

**ПРИВАТНЕ АКЦІОНЕРНЕ ТОВАРИСТВО «ВИЩИЙ
НАВЧАЛЬНИЙ ЗАКЛАД «МІЖРЕГІОНАЛЬНА АКАДЕМІЯ
УПРАВЛІННЯ ПЕРСОНАЛОМ»**

Факультет комп'ютерно-інформаційних технологій

Кафедра комп'ютерних інформаційних систем і технологій

Новак Вікторія Олександрівна

КВАЛІФІКАЦІЙНА РОБОТА

на тему: **«Розробка інформаційної системи автоматизації обліку клієнтів
та надання послуг»**

Група: ІК-9-22-Б1КНТ(4.0д)

Спеціальність: Комп'ютерні науки

Рівень вищої освіти: перший (бакалаврський)

Кваліфікаційна робота містить результати власних досліджень.

*Використання ідей, результатів, текстів інших авторів мають
посилання на відповідне джерело.*

Науковий керівник

професор

(підпис)

Новак В. О.

Заліський М.Ю., д.т.н.,

(підпис)

Допущено до захисту ЕК

Завідувач кафедри

(підпис)

Сергій КАВУН, д.е.н., професор

Київ 2026

Зміст

Вступ	4
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ФОРМУВАННЯ ВИМОГ ДО СИСТЕМИ	9
1.1 Опис бізнес-процесів та обґрунтування необхідності автоматизації	9
1.2 Порівняльний аналіз існуючих аналогів та обґрунтування власної розробки.....	13
1.3 Функціональні та технологічні вимоги до інформаційної системи	19
РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА ОБҐРУНТУВАННЯ АРХІТЕКТУРИ ІНФОРМАЦІЙНОЇ СИСТЕМИ	24
2.1 Обґрунтування вибору стеку технологій та архітектурних рішень	24
2.2 Інфологічне та даталогічне проектування бази даних.....	28
2.3 Об’єктно-орієнтоване моделювання поведінки та взаємодії компонентів системи.....	38
2.4 Проектування та опис програмного інтерфейсу REST API.....	40
РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ МОДУЛІВ СИСТЕМИ	48
3.1 Опис реалізації основних програмних компонентів.....	48
3.2 Програмна реалізація структури даних та серверних алгоритмів бізнес-логіки.....	53
3.3 Тестування працездатності та оцінка ефективності впровадження системи.....	55

ВИСНОВКИ.....	61
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	60
ДОДАТОК А.....	65
ДОДАТОК Б.....	67
ДОДАТОК В.....	69

Вступ

У сучасних умовах динамічного розвитку інформаційних технологій та глобальної цифровізації бізнес-процесів, ефективність функціонування будь-якого підприємства, що надає послуги, напряму залежить від швидкості та якості обробки інформації.

Об'єктом дослідження під час проходження переддипломної практики виступило Товариство з обмеженою відповідальністю «Звягель Медіа Груп». Основним напрямком діяльності даного підприємства є видавнича, інформаційна та рекламна діяльність, що включає випуск періодичних друкованих видань, забезпечення функціонування місцевих цифрових інформаційних ресурсів, а також надання широкого спектра рекламних послуг для приватних осіб та корпоративних клієнтів. Специфіка функціонування ТОВ «Звягель Медіа Груп» передбачає щоденну обробку значних обсягів інформації, яка надходить від контрагентів. Організаційна структура підприємства для забезпечення цих процесів містить такі ключові ланки:

- **Адміністрація та редакційна колегія:** відповідає за стратегічне планування, затвердження медіа-планів, формування цінової політики на рекламні площі та координацію роботи всіх підрозділів.
- **Відділ прийому замовлень та реклами (Менеджери):** здійснює безпосередню взаємодію з клієнтами (читачами, рекламодавцями), реєструє їхні анкетні дані, приймає тексти оголошень, макети банерів, фіксує обрані параметри публікації (рубрика, обсяг, терміни виходу) та контролює надходження оплати.
- **Технічний та дизайнерський відділ:** виконує верстку друкованих шпальт, підготовку графічних матеріалів до публікації та технічне розміщення контенту на інформаційних платформах.

Аналіз поточної організації бізнес-процесів у товаристві виявив наявність низки «вузьких місць» у системі обліку. Прийом заявок від рекламодавців та фіксація замовлень часто здійснюються із використанням розрізнених паперових журналів або локальних текстових таблиць. Це призводить до дублювання інформації, створює ризики втрати контактних даних клієнтів або помилок під час розрахунку вартості комплексних медіа-послуг. Окрім цього, відсутність єдиної централізованої бази даних ускладнює оперативний обмін інформацією між менеджерами та технічним відділом, і уповільнює процес формування аналітичної звітності для керівництва щодо ефективності продажів рекламних площ за звітний період. Саме тому розробка та впровадження спеціалізованої інформаційної системи автоматизації обліку клієнтів та надання послуг є критично необхідним кроком для компанії. Це дозволить консолідувати дані про контрагентів, автоматизувати розрахунок вартості замовлень, забезпечити прозорий контроль за етапами виконання заявок та підвищити загальну швидкість обслуговування клієнтів.

Традиційні паперові методи ведення обліку або використання розрізнених електронних таблиць уже не здатні забезпечити належний рівень конкурентоспроможності в умовах сучасного ринку [13]. Вони призводять до виникнення так званого «людського фактора», дублювання даних, втрати важливої інформації про клієнтів, затримок в обслуговуванні та, як наслідок, зниження лояльності споживачів і фінансових збитків підприємства [14].

Дослідження операційної ефективності підприємств свідчать, що відсутність автоматизованого контролю за клієнтськими даними та затримки в обробці лідів спричиняють втрату до 20% річного доходу компанії через відтік покупців до конкурентів [1]. В умовах високої щільності інформаційного ринку, де швидкість комунікації виступає основним індикатором надійності партнера, ручне координування дій персоналу стає деструктивним чинником. Виникають хронічні затримки на етапі узгодження

технічних параметрів замовлень між відділом маркетингу та безпосередніми технічними виконавцями. Більше того, відсутність автоматизованого контролю фінансових дедлайнів призводить до касових розривів, коли послуги надаються вчасно, а виставлення рахунків чи контроль дебіторської заборгованості затримується через банальну неуважність або перевантаженість менеджерів. Це підкреслює потребу у впровадженні інструментів інтелектуального менеджменту, які здатні повністю ізолювати рутинні операції від впливу суб'єктивних помилок персоналу та забезпечити прозорий аудит кожної транзакції.

Саме тому автоматизація процесів взаємодії з клієнтами (CRM — Customer Relationship Management) та оперативного обліку наданих послуг є одним із найважливіших стратегічних завдань для сучасного бізнесу [1]. Впровадження спеціалізованих інформаційних систем дозволяє консолідувати всі дані про клієнтську базу в єдиному інформаційному просторі, забезпечити прозорість бізнес-процесів, автоматизувати рутинні операції менеджерів, оптимізувати контроль за виконанням замовлень та суттєво підвищити швидкість прийняття управлінських рішень на основі аналітичних звітів.

Актуальність теми переддипломної практики зумовлена необхідністю подолання технологічного розриву в управлінні клієнтськими потоками та оптимізації облікових операцій шляхом проектування, розробки та впровадження комплексної інформаційної системи автоматизації обліку клієнтів та надання послуг, яка б відповідала специфіці діяльності сучасних компаній, забезпечувала надійне зберігання інформації та мала гнучку архітектуру для подальшого масштабування.

Об'єктом дослідження є процеси управління взаємовідносинами з клієнтами, реєстрації, обробки, обліку та контролю надання послуг у межах діяльності підприємства.

Предметом дослідження є методи, архітектурні рішення, програмні засоби та інструменти проектування і розробки інформаційної системи, що забезпечує автоматизацію зазначених процесів обліку та обслуговування.

Метою переддипломної практики є систематизація, розширення та поглиблення теоретичних знань, отриманих за час навчання, а також набуття практичних навичок у сфері системного аналізу, проектування баз даних та безпосередньої програмної реалізації інформаційної системи автоматизації обліку клієнтів та надання послуг.

Для досягнення поставленої мети в ході практики необхідно вирішити такі основні завдання:

1. Провести детальний аналіз предметної області, дослідити існуючі бізнес-процеси підприємства та виявити ключові недоліки поточної схеми обліку.
2. Сформулювати технічні та функціональні вимоги до майбутньої інформаційної системи, визначити ролі користувачів та рівні доступу до даних.
3. Здійснити інфологічне та даталогічне проектування бази даних, визначити сутності, їхні атрибути та зв'язки між ними, провести нормалізацію таблиць.
4. Розробити архітектуру інформаційної системи, обрати оптимальний стек технологій (мови програмування, СУБД, фреймворки).
5. Реалізувати програмні модулі системи: модуль авторизації, ведення картки клієнта, обліку та каталогізації послуг, формування звітності.
6. Провести тестування розробленого програмного забезпечення на предмет помилок та відповідності технічному завданню.

Методи дослідження. Для вирішення поставлених завдань у роботі застосовано комплекс сучасних методів наукового пізнання та інженерного аналізу: методи системного аналізу — для дослідження організаційної структури медіа-групи; методи об'єктно-орієнтованого аналізу та графічного

модельовання UML — для формалізації технічних вимог та прецедентів використання системи; принципи реляційної алгебри та теорії нормалізації — для інфологічного й даталогічного проєктування бази даних; метод триланкової архітектури та принципи об'єктно-орієнтованого програмування — для кодування серверної і клієнтської частин додатка; методи модульного, інтеграційного та функціонального тестування — для оцінки надійності й стабільності розробленого програмного забезпечення.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ФОРМУВАННЯ ВИМОГ ДО СИСТЕМИ

1.1 Опис бізнес-процесів та обґрунтування необхідності автоматизації

Для створення ефективної інформаційної системи автоматизації обліку клієнтів та надання послуг першочерговим завданням є детальне дослідження та формалізація бізнес-процесів, що відбуваються всередині підприємства. Основним рушійним елементом у структурі організації є взаємодія між клієнтом (замовником послуг) та менеджером (оператором або адміністратором системи), який координує виконання замовлень та веде первинний облік. У межах дослідження предметної області було виділено три ключові блоки бізнес-процесів:

- Реєстрація та ведення обліку клієнтів. Цей процес охоплює збір первинної інформації про контрагента (прізвище, ім'я, по батькові, контактний номер телефону, електронна адреса, реквізити компанії у випадку роботи з юридичними особами), внесення цих даних до бази, а також підтримку актуальності контактів та відстеження історії взаємодії.

