разів на тиждень, клієнт — нерегулярно, зазвичай раз на кілька тижнів. Другий вимір — тип виконуваних операцій: клієнт переважно читає і бронює, адміністратор читає, редагує і координує, майстер переважно читає, власник — переглядає агреговані дані. Третій вимір — рівень доступу до

даних: власник має найширший доступ до фінансових і аналітичних даних, адміністратор — до операційних даних усіх клієнтів і майстрів, майстер — до власного розкладу і клієнтської історії, клієнт — лише до власних даних. Ці відмінності безпосередньо визначають архітектуру системи і логіку розмежування прав доступу [13].

Клієнт є зовнішнім користувачем системи і водночас тією групою, взаємодія з якою визначає комерційну ефективність продукту. Його основний сценарій — самостійний запис на послугу: вибір процедури, майстра, зручного часу, підтвердження бронювання. Крім того, клієнт може переглядати власну історію відвідувань, отримувати автоматичні нагадування, змінювати або скасовувати записи. Важливою характеристикою цієї ролі є те, що клієнт взаємодіє із системою нерегулярно і не інвестує час у її освоєння: кожен сеанс використання є, по суті, першим із точки зору когнітивного навантаження. Krug зазначає, що ефективний мобільний інтерфейс — це той, де користувач досягає своєї мети без необхідності думати про те, як ним користуватися [23]. Для клієнтської частини системи це означає, що основний сценарій запису повинен виконуватися не більш ніж за чотири кроки, кожен із яких є інтуїтивно зрозумілим без жодних інструкцій. Chaffey додає, що зручність цифрового каналу взаємодії є одним із ключових факторів, що визначають вибір клієнта між конкурентами в сфері послуг, і що незручний інтерфейс є прямою причиною відтоку клієнтів до конкурентів із зручнішими цифровими рішеннями [3].

Додатково слід враховувати неоднорідність клієнтської аудиторії за рівнем цифрової грамотності. Частина клієнтів салону є активними користувачами смартфонів, для яких онлайн-запис є природним і звичним форматом. Інша частина — переважно клієнти старшого віку — може мати обмежений досвід взаємодії з мобільними сервісами і потребує максимально спрощеного інтерфейсу. Ця неоднорідність є додатковим аргументом на користь мінімалістичного дизайну клієнтської частини без зайвих опцій і ускладнень. Nielsen підкреслює, що хороша система повинна бути одночасно

зручною для початківця і ефективною для досвідченого користувача — і досягнення цього балансу є одним із найскладніших завдань UX-проєктування [12].

Адміністратор є центральним операційним користувачем системи — тим, хто взаємодіє з нею найінтенсивніше і для якого вона є основним робочим інструментом. До його функцій належать: створення, редагування і скасування записів, управління розкладом майстрів, ведення клієнтської бази, контроль оплат і комунікація з відвідувачами. На відміну від клієнта, адміністратор використовує систему щоденно і протягом тривалого часу, тому для нього критично важлива не лише простота, а й ефективність: мінімальна кількість кроків для виконання типових операцій, швидкий пошук клієнта, зручне відображення розкладу на день або тиждень. Nielsen підкреслює, що для систем із інтенсивним щоденним використанням продуктивність роботи досвідченого користувача є важливішою метрикою, ніж легкість освоєння для початківця [12].

Особливістю роботи адміністратора є те, що він часто виконує кілька задач одночасно: відповідає клієнту у салоні, приймає телефонний дзвінок і вносить зміни у розклад. У такому контексті система повинна бути стійкою до переривань: незавершена операція не повинна призводити до втрати введених даних, а повернення до перерваної задачі повинно бути природним і швидким. Garrett зазначає, що проєктування під реальний контекст використання, а не під ідеальні умови, є принципом, який відрізняє практично ефективні системи від теоретично коректних [16]. Крім того, адміністратор є основним каналом зворотного зв'язку від клієнтів до системи: саме він першим помічає функціональні прогалини або незручності і може надати найбільш цінні спостереження для подальшого розвитку продукту.

Майстер є внутрішнім користувачем системи з обмеженим, але чітко визначеним колом задач. Він не управляє записами інших майстрів і не має доступу до фінансових даних підприємства, однак потребує актуальної і деталізованої інформації про власний графік роботи, деталі запланованих

візитів і релевантну клієнтську історію. Саме доступ до клієнтської історії є тією функцією, яка найбільш суттєво впливає на якість роботи майстра: знаючи, які процедури клієнт отримував раніше, якими матеріалами користувався майстер і які індивідуальні особливості слід враховувати, фахівець може підготуватися до прийому значно ефективніше, ніж у разі кожного разу починати з нуля. Norman зазначає, що ефективна система повинна надавати користувачу саме ту інформацію, яка потрібна в конкретний момент, не перевантажуючи його зайвим і не примушуючи самостійно шукати потрібні дані в масиві непов'язаної інформації [17].

Характерною особливістю майстра як користувача є те, що він взаємодіє із системою у стані, коли його увага розподілена: він переглядає розклад між клієнтами, уточнює деталі запису безпосередньо перед початком процедури або отримує сповіщення про зміни у графіку під час роботи. У таких умовах інтерфейс для майстра повинен забезпечувати миттєвий доступ до ключової інформації — найближчого запису, імені клієнта, послуги і часу — без необхідності навігації через кілька екранів. Це є вимогою не лише до зручності, а й до безпеки: майстер, який відволікається на складний інтерфейс під час роботи з клієнтом, знижує якість обслуговування.

Власник бізнесу використовує систему переважно для аналітики та стратегічного контролю. Його задачі — перегляд фінансових показників, аналіз завантаженості майстрів, оцінка популярності послуг, моніторинг кількості нових і повторних клієнтів. Це якісно інший характер взаємодії порівняно з адміністратором або майстром: власник заходить у систему не щогодини, а кілька разів на тиждень, але потребує агрегованих, зрозуміло структурованих і візуалізованих даних. Для власника найважливішою є не швидкість виконання операцій, а якість і повнота аналітичної інформації. Laudon та Laudon підкреслюють, що аналітичні інструменти CRM-систем дозволяють власникам підприємств переходити від управління за відчуттями до управління на основі даних, що суттєво підвищує якість стратегічних рішень і знижує ризики [1]. Для малого б'юті-бізнесу це особливо актуально, оскільки власник часто

поєднує управлінські функції з безпосередньою участю у роботі салону і не має часу на трудомісткий ручний аналіз.

Системний адміністратор є допоміжною технічною роллю, яка не бере участі в операційній діяльності салону, але забезпечує технічне функціонування системи: управління обліковими записами, налаштування параметрів, оновлення програмного забезпечення, моніторинг безпеки та усунення технічних проблем. У контексті малого бізнесу ця роль часто виконується розробником або аутсорс-підрядником, а не штатним співробітником. Незважаючи на те, що системний адміністратор не є кінцевим бізнес-користувачем, його потреби також мають бути враховані при проєктуванні: зокрема, система повинна мати зрозумілу адміністративну панель, журнал подій для діагностики проблем і можливість гнучкого налаштування параметрів без необхідності зміни програмного коду.

Важливим результатом аналізу ролей є не лише опис кожної з них, а й розуміння типових сценаріїв взаємодії між ними в межах єдиної системи. Клієнт створює запис через мобільний інтерфейс — система автоматично оновлює розклад і надсилає підтвердження. Адміністратор бачить новий запис у зведеному календарі, за потреби вносить корективи. Майстер отримує актуальний графік і деталі клієнта перед початком прийому. Власник наприкінці тижня переглядає аналітику за завантаженістю та доходами. Цей ланцюг взаємодії демонструє, що всі ролі функціонують в єдиному інформаційному середовищі, і будь-який розрив у ньому — відсутність своєчасного оновлення даних або некоректний доступ — негативно впливає на якість обслуговування. Саме тому інтеграція ролей у єдину систему з чітко розмежованими правами доступу є не просто технічною вимогою, а функціональною основою продукту.

Таким чином, проведений аналіз користувачів і ролей системи дозволяє сформувати повне і диференційоване уявлення про те, для кого і як саме розробляється продукт. Визначені ролі, характер їхньої взаємодії із системою і відмінності у потребах є безпосередньою основою для формування

функціональних і нефункціональних вимог, розглянутих у наступному підрозділі.

2.3 Визначення функціональних та нефункціональних вимог до системи

Формування вимог є одним із найважливіших і водночас найбільш недооцінюваних етапів розробки програмного забезпечення. Якість визначених вимог безпосередньо впливає на всі наступні стадії проєкту: від проєктування архітектури до тестування і впровадження. Sommerville підкреслює, що вимоги мають описувати не те, як система реалізована, а те, що вона повинна робити і в яких умовах функціонувати [9]. Pressman додає, що розрізнення функціональних і нефункціональних вимог є принциповим, оскільки вони описують різні аспекти системи і визначають різні архітектурні рішення [10].

Функціональні вимоги

FR001. Реєстрація користувачів. Система повинна забезпечувати реєстрацію нових користувачів із верифікацією за номером телефону (SMS-код), перевіркою унікальності облікового запису і автоматичним присвоєнням ролі відповідно до типу реєстрації (клієнт, адміністратор, майстер, власник).

FR002. Авторизація та управління доступом. Система повинна забезпечувати захищену авторизацію за email і паролем із генерацією JWT-токена, розподіл прав доступу на основі ролей користувача (RBAC) і можливість відновлення пароля через підтверджений номер телефону.

FR003. Онлайн-запис клієнта. Система повинна надавати клієнту можливість самостійно обрати послугу з каталогу, обрати конкретного майстра або скористатися опцією «будь-який доступний», переглянути доступні часові слоти в реальному часі і завершити бронювання не більш ніж за чотири кроки без участі адміністратора.

FR004. Управління розкладом майстрів. Система повинна забезпечувати відображення зведеного календаря всіх майстрів у форматах дня, тижня і місяця, автоматичне блокування зайнятих слотів, запобігання накладанню

записів і можливість адміністратора вручну створювати, редагувати і скасовувати записи.

FR005. Клієнтська база та історія відвідувань. Система повинна централізовано зберігати контактні дані клієнтів, повну історію їхніх відвідувань, надані послуги і нотатки майстра, а також забезпечувати пошук клієнта за іменем або номером телефону.

FR006. Профілі майстрів і портфоліо. Система повинна надавати клієнту можливість переглянути профіль майстра із зазначенням спеціалізації, рейтингу і галереї виконаних робіт, структурованої за категоріями послуг, а майстру або адміністратору — завантажувати і редагувати фотографії портфоліо.

FR007. Автоматичні нагадування. Система повинна автоматично надсилати клієнту push-повідомлення або SMS за 24 години і за 2 години до запланованого візиту, логувати факт відправки і не дублювати нагадування при повторному запуску фонового процесу.

FR008. Облік оплат і фінансова звітність. Система повинна фіксувати факт оплати по кожному запису із зазначенням суми, форми розрахунку і дати, а також формувати зведені фінансові звіти за обраний період у розрізі послуг і майстрів.

FR009. Омніканальний чат. Система повинна об'єднувати вхідні повідомлення з Telegram, Viber, WhatsApp і Instagram Direct в єдиний inbox адміністратора через інтеграцію з агрегатором Chatwoot, забезпечуючи відправку відповіді у відповідний канал без переходу між додатками.

FR010. ШІ-асистент адміністратора. Система повинна аналізувати вхідні повідомлення клієнтів за допомогою API великої мовної моделі, розпізнавати намір і пропонувати адміністратору швидкі дії (запис, відправка прайсу, перенесення), що виконуються лише після явного підтвердження.

FR011. Управління каталогом послуг. Система повинна забезпечувати створення, редагування і архівування послуг із зазначенням назви, опису, тривалості і ціни, а також групування послуг за категоріями для відображення у формі запису клієнта.

FR012. Аналітична панель власника. Система повинна надавати власнику бізнесу доступ до агрегованих показників: загального доходу за обраний період, завантаженості кожного майстра у відсотках, рейтингу найпопулярніших послуг і кількості нових та повторних клієнтів.

Деталізований перелік функціональних вимог із пріоритизацією за методом MoSCoW представлено у таблиці 2.1.

Таблиця 2.1

Функціональні вимоги системи

Код	**Вимога**	**Опис**	**Пріоритет**
FR001	Реєстрація користувачів	Реєстрація з верифікацією, присвоєння ролі	Must have
FR002	Авторизація та RBAC	JWT-авторизація, розподіл прав за роллю	Must have
FR003	Онлайн-запис клієнта	Самостійне бронювання за 4 кроки	Must have
FR004	Управління розкладом	Календар день/тиждень/місяць, блокування слотів	Must have
FR005	Клієнтська база	Централізоване зберігання, пошук, нотатки	Must have
FR006	Профілі і портфоліо	Перегляд профілю майстра, галерея робіт	Should have
FR007	Автоматичні нагадування	Push / SMS за 24 год і 2 год до візиту	Must have
FR008	Облік оплат і	Фіксація розрахунків, фінансові	Must have

	звітність	звіти	
FR009	Омніканальний чат	Єдиний inbox TG, Viber, WA, Instagram	Should have
FR010	ШІ-асистент	Розпізнавання намірів, швидкі дії	Should have
FR011	Управління послугами	Каталог, ціни, тривалість, архівування	Must have
FR012	Аналітична панель	KPI власника, доходи, завантаженість	Should have
FR013	Онлайн-оплата	Інтеграція з LiqPay, webhook-підтвердження	Could have

Нефункціональні вимоги

NFR001. Зручність використання. Інтерфейс клієнтської частини повинен забезпечувати виконання основного сценарію запису без попереднього навчання. Оцінка зручності за шкалою SUS має становити не менше 70 балів зі 100. Мінімальний розмір інтерактивних елементів — 44×44 пікселі [12].

NFR002. Продуктивність. Час відгуку системи на API-запит не повинен перевищувати 2 секунди за нормальних умов навантаження. Система повинна стабільно функціонувати при одночасній роботі щонайменше 50 активних користувачів без деградації продуктивності [8].

NFR003. Надійність. Доступність системи після розгортання — не нижче 99%. Автоматичне резервне копіювання даних здійснюється щодобово. Система повинна відновлювати сесію користувача після короткочасного збою без втрати введених даних.

NFR004. Безпека. Увесь трафік між клієнтом і сервером передається по захищеному каналу HTTPS. Паролі зберігаються виключно у хешованому вигляді (bcrypt). Система відповідає вимогам законодавства України щодо захисту персональних даних [13].

NFR005. Масштабованість. Архітектура системи повинна забезпечувати можливість додавання нових майстрів, послуг і функціональних модулів без перепроєктування існуючих компонентів.

NFR006. Сумісність. Мобільний додаток повинен коректно функціонувати на iOS 14+ і Android 10+, підтримуючи адаптивний інтерфейс для різних розмірів екранів.

NFR007. Інтернаціоналізація. Система повинна підтримувати українську та англійську мови інтерфейсу з можливістю перемикання у налаштуваннях профілю.

Деталізований перелік нефункціональних вимог представлено у таблиці 2.2.

Таблиця 2.2

Нефункціональні вимоги системи

Код	**Характеристика**	**Вимога**
NFR001	Зручність використання	SUS ≥70 балів; основний сценарій — ≤4 кроки
NFR002	Продуктивність	Час відгуку ≤2 сек; підтримка ≥50 користувачів
NFR003	Надійність	Доступність ≥99%; резервне копіювання щодобово
NFR004	Безпека	HTTPS; bcrypt; відповідність законодавству
NFR005	Масштабованість	Додавання модулів без переробки архітектури
NFR006	Сумісність	iOS 14+, Android 10+; адаптивний інтерфейс
NFR007	Інтернаціоналізація	Українська та англійська мови

Сформовані нефункціональні вимоги є не менш критичними для успіху системи, ніж функціональні, оскільки визначають умови її реального

використання. Sommerville підкреслює, що нефункціональні вимоги нерідко стають вирішальним чинником при виборі користувачем між конкурентними рішеннями, адже саме вони визначають надійність, безпеку та зручність повсякденної роботи із системою [9]. У контексті мобільного CRM-додатку для б'юті-салону це означає, що навіть бездоганно реалізований функціонал запису не матиме цінності, якщо час відгуку перевищуватиме прийнятні межі або система виходитиме з ладу у пікові години завантаження.

Особливої уваги заслуговує вимога NFR001 щодо зручності використання: поріг SUS ≥70 балів і обмеження основного сценарію чотирма кроками є конкретними та вимірюваними критеріями, які можуть бути перевірені у ході UX-тестування з реальними користувачами. Вимога NFR002 щодо продуктивності — час відгуку ≤2 секунди при навантаженні ≥50 одночасних користувачів — відповідає стандартним очікуванням мобільної аудиторії та враховує потенційне масштабування системи. Вимоги NFR003 і NFR004 забезпечують базовий рівень довіри до системи з боку власника бізнесу: доступність ≥99% і щоденне резервне копіювання мінімізують ризик втрати клієнтських даних, а використання HTTPS і bcrypt відповідає чинним вимогам законодавства України щодо захисту персональних даних [13].

Таким чином, сукупність семи нефункціональних вимог формує якісний профіль системи відповідно до стандарту ISO/IEC 25010, охоплюючи характеристики зручності, продуктивності, надійності, безпеки, масштабованості, сумісності та інтернаціоналізації. Виконання цих вимог є обов'язковою умовою для успішного впровадження продукту в реальному операційному середовищі салону б'юті послуг.

2.4 Моделювання сценаріїв використання системи

Моделювання сценаріїв використання є методом опису поведінки системи через призму взаємодії користувача з нею для досягнення конкретної мети. На відміну від переліку функцій, сценарії дозволяють зрозуміти не що система робить, а як саме відбувається взаємодія: який крок іде за яким, що відбувається у нестандартних ситуаціях і який результат вважається успішним.

Cockburn зазначає, що якісно описані сценарії використання є найефективнішим інструментом для узгодження розуміння системи між замовником, аналітиком і командою розробки, оскільки вони описують поведінку у термінах, зрозумілих усім сторонам [20]. Стандарт UML визначає Use Case як формалізований опис взаємодії між актором і системою для досягнення конкретної цілі [21].

У межах даного проєкту виділяються чотири основні актори — клієнт, адміністратор, майстер і власник бізнесу — та відповідні їм сценарії використання. Кожен сценарій описується через передумову, основний потік подій, альтернативні потоки і постумову.

Центральним сценарієм для клієнта є «Запис на послугу». Передумовою є авторизація клієнта у системі. Основний потік: клієнт переходить до розділу запису, обирає категорію послуги, конкретну процедуру, майстра і зручний часовий слот, переглядає підсумок запису і підтверджує бронювання. Система зберігає запис, оновлює розклад майстра і надсилає клієнту підтвердження. Постумовою є активний запис у системі і оновлений розклад. Альтернативний потік передбачає ситуацію, коли обраний майстер або час є недоступними: система відображає повідомлення і пропонує альтернативні варіанти — інший час або іншого майстра. Якщо жоден варіант не підходить, клієнт може залишити запит і отримати сповіщення, коли з'явиться доступний слот.

Другим сценарієм клієнта є «Скасування або зміна запису». Клієнт переходить до списку своїх бронювань, обирає активний запис і вибирає дію — редагувати або скасувати. При редагуванні система відображає доступні альтернативи і після підтвердження оновлює запис. При скасуванні система запитує підтвердження, після якого запис видаляється з розкладу, а майстер отримує сповіщення. Цей сценарій є важливим для зменшення кількості незапланованих простоїв, оскільки дозволяє звільнити слот для іншого клієнта [3].

Для адміністратора ключовим є сценарій «Управління записами». Адміністратор може створювати новий запис вручну (наприклад, за

телефонним зверненням клієнта), обираючи клієнта з бази або додаючи нового, призначаючи послугу, майстра і час. Система перевіряє доступність слота і, за відсутності конфлікту, зберігає запис. Альтернативний потік: обраний слот зайнятий — система відображає це і пропонує найближчі доступні варіанти. Адміністратор також може редагувати або скасовувати будь-який існуючий запис, причому система автоматично надсилає клієнту сповіщення про зміну.

Сценарій «Управління розкладом» дозволяє адміністратору налаштовувати робочий графік майстрів: встановлювати робочі дні і години, фіксувати перерви, відпустки і неробочі дні. Система відображає всі ці параметри у зведеному календарі, автоматично блокуючи недоступні слоти для бронювання. Stair і Reynolds підкреслюють, що автоматизація управління розкладом є одним із основних факторів підвищення операційної ефективності в сервісному бізнесі [8].

Для майстра основним сценарієм є «Перегляд розкладу та деталей запису». Майстер входить у систему, відкриває власний календар на обраний день або тиждень, переглядає список запланованих візитів з деталями — часом, тривалістю, послугою і коротким профілем клієнта. При необхідності він може переглянути повну клієнтську історію, щоб підготуватися до прийому. Система також надсилає майстру push-сповіщення про нові або змінені записи в режимі реального часу.

Для власника бізнесу визначений сценарій «Перегляд аналітики». Власник обирає часовий діапазон і отримує зведений звіт: загальний дохід, кількість записів, середній чек, завантаженість кожного майстра, перелік найбільш затребуваних послуг. Ці дані дозволяють виявляти тенденції, оцінювати ефективність персоналу і приймати обґрунтовані рішення щодо розвитку бізнесу [1].

Окремо слід виділити наскрізний сценарій «Автоматичне нагадування», який не ініціюється жодним актором, а запускається системою автоматично. За встановлений проміжок часу до запланованого візиту система надсилає клієнту push-повідомлення або SMS із нагадуванням про дату, час, майстра і послугу.

Якщо клієнт підтверджує відвідування, статус запису оновлюється. Якщо клієнт скасовує — запускається сценарій скасування. Такий автоматизований цикл взаємодії дозволяє суттєво зменшити кількість пропущених візитів без залучення адміністратора [3].

На основі описаних сценаріїв побудовано діаграму варіантів використання, яка узагальнює функціональні можливості системи та взаємодію акторів, яку зображено на рис. 2.1.

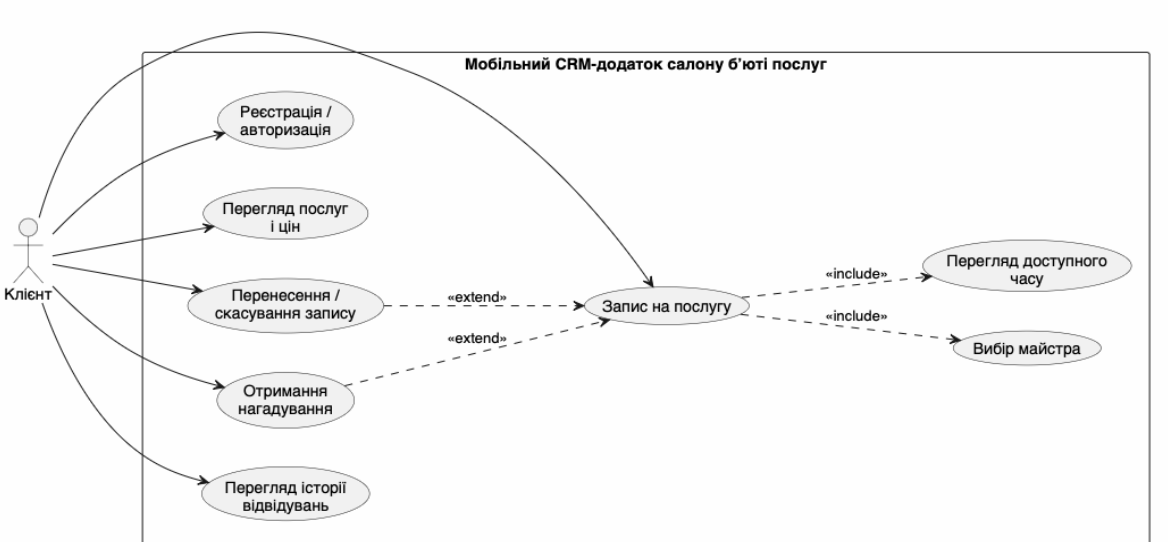


Рис. 2.1 Діаграма варіантів використання мобільного CRM-додатку салону б'юті послуг

Повна деталізована діаграма з усіма зв'язками між акторами і варіантами використання наведена у Додатку А.

