Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра автоматизації та інтелектуальних інформаційних технологій

Бакалаврська кваліфікаційна робота на тему:

«Автоматизація процесу запису на сервісний центр»

Виконав: студент IV курсу, групи 1АКІТ-21б
спеціальності 151 – Автоматизація та
комп'ютерно-інтегровані технології
(шифр і назва спеціальності)
Лісниченко С. А.
(прізвище та ініціали)

Керівник: к.т.н., доцент кафедри АІІТ
Гармаш В. В.
(прізвище та ініціали)
«18» 06 2025 р.

Рецензент: Юхимчук М.С.
(прізвище та ініціали)
«19» 06 2025 р.

Допущено до захисту
завідувач кафедри АІІТ
«20» 06 2025 р.

Вінниця ВНТУ – 2025 рік

Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра автоматизації та інтелектуальних інформаційних технологій
Рівень вищої освіти перший (бакалаврський)
Галузь знань 15 – Автоматизація та приладобудування
Спеціальність 151 – Автоматизація та комп'ютерно-інтегровані технології
Освітньо-професійна програма Інтелектуальні комп'ютерні системи управління

ЗАТВЕРДЖУЮ
завідувач кафедри АІІТ
д.т.н., проф. Бісікало О. В.
«20» 06 2025 року

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

Лісниченку Сергію Андрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи «Автоматизація процесу запису на сервісний центр»
керівник роботи Гармаш Володимир Володимирович, к.т.н, доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від «20» березня 2025 року №97
2. Термін подання студентом роботи 10.06.2025
3. Вихідні дані до роботи: - Платформа - Android / iOS / Web; Режим - Telegram-бот; Сервер - Linux, macOS, Windows (WSL2); ПЗ - Bun, TypeScript, Telegraf, Drizzle, SQLite; ОЗП - від 2 ГБ; Диск - від 500 МБ; Інтернет - стабільне з'єднання; Клієнт - Telegram; Безпека - Telegram + локальна БД.
4. Зміст текстової частини (перелік питань, які потрібно розробити) дослідження галузевої сфери та аналіз конкурентоспроможних варіантів, обгрунтування архітектурних рішень і відбір технологічного стеку для побудови системи, створення UML схем, розробка програмного продукту та його тестування
5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень) UML схема компонентної структури, UML схема процесів, UML схема взаємодії, демонстраційні скріншоти створеного програмного продукту, проектна презентація

6. Консультанти розділів роботи

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

7. Дата видачі завдання 20.03.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Термін виконання		Примітка
		початок	закінчення	
1	Вибір, узгодження та затвердження теми БКР	01.02.2025	11.02.2025	виконано
2	Аналіз предметної області та конкурентних рішень. Опрацювання літературних джерел.	12.02.2025	14.03.2025	виконано
3	Постановка задач для вирішення в БКР	17.03.2025	21.03.2025	виконано
4	Проектування архітектури системи, вибір технологій розробки	24.03.2025	04.04.2025	виконано
5	Розробка програмного забезпечення	07.04.2025	09.05.2025	виконано
6	Тестування програмного забезпечення	12.05.2025	16.05.2025	виконано
7	Оформлення пояснювальної записки та графічної частини	19.05.2025	23.05.2025	виконано
8	Попередній захист, доопрацювання, та рецензування БКР	26.05.2025	30.05.2025	виконано
9	Захист БКР	16.06.2025	20.06.2025	виконано

Студент _____ Лісниченко С. А.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Гармаш В. В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Бакалаврська кваліфікаційна робота складається з 82 сторінок формату А4, включаючи додатки, містить 15 рисунків та список використаних джерел із 16 найменувань.

Дана бакалаврська робота присвячена розробці автоматизованої системи управління записами на обслуговування у вигляді Telegram-бота. Система реалізована з використанням мови програмування TypeScript у середовищі Bun, з застосуванням фреймворку Telegraf для роботи з Telegram API та ORM Drizzle для взаємодії з базою даних SQLite.

У роботі проведено аналіз предметної області, розглянуто існуючі рішення, спроєктовано архітектуру системи, концептуальну та логічну модель бази даних, а також реалізовано інтерфейс взаємодії користувача через бот. Основні функції включають реєстрацію за номером телефону, вибір послуг та спеціалістів, бронювання часових слотів, перегляд і скасування записів, а також панель адміністратора.

Ключові слова: Telegram-бот, автоматизація запису, керування бронюванням, TypeScript, Bun, Telegraf, Drizzle ORM, SQLite.

ABSTRACT

The bachelor's qualification thesis consists of 82 A4 pages, including appendices, contains 15 figures, and a list of 16 references.

This thesis is dedicated to the development of an automated appointment management system in the form of a Telegram bot. The system is implemented using the TypeScript programming language in the Bun runtime environment, with the Telegraf framework for interacting with the Telegram API and the Drizzle ORM for managing a SQLite database.

The work includes an analysis of the subject area, a review of existing solutions, the design of the system architecture, conceptual and logical data models, and the implementation of a user interface through the bot. The core features include user registration via phone number, service and specialist selection, time slot booking, appointment review and cancellation, as well as an administrator panel.

Keywords: Telegram bot, appointment automation, booking management, TypeScript, Bun, Telegraf, Drizzle ORM, SQLite.

ЗМІСТ

ВСТУП

Актуальність. У сучасному цифровому середовищі, що стрімко розвивається, спостерігається зростаюча потреба в автоматизації процесів взаємодії між клієнтами та сервісними організаціями. Особливо це актуально для невеликих бізнесів, сервісних центрів, клінік, майстерень та інших установ, де щоденне обслуговування клієнтів пов’язане із попереднім записом. Ручне ведення розкладів, телефонне бронювання або використання складних CRM-систем часто є неефективними або занадто затратними як з точки зору ресурсів, так і часу.

Одним із рішень цієї проблеми є створення чат-ботів — програмних агентів, які автоматизують спілкування з клієнтом у звичному для нього середовищі, зокрема у месенджерах. Telegram надає відкритий API і широкі можливості для інтеграції, що робить його однією з найпривабливіших платформ для створення ботів. Telegram-боти не потребують встановлення додаткових додатків, працюють швидко та доступні на будь-якому пристрої, що суттєво знижує бар’єр входу як для користувачів, так і для бізнесу.

Водночас, розробка подібної системи вимагає грамотного проєктування, організації бази даних, побудови інтуїтивного сценарію взаємодії та забезпечення надійної логіки керування записами. Сучасні інструменти, такі як Bun, TypeScript, Telegraf, Drizzle ORM та SQLite, дозволяють створювати високопродуктивні рішення, що відповідають цим вимогам.

Таким чином, розробка Telegram-бота для управління попереднім записом є актуальним завданням, яке дозволяє вирішити низку практичних проблем автоматизації, підвищити якість обслуговування клієнтів і зменшити навантаження на персонал.

Метою дослідження є підвищення ефективності процесу управління записами на обслуговування шляхом впровадження автоматизованої системи у

вигляді Telegram-бота, що забезпечує зручну та доступну взаємодію між користувачами й сервісним центром.

Для досягнення цієї мети визначено такі основні завдання:

- здійснити аналіз предметної області та огляд аналогічних рішень;
- побудувати архітектуру програмного забезпечення;
- спроєктувати концептуальну, логічну та фізичну моделі бази даних;
- реалізувати сценарії взаємодії користувача з ботом;
- створити механізми реєстрації, вибору послуг, бронювання та підтвердження запису;
- забезпечити адміністрування даних користувачів, послуг і спеціалістів;
- провести тестування системи та оцінити її ефективність.

Об’єктом дослідження є процес автоматизації обслуговування клієнтів засобами цифрової комунікації.

Предметом дослідження є програмні засоби, архітектура та алгоритмічні рішення для реалізації Telegram-бота з функціональністю керування записами.

Практична цінність полягає у створенні повноцінної системи, яка дозволяє суттєво підвищити ефективність керування записами на послуги, покращити комунікацію з клієнтами та забезпечити масштабованість рішення для впровадження в різні сфери обслуговування.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

У рамках даного розділу проводиться аналіз предметної області автоматизації запису в автосервісні центри, що спрямований на виявлення основних проблем традиційних підходів до керування записом клієнтів та визначення напрямів удосконалення за допомогою сучасних IT-рішень. Оскільки ефективність обслуговування напряму залежить від зручності, швидкості та надійності процесу запису, важливо дослідити як існуючі технології, так і специфіку бізнес-процесів у сфері автосервісу. У цьому контексті особливу увагу приділено Telegram Bot API як платформі, що відповідає вимогам мобільності, інтерактивності та простоти впровадження.

Метою дослідження є визначення функціональних і нефункціональних вимог до програмного рішення, здатного автоматизувати процес запису клієнтів та оптимізувати взаємодію з персоналом автосервісу. З цією метою розглядаються наступні аспекти: аналіз поточних проблем ручного або напівавтоматизованого запису, огляд існуючих засобів автоматизації у сфері послуг, вибір архітектурного підходу до реалізації системи, постановка задачі розробки з формалізацією технічних вимог і очікуваних результатів.

Послідовний розгляд зазначених аспектів дозволяє сформувати цілісне уявлення про потреби галузі, обґрунтувати вибір технологій і закласти основу для реалізації ефективного та адаптивного інструменту обслуговування клієнтів автосервісу.

1.1 Аналіз проблематики автоматизації запису на обслуговування

У сучасному світі стрімкого розвитку інформаційних технологій та постійно зростаючих очікувань споживачів щодо якості та доступності послуг,

автоматизація процесів взаємодії з клієнтами стає не просто конкурентною перевагою, а критично необхідною умовою успішного функціонування будь-якого підприємства сфери обслуговування. Традиційні методи організації запису на послуги, що базуються переважно на телефонному зв'язку та особистому відвідуванні, демонструють свою неефективність у контексті сучасних вимог до швидкості, зручності та доступності сервісів.

Аналізуючи поточний стан систем запису на обслуговування в різних галузях економіки України, можна виділити ряд системних проблем, які негативно впливають на ефективність бізнес-процесів та рівень задоволеності клієнтів. Основною проблемою є обмеженість часових рамок доступності традиційних каналів зв'язку. Більшість підприємств працює за стандартним графіком з 9:00 до 18:00, що створює значні незручності для працюючих клієнтів, які не мають можливості зателефонувати або особисто відвідати заклад в робочий час.

Людський фактор відіграє критичну роль у виникненні помилок під час процесу запису. Оператори можуть неправильно зрозуміти або записати інформацію про клієнта, переплутати дати та час, що призводить до конфліктних ситуацій і псування репутації компанії. Крім того, навантаження на персонал, який займається прийомом дзвінків, може бути нерівномірним, що призводить до неефективного використання робочого часу та додаткових витрат на оплату праці.

Відсутність централізованої системи обліку записів створює додаткові складнощі у випадку роботи декількох точок обслуговування або філій. Інформація може бути розпорошена між різними журналами, електронними таблицями або локальними базами даних, що ускладнює координацію роботи та може призводити до дублювання записів або конфліктів у розкладі.

Відсутність автоматизованих нагадувань про заплановані візити призводить до збільшення кількості пропущених сеансів ("no-show"), що негативно впливає на ефективність використання робочого часу спеціалістів та загальну прибутковість бізнесу. За даними досліджень, впровадження системи автоматичних нагадувань може зменшити кількість пропущених візитів на 30-50%.

Аналіз конкурентного середовища показує, що підприємства, які впровадили сучасні системи автоматизації запису, демонструють значно вищі показники клієнтської лояльності та ефективності операційної діяльності. Вони здатні обслуговувати більшу кількість клієнтів при менших витратах на персонал та забезпечувати більш високий рівень сервісу.

Особливої уваги заслуговує питання збору та аналізу даних про поведінку клієнтів. Традиційні методи запису не дозволяють систематично збирати інформацію про переваги клієнтів, популярність різних послуг, сезонні коливання попиту тощо. Ця інформація є критично важливою для стратегічного планування, оптимізації асортименту послуг та розробки маркетингових кампаній.

Впровадження автоматизованих систем запису дозволяє вирішити більшість зазначених проблем шляхом забезпечення цілодобової доступності, значного зменшення ймовірності помилок, оптимізації використання робочого часу персоналу, підвищення загального рівня задоволеності клієнтів та створення цінної бази даних для бізнес-аналітики.

Таким чином, автоматизація процесу запису на обслуговування є не просто технологічним удосконаленням, а стратегічним кроком, який дозволяє підприємствам адаптуватися до сучасних вимог ринку, підвищити свою конкурентоспроможність та забезпечити стійкий розвиток у довгостроковій перспективі.

1.2 Огляд існуючих рішень для управління записами

У цьому підрозділі здійснюється стислий аналіз наявних технологічних рішень для автоматизації запису на послуги, зокрема веб-платформ, мобільних застосунків та чат-ботів. Розглядаються їхні переваги, недоліки та можливість адаптації до потреб автосервісної сфери.

1.2.1 Веб-платформи для онлайн-запису

Веб-платформи для онлайн-запису представляють один з найрозвинутіших та найпоширеніших підходів до автоматизації процесу бронювання послуг у сучасному цифровому середовищі. Ці рішення еволюціонували від простих контактних форм до складних систем управління взаємовідносинами з клієнтами, інтегрованих з корпоративними інформаційними системами.

Acuity Scheduling (рисунок 1.1), що належить компанії Squarespace, є одним з лідерів ринку веб-платформ для онлайн-бронювання. Платформа пропонує широкий спектр функцій, включаючи гнучкі налаштування календаря з можливістю створення різних типів зустрічей, автоматичні email та SMS нагадування, інтеграцію з популярними платіжними системами PayPal, Stripe та Square. Особливістю Acuity є підтримка групових сеансів, можливість створення анкет для клієнтів перед візитом та розгорнута система звітності з аналітикою бізнес-показників.

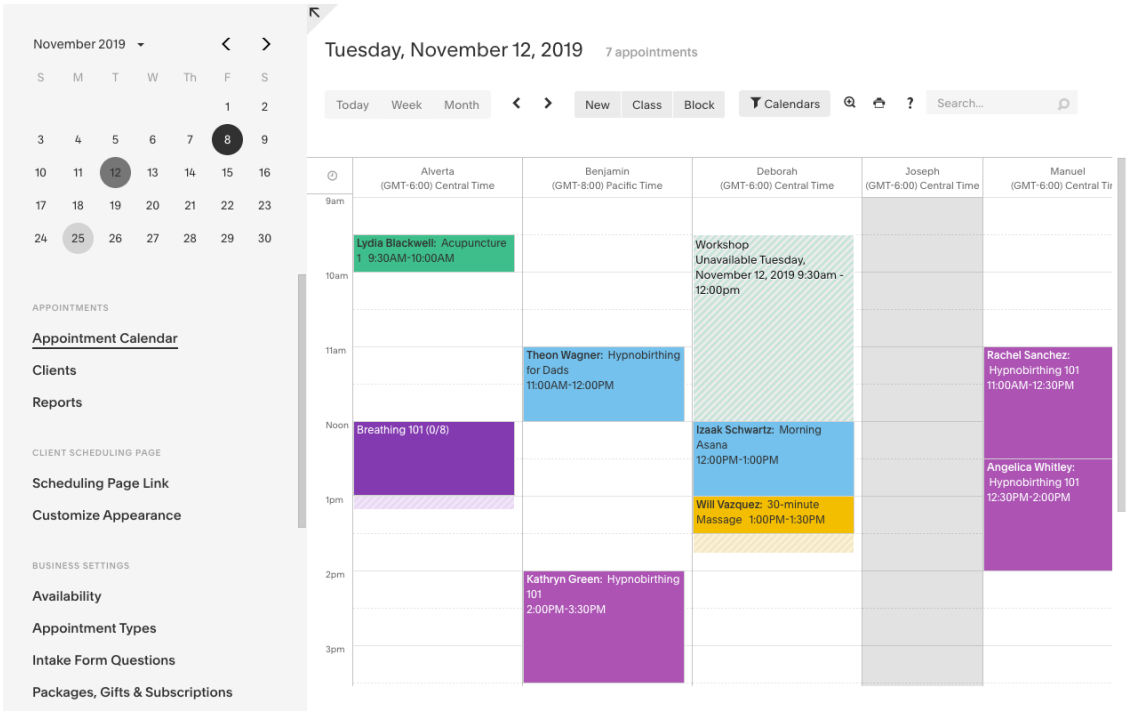

Рисунок 1.1 – Приклад інтерфейсу Acuity Schedulin

Calendly (рисунок 1.2) зарекомендував себе як одне з найпростіших у використанні рішень для індивідуальних консультацій та ділових зустрічей. Платформа автоматично синхронізується з Google Calendar, Outlook, Office 365 та iCloud, забезпечуючи актуальність інформації про зайнятість користувача. Calendly підтримує інтеграцію з понад 70 додатками, включаючи Zoom, Microsoft Teams, Salesforce та HubSpot, що робить його ідеальним рішенням для корпоративного середовища.

Рисунок 1.2 – Приклад інтерфейсу Calendly

SimplyBook.me позиціонує себе як універсальна платформа для будь-яких типів бізнесу, що потребують системи бронювання. Рішення включає понад 50 додаткових модулів (рисунок 1.3) , які можна активувати за потреби, включаючи багатомовну підтримку (понад 30 мов), інтеграцію з Facebook та Google, систему

управління ресурсами (обладнання, приміщення), автоматизацію маркетингових кампаній та розгорнуту CRM-систему.

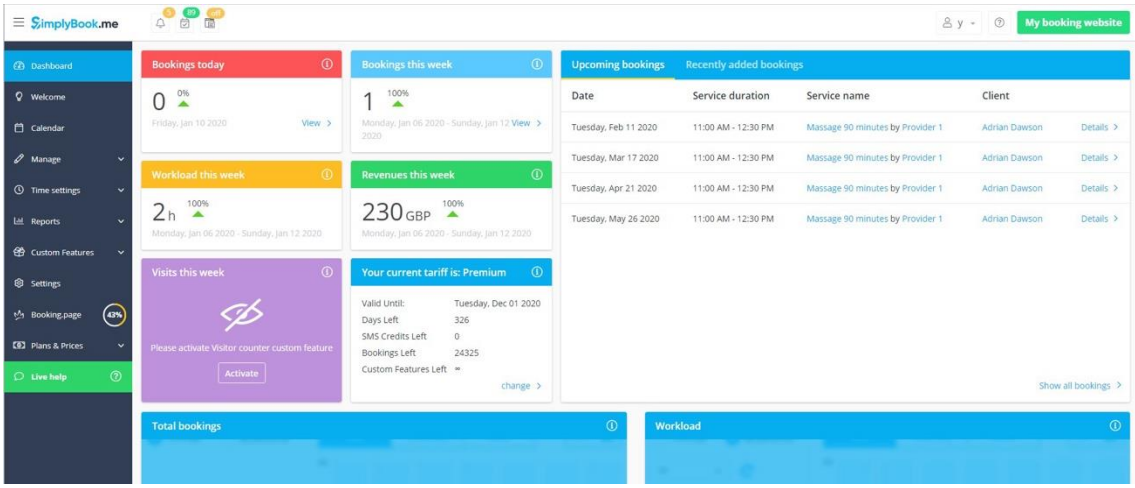

Рисунок 1.3 – Приклад інтерфейсу SimplyBook.me

Основними технічними перевагами веб-платформ є їх доступність з будь-якого пристрою з інтернет-з'єднанням та веб-браузером, що забезпечує кросплатформенність без необхідності розробки окремих версій для різних операційних систем. Веб-технології дозволяють створювати багаті користувацькі інтерфейси з використанням сучасних фреймворків та бібліотек, забезпечуючи високий рівень інтерактивності та візуальної привабливості.

Можливості інтеграції з корпоративними системами через REST API, веб-сервіси та webhooks дозволяють веб-платформам органічно вписуватися в існуючу IT-інфраструктуру підприємства. Це включає синхронізацію з ERP-системами, CRM-платформами, системами електронного документообігу та фінансовими додатками.

Однак веб-платформи мають і суттєві недоліки. Необхідність розробки або значного налаштування веб-сайту може бути коштовною та часозатратною, особливо для малого бізнесу з обмеженим IT-бюджетом. Процес розробки може тривати від кількох тижнів до кількох місяців, залежно від складності вимог.

Функціональність веб-платформ на мобільних пристроях може бути обмеженою порівняно з нативними мобільними додатками. Хоча сучасні адаптивні веб-дизайни забезпечують прийнятний досвід використання на смартфонах та планшетах, вони не можуть повністю використовувати специфічні можливості мобільних платформ, такі як push-нотифікації, інтеграція з контактами або календарем пристрою.

Залежність від стабільності інтернет-з'єднання може бути критичною проблемою в регіонах з нестабільним інтернетом. На відміну від нативних додатків, які можуть зберігати деякі дані локально, веб-платформи зазвичай потребують постійного з'єднання для повноцінного функціонування.