З позиції системного підходу, життєвий цикл взаємодії з контрагентом починається з моменту первинного референтного звернення, що вимагає негайної ідентифікації типу клієнта (фізична чи юридична особа) [13], оскільки це визначає подальший правовий та фінансовий формат супроводження угоди. У розрізі ТОВ «Звягель Медіа Груп» цей етап додатково ускладнюється необхідністю диференціації клієнтів за категоріями (наприклад: постійні рекламні агентства, локальні приватні рекламодавці, державні установи), що безпосередньо впливає на умови формування договірної документації. Крім того, підтримка актуальності контактів у неавтоматизованому середовищі призводить до децентралізації даних, коли зміни в реквізитах або номерах телефонів фіксуються лише одним

співробітником у його персональних нотатках, залишаючись недоступними для інших підрозділів компанії, що паралізує крос-функціональну взаємодію.

- Управління каталогом послуг та ціноутворенням. Процес включає формування переліку послуг, які здатне надати підприємство, визначення їхньої базової вартості, тривалості виконання, специфікацій та можливих знижок для лояльних клієнтів.

Динаміка сучасного медіа-ринку вимагає від підприємства високої гнучкості прейскуранту. Формування каталогу послуг повинно враховувати велику кількість змінних параметрів: коефіцієнти сезонності, терміновість виготовлення макетів, пріоритетність розміщення банерів на цифрових платформах або фіксовані шпальти в друкованій пресі. Ручний розрахунок вартості таких комплексних пакетів не лише уповільнює процес обслуговування, а й створює передумови для упущення комерційної вигоди через неправильне застосування знижок чи калькуляцію ПДВ. Формалізація цього процесу вимагає створення ієрархічної структури послуг із чітко визначеними межами лімітів часу на виконання, що дозволить системі автоматично верифікувати технічну спроможність виконання замовлення у задані клієнтом терміни.

- Облік та контроль процесу надання послуг. Охоплює етапи від моменту фіксації звернення клієнта, створення заявки (замовлення), призначення відповідального виконавця, відстеження поточного статусу виконання роботи (наприклад: «Очікує обробки», «В процесі», «Виконано», «Скасовано») до фінальної фіксації оплати та закриття угоди.

Управління стадіями виконання замовлення є найбільш критичною ланкою в операційному ланцюжку медіа-групи. Кожен статус електронної заявки повинен супроводжуватися логічною валідацією прав доступу користувачів та автоматичною зміною системних атрибутів. Без залучення засобів автоматизації неможливо забезпечити об'єктивний контроль за дотриманням нормативних термінів (KPI) виконавцями. Створюється

інформаційний вакуум, за якого менеджер змушений особисто дізнаватися у дизайнерів або технічних редакторів поточний стан готовності макета рекламного оголошення, що породжує хаос під час масових звернень клієнтів. Своєчасний перехід замовлення з одного статусу в інший є основою для коректного списання матеріальних ресурсів та фіксації платіжного балансу контрагента.

При традиційному (неавтоматизованому) підході кожен із цих етапів супроводжується значними часовими витратами. Менеджери вимушені дублювати інформацію в різних журналах або локальних файлах Excel, що суттєво підвищує ймовірність механічних помилок. Наприклад, неправильно введений номер телефону унеможливорює швидкий зв'язок із замовником, а відсутність єдиної бази замовлень призводить до того, що керівництво не може оперативно оцінити завантаженість персоналу та реальний фінансовий стан компанії за поточний період. Таким чином, розробка та впровадження спеціалізованого програмного забезпечення є критично необхідним кроком для централізації даних та підвищення загальної продуктивності праці.

Варто також підкреслити, що автоматизація облікових процесів дозволяє мінімізувати фінансові ризики, пов'язані з несвоєчасним виставленням рахунків або втратою інформації про надані послуги. У процесі щоденної діяльності підприємства виникає необхідність ведення детальної історії взаємодії з кожним клієнтом. Це стосується не лише фіксації його анкетних даних, а й аналізу його індивідуальних уподобань, частоти звернень та обсягів спожитих послуг. Без використання автоматизованої системи накопичення такої інформації стає хаотичним, що унеможливорює впровадження ефективних програм лояльності чи надання персональних знижок.

Крім того, важливим аспектом є оптимізація внутрішньої комунікації між різними підрозділами або окремими співробітниками компанії. Коли інформація про нове замовлення миттєво з'являється в загальній базі,

відповідальний виконавець може одразу приступити до роботи, не витрачаючи час на додаткові узгодження по телефону чи в месенджерах.

У контексті крос-модульної взаємодії це мінімізує так звані «інформаційні шуми» та виключає ймовірність спотворення технічного завдання на етапах його передачі. Структуризація внутрішніх каналів зв'язку через єдиний серверний простір дозволяє формалізувати вимоги до медіа-контенту, що розробляється. Дизайнерський та технічний відділи отримують не просто повідомлення про наявність замовлення, а вичерпну специфікацію: точні геометричні розміри рекламного банера, колірну гаму, текстовий масив оголошення та дедлайн для фінальної верстки. Це унеможливорює ситуації, коли через неузгодженість дій відділів макет доводиться переробляти кілька разів, що раніше призводило до зриву графіків виходу друкованих шпальт або оновлення цифрових ресурсів.

Керівник, у свою чергу, отримує інструмент для моніторингу ефективності роботи кожного менеджера в режимі реального часу: стає чітко видно, скільки замовлень було оброблено за день, які послуги мають найбільший попит, а на яких етапах виникають затримки. Можливість ретроспективного аналізу накопичених статистичних даних відкриває для менеджменту підприємства перспективи для стратегічного планування маркетингової активності. На основі генерації автоматичних звітів з'являється інструментарій для виявлення сезонних коливань попиту на конкретні види рекламних площ, оцінки рентабельності окремих категорій послуг та визначення реального фінансового внеску кожного співробітника в загальний дохід компанії. Це дозволяє оптимізувати штатний розклад у періоди пікових навантажень (наприклад, під час передоплачених передноворічних чи виборчих рекламних кампаній) та розробити об'єктивну систему мотивації персоналу, базуючись на реальних показниках ефективності обробки замовлень, а не на суб'єктивних оцінках. Таким чином, інформаційна система виступає не просто як цифровий журнал для записів, а

як повноцінне операційне середовище, що трансформує підходи до управління якістю сервісу та забезпечує масштабованість бізнесу в умовах жорсткої ринкової конкуренції.

1.2 Порівняльний аналіз існуючих аналогів та обґрунтування власної розробки

Для обґрунтування доцільності проектування та програмної реалізації індивідуальної інформаційної системи автоматизації обліку клієнтів та надання послуг у ТОВ «Звягель Медіа Груп» було введено детальний аналіз існуючих на ринку програмних продуктів класу CRM (Customer Relationship Management). Основними критеріями порівняння виступили: функціональна повнота щодо обліку специфічних медіа-послуг, гнучкість налаштування архітектури, вартість впровадження та володіння ліцензіями, рівень безпеки збереження персональних даних контрагентів, а також вимоги до апаратного забезпечення робочих станцій персоналу. Серед сучасних систем управління взаємовідносинами з клієнтами найбільше поширення в комерційному секторі отримали такі платформи, як AmoCRM, Bitrix24 та Salesforce. Розглянемо їхні технічні та експлуатаційні характеристики в контексті інтеграції в робоче середовище досліджуваного підприємства.

Для проектування моделей бізнес-процесів та автоматизації початкових етапів продажів у CRM-системах застосовують об'єктно-орієнтований аналіз та уніфіковану мову моделювання UML, що дозволяє гнучко інтегрувати комунікаційні канали [2].

Проте, аналіз архітектури даного продукту виявив суттєві обмеження:

- Система має слабкий вбудований інструментарій для ведення динамічного ієрархічного каталогу послуг зі складними алгоритмами ціноутворення (наприклад, розрахунок вартості друкованої площі оголошення залежно від рубрики та обсягу);

За умов відсутності нативного механізму для створення багатовимірних реляційних зв'язків між сутностями каталогу, спроба адаптувати AmoCRM під потреби ТОВ «Звягель Медіа Груп» змушує розробників створювати перевантажені текстові додаткові поля. Це призводить до повної втрати контролю над типами даних на рівні вводу, унеможлиблює автоматичну валідацію числових значень вартості та геометричних параметрів реклами, а також критично підвищує ризик виникнення системних помилок під час масової генерації фінансових документів. Будь-яка спроба реалізувати складну логіку ціноутворення через сторонні скрипти призводить до технологічної дестабілізації платформи та вимагає постійних витрат на супроводження зовнішнього коду.

- Відсутня можливість гнучкого проектування реляційних зв'язків між сутностями «замовлення» та «деталі замовлення» на рівні бази даних без придбання дорогих сторонніх віджетів або використання API;
- Платформа функціонує виключно за хмарною моделлю SaaS (Software as a Service), що створює ризики залежності від стабільності зовнішнього інтернет-каналу та сторонніх серверів.

Комплексне рішення Bitrix24 поєднує інструменти корпоративного управління та взаємодії з клієнтами, надаючи можливість локального розгортання (On-Premise), але створює надлишкове навантаження на систему [3]., яка дозволяє розгортати базу даних на локальних потужностях підприємства. Недоліками впровадження Bitrix24 для ТОВ «Звягель Медіа Груп» є:

- Надлишкова функціональність, яка перевантажує інтерфейс користувача (менеджера з прийому замовлень), уповільнює швидкість адаптації персоналу та вимагає значних обсягів оперативної пам'яті на робочих місцях;

Велика кількість невикористовуваних модулів (таких як складний облік матеріальних залишків, внутрішня соціономічна мережа компанії, бізнес-

процеси кадрового діловодства та табелювання) створює критичне когнітивне навантаження на операторів рекламного відділу. Замість швидкої фіксації параметрів замовлення клієнта, менеджер змушений здійснювати рулінг через багаторівневі та надлишкові інтерфейсні форми. Крім того, клієнтські скрипти Bitrix24 використовують значні обсяги апаратних ресурсів робочих станцій, що в локальній мережі регіонального підприємства викликає мікрозависання браузерів та безпосередньо знижує пропускну здатність робочого місця під час пікових навантажень прийому оголошень.

- Висока вартість комерційних ліцензій, яка є економічно необґрунтованою для підприємств малого та середнього бізнесу регіонального рівня;
- Складність архітектурної модифікації під специфіку обліку замовлень на публікацію періодичних видань та цифрової реклами.

Платформа **Salesforce** є світовим лідером серед CRM-систем завдяки необмеженим можливостям масштабування, підтримці об'єктно-орієнтованого моделювання великих масивів даних та впровадженню інтелектуальних аналітичних модулів. Водночас її використання в межах даного проекту є абсолютно недоцільним через надзвичайно високу вартість підписки для кожного окремого користувача системи, необхідність тривалого та вартісного залучення сертифікованих розробників для кастомізації системи під вимоги вітчизняного обліку, а також орієнтацію на великі транснаціональні корпорації, що робить архітектуру платформи критично надлишковою для локальної інформаційної системи обліку.