2.5 Концептуальна модель інформаційної системи

Концептуальна модель інформаційної системи є рівнем опису, що стоїть між вимогами і технічною реалізацією. Вона описує не код і не структуру бази даних, а логіку функціонування системи в цілому: з яких компонентів вона складається, хто з нею взаємодіє і як інформація переміщується між елементами. Sommerville зазначає, що концептуальна модель дозволяє сформувати спільне розуміння системи між усіма учасниками проєкту ще до початку технічної реалізації, що суттєво знижує ризик архітектурних помилок на пізніших етапах [9]. Правильно побудована концептуальна модель є також

інструментом перевірки повноти вимог: якщо якийсь процес або роль не знаходить відображення у моделі, це сигналізує про прогалину у специфікації.

Архітектурно система складається з чотирьох основних рівнів. Перший — мобільний клієнтський додаток, який є інтерфейсом взаємодії для всіх категорій користувачів. Другий — серверна частина (бекенд), яка реалізує бізнес-логіку, обробляє запити від клієнтського додатку і забезпечує взаємодію з базою даних. Третій — база даних, яка централізовано зберігає всю інформацію системи: дані про користувачів, клієнтів, записи, послуги, розклад і оплати. Четвертий — зовнішні сервіси, інтегровані через API: сервіс push-сповіщень, SMS-шлюз і платіжна система. Такий чотирирівневий поділ забезпечує модульність системи і можливість незалежного масштабування або заміни окремих компонентів без перебудови всієї архітектури [8].

Мобільний клієнтський додаток реалізує інтерфейс для всіх чотирьох ролей користувачів. Кожна роль бачить свою версію додатку — не у вигляді окремого застосунку, а у вигляді диференційованого інтерфейсу в межах однієї програми, де доступні функції та відображувані дані визначаються роллю авторизованого користувача. Клієнт бачить каталог послуг, форму запису і свою клієнтську історію. Адміністратор — зведений календар, клієнтську базу і панель управління. Майстер — власний розклад і деталі записів. Власник — аналітичну панель із ключовими показниками. Такий підхід відповідає принципу єдиної точки входу, який спрощує підтримку і оновлення системи [24].

Серверна частина реалізує бізнес-логіку і виступає посередником між клієнтським додатком і базою даних. Вона отримує запити від додатку, перевіряє права доступу відповідно до ролі користувача, виконує необхідні операції з даними і повертає результат. Серверна частина також відповідає за автоматизовані процеси, що не ініціюються користувачем безпосередньо, — зокрема за відправку нагадувань у визначений час і оновлення статусів записів. Larman зазначає, що відокремлення бізнес-логіки від інтерфейсу є ключовим

принципом побудови масштабованих інформаційних систем, оскільки дозволяє змінювати інтерфейс без зміни логіки і навпаки [22].

Взаємодія між компонентами системи будується на основі запит-відповідь через захищений канал. Клієнтський додаток надсилає запит на сервер, сервер обробляє його і за необхідності звертається до бази даних або зовнішнього сервісу, після чого повертає відповідь додатку. Усі запити проходять через рівень автентифікації і авторизації, що гарантує відповідність виконуваних операцій правам поточного користувача [13].

Скорочена версія концептуальної моделі наведена у рис. 2.2. Модель наочно демонструє, що всі чотири ролі користувачів взаємодіють із системою через єдиний мобільний інтерфейс, тоді як серверна частина виступає централізованим вузлом обробки, що координує звернення до бази даних і зовнішніх сервісів. Така архітектура відповідає принципам розділення відповідальності і є типовою для сучасних мобільних клієнт-серверних застосунків [8].

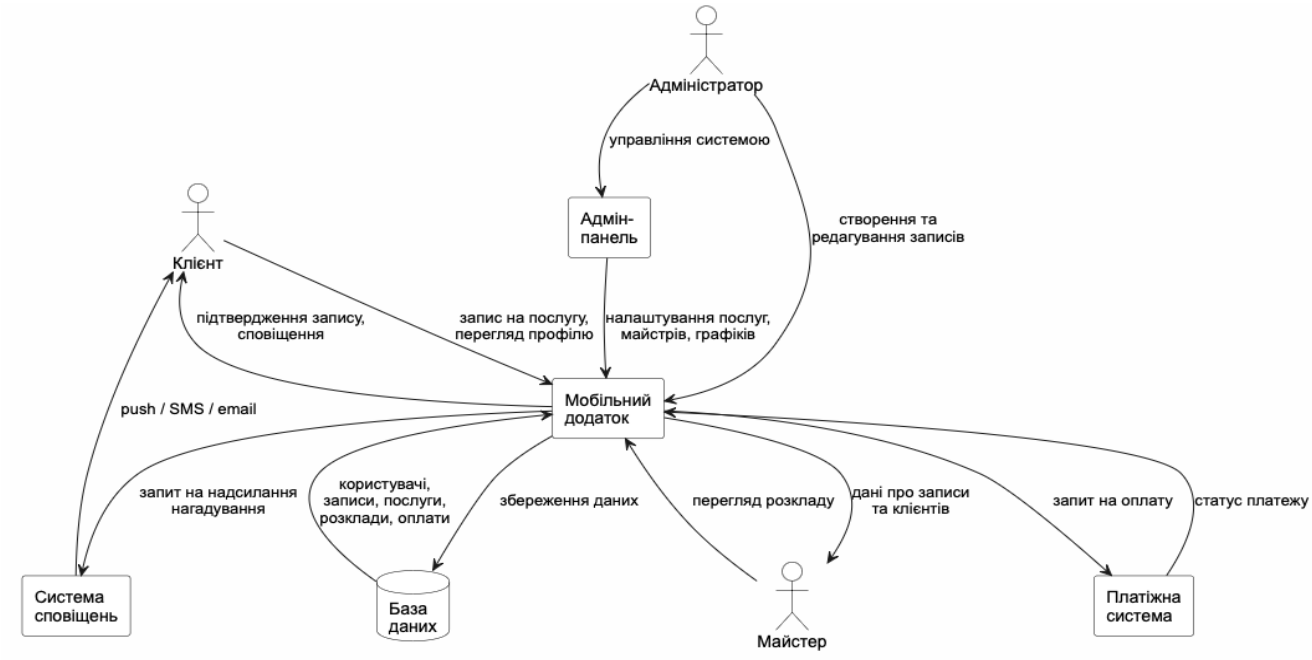


Рис. 2.2 – Концептуальна модель інформаційної системи мобільного CRM-додатку

Повна версія концептуальної моделі з деталізацією потоків даних між компонентами наведена у Додатку Б.

Для систематизації архітектурних рішень усі ключові компоненти системи, їх рівень розміщення та функціональне призначення зведено до таблиці 2.3. Таблиця охоплює дев'ять компонентів — від мобільного інтерфейсу і модулів авторизації, запису, розкладу, клієнтської бази, послуг і оплат до модуля нагадувань та аналітики — і для кожного з них визначає рівень архітектури (клієнтський, серверний, інфраструктурний або зовнішній) та конкретне функціональне призначення (таблиці 2.3). Такий структурований опис дозволяє отримати цілісне уявлення про склад системи ще до переходу до детального проєктування і слугує орієнтиром для розподілу відповідальності між членами команди розробки.

Таблиця 2.3

Основні компоненти концептуальної моделі

Компонент	Рівень	Призначення
Мобільний додаток (UI)	Клієнтський	Інтерфейс взаємодії для всіх ролей користувачів
Модуль авторизації	Клієнтський / Серверний	Реєстрація, вхід, управління сесіями і ролями
Модуль онлайн-запису	Клієнтський / Серверний	Бронювання послуг, перегляд і зміна записів
Модуль розкладу	Серверний	Управління графіком майстрів, контроль доступності слотів
Модуль клієнтської бази	Серверний	Зберігання і пошук даних про клієнтів та їхню історію
Модуль послуг	Серверний	Каталог послуг, цін і тривалості процедур
Модуль оплат	Серверний	Облік розрахунків, інтеграція з платіжним шлюзом
Модуль нагадувань	Серверний / Зовнішній	Автоматичне надсилання push і SMS у визначений час

Модуль аналітики	Серверний	Формування звітів і агрегованих показників
База даних	Інфраструктурний	Централізоване зберігання всіх даних системи
Сервіс push-сповіщень	Зовнішній	Доставка push-повідомлень на мобільні пристрої
SMS-шлюз	Зовнішній	Надсилання SMS-нагадувань клієнтам
Платіжна система	Зовнішній	Обробка онлайн-платежів

Концептуальна модель підтверджує, що всі визначені функціональні вимоги мають своє місце в архітектурі системи і реалізуються через конкретні компоненти відповідного рівня. Кожен із дев'яти компонентів, описаних у таблиці 2.3, відповідає одній або кільком функціональним вимогам із переліку FR001–FR012, що свідчить про повноту і несуперечливість розробленої специфікації. Водночас модель виявляє міжкомпонентні залежності, які необхідно враховувати під час проєктування: модуль нагадувань залежить від модуля запису — для отримання даних про заплановані візити — і від зовнішнього сервісу push-сповіщень та SMS-шлюзу для фактичної доставки повідомлень; модуль оплат залежить від модуля запису і від платіжної системи як зовнішнього провайдера; модуль аналітики агрегує дані з кількох модулів одночасно — записів, оплат і клієнтської бази.

Ці залежності є не просто архітектурними деталями, а прямими вимогами до структури бази даних: таблиці повинні бути спроєктовані так, щоб забезпечити коректні зв'язки між сутностями і підтримувати необхідні запити без надлишкового дублювання даних. Larman зазначає, що виявлення залежностей між компонентами на етапі концептуального моделювання є ключовою умовою уникнення архітектурного боргу на пізніших стадіях розробки [22].

Саме тому результати концептуального моделювання безпосередньо визначають підходи до проєктування структури бази даних і архітектури програмного забезпечення, що є предметом наступного розділу роботи.

РОЗДІЛ 3. ПЛАНУВАННЯ ТА УПРАВЛІННЯ ПРОЄКТОМ РОЗРОБКИ

3.1 Вибір методології управління проєктом

Вибір методології управління проєктом є одним із перших і найбільш визначальних рішень на етапі планування, оскільки саме методологія задає загальну логіку організації роботи команди, спосіб взаємодії із замовником і підхід до управління змінами протягом усього проєктного циклу. Не існує універсальної методології, придатної для будь-якого типу проєктів: вибір залежить від характеру продукту, рівня визначеності вимог, розміру команди і очікувань замовника. Pressman зазначає, що правильно обрана методологія є не адміністративним формалізмом, а практичним інструментом, що дозволяє команді зосередитися на створенні цінності замість управління хаосом [10].

У теорії управління ІТ-проєктами традиційно розрізняють два принципово різних підходи: каскадний (Waterfall) і гнучкий (Agile). Каскадна модель передбачає послідовне виконання чітко визначених фаз — аналіз вимог, проєктування, розробка, тестування, впровадження — де кожна наступна фаза починається лише після повного завершення попередньої. Ця модель добре підходить для проєктів зі стабільними і повністю визначеними вимогами, де зміни в процесі розробки є малоймовірними або неприйнятними. Проте вітчизняні дослідники зазначають, що для ІТ-проєктів каскадна модель є надмірно жорсткою і не враховує специфіку галузі, де вимоги нерідко змінюються вже у процесі реалізації, а отже, застосування виключно лінійного підходу несе підвищений ризик невідповідності фінального продукту реальним потребам замовника [44].

Agile-підхід, навпаки, будується на ітеративному і інкрементальному принципі: замість того щоб намагатися повністю спланувати проєкт на початку, команда розробляє продукт короткими циклами — ітераціями або спринтами — в кінці кожного з яких отримує робочий фрагмент функціоналу, придатний для демонстрації та оцінки замовником. Agile Manifesto проголошує пріоритет робочого програмного забезпечення над вичерпною документацією, взаємодії з

клієнтом над формальними контрактами і реагування на зміни над дотриманням початкового плану [39]. Для проєкту розробки мобільного CRM-додатку, де частина вимог формувалася і уточнювалася в процесі аналізу предметної області, такий підхід є значно більш природним і ефективним.

Серед конкретних фреймворків Agile найбільш поширеним у розробці мобільних додатків є Scrum, що за різними оцінками охоплює до 70% усіх Agile-проєктів у світі [45]. Scrum визначає конкретну структуру ролей (Product Owner, Scrum Master, команда розробки), артефактів (Product Backlog, Sprint Backlog, Increment) і церемоній (Sprint Planning, Daily Standup, Sprint Review, Sprint Retrospective) [40]. Двотижневі спринти дозволяють підтримувати регулярний ритм розробки і забезпечують замовнику можливість бачити і оцінювати прогрес кожні два тижні, а не чекати фінального результату місяцями. Дослідження у сфері мобільної розробки підтверджують, що команди, які використовують Scrum, демонструють вищу передбачуваність термінів і кращу якість кінцевого продукту порівняно з командами, що застосовують виключно каскадний підхід [46].

Для даного проєкту обрано методологію Scrum із двотижневими спринтами як основний фреймворк управління. Враховуючи невеликий розмір команди і обмежений бюджет, ряд Scrum-практик адаптується до реального контексту. Зокрема, роль Scrum Master суміщається з обов'язками менеджера проєкту, а замість щоденних standup-зустрічей проводяться три зустрічі на тиждень, що є прийнятним для розподіленої команди малого розміру. Sommerville зазначає, що гнучкість Agile полягає не у формальному дотриманні всіх елементів фреймворку, а у відданості принципам ітеративної розробки та тісної взаємодії з замовником [9].

Додатково для управління задачами і візуалізації стану роботи використовується Kanban-дошка, інтегрована в середовище розробки GitHub Projects. Kanban дозволяє в будь-який момент бачити, які задачі знаходяться в черзі, яка знаходиться в розробці, яка — у тестуванні і яка завершена, що забезпечує прозорість для всіх учасників команди і замовника без додаткових

звітних зустрічей. Поєднання Scrum як ритмізуючого фреймворку і Kanban як інструменту візуалізації є поширеною практикою, відомою як Scrumban, і дозволяє отримати переваги обох підходів одночасно [40].

Загальна тривалість проєкту становить 52 тижні (12 місяців), що відповідає 24 двотижневим спринтам. Такий горизонт є реалістичним для команди з п'яти осіб при розробці повноцінного продукту з розширеним функціоналом — омніканальним чатом, ШІ-асистентом і портфоліо майстрів — і дозволяє забезпечити достатню якість продукту без надмірного скорочення часу на тестування і документацію.

3.2 Імплементація фреймворку Scrum. User Story та формування беклогу продукту

Практична імплементація Scrum розпочинається з формування Product Backlog — впорядкованого переліку всього функціоналу, що має бути реалізований у межах проєкту. Product Backlog є живим документом: він поповнюється, переупорядковується і уточнюється протягом усього проєктного циклу відповідно до нового розуміння вимог і зворотного зв'язку від замовника. Schwaber і Sutherland визначають Product Backlog як єдине джерело вимог для будь-яких змін, що вносяться в продукт, і підкреслюють, що відповідальність за його наповнення і пріоритизацію лежить на Product Owner [40]. Основним інструментом наповнення беклогу є User Stories — короткі описи функціоналу з точки зору кінцевого користувача.

User Story є ключовим артефактом Agile-розробки, що відображає потребу конкретного типу користувача у форматі, зрозумілому як бізнесу, так і команді розробки. Стандартний шаблон User Story має вигляд: «Як [роль], я хочу [дія], щоб [результат/цінність]». Такий формат забезпечує три ключові елементи: визначення актора (хто використовує функцію), опис дії (що саме потрібно зробити) і обґрунтування цінності (навіщо це потрібно). Cohn зазначає, що User Story є не просто технічною специфікацією, а запрошенням до розмови між командою розробки і замовником — саме ця розмова, а не письмовий текст, є носієм реального розуміння вимог [47]. Кожна User Story

повинна відповідати критеріям INVEST: бути незалежною (Independent), припускати переговори (Negotiable), мати цінність (Valuable), бути оцінюваною (Estimable), невеликою (Small) і тестованою (Testable) [47].

На початку проєкту проводиться Story Writing Workshop — спільна сесія Product Owner, бізнес-аналітика і команди розробки, на якій формулюються User Stories для всього запланованого функціоналу. Великі і узагальнені User Stories (Epics) декомпозуються на менші, придатні для реалізації в межах одного спринту. Кожній User Story присвоюються Story Points — відносна оцінка складності, що враховує обсяг роботи, складність реалізації і ступінь невизначеності. Для оцінки використовується шкала Фібоначчі (1, 2, 3, 5, 8, 13), де менше значення відповідає простішій задачі. Story Points є інструментом відносної оцінки, а не прямого перекладу в години, що дозволяє команді планувати обсяг спринту на основі накопиченого досвіду (velocity) без прив'язки до календарного часу [48].

Фрагмент сформованого Product Backlog із переліком User Stories, критеріями приймання, пріоритетами та оцінкою в Story Points наведено у таблиці 3.1.

Таблиця 3.1

Фрагмент Product Backlog із User Stories та критеріями приймання

Код US	User Story	Критерії приймання	Пріоритет	SP
1	2	3	4	5
US001	Як клієнт, я хочу зареєструватися за номером телефону, щоб мати власний обліковий запис	1. Форма містить поля: ім'я, телефон, пароль. 2. Система перевіряє унікальність номера. 3. Після реєстрації надсилається SMS-підтвердження. 4. Після підтвердження створюється обліковий запис	Must have	3

Закінчення таблиці 3.1

1	2	3	4	5
US002	Як адміністратор, я хочу входити за email і паролем, щоб мати доступ до адмін-панелі	1. Форма містить поля email і пароль. 2. При невірних даних — повідомлення про помилку. 3. Після успішного входу — перехід до календаря	Must have	2
US003	Як клієнт, я хочу переглянути список послуг із цінами, щоб обрати потрібну процедуру	1. Послуги згруповані за категоріями. 2. Для кожної послуги відображається назва, тривалість і ціна. 3. Є рядок пошуку і фільтр за категорією	Must have	3
US004	Як клієнт, я хочу обрати майстра або «будь-який доступний», щоб записатися зручним способом	1. Список майстрів із фото, іменем і рейтингом. 2. Опція «будь-який доступний» показує найближчий вільний слот. 3. При виборі майстра відображаються його доступні часові слоти	Must have	3
US005	Як клієнт, я хочу бачити вільні часові слоти у календарі, щоб обрати зручний час	1. Відображення доступних слотів по годинах. 2. Зайняті слоти відображаються закресленими. 3. Після вибору — екран підтвердження	Must have	8
…				
US016	Як клієнт, я хочу оплатити послугу онлайн, щоб не носити готівку	1. Інтеграція з LiqPay. 2. Платіжні дані не зберігаються на сервері. 3. Після оплати — підтвердження і зміна статусу запису	Could have	13

Повна версія таблиці в додатку В.

Sprint Backlog формується на кожному спринт-плануванні шляхом відбору User Stories з верхівки Product Backlog відповідно до їхнього пріоритету

і планової velocity команди. Відібрані Stories декомпозуються на конкретні технічні задачі (Tasks), які розподіляються між членами команди і відображаються на Kanban-дошці у стовпцях «To Do», «In Progress», «Review» і «Done». Щоденний моніторинг прогресу здійснюється через Burndown Chart — графік залишкового обсягу роботи, що дозволяє рано виявляти відставання від плану [40].

Для ілюстрації деталізації спринтового планування у таблиці 3.2 наведено план виконання робіт Спринту 1, що охоплює реалізацію модуля авторизації на серверному і клієнтському рівнях, налаштування середовища розробки та підготовку початкових макетів.

Для ілюстрації деталізації спринтового планування у таблиці 3.2 наведено план виконання робіт Спринту 1.

Таблиця 3.2

План виконання робіт Спринту 1

Задача / Підзадача	Назва роботи	Трив. (год)	Виконавець	Технології
1	2	3	4	5
ST01.1.1	Проєктування схеми БД — таблиці User, Client, Master	4	Backend Dev	PostgreSQL, Prisma
ST01.1.2	Написання міграцій і seed-даних	3	Backend Dev	Prisma Migrate
ST01.2.1	Розробка API-ендпоінту POST /auth/register	4	Backend Dev	Node.js, Express
ST01.2.2	Реалізація валідації вхідних даних (email, пароль)	3	Backend Dev	Joi Library
ST01.3.1	Підключення бібліотеки хешування паролів	2	Backend Dev	bcrypt
ST01.3.2	Реалізація генерації та верифікації JWT-токена	4	Backend Dev	JSON Web Token
ST02.1.1	Написання Middleware для перевірки автентифікації	3	Backend Dev	Node.js

Закінчення таблиці 3.2

1	2	3	4	5
ST02.1.2	Реалізація RBAC — перевірка ролей користувача	3	Backend Dev	Node.js
ST03.1.1	Верстка екрану реєстрації (поля введення)	4	Mobile Dev	React Native
ST03.1.2	Верстка екрану входу в систему	3	Mobile Dev	React Native
ST03.2.1	Підключення React Navigation — базова навігація	4	Mobile Dev	React Navigation
ST03.2.2	Підключення Redux Toolkit — управління станом	3	Mobile Dev	Redux Toolkit
ST04.1.1	Макети екранів авторизації у Figma	4	UX/UI Designer	Figma
ST04.1.2	Дизайн-токени: кольори, типографіка, відступи	3	UX/UI Designer	Figma
ST05.1.1	Юніт-тести модуля авторизації	4	QA Engineer	Jest
ST05.1.2	Smoke testing API через Postman	2	QA Engineer	Postman
РАЗОМ		**55**	**Команда**	

Як видно з таблиці 3.2, перший спринт є фундаментальним для всього подальшого розвитку системи: більшість задач виконує бекенд-розробник, що відображає пріоритет серверної інфраструктури на початковому етапі. Загальний обсяг робіт спринту становить 34 години, з яких 26 припадає на бекенд і 11 — на мобільну частину. Паралельне виконання серверних і клієнтських задач дозволяє уникнути простою мобільного розробника в очікуванні готового API, оскільки верстка екранів і налаштування навігації не залежать від фінальної реалізації серверних ендпоінтів. Такий підхід до організації першого спринту відповідає рекомендаціям Scrum Guide щодо формування інкременту, що має самостійну цінність і може бути продемонстрований замовнику вже за підсумками спринту [39].