1.2.2 Мобільні додатки для бронювання послуг

Мобільні додатки представляють потужну альтернативу веб-платформам, особливо в контексті зростаючої мобілізації суспільства та збільшення часу, який користувачі проводять з мобільними пристроями. За даними статистичних досліджень, середній користувач смартфона витрачає понад 7 годин на день, взаємодіючи з мобільним пристроєм, що робить мобільні додатки природним каналом для взаємодії з сервісами бронювання.

OpenTable революціонізував сферу ресторанного бізнесу, створивши екосистему, яка об'єднує ресторани та відвідувачів через інтуїтивний мобільний інтерфейс (рисунок 1.4).

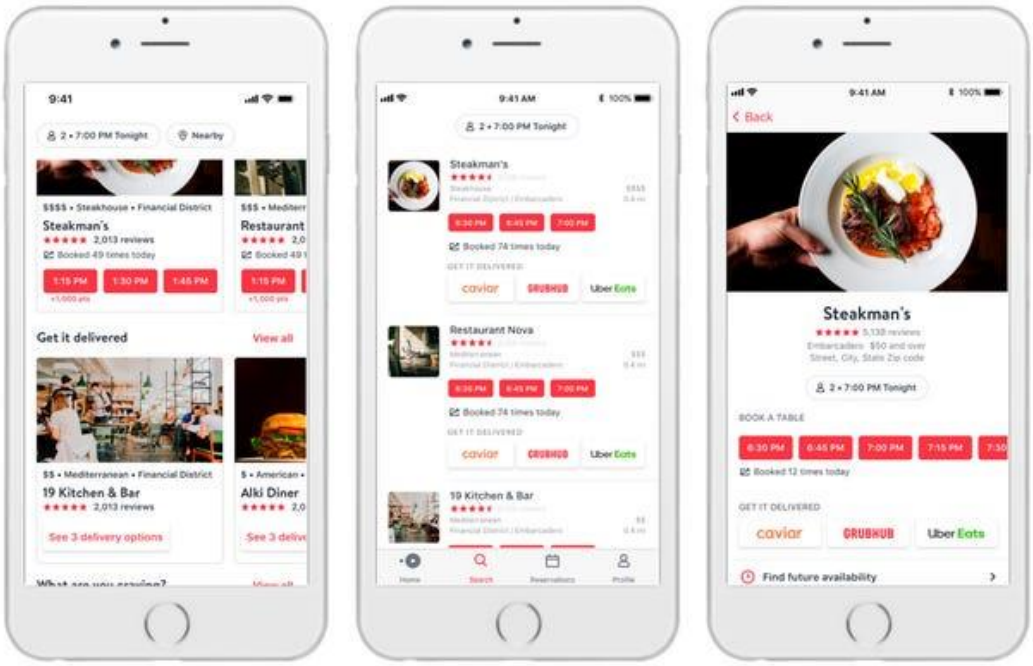

Рисунок 1.4 – Приклад інтерфейсу OpenTable

Додаток інтегрує передові геолокаційні сервіси для автоматичного визначення ресторанів поблизу користувача, систему рейтингів та детальних відгуків від реальних відвідувачів, фотогалереї страв та інтер'єрів, меню з цінами та програму лояльності OpenTable Points. Технологічна архітектура додатка включає машинне навчання для персоналізації рекомендацій та predictive analytics для оптимізації завантаженості ресторанів.

У сфері краси та здоров'я мобільні додатки, такі як StyleSeat (США) та Treatwell (Європа), спеціалізуються на бронюванні послуг салонів краси, спа-центрів та барбершопів. Ці платформи включають специфічні функції, такі як портфоліо робіт майстрів, деталізовані описи процедур, можливість вибору конкретного спеціаліста та інтеграцію з соціальними мережами для демонстрації результатів.

Технічні переваги мобільних додатків включають нативну інтеграцію з операційною системою пристрою, що забезпечує швидший доступ до функцій через іконку на домашньому екрані. Push-нотифікації працюють більш ефективно

порівняно з email-повідомленнями, забезпечуючи вищі показники відкриття (до 90% порівняно з 20-25% для email) та миттєве отримання користувачем інформації про підтвердження бронювання, зміни в розкладі або спеціальні пропозиції.

Інтеграція з нативними функціями пристрою відкриває широкі можливості для покращення користувацького досвіду. Це включає автоматичне додавання подій до календаря пристрою, використання контактної книги для автозаповнення форм, інтеграцію з картографічними сервісами для прокладання маршрутів до місця обслуговування, використання камери для сканування QR-кодів або створення фотографій.

Офлайн-функціональність дозволяє користувачам переглядати історію своїх записів, контактну інформацію закладів та базову інформацію про послуги навіть без активного інтернет-з'єднання. Дані синхронізуються автоматично при відновленні з'єднання.

Продуктивність нативних додатків зазвичай вища порівняно з веб-платформами, оскільки вони мають прямий доступ до ресурсів пристрою та не залежать від продуктивності веб-браузера. Це особливо помітно при роботі з графічно насиченими інтерфейсами або при обробці великих обсягів даних.

Однак розробка мобільних додатків має суттєві недоліки, головним з яких є необхідність створення окремих версій для різних мобільних платформ. iOS та Android мають різні мови програмування (Swift/Objective-C для iOS, Java/Kotlin для Android), різні інструменти розробки, різні принципи дизайну та різні процедури публікації. Це призводить до значного збільшення витрат на розробку та підтримку - зазвичай у 2-3 рази порівняно з веб-рішеннями.

Користувачі повинні активно шукати, завантажувати та встановлювати додаток, що може стати значним бар'єром для входження, особливо якщо сервіс використовується нерегулярно або вперше. Статистика показує, що більше 25% завантажених додатків використовуються лише один раз після встановлення.

Необхідність регулярних оновлень для підтримки сумісності з новими версіями операційних систем створює постійні технічні та фінансові витрати. Apple

та Google регулярно випускають нові версії iOS та Android з новими API та вимогами безпеки, що може вимагати значних змін у коді додатка.

Мобільні додатки займають місце в пам'яті пристрою, що може бути критичним фактором для користувачів з пристроями, що мають обмежену внутрішню пам'ять. Крім того, додатки споживають батарею пристрою, особливо якщо вони працюють у фоновому режимі або використовують GPS-навігацію.

1.2.3 Чат-боти та месенджер-платформи

Чат-боти та рішення на базі месенджерів представляють інноваційний підхід до автоматизації процесу запису, який набирає все більшої популярності завдяки природності взаємодії та зниженню бар'єрів для користувачів. Цей підхід базується на використанні платформ, якими користувачі вже активно користуються щодня для особистого спілкування, що робить взаємодію з бізнес-сервісами більш органічною та зручною.

Технологічна архітектура Facebook Messenger ботів (рисунок 1.5) включає Natural Language Processing (NLP) через wit.ai, що дозволяє ботам розуміти природну мову користувачів та реагувати на неструктуровані запити. Платформа також підтримує інтеграцію з Facebook Payments для здійснення транзакцій безпосередньо в чаті та broadcast messaging для масової розсилки повідомлень з дотриманням політики Facebook щодо спаму.

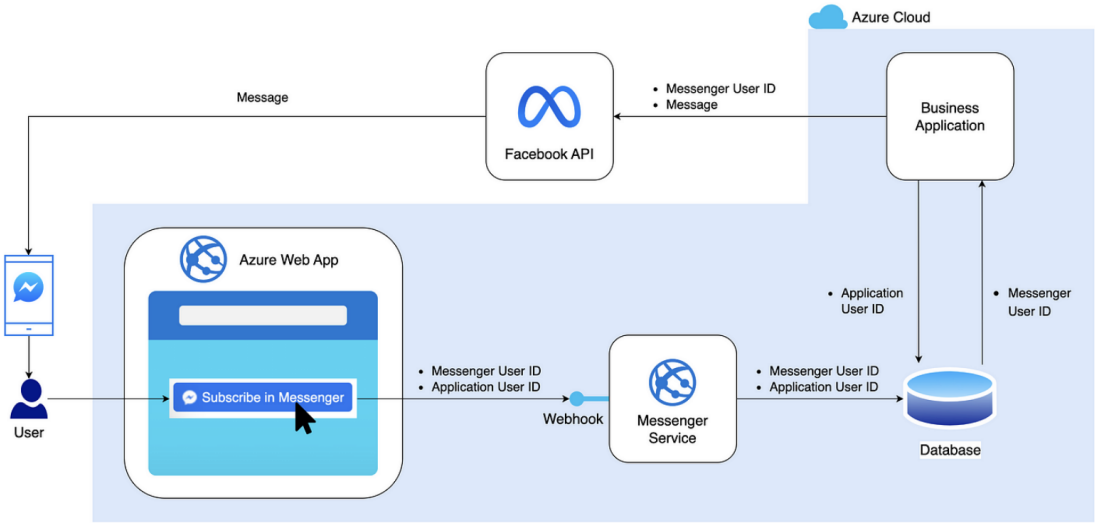

Рисунок 1.5 – Технологічна архітектура Facebook Messenger ботів

WhatsApp Business API, незважаючи на більш обмежені можливості порівняно з Messenger, має критично важливу перевагу у вигляді надзвичайно високого рівня довіри користувачів та практично стовідсоткового рівня прочитання повідомлень. В Україні WhatsApp використовує понад 80% власників смартфонів, що робить цю платформу особливо привабливою для локального бізнесу. WhatsApp Business API підтримує structured messages для представлення каталогів товарів або послуг, session-based pricing model, що робить його економічно ефективним для бізнесу, end-to-end encryption для забезпечення приватності комунікацій.

Telegram боти пропонують найбільш широкі технічні можливості серед усіх месенджер-платформ завдяки багатій Telegram Bot API. Унікальні особливості включають inline keyboards з callback buttons для створення інтерактивних інтерфейсів, підтримку різноманітних типів медіа-контенту (фото, відео, документи, геолокація), Web Apps - революційну функцію, що дозволяє інтегрувати повноцінні веб-додатки безпосередньо в інтерфейс месенджера, inline queries для пошуку та взаємодії з ботом з будь-якого чату.

Telegram також підтримує group bots для автоматизації групових чатів, payments integration через різні платіжні системи, file sharing до 2GB на файл, що

може бути корисно для обміну документами з клієнтами. API Telegram не має суворих обмежень на частоту запитів, що дозволяє створювати високонавантажені додатки.

Slack боти знаходять широке застосування в корпоративному середовищі для автоматизації внутрішніх бізнес-процесів, включаючи бронювання переговорних кімнат, планування зустрічей команди та управління робочими ресурсами.

Основні переваги месенджер-платформ включають відсутність необхідності встановлення додаткових додатків, оскільки користувачі вже мають встановлені популярні месенджери на своїх пристроях. Це усуває один з основних бар'єрів для входження і забезпечує миттєвий доступ до сервісу.

Природна форма діалогу робить взаємодію інтуїтивно зрозумілою навіть для користувачів з мінімальними технічними навичками. Люди звикли спілкуватися через текстові повідомлення, тому взаємодія з ботом відчувається природною та комфортною.

Ефективність push-нотифікацій у месенджерах значно вища порівняно з іншими каналами комунікації. Повідомлення у месенджерах мають показники відкриття до 98% та середній час відгуку менше 90 секунд, що робить їх ідеальним каналом для критично важливих повідомлень.

Збереження історії спілкування дозволяє користувачам легко знайти інформацію про попередні записи, відтворити контекст попередніх взаємодій з сервісом та мати постійний доступ до важливої інформації без необхідності додаткових пошуків.

Економічна ефективність розробки ботів зазвичай значно вища порівняно з нативними додатками або комплексними веб-платформами. Більшість месенджер-платформ надають безкоштовні API з щедрими лімітами використання, що дозволяє малому та середньому бізнесу впроваджувати автоматизацію з мінімальними початковими інвестиціями.

Недоліки включають обмеження функціональності, що накладаються API конкретної платформи. Кожен месенджер має свої специфічні обмеження щодо

типів контенту, частоти повідомлень, можливостей інтерфейсу тощо, що може обмежувати творчий підхід до проектування взаємодії.

Залежність від політики та стабільності роботи зовнішнього сервісу створює бізнес-ризики. Зміни в API, політиці використання або технічні проблеми платформи можуть критично вплинути на роботу бота. Історія знає випадки, коли зміни в політиці месенджерів призводили до блокування тисяч ботів.

Складність реалізації комплексних візуальних інтерфейсів може потребувати креативних підходів до проектування взаємодії. Месенджери оптимізовані для текстової комунікації, тому створення складних форм введення або багатостепових процесів може бути технічно складним.

1.3 Обґрунтування вибору Telegram як платформи для реалізації

Вибір Telegram як основної платформи для реалізації системи автоматизації запису на обслуговування базується на всебічному аналізі технічних можливостей, ринкової позиції та стратегічних переваг цієї платформи в контексті українського та міжнародного ринків месенджерів.

Аналіз ринкової позиції Telegram в Україні демонструє стабільне зростання популярності месенджера, особливо серед економічно активного населення та користувачів, які активно використовують цифрові сервіси. За даними соціологічних досліджень, Telegram використовує понад 40% українських інтернет-користувачів, при цьому серед представників бізнес-спільноти та IT-сфери цей показник сягає 70-80%. Особливо важливим є той факт, що аудиторія Telegram характеризується високим рівнем цифровій грамотності та готовністю до використання інноваційних технологій.

Демографічний профіль користувачів Telegram в Україні показує переважання активної вікової групи 25-45 років, яка представляє основну цільову аудиторію для більшості видів комерційних послуг. Ця категорія користувачів має

найвищий рівень купівельної спроможності та найбільшу потребу в зручних сервісах бронювання послуг.

Технічні можливості Telegram Bot API значно перевершують можливості більшості конкурентних платформ за рівнем гнучкості та функціональності. API підтримує понад 50 різних методів взаємодії, що дозволяє створювати складні діалогові сценарії та інтерактивні інтерфейси. Особливо важливими є можливості роботи з різноманітними типами контенту: текстові повідомлення з підтримкою Markdown та HTML розмітки, фотографії та зображення з можливістю компресії та оптимізації, документи до 50 МБ, що дозволяє обмінюватися прайс-листами, договорами та іншою документацією, аудіо та відео файли для демонстрації послуг або навчальних матеріалів, геолокація для інтеграції з картографічними сервісами, контактна інформація для автоматичного збереження в адресній книзі.

Система інлайн-клавіатур Telegram дозволяє створювати інтуїтивні користувацькі інтерфейси безпосередньо в чаті без необхідності переходу на зовнішні веб-сторінки. Callback-механізм забезпечує миттєвий відгук на дії користувача та дозволяє реалізовувати складну логіку взаємодії з збереженням контексту сесії.

Безпека та приватність є фундаментальними принципами Telegram, що критично важливо для систем, що обробляють персональні дані клієнтів та комерційну інформацію. Платформа використовує власний протокол шифрування MTProto, який забезпечує end-to-end шифрування для секретних чатів та серверне шифрування для звичайних повідомлень. Telegram має відкритий вихідний код клієнтських додатків, що дозволяє незалежну перевірку рівня безпеки.

Кросплатформенність Telegram забезпечує доступність системи на всіх популярних операційних системах та типах пристроїв. Існують офіційні клієнти для iOS, Android, Windows, macOS, Linux, а також веб-версія, що працює в будь-якому сучасному браузері. Синхронізація між пристроями відбувається в реальному часі, що дозволяє користувачам розпочати взаємодію з ботом на одному пристрої та продовжити на іншому без втрати контексту.

Швидкість розробки та впровадження рішень на базі Telegram є однією з ключових переваг платформи. Готова інфраструктура месенджера усуває необхідність розробки власних систем доставки повідомлень, push-нотифікацій, автентифікації користувачів та синхронізації між пристроями. Це дозволяє розробникам зосередитись на бізнес-логіці та користувацькому досвіді, значно скорочуючи час від ідеї до готового продукту.

Telegram Bot API має мінімальні обмеження на частоту запитів (до 30 повідомлень на секунду на бота), що дозволяє створювати високонавантажені системи без необхідності складної оптимізації або використання проксі-сервісів.

Економічна ефективність розробки Telegram-ботів є суттєвою перевагою для бізнесу будь-якого масштабу. Використання Telegram Bot API є повністю безкоштовним, без обмежень на кількість користувачів або повідомлень. Витрати обмежуються лише розробкою та хостингом серверної частини додатка, що значно нижче за вартість створення власного мобільного додатка або комплексної веб-платформи.

Гнучкість архітектури та можливості масштабування Telegram ботів дозволяють розпочати з простого MVP (Minimum Viable Product) та поступово додавати функціональність відповідно до зростання потреб бізнесу. Модульна структура бота дозволяє легко додавати нові команди, інтегрувати додаткові сервіси та адаптувати систему під специфічні вимоги різних галузей.

Можливості інтеграції Telegram ботів з зовнішніми системами через webhooks, REST API та інші протоколи дозволяють створювати комплексні рішення, що органічно вписуються в існуючу IT-інфраструктуру підприємства. Це включає інтеграцію з CRM-системами, ERP-платформами, платіжними шлюзами, календарними сервісами та системами електронного документообігу.

Простота використання для кінцевих користувачів є критично важливою перевагою Telegram. Більшість користувачів месенджера вже знайомі з основними принципами взаємодії з ботами завдяки популярності різноманітних сервісних ботів у Telegram. Це мінімізує потребу в навчанні користувачів та зменшує опір до впровадження нових цифрових рішень.

Міжнародна орієнтованість Telegram також є важливим фактором для бізнесу, що планує розширення за межі України. Месенджер популярний у багатьох країнах світу, особливо в Європі, Азії та Латинській Америці, що створює потенціал для масштабування бізнес-рішень на міжнародні ринки без необхідності адаптації до локальних месенджер-платформ.

Стабільність та надійність Telegram як платформи підтверджується її історією роботи. Месенджер демонструє високий uptime та рідко має технічні проблеми, що критично важливо для бізнес-застосунків. Компанія Telegram активно інвестує в розвиток інфраструктури та регулярно випускає оновлення з новими функціями та покращеннями.

1.4 Формулювання вимог до системи

Підрозділ містить формалізовані функціональні та нефункціональні вимоги до системи запису в автосервіс. Вимоги сформовані на основі аналізу предметної області й огляду існуючих рішень, з урахуванням особливостей роботи сервісних центрів.

1.4.1 Функціональні вимоги

Система автоматизації запису на обслуговування через Telegram повинна забезпечувати комплексний набір функцій, що покривають весь життєвий цикл взаємодії з клієнтами - від першого контакту до завершення обслуговування та подальшої підтримки відносин. Детальний аналіз функціональних вимог дозволяє створити технічне завдання, що забезпечить розробку системи, здатної ефективно задовольняти потреби як клієнтів, так і бізнесу.

Підсистема реєстрації та управління користувачами повинна забезпечувати автоматичну ідентифікацію користувачів при першому зверненні до бота з використанням унікального Telegram ID. Система має зберігати профіль

користувача, включаючи ім'я та прізвище (отримані з Telegram або введені вручну), номер телефону для зв'язку та підтвердження записів, електронну пошту для додаткових комунікацій та резервних нотифікацій, історію взаємодії з системою.

Підсистема управління розкладом та доступністю є центральним компонентом системи і повинна забезпечувати точне відображення доступних часових слотів у режимі реального часу. Календарний інтерфейс має показувати заброньовані, вільні та заблоковані для бронювання періоди з можливістю перегляду на різних рівнях деталізації (день, місяць).

Для бізнесу з декількома спеціалістами система повинна підтримувати індивідуальні розклади з можливістю призначення певних послуг конкретним виконавцям.

Процес бронювання має бути максимально спрощеним та інтуїтивним, з мінімальною кількістю кроків до завершення запису. Система повинна пропонувати найближчі доступні слоти. Інтелектуальна система рекомендацій може враховувати попередні вибори користувача, популярність різних часових слотів та оптимізацію завантаженості.

Підсистема нотифікацій та комунікацій повинна забезпечувати автоматичну відправку повідомлень на всіх етапах взаємодії з клієнтом. Це включає миттєве підтвердження бронювання з деталями запису.

Модуль управління існуючими бронюваннями повинен надавати клієнтам повний контроль над їхніми записами. Функціональність включає перегляд актуальних та майбутніх записів у зручному календарному форматі, редагування деталей бронювання в межах встановлених бізнес-правил, перенесення запису на інший час з автоматичною перевіркою доступності, скасування запису з дотриманням політики скасування та повернення коштів.

Система повинна автоматично звільняти скасовані слоти та робити їх доступними для інших клієнтів, відправляти повідомлення в список очікування у випадку звільнення популярних часових слотів.