Для наочного порівняння та систематизації результатів аналізу аналогів було сформовано зведену техніко-експлуатаційну таблицю.

Таблиця 1.1. Порівняльний аналіз існуючих CRM-систем та проектованої системи

Критерії порівняння	AmoCRM	Bitrix24	Salesforce	Проектована система

Тип архітектурного розгортання	Хмарна (SaaS)	Хмарна/Локальна	Хмарна (SaaS)	Триланкова локальна (On-Premise)
Облік специфічних послуг медіа	Низький	Середній	Висока (надлишковий)	Повний (абсолютна відповідність)
Складність інтерфейсу оператора	Низька	Висока (перезавантажений)	Дуже висока	Низька (мінімалістичний дизайн)
Швидкодія пошуку в базі даних	Залежить від мережі	Середня	Висока	Максимальна (B-Tree індекси)
Вартість ліцензій та впровадження	Абонентська плата	Висока вартість коробки	Критично висока	Безкоштовно (власна розробка)
Захист персональних даних	На стороні провайдера	На локальному сервері	На стороні провайдера	Повний контроль (HTTPS + шифрування)

Таким чином, проведений аналіз технічних рішень доводить, що жоден із існуючих комерційних аналогів не здатний забезпечити точне, швидке та економічно вигідне вирішення завдань автоматизації для ТОВ «Звягель Медіа Груп» без суттєвих фінансових витрат та архітектурних переробок. Розробка власної практично-орієнтованої інформаційної системи на базі сучасного та безкоштовного стеку технологій (Java, Spring Boot, MySQL) дозволяє створити легку архітектуру, що повністю ліквідує «вузькі місця» в обліку замовлень рекламного відділу компанії, унеможливить механічні помилки та дублювання даних за рахунок жорстких обмежень цілісності реляційної бази даних, а також забезпечить повну незалежність підприємства від ліцензійної політики сторонніх вендорів.

Окрему увагу при формуванні технологічних вимог було приділено параметрам експлуатаційної надійності, відмовостійкості та інформаційної безпеки системи, оскільки вона оперуватиме конфіденційними персональними даними клієнтів ТОВ «Звягель Медіа Груп» та фінансовими показниками замовлень. Програмний комплекс повинен гарантувати стабільне функціонування в режимі 24/7 із коефіцієнтом доступності не менше 99.5%. Середній час відгуку системи на стандартні пошукові HTTP-запити до бази даних (наприклад, фільтрація замовлень за статусом або пошук картки контрагента за номером телефону) не повинен перевищувати 1.5 секунди в умовах одночасної роботи всіх операторів локальної мережі.

У розрізі безпеки даних обов'язковою є реалізація механізму захисту від типових веб-вразливостей. Система має бути захищена від SQL-ін'єкцій (SQL Injections) шляхом обов'язкового використання параметризованих запитів та технології Hibernate/JPA на серверному рівні, а також від міжсайтового скриптингу (XSS) через примусову валідацію та екранування символів у вхідних формах веб-інтерфейсу, що відповідає вимогам стандарту безпеки OWASP Top Ten [10]. Всі паролі користувачів перед збереженням у таблицю users повинні проходити процедуру незворотного шифрування за допомогою стійкого алгоритму криптографічного хешування. Також обов'язковою є конфігурація автоматичного завершення поточної сесії користувача (Session Timeout) у разі його бездіяльності протягом 30 хвилин, що мінімізує ризик несанкціонованого доступу до системи сторонніх осіб у разі залишення робочого місця менеджером.

Окремої уваги заслуговує ще одна широко розповсюджена платформа управління взаємовідносинами з клієнтами — Zoho CRM. Дана система позиціонується розробником як доступне хмарне рішення для малого та середнього бізнесу і має значно нижчу вартість підписки порівняно з Salesforce. Проте детальний аналіз можливостей Zoho CRM у контексті

специфіки діяльності ТОВ «Звягель Медіа Груп» виявив суттєві обмеження, аналогічні до описаних вище платформ.

По-перше, система не передбачає нативної підтримки ієрархічного каталогу медіа-послуг із багаторівневим ціноутворенням. Налаштування такого каталогу вимагає придбання додаткових модулів або ручного конфігурування нестандартних полів, що суттєво ускладнює впровадження та збільшує вартість володіння системою. По-друге, всі дані клієнтів зберігаються виключно на хмарних серверах постачальника, що суперечить вимогам до забезпечення конфіденційності персональних даних контрагентів і створює залежність від стабільності зовнішнього інтернет-каналу. По-третє, локалізація системи для україномовного середовища є неповною, що ускладнює роботу персоналу та підвищує ризик операційних помилок під час введення даних.

Таким чином, після всебічного аналізу чотирьох провідних CRM-платформ (AmoCRM, Bitrix24, Salesforce та Zoho CRM) можна зробити однозначний висновок: жодна з них не задовольняє комплекс вимог, сформульованих для інформаційної системи ТОВ «Звягель Медіа Груп», без значних фінансових вкладень та суттєвого доопрацювання архітектури.

Ключовими недоліками всіх розглянутих аналогів є: відсутність гнучкого модуля обліку специфічних медіа-послуг із динамічним ціноутворенням; хмарна модель розгортання, що не відповідає вимогам локальної інфраструктури підприємства; надлишковість функціоналу, що перевантажує інтерфейс операторів; висока вартість ліцензій, економічно необґрунтована для регіонального підприємства малого та середнього бізнесу.

Саме ці чинники обумовлюють доцільність проектування та розробки власної спеціалізованої інформаційної системи на базі безкоштовного стеку технологій із відкритим вихідним кодом. Такий підхід дозволяє досягти повної відповідності функціональних можливостей системи специфіці операційної діяльності підприємства, забезпечити розміщення бази даних на

локальному сервері компанії, гарантувати незалежність від ліцензійної політики сторонніх постачальників програмного забезпечення, а також закласти архітектурний фундамент для подальшого масштабування системи відповідно до зростаючих потреб бізнесу.

1.3 Функціональні та технологічні вимоги до інформаційної системи

На основі проведеного аналізу предметної області було сформовано перелік чітких функціональних вимог до проектованої інформаційної системи. Система має бути розділена на логічні модулі та забезпечувати виконання таких операцій:

- Модуль автентифікації та розмежування прав доступу: безпечний вхід у систему за логіном і паролем, розподіл користувачів на ролі (наприклад, «Адміністратор» з повними правами на редагування та видалення даних, та «Менеджер» з правами на внесення клієнтів та оформлення послуг).
 - Модуль керування клієнтською базою: створення, перегляд, редагування та деактивація карток клієнтів; швидкий пошук за ПІБ, номером телефону або унікальним ідентифікатором (ID).
 - Модуль обліку послуг: ведення динамічного довідника послуг, можливість оперативно змінювати ціни, додавати нові позиції або приховувати ті, що тимчасово не надаються.
 - Модуль замовлень: формування електронного документа замовлення із прив'язкою до конкретного клієнта, вибір однієї або кількох послуг із каталогу, автоматичний розрахунок загальної вартості, фіксація дати створення та планованої дати виконання.
- З технічної точки зору до системи висуваються вимоги щодо надійності зберігання даних, швидкодії при виконанні пошукових запитів, інтуїтивно зрозумілого інтерфейсу користувача, що відповідає принципам юзабіліті програмного забезпечення [24], а також мобільності — архітектура має

дозволяти розгортання системи в локальній мережі або на віддаленому сервері для забезпечення доступу з різних робочих місць.

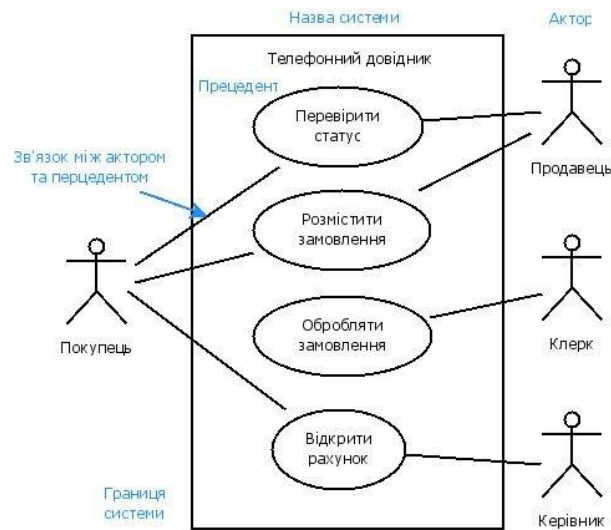


Рисунок 1.1 — Діаграма прецедентів (Use Case) інформаційної системи обліку клієнтів та надання послуг

Для детального розуміння архітектурних меж проектованої системи необхідно деталізувати ролі акторів, представлених на діаграмі прецедентів (Use Case). Кожен актор має строго визначені межі відповідальності та унікальний набір прецедентів взаємодії з ядром системи:

1. Актор "Покупець" (Клієнт): Виступає зовнішнім ініціатором процесу. Його взаємодія з системою є опосередкованою або ініціюється через запити. Головний прецедент для нього — "Перевірити статус", що дозволяє отримати актуальну інформацію про стан виконання його медіа-продукту.
2. Актор "Продавець" (Менеджер з реклами): Ключовий оператор системи першої лінії. Він взаємодіє з прецедентами "Розмістити замовлення" та "Перевірити статус". Його обов'язком є первинна валідація даних, робота з прейскурантом та фіксація фінансових параметрів угоди.
3. Актор "Клерк" (Технічний виконавець / Дизайнер / Верстальник): Відповідає за прецедент "Обробити замовлення". Цей актор не має доступу до фінансових модулів чи редагування карток клієнтів, проте

він змінює технологічні статуси виконання макетів, забезпечуючи наскрізний моніторинг KPI.

4. Актор "Керівник" (Адміністратор системи / Директор): Наділений повним спектром прав доступу. Його ключовий прецедент — "Відкрити рахунок" та "Генерація аналітичної звітності". Він аналізує агреговані дані, керує обліковими записами інших користувачів та має право модифікувати довідник послуг.

Для деталізації логіки роботи системи та точного проєктування інтерфейсу користувача розроблено специфікації основних прецедентів використання, що відображені на діаграмі (див. рис. 1.1). У таблицях 1.2 та 1.3 наведено детальний опис ключових сценаріїв взаємодії акторів із системою.

Таблиця 1.2. Специфікація прецеденту «Розмістити замовлення»

Елемент специфікації	Опис та характеристики
Актор (Користувач)	Продавець/Менеджер з реклами
Передумови	Менеджер успішно пройшов автентифікацію в системі
Основний сценарій	1. Менеджер відкриває вкладку «Створити замовлення». 2. Система відображає форму та підтягує актуальний довідник послуг. 3. Менеджер вибирає клієнта з бази даних та додає необхідні послуги. 4. Система автоматично розраховує фінальну суму (total_price). 5. Менеджер натискає «Зберегти», система записує дані в БД та присвоює статус «Нове».