3.3 Організаційна структура команди проєкту

Формування команди проєкту є практичним втіленням організаційного плану: саме від складу команди, розподілу ролей і механізмів комунікації між

учасниками залежить спроможність реалізувати заплановані роботи у встановлені терміни і в межах бюджету. Pressman підкреслює, що невелика, але добре організована команда з чіткими ролями і ефективною комунікацією є більш продуктивною, ніж велика команда з розмитою відповідальністю і неузгодженими очікуваннями [10]. Для проєкту розробки мобільного CRM-додатку сформована команда з п'яти осіб, кожна з яких виконує чітко визначену роль.

Менеджер проєкту є ключовою фігурою команди і несе відповідальність за загальну координацію робіт, дотримання термінів і бюджету, управління ризиками і комунікацію із замовником. У контексті обраної методології Scrum менеджер проєкту виконує також функції Scrum Master: організовує спринт-планування і ретроспективи, усуває перешкоди для команди і забезпечує дотримання Scrum-практик. Крім того, менеджер проєкту відповідає за ведення реєстру ризиків і актуалізацію плану при відхиленнях від графіку.

Бізнес-аналітик виконав основну частину своєї роботи на аналітичному етапі — визначення вимог, аналіз зацікавлених сторін, моделювання сценаріїв використання. На етапі розробки він залишається в команді як консультант з предметної області, виконує функції Product Owner і відповідає за приймальне тестування реалізованого функціоналу. У разі виникнення питань щодо бізнес-логіки або необхідності уточнення вимог саме бізнес-аналітик є першою точкою контакту між командою розробки і замовником.

Мобільний розробник є основним виконавцем клієнтської частини системи. Він відповідає за реалізацію мобільного додатку на React Native, інтеграцію з серверним API, реалізацію навігаційної структури і забезпечення коректної роботи на платформах iOS і Android. Для даного проєкту виділено одного мобільного розробника, що є прийнятним для MVP, однак є потенційним ризиком у разі виникнення складних технічних проблем — цей ризик врахований у реєстрі ризиків і відпрацьований відповідними стратегіями реагування.

Бекенд-розробник відповідає за реалізацію серверного рівня: розробку REST API, реалізацію бізнес-логіки у відповідних модулях, проєктування і міграцію бази даних, налаштування зовнішніх інтеграцій і розгортання системи на хмарній платформі. Бекенд-розробник також відповідає за безпеку серверного рівня: реалізацію JWT-автентифікації, валідацію вхідних даних і налаштування HTTPS.

UX/UI-дизайнер і тестувальник є суміщеною роллю, що є типовим рішенням для команд малого розміру в умовах обмеженого бюджету. На початку проєкту ця особа відповідає за підготовку макетів усіх екранів у Figma, проєктування навігаційних потоків і узгодження дизайн-системи. На пізніших етапах фокус зміщується на функціональне і регресійне тестування, верифікацію реалізованого функціоналу і документування знайдених дефектів.

На пізніших етапах фокус зміщується на функціональне і регресійне тестування, верифікацію реалізованого функціоналу і документування знайдених дефектів. Морозов В.В. зазначає, що ефективність команди проєкту визначається не лише кваліфікацією окремих учасників, а й якістю розподілу ролей і механізмами комунікації між ними **[52]**.

Зведений склад команди проєкту з розподілом ролей, основних обов'язків та відповідальності кожного учасника наведено у таблиці 3.3.

Таблиця 3.3

Склад команди проєкту та розподіл відповідальності

Роль	Scrum-роль	Основні обов'язки	Зайнятість
Менеджер проєкту	Scrum Master	Планування, координація, ризики, звітність	50%
Бізнес-аналітик	Product Owner	Вимоги, беклог, приймальне тестування	50%
Мобільний розробник	Development Team	React Native, клієнтський додаток, iOS/Android	100%
Бекенд-розробник	Development Team	Node.js, API, БД, інтеграції, деплой	100%

UX/UI-дизайнер / QA	Development Team	Макети Figma, тестування, документація дефектів	100%

Комунікація в команді організована на трьох рівнях. Оперативний рівень — тричі на тиждень проводяться короткі синхронізаційні зустрічі тривалістю до 15 хвилин, де кожен учасник повідомляє про прогрес і потенційні перешкоди. Спринтовий рівень — на початку кожного двотижневого спринту проводиться планування обсягу робіт, наприкінці — демонстрація результатів замовнику (Sprint Review) і ретроспектива (Sprint Retrospective). Проєктний рівень — щомісячний статусний звіт для замовника із відображенням фактичного прогресу відносно плану, витрат бюджету і актуального стану реєстру ризиків. Усі артефакти проєкту зберігаються у спільному репозиторії GitHub і доступні всім учасникам у режимі реального часу.

Схему організаційної структури команди і комунікаційних зв'язків між учасниками наведено на рис. 3.2.

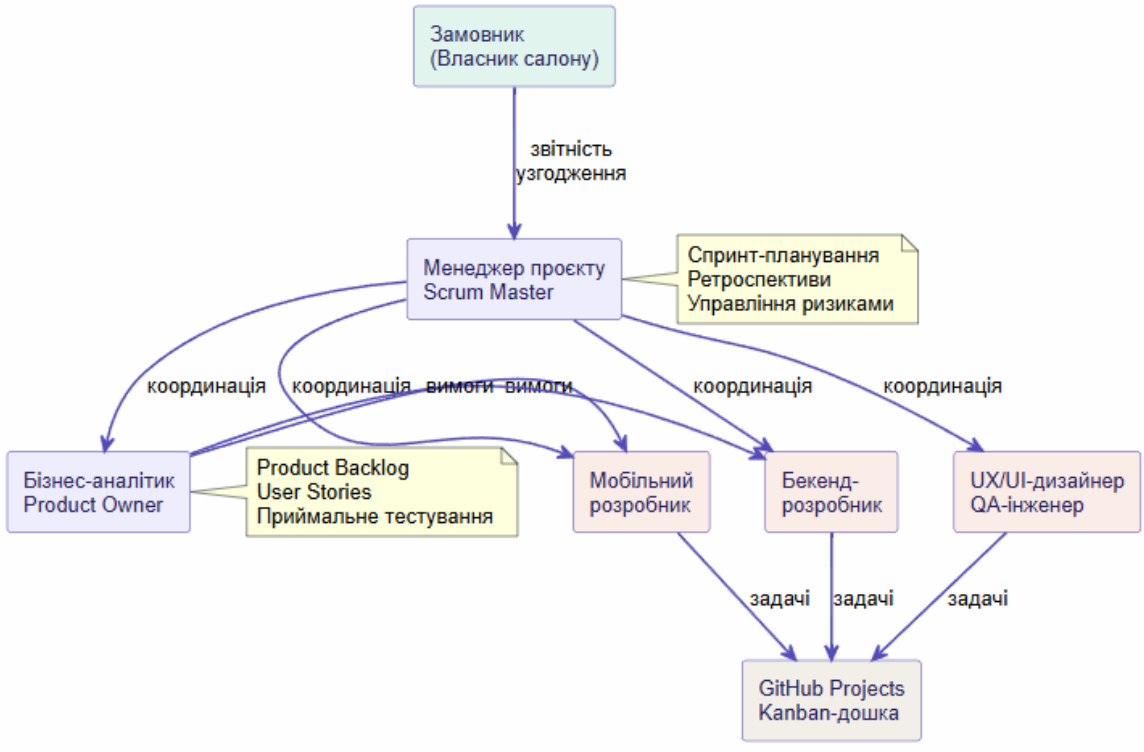


Рис. 3.2 Схема організаційної структури команди проєкту

Схема організаційної структури команди проєкту з відображенням ліній підпорядкування, каналів координації та зон відповідальності кожного

учасника наведена на рис. 3.2. Як видно зі схеми, замовник у ролі власника салону взаємодіє з командою виключно через менеджера проєкту, що забезпечує єдину точку контакту і унеможливлює неузгоджені зміни у вимогах безпосередньо від замовника до розробників. Менеджер проєкту, поєднуючи функції Scrum Master, координує роботу всіх чотирьох виконавців і відповідає за організацію спринт-планування, ретроспектив та управління ризиками. Бізнес-аналітик у ролі Product Owner формує і підтримує Product Backlog, пріоритизує User Stories і виконує приймальне тестування реалізованого функціоналу. Мобільний розробник, бекенд-розробник і UX/UI-дизайнер-тестувальник отримують задачі через GitHub Projects з інтегрованою Kanban-дошкою, що забезпечує прозорість стану робіт для всіх учасників команди без додаткових звітних зустрічей. Така організаційна структура є компактною і водночас достатньою для реалізації проєкту заданого обсягу, а чіткий розподіл відповідальності мінімізує ризик дублювання функцій або безвідповідальних зон.

3.4 Декомпозиція робіт (WBS)

Декомпозиція робіт є фундаментальним інструментом планування, що дозволяє перетворити загальну мету проєкту на ієрархічну структуру конкретних, вимірюваних і керованих пакетів робіт. Без такої декомпозиції оцінка трудовитрат і термінів є лише приблизною, а контроль прогресу — ілюзорним. Відповідно до стандарту PMI, WBS є ієрархічною декомпозицією усього обсягу робіт, що має бути виконаний командою для досягнення цілей проєкту і створення запланованих результатів [41].

Кожен елемент нижнього рівня WBS є пакетом робіт — конкретним завданням із вимірюваним результатом, яке може бути призначене відповідальному виконавцю і оцінене за трудовитратами.

Морозов В.В. підкреслює, що якісно побудована WBS є основою для всіх подальших процесів планування — від оцінки трудовитрат і формування бюджету до побудови календарного плану і розподілу відповідальності між членами команди [51].

WBS проєкту організована у шести фазах, що відповідають логічній послідовності робіт від ініціювання до закриття проєкту. Діаграму WBS наведено на рис. 3.3.

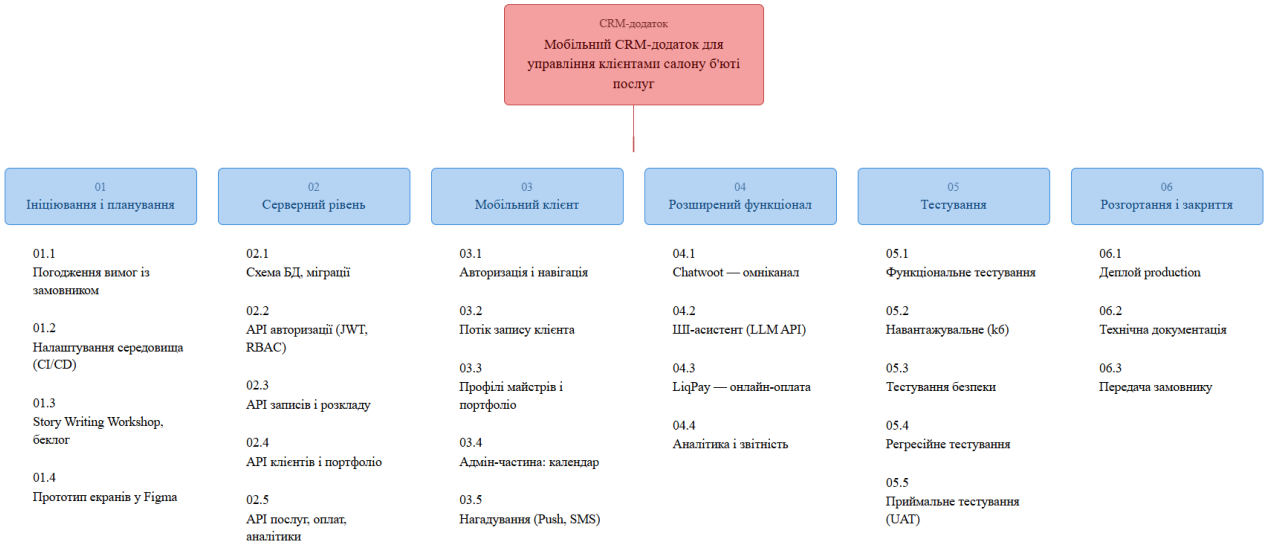


Рис. 3.3 WBS проєкту розробки мобільного CRM-додатку

Фаза 1. Ініціювання і планування (Спринти 1–2, тижні 1–4). Фаза охоплює фіналізацію і узгодження вимог із замовником, підготовку середовища розробки (репозиторій, CI/CD, хмарна інфраструктура), проведення Story Writing Workshop і наповнення Product Backlog з оцінкою Story Points, розробку повного прототипу всіх екранів у Figma і їх узгодження із замовником, а також фіналізацію архітектурних рішень.

Фаза 2. Розробка серверного рівня (Спринти 3–6, тижні 5–12). Фаза охоплює проєктування і реалізацію схеми бази даних PostgreSQL з міграціями, розробку модуля авторизації (реєстрація, JWT, RBAC), розробку модуля управління записами з транзакційною перевіркою доступності слотів, розробку модулів розкладу, клієнтської бази і портфоліо, модулів послуг і оплат, а також базове покриття API юніт-тестами.

Фаза 3. Розробка мобільного клієнта (Спринти 7–12, тижні 13–24). Фаза включає реалізацію навігаційної структури і системи авторизації у React Native, розробку клієнтської частини (головний екран, потік запису, профілі майстрів, портфоліо), розробку адміністративної частини (календар у трьох форматах,

деталі запису, клієнтська база), реалізацію push-нагадувань і інтеграцію з серверним API для всіх реалізованих модулів.

Фаза 4. Розширений функціонал (Спринти 13–16, тижні 25–32). Фаза охоплює інтеграцію омніканального чату через Chatwoot API, реалізацію ШІ-асистента (підключення до LLM API, логіка швидких дій), розробку модуля аналітики і фінансової звітності, інтеграцію платіжної системи LiqPay, налаштування SMS-шлюзу як резервного каналу нагадувань.

Фаза 5. Тестування і виправлення (Спринти 17–22, тижні 33–44). Фаза охоплює функціональне тестування всіх модулів відповідно до вимог, регресійне тестування після виправлення дефектів, навантажувальне тестування і тестування безпеки, тестування на реальних пристроях iOS і Android, а також приймальне тестування із залученням замовника і кінцевих користувачів.

Фаза 6. Розгортання і закриття (Спринти 23–24, тижні 45–52). Завершальна фаза включає розгортання виробничої версії на хмарній платформі, підготовку технічної документації і користувацьких інструкцій, підготовку фінального звіту проєкту, передачу продукту замовнику і офіційне закриття проєкту.

Таблиця 3.4

Зведена WBS із трудовитратами

Фаза	Спринти	Виконавець	Трудовитрати (год)
1.Ініціювання і планування	С1–С2	PM, BA, Designer	320
2. Серверний рівень	С3–С6	Backend	1 280
3. Мобільний клієнт	С7–С12	Mobile	1 920
4.Розширений функціонал	С13–С16	Backend + Mobile	1 280
5. Тестування	С17–С22	QA + BA	1 600
6. Розгортання і закриття	С23–С24	PM + Backend	640
Разом			7 040

Як видно з таблиці 3.4, найбільш трудомісткою фазою є розробка мобільного клієнта — 1 920 годин (27% від загального обсягу), що пояснюється

необхідністю реалізації значної кількості екранів, навігаційних потоків і забезпечення коректної роботи на двох платформах одночасно. Серверний рівень і розширений функціонал мають однаковий обсяг — по 1 280 годин кожен, що відображає рівномірне навантаження на бекенд-розробника протягом більшої частини проєкту. Фаза тестування з обсягом 1 600 годин є свідомо великою: у контексті мобільного продукту, що обробляє персональні дані і платіжні операції, якість і безпека є критичними характеристиками, які не можуть бути забезпечені мінімальним тестуванням. Найменш трудомісткими є фази ініціювання і планування та розгортання і закриття — 320 і 640 годин відповідно, що є типовим розподілом для проєктів із чітко визначеними вимогами і добре структурованою командою. Загальний обсяг трудовитрат у 7 040 годин формує основу для розрахунку бюджетного плану проєкту, який розглядається у підрозділі 3.7.

3.5 Календарний план і діаграма Ганта

Календарний план є інструментом перетворення декомпозованих пакетів робіт у часову послідовність із конкретними датами початку і завершення, відповідальними виконавцями і залежностями між задачами. Він дозволяє визначити критичний шлях проєкту — послідовність задач, затримка будь-якої з яких безпосередньо впливає на дату завершення проєкту в цілому. Стандарт PMI визначає розробку розкладу проєкту як ітеративний процес, що уточнюється в міру надходження нової інформації і реагує на зміни у виконанні [41].

Проєкт розпочинається у жовтні 2025 року і завершується у вересні 2026 року, охоплюючи 52 тижні і 24 двотижневих спринти.

Таблиця 3.5

Календарний план по спринтах

Спринт	Тижні	Основні задачі	Результат
1	2	3	4
С1	1–2	Вимоги, середовище, Story Writing Workshop	Product Backlog, налаштоване середовище

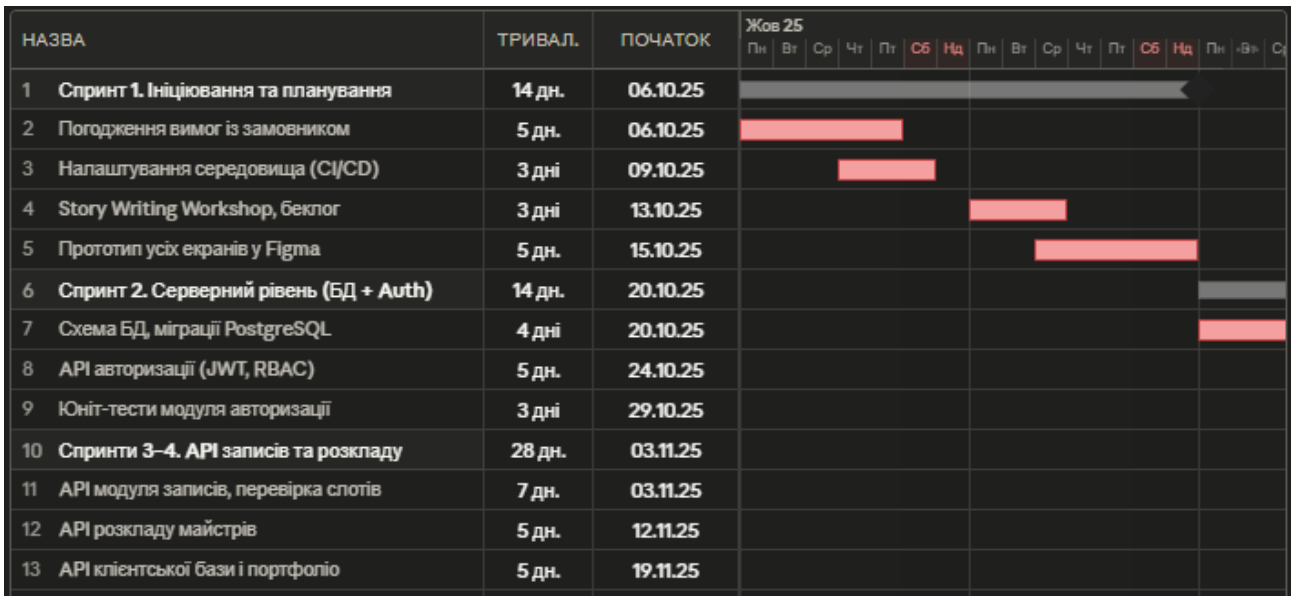
С2	3–4	Макети всіх екранів у Figma, дизайн-система	Узгоджений прототип
С3	5–6	Схема БД, міграції, модуль авторизації	Робочий API авторизації
С4	7–8	API записів, перевірка доступності слотів	Модуль записів

Закінчення таблиці 3.5

1	2	3	4
С5	9–10	API розкладу, клієнтська база	API розкладу і клієнтів
С6	11–12	API послуг, оплат, базова аналітика	Повний серверний API MVP
С7	13–14	React Native: авторизація, навігація	Базова структура додатку
…			
С23	45–48	Розгортання production, документація	Система у production
С24	49–52	Інструкції, фінальний звіт, передача	Проєкт закрито

Повна версія таблиці наведена в додатку Г.

Критичний шлях проєкту проходить через: проєктування схеми БД → розробка API авторизації → розробка API записів → розробка мобільного клієнта (потік запису) → інтеграція клієнта з API → функціональне тестування → розгортання. Будь-яка затримка на цьому шляху безпосередньо переноситиме дату завершення проєкту. Паралельно з серверною розробкою виконується підготовка макетів і розробка окремих компонентів мобільного клієнта, що мають певний резерв часу. Графічне відображення календарного плану у вигляді діаграми Ганта з розбивкою по спринтах і виконавцях наведено на рис. 3.4.



	НАЗВА	ТРИВАЛ.	ПОЧАТОК
1	**Спринт 1. Ініціювання та планування**	14 дн.	06.10.25
2	Погодження вимог із замовником	5 дн.	06.10.25
3	Налаштування середовища (CI/CD)	3 дні	09.10.25
4	Story Writing Workshop, беклог	3 дні	13.10.25
5	Прототип усіх екранів у Figma	5 дн.	15.10.25
6	**Спринт 2. Серверний рівень (БД + Auth)**	14 дн.	20.10.25
7	Схема БД, міграції PostgreSQL	4 дні	20.10.25
8	API авторизації (JWT, RBAC)	5 дн.	24.10.25
9	Юніт-тести модуля авторизації	3 дні	29.10.25
10	**Спринти 3–4. API записів та розкладу**	28 дн.	03.11.25
11	API модуля записів, перевірка слотів	7 дн.	03.11.25
12	API розкладу майстрів	5 дн.	12.11.25
13	API клієнтської бази і портфоліо	5 дн.	19.11.25

Рис. 3.4 Діаграма Ганта проєкту розробки мобільного CRM-додатку

Діаграма Ганта наочно демонструє рівномірне завантаження команди протягом усіх 24 спринтів і відсутність тривалих простоїв у будь-кого з виконавців. Паралельне виконання серверних і клієнтських задач починаючи з третього спринту суттєво скорочує загальну тривалість проєкту порівняно з послідовним підходом. Візуалізація критичного шляху дозволяє менеджеру проєкту зосередити контрольні зусилля на ключових задачах, своєчасно виявляти відхилення від графіку і приймати обґрунтовані рішення про перерозподіл ресурсів у разі виникнення ризикових ситуацій [41].

3.6 Управління ризиками проєкту

Управління ризиками є однією з найважливіших і водночас найбільш складних дисциплін управління проєктом, що дозволяє проактивно виявляти потенційні загрози ще до їхнього настання і формувати стратегії реагування, що мінімізують їхній вплив на результати проєкту. Відповідно до стандарту PMI, управління ризиками включає ідентифікацію ризиків, їхню якісну і кількісну оцінку, планування реагування і моніторинг протягом усього проєктного циклу [41]. Вітчизняні дослідники у галузі класифікації ризиків програмних проєктів підкреслюють, що проєкти з розробки програмного забезпечення мають свою специфіку, пов'язану зі швидкими темпами розробки і численними змінами під час неї, що робить реєстр ризиків принципово важливим інструментом

управління [50]. У контексті Agile-методологій реєстр ризиків переглядається щонайменше раз на спринт і актуалізується відповідно до нових даних [49].