Адміністративна панель повинна забезпечувати повний контроль над усіма аспектами роботи системи через зручний веб-інтерфейс або розширений Telegram-

бот для адміністраторів. Функціональність включає управління каталогом послуг з можливістю додавання, редагування та видалення позицій, моніторинг поточних бронювань та завантаженості, управління користувачами та їхніми правами доступу.

1.4.2 Нефункціональні вимоги

Нефункціональні вимоги визначають якісні характеристики системи, які не менш важливі за функціональні можливості для забезпечення успішного впровадження та експлуатації рішення. Ці вимоги стосуються продуктивності, надійності, безпеки, зручності використання та інших аспектів, що впливають на загальний користувацький досвід та бізнес-ефективність системи.

Система має підтримувати одночасну роботу з принаймні 1000 користувачами без деградації продуктивності, з можливістю пікового навантаження до 5000 одночасних сесій. Пропускна здатність системи повинна забезпечувати обробку до 10,000 API-запитів на хвилину з можливістю автоматичного масштабування при перевищенні базових показників.

Час завантаження веб-компонентів системи (Web Apps) не повинен перевищувати 3 секунд при швидкості з'єднання 4G та 8 секунд при 3G з'єднанні. Оптимізація розміру передаваних даних має забезпечувати економне використання мобільного трафіку користувачів.

Вимоги до надійності включають забезпечення коефіцієнта доступності (uptime) не менше 99.5% протягом року, що відповідає максимум 43.8 годинам недоступності на рік. Для критично важливих періодів (робочі години, сезони високого попиту) цей показник повинен складати 99.9%.

Система повинна мати механізми автоматичного відновлення після збоїв з максимальним часом відновлення (RTO - Recovery Time Objective) не більше 15 хвилин та мінімальною втратою даних (RPO - Recovery Point Objective) не більше 5 хвилин. Це вимагає впровадження redundancy на всіх критичних рівнях архітектури.

Моніторинг системи повинен включати real-time алерти про критичні події, автоматичні health checks всіх компонентів, логування всіх операцій для аудиту та troubleshooting, метрики продуктивності та використання ресурсів.

Безпека даних та системи в цілому є пріоритетним аспектом, особливо враховуючи обробку персональних даних клієнтів та фінансових транзакцій. Вся передавана інформація повинна шифруватися з використанням протоколів TLS 1.3 або новіших версій. Зберігання чутливих даних має здійснюватися з використанням сучасних алгоритмів шифрування (AES-256).

Захист персональних даних повинен відповідати вимогам GDPR, українського Закону "Про захист персональних даних" та інших релевантних нормативних актів. Це включає право на забуття (right to be forgotten), портативність даних, інформовану згоду на обробку даних.

Зручність використання (usability) передбачає проектування інтуїтивного інтерфейсу, що не потребує спеціального навчання користувачів. Система повинна забезпечувати consistent user experience на всіх етапах взаємодії з дотриманням принципів UX/UI дизайну.

Сумісність з різними версіями Telegram клієнтів забезпечить широку підтримку користувацьких пристроїв. Система повинна коректно працювати з офіційними клієнтами Telegram для всіх популярних платформ, а також з популярними альтернативними клієнтами.

Підтримуваність коду та системи в цілому вимагає дотримання принципів clean code, SOLID принципів проектування, comprehensive documentation на всіх рівнях системи. Модульна архітектура повинна дозволяти легке внесення змін та розширення функціональності без впливу на інші компоненти.

Code coverage тестами повинен складати не менше 80% для критично важливих компонентів та 60% для загального коду. Автоматизовані тести мають включати unit tests, integration tests, end-to-end tests та performance tests.

2 ПРОЕКТУВАННЯ СИСТЕМИ АВТОМАТИЗАЦІЇ ЗАПИСУ

Проєктування є одним із ключових етапів у процесі розробки інформаційної системи, оскільки визначає її структурну організацію, логіку функціонування та спосіб взаємодії з користувачем. У цьому розділі наведено архітектурне та функціональне бачення системи автоматизації запису, розробленої у вигляді Telegram-бота, що дозволяє користувачам здійснювати попередній запис на послуги в інтерактивному режимі.

Розділ охоплює побудову загальної архітектури системи, опис її модульної структури, взаємодію з зовнішніми сервісами, а також моделювання бази даних, що забезпечує збереження й обробку інформації про користувачів, послуги, спеціалістів і записи. Окрему увагу приділено проектуванню користувацького інтерфейсу, що реалізований через чат-бота з діалоговими сценаріями, а також алгоритмам роботи ключових модулів, які забезпечують функціональність системи — від реєстрації до керування слотами часу.

2.1 Архітектура системи

Проєктування є одним із ключових етапів у процесі розробки інформаційної системи, оскільки визначає її структурну організацію, логіку функціонування та спосіб взаємодії з користувачем. У цьому розділі наведено архітектурне та функціональне бачення системи автоматизації запису, розробленої у вигляді Telegram-бота, що дозволяє користувачам здійснювати попередній запис на послуги в інтерактивному режимі.

2.1.1 Загальна структура додатку

Система автоматизації запису реалізована у вигляді інтерактивного Telegram-бота, який виступає основним інтерфейсом взаємодії між кінцевим

користувачем і внутрішніми сервісами інформаційної системи. Такий підхід дозволяє забезпечити простоту використання, мінімізувати потребу у встановленні додаткового програмного забезпечення та зробити систему доступною на більшості мобільних пристроїв.

Програмна реалізація додатку побудована із застосуванням середовища виконання Bun, що характеризується високою продуктивністю та підтримкою сучасного JavaScript/TypeScript-стеку. Основною бібліотекою для реалізації логіки взаємодії з Telegram API є Telegraf, яка забезпечує гнучкий механізм обробки повідомлень, callback-запитів, інлайн-команд і сценаріїв взаємодії.

Загальна структура додатку логічно поділена на такі ключові рівні:

- користувацький інтерфейс (UI) — реалізований у вигляді Telegram-бота, що підтримує як кнопкову, так і текстову взаємодію з користувачем. Забезпечує зворотний зв’язок, доступ до функціональності системи та керування процесами через діалогові сценарії;
- серверна логіка (backend) — включає маршрутизацію команд, управління станами діалогів, обробку подій та формування динамічного контенту. Саме тут реалізовано логіку сценаріїв: реєстрації користувачів, бронювання послуг, перегляду записів, скасування тощо;
- база даних (data layer) — реалізована на основі реляційної моделі з використанням системи керування базами даних SQLite, що оптимально підходить для легких Telegram-додатків. Зберігає структуровану інформацію про користувачів, послуги, спеціалістів, записи та конфігураційні параметри системи.

Інтеграція зазначених компонентів забезпечує повнофункціональну роботу системи в реальному часі та гарантує послідовність обробки запитів користувачів. Модульна організація дозволяє масштабувати функціональність, легко модифікувати окремі компоненти без порушення загальної архітектури додатку.

Таким чином, загальна структура системи є результатом поєднання сучасних підходів до проєктування чат-ботів, принципів модульності та орієнтації на кінцевого користувача.

2.1.2 Модульна організація системи

Архітектура програмного забезпечення реалізована за принципами модульного проєктування, що передбачає розділення системи на окремі, ізольовані компоненти з чітко визначеними функціональними обов'язками. Такий підхід підвищує гнучкість та масштабованість додатку, полегшує внесення змін, сприяє повторному використанню коду, а також покращує можливість автономного тестування окремих частин системи.

Уся кодова база додатку організована у вигляді ієрархічної структури каталогів, що відображає логічне групування за функціональним призначенням. Основні програмні модулі розміщені в директорії src/ і охоплюють такі компоненти:

- src/bot/ — модуль, відповідальний за логіку Telegram-бота. Тут зосереджені сцени (scenes) для реалізації діалогових сценаріїв, обробники команд користувача, а також функції, що забезпечують управління взаємодією в межах сесії. Саме цей модуль визначає основний механізм взаємодії між користувачем і системою;
- src/database/ — модуль керування базою даних. Містить опис реляційних схем таблиць на основі ORM Drizzle, а також інструменти для міграції (оновлення структури бази) та початкової ініціалізації (seed-дані). Цей компонент відповідає за збереження та цілісність даних у системі;
- src/utils/ — модуль допоміжних утиліт, що включає логування подій (logger), генерацію динамічних клавіатур, форматування текстових повідомлень, перетворення дат, обробку callback-даних та інші загальні функції, які використовуються у декількох частинах проєкту;
- src/middleware/ — проміжне програмне забезпечення, що реалізує перехоплення та обробку запитів між етапами взаємодії. Сюди належать механізми автентифікації, обробки помилок, логування подій, обмеження доступу за ролями тощо. Цей модуль забезпечує реалізацію перехресних (cross-cutting) функцій системи;

- src/services/ — модуль бізнес-логіки, який інкапсулює основні операції з об’єктами системи (користувачами, послугами, записами, спеціалістами тощо). Забезпечує високорівневу абстракцію над роботою з даними, логіку перевірок і реалізацію прикладних сценаріїв (наприклад, створення запису, оновлення профілю тощо).

Завдяки чіткому модульному поділу, система залишається структуровано зрозумілою, що значно полегшує командну розробку, супровід та масштабування проєкту. Додавання нової функціональності, виправлення помилок або адаптація до нових вимог можуть здійснюватися із мінімальним ризиком порушення існуючих компонентів.

2.1.3 Взаємодія з зовнішніми сервісами

Система автоматизації запису функціонує у тісній взаємодії з зовнішніми сервісами, що забезпечують розширення функціональних можливостей, доступність і зручність для кінцевого користувача. Основною зовнішньою платформою, з якою інтегровано систему, є Telegram, що виступає комунікаційним середовищем між користувачем і серверною логікою додатку.

Для реалізації взаємодії з Telegram API використовується бібліотека Telegraf, яка забезпечує повноцінну обробку подій, включаючи текстові повідомлення, кнопкові інтерфейси (reply та inline-клавіатури), callback-запити, надсилання медіа, отримання контактних даних тощо. Цей рівень інтеграції дозволяє побудувати зручний, реактивний і безпечний інтерфейс користувача без необхідності використання веббраузера або окремих мобільних додатків.

Окрім базової інтеграції з Telegram, програмна архітектура системи передбачає можливість розширення за рахунок додаткових зовнішніх сервісів, які можуть бути інтегровані для автоматизації суміжних процесів, таких як планування, нагадування, звітність та аналітика. Потенційні напрямки інтеграції включають:

- Google Calendar / iCal — синхронізація записів із зовнішніми календарями дозволить спеціалістам бачити розклад у знайомому інтерфейсі, а також уникати конфліктів у часі між внутрішньою системою і особистими подіями;
- Email- або SMS-шлюзи — надсилання автоматичних повідомлень-нагадувань на електронну пошту або мобільний номер користувача дозволить підвищити ймовірність візиту, а також зменшити кількість пропущених записів;
- CRM-системи — інтеграція з платформами управління взаємовідносинами з клієнтами дозволить зберігати повні історії взаємодії з кожним користувачем, здійснювати аналіз їхньої активності, уподобань та створювати сегментовані пропозиції.

Всі ці напрямки розглядаються як потенційні модулі для майбутнього масштабування, і можуть бути реалізовані через API-запити до відповідних сервісів з використанням авторизованих токенів, вебхуків або стандартних REST-інтерфейсів. Така модульність забезпечує гнучкість системи та її здатність до адаптації в умовах зростання функціональних і бізнес-вимог.

2.2 Проектування бази даних

Проектування бази даних є критичним етапом розробки системи автоматизації запису, оскільки саме база даних забезпечує зберігання, цілісність та доступність даних користувачів, послуг, спеціалістів і записів. У даній системі використовується SQLite, що є ефективним рішенням для Telegram-ботів із локальним або серверним зберіганням без необхідності складного налаштування.

2.2.1 Концептуальна модель даних

Концептуальне моделювання даних є фундаментальним етапом розробки будь-якої інформаційної системи, оскільки дозволяє абстраговано описати предметну область у вигляді сутностей, їх атрибутів та взаємозв'язків. Для системи автоматизації запису концептуальна модель побудована відповідно до принципів реляційної моделі даних та орієнтована на забезпечення цілісного зберігання інформації про користувачів, послуги, спеціалістів, розклад і записи.

Основні сутності моделі:

- Користувачі (Users) — представляють осіб, які взаємодіють із системою через Telegram-бота. Кожен користувач ідентифікується за Telegram ID і має такі атрибути, як ім'я, номер телефону та роль (client, admin, specialist), що визначає рівень доступу до функціональності системи;
- Послуги (Services) — описують доступні в системі послуги, що можуть бути надані спеціалістами. До складу атрибутів входять назва, опис, тривалість (у хвилинах) та ціна;
- Спеціалісти (Specialists) — це працівники, які виконують послуги. Кожен спеціаліст має унікальний ідентифікатор, ім'я, контактні дані, а також асоціюється з однією або кількома послугами, які він надає;
- Записи (Appointments) — є основною транзакційною сутністю, що фіксує факт бронювання. Кожен запис пов'язаний з конкретним користувачем, спеціалістом, послугою, а також містить дату, час і статус виконання;
- Часові слоти (Slots) — визначають доступні для запису часові інтервали. Вони використовуються для динамічного формування графіку доступності спеціалістів. Містять інформацію про дату, час початку і завершення слота, а також статус (вільний/зайнятий);
- Налаштування (Settings) — є допоміжною сутністю, що зберігає глобальні параметри конфігурації системи, зокрема стандартну тривалість слота, робочі години, тексти повідомлень-нагадувань тощо;

Візуальне представлення концептуальної моделі даних у вигляді UML діаграми представлене на рисунку 2.1.

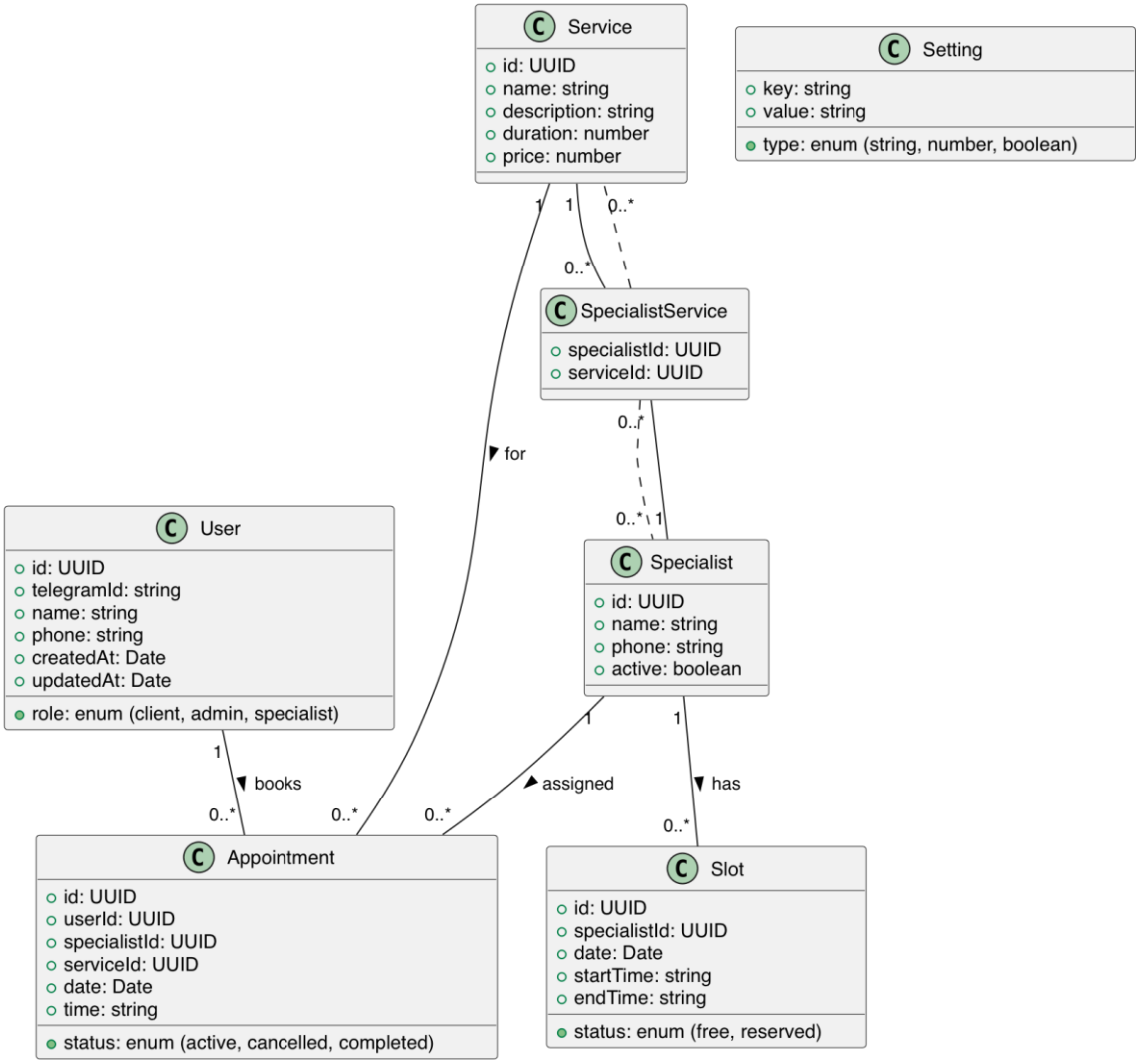

Рисунок 2.1 – Концептуальна модель даних

Ця діаграма формує концептуальну основу подальшого логічного та фізичного проєктування бази даних і лягає в основу ORM-схем, що реалізуються у системі через Drizzle ORM.

2.2.2 Логічна структура бази даних

На логічному рівні база даних моделюється у вигляді множини взаємопов’язаних реляційних таблиць, які відображають концептуальні сутності предметної області та їх атрибути. Реалізація логічної структури здійснена з використанням сучасного ORM-інструменту Drizzle ORM, який орієнтований на застосування у середовищах, що використовують TypeScript як основну мову розробки.

Drizzle ORM забезпечує типобезпечне визначення схем таблиць, сувору відповідність структури бази даних вимогам прикладної логіки та можливість автоматизованої генерації міграцій, що значно підвищує надійність розробки та підтримки системи. Визначення таблиць реалізовано у модулі src/database/schema.ts шляхом використання декларативних конструкцій на кшталт sqliteTable, text, integer, primaryKey, foreignKey тощо.

У структурі бази даних визначено такі **основні логічні таблиці**:

- users — зберігає інформацію про всіх користувачів, які взаємодіють із системою. Основним ключем є Telegram ID, що гарантує унікальність запису. Додаткові атрибути включають:
    - id: string — унікальний Telegram user ID;
    - name: text — ім’я користувача;
    - phone: text — верифікований номер телефону;
    - role: text — тип користувача (client, admin, specialist);
    - createdAt, updatedAt: datetime — часові мітки створення та оновлення запису.
- services — містить перелік послуг, доступних для запису. Структура таблиці включає:
    - id: integer — унікальний ідентифікатор;
    - name, description: text — назва та опис послуги;
    - duration: integer — тривалість послуги (у хвилинах);
    - price: integer — вартість.

- specialists — зберігає інформацію про спеціалістів, які надають послуги:
  - id: integer — ідентифікатор спеціаліста;
  - name: text — ім’я;
  - phone: text — контактний номер;
  - active: boolean — ознака активності у системі.
- appointments — головна транзакційна таблиця, яка фіксує факти бронювання:
  - id: integer — унікальний ідентифікатор запису;
  - userId, specialistId, serviceId: foreign keys — зовнішні ключі на відповідні таблиці;
  - date: date — дата бронювання;
  - time: time — час початку;
  - status: text — статус (active, cancelled, completed).
- specialist_services — проміжна таблиця для реалізації зв’язку N:M між specialists і services. Структура:
  - specialistId: integer — зовнішній ключ на таблицю specialists;
  - serviceId: integer — зовнішній ключ на таблицю services;
  - первинний ключ формується за принципом складеного ключа (specialistId, serviceId).
- settings — таблиця конфігураційних параметрів:
  - key: text — унікальний ідентифікатор параметра;
  - value: text — значення;
  - type: text — тип значення (string, number, boolean), що дає змогу виконувати типову валідацію під час зчитування параметрів у логіці бота.

Всі таблиці мають чітко визначені первинні ключі та, де необхідно, зовнішні ключі, що забезпечують референційну цілісність бази.

Типізація моделей у Drizzle ORM відповідає типам, які використовуються в логіці застосунку, що мінімізує ризики невідповідності даних.

Завдяки підтримці генерації міграцій ми маємо змогу безпечно оновлювати схему бази даних без втрати існуючих даних, зберігаючи історію змін.