Альтернативний сценарій	Якщо обов'язкові поля не заповнені, система підсвічує їх червоним, блокує відправку запиту і виводить інформаційне повідомлення.
-------------------------	--

Таблиця 1.3. Специфікація прецеденту «Перевірити статус»

Елемент специфікації	Опис та характеристики
Актор (Користувач)	Покупець (Клієнт)/Продавець
Передумови	Замовлення вже зареєстроване в базі даних
Основний сценарій	1. Користувач вводить унікальний ID замовлення або номер телефону клієнта. 2. Система виконує SQL-запит до таблиці Orders із використанням індексу статусу. 3. На екран виводиться поточний етап обробки: «Нове», «В процесі» або «Виконано».

Таблиця 1.4. Специфікація прецеденту «Обробити замовлення»

Елемент специфікації	Опис та характеристики
Актор (Користувач)	Клерк (Технічний спеціаліст / Дизайнер)
Передумови	Замовлення перебуває у статусі "Нове" та призначене на відповідний відділ
Основний сценарій	1. Клерк авторизується та відкриває робочий стіл із чергою нових завдань. 2. Обирає замовлення, система змінює статус на "В процесі". 3. Після завершення технічної верстки макета клерк завантажує звіт про виконання і натискає "Готово". 4. Система змінює статус на

	"Виконано" і надсилає сповіщення менеджеру.
Альтернативний сценарій	Якщо клерк виявляє помилку в ТЗ від менеджера, він обирає опцію "Повернути на доопрацювання", статус змінюється на "Призупинено", система вимагає введення текстового коментаря з описом проблеми.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА ОБҐРУНТУВАННЯ АРХІТЕКТУРИ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Обґрунтування вибору стеку технологій та архітектурних рішень

Для програмної реалізації інформаційної системи автоматизації обліку клієнтів та надання послуг було обрано триланкову архітектуру (Three-tier architecture), яка включає в себе рівень представлення (інтерфейс користувача), рівень бізнес-логіки (серверна частина) та рівень керування даними (база даних). Такий підхід забезпечує високу гнучкість системи, незалежність окремих модулів, простоту масштабування та підвищену безпеку [4], оскільки користувач не має прямого доступу до таблиць бази даних, а взаємодіє з ними виключно через проміжний серверний шар.

Взаємодія між рівнем представлення та сервером додатків реалізована за допомогою архітектурного стилю REST API, концепцію якого було формалізовано у дисертаційній роботі Р. Філдінга [5].

Клієнтська частина надсилає HTTP-запити (GET, POST, PUT, DELETE), а сервер повертає дані у форматі JSON. Це дозволяє повністю приховати внутрішню структуру бази даних від зовнішнього середовища. Для захисту передачі даних та авторизації користувачів у системі передбачено інтеграцію модулів Spring Security, що унеможливорює несанкціонований доступ до фінансової інформації [18] ТОВ «Звягель Медіа Груп» з боку сторонніх осіб.

У контексті ТОВ «Звягель Медіа Груп» триланкова архітектура дозволяє повністю ізолювати клієнтські робочі місця менеджерів від безпосереднього виконання важких SQL-запитів до бази даних, переносячи все обчислювальне навантаження на виділений сервер додатків. Це гарантує стабільну роботу системи навіть на технічно застарілих комп'ютерах у локальній мережі підприємства. Крім того, подібний розподіл відповідальності компонентів суттєво спрощує модернізацію системи: за необхідності зміни дизайну або логіки інтерфейсу користувача (Presentation Layer), серверна бізнес-логіка (Application Layer) та структура таблиць у

СКБД (Data Layer) залишаються незмінними, що мінімізує ризики внесення нових програмних помилок у процесі супроводження софту.

Розподіл системних компонентів на три автономні рівні (представлення, бізнес-логіки та збереження даних) є базовим патерном під час проектування сучасних корпоративних додатків, що мінімізує жорстку зв'язаність модулів [4].

Рівень бізнес-логіки (Application/Logic Layer), винесений на окремий сервер додатків, бере на себе всю обчислювальну потужність, виконуючи роль єдиної точки автентифікації, маршрутизації та валідації запитів. Це дозволяє суттєво розвантажити клієнтські машини менеджерів медіа-групи, оскільки браузер лише рендерить готовий JSON-контент. Нарешті, рівень керування даними (Data Tier) стає повністю ізольованим від зовнішнього середовища: база даних MySQL приймає SQL-з'єднання виключно від довіреної IP-адреси сервера бізнес-логіки. Таке архітектурне рішення ліквідує загрозу прямих SQL-ін'єкцій з боку злоумисників та спрощує процедуру резервного копіювання (backup) та реплікації даних без зупинки операційного процесу компанії.

При виборі технологій для реалізації проекту враховувалися вимоги щодо швидкодії, надійності, кросплатформеності та наявності розвиненої екосистеми інструментів розробки:

- Система керування базами даних (СУБД): Як реляційне сховище даних було обрано СУБД MySQL (або PostgreSQL). Вона є безкоштовною, має відкритий вихідний код, забезпечує високу швидкість обробки SQL-запитів, підтримує механізми транзакційності (ACID) та гарантує цілісність даних завдяки використанню зовнішніх ключів і каскадних операцій.
- Серверна частина (Backend): Для реалізації бізнес-логіки системи, обробки HTTP-запитів та взаємодії з базою даних доцільно використовувати об'єктно-орієнтований підхід. Основними

інструментами обрано мову програмування Java (з фреймворком Spring Boot) або C# (з платформою .NET Core / ASP.NET). Ці технології володіють потужними засобами для розмежування прав доступу, побудови REST API та автоматичного мапінгу об'єктів на таблиці БД (ORM Hibernate / Entity Framework).

- Клієнтська частина (Frontend): Інтерфейс користувача реалізується як веб-додаток або кросплатформений десктопний застосунок. Для веб-версії використовується стек HTML5, CSS3 для верстки адаптивних форм та мова JavaScript (разом із фреймворком Vue.js або React) для забезпечення динамічної взаємодії з сервером без повного перезавантаження сторінок (AJAX-запити).

Обґрунтовуючи вибір архітектури, окрему увагу було приділено організації взаємодії між клієнтською та серверною частинами за допомогою безпечного протоколу передачі даних HTTPS. Оскільки система оперує персональними даними клієнтів (номери телефонів, електронні адреси, комерційні замовлення), захист інформації від несанкціонованого доступу або перехоплення є першочерговим технологічним завданням. Використання триланкової моделі дозволяє винести всю логіку перевірки прав доступу на рівень сервера (Backend), що повністю унеможливорює підміну даних на стороні браузера користувача.

Додатковою перевагою обраного технологічного стеку є можливість легкого інтегрування системи з іншими зовнішніми сервісами у майбутньому за допомогою REST API. Наприклад, архітектура дозволяє без зміни корінної логіки підключити модулі SMS-розсилок для автоматичного сповіщення клієнтів про готовність замовлення, або інтегрувати платіжні шлюзи для онлайн-оплати послуг. Крім того, використання шаблону проектування MVC (Model-View-Controller) на серверному рівні забезпечує чітке розділення коду на три незалежні компоненти: об'єкти даних, логіку обробки та логіку відображення [9]. Це дозволяє розробникам паралельно вносити зміни в

інтерфейс системи, не ризикуючи порушити стабільність роботи бази даних чи алгоритмів підрахунку фінансових показників, що суттєво знижує витрати на подальше технічне обслуговування та модернізацію програмного продукту.

Архітектурний каркас розроблюваного програмного забезпечення базується на концептуальному патерні проектування MVC (Model-View-Controller), який забезпечує чітке розділення відповідальності між даними системи, інтерфейсом користувача та керуючою логікою. Рівень моделі (Model) представлений класами-сутностями Java, які повністю відображають структуру таблиць бази даних та інкапсулюють їхній стан. Рівень представлення (View) винесений на сторону клієнтського браузера, що дозволяє розвантажити обчислювальні потужності сервера. Рівень контролера (Controller) реалізований за допомогою REST-контролерів Spring Boot, які приймають веб-запити, викликають відповідні сервіси бізнес-логіки та повертають результат.

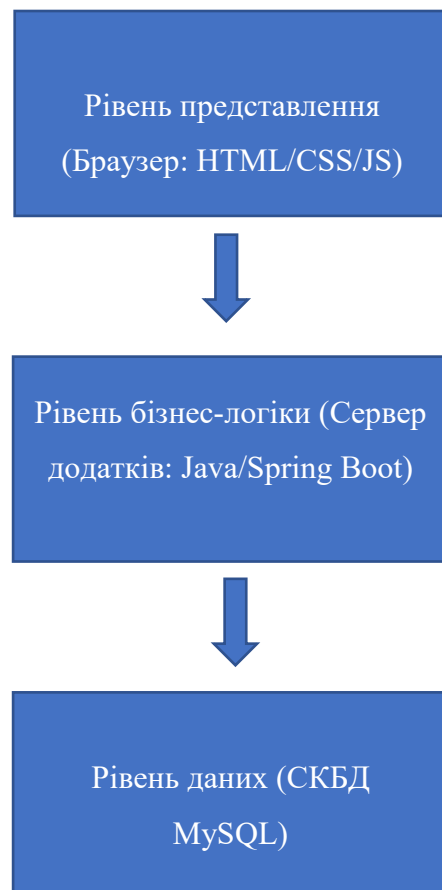


Рис. 2.1 Схема триланкової архітектури інформаційної системи підприємства

Вибір фреймворку Spring Boot замість класичних рішень Java EE (Jakarta EE) зумовлений його філософією «угоди понад конфігурацією» (Convention over Configuration) [3]. Це дозволило уникнути написання величезної кількості шаблонного XML-коду для налаштування компонентів програми. Вбудований сервісний контейнер сервлетів Apache Tomcat дає можливість запускати додаток як самодостатній файл (executable JAR) без необхідності попереднього встановлення та складного адміністрування важковагових зовнішніх серверів додатків [25] (таких як WildFly чи GlassFish), що суттєво спрощує технічне супроводження системи в IT-інфраструктурі підприємства.

2.2 Інфологічне та даталогічне проектування бази даних

Центральним елементом проектованої інформаційної системи є реляційна база даних, яка повинна зберігати всю інформацію про клієнтів, послуги, замовлення та користувачів системи у нормалізованому вигляді (мінімум у третій нормальній формі — 3NF) для уникнення аномалій вставки, оновлення та видалення даних.

Процес проектування логічної схеми бази даних базувався на суворій декомпозиції сутностей для досягнення стану абсолютної відповідності вимогам третьої нормальної форми, що є загальноприйнятим стандартом проектування реляційних сховищ даних [15].