Для систематизації ризиків проєкту використовується багатофакторна класифікація, що враховує специфіку розробки мобільного ПЗ [50]. Виділяються шість категорій ризиків: технічні, управлінські, операційні, юридичні, взаємодії з користувачами і форс-мажорні. Для кількісної оцінки кожен ризик оцінюється за чотирма параметрами: затримки у часі, фінансові витрати, імовірність і частота — у якісній (ЯО) і кількісній (КО) шкалах від 1 до 9. Підсумкова важливість ризику визначається як добуток чотирьох кількісних оцінок. Ризики з важливістю вище 3 000 відносяться до критичних, від 1 000 до 3 000 — до суттєвих, нижче 1 000 — до помірних.

Шкала якісних оцінок: ВВ — дуже висока (9), ВС — висока (8), ВН — вище середнього (7), СС — середня (5), СН — нижче середнього (4), НС — низька (2), НН — дуже низька (1).

Реєстр ідентифікованих ризиків із кількісною оцінкою за чотирма параметрами, визначеною важливістю та категорією кожного ризику наведено у таблиці 3.6.

Таблиця 3.6

Ідентифікація ризиків проєкту

№	Тип ризику	Ризикова подія	Сила впливу	Керованість
1	2	3	4	5
Р01	Технічний	Критичні помилки (баги) у мобільному додатку після релізу	Висока	Висока
Р02	Технічний	Технічні складнощі інтеграції Chatwoot API	Висока	Середня
Р03	Технічний	Низька якість розпізнавання намірів ШІ-асистентом	Середня	Висока
Р04	Технічний	Низька продуктивність додатку на старих смартфонах	Середня	Середня
Р05	Технічний	Конфлікт синхронізації даних між клієнтом і сервером	Висока	Висока

Закінчення таблиці 3.6

1	2	3	4	5
Р29	Форс-мажор	Безпекові ризики (воєнний стан, ракетні обстріли)	Висока	Низька
Р30	Форс-мажор	Різке коливання валютного курсу (вплив на бюджет у гривнях)	Середня	Низька

Повна версія таблиці наведена в додатку Г.

На основі ідентифікованих ризиків проведено їхню кількісну оцінку за чотирма параметрами: затримки у часі, фінансові витрати, імовірність і частота. Підсумкова важливість кожного ризику визначається як добуток чотирьох кількісних оцінок, що дозволяє об'єктивно ранжувати ризики за ступенем загрози для проєкту та пріоритизувати зусилля команди з управління ними. Такий підхід до кількісної оцінки є більш інформативним порівняно зі стандартними матрицями імовірність-вплив, оскільки враховує не лише ймовірність настання ризику та силу його впливу, а й часовий і фінансовий виміри окремо, що дає змогу диференціювати ризики, які є критичними за вартістю, від тих, що є критичними за часом.

Аналіз результатів оцінки виявляє закономірний розподіл: найвищу важливість отримали ризики форс-мажорного характеру, зокрема безпекові ризики, пов'язані з воєнним станом (важливість 6 561), що відображає реалії поточного операційного середовища. Серед технічних ризиків найбільш критичним є виникнення помилок після релізу (важливість 4 096), оскільки такий ризик є одночасно ймовірним, фінансово витратним і важко контрольованим після виходу продукту у виробництво. Ризики середнього рівня важливості, зокрема конфлікт синхронізації даних (1 600) і складнощі інтеграції Chatwoot (1 400), потребують проактивного моніторингу, але не є критичними за умови своєчасного виявлення. Повна версія таблиці з усіма тридцятьма ризиками та триступеневими протиризиковими заходами наведена у додатку Д. Результати кількісної оцінки та ранжування наведено у таблиці 3.7.

Таблиця 3.7

Кількісна оцінка та ранжування ризиків

№	Ризикова подія	Затримки		Фін. витрати		Імовірність		Частота		Важливість
		ЯО	КО	ЯО	КО	ЯО	КО	ЯО	КО	
Р01	Критичні баги після релізу	ВС	8	ВС	8	ВС	8	ВС	8	4096
Р02	Складнощі інтеграції Chatwoot	ВС	8	СС	5	ВН	7	СС	5	1400
Р03	Якість ШІ-асистента	СН	4	СН	4	ВН	7	СН	4	448
Р04	Продуктивність на старих пристроях	СС	5	СН	4	ВН	7	СС	5	700
Р05	Конфлікт синхронізації даних	ВС	8	ВС	8	СС	5	СС	5	1600
…										
Р29	Безпекові ризики (война)	ВВ	9	ВВ	9	ВВ	9	ВВ	9	6561
Р30	Коливання валютного курсу	СН	4	СС	5	ВС	8	СС	5	800

Повна версія таблиці наведена в додатку Г.

Для кожного критичного і суттєвого ризику розроблено триступеневу систему реагування: превентивні заходи до настання ризику, оперативні дії у разі його виявлення та заходи з мінімізації наслідків. Превентивні заходи є пріоритетними, оскільки усунення причини ризику завжди є менш витратним, ніж подолання його наслідків. Оперативні заходи визначають конкретні дії команди у момент виявлення відхилення — тригери, відповідальних і терміни реагування. Заходи з мінімізації наслідків є резервним сценарієм на випадок, якщо ризик реалізувався повністю і потребує антикризового управління.

Відповідно до рекомендацій PMI, план реагування є живим документом і переглядається щоспринту на ретроспективній зустрічі команди [41]. План реагування для ключових ризиків наведено у таблиці 3.8.

Таблиця 3.8

Протиризикові заходи (план реагування)

№	Ризикова подія	ПРЗ 1: Профілактика	Симптом (рання ознака)	ПРЗ 2: При симптомах	ПРЗ 3: При проблемі
1	2	3	4	5	6
P01	Критичні баги після релізу	Code review кожного PR; покриття тестами >80%; регресійне тестування перед релізом	Зростання кількості дефектів у спринті; сповіщення від моніторингу Sentry	Залучення другого розробника до дебагу; пріоритизація виправлення P0-дефектів	Відкат до попередньої стабільної версії; екстрений хотфікс; сповіщення користувачів
P02	Складнощі інтеграції Chatwoot	Тестування API Chatwoot на етапі планування; розробка mock-об'єктів для незалежної розробки	Відсутність коректної відповіді API понад 48 год; помилки у webhook	Перехід на альтернативний агрегатор (Respond.io); збільшення часового буфера спринту	Реалізація спрощеної версії inbox без агрегатора; відтермінування функції до наступного релізу
P03	Якість ШІ-асистента	Тестування промптів на реальних прикладах звернень; логування всіх запитів до LLM	Точність розпізнавання нижче 80% за результатами логів	Ітеративне налаштування промптів; звуження переліку підтримуваних намірів	Відключення ШІ-підказок; перехід до повністю ручної обробки звернень адміністратором

Закінчення таблиці 3.8

1	2	3	4	5	6
Р04	Продуктивність на старих пристроях	Тестування на пристроях з характеристиками нижче середніх; профілювання продуктивності	Час завантаження екранів >3 сек на тестових пристроях	Оптимізація рендерингу компонентів; ліниве завантаження зображень	Встановлення мінімальних системних вимог; виключення підтримки пристроїв нижче порогу
…					
Р29	Безпекові ризики (война)	Дистанційна робота; резервні сервери за кордоном; резервні копії поза Україною	Оголошення повітряних тривог; евакуаційні попередження	Перехід в укриття; зупинка синхронних нарад; асинхронна комунікація	Призупинення проєкту; передача критичних задач учасникам за кордоном; активація плану безперервності
Р30	Коливання валютного курсу	Фіксація вартості API-послуг у гривнях у договорах; резервний фонд	Курс долара зростає понад 10% відносно планового	Перегляд бюджету на зовнішні сервіси; оптимізація витрат	Переговори з замовником щодо коригування бюджету; скорочення використання платних API

Повна версія таблиці наведена в додатку Г.

Реєстр ризиків є живим документом і переглядається менеджером проєкту на кожному спринт-плануванні. Нові ризики додаються до реєстру в міру їхнього виявлення, а статус існуючих актуалізується відповідно до

фактичного стану проєкту. Критичні ризики (Р29, Р01, Р11, Р13) потребують щотижневого моніторингу і є постійним пунктом порядку денного синхронізаційних зустрічей команди. Запропонована класифікація ризиків враховує часові, бюджетні і якісні виміри проєкту, що дозволяє менеджеру проєкту системно підходити до управління загрозами і приймати обґрунтовані рішення щодо реагування [50]. Формалізований реєстр ризиків із протиризиковими заходами трьох рівнів забезпечує команді чіткий алгоритм дій у будь-якій ризиковій ситуації — від профілактики до кризового реагування.

3.7 Бюджетний план проєкту

Бюджетний план є фінансовим вираженням проєктного плану і відображає сукупні витрати на реалізацію проєкту у розрізі категорій витрат і часових фаз. Відповідно до стандарту PMI, бюджет проєкту формується шляхом агрегування оцінок вартості окремих пакетів робіт і включає резерв на непередбачені витрати [41]. Для даного проєкту бюджет складається з двох основних категорій: витрати на оплату праці команди та операційні й інфраструктурні витрати.

Витрати на оплату праці є найбільшою статтею бюджету і формуються виходячи з трудовитрат кожного учасника команди і відповідних середньомісячних ставок відповідно до середньоринкових показників українського ринку IT-фахівців станом на 2025 рік.

Розрахунок фонду оплати праці в розрізі ролей команди наведено у таблиці 3.9.

Таблиця 3.9

Розрахунок фонду оплати праці

Роль	Середня ЗП (грн/міс)	Формула розрахунку	Разом (грн)
Менеджер проєкту	42 500	42 500 × 0,5 × 12	255 000
Бізнес-аналітик / PO	38 000	38 000 × 0,5 × 12	228 000
Мобільний розробник	55 000	55 000 × 1,0 × 12	660 000
Бекенд-розробник	55 000	55 000 × 1,0 × 12	660 000

UX/UI-дизайнер / QA	38 000	38 000 × 1,0 × 12	456 000
РАЗОМ			2 259 000

Як видно з таблиці 3.9, найбільші витрати припадають на мобільного і бекенд-розробників — по 660 000 грн кожен, що зумовлено їхньою повною зайнятістю протягом усіх 12 місяців проєкту. Менеджер проєкту і бізнес-аналітик залучені на половину ставки, що відображає реальний розподіл навантаження: після завершення аналітичного етапу їхня участь переходить у режим координації і супроводу. Загальний фонд оплати праці становить 2 259 000 грн і формує основу бюджету проєкту.

Другою складовою бюджету є операційні та інфраструктурні витрати, що охоплюють хмарний хостинг, сховище даних, ліцензії на програмне забезпечення, підключення зовнішніх сервісів та публікацію додатку. Детальний розподіл операційних витрат наведено у таблиці 3.10.

Таблиця 3.10

Операційні та інфраструктурні витрати

Витрата	Опис	Вартість (грн/рік)
Хмарний хостинг	Railway/Render — серверний рівень	9 600
Хмарне сховище	Cloudflare R2 — фото портфоліо майстрів	4 800
LLM API	Claude API / OpenAI для ШІ-асистента	29 000
SMS-шлюз	Twilio — резервні SMS-нагадування	8 700
Домен та SSL	Річна оренда домену та сертифікат безпеки	2 500
Ліцензії ПЗ	Figma Professional, GitHub Pro, Jira	18 000
Apple Developer Program	Річна підписка для публікації iOS-додатку	4 100
РАЗОМ		76 700

Резервний фонд встановлюється на рівні 10% від суми основних витрат — 233 570 грн. Такий рівень резерву є стандартною практикою для проєктів із помірним рівнем ризику [41]. Зведений бюджетний план із розбивкою за статтями витрат і часткою кожної з них у загальному бюджеті наведено у таблиці 3.11.

Таблиця 3.11

Зведений бюджетний план проєкту

Стаття витрат	Сума (грн)	Частка
Фонд оплати праці	2 259 000	87,9%
Операційні та інфраструктурні витрати	76 700	3,0%
Всього основних витрат	2 335 700	90,9%
Резервний фонд (10%)	233 570	9,1%
ЗАГАЛЬНИЙ БЮДЖЕТ ПРОЄКТУ	2 569 270	100%

Як видно з таблиці 3.11, домінуючою статтею бюджету є фонд оплати праці — 2 259 000 грн, що становить 87,9% від загального бюджету проєкту. Це є характерним розподілом для IT-проєктів, де основним ресурсом є людський капітал, а не матеріальні активи. Операційні та інфраструктурні витрати складають лише 3,0% (76 700 грн), що пояснюється використанням хмарних сервісів з оплатою за фактичним споживанням замість придбання власної інфраструктури — підхід, який суттєво знижує капітальні витрати на старті проєкту і забезпечує гнучкість масштабування. Резервний фонд у розмірі 9,1% (233 570 грн) відповідає рекомендованому діапазону для проєктів із помірним рівнем ризику відповідно до стандарту PMI [41]. Загальний бюджет проєкту становить 2 569 270 грн і є реалістичною оцінкою вартості розробки мобільного CRM-додатку командою з п'яти осіб протягом 12 місяців у поточних ринкових умовах українського IT-ринку. Затверджений бюджетний план є базовим орієнтиром для контролю витрат протягом усього проєктного циклу і підлягає перегляду у разі суттєвих відхилень від графіку або реалізації критичних ризиків.

3.8 Ключові показники ефективності (KPI)

Ключові показники ефективності є інструментом вимірюваної оцінки успішності проєкту і якості кінцевого продукту. Їхнє визначення на етапі планування є принципово важливим, оскільки без заздалегідь встановлених

критеріїв успіху оцінка результатів проєкту після його завершення є суб'єктивною і суперечливою. Laudon та Laudon зазначають, що вимірювання ефективності інформаційних систем повинне охоплювати як технічні показники, так і бізнесові результати, оскільки технічно коректна система, що не приносить бізнесової цінності, не може вважатися успішною [1].

KPI поділяються на дві групи: показники ефективності процесу розробки і показники якості продукту. Показники ефективності процесу охоплюють дотримання термінів спринтів (відхилення не більше 10% від запланованого velocity), дотримання бюджету (фактичні витрати не перевищують плановий бюджет більш ніж на 10%), стабільне або зростаюче значення velocity команди від спринту до спринту, а також кількість дефектів, перенесених у наступний спринт — не більше 20% від загальної кількості виявлених.

Показники якості продукту охоплюють реалізацію 100% функцій категорії «Must have» за результатами приймального тестування, час відгуку API не більше 2 секунд за нормального навантаження, підтримку одночасного навантаження не менше 50 активних користувачів, доступність системи після розгортання не нижче 99%, відсутність критичних дефектів у версії, переданій замовнику, частку онлайн-записів серед тестових користувачів не менше 70% протягом першого місяця, а також оцінку зручності інтерфейсу за шкалою SUS не нижче 70 балів зі 100.

Визначені показники є конкретними, вимірюваними і прив'язаними до часових меж, що відповідає принципу SMART у формулюванні цілей і критеріїв оцінки [10]. Кожен KPI має чітко визначений цільовий поріг, метод вимірювання і момент перевірки — після кожного спринту для процесних показників і після завершення приймального тестування для продуктових. Така структура дозволяє менеджеру проєкту оперативно виявляти відхилення і приймати коригувальні рішення ще до завершення проєкту, а не лише констатувати результат після його закриття.

Такий підхід до планування бюджету відповідає рекомендаціям вітчизняних дослідників у галузі управління IT-проєктами щодо структурування витрат у розрізі фаз і ролей команди **[51]**.

Зведений перелік KPI з цільовими значеннями, методами вимірювання та відповідальними особами наведено у таблиці 3.12.

Таблиця 3.12

Ключові показники ефективності проєкту та продукту

Показник	Ціль	Метод вимірювання
Дотримання термінів спринтів	Відхилення ≤10%	Burndown Chart, порівняння velocity
Дотримання бюджету	Перевищення ≤10%	Щотижневий облік витрат
Реалізація Must have вимог	100%	Результати приймального тестування
Час відгуку API	≤2 сек	Навантажувальне тестування (k6)
Одночасне навантаження	≥50 користувачів	Навантажувальне тестування
Доступність системи	≥99%	Моніторинг uptime (UptimeRobot)
Критичні дефекти при передачі	0	Звіт QA
Частка онлайн-записів	≥70% за місяць	Аналітика додатку
Оцінка зручності (SUS)	≥70 балів	Опитування тестових користувачів

Регулярний моніторинг визначених показників здійснюється менеджером проєкту і відображається у щотижневих статусних звітах. У разі відхилення будь-якого показника за встановлені межі ініціюється коригувальна дія: перерозподіл задач у поточному спринті, залучення резервного часу або перегляд пріоритетів Product Backlog.

Таким чином система KPI виконує не лише вимірювальну, а й управлінську функцію — вона є інструментом раннього виявлення відхилень і оперативного реагування на них.

РОЗДІЛ 4. РОЗРОБКА ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО

4.1 Концептуальна архітектура мобільного додатку

Концептуальна архітектура є першим і найбільш визначальним технічним рішенням у процесі розробки програмного продукту. Вона встановлює принципи організації системи, визначає її основні структурні компоненти та способи їхньої взаємодії ще до того, як починається написання коду або проєктування бази даних. Правильно обрана архітектура забезпечує масштабованість, підтримуваність і надійність системи протягом усього її життєвого циклу, тоді як архітектурні помилки, допущені на ранніх етапах, мають тенденцію багатократно збільшуватися у вартості виправлення на пізніших стадіях. Fowler зазначає, що архітектурні рішення є найважчими для зміни після початку реалізації, тому їхнє обдумування і обґрунтування є одним із найціннішим вкладів у проєкт [19]. Bass, Clements і Kazman підкреслюють, що архітектура системи безпосередньо визначає її якісні характеристики — продуктивність, безпеку, надійність і модифікованість — і жодна з цих характеристик не може бути забезпечена після завершення розробки, якщо вона не закладена в архітектуру з самого початку [32].

Для розроблюваного мобільного CRM-додатку обрано клієнт-серверну архітектуру з чітким розподілом на три рівні: клієнтський рівень (мобільний додаток), серверний рівень (бекенд з бізнес-логікою) та рівень даних (база даних). Така трирівнева архітектура є стандартним підходом для мобільних сервісних додатків, оскільки забезпечує незалежність рівнів між собою, що дозволяє змінювати або масштабувати кожен із них без перебудови решти системи [8]. Взаємодія між клієнтським і серверним рівнями реалізується через REST API — набір уніфікованих ендпоінтів, що обробляють HTTP-запити і повертають відповіді у форматі JSON. Вибір REST API зумовлений його простотою, широкою підтримкою у мобільних фреймворках і зрілістю екосистеми інструментів для розробки і тестування [33].

Клієнтський рівень реалізує інтерфейс взаємодії для всіх чотирьох ролей користувачів в межах єдиного мобільного застосунку. Диференціація інтерфейсу між ролями досягається не через окремі додатки, а через механізм умовного рендерингу: після авторизації система визначає роль користувача і відображає відповідний набір екранів і функцій. Це спрощує підтримку і оновлення продукту, оскільки зміни в бізнес-логіці або дизайні застосовуються одноразово для всіх ролей, а не окремо для кожної версії. Клієнтський рівень є stateless у тому сенсі, що він не зберігає бізнес-дані локально — всі операції з даними здійснюються через запити до серверного рівня. Локально кешуються лише дані сесії користувача і незначний обсяг допоміжної інформації для прискорення роботи інтерфейсу.

Серверний рівень є ядром системи з точки зору бізнес-логіки. Він отримує запити від клієнтського додатку, виконує їхню валідацію і авторизацію, реалізує бізнес-правила і звертається до бази даних або зовнішніх сервісів для виконання операцій. Серверний рівень організований за принципом модульності: кожен функціональний домен системи — авторизація, записи, розклад, клієнти, послуги, оплати, нагадування, аналітика — реалізований як окремий модуль із власною логікою і незалежним набором ендпоінтів. Такий поділ спрощує тестування, локалізацію помилок і подальше розширення системи: додавання нового модуля не вимагає змін у існуючих [34]. Серверний рівень також відповідає за автоматизовані процеси, що виконуються за розкладом без безпосередньої ініціації користувачем — зокрема за відправку нагадувань клієнтам у визначений час до запланованого візиту.

Рівень даних реалізований на основі реляційної бази даних, що забезпечує структурованість інформації, цілісність зв'язків між сутностями і ефективність виконання складних запитів. Вибір реляційної моделі зумовлений характером даних системи: записи, клієнти, майстри, послуги і оплати є чітко визначеними сутностями з фіксованими зв'язками між ними, що є класичним сценарієм використання реляційних СУБД [28]. Доступ до бази даних здійснюється виключно через серверний рівень — клієнтський додаток не має

прямого з'єднання з базою, що є важливою архітектурною вимогою з точки зору безпеки.

Зовнішні сервіси інтегруються в систему через серверний рівень за допомогою відповідних API. До зовнішніх сервісів належать: сервіс push-сповіщень (Firebase Cloud Messaging) для доставки нагадувань на мобільні пристрої, SMS-шлюз для альтернативного каналу нагадувань і платіжна система для обробки онлайн-транзакцій. Інтеграція зовнішніх сервісів через серверний рівень, а не безпосередньо з клієнтського додатку, є архітектурним рішенням, що забезпечує безпеку ключів API і гнучкість у заміні провайдера без змін у клієнтській частині.

Безпека є наскрізним архітектурним принципом, що реалізується на всіх трьох рівнях. На клієнтському рівні — через безпечне зберігання токенів сесії і відсутність локального зберігання чутливих даних. На серверному рівні — через перевірку JWT-токена у кожному запиті, валідацію вхідних даних і застосування принципу мінімальних привілеїв при доступі до даних. На рівні даних — через шифрування чутливих полів і механізми резервного копіювання. Увесь трафік між клієнтом і сервером передається по захищеному каналу HTTPS [13].

Масштабованість архітектури забезпечується кількома механізмами. По-горизонталі — можливістю запуску кількох екземплярів серверного рівня за балансувальником навантаження при зростанні кількості користувачів. По-вертикалі — можливістю збільшення ресурсів окремих компонентів без змін у архітектурі. Модульність серверного рівня дозволяє у майбутньому перейти до мікросервісної архітектури, виокремивши найбільш навантажені модулі в окремі сервіси, якщо це стане необхідним [34].

Таким чином, обрана трирівнева клієнт-серверна архітектура з REST API, модульним серверним рівнем і реляційною базою даних відповідає вимогам масштабованості, безпеки і підтримуваності системи, визначеним у другому розділі (рис. 4.1). Вона створює структурну основу для проєктування бази даних і функціональних модулів, що є предметом наступних підрозділів.