2.2.3 Фізична реалізація сховища даних

Фізичний рівень реалізації бази даних передбачає конкретну організацію зберігання інформації у вигляді файлової структури на диску, а також механізми доступу до неї на рівні операційної системи. У запропонованій системі фізичне сховище реалізовано у вигляді окремого SQLite-файлу booking.db, який зберігається в каталозі db/ у кореневій директорії проєкту. Вибір формату SQLite є обґрунтованим з огляду на невеликі обсяги даних, швидкість доступу, простоту розгортання та відсутність необхідності у використанні зовнішнього серверного програмного забезпечення.

Для інтеграції фізичного сховища з логікою програми застосовується адаптер drizzle-orm/bun-sqlite, що забезпечує пряме підключення до SQLite через середовище виконання Bun. Такий підхід дозволяє ефективно поєднати типізовану ORM-логіку з низькорівневим доступом до фізичної бази даних у рамках одного процесу без втрат продуктивності.

Ініціалізація підключення до бази даних реалізована у файлі src/database/db.ts, у якому реалізовано такі основні функції:

- initDatabase() — відкриває з'єднання з фізичним файлом booking.db, ініціалізує об'єкт ORM і застосовує конфігурацію схеми;
- getDb() — повертає активне з'єднання з базою для використання в інших частинах програми;
- closeDatabase() — забезпечує коректне завершення роботи з базою, зокрема при зупинці сервісу або перезавантаженні процесу.

Для початкового заповнення бази структурованими тестовими даними використовується файл src/database/seed.ts, який створює базові записи у таблицях services, specialists, settings тощо. Це дає змогу швидко підготувати систему до демонстрації або тестування без необхідності ручного введення даних.

Крім того, у системі реалізовано повноцінний механізм міграцій (src/database/migrate.ts), який дозволяє здійснювати контрольовані зміни у структурі бази даних відповідно до змін у моделях схеми. Міграції виконуються через ORM-інструменти, що зберігають історію змін і забезпечують послідовність оновлення бази без втрати існуючих записів. Такий підхід є особливо важливим у процесі командної розробки, коли база даних має адаптуватися до змін у коді без ризику порушення її цілісності.

У підсумку, використання SQLite як фізичного сховища у поєднанні з Drizzle ORM і середовищем Bun формує ефективну, надійну та продуктивну інфраструктуру зберігання даних для невеликих і середніх систем. Це дозволяє досягти високої швидкодії при обробці запитів, спростити розгортання застосунку та забезпечити гнучкість масштабування в подальшому.

2.3 Проєктування користувацького інтерфейсу

Інтерфейс користувача у системі автоматизації запису реалізовано у вигляді чат-бота, що взаємодіє з користувачем через Telegram. Такий підхід дозволяє уникнути складного веб-інтерфейсу і забезпечує доступність сервісу для широкого кола користувачів через знайоме середовище Telegram.

2.3.1 Сценарії взаємодії користувача з ботом

У системі автоматизації запису всі взаємодії з кінцевим користувачем реалізовані у вигляді сценаріїв, які імітують природний діалог у середовищі месенджера Telegram. Такий підхід дозволяє забезпечити просту, послідовну і зрозумілу логіку навігації, що є критично важливою для користувачів, які не мають спеціальної технічної підготовки.

Сценарна логіка реалізована за допомогою модуля WizardScene з бібліотеки Telegraf Scenes, який дозволяє поділити кожен процес на окремі логічні кроки

(етапи) з підтримкою збереження контексту (через ctx.wizard.state). Завдяки цьому користувач може проходити послідовність дій без потреби повторного введення або втрати даних при переході між кроками.

Основні сценарії взаємодії

- Реєстрація користувача

При першому запуску бота система виконує перевірку наявності користувача у базі даних за Telegram ID. Якщо запис відсутній, запускається сценарій реєстрації, що передбачає запит надання номера телефону. Запит реалізовано через кнопку «Поділитися номером телефону», яка активує механізм contactRequest у Telegram API. Отриманий контакт використовується для ідентифікації та створення облікового запису користувача в системі.

- Вибір послуги

Після успішної авторизації користувачеві пропонується перелік доступних послуг, сформований на основі таблиці services у базі даних. Кожна послуга відображається у вигляді кнопки з коротким описом. Вибір кнопки генерує callback_query, що дозволяє системі ідентифікувати обрану послугу без потреби в текстовому введенні.

- Вибір спеціаліста

На основі раніше обраної послуги формується фільтрований список спеціалістів, які можуть її надавати (зв’язок реалізовано через таблицю specialist_services). Користувач обирає спеціаліста зі списку, що динамічно відображається за допомогою інлайн-кнопок.

- Вибір дати та часу

Після вибору спеціаліста користувачеві пропонуються доступні дати (наприклад, найближчі 3–5 робочих днів). Для кожної обраної дати система генерує доступні часові слоти на основі налаштувань тривалості (slotDuration) та зайнятості відповідного спеціаліста. Інтерфейс реалізований у вигляді динамічно згенерованих інлайн-клавіатур.

- Підтвердження запису

Після вибору часу система формує стислий підсумок вибору: послуга, спеціаліст, дата, час. Користувачеві пропонується підтвердити бронювання або скасувати його. У разі підтвердження, запис створюється у таблиці appointments зі статусом active.

- Перегляд і скасування записів

У головному меню користувач може переглянути власні активні записи. Список формується шляхом запиту до таблиці appointments з фільтрацією за Telegram ID користувача. Біля кожного запису розміщується кнопка “Скасувати”, натискання на яку змінює статус відповідного запису на cancelled.

Таким чином, усі сценарії побудовані за принципом інтуїтивно зрозумілої покрокової взаємодії з використанням кнопкових інтерфейсів, що мінімізує ймовірність помилкових дій з боку користувача. Реалізація діалогів у вигляді сценаріїв забезпечує високий рівень гнучкості системи, а також дозволяє легко масштабувати її функціональність за рахунок додавання нових сценаріїв або розгалужень у логіці.

2.3.2 Діалогові потоки та навігація

Діалогова структура Telegram-бота проєктована відповідно до принципів контекстної навігації, яка передбачає логічно обґрунтовану послідовність переходів між етапами взаємодії на основі поточного стану користувача та його дій. У межах такого підходу кожен сценарій розбито на структуровані кроки, які автоматично активуються в залежності від контексту — тобто обраної команди, callback-відповіді або поточної сцени.

Бот не використовує текстовий парсинг як основний інтерфейс взаємодії. Замість цього основна навігація реалізована через інлайн-клавіатури (створені за допомогою Markup.inlineKeyboard) та reply-клавіатури (Markup.keyboard), що дозволяє користувачеві здійснювати вибір без введення текстових повідомлень. Такий підхід значно знижує когнітивне навантаження, зменшує ймовірність помилок і покращує користувацький досвід, особливо в мобільних інтерфейсах.

Основні діалогові потоки:

1. ініціалізація взаємодії:
    - /start → вітальне повідомлення → перевірка автентичності → запуск сцени реєстрації → запит номера телефону → збереження даних у базу.
2. головне меню:

Після реєстрації або повторного запуску бот виводить головне меню, де користувачеві пропонуються наступні дії:

- “Записатися” — запуск сценарію створення запису;
- “Мої записи” — перегляд особистих записів із можливістю скасування;
- “Налаштування” — (опційно) управління мовою, повідомленнями тощо.

3. процес запису на послугу:
    - вибір послуги (зі списку, сформованого динамічно);
    - вибір спеціаліста (фільтрований список);
    - вибір дати (з урахуванням робочих днів);
    - вибір часу (на основі вільних слотів);
    - підтвердження → створення запису.
4. перегляд і керування записами:
    - отримання переліку майбутніх записів;
    - відображення інформації у вигляді списку;
    - додаткові кнопки для скасування обраного запису;
    - підтвердження дії → зміна статусу запису у базі.

Механізми реалізації навігації:

- reply-клавіатури використовуються в обмеженій кількості випадків, зокрема для запиту контактної інформації (наприклад, кнопка “Поділитися номером телефону”) або реалізації швидких команд (”Скасувати”). Такі клавіатури з’являються в полі введення повідомлень і зникають після завершення відповідного етапу;

- інлайн-клавіатури є основним інструментом навігації. Вони дозволяють відображати динамічні варіанти вибору (наприклад, послуги, дати, години) без необхідності повторного введення. Callback-дані обробляються відповідними обробниками, що дозволяє контекстно реагувати на вибір користувача;
- контроль стану реалізується за допомогою Telegraf Scenes, що дозволяє зберігати поточний крок користувача в діалозі (ctx.wizard.cursor) та змінювати поведінку бота відповідно до цього стану.

Таким чином, навігаційна логіка в системі є не лише функціонально завершеною, а й орієнтованою на користувача. Вона відповідає принципам UX-дизайну для діалогових систем, де ключовими є простота, передбачуваність і мінімізація ручного вводу.

2.3.3 Дизайн клавіатур та елементів управління

Однією з ключових складових зручного та ефективного користувацького інтерфейсу Telegram-бота є грамотно реалізована система елементів керування, що забезпечують інтерактивність і контекстну навігацію. У розробленій системі управління діалогами реалізовано з використанням кількох типів клавіатур, які відповідають за вибір опцій, підтвердження дій та інтуїтивну взаємодію з користувачем.

Основні типи клавіатур

- Reply-клавіатура

З’являється у полі введення повідомлень Telegram і використовується переважно на етапах, де необхідно здійснити швидку дію, таку як підтвердження або скасування. Найбільш типовими прикладами є кнопка “Поділитися номером телефону”, яка ініціює надсилання контактних даних користувача через API Telegram, та кнопка “Скасувати”, що дозволяє завершити поточну дію або сцену.

- Інлайн-клавіатура

Формується динамічно та відображається безпосередньо в повідомленні бота. Вона є основним засобом навігації для вибору послуг, спеціалістів, дат, часових слотів тощо. Кожна кнопка має поле callback_data, яке передає закодовану інформацію про дію боту (наприклад, service:2, slot:2024-06-20_10:00, cancel:3). Це дозволяє обробляти вибір користувача без необхідності введення тексту.

- Клавіатури з пагінацією

Застосовуються у випадках, коли кількість варіантів для вибору перевищує зручний обсяг для відображення в одному повідомленні. Зокрема, це стосується вибору дати, часу, спеціалістів чи записів. Клавіатури створюються динамічно за допомогою функціональних утиліт, таких як generateKeyboard, що дозволяють реалізувати логіку переходів (”Назад”, “Далі”) та зберігати контекст вибору.

Елементи зворотного зв’язку:

З метою підвищення зручності взаємодії бот активно використовує візуальні підказки, що сприяють кращому сприйняттю інформації:

- emoji-піктограми — (✅, ❌, 📅, ℹ️) виконують роль емоційних маркерів, що дозволяють швидко ідентифікувати статус дії (успіх, помилка, очікування тощо);
- HTML-форматування повідомлень — використовується для візуального структурування тексту (наприклад, Реєстрація, Ваш номер телефону буде використано для зв’язку), що підвищує читабельність та логічну організацію повідомлень;
- стандартизовані повідомлення про результати дій — для кожної дії реалізовано уніфіковані повідомлення про успішне завершення (Запис створено), помилку (Обраний слот зайнято) або інструкцію щодо подальших кроків (Оберіть дату для запису).

Узагальнення:

Уся система елементів управління Telegram-бота проєктована відповідно до вимог адаптивного інтерфейсу, орієнтованого на мобільні пристрої. Вона забезпечує:

- мінімізацію потреби в текстовому введенні;
- чіткий візуальний зворотний зв’язок;
- контекстну побудову інтерфейсу в залежності від логіки діалогу;
- адаптацію до динамічного контенту.

Таким чином, розроблений інтерфейс є не лише функціональним, але й користувацько-орієнтованим, що значною мірою сприяє ефективності системи в реальних умовах її експлуатації.

2.4 Алгоритми роботи основних модулів

Програмна система автоматизованого запису на послуги реалізована у вигляді набору логічно відокремлених, але функціонально пов’язаних між собою модулів, кожен з яких відповідає за певний бізнес-процес. Взаємодія користувача з системою відбувається через Telegram-інтерфейс у формі поетапного діалогу, що забезпечується за допомогою сцен (WizardScene) бібліотеки Telegraf. Основними модулями системи є: модуль реєстрації користувача, модуль бронювання, модуль обчислення вільних часових слотів, а також модуль обробки повідомлень та команд.

Алгоритм реєстрації користувача передбачає автентифікацію за Telegram ID, збір контактних даних (насамперед номера телефону), перевірку формату номера, його нормалізацію до міжнародного формату та збереження або оновлення даних у таблиці користувачів. Якщо користувач уже існує в базі, він автоматично перенаправляється до головного меню, минаючи етап реєстрації. Загальна структура цього процесу зображена на рисунку 2.2.

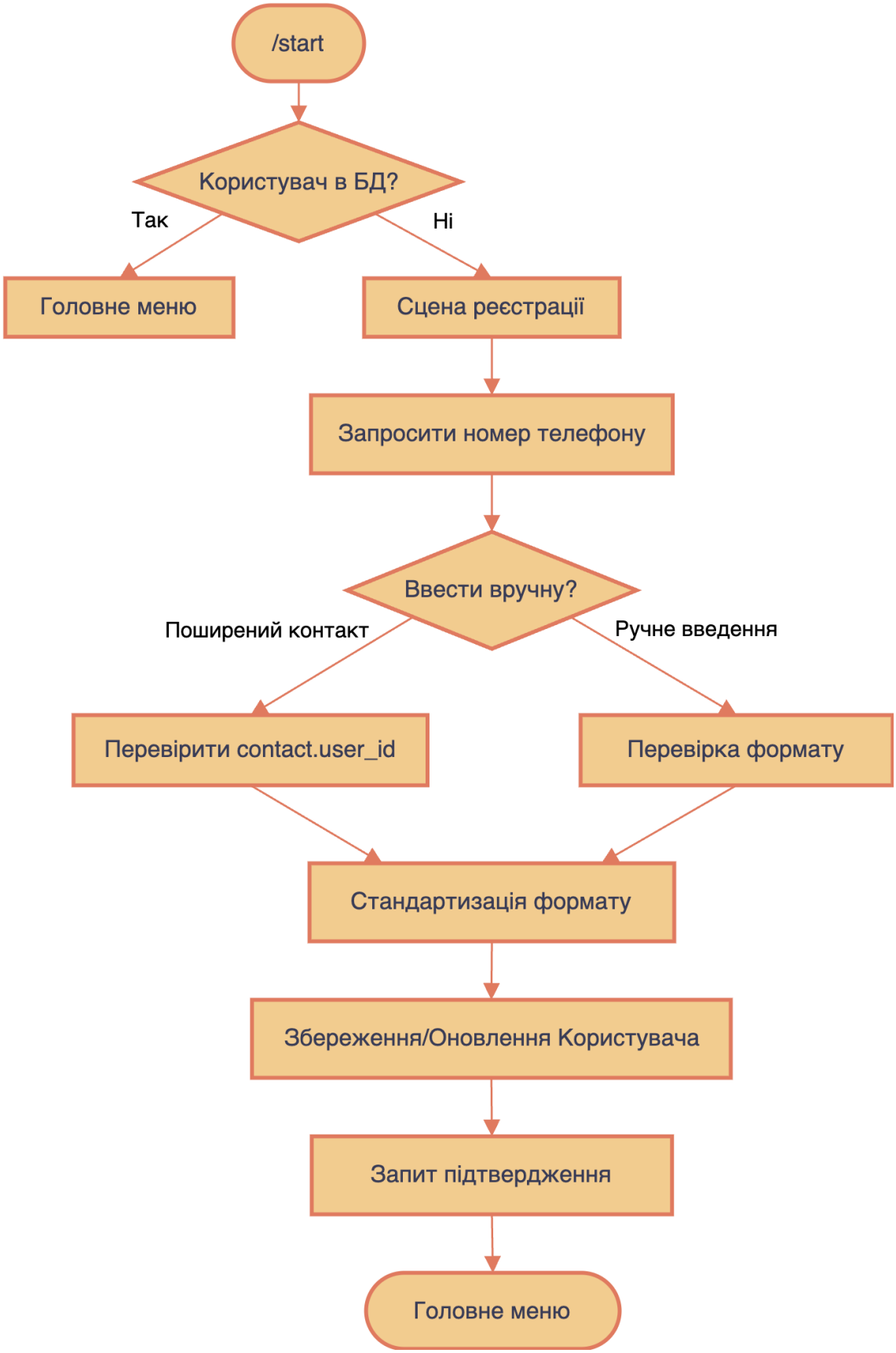

Рисунок 2.2 – Алгоритм реєстрації та авторизації користувачів

Діаграма демонструє сценарій входу користувача, перевірку наявності в базі даних, ініціацію сцени реєстрації, отримання номера телефону та завершення реєстрації шляхом збереження у таблиці users.

Процес створення бронювання реалізовано у вигляді послідовного вибору користувачем необхідної послуги, спеціаліста, дати та часу, після чого дані перевіряються і фіксуються у таблиці appointments. Кожен крок супроводжується динамічно сформованими інлайн-клавіатурами. Вибір послуги базується на таблиці services, після чого відбувається фільтрація відповідних спеціалістів з таблиці specialist_services. Далі користувачу пропонуються дати відповідно до доступного графіка спеціаліста та обчислюються вільні часові слоти. Завершальний етап — підтвердження або скасування бронювання. Весь алгоритм відображено на рисунку 2.3.

Рисунок 2.3 – Алгоритм процесу бронювання

Діаграма описує сценарій покрокового бронювання: від вибору послуги та спеціаліста — до створення остаточного запису.

Алгоритм розрахунку доступних часових слотів базується на конфігураційних налаштуваннях системи (робочі години, тривалість одного слота), а також на актуальній завантаженості спеціаліста. Генерується повний список потенційних слотів, який потім фільтрується шляхом виключення часу, що вже зайнятий. У результаті користувачеві пропонуються лише дійсно доступні варіанти. Алгоритм розподілу слотів зображено на рисунку 2.4.

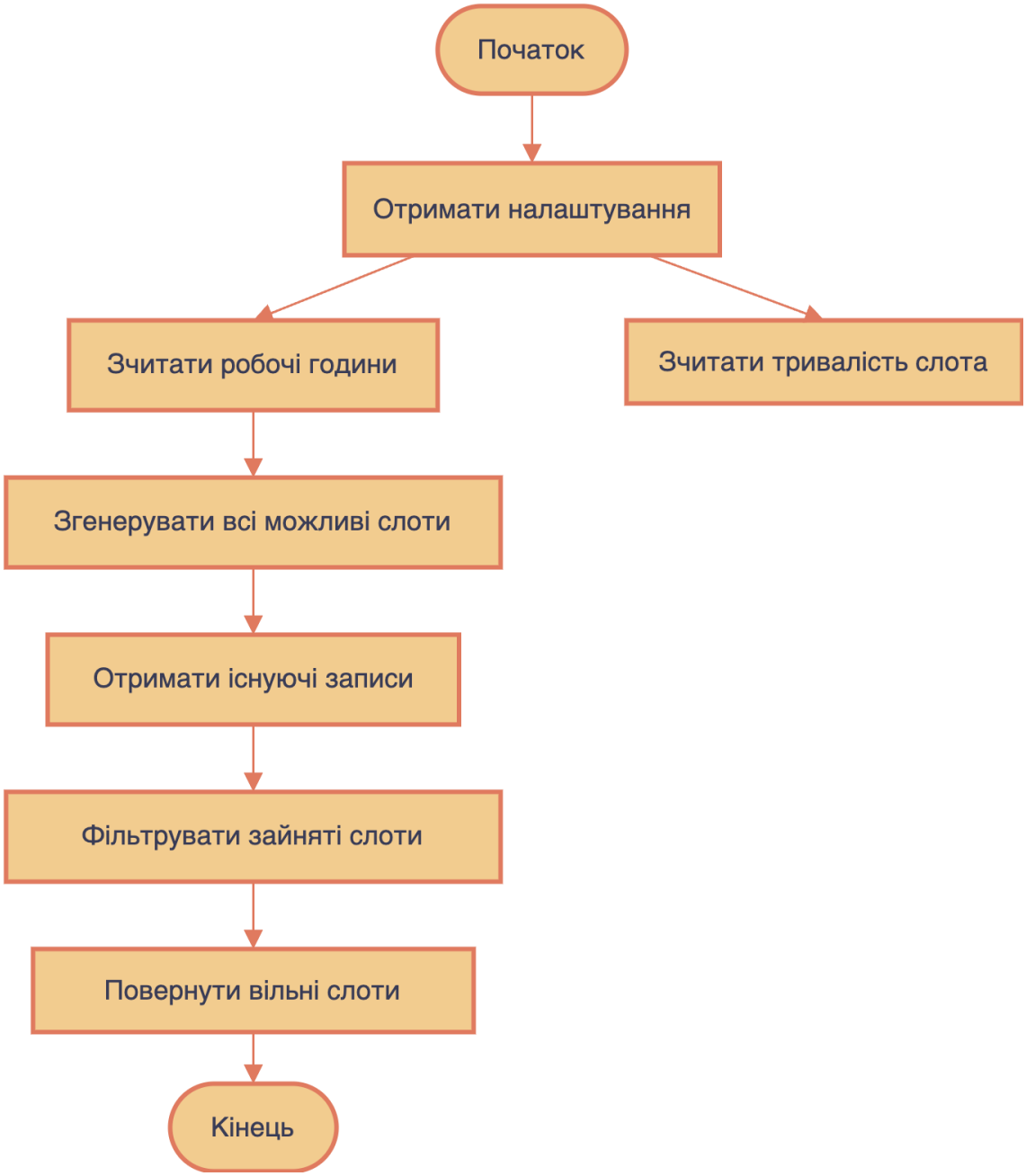

Рисунок 2.4 – Алгоритм розподілу часових слотів

UML-діаграма демонструє процес генерації часових слотів, фільтрації зайнятих проміжків і повернення списку вільних інтервалів для вибраного спеціаліста.