Це означає, що кожна таблиця системи містить лише атомарні атрибути, кожен неключовий атрибут повністю залежить від первинного ключа (Primary Key) і при цьому виключені будь-які транзитивні залежності між неключовими полями. Завдяки цьому вдалося усунути надлишковість даних на рівні сховища, що суттєво економить дисковий простір сервера та

мінімізує час виконання операцій запису й оновлення. На рівні фізичного проектування в СКБД MySQL особлива увага була приділена налаштуванню механізмів підтримання посиловальної цілісності даних (Referential Integrity). Зв'язки між сутностями типу «один-до-багатьох» (1:M) реалізовані через впровадження зовнішніх ключів (Foreign Keys). Для оптимізації роботи з пов'язаними записами було застосовано каскадні обмеження. Зокрема, для зв'язку між таблицею клієнтів та таблицею замовлень встановлено правило ON DELETE RESTRICT, яке блокує видалення картки клієнта, якщо в системі існує хоча б одне зареєстроване на нього замовлення. Це захищає бізнес-логіку системи від появи «сирітських» записів та фінансової плутанини в історичних звітах компанії. На етапі інфологічного проектування (визначення ER-моделі сутностей та зв'язків) було виділено такі основні інформаційні об'єкти:

- Користувачі (Users): зберігає дані для авторизації співробітників (менеджерів та адміністраторів). Атрибути: ID_користувача (Primary Key), Логін, Хеш_пароллю, Роль (Адміністратор/Менеджер), ПІБ_співробітника.

Для забезпечення високого рівня інформаційної безпеки та розмежування прав доступу на рівні автентифікації користувачів (Role-Based Access Control - RBAC), у таблиці users поле для збереження паролів проектується з урахуванням сучасних криптографічних стандартів. Паролі користувачів ніколи не зберігаються у відкритому вигляді; замість цього використовується алгоритм хешування BCrypt із генерацією унікальної випадкової солі (salt) для кожного запису, що відповідає сучасним криптографічним стандартам захисту персональних даних [11]. Поле ролі (role) реалізовано за допомогою перелічуваного типу даних (ENUM), що обмежує можливі значення системи строгими константами ('ADMIN', 'MANAGER') на рівні ядра СКБД, унеможливаючи несанкціоноване

підняття привілеїв або ін'єкцію невалідних ролей через форми реєстрації чи редагування профілів.

- Клієнти (Clients): містить картки контрагентів підприємства.
Атрибути: ID_клієнта (Primary Key), Прізвище, Ім'я, По_батькові, Телефон, Електронна_пошта, Дата_реєстрації.

Проектування полів таблиці clients враховує специфіку роботи медіа-групи як із фізичними особами (роздрібними замовниками приватних оголошень), так і з великими комерційними контрагентами чи рекламними агенціями (юридичними особами). Поле «назва компанії» (company_name) визначено з параметром логічної порожнечі (NULL), оскільки приватні особи не мають корпоративної приналежності. Навпаки, поля контактних даних, такі як номер телефону (phone) та адреса електронної пошти (email), захищені обмеженнями NOT NULL. Додатково на рівні СКБД для поля email накладено унікальний індекс (UNIQUE INDEX), що запобігає дублюванню карток клієнтів у базі даних та спрощує процедуру пошуку й ідентифікації контрагентів при повторних зверненнях.

- Послуги (Services): містить прейскурант та опис доступних послуг.
Атрибути: ID_послуги (Primary Key), Назва_послуги, Опис, Вартість, Термін_виконання.

Оскільки прейскурант ТОВ «Звягель Медіа Груп» є динамічним і залежить від ринкових факторів, поле «базова ціна» (base_price) спроектовано з використанням точного числового типу даних DECIMAL(10,2). Використання типів із плаваючою комою (таких як FLOAT або DOUBLE) у фінансових модулях є неприпустимим через накопичення похибок округлення під час математичних операцій, що може призвести до розбіжностей у бухгалтерській звітності. Поле опису послуги (description) має гнучкий тип TEXT, що дозволяє зберігати детальні технічні специфікації (наприклад, вимоги до роздільної здатності зображень для банерів, обмеження по кількості знаків для текстових публікацій, чи періодичність

виходу рекламних блоків). Додатково на рівні фізичної схеми бази даних для сутності Services впроваджено перевірочне обмеження CHECK (price >= 0), що діє як апаратний фільтр СКБД і унеможливорює випадкове або навмисне внесення від'ємних значень вартості послуг менеджерами чи адміністраторами. Атрибут duration_days, який визначає термін виконання замовлення у днях, також обмежений правилом CHECK (duration_days > 0), оскільки будь-яка медіа-послуга (від верстки оголошення до друку на шпальтах) потребує щонайменше одного робочого дня для технологічної обробки відповідними відділами компанії.

- Замовлення (Orders): фіксує факт звернення клієнта та загальні параметри угоди. Атрибути: ID_замовлення (Primary Key), ID_клієнта (Foreign Key), ID_користувача (Foreign Key, менеджер, що оформив угоду), Дата_створення, Статус_замовлення (Нове / В процесі / Виконано), Загальна_вартість.

Таблиця orders виступає головним консолідуючим вузлом для фіксації фінансово-господарських операцій у системі. Поле order_date проектується з типом даних DATETIME та значенням за замовчуванням DEFAULT CURRENT_TIMESTAMP, що забезпечує автоматичну реєстрацію точного часу створення заявки на сервері бази даних, виключаючи маніпуляції з датами з боку користувачів. Поле status використовує строгий тип VARCHAR з обмеженням довжини та накладеним некластеризованим індексом. Це архітектурне рішення є критично важливим для швидкої фільтрації та побудови оперативних звітів керівництва (наприклад, для миттєвого виведення всіх замовлень у статусі «В процесі» або «Очікує обробки»). Поле total_price відображає агреговану суму всіх доданих у замовлення послуг і оновлюється автоматично за допомогою тригерів СКБД або через сервісний шар бізнес-логіки після валідації кожної транзакції.

- Деталі замовлення (Order_Details): проміжна таблиця для реалізації зв'язку «багато-до-багатьох» між замовленнями та послугами,

оскільки одне замовлення може містити декілька різних послуг. Атрибути: ID_деталей (Primary Key), ID_замовлення (Foreign Key), ID_послуги (Foreign Key), Кількість, Ціна_на_момент_продажу.

- Зв'язки між таблицями будуються на основі ідентифікаторів: таблиця Orders пов'язана зі специфічними записами в Clients та Users відношенням «один-до-багатьох» (one-to-many). Таблиця Order_Details деталізує склад кожного замовлення, зв'язуючи його з конкретними послугами з довідника Services.

Математична модель розрахунку загальної вартості кожної позиції у замовленні на рівні сервісного шару програми описується формулою (2.1):

$$\text{Total_Price} = \sum_{i=1}^n (\text{Price}_i \times \text{Quantity}_i)$$

де Price_i - фіксована вартість одиниці конкретної медіа-послуги на момент оформлення договору (тип DECIMAL); Quantity_i - кількісний показник замовлених послуг (наприклад, кількість виходів рекламного оголошення або кількість друкованих примірників).

Необхідність винесення специфікації замовлення в окрему сутність order_details зумовлена правилами нормалізації реляційних баз даних для усунення багатозначних залежностей. Поле sale_price дублює вартість послуги на момент оформлення угоди. Це важливий інженерний крок: якщо в майбутньому адміністратор змінить базову ціну послуги в довіднику services, історичні дані про вже виконані та оплачені замовлення в таблиці order_details залишаться незмінними, що гарантує фінансову стабільність та коректність аналітики за минулі періоди. Для зовнішнього ключа order_id налаштовано каскадне видалення ON DELETE CASCADE. Це означає, що у випадку анулювання всього замовлення, всі його деталізовані рядки будуть автоматично вилучені зі сховища, запобігаючи захаращенню бази даних інформаційним сміттям.

Окрему увагу при проектуванні схеми бази даних було приділено забезпеченню цілісності даних на рівні обмежень самої СУБД (Database Constraints). Для всіх зовнішніх ключів (Foreign Keys) було налаштовано правила каскадного оновлення (ON UPDATE CASCADE) та обмеження на видалення (ON DELETE RESTRICT). Це означає, що при зміні унікального ідентифікатора клієнта всі пов'язані з ним замовлення автоматично оновляться, проте видалити картку клієнта, який уже має історію замовлень у системі, буде неможливо до тих пір, поки ці замовлення не будуть архівовані або видалені. Такий підхід повністю виключає появу в базі даних так званих «записів-сиріт», які порушують логіку формування фінансової звітності.

Окрім того, на етапі фізичного проектування було проведено індексацію ключових полів для оптимізації швидкодії пошукових запитів. Індеси були створені для стовпців Телефон у таблиці клієнтів та Статус_замовлення у таблиці замовлень. Оскільки обсяг клієнтської бази з часом буде лише зростати, прямий лінійний пошук по всій таблиці уповільнював би роботу інтерфейсу менеджерів. Використання збалансованих дерев (B-Tree індексів) дозволяє СУБД знаходити потрібний запис за фіксовану, мінімальну кількість операцій, незалежно від того, чи містить таблиця тисячу записів, чи сотні тисяч. Також для забезпечення унікальності логінів користувачів та електронних адрес клієнтів було застосовано обмеження UNIQUE, що запобігає випадковому дублюванню облікових записів під час реєстрації.

Для забезпечення фізичної реалізації спроектованої бази даних у середовищі обраної СУБД було розроблено детальні специфікації кожної сутності.

На рівні даталогічної моделі між таблицями «Послуги» (Services) та «Замовлення» (Orders) реалізовано ідентифікуючий зв'язок типу «один-до-багатьох» (1:M). Для забезпечення цілісності даних у СКБД налаштовано правило каскадного оновлення (ON UPDATE CASCADE). Це означає, що

при зміні унікального коду послуги в головній таблиці, всі пов'язані записи в архіві замовлень оновляться автоматично. Водночас, для запобігання випадкового видалення фінансової історії, встановлено обмеження на видалення (ON DELETE RESTRICT) — систему заблокує спробу стерти послугу, якщо вона вже фігурує в активних або завершених договорах клієнтів.

Нижче наведено структуру та характеристики основних таблиць бази даних автоматизованої системи.