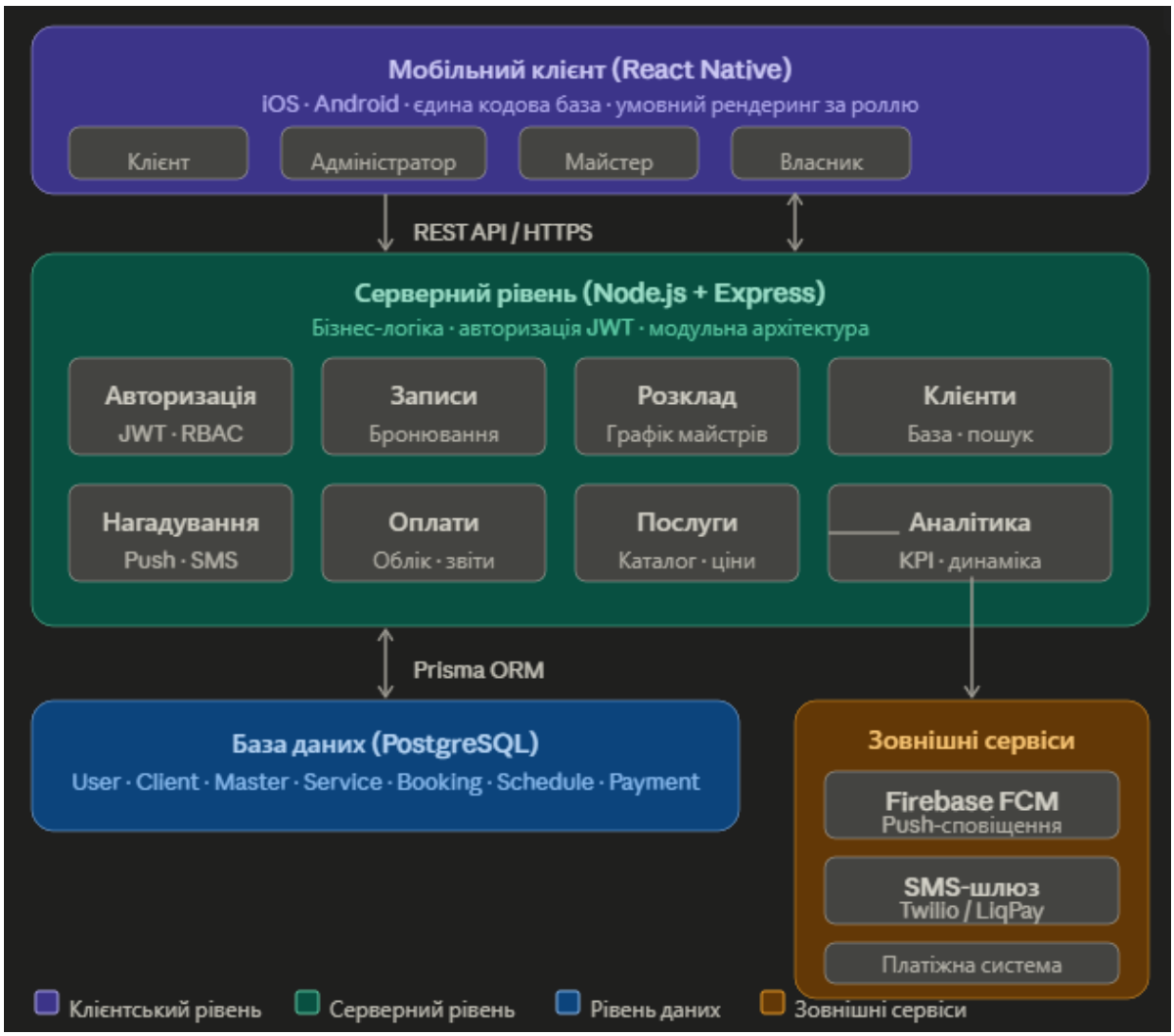


Рис. 3.1 Концептуальна архітектура мобільного CRM-додатку

Таким чином, обрана трирівнева клієнт-серверна архітектура з REST API, модульним серверним рівнем і реляційною базою даних відповідає вимогам масштабованості, безпеки і підтримуваності системи, визначеним у другому розділі. Вона створює структурну основу для проєктування бази даних і функціональних модулів, що є предметом наступних підрозділів.

4.2 Проєктування структури бази даних

Проєктування структури бази даних є визначальним технічним кроком, що безпосередньо забезпечує реалізацію більшості функціональних вимог системи. База даних є не просто сховищем інформації — вона реалізує модель предметної області у формалізованому вигляді, відображаючи сутності реального світу, їхні атрибути і зв'язки між ними. Якість проєктування БД визначає не лише коректність збереження даних, а й продуктивність запитів,

цілісність інформації і складність підтримки системи у майбутньому. Elmasri і Navathe зазначають, що добре спроєктована база даних є фундаментом надійної інформаційної системи і що помилки на рівні логічної моделі даних є найбільш дорогими у виправленні після початку експлуатації [28].

Проєктування здійснюється у два етапи. Перший — концептуальне проєктування, на якому визначаються сутності і зв'язки між ними незалежно від конкретної СУБД. Другий — логічне проєктування, на якому концептуальна модель трансформується у конкретні таблиці, атрибути і обмеження цілісності. У межах даної роботи розглядаються обидва етапи, оскільки їх поєднання дає найповніше уявлення про структуру системи [29].

Аналіз предметної області і визначених функціональних вимог дозволяє виділити сім основних сутностей системи: Користувач (User), Клієнт (Client), Майстер (Master), Послуга (Service), Запис (Booking), Розклад (Schedule) та Оплата (Payment). Кожна з цих сутностей відповідає конкретному об'єкту предметної області і реалізує певну групу функціональних вимог.

Сутність User є центральною з точки зору безпеки і контролю доступу. Вона зберігає облікові дані всіх користувачів системи незалежно від їхньої ролі, що дозволяє реалізувати єдиний механізм авторизації. Роль користувача визначається атрибутом role, який може набувати одного з чотирьох значень: client, master, admin, owner. Саме значення цього атрибута визначає, які функції і дані будуть доступні користувачу після авторизації. Зберігання паролів здійснюється виключно у хешованому вигляді з використанням алгоритму bcrypt — оригінальний пароль ніколи не зберігається в базі даних [13].

Сутність Client зберігає інформацію про відвідувачів салону — ім'я, контактний телефон, email і нотатки майстра. Вона пов'язана із сутністю User відношенням один-до-одного: кожен клієнт має власний обліковий запис, однак клієнтський профіль містить додаткову бізнес-інформацію, що не є частиною облікових даних. Такий поділ між обліковими і профільними даними є стандартною практикою проєктування, що дозволяє незалежно управляти безпекою і бізнес-інформацією [30].

Сутність Master аналогічно пов'язана з User відношенням один-до-одного і зберігає професійні характеристики майстра: спеціалізацію, опис і статус активності. Статус активності дозволяє тимчасово відключати майстра від системи запису без видалення його профілю і пов'язаної з ним історії.

Сутність Service описує перелік послуг, доступних у салоні. Кожна послуга має назву, опис, тривалість у хвилинах і вартість. Тривалість є критичним атрибутом з точки зору управління розкладом: саме вона визначає, скільки часового слота займає кожна процедура і, відповідно, коли стає доступним наступний вільний час майстра. Silberschatz, Korth і Sudarshan підкреслюють важливість коректного вибору типів даних для числових атрибутів: для зберігання вартості послуг слід використовувати тип DECIMAL(10,2), а не FLOAT, оскільки останній може вносити похибку при фінансових розрахунках [30].

Сутність Booking є центральною в системі і реалізує ключовий бізнес-процес — управління записами. Вона пов'язує клієнта, майстра і послугу, фіксуючи конкретну дату, час початку і статус запису. Статус запису реалізований як перелічуваний тип (enum) з чотирма значеннями: pending (очікує підтвердження), confirmed (підтверджений), completed (виконаний), cancelled (скасований). Цей атрибут дозволяє відстежувати повний життєвий цикл запису і є основою для аналітики — зокрема для аналізу причин і частоти скасувань. Час завершення запису є обчислювальним значенням і визначається як сума часу початку і тривалості послуги, що унеможливлює некоректне накладання записів при перевірці доступності [28].

Сутність Schedule визначає робочий графік кожного майстра на рівні окремих часових блоків. Вона зберігає дату, час початку і завершення робочого блоку, а також ознаку доступності. Це дозволяє гнучко управляти графіком: фіксувати стандартні робочі години, перерви, відпустки і нестандартні зміни без зміни структури бази даних. При перевірці доступності слота для нового запису система звертається до цієї сутності, щоб переконатися, що запитуваний

час потрапляє у робочий блок майстра і не перетинається з існуючими записами.

Сутність Payment фіксує факт оплати за конкретний запис. Вона пов'язана з Booking відношенням один-до-одного і зберігає суму, форму розрахунку (готівка або безготівковий платіж) і дату транзакції. Наявність окремої сутності для оплат, а не простого атрибута у Booking, дозволяє у майбутньому розширити функціонал — наприклад, підтримку часткової оплати або повернення коштів — без зміни структури основної таблиці записів [31].

Аналіз предметної області і визначених функціональних вимог дозволяє виділити сім основних сутностей системи: Користувач (User), Клієнт (Client), Майстер (Master), Послуга (Service), Запис (Booking), Розклад (Schedule) та Оплата (Payment). Кожна з цих сутностей відповідає конкретному об'єкту предметної області і реалізує певну групу функціональних вимог. Основні сутності з їхніми ключовими атрибутами та призначенням наведено у таблиці 3.1.

Таблиця 3.1

Основні сутності бази даних та їхні ключові атрибути

Сутність	Ключові атрибути	Призначення
User	id, email, password_hash, role, created_at	Облікові дані і роль користувача
Client	id, user_id, name, phone, notes	Профіль клієнта салону
Master	id, user_id, name, specialization, is_active	Профіль майстра
Service	id, name, duration_min, price, description	Каталог послуг і процедур
Booking	id, client_id, master_id, service_id, date, start_time, status	Запис клієнта на послугу
Schedule	id, master_id, date, start_time, end_time, is_available	Робочий графік майстра
Payment	id, booking_id, amount, method, paid_at	Облік оплат за послуги

Portfolio	id, master_id, photo_url, category, description, is_public, created_at	Портфоліо робіт майстра

Зв'язки між сутностями відображають бізнес-логіку системи і реалізуються через зовнішні ключі з відповідними обмеженнями цілісності. Основним типом зв'язку є один-до-багатьох: один клієнт може мати багато записів, один майстер — багато записів і багато блоків у розкладі, одна послуга — багато записів. Зв'язок між Booking і Payment є один-до-одного, оскільки кожен запис може мати не більше однієї оплати у базовій версії системи. Зовнішні ключі налаштовані з правилом CASCADE для операцій оновлення і RESTRICT для операцій видалення, що забезпечує цілісність даних при будь-яких змінах [28]. Повний перелік зв'язків між сутностями з визначенням типу відношення та зовнішніх ключів наведено у таблиці 3.2.

Таблиця 3.2

Основні зв'язки між сутностями

Сутність 1	Сутність 2	Тип зв'язку	Зовнішній ключ
User	Client	1:1	client.user_id → user.id
User	Master	1:1	master.user_id → user.id
Client	Booking	1:N	booking.client_id → client.id
Master	Booking	1:N	booking.master_id → master.id
Service	Booking	1:N	booking.service_id → service.id
Master	Schedule	1:N	schedule.master_id → master.id
Booking	Payment	1:1	payment.booking_id → booking.id
Master	Portfolio	1:N	portfolio.master_id → master.id

Нормалізація структури здійснена до третьої нормальної форми (3NF), що означає відсутність транзитивних залежностей між неключовими атрибутами. Зокрема, вартість послуги зберігається у сутності Service, а не дублюється у Booking, що забезпечує консистентність даних при зміні цін — при цьому фактична вартість проведеної процедури фіксується в Payment у

момент оплати, що дозволяє відстежувати історичні ціни. Date і час також зберігаються у Booking у нормалізованому вигляді, а не як один текстовий рядок, що забезпечує можливість ефективних запитів за часовими діапазонами [29].

Для забезпечення продуктивності системи на найбільш критичних запитах створюються індекси. Індекс на поле date у таблиці Booking прискорює вибірку записів за конкретну дату при формуванні розкладу. Індекс на master_id у таблицях Booking і Schedule прискорює запити на перевірку доступності майстра. Складений індекс на (master_id, date) у таблиці Schedule є найбільш критичним для операції перевірки вільних слотів, яка виконується при кожному новому бронюванні. Connolly і Begg зазначають, що правильно підібрані індекси можуть підвищити швидкість виконання читаючих запитів на один-два порядки, водночас незначно сповільнюючи операції запису, що є прийнятним компромісом для систем із переважанням операцій читання [31]. Детальну структуру таблиці Booking з переліком полів, типів даних, обмежень та індексів як найбільш показового прикладу наведено у таблиці 3.3.

Таблиця 3.3

Приклад детальної структури таблиці Booking

Атрибут	Тип	Обмеження	Опис
id	INT	PRIMARY KEY, AUTO_INCREMENT	Унікальний ідентифікатор
client_id	INT	FOREIGN KEY, NOT NULL	Посилання на клієнта
master_id	INT	FOREIGN KEY, NOT NULL	Посилання на майстра
service_id	INT	FOREIGN KEY, NOT NULL	Посилання на послугу
date	DATE	NOT NULL	Дата запису
start_time	TIME	NOT NULL	Час початку
status	ENUM	NOT NULL, DEFAULT 'pending'	Стан запису
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Дата створення запису
updated_at	TIMESTAMP	ON UPDATE CURRENT_TIMESTAMP	Дата останнього оновлення

Таким чином, спроєктована структура бази даних повністю охоплює всі функціональні вимоги системи, визначені у другому розділі, забезпечує цілісність і консистентність даних через механізми зовнішніх ключів і обмежень, оптимізована для продуктивності на найбільш критичних запитах і відкрита для розширення без необхідності суттєвої перебудови при додаванні нових функцій у наступних версіях продукту. Графічне відображення структури бази даних у вигляді ER-діаграми з усіма сутностями, атрибутами і зв'язками між ними наведено на рис. 3.2.

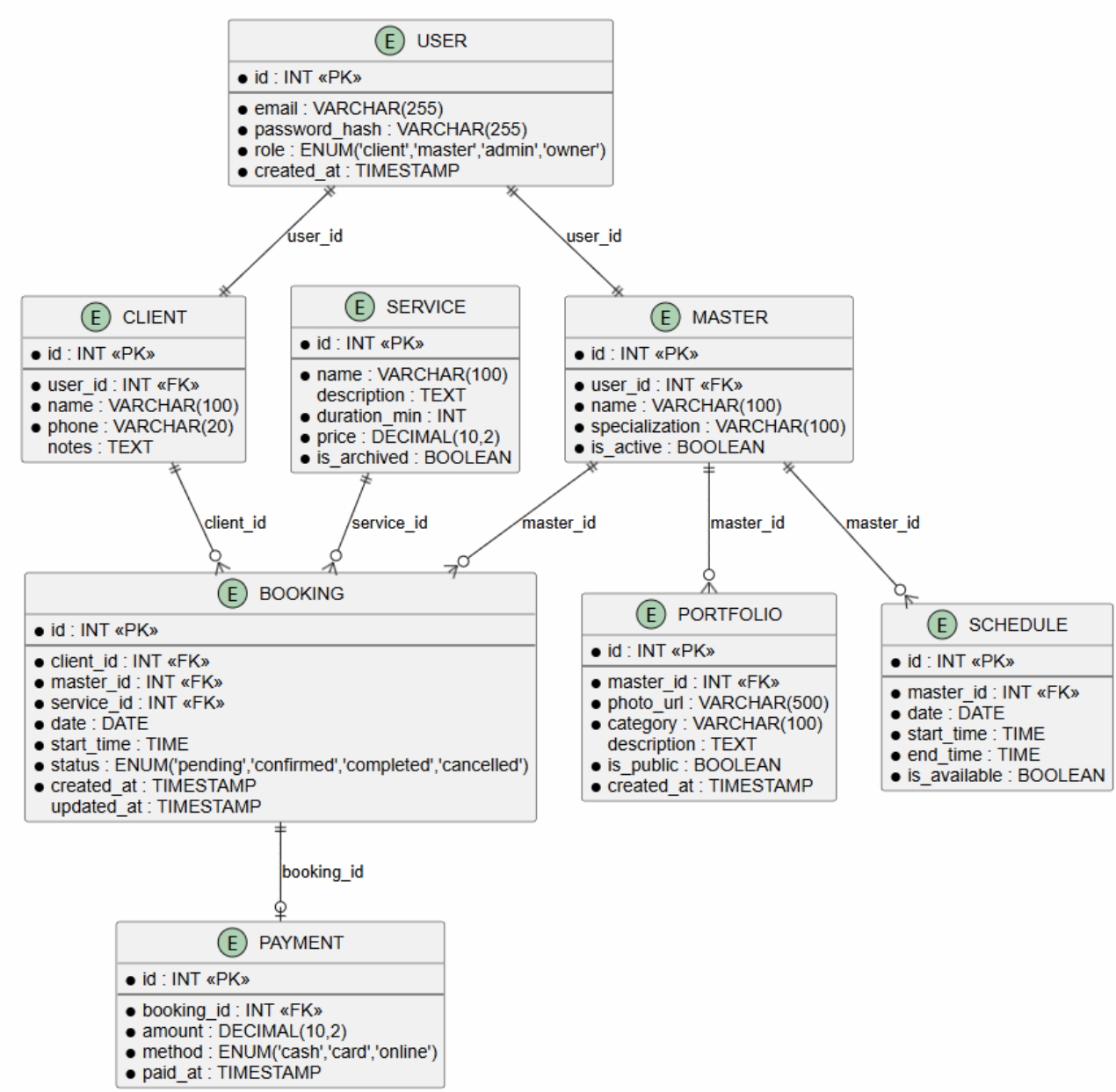


Рис. 3.2 ER-діаграма бази даних мобільного CRM-додатку

ER-діаграма наочно демонструє центральну роль сутності Booking у структурі бази даних: вона є точкою перетину між клієнтом, майстром, послугою і оплатою, що відображає ключовий бізнес-процес системи — управління записами. Сутності User і Client, пов'язані відношенням один-до-одного, утворюють дворівневу модель ідентифікації користувача, яка розділяє облікові і профільні дані. Сутність Schedule, пов'язана з Master відношенням один-до-багатьох, забезпечує гнучке управління робочим графіком без прив'язки до конкретних записів, що дозволяє незалежно налаштовувати доступність майстра. Така архітектура даних є логічним фундаментом для реалізації функціональних модулів системи, що розглядається у наступному підрозділі.

4.3 Розробка функціональних модулів мобільного додатку

Функціональні модулі є основними будівельними блоками програмної системи, кожен з яких реалізує визначену групу функціональних вимог і відповідає конкретному домену предметної області. Модульна організація програмного забезпечення є одним із фундаментальних принципів сучасної розробки: вона забезпечує розподіл відповідальності, спрощує тестування і локалізацію помилок, полегшує подальшу підтримку і розширення системи. Pressman зазначає, що модульність є тією характеристикою системи, яка дозволяє розробляти, тестувати і розгортати окремі частини незалежно одна від одної, що суттєво підвищує ефективність команди і знижує ризики при внесенні змін [10]. У межах розроблюваного мобільного CRM-додатку виділяється вісім функціональних модулів, кожен із яких описується через його відповідальність, основні операції і взаємодію з іншими модулями та базою даних.

Модуль авторизації є точкою входу в систему для всіх категорій користувачів і відповідає за реєстрацію, автентифікацію і управління сесіями. При реєстрації модуль валідує вхідні дані (формат email, складність пароля, унікальність облікового запису), хешує пароль і зберігає запис у таблиці User із відповідним значенням ролі. При авторизації модуль перевіряє облікові дані,

генерує JWT-токен із заданим терміном дії і повертає його клієнтському додатку. JWT-токен використовується у всіх наступних запитах як підтвердження автентифікації і містить у своєму payload ідентифікатор користувача і його роль. Кожен інший модуль серверного рівня при отриманні запиту першочергово верифікує токен і перевіряє відповідність ролі дозволеним операціям. Такий підхід забезпечує централізований і консистентний контроль доступу без дублювання логіки авторизації в кожному модулі [13].

Модуль управління записами є функціонально центральним і реалізує основний бізнес-процес системи — бронювання послуг. Його основні операції: створення нового запису, перегляд існуючих, редагування і скасування. При створенні запису модуль виконує ключову бізнес-операцію — перевірку доступності: він звертається до таблиць Schedule і Booking, щоб переконатися, що запитуваний часовий слот потрапляє у робочий час майстра і не перетинається з жодним існуючим записом. Перевірка доступності є транзакційною операцією — вона виконується в межах атомарної транзакції бази даних, щоб унеможливити ситуацію, коли два одночасні запити резервують один і той самий слот. Після успішного створення запису модуль надсилає подію до модуля нагадувань для планування автоматичних сповіщень [8].

Модуль управління клієнтами реалізує функції роботи з клієнтською базою. Він забезпечує перегляд списку клієнтів з можливістю пошуку за ім'ям або номером телефону, перегляд повного профілю конкретного клієнта з його записами і нотатками, а також редагування профілю. Пошук реалізований з підтримкою часткового збігу, що дозволяє знайти клієнта навіть при неточному введенні. Модуль також забезпечує формування агрегованих даних про клієнта: загальну кількість відвідувань, суму витрат і останню дату візиту — ці дані відображаються у профілі клієнта для швидкого орієнтування адміністратора або майстра.

Модуль управління розкладом реалізує функції планування робочого часу майстрів. Він дозволяє встановлювати стандартний тижневий графік для кожного майстра, вносити виключення для конкретних дат (скорочений день, вихідний, відпустка), а також переглядати зведений календар завантаженості у форматах дня, тижня і місяця. Зведений календар є особливо важливим інструментом для адміністратора: він дозволяє одночасно бачити розклади всіх майстрів і оперативно реагувати на зміни. Модуль також надає API-ендпоінт для отримання списку доступних слотів для конкретного майстра на конкретну дату, який використовується клієнтським додатком при формуванні форми запису [8].

Модуль нагадувань реалізує автоматизований цикл комунікації з клієнтами щодо запланованих візитів. Він функціонує як фоновий сервіс, що за розкладом — наприклад, кожні 15 хвилин — перевіряє базу записів на наявність візитів, що відбудуться протягом наступних 24 або 2 годин (залежно від налаштованих інтервалів), і для кожного такого запису надсилає нагадування клієнту через push-сповіщення або SMS. Модуль зберігає лог відправлених нагадувань, що запобігає дублюванню повідомлень при повторному запуску. Адміністратор може налаштовувати інтервали нагадувань і обирати канал доставки. Chaffey підкреслює, що автоматизована комунікація з клієнтами є одним із найбільш економічно ефективних інструментів підвищення лояльності в сервісному бізнесі [3].

Модуль управління послугами реалізує CRUD-операції для каталогу послуг: створення нової послуги, перегляд і редагування існуючих, архівування (м'яке видалення без фізичного видалення запису). М'яке видалення є принципово важливим: видалена послуга може бути присутньою у вже закритих записах і оплатах, тому фізичне видалення порушило б цілісність даних. Замість цього послуга отримує статус «архівована» і перестає відображатися у формі запису, але залишається доступною для перегляду в історії. Модуль також забезпечує групування послуг за категоріями і підтримку

множинних цін — наприклад, якщо послуга має різну вартість залежно від тривалості або майстра.

Модуль обліку оплат фіксує фінансові транзакції і формує звітність. При завершенні прийому адміністратор або майстер позначає запис як виконаний і вносить інформацію про оплату: суму, форму розрахунку і дату. Модуль автоматично заповнює суму з вартості послуги, залишаючи можливість її коригування (наприклад, при застосуванні знижки). На основі накопичених даних модуль формує звіти: загальний дохід за обраний період, дохід у розрізі послуг і майстрів, кількість виконаних і скасованих записів. Ці звіти є основним інструментом аналітики для власника бізнесу [7].