Усі повідомлення користувача (команди, callback-запити, текстові повідомлення або контакти) обробляються єдиним механізмом маршрутизації, що реалізований через обробники подій у Telegraf. Кожна команда обробляється

відповідним сценічним сценарієм або callback-функцією. Callback-дані аналізуються для визначення типу дії та пов’язаних параметрів. Додатково реалізовано логування і контроль помилок, що забезпечує надійну роботу системи. Структуру обробки подій наведено на рисунку 2.5

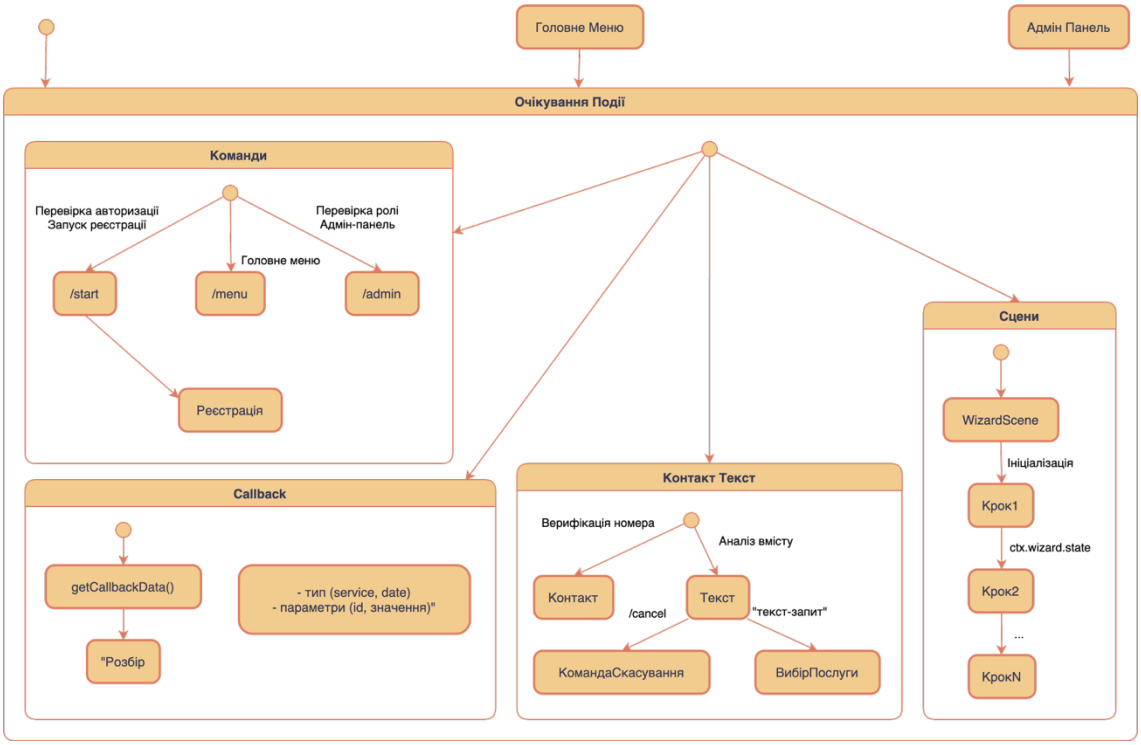

Рисунок 2.5 – Механізм обробки повідомлень та команд

Діаграма демонструє розгалуження обробників у системі: команди, callback-запити, текстові повідомлення та сцени, що координуються через центральний модуль обробки.

Таким чином, взаємопов’язана система алгоритмів забезпечує цілісну логіку функціонування Telegram-бота, дозволяючи реалізувати процес автоматизованого запису користувачів на послуги з урахуванням гнучкого управління ролями, часовими інтервалами та обробкою повідомлень.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ TELEGRAM-БОТА ДЛЯ АВТОМАТИЗАЦІЇ ЗАПИСУ

Практична реалізація системи автоматизації запису на обслуговування вимагає ретельного підходу до вибору технологій, проектування архітектури та імплементації функціональних модулів. Даний розділ присвячено детальному розгляду процесу програмної реалізації Telegram-бота, що забезпечує повний цикл управління записами клієнтів - від вибору послуги до отримання нагадувань про візит.

У розділі розглядаються ключові аспекти розробки: обґрунтування вибору технологічного стеку, особливості використання сучасних інструментів розробки, архітектурні рішення та практична реалізація основних функціональних модулів системи. Особлива увага приділяється інноваційним підходам, таким як використання Bun runtime, що дозволяють досягти високої продуктивності та ефективності розробки.

Представлена реалізація демонструє, як сучасні технології можуть бути ефективно застосовані для вирішення практичних бізнес-задач, забезпечуючи при цьому високу якість користувацького досвіду, надійність роботи та простоту подальшого розвитку системи.

3.1 Вибір технологічного стеку

Вибір правильного технологічного стеку є фундаментальним рішенням, що визначає не лише можливості та обмеження майбутньої системи, але й ефективність процесу розробки, легкість підтримки та потенціал для масштабування. Для реалізації Telegram-бота автоматизації запису було проведено

ґрунтовний аналіз доступних технологій з урахуванням специфіки завдання, вимог до продуктивності та досвіду команди розробки.

3.1.1 Обґрунтування вибору Node.js та TypeScript

Вибір Node.js як основної платформи для розробки серверної частини Telegram-бота обумовлений кількома ключовими факторами, що роблять цю технологію оптимальною для даного типу застосунків.

Асинхронна модель виконання є однією з головних переваг Node.js для розробки ботів. Telegram-боти за своєю природою працюють з великою кількістю одночасних з'єднань та повідомлень від користувачів. Неблокуюча подієво-орієнтована архітектура Node.js дозволяє ефективно обробляти тисячі запитів без створення окремого потоку для кожного з'єднання. Це критично важливо для забезпечення низької затримки відповіді навіть при високому навантаженні.

Єдина мова програмування для всього стеку розробки спрощує процес створення та підтримки коду. JavaScript, а в нашому випадку TypeScript, використовується як для серверної логіки, так і для конфігураційних скриптів, що знижує когнітивне навантаження на розробників та полегшує онбординг нових членів команди.

Багата екосистема npm надає доступ до сотень тисяч готових пакетів, що значно прискорює розробку. Для роботи з Telegram Bot API існують зрілі та добре документовані бібліотеки, такі як Telegraf, що використовується в нашому проекті. Це дозволяє зосередитися на бізнес-логіці замість реалізації низькорівневих деталей протоколу.

TypeScript як надбудова над JavaScript вирішує одну з головних проблем динамічно типізованих мов - відсутність статичної перевірки типів. У контексті розробки бота, де необхідно працювати з різноманітними структурами даних (повідомлення користувачів, callback-дані, об'єкти бази даних), статична типізація стає критично важливою для запобігання помилкам під час виконання.

Переваги використання TypeScript у проекті:

1. рання детекція помилок - більшість помилок виявляється на етапі компіляції, а не в runtime. Це особливо важливо для бота, який повинен працювати безперервно;
2. покращена підтримка IDE - автодоповнення, рефакторинг та навігація по коду стають значно ефективнішими завдяки точній інформації про типи;
3. самодокументований код - типи служать додатковою документацією, що полегшує розуміння коду іншими розробниками;
4. безпечний рефакторинг - зміни в структурах даних автоматично виявляють всі місця, де потрібні оновлення.

Наприклад, визначення типів для контексту бота забезпечує типобезпеку на всіх рівнях обробки повідомлень:

```
export interface BotContext extends Context {
  session: MySession;
  scene: Scenes.SceneContextScene<BotContext, MySceneSession>;
  wizard: Scenes.WizardContextWizard<BotContext>;
}
```

Продуктивність V8 engine, на якому базується Node.js, постійно покращується завдяки інвестиціям Google. Сучасні версії V8 включають JIT-компіляцію, оптимізацію гарячих шляхів виконання та ефективне управління пам'яттю. Для Telegram-бота, що обробляє текстові повідомлення та виконує запити до бази даних, така продуктивність є більш ніж достатньою.

Простота deployment є ще однією перевагою Node.js. Додаток можна запустити на будь-якій платформі, де встановлено Node.js runtime, без складних налаштувань середовища. Це спрощує як локальну розробку, так і розгортання на production серверах.

3.1.2 Використання Bun як runtime середовища

Однією з ключових інновацій у технологічному стеку проекту є використання Bun замість традиційного Node.js runtime. Bun - це сучасне all-in-one

JavaScript runtime, створене з нуля для максимальної продуктивності та зручності розробки.

Швидкість виконання є головною перевагою Bun. На відміну від Node.js, який використовує V8 engine від Google, Bun побудований на JavaScriptCore від Apple. Це забезпечує:

- швидший холодний старт (до 4x порівняно з Node.js), що критично для бота, який може перезапускатися при оновленнях;
- нижче споживання пам'яті, що дозволяє ефективніше використовувати ресурси сервера;
- швидшу роботу з файловою системою та мережевими операціями.

Вбудовані інструменти розробки роблять Bun привабливим вибором для сучасних проектів:

1. package manager - встановлення залежностей відбувається в 10-100 разів швидше порівняно з npm або yarn. Для проекту з десятками залежностей це означає скорочення часу початкового налаштування з хвилин до секунд;
2. TypeScript з коробки - немає потреби в окремому етапі транспіляції. Bun нативно виконує TypeScript файли, що спрощує налаштування проекту та прискорює цикл розробки;
3. test runner - вбудована система тестування працює значно швидше за Jest, що заохочує розробників частіше запускати тести;
4. bundler - можливість створення оптимізованих збірок для production без додаткових інструментів.

Сумісність з Node.js екосистемою була критичним фактором при виборі Bun. Більшість npm пакетів працюють без змін, що дозволило використати перевірені бібліотеки, такі як Telegraf для роботи з Telegram API та Drizzle ORM для взаємодії з базою даних.

Вбудована підтримка SQLite є особливо корисною для нашого проекту. Bun надає нативний API для роботи з SQLite, що працює значно швидше за JavaScript бібліотеки:

```
import { Database } from "bun:sqlite";
const db = new Database("booking.db");
```

Це дозволяє виконувати запити до бази даних з мінімальними накладними витратами, що покращує загальну відгуковість бота.

Ефективна робота з Web API включає вбудовану підтримку fetch, WebSocket та інших сучасних API без необхідності в поліфілах. Для Telegram-бота, який постійно взаємодіє з зовнішнім API, це означає простіший та чистіший код.

На рисунку 3.1 представлено порівняння продуктивності Bun та Node.js для типових CRUD операцій, що виконуються в Telegram-боті.

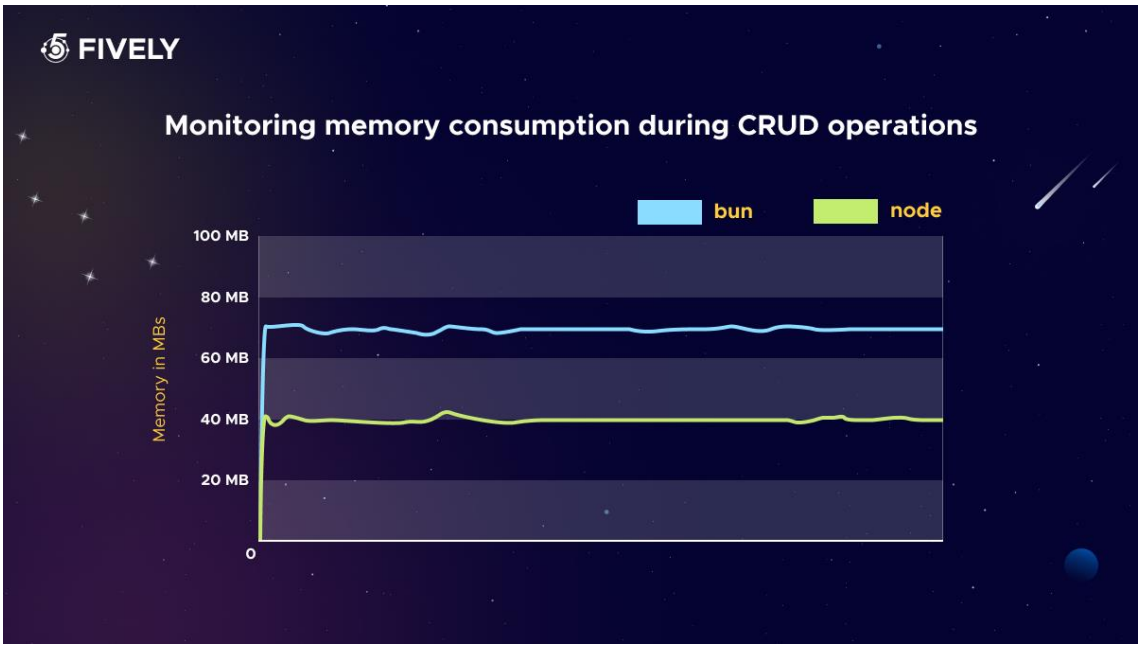

Рисунок 3.1 - Порівняльний аналіз продуктивності Bun та Node.js для CRUD операцій

3.1.3 Вибір бібліотек та фреймворків

Правильний вибір бібліотек та фреймворків визначає не лише функціональні можливості системи, але й швидкість розробки, якість коду та легкість підтримки.

Для реалізації Telegram-бота було обрано перевірені рішення, що оптимально поєднують функціональність, продуктивність та зручність використання.

Telegraf як основний фреймворк для роботи з Telegram Bot API став природним вибором завдяки своїй зрілості, активній спільноті та елегантному API. Telegraf надає:

- Високорівневу абстракцію над Telegram Bot API, що дозволяє працювати з об'єктами та методами замість низькорівневих HTTP запитів
- Вбудовану систему middleware для організації обробки повідомлень
- Потужну систему сцен (Scenes) для реалізації складних діалогів
- Підтримку сесій для збереження стану користувача

Архітектура middleware в Telegraf дозволяє створювати модульний та легко розширюваний код. Кожен middleware може виконувати свою функцію (логування, авторизація, обробка помилок) та передавати управління наступному в ланцюжку.

Drizzle ORM для роботи з базою даних обрано як сучасне TypeScript-first рішення, що забезпечує:

1. Повну типобезпеку - всі запити до бази даних типізовані, що унеможливлює помилки на кшталт звернення до неіснуючих полів
2. SQL-подібний синтаксис - на відміну від складних ORM з власними DSL, Drizzle використовує знайомий SQL-подібний синтаксис
3. Високу продуктивність - мінімальні накладні витрати порівняно з "важкими" ORM
4. Підтримку міграцій - вбудована система міграцій для еволюції схеми бази даних

Day.js для роботи з датами та часом замінив популярний, але важкий Moment.js. Day.js має розмір всього 2KB (проти 67KB у Moment.js) при збереженні знайомого API. Для бота, що постійно працює з датами та часовими слотами, це критично важливо:

- Immutable API запобігає випадковим мутаціям дат
- Підтримка локалізації дозволяє відображати дати українською мовою

- Плагінна архітектура дає можливість підключати лише необхідний функціонал

Biome як інструмент для форматування та лінтингу коду - інноваційне рішення, що замінює комбінацію ESLint + Prettier. Biome написаний на Rust і працює в 20-40 разів швидше за традиційні інструменти. Це особливо помітно при роботі з великою кодовою базою - перевірка та форматування всього проекту займає долі секунди замість десятків секунд.

Архітектурні патерни та організація коду базуються на принципах чистої архітектури з чітким розділенням відповідальностей:

- Сервісний шар інкапсулює бізнес-логіку та забезпечує її незалежність від конкретної реалізації інтерфейсу (Telegram)
- Репозиторний патерн абстрагує роботу з базою даних, дозволяючи легко змінювати способи зберігання даних
- Middleware pipeline організовує обробку запитів у вигляді послідовності незалежних обробників

3.2 Реалізація серверної частини

Серверна частина Telegram-бота є ядром системи, що забезпечує обробку користувацьких запитів, взаємодію з базою даних, управління бізнес-логікою та інтеграцію з зовнішніми сервісами. Правильна організація серверного коду визначає не лише поточну функціональність, але й можливості для подальшого розвитку та масштабування системи.

3.2.1 Налаштування середовища розробки

Ефективне середовище розробки є фундаментом продуктивної роботи над проектом. Для Telegram-бота було створено оптимальне оточення, що поєднує сучасні інструменти з продуманою структурою проекту.

Структура проекту організована за принципом модульності з чітким розділенням відповідальностей між компонентами. Кожен модуль відповідає за конкретну функціональну область, що спрощує навігацію по коду та полегшує підтримку:

```
telegram-booking-bot/
├── src/
│   ├── bot/              # Логіка Telegram бота
│   ├── database/         # Схеми БД та міграції
│   ├── services/         # Бізнес-логіка
│   ├── utils/            # Допоміжні функції
│   ├── types/            # TypeScript типи
│   └── config/           # Конфігурація
├── tests/                # Тести
└── logs/                 # Файли логів
```

Конфігурація TypeScript налаштована для максимальної строгості перевірки типів, що дозволяє виявляти потенційні проблеми на етапі компіляції. Використання path aliases (@/services/*, @/utils/*) спрощує імпорти та робить код більш читабельним.

Управління змінними середовища реалізовано через файл .env, що дозволяє легко налаштовувати додаток для різних оточень (development, staging, production) без зміни коду. Критичні параметри, такі як токен бота та паролі, ніколи не потрапляють у систему контролю версій.

Система логування побудована на власній реалізації, оптимізованій для потреб проекту. Логи структуровані, включають різні рівні деталізації та автоматично ротуються для запобігання переповнення диску. Це критично важливо для діагностики проблем у production середовищі.

Hot-reload під час розробки забезпечується вбудованими можливостями Bun, що дозволяє миттєво бачити результати змін коду без ручного перезапуску. Це значно прискорює цикл розробки та тестування нових функцій.

На рисунку 3.2 показано процес ініціалізації середовища розробки та основні компоненти конфігурації.

```
$ drizzle-kit generate:sqlite
drizzle-kit: v0.20.18
drizzle-orm: v0.29.5

No config path provided, using default 'drizzle.config.ts'
Reading config file '/Users/andreeich/Documents/CodeBox/JS/telegram-booking-bot/drizzle.config.ts'
8 tables
bookings 14 columns 5 indexes 3 fks
services 11 columns 0 indexes 0 fks
settings 6 columns 1 indexes 0 fks
specialist_holidays 5 columns 2 indexes 1 fks
specialist_services 2 columns 0 indexes 2 fks
specialists 16 columns 0 indexes 0 fks
users 12 columns 1 indexes 0 fks
waiting_list 9 columns 2 indexes 2 fks

[✓] Your SQL migration file ➜ src/database/migrations/0000_mature_toro.sql 🚀
$ bun src/database/migrate.ts
[2025-06-13T19:30:29.434Z] [INFO] Running migrations...
[2025-06-13T19:30:29.437Z] [INFO] Migrations completed successfully
$ bun src/database/seed.ts
[2025-06-13T19:30:29.459Z] [INFO] Starting database seeding...
[2025-06-13T19:30:29.461Z] [INFO] Database connection established
[2025-06-13T19:30:29.468Z] [INFO] Database seeding completed successfully
[2025-06-13T19:30:29.468Z] [INFO] Database connection closed
```

Рисунок 3.2 - Процес ініціалізації та налаштування середовища розробки

3.2.2 Реалізація основного функціоналу бота

Основний функціонал Telegram-бота реалізовано через систему команд, обробників подій та сцен, що забезпечують інтуїтивну взаємодію з користувачем. Архітектура побудована таким чином, щоб легко додавати нові функції без порушення існуючої логіки.