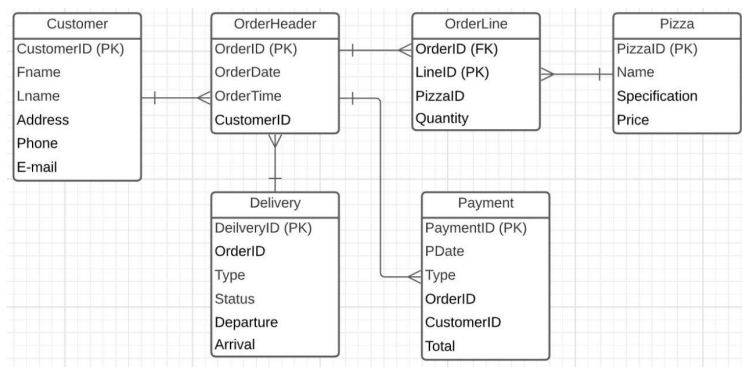


Рисунок 2.2 — Логічна схема взаємозв'язків сутностей бази даних (ER-діаграма)

Таблиця 2.1. Структура таблиці користувачів (Users)

Назва поля	Тип даних	Розмір/Опис полів	Ключ	Опис призначення поля
Id_user	INT	AUTO_INCREMENT	PK	Унікальний ідентифікатор співробітника
login	VARCHAR	50, NOT NULL, UNIQUE	-	Логін користувача для входу в систему
password_hash	VARCHAR	255, NOT NULL	-	Хеш-код пароля користувача (безпека)
role	VARCHAR	20, NOT NULL	-	Роль у системі (Адміністратор/Менеджер)
full_name	VARCHAR	100, NOT NULL	-	Прізвище, ім'я та по батькові співробітника

Таблиця 2.2. Структура таблиці клієнтів (Clients)

Назва поля	Тип даних	Розмір/Опис полів	Ключ	Опис призначення поля
id_client	INT	AUTO_INCREMENT	PK	Унікальний ідентифікатор картки клієнта
last_name	VARCHAR	50, NOT NULL	-	Прізвище клієнта або назва контрагента
first_name	VARCHAR	50, NOT NULL	-	Ім'я клієнта
middle_name	VARCHAR	50	-	По батькові клієнта (за наявності)
phone	VARCHAR	20, NOT NULL, INDEX	-	Контактний номер телефону клієнта
email	VARCHAR	100, UNIQUE	-	Адреса електронної пошти клієнта
reg_date	DATETIME	DEFAULT CURRENT_TIMESTAMP	-	Дата та час реєстрації клієнта в системі

Таблиця 2.3. Структура довідника послуг (Services)

Назва поля	Тип даних	Розмір/Опис полів	Ключ	Опис призначення поля
id-service	INT	AUTO_INCREMENT	PK	Унікальний

				ідентифікатор послуги
service_name	VARCHAR	150, NOT NULL	-	Повна назва послуги з прейскуранту
Description	TEXT	NULL	-	Детальний опис або специфікація послуги
Price	DECIMAL	10, 2, NOT NULL	-	Базова вартість послуги (грн)
duration_days	INT	NOT NULL, DEFAULT 1	-	Термін виконання послуги у днях

Таблиця 2.4. Структура таблиці замовлень (Orders)

Назва поля	Тип даних	Розмір/Опис полів	Ключ	Опис призначення поля
id_order	INT	AUTO_INCREMENT	PK	Унікальний номер електронного замовлення
client_id	INT	NOT NULL	FK	Зв'язок із таблицею Clients (id_client)
user_id	INT	NOT NULL	FK	Зв'язок із таблицею Users (id_user)
order_date	DATETIME	DEFAULT CURRENT_TIMESTAMP	-	Дата та час оформлення замовлення

status	VARCHAR	30, NOT NULL, INDEX	-	Поточний статус (Нове, В процесі, Виконано)
total_price	DECIMAL	10, 2, NOT NULL	-	Загальна підсумкова вартість замовлення

Таблиця 2.5. Структура проміжної таблиці деталей

Назва поля	Тип даних	Розмір/Опис полів	Ключ	Опис призначення поля
id_detail	INT	AUTO_INCREMENT	PK	Унікальний ідентифікатор рядка деталізації
order_id	INT	NOT NULL	FK	Зв'язок із таблицею Orders (id_order)
service_id	INT	NOT NULL	FK	Зв'язок із таблицею Services (id_service)
quantity	INT	NOT NULL, DEFAULT 1	-	Кількість одиниць цієї послуги в замовленні
sale_price	DECIMAL	10, 2, NOT NULL	-	Фіксована ціна послуги на момент

				продажу
--	--	--	--	---------

2.3 Об'єктно-орієнтоване моделювання поведінки та взаємодії компонентів системи

Для формалізації динамічних аспектів функціонування інформаційної системи автоматизації обліку, окрім статичної структури даних, необхідно детально спроектувати логіку взаємодії об'єктів у часі засобами уніфікованої мови моделювання [23].

Об'єктно-орієнтований підхід передбачає, що кожен бізнес-процес розбивається на послідовність повідомлень, які передаються між компонентами рівня представлення (Frontend), серверної логіки (Backend) та шару збереження даних (Database).

Діаграма класів (Class Diagram)

Діаграма класів відображає статичну структуру об'єктно-орієнтованої моделі серверної частини системи. Основними класами-сутностями (Entity-класами), що відповідають таблицям реляційної бази даних, є: User, Client, Service, Order та OrderDetail.

Клас **User** інкапсулює атрибути облікового запису співробітника: ідентифікатор (idUser), логін (login), хеш пароллю (passwordHash), роль у системі (role) та повне ім'я (fullName). Методи класу забезпечують отримання та встановлення значень відповідних полів.

Клас **Client** містить анкетні дані контрагента: ідентифікатор (idClient), прізвище (lastName), ім'я (firstName), по батькові (middleName), номер телефону (phone), електронну пошту (email) та дату реєстрації (regDate). З класом Order даний клас пов'язаний відношенням агрегації «один-до-багатьох» — один клієнт може мати довільну кількість замовлень.

Клас **Service** представляє позицію прейскуранту підприємства та містить атрибути: ідентифікатор (idService), назву послуги (serviceName),

опис (description), базову ціну (price) та термін виконання у днях (durationDays).

Клас **Order** є центральним об'єктом системи та реалізує зв'язок між клієнтом і переліком замовлених послуг. Його атрибути: ідентифікатор замовлення (idOrder), посилання на клієнта (client), посилання на менеджера (user), дата оформлення (orderDate), поточний статус (status) та загальна вартість (totalPrice). Клас агрегує колекцію об'єктів OrderDetail.

Клас **OrderDetail** є проміжною сутністю, що реалізує зв'язок «багато-до-багатьох» між замовленнями та послугами. Його атрибути: ідентифікатор рядка (idDetail), посилання на замовлення (order), посилання на послугу (service), кількість одиниць (quantity) та зафіксована ціна на момент продажу (salePrice).

Основним динамічним процесом у системі, що безпосередньо впливає на швидкість обслуговування в ТОВ «Звягель Медіа Груп», є оформлення нового замовлення на рекламу чи публікацію. Алгоритм взаємодії компонентів під час виконання цього процесу складається з таких кроків:

1. Користувач (менеджер) через веб-інтерфейс форми створення замовлення ініціює подію вибору клієнта та переліку послуг.
2. Клієнтська частина додатка збирає введені ідентифікатори (clientId, serviceId) та кількість одиниць послуг, формуючи об'єкт запиту типу JSON.
3. За допомогою асинхронного HTTP POST-запиту дані передаються на серверний контролер OrderService.
4. Серверний шар бізнес-логіки відкриває транзакцію, звертається до репозиторію для валідації сутностей у базі даних MySQL і, в разі успіху, викликає метод збереження.
5. Після успішного комміту транзакції СУБД повертає згенерований унікальний id_order, а сервер відправляє клієнту підтвердження зі статусом 200 OK для оновлення інтерфейсу користувача.

Окрему увагу в межах моделювання поведінки системи приділено забезпеченню стійкості архітектури до виключних ситуацій (Exception Handling). Оскільки система оперує фінансовими показниками (ціна, загальна вартість замовлення), будь-який збій на етапі запису деталей замовлення у проміжну таблицю Order_Details призведе до порушення цілісності обліку. Моделювання поведінки передбачає використання ізольованих сервісних методів із міткою транзакційності. Це гарантує, що якщо база даних заблокована або відсутній зв'язок із таблицею Services, процес створення електронного документа повністю переривається (відбувається Rollback), а менеджер отримує інформативне вікно помилки замість некоректно збереженого «пустого» замовлення. Такий підхід мінімізує вплив людського фактора та забезпечує повну відповідність розробленої архітектури бізнес-вимогам підприємства.

2.4 Проектування та опис програмного інтерфейсу REST API

Взаємодія між клієнтською частиною системи (frontend) та серверною частиною (backend) реалізована на основі архітектурного стилю REST (Representational State Transfer). Даний підхід є галузевим стандартом побудови веб-сервісів і передбачає використання стандартних HTTP-методів (GET, POST, PUT, DELETE) для виконання операцій над ресурсами системи. Усі запити та відповіді передаються у форматі JSON (JavaScript Object Notation), що забезпечує платформонезалежність та зручність інтеграції з будь-яким клієнтом [19].

Базовий URL серверного додатку у середовищі розробки: **http://localhost:8080/api**. Усі кінцеві точки API (endpoints) згруповані за функціональними модулями: управління користувачами (/users), управління клієнтами (/clients), управління послугами (/services) та управління замовленнями (/orders).

- **Модуль автентифікації**

Перед отриманням доступу до будь-якого захищеного ресурсу системи клієнт повинен пройти процедуру автентифікації. Для цього використовується механізм сесійних куки (Session Cookie), що підтримується Spring Security.

Endpoint: POST /api/auth/login Призначення: автентифікація користувача за логіном та паролем.

Тіло запиту (Request Body):

```
{
  "login": "manager01",
  "password": "securePass123"
}
```

Відповідь у разі успішної автентифікації (HTTP 200 OK):

```
{
  "status": "success",
  "role": "MANAGER",
  "fullName": "Новак Вікторія Олександрівна",
  "message": "Автентифікація успішна"
}
```

Відповідь у разі невірних облікових даних (HTTP 401 Unauthorized):

```
{
  "status": "error",
  "message": "Невірний логін або пароль"
}
```

Endpoint: POST /api/auth/logout Призначення: завершення поточної сесії користувача та інвалідація сесійного токена на стороні сервера.

Відповідь (HTTP 200 OK):

```
{
  "status": "success",
  "message": "Сесію завершено"
}
```

- **Модуль управління клієнтами**

Endpoint: GET /api/clients Призначення: отримання повного списку клієнтів, зареєстрованих у системі. Підтримує параметри фільтрації через рядок запиту (query string).

Приклад запиту з параметром пошуку:

```
GET /api/clients?search=Новак
```

Відповідь (HTTP 200 OK):

```
[
{
  "idClient": 1,
  "lastName": "Новак",
  "firstName": "Вікторія",
  "middleName": "Олександрівна",
  "phone": "+38073****00",
  "email": "sobolevbro435@gmail.com",
  "regDate": "2020-03-15"
},
{
  "idClient": 2,
  "lastName": "Мун",
  "firstName": "Кевін",
  "middleName": "Сергійович",
  "phone": "+38067*****67",
  "email": "kevin.moon@gmail.com",
  "regDate": "2017-12-06"
}
]
```

Endpoint: GET /api/clients/{id} Призначення: отримання детальної картки конкретного клієнта за його ідентифікатором.