Модуль аналітики агрегує дані з усіх інших модулів і формує зведені показники ефективності роботи салону. Він забезпечує перегляд динаміки кількості записів і доходів у часі, аналіз завантаженості кожного майстра, визначення найбільш популярних послуг і найбільш активних клієнтів. Розрахунки здійснюються на серверному рівні, а не в клієнтському додатку, що знижує навантаження на пристрій і забезпечує актуальність даних. Laudon та Laudon підкреслюють, що аналітичні інструменти є тим компонентом CRM-системи, який найбільш прямо впливає на якість стратегічних рішень власника бізнесу [1].

Модуль профілів майстрів і портфоліо розширює клієнтську частину системи, надаючи відвідувачу можливість приймати зважене рішення про запис на основі не лише рейтингу і спеціалізації майстра, а й реального прикладу його робіт. Профіль майстра містить фотографію, ім'я, спеціалізацію, рейтинг і галерею портфоліо, структуровану за категоріями послуг — стрижки, фарбування, манікюр тощо. Адміністративна частина надає майстру або адміністратору інструменти для завантаження фотографій, їх категоризації і керування видимістю. З точки зору бази даних портфоліо реалізується через окрему таблицю PORTFOLIO із зовнішнім ключем на MASTER, що дозволяє зберігати необмежену кількість робіт без зміни структури основних таблиць. Зображення зберігаються у хмарному сховищі (наприклад, AWS S3 або

Cloudflare R2), а в базі даних фіксуються лише метадані — URL, категорія, дата завантаження і ознака публічності. Preece, Rogers і Sharp зазначають, що надання користувачу релевантного контексту перед прийняттям рішення є одним із ключових принципів проєктування взаємодії, що безпосередньо підвищує задоволеність і рівень конверсії [11].

Модуль омніканального чату вирішує одну з найбільш практичних проблем сучасного б'юті-бізнесу: клієнти звертаються до салону через різні месенджери — Telegram, Viber, WhatsApp та Instagram Direct, — і без централізованого інструменту адміністратор змушений переключатися між кількома додатками, що призводить до пропущених повідомлень і затримок у відповідях. Інтеграція реалізується не через пряме підключення до кожного месенджера окремо, а через омніканальний API-агрегатор — наприклад, Chatwoot або Respond.io, — який об'єднує всі вхідні канали в єдиний потік повідомлень і надає уніфікований API для їх обробки [37]. Серверний рівень системи підключається до агрегатора через webhook: при надходженні нового повідомлення з будь-якого каналу агрегатор надсилає подію на сервер, де вона обробляється і відображається в inbox адміністратора в межах основного додатку. Відповідь адміністратора також надсилається через API агрегатора у відповідний канал, при цьому клієнт отримує повідомлення у тому самому месенджері, з якого звернувся. Таким чином, адміністратор працює в одному інтерфейсі незалежно від того, через який канал прийшло звернення.

Зведений перелік усіх функціональних модулів системи з визначенням їхньої відповідальності, основних операцій та відповідних функціональних вимог із розділу 2 наведено у таблиці 4.4.

Таблиця 4.4

Функціональні модулі системи та їхня відповідність вимогам

Модуль	**Основні операції**	**Вимоги (з таблиці 2.1)**
Авторизації	Реєстрація, вхід, управління токенами, RBAC	FR001, FR002

Управління записами	Створення, перегляд, редагування, скасування	FR003
Управління розкладом	Календар день/тиждень/місяць, блокування слотів	FR004
Клієнтської бази	Пошук, перегляд профілю, нотатки, історія	FR005
Профілів і портфоліо	Завантаження фото, галерея, перегляд профілю майстра	FR006
Нагадувань	Автоматичні push і SMS за розкладом	FR007
Обліку оплат	Фіксація розрахунків, фінансові звіти	FR008
Омніканального чату	Єдиний inbox TG, Viber, WA, Instagram	FR009
ШІ-асистента	Розпізнавання намірів, швидкі дії	FR010
Управління послугами	Каталог, ціни, тривалість, архівування	FR011
Аналітики	KPI власника, доходи, завантаженість майстрів	FR012
Онлайн-оплати	Інтеграція з LiqPay, webhook-підтвердження	FR013

Як видно з табл. 4.4, усі дванадцять функціональних вимог категорії «Must have» покриваються визначеними модулями без прогалин, що підтверджує повноту архітектурного рішення. Кожен модуль має чітко окреслену зону відповідальності і взаємодіє з іншими виключно через визначені інтерфейси, що відповідає принципу слабкого зв'язування і забезпечує незалежність їхнього тестування і розгортання. Модулі авторизації і управління записами є системоутворюючими: перший забезпечує безпеку і контроль доступу для всіх інших модулів, другий реалізує центральний бізнес-процес, від якого залежать модулі нагадувань, оплат і аналітики. Модулі портфоліо і омніканального чату є розширювальними — вони не є критичними для базового функціонування системи, але суттєво підвищують якість клієнтського досвіду і конкурентоспроможність продукту. Така модульна організація системи створює надійну основу для поетапного нарощування функціоналу у наступних версіях продукту без необхідності перебудови існуючої архітектури.

4.4 Проєктування інтерфейсу користувача (UI/UX)

Проєктування інтерфейсу користувача є тим етапом розробки, де технічні рішення набувають вигляду, безпосередньо сприйманого кінцевим

користувачем. Якість інтерфейсу визначає, наскільки ефективно користувач може взаємодіяти з системою і чи вирішує вона його задачі у реальних умовах використання, а не лише в теоретично ідеальних сценаріях. Як зазначає Don Norman, хороший дизайн є «невидимим»: користувач концентрується на досягненні мети, а не на самому процесі взаємодії [17]. Водночас Jesse James Garrett визначає UX-проєктування як процес прийняття взаємопов'язаних рішень — від інформаційної архітектури до візуального оформлення — кожне з яких впливає на кінцевий досвід користувача [16].

У межах розроблюваного мобільного CRM-додатку інтерфейс проєктується з урахуванням двох принципово різних контекстів використання: клієнтського та адміністративного. Ці контексти відрізняються як за частотою використання, так і за складністю сценаріїв, що обумовлює необхідність реалізації двох різних UX-підходів у межах одного продукту [12]. Додаток функціонує у двох режимах, які визначаються роллю користувача після авторизації, що дозволяє забезпечити релевантний інтерфейс для кожного типу взаємодії.

Клієнтський інтерфейс орієнтований на нерегулярного користувача, який взаємодіє із системою епізодично та виконує обмежену кількість завдань. Основним сценарієм є запис на послугу, тому навігаційна структура побудована за принципом мінімальної глибини: ключова дія доступна не більше ніж за один-два кроки від головного екрана. Головний екран містить домінуючу кнопку запису, блок найближчих записів і перелік доступних послуг у вигляді карток. Такий підхід відповідає принципу пріоритизації найчастішого сценарію використання, коли найбільш імовірна дія розташовується у зоні швидкого доступу [23].

Навігаційний потік клієнта реалізований у вигляді покрокового (wizard) процесу, що включає послідовність: вибір послуги, вибір майстра, вибір дати і часу, а також підтвердження запису. Кожен етап є логічно завершеним і супроводжується чітким візуальним зворотним зв'язком (наприклад, чекмарки, підсвічування вибраних елементів), що дозволяє користувачу контролювати

процес. На етапі вибору майстра передбачена можливість автоматичного підбору («Any available»), що знижує когнітивне навантаження для користувачів без чітких уподобань. Екран вибору дати реалізований як інтерактивний календар із відображенням доступних і недоступних часових слотів, причому недоступні значення не приховуються, а позначаються, що забезпечує повніше уявлення про завантаженість. Підсумковий екран підтвердження узагальнює всі параметри запису і знижує невизначеність щодо подальших дій, що відповідає рекомендаціям Steve Krug щодо важливості фінального етапу взаємодії [23].

Адміністративний інтерфейс, на відміну від клієнтського, розрахований на інтенсивне щоденне використання і підтримує широкий спектр операцій. Центральним елементом є календар записів, який відкривається одразу після авторизації та виступає основним робочим інструментом адміністратора. Календар підтримує три режими перегляду — день, тиждень і місяць — що дозволяє швидко перемикатися між детальним та агрегованим рівнями інформації без зміни контексту роботи. Денний вигляд відображає розклад майстрів у часовій шкалі, тижневий — зведену інформацію із ключовими метриками, а місячний — загальну картину завантаженості у вигляді теплової карти, що сприяє стратегічному плануванню.

Інтеракція з календарем забезпечує швидкий доступ до деталей запису, які містять повну інформацію про клієнта, послугу, статус оплати та додаткові нотатки. Основні дії, такі як перенесення запису або підтвердження візиту, виконуються безпосередньо з цього екрану, що мінімізує кількість переходів між розділами. Такий підхід відповідає принципу оптимізації робочих процесів та скорочення кількості дій для виконання типових операцій, який є ключовим у системах із високою інтенсивністю використання [12].

Візуальний дизайн додатку базується на компонентному підході із використанням рекомендацій Google Material Design, що забезпечує консистентність інтерфейсу та знайомі для користувача патерни взаємодії [27]. Основним акцентним кольором обрано фіолетовий, який використовується для

інтерактивних елементів, тоді як фонові кольори залишаються нейтральними для зниження візуального шуму. Кольорове кодування застосовується також для диференціації записів у календарі, що дозволяє швидко орієнтуватися без необхідності читання тексту. Інтерфейс підтримує світлу і темну теми, що забезпечує комфортне використання в різних умовах освітлення. Мінімальний розмір інтерактивних елементів відповідає стандарту 44×44 пікселі, що гарантує зручність використання на мобільних пристроях [27].

Особлива увага приділяється обробці станів системи, включаючи завантаження, помилки та порожні стани. Кожна операція, що потребує мережевої взаємодії, супроводжується індикатором виконання, а помилки відображаються у зрозумілій для користувача формі з можливістю повторної спроби. Критичні дії потребують підтвердження для запобігання випадковим помилкам. Відображення статусів (наприклад, «PAID», «CONFIRMED») забезпечує прозорість процесів і підвищує довіру до системи. Як підкреслює Garrett, проєктування всіх можливих станів системи є невід'ємною частиною UX і суттєво впливає на сприйняття продукту користувачем [16].

Таким чином, інтерфейс мобільного CRM-додатку формується як узгоджена система двох взаємодоповнюючих UX-флоу — клієнтського, орієнтованого на простоту і швидкість, та адміністративного, орієнтованого на ефективність і багатофункціональність, що дозволяє забезпечити високий рівень зручності використання для всіх категорій користувачів.

Таким чином, інтерфейс мобільного CRM-додатку формується як узгоджена система двох взаємодоповнюючих UX-флоу — клієнтського, орієнтованого на простоту і швидкість, та адміністративного, орієнтованого на ефективність і багатофункціональність, що дозволяє забезпечити високий рівень зручності використання для всіх категорій користувачів. Макети екранів клієнтської частини додатку наведено на рис. 4.3 та рис. 4.4.

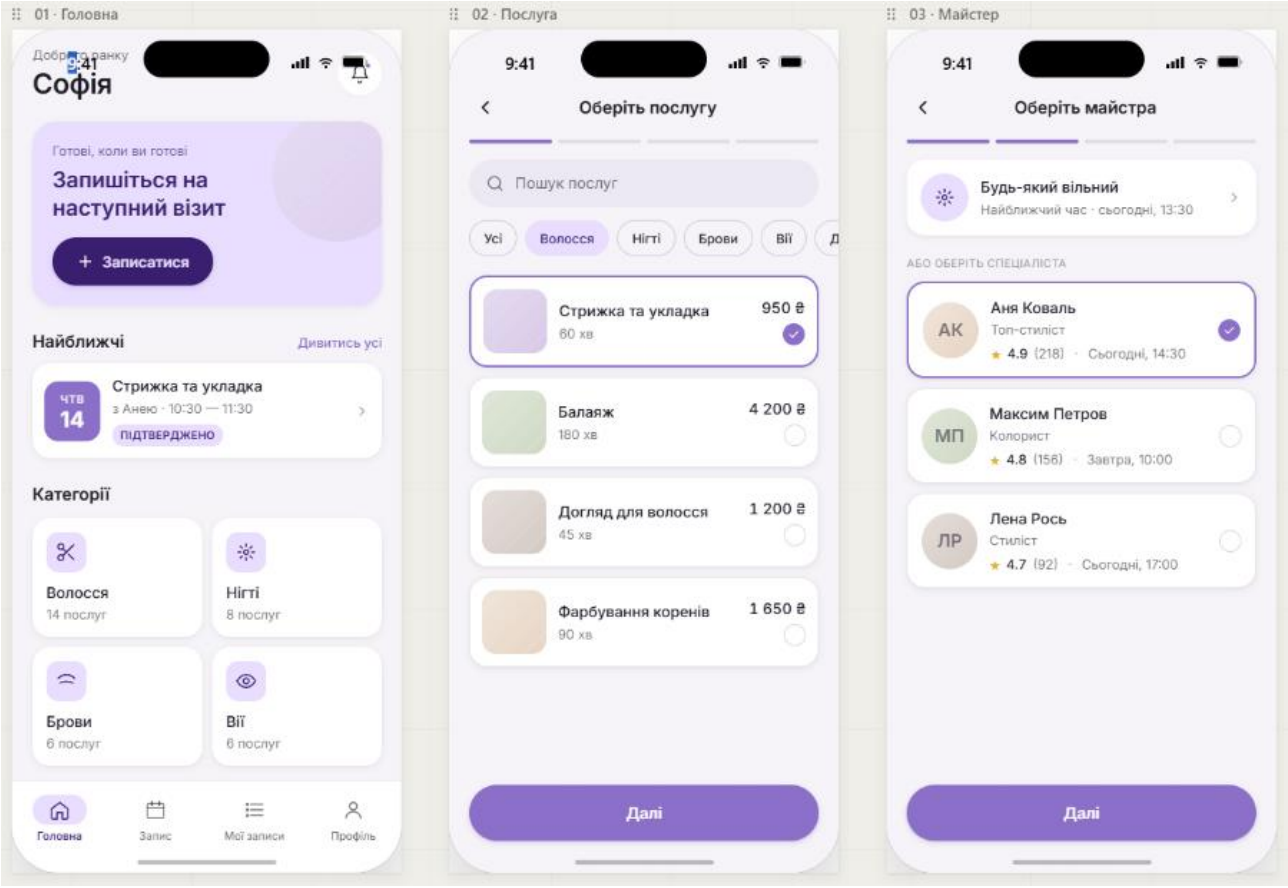


Рис. 4.3 Клієнтська частина

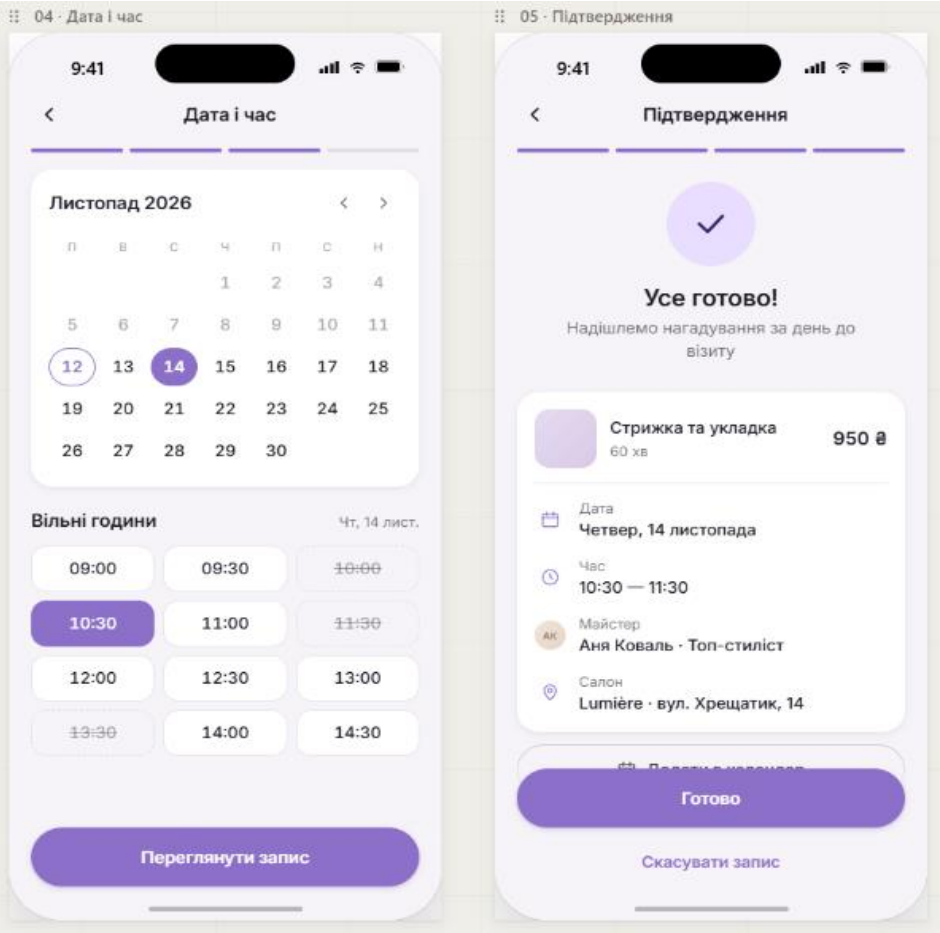


Рис. 4.4 Продовження клієнтської частини

Клієнтська частина інтерфейсу, представлена на рис. 4.3 та рис. 4.4, демонструє мінімалістичний підхід до проєктування користувацького досвіду: основний сценарій запису на послугу реалізований у чотири кроки — вибір послуги, вибір майстра, вибір дати і часу, підтвердження — що відповідає вимозі NFR001 щодо обмеження основного сценарію чотирма кроками. Навігаційна структура клієнтської частини побудована на основі нижньої панелі з чотирма вкладками, що є стандартом мобільного UX і дозволяє користувачу в будь-який момент перейти до потрібного розділу без необхідності повертатися до головного екрану.

Адміністративна частина, наведена на рис. 4.5, вирішує принципово інше завдання: замість спрощення вона забезпечує ефективність роботи з великим обсягом інформації.

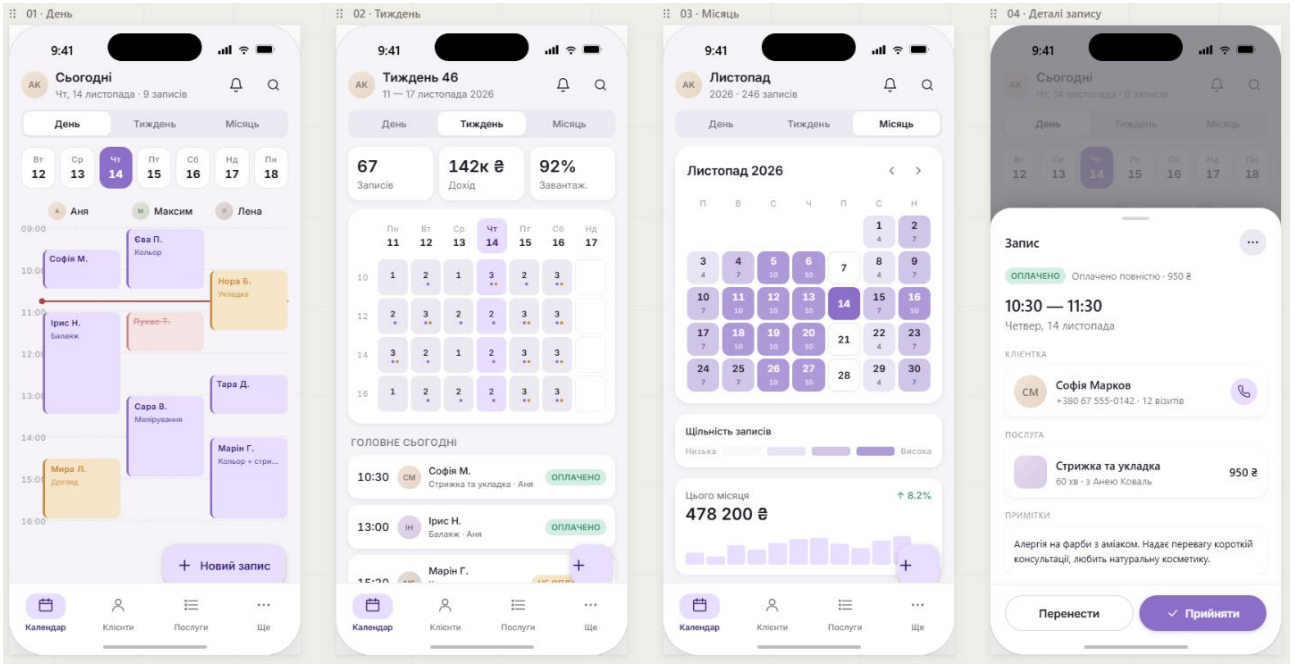


Рис. 4.5 Адміністративна частина

Зведений календар є центральним елементом адміністративного інтерфейсу і відображає записи всіх майстрів у форматі тижня з можливістю перемикання на перегляд дня або місяця. Панель клієнтської бази забезпечує швидкий пошук і доступ до повного профілю клієнта з його історією відвідувань. Таке розділення інтерфейсів за роллю користувача при збереженні єдиної кодової бази є одним із ключових архітектурних рішень, що забезпечує масштабованість і керованість системи у довгостроковій перспективі.

4.5 Інтеграція зовнішніх сервісів та ШІ-асистента адміністратора

Сучасні мобільні додатки рідко функціонують як повністю автономні системи — їхня практична цінність значною мірою визначається здатністю інтегруватися із зовнішніми сервісами, що розширюють базовий функціонал без необхідності власної реалізації складної інфраструктури. Для розроблюваного CRM-додатку зовнішні інтеграції охоплюють чотири функціональних напрями: доставка push-сповіщень, надсилання SMS, обробка онлайн-платежів і омніканальна комунікація з клієнтами з підтримкою ШІ-асистента. Кожна інтеграція реалізується на серверному рівні через відповідні API, що забезпечує централізований контроль, безпеку ключів доступу і можливість заміни провайдера без змін у клієнтському додатку. Pressman

зазначає, що надійна інтеграція з зовнішніми системами вимагає проєктування не лише номінального сценарію взаємодії, а й сценаріїв відмов і відновлення, оскільки зовнішні сервіси є джерелом залежностей, що знаходяться поза прямим контролем команди [10].

Push-сповіщення є основним каналом автоматичних нагадувань клієнтам про заплановані візити. Реалізація здійснюється через Firebase Cloud Messaging (FCM) — сервіс Google для доставки повідомлень на пристрої з Android і iOS. При першому запуску додатку клієнтська частина запитує дозвіл на отримання сповіщень і у разі його надання отримує від FCM унікальний токен пристрою. Цей токен передається на серверний рівень і зберігається у базі даних у прив'язці до облікового запису користувача. Модуль нагадувань за розкладом перевіряє наближення запланованих візитів і формує запит до FCM API із токеном пристрою і текстом повідомлення. У разі, якщо FCM повертає помилку застарілого токена — наприклад, клієнт перевстановив додаток, — сервер видаляє недійсний токен і переходить до резервного каналу SMS. Усі відправлені сповіщення логуються в базі даних для запобігання дублюванню при повторному запуску фонового процесу.