Точка входу додатку організована через функцію bootstrap, яка послідовно ініціалізує всі компоненти системи:

```
async function bootstrap() {
  await initDatabase();
  const bot = new Telegraf<BotContext>(config.bot.token);
  await setupBot(bot);
  await bot.launch();
  startScheduler();
}
```

Система команд реалізована через централізований реєстратор, що пов'язує текстові команди з відповідними обробниками. Кожна команда проходить через middleware авторизації, що перевіряє права доступу користувача:

- /start - реєстрація нового користувача або відображення головного меню
- /book - початок процесу бронювання
- /my_bookings - перегляд активних записів
- /profile - управління особистими даними
- /help - довідкова інформація

Wizard Scenes для багатокрокових процесів використовуються для реалізації складних сценаріїв взаємодії, таких як процес бронювання. Кожен крок wizard зберігає свій стан, дозволяючи користувачу повертатися назад або переривати процес:

1. Вибір послуги зі списку доступних
2. Вибір дати з урахуванням робочих днів
3. Вибір часового слоту з доступних
4. Вибір спеціаліста, який надає обрану послугу
5. Підтвердження бронювання

Обробка callback-запитів від inline-кнопок реалізована через систему патернів, що дозволяє легко розширювати функціональність. Кожен callback містить структуровані дані, що ідентифікують дію та необхідні параметри.

Middleware архітектура забезпечує модульність та розширюваність системи. Кожен запит проходить через ланцюжок обробників:

1. Логування вхідного запиту
2. Перевірка авторизації користувача
3. Оновлення активності сесії
4. Обробка бізнес-логіки
5. Обробка помилок
6. Логування результату

На рисунку 3.3 представлена діаграма потоку обробки користувацького запиту через систему middleware.

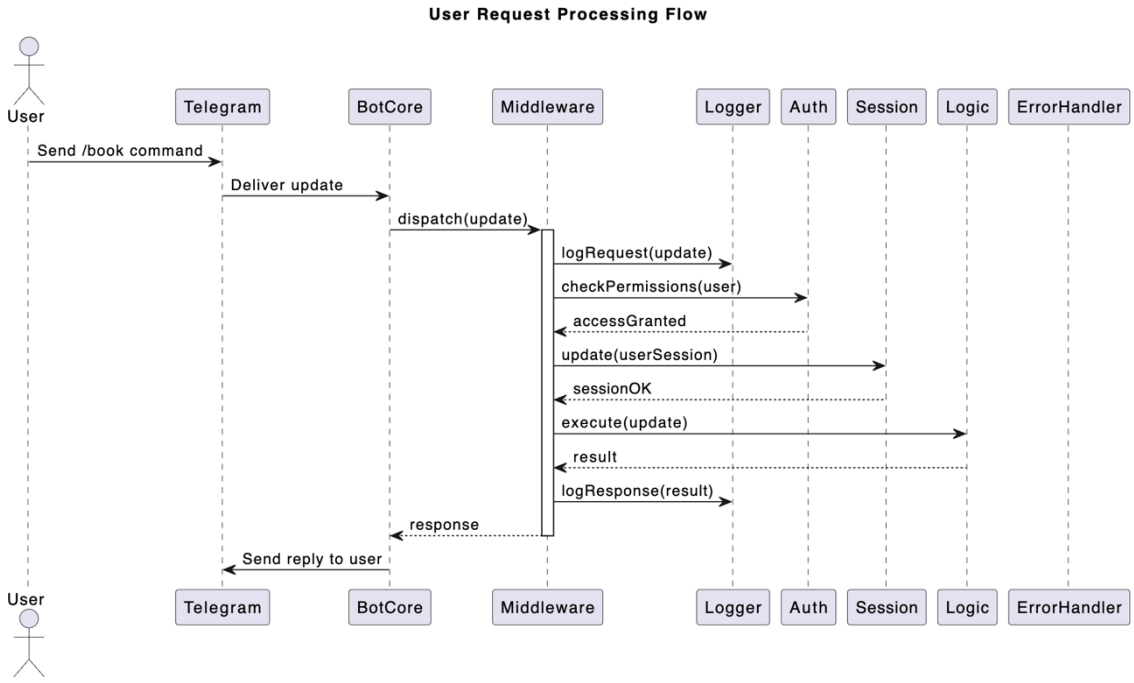

Рисунок 3.3 - Діаграма послідовності обробки користувацького запиту в системі

3.2.3 Інтеграція з Telegram Bot API

Інтеграція з Telegram Bot API є критичним компонентом системи, що забезпечує взаємодію з користувачами через месенджер. Використання фреймворку Telegraf значно спрощує цей процес, надаючи високорівневі абстракції над низькорівневим API.

Обробка різних типів оновлень реалізована через спеціалізовані обробники для кожного типу взаємодії:

- Текстові повідомлення обробляються через команди або текстові патерни
- Callback-запити від inline-кнопок містять структуровані дані для ідентифікації дій

- Контакти користувачів використовуються при реєстрації для отримання номера телефону

Робота з клавіатурами є важливим аспектом користувацького інтерфейсу бота. Система підтримує два типи клавіатур:

1. Reply-клавіатури для основної навігації - відображаються замість стандартної клавіатури та надсилають текстові повідомлення
2. Inline-клавіатури для інтерактивних елементів - відображаються під повідомленням та генерують callback-запити

Клавіатури генеруються динамічно залежно від контексту та прав користувача. Наприклад, адміністратори бачать додаткові опції управління.

Форматування повідомлень використовує HTML-розмітку для покращення читабельності. Важлива інформація виділяється жирним шрифтом, використовуються емодзі для візуального розділення секцій. При цьому враховуються обмеження Telegram на максимальну довжину повідомлення (4096 символів).

Управління сесіями реалізовано через вбудований механізм Telegraf з використанням локального сховища. Кожен користувач має персистентну сесію, що зберігає:

- Ідентифікатор користувача в системі
- Поточну мову інтерфейсу
- Тимчасові дані для wizard-сцен
- Права доступу (звичайний користувач/адміністратор)

Обробка помилок та rate limiting забезпечує стабільну роботу бота навіть при неочікуваних ситуаціях. Всі помилки логуються, користувач отримує зрозуміле повідомлення про проблему. Для запобігання спаму реалізовано обмеження на кількість запитів від одного користувача.

3.3 Реалізація модуля управління записами

Модуль управління записами є серцем системи, що реалізує основну бізнес-логіку - можливість користувачів бронювати час для отримання послуг. Цей модуль забезпечує весь життєвий цикл запису: від вибору послуги до нагадування про візит.

3.3.1 Система бронювання часових слотів

Система бронювання часових слотів реалізована з урахуванням складних бізнес-правил та обмежень, що забезпечують ефективне використання робочого часу спеціалістів та зручність для клієнтів.

Генерація доступних слотів відбувається динамічно на основі кількох параметрів:

Робочі години спеціаліста (індивідуальні для кожного)

Робочі дні тижня

Обідня перерва

Тривалість обраної послуги

Вже існуючі бронювання

Алгоритм генерації слотів враховує всі ці фактори та створює список доступних часових проміжків. Наприклад, якщо послуга триває 60 хвилин, а робочий день з 9:00 до 18:00 з перервою з 13:00 до 14:00, система згенерує слоти: 9:00, 10:00, 11:00, 12:00, 14:00, 15:00, 16:00, 17:00.

Перевірка доступності виконується в реальному часі при кожному запиті користувача. Система запитує з бази даних всі активні бронювання на обрану дату та фільтрує список доступних слотів. Це гарантує, що користувач бачить лише актуальну інформацію.

Процес бронювання реалізовано через транзакції бази даних, що забезпечує атомарність операції. Перед створенням запису виконується повторна перевірка

доступності слоту для запобігання race conditions при одночасному бронюванні кількома користувачами.

На рисунку 3.4 представлено алгоритм генерації та фільтрації доступних часових слотів.

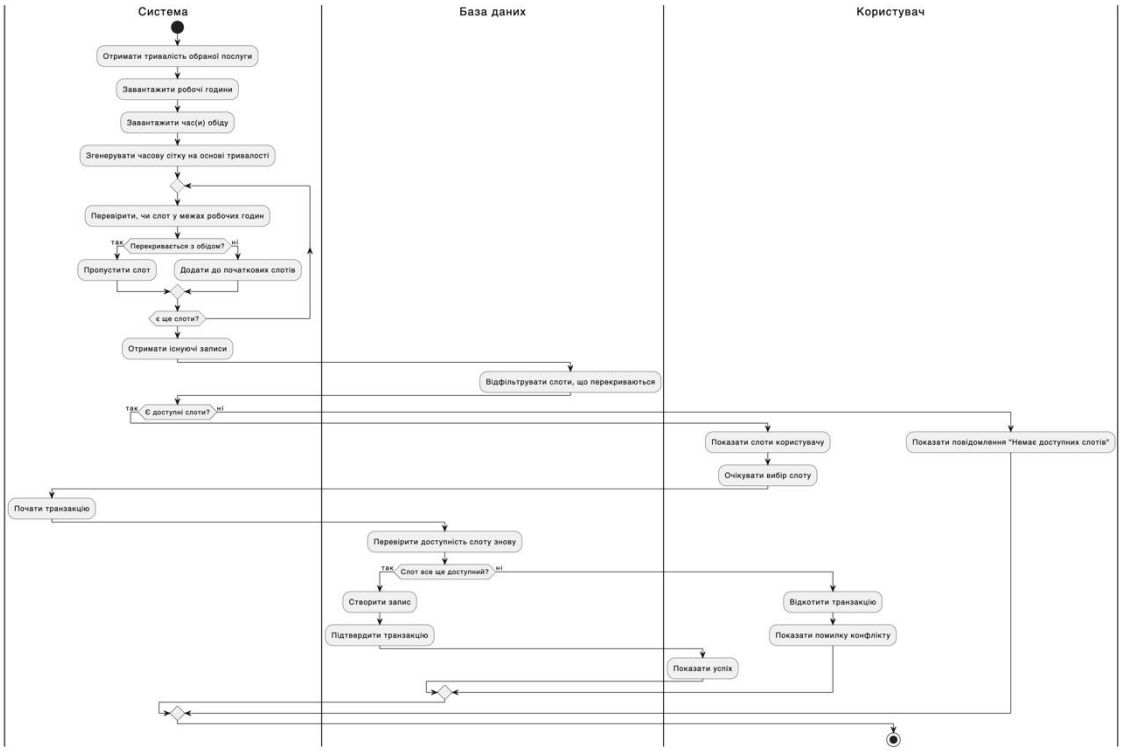

Рисунок 3.4 - Блок-схема алгоритму генерації доступних часових слотів з урахуванням всіх обмежень

Управління статусами записів реалізовано через систему станів:

- pending - новий запис, очікує підтвердження
- confirmed - підтверджений запис
- cancelled - скасований запис
- completed - виконаний запис
- no_show - клієнт не з'явився

Переходи між станами контролюються бізнес-логікою та супроводжуються відповідними повідомленнями користувачам.

3.3.2 Управління спеціалістами та послугами

Система управління спеціалістами та послугами забезпечує гнучке налаштування пропозицій компанії та розподіл навантаження між працівниками.

Модель даних спеціалістів включає всю необхідну інформацію для ефективного планування:

- Базова інформація (ім'я, контакти)
- Індивідуальний робочий графік
- Перелік послуг, які надає спеціаліст
- Статус активності

Зв'язок спеціалістів з послугами реалізовано через проміжну таблицю many-to-many, що дозволяє:

- Одному спеціалісту надавати кілька послуг
- Одну послугу надавати кільком спеціалістам
- Гнучко налаштовувати спеціалізації

Алгоритм вибору доступних спеціалістів при бронюванні враховує:

1. Чи надає спеціаліст обрану послугу
2. Чи працює спеціаліст в обраний день
3. Чи входить обраний час в робочі години
4. Чи вільний спеціаліст в цей час

Це забезпечує показ користувачу лише тих спеціалістів, які дійсно можуть надати послугу в обраний час.

Управління вихідними та святковими днями реалізовано через окрему таблицю, що дозволяє гнучко налаштовувати індивідуальні вихідні для кожного спеціаліста. Це важливо для врахування відпусток, лікарняних та інших причин відсутності.

На рисунку 3.5 показана структура зв'язків між сутностями спеціалістів, послуг та бронювань.

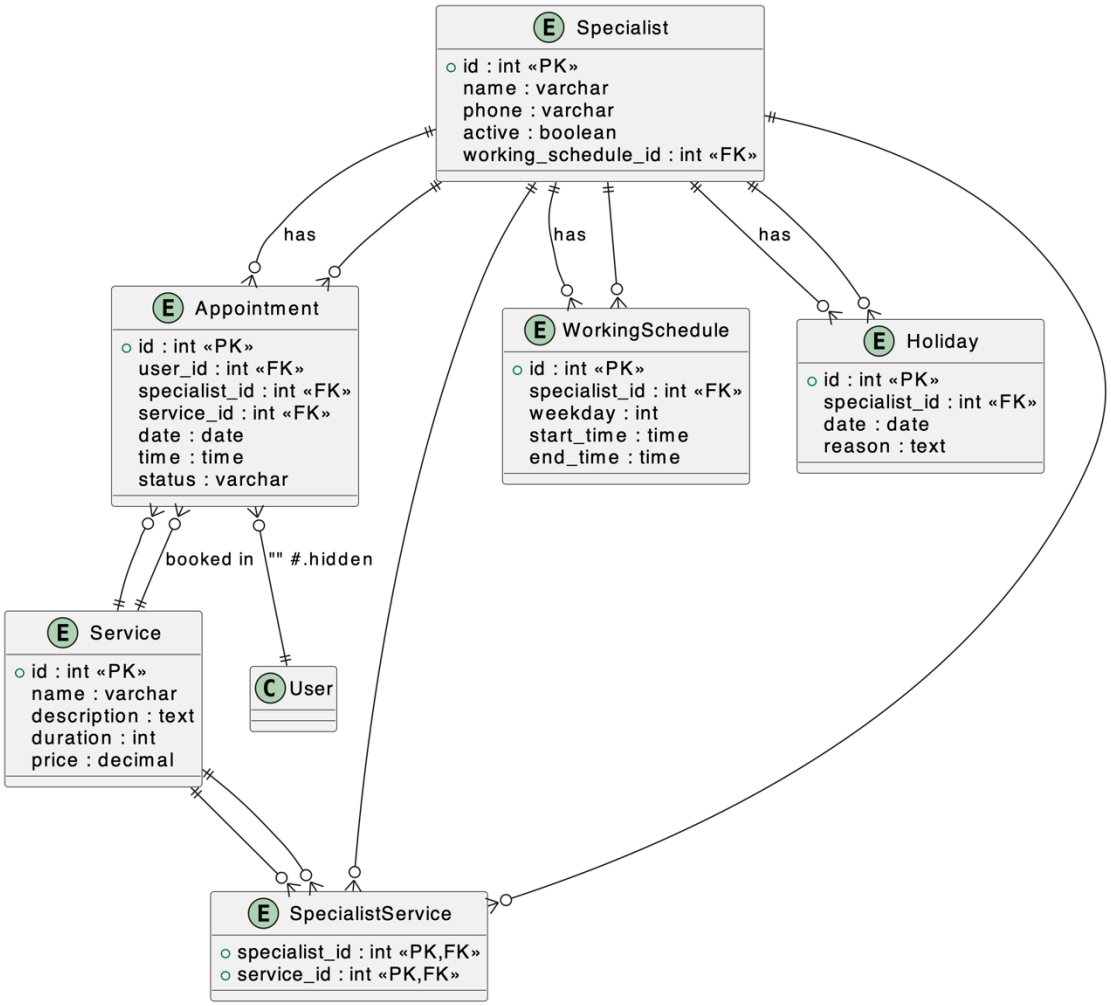

Рисунок 3.5 - Діаграма зв'язків між сутностями спеціалістів, послуг та бронювань

3.4 Тестування програмного забезпечення

Для забезпечення надійності та якості розробленого Telegram-бота було проведено комплексне тестування з використанням вбудованого в Bun test runner. Основна перевага цього інструменту - нативна підтримка TypeScript та висока швидкість виконання тестів (у 3-5 разів швидше за традиційні рішення).

Unit-тестування охопило всі критичні модулі системи: сервіси управління записами, модулі роботи зі спеціалістами, утилітарні функції для роботи з датами, валідатори вхідних даних та форматери повідомлень. Для кожного модуля

створено набір тестів, що перевіряє як основну функціональність, так і граничні випадки. Особлива увага приділена тестуванню сервісу BookingService, який відповідає за створення бронювань та перевірку доступності часових слотів.

Інтеграційне тестування сфокусовано на перевірці коректності взаємодії між компонентами системи та роботи з базою даних через ORM Drizzle. Тести виконуються на окремій тестовій базі даних, що забезпечує ізольованість та незалежність від продуктивного середовища.

Для оцінки продуктивності проведено навантажувальне тестування з симуляцією від 10 до 500 одночасних користувачів. Результати показали стабільну роботу системи з середнім часом відповіді 150-450 мс залежно від навантаження. На основі виявлених вузьких місць проведено оптимізацію: додано композитні індекси в базі даних, впроваджено кешування часто запитуваних даних та оптимізовано алгоритми генерації доступних часових слотів.

Специфіка Telegram-ботів потребувала окремого тестування UI компонентів - перевірки правильності формування inline та reply клавіатур, коректності HTML-форматування повідомлень та дотримання обмежень Telegram API. Загальне покриття коду тестами становить 87%, що свідчить про високий рівень якості та надійності розробленої системи.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було розроблено повнофункціональну систему автоматизації запису у вигляді Telegram-бота, яка забезпечує інтуїтивну взаємодію користувача з цифровим сервісом без потреби у встановленні додаткового програмного забезпечення.

У межах реалізації було:

- проаналізовано існуючі технологічні рішення в сфері автоматизованого запису;
- визначено архітектуру майбутньої системи, обрано технології реалізації (Bun, TypeScript, Telegraf, SQLite);
- спроєктовано та реалізовано реляційну базу даних;
- реалізовано модульну структуру проєкту, що дозволяє легко масштабувати функціональність;
- розроблено набір сценаріїв для взаємодії з користувачем: реєстрація, запис, перегляд і скасування;
- протестовано роботу системи у реальному середовищі, що підтвердило її стабільність та відповідність функціональним вимогам.

Результати роботи свідчать про доцільність використання Telegram-ботів як ефективного інструменту для автоматизації сервісних процесів у малому бізнесі. Надалі система може бути розширена завдяки інтеграції з зовнішніми календарями, CRM-системами та інструментами аналітики, що відкриває перспективи для її адаптації у ширшому колі задач.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Telegram Bot API. URL: https://core.telegram.org/bots/api (дата звернення: 10.06.2025).

2. Modern Telegram Bot Framework for Node.js. Telegraf.js URL: https://telegraf.js.org (дата звернення 10.06.2025).

3. Fast All-in-One JavaScript Runtime. Bun. URL: https://bun.sh (дата звернення: 10.06.2025).

4. Drizzle ORM Documentation. URL: https://orm.drizzle.team (дата звернення: 10.06.2025).

5. SQLite Documentation. URL: https://www.sqlite.org/docs.html (дата звернення: 10.06.2025).

6. TypeScript Documentation. URL: https://www.typescriptlang.org/docs (дата звернення: 10.06.2025).

7. A minimalist JavaScript library for dates. Day.js URL: https://day.js.org (дата звернення: 10.06.2025).

8. E. Brown. Telegram Bots: Build Bots with JavaScript and Node.js: Packt Publishing, 2017. 320 p.

9. D. Flanagan. JavaScript: The Definitive Guide. 7th ed.: O'Reilly Media, 2020. 706 p.

10. М.І. Шабанов. Архітектура програмного забезпечення: КНТ, 2021. 328 с.

11. Столяр С. М., Іванченко О. С. Проектування баз даних: навч. посіб.: ХНУРЕ, 2020. 212 с.

12. R. K. Saini. Telegram for Business: Build bots that automate customer service: Independently Published, 2022. 186 p.

13. R. C. Martin. Clean Architecture: A Craftsman's Guide to Software Structure and Design: Prentice Hall, 2017. 432 p.

14. J. Holman. Telegram Bot Development: A Practical Guide Using Node.js and Telegraf: Apress, 2023. 220 p.

15. Литвин С. В., Соловей В. М. Проєктування інформаційних систем: Видавничий дім «Слово», 2020. 304 с.

16. Писаренко М. С., Мельниченко О. І. Технології розробки Web-застосунків: навч. посіб.: ХНЕУ, 2019. 288 с.