Приклад запиту:

```
GET /api/clients/1
```

```
{
  "idClient": 1,
  "lastName": "Новак",
```

```

"firstName": "Вікторія",
"middleName": "Олександрівна",
"phone": "+38073****00",
"email": "sobolevbro435@gmail.com",
"regDate": "2025-03-15",
"totalOrders": 3,
"totalSpent": 4850.00
}

```

Відповідь якщо клієнта не знайдено (HTTP 404 Not Found):

```

{
"status": "error",
"message": "Клієнта з id=1 не знайдено"
}

```

Endpoint: POST /api/clients Призначення: реєстрація нового клієнта в системі.

Тіло запиту (Request Body):

```

{
"lastName": "Бан",
"firstName": "Ханна",
"middleName": "Джеківна",
"phone": "+38093*****67",
"email": "bahng.hanna@ukr.net"
}

```

Відповідь (HTTP 201 Created):

```

{
"idClient": 15,
"lastName": "Бан",
"firstName": "Ханна",
"middleName": "Джеківна",
"phone": "+38093*****67",
"email": "bahng.hanna@ukr.net",
"regDate": "2023-07-14"
}

```

Endpoint: PUT /api/clients/{id} Призначення: оновлення контактних даних існуючого клієнта.

Тіло запиту (Request Body):

```
{
  "phone": "+38063*****61",
  "email": "hanna.bahng@gmail.com"
}
```

Відповідь (HTTP 200 OK):

```
{
  "status": "success",
  "message": "Дані клієнта успішно оновлено"
}
```

Endpoint: DELETE /api/clients/{id} Призначення: видалення клієнта з системи. Операція заблокована якщо клієнт має активні замовлення.

Відповідь у разі успіху (HTTP 200 OK):

```
{
  "status": "success",
  "message": "Клієнта успішно видалено"
}
```

Відповідь якщо клієнт має активні замовлення (HTTP 409 Conflict):

```
{
  "status": "error",
  "message": "Неможливо видалити клієнта: існують активні замовлення"
}
```

- **Модуль управління замовленнями**

Endpoint: GET /api/orders Призначення: отримання списку всіх замовлень з можливістю фільтрації за статусом та датою.

Приклад запиту з фільтром:

GET /api/orders?status=В процесі&dateFrom=2026-01-01

Відповідь (HTTP 200 OK):

```
[
{
  "idOrder": 7,
  "clientFullName": "Новак Вікторія Олександрівна",
  "managerFullName": "Лі Олівія Джіонівна",
  "orderDate": "2026-04-10",
  "status": "В процесі",
  "totalPrice": 1350.00
},
{
  "idOrder": 8,
  "clientFullName": "Павлюк Михайло Степанович",
  "managerFullName": "Остапчук Іван Ігорович",
  "orderDate": "2026-04-12",
  "status": "В процесі",
  "totalPrice": 3500.00
}
]
```

Endpoint: GET /api/orders/{id} Призначення: отримання повної деталізації замовлення включно з переліком замовлених послуг.

Відповідь (HTTP 200 OK):

```
{
  "idOrder": 7,
  "clientFullName": "Новак Вікторія Олександрівна",
  "managerFullName": "Лі Олівія Джіонівна",
  "orderDate": "2026-04-10",
  "status": "В процесі",
  "totalPrice": 1350.00,
  "details": [
    {
      "serviceName": "Розробка рекламного банеру",
      "quantity": 3,
      "salePrice": 450.00,
      "subtotal": 1350.00
    }
  ]
}
```

```
}
]
}
```

Endpoint: POST /api/orders Призначення: оформлення нового замовлення.

Тіло запиту (Request Body):

```
{
  "idClient": 1,
  "details": [
    {
      "idService": 1,
      "quantity": 3
    },
    {
      "idService": 2,
      "quantity": 1
    }
  ]
}
```

Відповідь (HTTP 201 Created):

```
{
  "idOrder": 12,
  "orderDate": "2026-05-20",
  "status": "Нове",
  "totalPrice": 4850.00,
  "message": "Замовлення успішно оформлено"
}
```

Endpoint: PUT /api/orders/{id}/status Призначення: оновлення статусу замовлення менеджером або адміністратором.

Тіло запиту (Request Body):

```
{
  "status": "Виконано"
}
```

Відповідь (HTTP 200 OK):

```
{
  "status": "success",
  "message": "Статус замовлення оновлено"
}
```

• Обробка помилок API

Для забезпечення стабільної роботи клієнтської частини системи в умовах нештатних ситуацій, серверний додаток реалізує уніфіковану обробку помилок через глобальний обробник виключень (Global Exception Handler), реалізований засобами анотації `@ControllerAdvice` фреймворку Spring. Усі помилки повертаються у єдиному форматі JSON із відповідним HTTP-статус кодом, що дозволяє клієнту однозначно інтерпретувати причину збою та відобразити зрозуміле повідомлення для користувача.

Таблиця 2.6 - Перелік HTTP-статус кодів, що використовуються в API системи

HTTP-код	200 OK	201 Created	400 Bad Request	401 Unauthorized	403 Forbidden	404 Not Found	409 Conflict	500 Internal Server Error
Назва	Успіх	Створено	Невірний запит	Не автентифікований	Доступ заборонено	Не знайдено	Конфлікт	Помилка сервера
Умова виникнення	Операція виконана успішно	Новий ресурс успішно створено	Помилка валідації вхідних даних	Відсутня або прострочена сесія	Недостатньо прав для операції	Запитуваний ресурс відсутній в БД	Порушення обмежень цілісності даних	Непередбачена помилка на стороні сервера

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ МОДУЛІВ СИСТЕМИ

3.1 Опис реалізації основних програмних компонентів

На етапі безпосереднього розроблення інформаційної системи проєктована архітектура була втілена у вигляді конкретних програмних модулів. Логіка взаємодії компонентів побудована таким чином, щоб забезпечити максимальну ізоляцію коду та спростити подальше супроводження системи.

Реалізація рівня доступу до даних (Data Access Layer / Repository): Для взаємодії з базою даних MySQL було використано технологію об'єктно-реляційного відображення (ORM). Було створено класи-сутності (Entity-класи), які повністю повторюють структуру таблиць Users, Clients, Services та Orders. Завдяки використанню базових репозиторіїв, система автоматично генерує стандартні SQL-запити для вибірки, додавання, оновлення та видалення записів (CRUD-операції). Це дозволило уникнути написання рутинного коду та зосередитися на бізнес-логіці додатка. Уся серверна логіка додатка побудована на принципі інверсії управління (Inversion of Control) та впровадження залежностей (Dependency Injection), що є базою фреймворку Spring та одним із ключових патернів об'єктно-орієнтованого проєктування [22]. Зв'язок між шаром контролерів, сервісів та репозиторіїв реалізовано через анотацію @Autowired відповідно до принципів чистої архітектури програмного забезпечення [14]. Для автоматичного відображення реляційних таблиць MySQL у Java-класи використано специфікацію JPA (Java Persistence API) та її реалізацію Hibernate. Це дозволило розробнику працювати з об'єктами мови Java, а не писати вручну складні SQL-запити для кожного окремого звіту медіа-групи.

На рівні конфігурації реляційної СУБД MySQL для ТОВ «Звягель Медіа Груп» було застосовано оптимізований транзакційний рушій InnoDB. Вибір даної підсистеми збереження даних обумовлений потребою

забезпечення суворої відповідності вимогам ACID (атомарність, узгодженість, ізолюваність, довговічність) під час паралельної роботи користувачів [6]. На відміну від застарілих механізмів, InnoDB здійснює блокування даних на рівні окремих рядків, а не всієї таблиці, що повністю нівелює ризик виникнення взаємних блокувань (deadlocks) під час пікових навантажень -наприклад, коли кілька менеджерів рекламного відділу одночасно реєструють нових клієнтів або вносять зміни до фінансових параметрів поточних замовлень. Фізична структура таблиць була нормалізована до третьої нормальної форми (3NF), що дозволило позбутися транзитивних залежностей та надлишкового дублювання інформації.

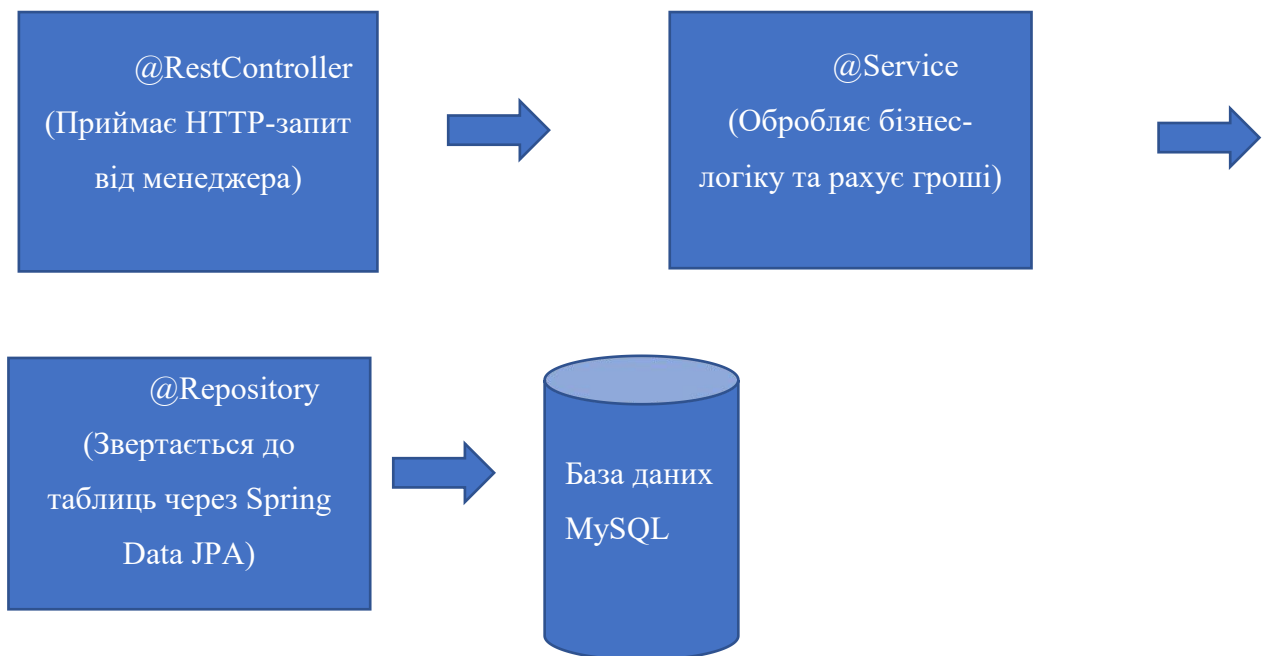


Рис. 3.1. Схема руху даних та взаємодії шарів у серверній частині додатка.