SMS є резервним каналом комунікації для клієнтів, які не встановили мобільний додаток або не надали дозвіл на push-сповіщення. На відміну від push, SMS не залежить від наявності додатку на пристрої і забезпечує вищу гарантію доставки, однак є коштовнішим у використанні. Для українського ринку як основний провайдер обрано Twilio або вітчизняний аналог із підтримкою відправки на українські номери. Алгоритм відправки SMS аналогічний push: серверний рівень формує запит до API шлюзу із номером телефону і текстом, шлюз здійснює доставку і повертає статус, який фіксується у лозі нагадувань. Адміністратор може налаштовувати інтервали нагадувань і визначати пріоритет каналу для кожного клієнта окремо або глобально для всього салону.

Онлайн-оплата реалізована у форматі MVP через LiqPay як провідного провайдера для українського ринку з підтримкою карток Visa і Mastercard у

гривні. Процес оплати побудований таким чином, що платіжні дані клієнта ніколи не проходять через серверний рівень системи — вони обробляються виключно SDK LiqPay і серверами платіжної системи, що знімає з команди відповідальність за відповідність стандарту PCI DSS. Технічна схема взаємодії включає такі кроки: клієнт ініціює оплату у додатку; серверний рівень звертається до LiqPay API і отримує унікальний ідентифікатор платіжної сесії; клієнтський додаток відображає стандартну платіжну форму LiqPay SDK; після завершення транзакції LiqPay надсилає webhook-повідомлення на серверний рівень із результатом; сервер оновлює статус оплати у таблиці PAYMENT і надсилає клієнту підтвердження. Webhook обробляється ідемпотентно — повторне отримання того самого повідомлення не призводить до подвійного запису оплати.

Інтеграція месенджерів реалізована не через пряме підключення до кожного каналу окремо, а через омніканальний агрегатор Chatwoot — відкриту платформу для управління клієнтськими комунікаціями, що об'єднує Telegram, Viber, WhatsApp і Instagram Direct в єдиний потік повідомлень [42]. Chatwoot розгортається як self-hosted рішення, що виключає залежність від хмарного SaaS-провайдера і забезпечує повний контроль над даними переписки. Серверний рівень CRM-системи підключається до Chatwoot через webhook: при надходженні нового повідомлення з будь-якого каналу Chatwoot надсилає подію на сервер, де вона обробляється і відображається в inbox адміністратора в межах основного додатку. Відповідь адміністратора надсилається назад через Chatwoot API у відповідний канал, де клієнт отримує її у тому ж месенджері, через який звернувся. Таким чином адміністратор працює в єдиному інтерфейсі незалежно від каналу звернення.

ШІ-асистент є надбудовою над омніканальним чатом і реалізований за принципом «AI in the loop» — штучний інтелект аналізує вхідні повідомлення і пропонує готові дії, однак жодна з них не виконується без явного підтвердження адміністратора [43]. Це принципова відмінність від автономного чат-бота: система підвищує ефективність адміністратора, але не замінює його

рішення. Технічна реалізація будується на основі API великої мовної моделі — Claude API від Anthropic або GPT-4o від OpenAI. При надходженні нового повідомлення від клієнта серверний рівень формує структурований запит до LLM API, що містить текст повідомлення, скорочену історію попередніх звернень клієнта, його профіль і список доступних послуг та майстрів на найближчі дні. Модель повертає відповідь у форматі JSON із розпізнаним наміром і переліком запропонованих швидких дій, які відображаються адміністратору у вигляді карток із кнопками безпосередньо під вхідним повідомленням.

Типові сценарії роботи асистента охоплюють найбільш поширені запити. Якщо клієнт пише «Хочу записатися на стрижку в п'ятницю», асистент розпізнає намір на запис і пропонує конкретні доступні слоти з іменами майстрів як окремі кнопки. Якщо клієнт запитує ціну — асистент формує готову відповідь із актуальним прайсом і пропонує кнопку «Надіслати». При запиті на скасування — відображає деталі активного запису і кнопки «Скасувати» або «Перенести». У разі нерозпізнаного наміру асистент сигналізує про це і залишає відповідь повністю на розсуд адміністратора. Усі запити до LLM API логуються на серверному рівні для моніторингу якості розпізнавання і виявлення систематичних помилок.

З точки зору безпеки всі виклики зовнішніх API є асинхронними і не блокують основний потік обробки запитів від користувачів. Ключі доступу до зовнішніх сервісів зберігаються виключно у змінних середовища серверного рівня і ніколи не передаються у клієнтський додаток. Кожна інтеграція реалізована з механізмом повторних спроб при тимчасових помилках із експоненційним відступом між спробами, що забезпечує стійкість системи до короткочасної недоступності зовнішніх сервісів.

Зведений перелік усіх зовнішніх інтеграцій із визначенням їхнього призначення, використовуваного API та механізмів забезпечення надійності наведено у таблиці 4.3.

Таблиця 4.3

Зовнішні сервіси та параметри інтеграції

Сервіс	Призначення	Протокол	Резервний варіант
Firebase FCM	Push-нагадування	HTTPS API	SMS-шлюз
Twilio / LiqPay SMS	Резервні SMS-нагадування	HTTPS API	Ручне нагадування
LiqPay	Онлайн-оплата	SDK + Webhook	Оплата готівкою
Chatwoot	Омніканальний чат	Webhook + REST API	Прямі месенджери
Claude API / GPT-4o	ШІ-асистент	HTTPS API	Ручна відповідь

Таким чином, інтеграція зовнішніх сервісів перетворює базовий CRM-додаток на комплексний інструмент управління комунікацією з клієнтами. Push-сповіщення і SMS забезпечують автоматизовану комунікацію щодо записів, платіжна система — зручність розрахунків, омніканальний чат — централізацію всіх звернень, а ШІ-асистент — якісно новий рівень ефективності адміністратора, що дозволяє обробляти більший обсяг звернень з меншими трудовитратами і меншою кількістю пропущених запитів.

ВИСНОВКИ

У кваліфікаційній роботі досліджено технології управління проєктом створення мобільного додатку для управління клієнтами салону б'юті послуг. Тема є актуальною, зокрема через те, що більшість малих і середніх салонів краси досі не використовують спеціалізованих цифрових рішень для управління клієнтами, а наявні на ринку платформи не задовольняють потреб цього сегменту. За результатами дослідження можна сформулювати такі висновки.

1. Проведено комплексний аналіз предметної області, що охопив дослідження бізнес-процесів салону б'юті послуг, порівняльний аналіз існуючих CRM-платформ (Fresha, Booksy, YCLIENTS) та виявлення ключових проблем управління клієнтами. Встановлено, що жодне з розглянутих рішень не є оптимальним для невеликого українського салону, що обґрунтовує доцільність власної розробки. Побудовано дерево проблем і дерево цілей проєкту, визначено зацікавлені сторони та розроблено паспорт проєкту.

2. Здійснено моделювання системи та формування вимог. Визначено чотири основні ролі користувачів і реалізовано принцип рольового контролю доступу. Сформовано 12 функціональних вимог із кодами FR001–FR012 і 7 нефункціональних вимог (NFR001–NFR007), пріоритизованих за методом MoSCoW. Побудовано Use Case-модель і концептуальну архітектуру інформаційної системи, що охоплює чотири рівні: мобільний клієнт, серверний рівень, базу даних і зовнішні сервіси.

3. Розроблено повний план управління проєктом розробки на основі методології Scrum із двотижневими спринтами. Сформовано Product Backlog із 16 User Stories та критеріями приймання, реалізовано принцип Planning Poker для оцінки Story Points. Розроблено організаційну структуру команди з п'яти осіб із чітким розподілом Scrum-ролей.

4. Побудовано WBS із декомпозицією на шість фаз і загальним обсягом 7 040 годин трудовитрат, а також детальний календарний план на 52 тижні (24 спринти) з діаграмою Ганта. Визначено критичний шлях проєкту і

забезпечено паралельне виконання серверних і клієнтських задач для мінімізації загальної тривалості розробки.

5. Сформовано реєстр ризиків із 30 позицій у шести категоріях із кількісною оцінкою за чотирма параметрами і триступеневими протиризиковими заходами. Розроблено бюджетний план загальним обсягом 2 569 270 грн, де фонд оплати праці становить 87,9% від загального бюджету, та визначено 9 ключових показників ефективності проєкту і продукту.

6. Реалізовано проєктування інформаційного та програмного забезпечення. Обрано трирівневу клієнт-серверну архітектуру на базі React Native, Node.js + Express і PostgreSQL. Спроєктовано структуру бази даних із восьми сутностей, нормалізовану до третьої нормальної форми, розроблено вісім функціональних модулів системи та інтерфейс користувача для клієнтської і адміністративної частин.

7. Реалізовано інтеграцію п'яти зовнішніх сервісів: Firebase FCM (push-нагадування), Twilio/LiqPay SMS (резервний канал), LiqPay (онлайн-оплата), Chatwoot (омніканальний чат) і LLM API (ШІ-асистент адміністратора за принципом «AI in the loop»). Усі інтеграції реалізовано з механізмами повторних спроб і безпечного зберігання ключів доступу.

Таким чином, усі поставлені завдання дослідження виконано, що дозволило досягти головної мети — дослідити технології управління проєктом створення мобільного CRM-додатку для салону б'юті послуг і розробити комплексну концепцію програмного продукту. Практичне значення результатів полягає в тому, що розроблена концепція, план управління проєктом і проєктна документація можуть бути безпосередньо використані для реалізації реального програмного продукту. Подальший розвиток дослідження може бути спрямований на реалізацію і тестування MVP-версії додатку, розширення функціоналу ШІ-асистента та масштабування архітектури до multi-tenant моделі для підтримки мережевих салонів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Laudon K., Laudon J. Management Information Systems: Managing the Digital Firm. Pearson Education, 2020.
2. Kotler P., Keller K. Marketing Management. Pearson, 2016.
3. Chaffey D. Digital Business and E-Commerce Management. Pearson, 2019.
4. Fresha – платформа для управління салонами краси: https://www.fresha.com
5. Booksy – сервіс онлайн-запису для б’юті індустрії: https://biz.booksy.com
6. YCLIENTS – CRM система для салонів краси: https://www.yclients.com
7. Buttle F., Maklan S. Customer Relationship Management: Concepts and Technologies. Routledge, 2019.
8. Stair R., Reynolds G. Principles of Information Systems. Cengage Learning, 2021.
9. Sommerville I. Software Engineering. Pearson, 2016.
10. Pressman R. Software Engineering: A Practitioner's Approach. McGraw-Hill, 2014.
11. Preece J., Rogers Y., Sharp H. Interaction Design: Beyond Human-Computer Interaction. Wiley, 2019.
12. Nielsen J. Usability Engineering. Morgan Kaufmann, 1993.
13. ISO/IEC 25010:2011. Systems and software engineering - System and software quality models.
14. McKinsey & Company. The future of the beauty industry and digital transformation. https://www.mckinsey.com
15. Statista. Beauty & Personal Care Market - Digital Trends. https://www.statista.com
16. Garrett J. The Elements of User Experience. New Riders, 2011.
17. Norman D. The Design of Everyday Things. Basic Books, 2013.
18. Deloitte. Digital transformation in the beauty industry. https://www2.deloitte.com

19. Fowler M. UML Distilled: A Brief Guide to the Standard Object Modeling Language. Addison-Wesley, 2004.
20. Cockburn A. Writing Effective Use Cases. Addison-Wesley, 2000.
21. OMG. Unified Modeling Language (UML) Specification. https://www.omg.org/spec/UML
22. Larman C. Applying UML and Patterns. Pearson, 2004.
23. Krug S. Don’t Make Me Think: A Common Sense Approach to Web Usability. New Riders, 2014.
24. Wroblewski L. Mobile First. A Book Apart, 2011.
25. Salesforce. State of the Connected Customer Report. https://www.salesforce.com
26. Gartner. CRM and Customer Experience Trends. https://www.gartner.com
27. Google. Material Design Guidelines. https://material.io
28. Elmasri R., Navathe S. Fundamentals of Database Systems. Pearson, 2016.
29. Date C. An Introduction to Database Systems. Addison-Wesley, 2004.
30. Silberschatz A., Korth H., Sudarshan S. Database System Concepts. McGraw-Hill, 2019.
31. Connolly T., Begg C. Database Systems: A Practical Approach. Pearson, 2015.
32. Retailer.com.ua. Б'юті-ринок України 2024. URL: https://retailer.com.ua/...
33. Barb.ua. Рейтинг салонів краси 2024. URL: https://barb.ua/uk/news/salon-rating-2024
34. Salonmarketing.pro. 25% українських салонів краси користуються російською CRM. URL: https://ua.salonmarketing.pro/...
35. Myvin.com.ua. Від паперового журналу до цифровізації. URL: https://www.myvin.com.ua/...
36. EasyWeek. Топ-5 систем онлайн-запису до салону краси. URL: https://easyweek.com.ua/...
37. Bass L., Clements P., Kazman R. Software Architecture in Practice. Addison-Wesley, 2021.

38. Fielding R. T. Architectural Styles and the Design of Network-based Software Architectures. University of California, Irvine, 2000. URL: https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm
39. Tilkov S., Vinoski S. Node.js: Using JavaScript to Build High-Performance Network Programs. IEEE Internet Computing, 2010. Vol. 14, No. 6. P. 80–83.
40. LiqPay. Документація платіжного API. URL: https://www.liqpay.ua/documentation
41. Facebook Inc. React Native Documentation. URL: https://reactnative.dev/docs/getting-started
42. Chatwoot. Open-source customer engagement platform. URL: https://www.chatwoot.com/docs
43. Amershi S. et al. Software Engineering for Machine Learning: A Case Study. IEEE/ACM International Conference on Software Engineering, 2019. P. 291–300.
44. Лановий А. Ф., Чабаненко М. В. Управління IT-проєктами з використанням гнучких методологій. Науковий журнал НАУ «Проблеми інформатизації та управління». 2022. № 1 (69). URL: https://jrnl.nau.edu.ua/index.php/DEU/article/download/17349/24656/42617
45. PremierAgile. Scrum Framework Explained (2025): Roles, Events, Values & Rules. URL: https://premieragile.com/scrum-framework-explained-2025/
46. Galeano I., Merín M., González M., Cernuzzi L. A methodological approach for mobile applications development. Science of Computer Programming. 2023. Vol. 230. DOI: 10.1016/j.scico.2023.102986
47. Cohn M. User Stories Applied: For Agile Software Development. Addison-Wesley, 2004.
48. Asana. User Stories Explained: Tips, Templates, and Examples. 2025. URL: https://asana.com/resources/user-stories
49. Вісник Східноукраїнського національного університету ім. В. Даля. Управління ризиками в IT проєктах з гнучкими методологіями. 2024. № 1 (281). URL:

https://journals.snu.edu.ua/index.php/VisnikSNU/article/download/796/758/782

50. Вісник Херсонського національного технічного університету. Класифікація ризиків у проєктах із розробки програмного забезпечення. 2023. № 3(86). URL: https://journals.kntu.kherson.ua/index.php/visnyk_kntu/article/view/456
51. Морозов В.В., Хандрік О.В., Коломієць А.С. Управління проектами розвитку ІТ організацій: Навчальний посібник. – К.: ВПЦ «Київський університет», 2020. – 339 с.
52. Морозов В.В., Чередніченко А.М., Шпильова Т.Ш. Формування, управління та розвиток команди проекту (поведінкові компетенції): Навчальний посібник. – К.: Таксон, 2009. – 464 с.

Додаток А

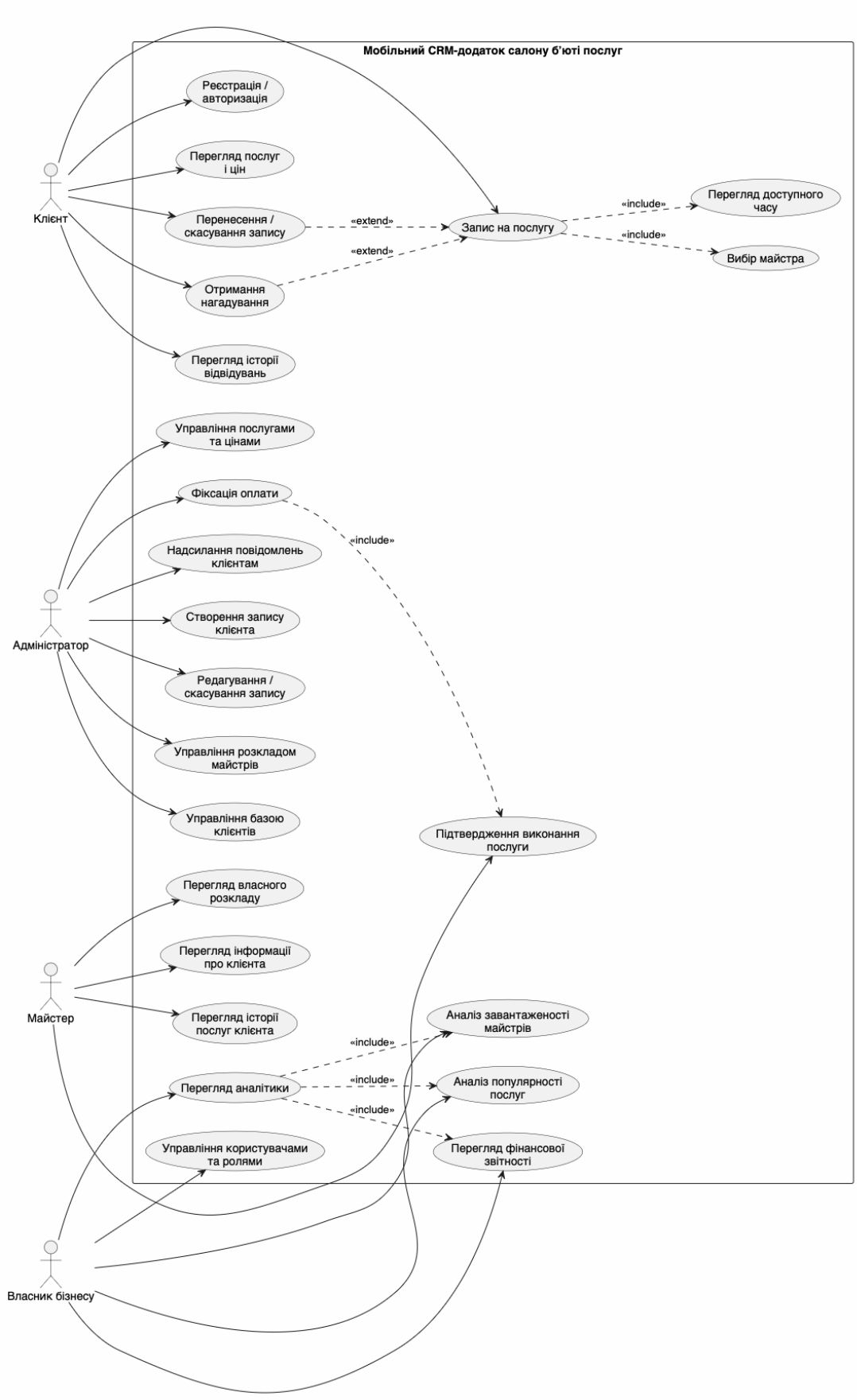


Рис. А.1 Діаграма варіантів використання мобільного CRM-додатку салону б'юті послуг

Додаток Б

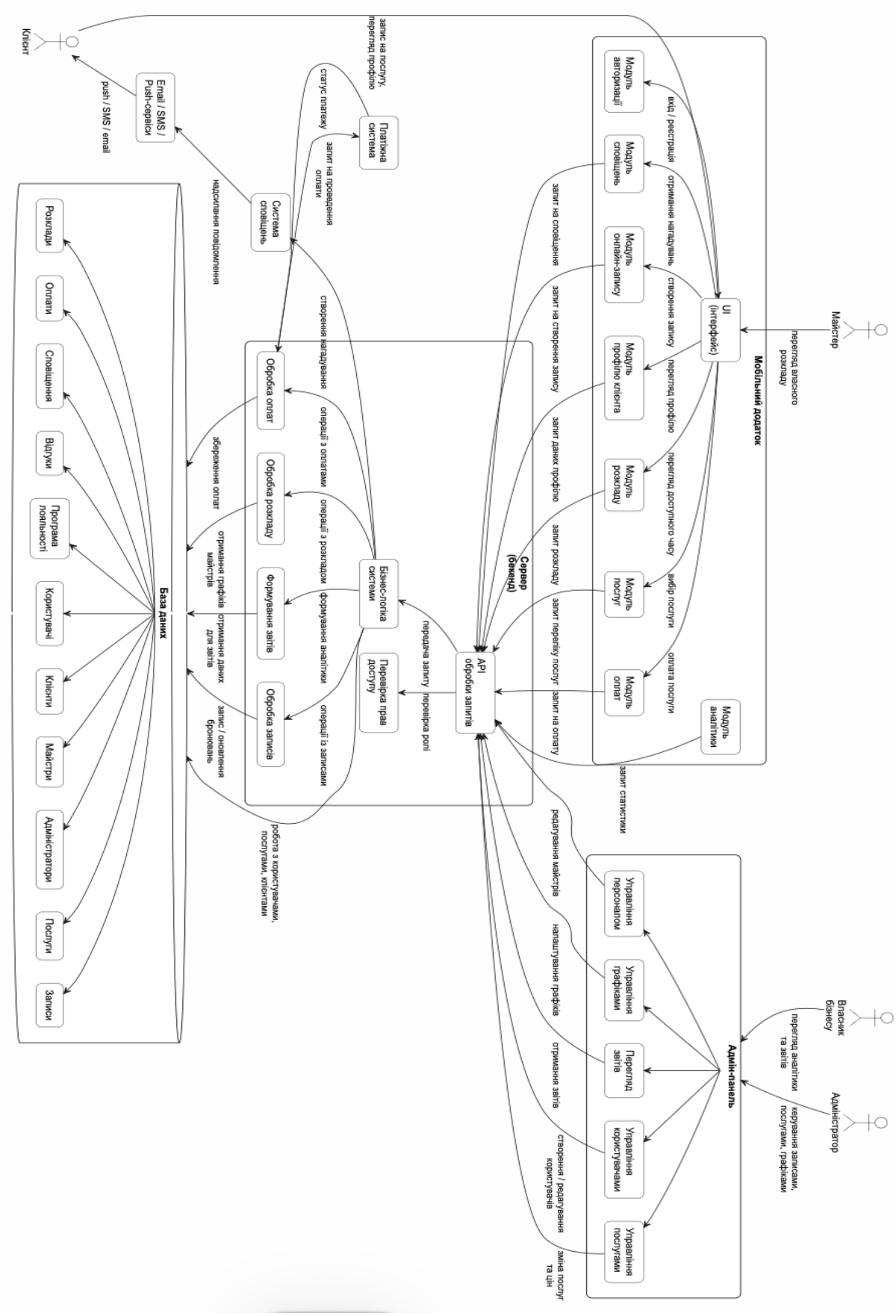


Рис. Б.1 Діаграма варіантів використання мобільного CRM-додатку салону б'юті послуг