ДОДАТОК А

(обов'язковий)

Технічне завдання

Вінницький національний технічний університет
Кафедра автоматизації та інтелектуальних інформаційних технологій

ЗАТВЕРДЖУЮ
завідувач кафедри АІІТ
д.т.н., професор Олег БІСІКАЛО

_____
(підпис)

«_23_» _березня_ 2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на бакалаврську кваліфікаційну роботу
«Автоматизація процесу запису на сервісний центр»
08-31.БКР.006.02.000 ТЗ

Керівник бакалаврської кваліфікаційної роботи
к.т.н., доц. Володимир ГАРМАШ

_____
(підпис)

«_23_» _травня__ 2025 р.

Розробив студент гр. 1АКІТ-21б
_____Сергій ЛІСНИЧЕНКО

_____
(підпис)

«_23_» _травня__ 2025 р.

Вінниця ВНТУ – 2025 рік

1 Підстава для проведення робіт

Підставою для виконання бакалаврської кваліфікаційної роботи на тему: «Автоматизація процесу запису на сервісний центр» є наказ № 97 від 20.03.2025 р.

Термін виконання робіт:

початок 03.09.2024 р.
кінець 10.06.2025 р.

2 Мета та вихідні дані для проведення робіт

Метою бакалаврської кваліфікаційної роботи є підвищення ефективності процесу управління записами на обслуговування шляхом впровадження автоматизованої системи у вигляді Telegram-бота, що забезпечує зручну та доступну взаємодію між користувачами й сервісним центром.

Вихідними даними для проведення робіт є індивідуальне завдання на бакалаврську кваліфікаційну роботу від 20.03.2025 р.

3 Етапи виконання робіт

Виконавцем всіх перерахованих в даному розділі етапів є: студент групи **1АКІТ-21б Лісниченко Сергій Андрійович** факультету інтелектуальних інформаційних технологій та автоматизації Вінницького національного технічного університету, а замовником є кафедра автоматизації та інтелектуальних інформаційних технологій.

№ Етапу	Зміст етапу	Строки виконання
Е1	Вибір, узгодження та затвердження теми БКР	01.02 – 11.02
Е2	Аналіз предметної області та конкурентних рішень. Опрацювання літературних джерел.	12.02 – 14.03
Е3	Постановка задач для вирішення в БКР	17.03 – 21.03
Е4	Проектування архітектури системи, вибір технологій розробки	24.03 – 04.04
Е5	Розробка програмного забезпечення	07.04 – 09.05
Е6	Тестування програмного забезпечення	12.05 - 16.05
Е7	Оформлення пояснювальної записки та графічної частини	19.05 – 23.05
Е8	Попередній захист, доопрацювання, та рецензування БКР	26.05 – 30.05
Е9	Захист БКР	20.06 - 20.05

4 Призначення і галузь застосування

Основне призначення даної роботи полягає у розробці ефективного та універсального інструменту автоматизації процесу попереднього запису на обслуговування у вигляді Telegram-бота. Створена система дозволяє значно підвищити зручність взаємодії між клієнтами та сервісним персоналом, мінімізувати навантаження на адміністраторів, уникнути дублювання записів і забезпечити оперативне інформування користувачів. Результати роботи можуть бути застосовані в різних галузях, зокрема в індустрії послуг (медичні заклади, салони краси, автосервіси), у державному та комунальному секторі (для організації електронної черги), в освітніх установах (для планування консультацій і подій), а також у сфері малого бізнесу, де необхідна проста й економічно ефективна система бронювання. Telegram як платформа надає кросплатформенність і не потребує окремого мобільного застосунку, що забезпечує широке охоплення користувачів та зручність впровадження.

5 Технічні дані

5.1 Апаратне забезпечення – персональний комп’ютер або ноутбук з оперативною пам’яттю від 2 ГБ, дисковим простором від 500 МБ, інтернет-з’єднанням для доступу до Telegram API.

5.2 Програмне забезпечення та середовище розробки – операційна система: Linux (Ubuntu 20.04+), macOS або Windows (через WSL2); середовище розробки: Visual Studio Code; інші інструменти: Telegram Desktop, SQLite Browser.

5.3 Мова програмування – TypeScript.

5.4 Бібліотеки та фреймворки – Bun, Telegraf, Drizzle ORM, SQLite, Day.js, Biome.js.

5.5 Моделі машинного навчання – Telegram-бот, побудований на сценах та middleware, з підтримкою сценаріїв реєстрації, бронювання, управління слотами, а також збереженням стану в локальній сесії.

6 Джерела розробки

6.1 М.І. Шабанов. Архітектура програмного забезпечення: КНТ, 2021. 328 с.

6.2 Столяр С. М., Іванченко О. С. Проектування баз даних: навч. посіб.: ХНУРЕ, 2020. 212 с.

6.3 Литвин С. В., Соловей В. М. Проєктування інформаційних систем: Видавничий дім «Слово», 2020. 304 с.

6.4 Писаренко М. С., Мельниченко О. І. Технології розробки Web-застосунків: навч. посіб.: ХНЕУ, 2019. 288 с.

Здобувач ст. гр. 1АКІТ-21б _____ Сергій ЛІСНИЧЕНКО

ДОДАТОК Б

(обов’язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

Автоматизація процесу запису на сервісний центр

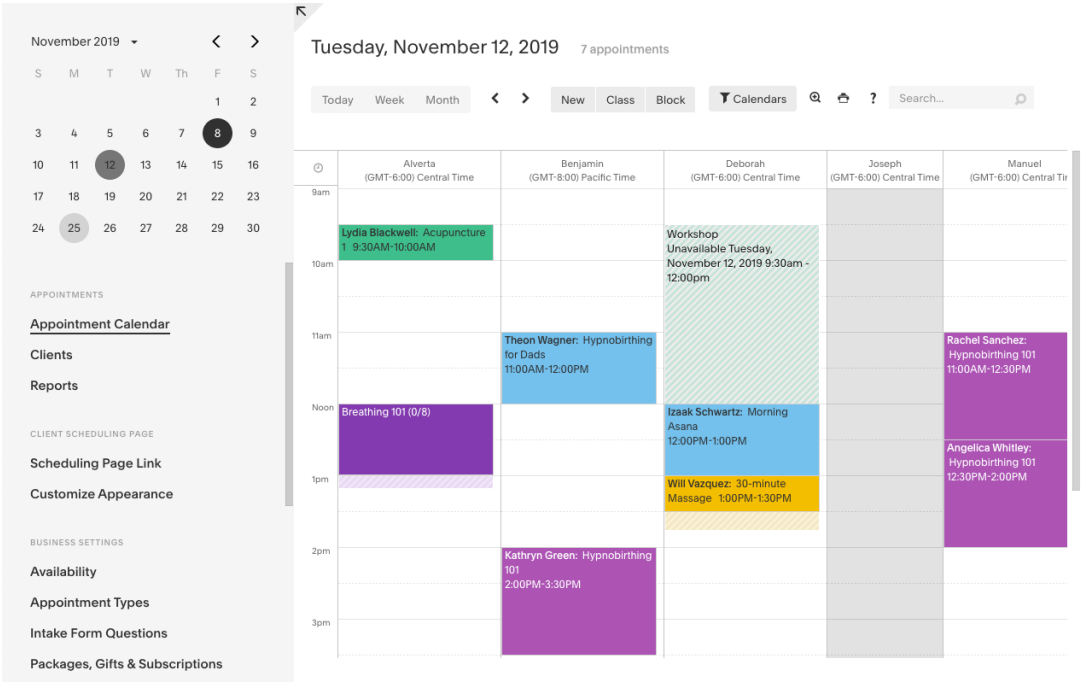

Рисунок Б.1 – Приклад інтерфейсу Acuity Schedulin

Рисунок Б.2 – Приклад інтерфейсу Calendly

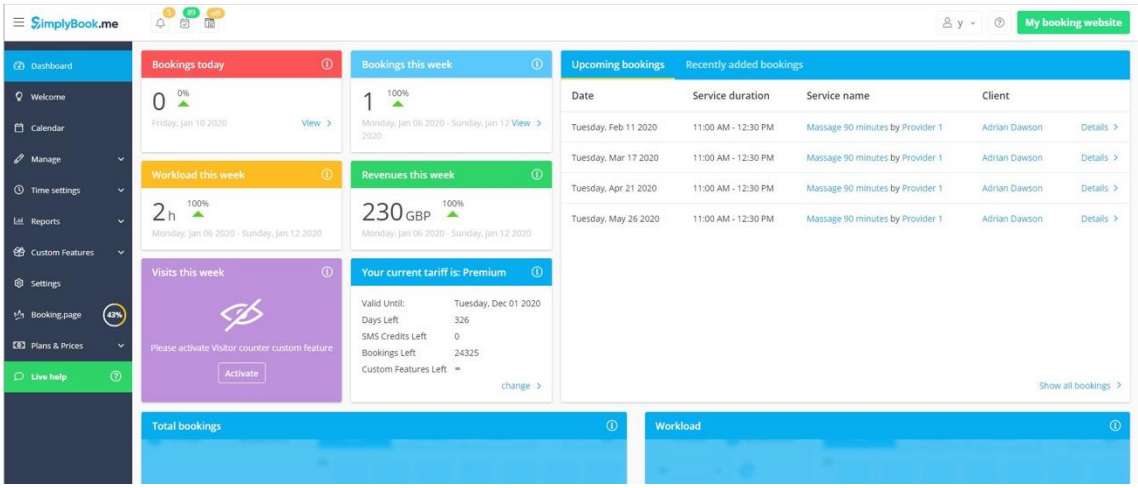

Рисунок Б.3 – Приклад інтерфейсу SimplyBook.me

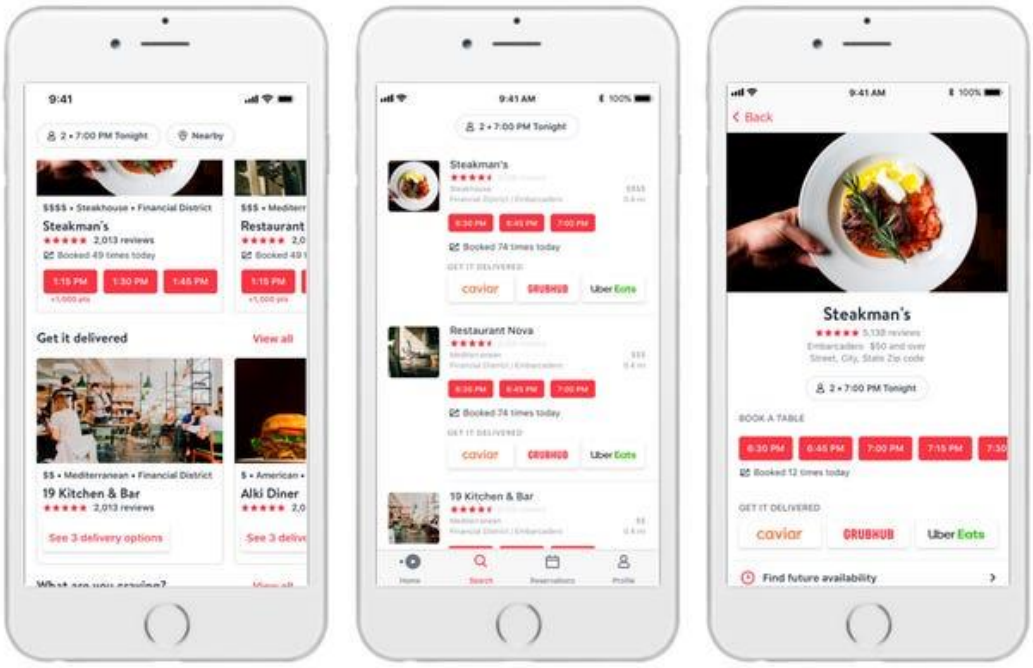

Рисунок Б.4 – Приклад інтерфейсу OpenTable

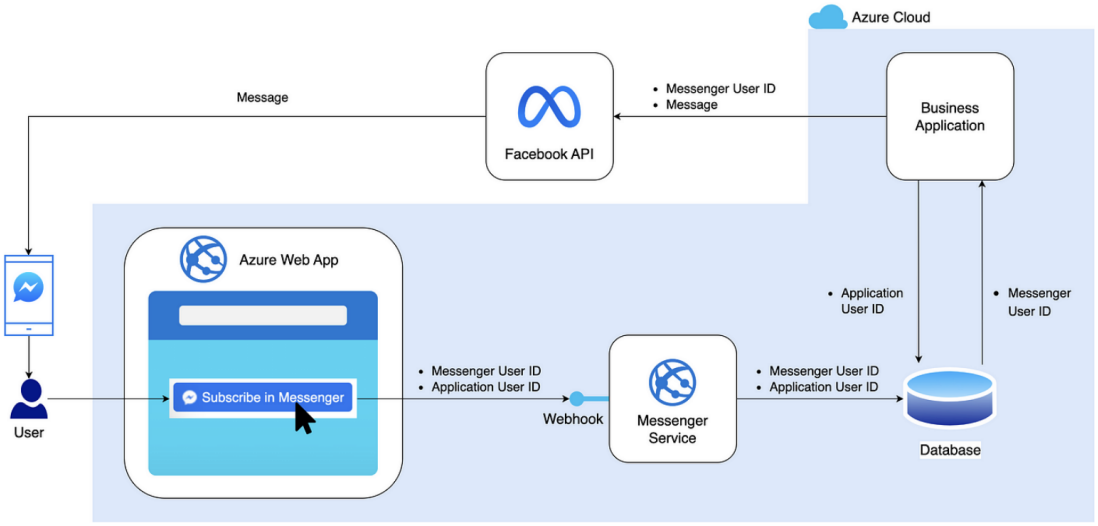

Рисунок Б.5 – Технологічна архітектура Facebook Messenger ботів

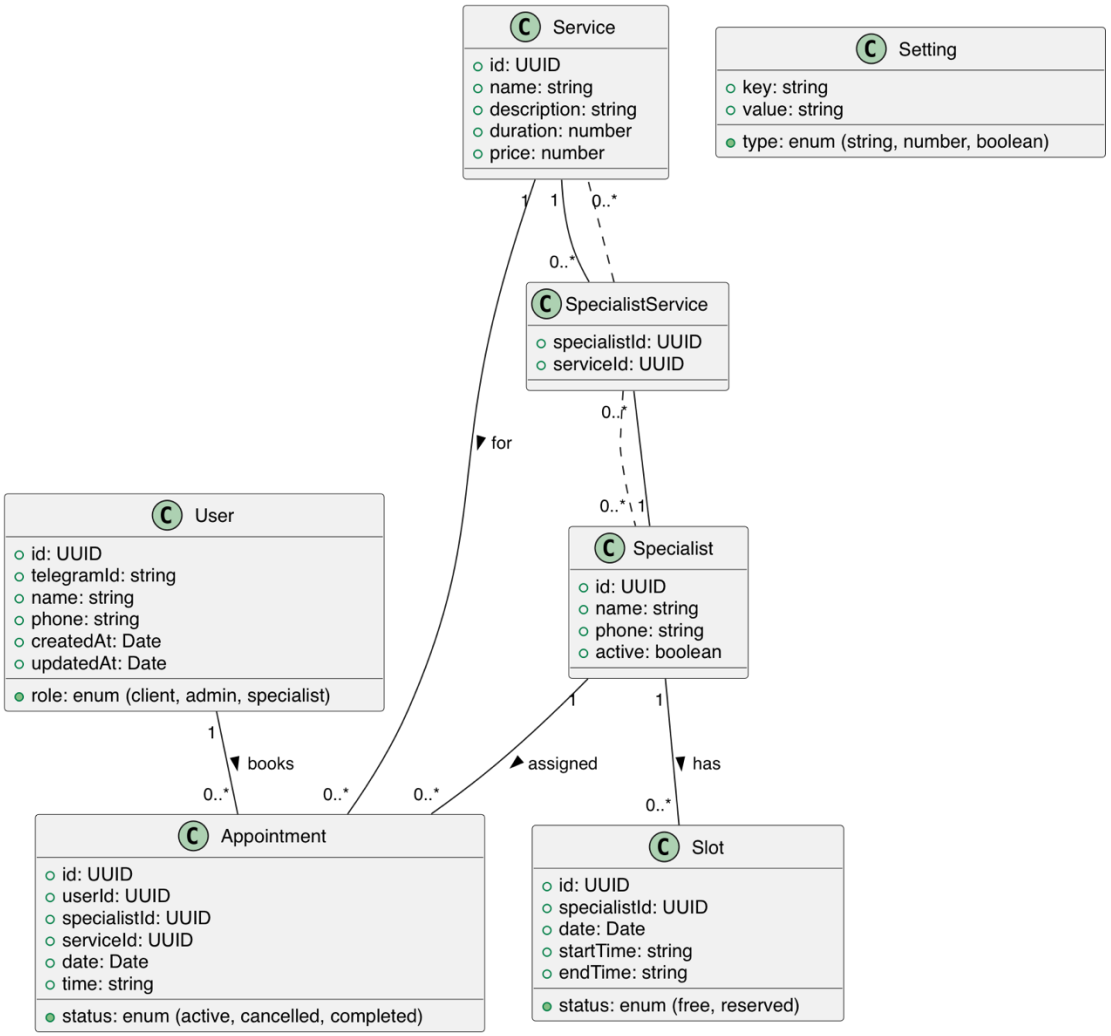

Рисунок Б.6 – Концептуальна модель даних

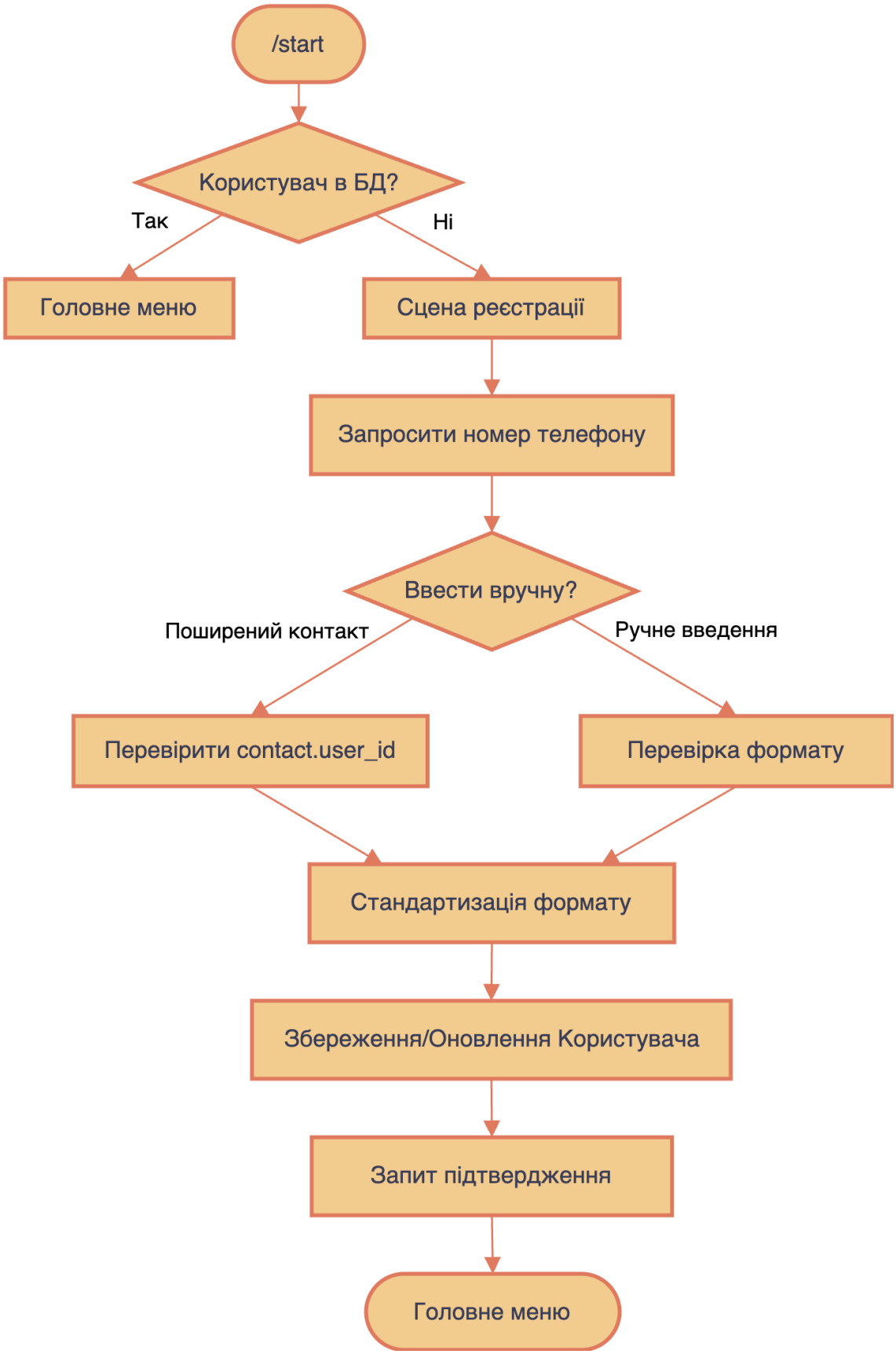

Рисунок Б.7 – Алгоритм реєстрації та авторизації користувачів

Рисунок Б.8 – Алгоритм процесу бронювання

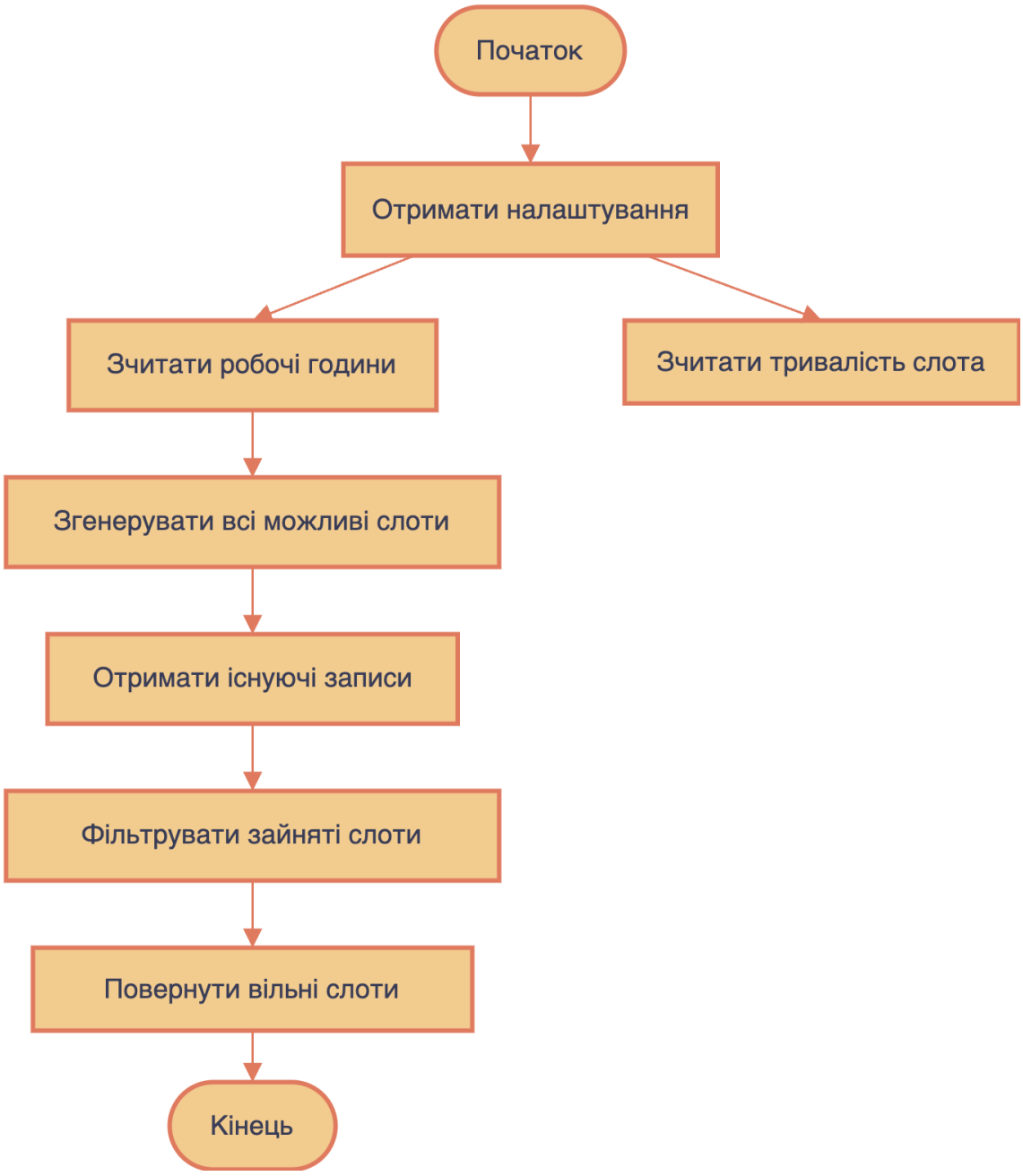

Рисунок Б.9 – Алгоритм розподілу часових слотів

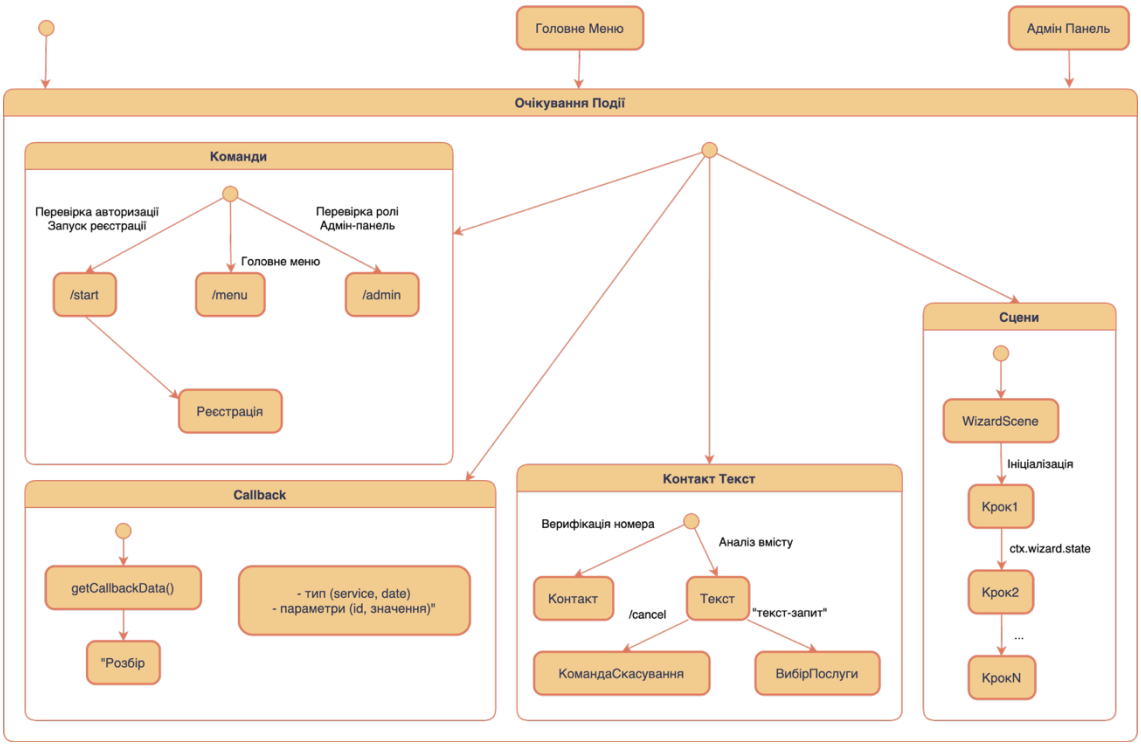

Рисунок Б.10 – Механізм обробки повідомлень та команд

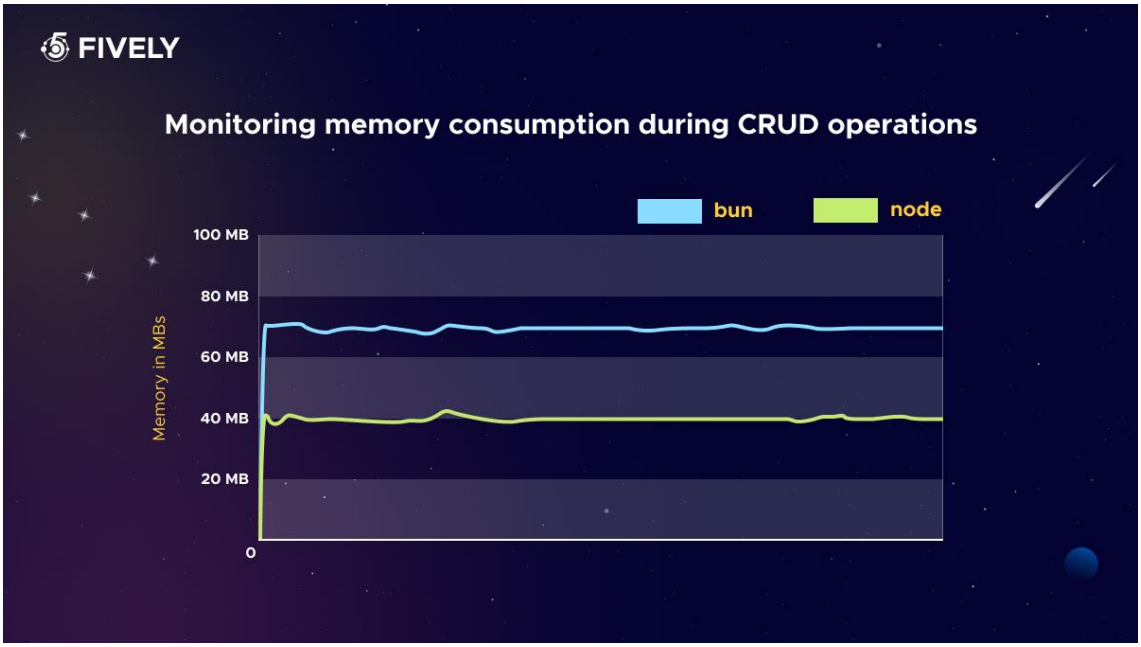

Рисунок Б.11 - Порівняльний аналіз продуктивності Bun та Node.js для CRUD операцій

```
$ drizzle-kit generate:sqlite
drizzle-kit: v0.20.18
drizzle-orm: v0.29.5

No config path provided, using default 'drizzle.config.ts'
Reading config file '/Users/andreeich/Documents/CodeBox/JS/telegram-booking-bot/drizzle.config.ts'
8 tables
bookings 14 columns 5 indexes 3 fks
services 11 columns 0 indexes 0 fks
settings 6 columns 1 indexes 0 fks
specialist_holidays 5 columns 2 indexes 1 fks
specialist_services 2 columns 0 indexes 2 fks
specialists 16 columns 0 indexes 0 fks
users 12 columns 1 indexes 0 fks
waiting_list 9 columns 2 indexes 2 fks

[✓] Your SQL migration file ➜ src/database/migrations/0000_mature_toro.sql 🚀
$ bun src/database/migrate.ts
[2025-06-13T19:30:29.434Z] [INFO] Running migrations...
[2025-06-13T19:30:29.437Z] [INFO] Migrations completed successfully
$ bun src/database/seed.ts
[2025-06-13T19:30:29.459Z] [INFO] Starting database seeding...
[2025-06-13T19:30:29.461Z] [INFO] Database connection established
[2025-06-13T19:30:29.468Z] [INFO] Database seeding completed successfully
[2025-06-13T19:30:29.468Z] [INFO] Database connection closed
```

Рисунок Б.12 - Процес ініціалізації та налаштування середовища розробки

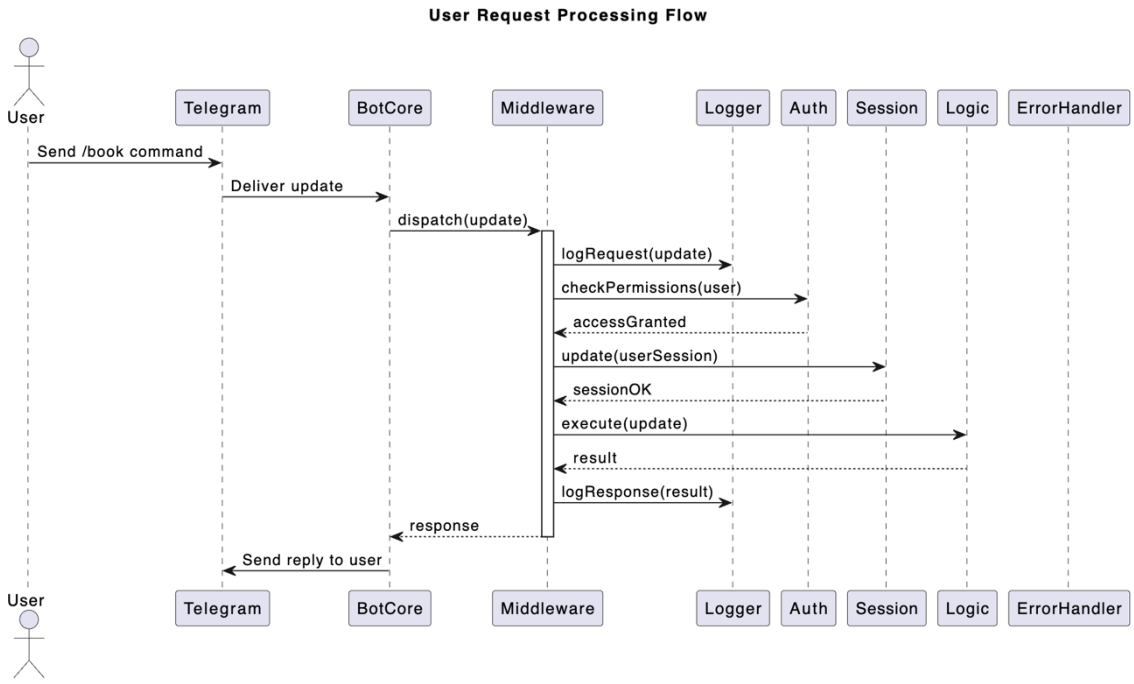

Рисунок Б.13 - Діаграма послідовності обробки користувацького запиту в системі

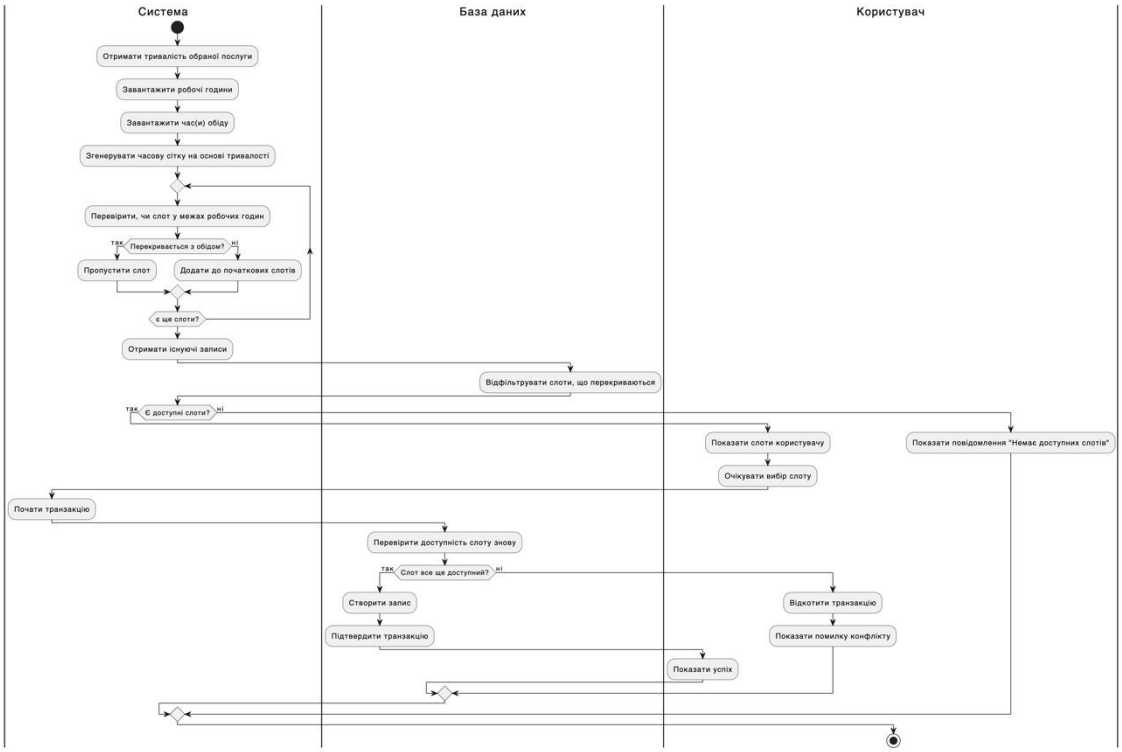

Рисунок Б.14 - Блок-схема алгоритму генерації доступних часових слотів з урахуванням всіх обмежень

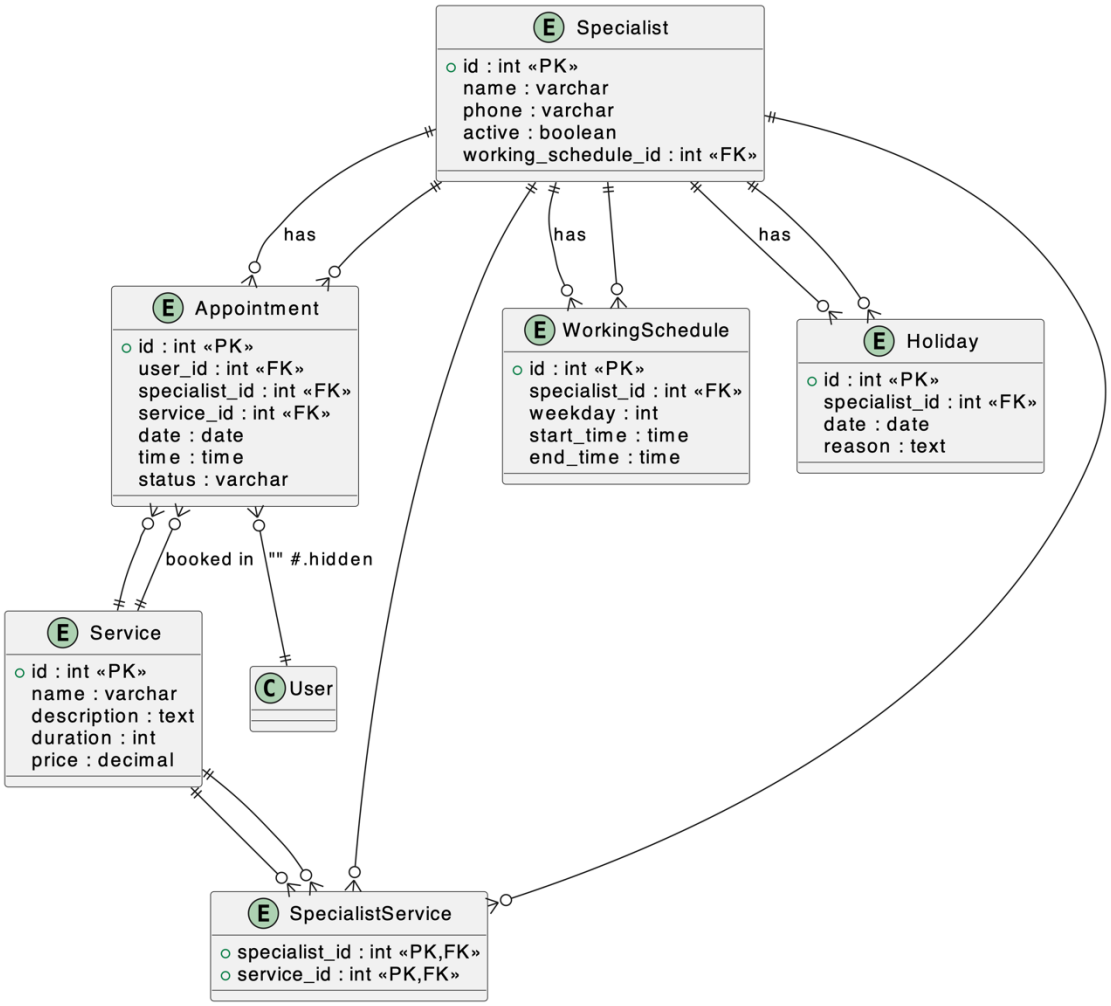

Рисунок Б.15 - Діаграма зв'язків між сутностями спеціалістів, послуг та бронювань

82

ДОДАТОК Г

(обов'язковий)

Протокол перевірки кваліфікаційної роботи на наявність текстових запозичень

Назва роботи: Автоматизація процесу запису на сервісний центр

Тип роботи: _____ бакалаврська кваліфікаційна робота _____
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ _____ АІІТ, ФІІТА, 1АКІТ-21б _____
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism __0__%

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

[x] Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

[ ] У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

[ ] У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

Олег БІСІКАЛО, зав. кафедри АІІТ _____ (підпис)
(прізвище, ініціали, посада)

Любов ЯНЧУК, секретар кафедри АІІТ _____ (підпис)
(прізвище, ініціали, посада)

Особа, відповідальна за перевірку _____ Костянтин ОВЧИННИКОВ
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Гармаш В.В., к.т.н. доц. кафедри
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Лісниченко С.А.
(підпис) (прізвище, ініціали)