Додаток В

Таблиця В.1

Фрагмент Product Backlog із User Stories та критеріями приймання

Код US	User Story	Критерії приймання	Пріоритет	SP
1	2	3	4	5
US001	Як клієнт, я хочу зареєструватися за номером телефону, щоб мати власний обліковий запис	1. Форма містить поля: ім'я, телефон, пароль. 2. Система перевіряє унікальність номера. 3. Після реєстрації надсилається SMS-підтвердження. 4. Після підтвердження створюється обліковий запис	Must have	3
US002	Як адміністратор, я хочу входити за email і паролем, щоб мати доступ до адмін-панелі	1. Форма містить поля email і пароль. 2. При невірних даних — повідомлення про помилку. 3. Після успішного входу — перехід до календаря	Must have	2
US003	Як клієнт, я хочу переглянути список послуг із цінами, щоб обрати потрібну процедуру	1. Послуги згруповані за категоріями. 2. Для кожної послуги відображається назва, тривалість і ціна. 3. Є рядок пошуку і фільтр за категорією	Must have	3
US004	Як клієнт, я хочу обрати майстра або «будь-який доступний», щоб записатися зручним способом	1. Список майстрів із фото, іменем і рейтингом. 2. Опція «будь-який доступний» показує найближчий вільний слот. 3. При виборі майстра відображаються його доступні часові слоти	Must have	3

Продовження таблиці В.1

1	2	3	4	5
US005	Як клієнт, я хочу бачити вільні часові слоти у календарі, щоб обрати зручний час	1. Відображення доступних слотів по годинах. 2. Зайняті слоти відображаються закресленими. 3. Після вибору — екран підтвердження	Must have	8
US006	Як клієнт, я хочу отримати підтвердження запису, щоб бути впевненим у бронюванні	1. Екран підтвердження показує: послугу, майстра, дату, час, ціну. 2. Надсилається push-повідомлення. 3. Запис з'являється у розділі «Мої записи»	Must have	3
US007	Як клієнт, я хочу перенести або скасувати запис, щоб змінити плани без дзвінка	1. Активний запис доступний для редагування. 2. При скасуванні — запит підтвердження. 3. Майстер отримує push-сповіщення про зміну	Must have	5
US008	Як адміністратор, я хочу бачити денний календар усіх майстрів, щоб контролювати розклад	1. Стовпець для кожного майстра з блоками записів. 2. Колір блоку відповідає категорії послуги. 3. Натискання на блок відкриває деталі запису	Must have	8
US009	Як адміністратор, я хочу бачити тижневий вигляд із метриками, щоб оцінювати ефективність	1. Відображення кількості записів і доходу за тиждень. 2. Завантаженість кожного майстра у відсотках. 3. Список записів поточного дня нижче	Must have	5
US010	Як майстер, я хочу переглянути деталі запису, щоб підготуватися до прийому	1. Відображення: ім'я клієнта, послуга, час, нотатки. 2. Доступна клієнтська історія попередніх відвідувань. 3. Кнопки «Check in» і «Reschedule»	Must have	5
US011	Як клієнт, я хочу отримати нагадування за 24 год до запису, щоб не забути про візит	1. Push-нагадування надсилається автоматично за 24 год. 2. Нагадування містить: дату, час, майстра, послугу. 3. SMS як резервний канал при відсутності push	Must have	5

Закінчення таблиці В.1

1	2	3	4	5
US012	Як клієнт, я хочу переглянути портфоліо майстра, щоб оцінити якість роботи	1. Галерея фото, згрупована за категоріями. 2. Можливість збільшення фото. 3. Кнопка «Записатися» доступна з профілю	Should have	5
US013	Як адміністратор, я хочу бачити всі повідомлення в одному inbox, щоб не пропускати звернення	1. Повідомлення з TG, Viber, WA, Instagram в одному інтерфейсі. 2. Канал відправника позначений іконкою. 3. Відповідь надсилається у відповідний канал	Should have	13
US014	Як адміністратор, я хочу отримувати підказки від ШІ щодо запитів клієнтів, щоб швидше відповідати	1. ШІ розпізнає намір і пропонує швидкі дії. 2. Дія виконується лише після підтвердження адміністратором. 3. Логування всіх запитів до ШІ	Should have	13
US015	Як власник, я хочу бачити аналітику доходів і завантаженості, щоб управляти бізнесом на основі даних	1. Загальний дохід за обраний період. 2. Завантаженість кожного майстра у відсотках. 3. Топ-5 найпопулярніших послуг	Should have	8
US016	Як клієнт, я хочу оплатити послугу онлайн, щоб не носити готівку	1. Інтеграція з LiqPay. 2. Платіжні дані не зберігаються на сервері. 3. Після оплати — підтвердження і зміна статусу запису	Could have	13

Додаток Г

Таблиця Г.1

Календарний план по спринтах

Спринт	Тижні	Місяць	Основні задачі	Результат
1	2	3	4	5
С1	1–2	Жовтень 2025	Вимоги, середовище, Story Writing Workshop	Product Backlog, налаштоване середовище
С2	3–4	Жовтень 2025	Макети всіх екранів у Figma, дизайн-система	Узгоджений прототип
С3	5–6	Листопад 2025	Схема БД, міграції, модуль авторизації	Робочий API авторизації
С4	7–8	Листопад 2025	API записів, перевірка доступності слотів	Модуль записів
С5	9–10	Грудень 2025	API розкладу, клієнтська база	API розкладу і клієнтів
С6	11–12	Грудень 2025	API послуг, оплат, базова аналітика	Повний серверний API MVP
С7	13–14	Січень 2026	React Native: авторизація, навігація	Базова структура додатку
С8	15–16	Січень 2026	Клієнтська частина: потік запису	Повний потік запису клієнта
С9	17–18	Лютий 2026	Профілі майстрів, портфоліо (клієнт)	Екрани профілів і галереї
С10	19–20	Лютий 2026	Адмін: денний і тижневий календар	Основний календар адміністратора
С11	21–22	Березень 2026	Адмін: місячний вигляд, деталі запису	Повна адміністративна частина
С12	23–24	Березень 2026	Push-нагадування, SMS-шлюз	Автоматичні нагадування
С13	25–26	Квітень 2026	Інтеграція Chatwoot (омніканальний чат)	Єдиний inbox для месенджерів
С14	27–28	Квітень 2026	ШІ-асистент: розпізнавання намірів, дії	Робочий ШІ-асистент

Закінчення таблиці Г.1

1	2	3	4	5
С15	29–30	Травень 2026	Інтеграція LiqPay, онлайн-оплата	Платіжний модуль
С16	31–32	Травень 2026	Модуль аналітики, фінансова звітність	Панель аналітики власника
С17	33–34	Червень 2026	Функціональне тестування (модулі 1–4)	Звіт дефектів, виправлення
С18	35–36	Червень 2026	Функціональне тестування (модулі 5–8)	Звіт дефектів, виправлення
С19	37–38	Липень 2026	Тестування інтеграцій, навантажувальне	Підтверджена продуктивність
С20	39–40	Липень 2026	Тестування безпеки, аудит даних	Звіт безпеки
С21	41–42	Серпень 2026	Регресійне тестування, залишкові дефекти	Стабільна версія
С22	43–44	Серпень 2026	Приймальне тестування із замовником	Підписаний акт приймання
С23	45–48	Вересень 2026	Розгортання production, документація	Система у production
С24	49–52	Вересень 2026	Інструкції, фінальний звіт, передача	Проєкт закрито

Таблиця Г.2

Ідентифікація ризиків проєкту

№	Тип ризику	Ризикова подія	Сила впливу	Керованість
1	2	3	4	6
Р01	Технічний	Критичні помилки (баги) у мобільному додатку після релізу	Висока	Висока
Р02	Технічний	Технічні складнощі інтеграції Chatwoot API	Висока	Середня

Продовження таблиці Г.2

1	2	3	4	6
Р03	Технічний	Низька якість розпізнавання намірів ШІ-асистентом	Середня	Висока
Р04	Технічний	Низька продуктивність додатку на старих смартфонах	Середня	Середня
Р05	Технічний	Конфлікт синхронізації даних між клієнтом і сервером	Висока	Висока
Р06	Технічний	Несумісність бібліотек React Native з новими версіями iOS/Android	Середня	Висока
Р07	Технічний	Збій або недоступність зовнішнього LLM API	Середня	Низька
Р08	Технічний	Відмова у публікації додатку в App Store / Google Play	Висока	Середня
Р09	Технічний	Некоректна робота платіжного шлюзу LiqPay	Висока	Середня
Р10	Технічний	Втрата даних при збої бази даних	Висока	Висока
Р11	Управлінський	Scope creep — неконтрольоване розширення вимог замовником	Висока	Середня
Р12	Управлінський	Відсутність технічної документації коду	Середня	Висока
Р13	Управлінський	Недооцінка трудовитрат при плануванні спринтів	Висока	Висока
Р14	Управлінський	Конфлікти всередині команди щодо пріоритетів	Середня	Висока
Р15	Управлінський	Нечіткі вимоги замовника на початку проєкту	Висока	Висока
Р16	Управлінський	Зміна пріоритетів замовника у процесі розробки	Висока	Середня
Р17	Операційний	Недоступність ключового учасника команди (хвороба, звільнення)	Висока	Низька
Р18	Операційний	Збій хмарного хостингу (Railway/Render)	Середня	Низька
Р19	Операційний	Перевищення планового бюджету проєкту	Висока	Середня
Р20	Операційний	Відмова партнерів або зовнішніх підрядників	Середня	Низька
Р21	Операційний	Зростання вартості зовнішніх API-сервісів	Середня	Низька
Р22	Операційний	Відключення електроенергії (блекаути)	Висока	Низька

Закінчення таблиці Г.2

1	2	3	4	5
Р23	Юридичний	Порушення законодавства про захист персональних даних	Висока	Висока
Р24	Юридичний	Претензії від користувачів через некоректні поради ШІ	Середня	Середня
Р25	Юридичний	Порушення ліцензійних умов сторонніх бібліотек	Низька	Висока
Р26	Взаємодії	Негативні відгуки через баги MVP у перших користувачів	Висока	Висока
Р27	Взаємодії	Низька конверсія клієнтів у онлайн-запис (опір змінам)	Середня	Середня
Р28	Взаємодії	Відмова адміністраторів від використання нової системи	Середня	Висока
Р29	Форс-мажор	Безпекові ризики (воєнний стан, ракетні обстріли)	Висока	Низька
Р30	Форс-мажор	Різке коливання валютного курсу (вплив на бюджет у гривнях)	Середня	Низька

Таблиця Г.3

Ідентифікація ризиків проєкту

№	Ризикова подія	Затримки		Фін. витрати		Імовірність		Частота		Важливість
		ЯО	КО	ЯО	КО	ЯО	КО	ЯО	КО	
1	2	3	4	5	6	7	8	9	10	11
Р01	Критичні баги після релізу	ВС	8	ВС	8	ВС	8	ВС	8	4096
Р02	Складнощі інтеграції Chatwoot	ВС	8	СС	5	ВН	7	СС	5	1400
Р03	Якість ШІ-асистента	СН	4	СН	4	ВН	7	СН	4	448
Р04	Продуктивність на старих пристроях	СС	5	СН	4	ВН	7	СС	5	700

Закінчення таблиці Г.3

1	2	3	4	5	6	7	8	9	10	11
P05	Конфлікт синхронізації даних	ВС	8	ВС	8	СС	5	СС	5	1600
P06	Несумісність бібліотек	СС	5	СС	5	СН	4	СН	4	400
P07	Збій LLM API	СС	5	СН	4	СН	4	СН	4	320
P08	Відмова App Store / Google Play	ВВ	9	ВН	7	НС	2	НН	1	126
P09	Збій платіжного шлюзу	ВН	7	ВН	7	СН	4	СН	4	784
P10	Втрата даних БД	ВВ	9	ВВ	9	НС	2	НН	1	162
P11	Scope creep	ВС	8	ВС	8	ВС	8	ВС	8	4096
P12	Відсутність документації	СС	5	СС	5	ВН	7	СС	5	875
P13	Недооцінка трудовитрат	ВС	8	ВС	8	ВН	7	ВН	7	3136
P14	Конфлікти в команді	СС	5	СН	4	СН	4	СН	4	320
P15	Нечіткі вимоги замовника	ВС	8	ВС	8	ВН	7	СС	5	2240
P16	Зміна пріоритетів замовника	ВН	7	СС	5	ВС	8	СС	5	1400
P17	Недоступність учасника	ВС	8	СС	5	СН	4	СС	5	800
P18	Збій хостингу	СС	5	СС	5	СН	4	СН	4	400
P19	Перевищення бюджету	ВС	8	ВС	8	СН	4	СН	4	1024
P20	Відмова підрядників	СС	5	СС	5	НС	2	НС	2	100
P21	Зростання вартості API	СН	4	СС	5	СН	4	СН	4	320
P22	Відключення електроенергії	СС	5	СС	5	ВС	8	ВН	7	1400
P23	Витік персональних даних	НС	2	ВВ	9	НС	2	НН	1	36
P24	Претензії через ШІ-поради	СН	4	СС	5	НС	2	НС	2	80
P25	Ліцензійні порушення	СН	4	СС	5	НН	1	НН	1	20
P26	Негативні відгуки MVP	ВН	7	ВС	8	ВН	7	СС	5	1960
P27	Низька конверсія клієнтів	СС	5	ВН	7	СС	5	СС	5	875
P28	Опір адміністраторів	СС	5	ВН	7	СС	5	СН	4	700
P29	Безпекові ризики (війна)	ВВ	9	ВВ	9	ВВ	9	ВВ	9	6561
P30	Коливання валютного курсу	СН	4	СС	5	ВС	8	СС	5	800

Таблиця Г.4

Протиризикові заходи (план реагування)

№	Ризикова подія	ПРЗ 1: Профілактика	Симптом (рання ознака)	ПРЗ 2: При симптомах	ПРЗ 3: При проблемі
1	2	3	4	5	6
Р01	Критичні баги після релізу	Code review кожного PR; покриття тестами >80%; регресійне тестування перед релізом	Зростання кількості дефектів у спринті; сповіщення від моніторингу Sentry	Залучення другого розробника до дебагу; пріоритизація виправлення Р0-дефектів	Відкат до попередньої стабільної версії; екстрений хотфікс; сповіщення користувачів
Р02	Складнощі інтеграції Chatwoot	Тестування API Chatwoot на етапі планування; розробка mock-об'єктів для незалежної розробки	Відсутність коректної відповіді API понад 48 год; помилки у webhook	Перехід на альтернативний агрегатор (Respond.io); збільшення часового буфера спринту	Реалізація спрощеної версії inbox без агрегатора; відтермінування функції до наступного релізу
Р03	Якість ШІ-асистента	Тестування промптів на реальних прикладах звернень; логування всіх запитів до LLM	Точність розпізнавання нижче 80% за результатами логів	Ітеративне налаштування промптів; звуження переліку підтримуваних намірів	Відключення ШІ-підказок; перехід до повністю ручної обробки звернень адміністратором
Р04	Продуктивність на старих пристроях	Тестування на пристроях з характеристиками нижче середніх; профілювання продуктивності	Час завантаження екранів >3 сек на тестових пристроях	Оптимізація рендерингу компонентів; ліниве завантаження зображень	Встановлення мінімальних системних вимог; виключення підтримки пристроїв нижче порогу
Р05	Конфлікт синхронізації даних	Реалізація транзакційних операцій у БД; версіонування API; тести на паралельні запити	Дублювання записів або невідповідність даних між клієнтом і сервером	Примусова синхронізація клієнта з сервером; аналіз лога транзакцій	Відновлення даних з резервної копії; тимчасове переведення у режим читання

Продовження таблиці Г.2

1	2	3	4	5	6
Р06	Несумісність бібліотек	Фіксація версій залежностей у package.json; регулярний аудит сумісності	Помилки збірки після оновлення iOS/Android SDK	Тимчасове блокування оновлення SDK; пошук сумісних версій бібліотек	Міграція на підтримувану альтернативну бібліотеку; відтермінування оновлення платформи
Р07	Збій LLM API	Реалізація fallback-логіки: при недоступності API ШІ-підказки не відображаються	Таймаут запитів до LLM API понад 5 сек	Переключення на резервний LLM-провайдер (Claude↔GPT)	Повне відключення модуля ШІ; адміністратор працює без підказок
Р08	Відмова App Store / Google Play	Дотримання гайдлайнів Apple і Google з першого дня; тестування на відповідність вимогам	Отримання листа з вимогами доопрацювань	Внесення необхідних змін протягом 5 робочих днів	Публікація веб-версії адмін-панелі як тимчасової альтернативи
Р09	Збій платіжного шлюзу	Тестування webhook-обробника LiqPay; ідемпотентна обробка платіжних подій	Відсутність підтвердження оплати протягом 10 хв	Ручна перевірка статусу транзакції в адмін-панелі LiqPay	Тимчасове відключення онлайн-оплати; перехід на готівковий розрахунок
Р10	Втрата даних БД	Автоматичне резервне копіювання щодобово; зберігання резервних копій у хмарному сховищі	Помилки цілісності даних; неочікувана зупинка СУБД	Переключення на резервний сервер; аналіз лога БД	Відновлення з останньої резервної копії; повідомлення замовника про втрачені дані
Р11	Scope creep	Підписання переліку вимог замовником; формальний процес управління змінами; контрольна форма на кожну нову вимогу	Збільшення кількості нових запитів від замовника понад 2 на спринт	Оцінка впливу змін на терміни і бюджет до прийняття; переговори щодо пріоритетів	Перегляд пріоритетів беклогу; виключення Could have; документування відстрочених вимог

Продовження таблиці Г.2

1	2	3	4	5	6
Р13	Недооцінка трудовитрат	Planning Poker для оцінки Story Points; аналіз velocity минулих спринтів; буфер 20% у плані	Фактичний velocity нижче планового два спринти підряд	Перегляд обсягу поточного спринту; перерозподіл задач	Перегляд загального плану проєкту; збільшення тривалості або скорочення функціоналу
Р14	Конфлікти в команді	Чіткий розподіл зон відповідальності; ретроспективи з анонімним зворотним зв'язком	Погіршення комунікації; затримки через неузгодженість рішень	Медіація менеджера проєкту; перерозподіл задач	Зміна складу команди або структури взаємодії
Р15	Нечіткі вимоги замовника	Story Writing Workshop на початку проєкту; фіксація вимог у підписаному документі	Команда регулярно уточнює деталі з замовником понад планові зустрічі	Додаткові зустрічі для прояснення вимог; деталізація User Stories	Залучення бізнес-аналітика для повторного аналізу вимог; коригування беклогу
Р16	Зміна пріоритетів замовника	Фіксація пріоритетів у беклозі; правило: зміни пріоритетів — тільки між спринтами	Замовник запитує зміни у поточному спринті	Оцінка впливу; узгодження перенесення до наступного спринту	Екстрений перегляд беклогу; компенсація за рахунок резервного часу
Р17	Недоступність учасника	Крос-функціональне навчання; документація коду; розподіл критичних знань між учасниками	Відсутність понад 3 робочих дні без попередження	Перерозподіл задач; зменшення обсягу спринту	Залучення зовнішнього підрядника на час відсутності
Р18	Збій хостингу	Вибір надійного провайдера з SLA 99,9%; налаштування моніторингу uptime	Downtime понад 30 хвилин; сповіщення UptimeRobot	Переключення на резервний регіон хостингу	Міграція на альтернативного провайдера (Railway→Render)
Р19	Перевищення бюджету	Щотижневий моніторинг витрат; резервний фонд 10%; MoSCoW-пріоритизація	Фактичні витрати перевищують план на 7%	Скорочення функціоналу Could have; оптимізація операційних витрат	Переговори з замовником щодо збільшення бюджету або скорочення обсягу

Продовження таблиці Г.2

1	2	3	4	5	6
P20	Відмова підрядників	Попередня перевірка надійності підрядника; контрактна відповідальність	Підрядник не виходить на зв'язок понад 2 дні	Пошук альтернативного підрядника; виконання задачі власними силами	Реорганізація плану спринту; компенсація за рахунок резервного часу
P21	Зростання вартості API	Встановлення лімітів витрат у налаштуваннях API-провайдерів	Місячні витрати на API перевищують план на 30%	Оптимізація частоти запитів; кешування результатів	Перехід на дешевший провайдер або self-hosted модель
P22	Відключення електроенергії	Дистанційна робота з різних міст; хмарне середовище розробки (GitHub Codespaces)	Оголошення графіку відключень понад 6 год/добу	Перехід на автономне живлення (павербанки, генератор); зміна графіку роботи	Тимчасове призупинення активної розробки; перехід до аналітичних і документаційних задач
P23	Витік персональних даних	Шифрування HTTPS; хешування паролів; аудит безпеки перед релізом; мінімізація даних	Підозрілі аномальні запити до API; сповіщення від WAF	Блокування підозрілих IP; примусовий вихід усіх користувачів; аналіз лога	Сповіщення регулятора та користувачів; виправлення вразливості; повторний аудит
P24	Претензії через ШІ-поради	Відмова від відповідальності в умовах використання; підказки ШІ є рекомендаційними	Надходження скарг від користувачів на некоректні підказки	Вимкнення конкретного сценарію підказок; уточнення дисклеймера	Консультація юриста; обмеження функціоналу ШІ до безпечних сценаріїв
P25	Ліцензійні порушення	Аудит ліцензій усіх бібліотек перед використанням; використання тільки MIT/Apache-ліцензованих компонентів	Виявлення бібліотеки з несумісною ліцензією	Заміна на ліцензійно сумісний аналог	Видалення функціоналу, що використовує порушену бібліотеку

Закінчення таблиці Г.2

1	2	3	4	5	6
P26	Негативні відгуки MVP	Закрите бета-тестування перед публічним релізом; збір зворотного зв'язку від тестових користувачів	Рейтинг додатку нижче 3,5 зірки в перший тиждень	Термінове виправлення топ-3 проблем за відгуками; публічна відповідь на рецензії	Тимчасове зняття додатку з публікації; значний патч-реліз
P27	Низька конверсія клієнтів	UX-тестування з реальними клієнтами салону до релізу; навчальні матеріали для клієнтів	Частка онлайн-записів нижче 40% після місяця роботи	Спрощення потоку запису; push-кампанія для залучення	Додаткове навчання персоналу салону; зміна onboarding-сценарію
P28	Опір адміністраторів	Залучення адміністраторів до тестування системи на етапі UAT; навчання і демонстрація переваг	Адміністратори продовжують вести записи вручну паралельно із системою	Додаткові навчальні сесії; спрощення адмін-інтерфейсу	Тимчасовий гібридний режим (паралельне ведення записів); поступовий перехід
P29	Безпекові ризики (війна)	Дистанційна робота; резервні сервери за кордоном; резервні копії поза Україною	Оголошення повітряних тривог; евакуаційні попередження	Перехід в укриття; зупинка синхронних нарад; асинхронна комунікація	Призупинення проєкту; передача критичних задач учасникам за кордоном; активація плану безперервності
P30	Коливання валютного курсу	Фіксація вартості API-послуг у гривнях у договорах; резервний фонд	Курс долара зростає понад 10% відносно планового	Перегляд бюджету на зовнішні сервіси; оптимізація витрат	Переговори з замовником щодо коригування бюджету; скорочення використання платних API