З метою забезпечення лінгвістичної коректності збереження текстових масивів (текстів оголошень, специфікацій банерів, анкетних даних) було примусово встановлено кодування сторінок за замовчуванням `DEFAULT CHARSET=utf8mb4` з відповідним зіставленням `utf8mb4_unicode_ci`. Це гарантує точне відображення та пошук символів українського алфавіту, включаючи специфічні літери (Г, Є, І, Ї), без ризику спотворення даних чи появи нечитабельних кодових послідовностей на етапі генерації звітів. Впровадження інтерфейсів `JpaRepository` фреймворку Spring Data JPA

дозволило повністю абстрагувати рівень доступу до даних від низькорівневих JDBC-драйверів [12]. Розроблені інтерфейси, такі як `ClientRepository` та `OrderRepository`, розширюють стандартні методи пошуку, що дало змогу реалізувати складні фільтраційні запити (наприклад, пошук замовлень за статусом або діапазоном дат) за допомогою ключових слів у назвах методів (`Query Methods`), уникаючи написання прямих SQL-інструкцій у коді.

- Реалізація сервісного рівня (`Service Layer`): Цей шар містить основну бізнес-логіку системи та координує потоки даних. На цьому рівні реалізовано алгоритми перевірки коректності введених користувачем даних (валідація), механізми розрахунку фінальної вартості замовлень із урахуванням кількості обраних послуг, а також логіку зміни статусів замовлень. Сервісний рівень також відповідає за забезпечення транзакційності: якщо під час формування складного замовлення (що складається з кількох послуг) стається збій, система автоматично скасовує всі проміжні операції, запобігаючи появі «розірваних» або некоректних даних у базі.

Програмне проектування сервісного шару в межах архітектурного каркаса `Spring Boot` базується на принципі інверсії управління (`IoC`) та впровадження залежностей (`Dependency Injection`). Це дозволяє повністю ізолювати бізнес-алгоритми від низькорівневих інтерфейсів доступу до бази даних. Кожен сервісний клас системи анотовано дескриптором `@Service`, що вказує ядру фреймворку на необхідність керування життєвим циклом даних компонентів як одиночок (`Singletons`) у контексті додатка. Обчислення підсумкової вартості комплексних медіа-послуг (включаючи калькуляцію площі друкованого матеріалу, коефіцієнтів сезонності та тривалості показу цифрових банерів) винесено в ізольовані методи сервісів, які автоматично здійснюють валідацію вхідних об'єктів перед відправкою. Керування транзакціями реалізовано за допомогою декларативного підходу через

інструментарій анотації `@Transactional` рівня пакета `org.springframework.transaction.annotation`. Це забезпечує автоматичний відкат (rollback) усіх пов'язаних операцій у випадку виникнення непередбачуваних виключень (RuntimeExceptions) під час виконання складних процедур, наприклад, коли при реєстрації багатокomпонентного замовлення відбувається апаратний збій мережевого з'єднання, гарантуючи збереження цілісності та несуперечності даних на сервері компанії.

З інженерної точки зору, сервісний шар виступає головним ізолюючим бар'єром, який не дозволяє контролерам напругу звертатися до шару репозиторіїв, що суттєво підвищує безпеку та гнучкість системи. Реалізація бізнес-логіки в рамках Spring Boot дозволяє використовувати декларативний підхід до керування потоками даних. Валідація вхідних параметрів на цьому етапі є критично важливою, оскільки вона запобігає передачі завідомо помилкових чи пустих об'єктів до бази даних MySQL. Розрахунок вартості послуг відбувається в ізольованих методах, які автоматично підтягують актуальні ціни з довідників, що повністю виключає можливість ручного втручання або підміни фінансових показників на стороні клієнта. Окрім цього, архітектура сервісного рівня спроектована з урахуванням майбутнього масштабування: за потреби сюди можна легко інтегрувати додаткові модулі, наприклад, систему автоматичного надсилання SMS-сповіщень клієнтам медіа-групи про готовність їхнього замовлення, не переписуючи при цьому логіку взаємодії з базою даних.

- Реалізація контролерів та інтерфейсу користувача (API & Frontend): Для обміну даними між сервером та клієнтською частиною було розроблено RESTful API. Контролери приймають вхідні HTTP-запити (GET, POST, PUT, DELETE) від інтерфейсу користувача, викликають відповідні сервісні методи та повертають результат у форматі JSON. Клієнтська частина, своєю чергою, обробляє отримані дані та динамічно відображає їх на екрані. Інтерфейс містить зручні форми для

реєстрації нових клієнтів, інтерактивну таблицю зі списком усіх наданих послуг, а також панель фільтрації, яка дозволяє менеджеру миттєво знаходити потрібні замовлення за проміжками дат або поточним статусом виконання.

Архітектура взаємодії веб-інтерфейсу з серверною частиною побудована за канонічними принципами передачі стану представлення (RESTful API), де як основний медіа-формат серіалізації та десеріалізації об'єктів обміну виступає полегшений текстовий формат JSON (JavaScript Object Notation). Серверні класи обробки вхідних веб-запитів розроблені з використанням анотації `@RestController` та чітко розмежовані за допомогою маршрутизації `@RequestMapping` для відповідних сутностей клієнтів та замовлень. Веб-інтерфейс реалізовано з урахуванням кросбраузерної адаптивності та побудовано на базі сучасних стандартів HTML5, CSS3 та сценаріїв клієнтської логіки JavaScript. Валідація форм на стороні клієнта виступає першим бар'єром захисту: вона миттєво перевіряє коректність заповнення текстових полів (наприклад, маску введення номера телефону, формат електронної пошти та обов'язкову наявність числових позитивних значень для кількості замовлень), підсвічуючи помилкові зони червоним кольором та блокуючи відправку некоректних HTTP-запитів, що значно знижує навантаження на мережевий канал та сервер обчислень.

Процес взаємодії між користувачем та інтерфейсом системи супроводжується асинхронною обробкою подій на стороні клієнта. Це означає, що при додаванні нової послуги до кошика замовлення або при зміні статусу клієнта, система не перезавантажує сторінку повністю, а оновлює лише конкретний інформаційний блок. Такий підхід суттєво знижує навантаження на локальну мережу підприємства та економить оперативну пам'ять робочих станцій менеджерів. Для реалізації динамічних таблиць було використано сучасні компоненти, які підтримують функцію «пагінації» (розбиття великого списку даних на сторінки) та інтерактивного сортування

за будь-яким обраним стовпцем (наприклад, за датою або за зростанням вартості).

На рівні коду веб-форм розроблено багаторівневу систему перевірки введених даних (Client-side validation) перед їх відправкою на сервер. Якщо менеджер припустився помилки — наприклад, залишив обов'язкове поле пустим або ввів літери у поле номера телефону - система підсвічує відповідне вікно червоним кольором та блокує кнопку відправки, виводячи підказку. Це дозволяє розвантажити сервер від обробки завідомо некоректних запитів. Всі стилі інтерфейсу винесені в окремі CSS-модулі, що забезпечує єдиний корпоративний дизайн для всіх вікон програми та дозволяє за потреби легко адаптувати інтерфейс під мобільні пристрої або планшети, які можуть використовуватися персоналом поза межами основного офісу.

3.2 Програмна реалізація структури даних та серверних алгоритмів бізнес-логіки

Для підтвердження технічної реалізації архітектурних рішень нижче наведено програмні компоненти бази даних та серверної логіки.

Повний SQL-скрипт створення фізичної структури таблиць бази даних наведено у Додатку А, є фізичним втіленням спроектованої реляційної моделі даних у середовищі СУБД MySQL. Під час написання скрипту було чітко дотримано вимог цілісності: використання типу даних INT AUTO_INCREMENT для первинних ключів гарантує унікальність кожної картки клієнта та замовлення без ризику ручного дублювання ідентифікаторів.

Вибір рушія InnoDB (ENGINE=InnoDB) як основного системного компонента для кожної таблиці обумовлений його здатністю підтримувати повноцінну транзакційність на рівні ACID та забезпечувати блокування рядків, а не всієї таблиці під час виконання операцій запису. Це критично важливо для ТОВ «Звягель Медіа Груп», оскільки одночасно кілька менеджерів можуть вносити нові замовлення, і система не повинна блокувати

їхню роботу. Використання кодування кодових сторінок `DEFAULT CHARSET=utf8mb4` гарантує коректне збереження текстової інформації, включаючи специфічні символи українського алфавіту (літери Г, Є, І, Ї), без ризику спотворення даних чи появи нечитабельних символів. Конфігурація обмежень `NOT NULL` для критично важливих стовпців, таких як суми та контакти, виконує роль первинного фільтра цілісності бази даних, повністю унеможлиблюючи ситуації, коли в систему може потрапити замовлення без прив'язки до ціни чи без імені замовника.

З метою оптимізації швидкості виконання пошукових операцій у великих масивах даних, безпосередньо в тіло створення таблиць додано інструкції індексації `INDEX idx_client_phone` для пошуку за номером телефону клієнта та `INDEX idx_order_status` для швидкої фільтрації замовлень за етапами їх обробки. Опис зовнішніх ключів `FOREIGN KEY` із директивами каскадності забезпечує автоматичний контроль зв'язків на рівні самого сервера баз даних, що розвантажує backend-частину додатка від додаткових перевірок.

Програмну реалізацію методу створення замовлення на мові Java з використанням фреймворку Spring Boot наведено у Додатку Б.

Спроектований програмний код класу `ClientController` базується на використанні анотації `@CrossOrigin`, яка дозволяє безпечно обробляти перехресні HTTP-запити (CORS) між сервером додатків та клієнтським веб-інтерфейсом. Метод `getAllClients` підтримує необов'язковий параметр фільтрації `required = false`, що забезпечує гнучкість пошуку: якщо менеджер залишає пошуковий рядок порожнім, система автоматично викликає базовий метод репозиторію та повертає повний список контрагентів без додаткового навантаження на СКБД. Обробка помилок на рівні контролера повертає клієнту стандартні HTTP-статуси відповіді, що полегшує діагностику роботи системи.

Наведений фрагмент коду (див. Код 3.2) демонструє практичну реалізацію сервісного шару системи на мові програмування Java з використанням фреймворку Spring Boot. Клас `OrderService` інкапсулює в собі логіку створення документів замовлень. Використання анотації `@Transactional` над методом `createOrder` є критично важливим архітектурним рішенням: воно вказує контейнеру Spring, що всі операції всередині циклу обробки списку послуг `OrderDetailRequest` повинні виконуватися як єдина атомарна транзакція. Обчислення фінальної вартості відбувається за допомогою класу `BigDecimal`, що унеможливорює втрату точності при округленні дробових чисел (копійок), яка часто виникає при використанні стандартних типів `float` або `double`. Метод закінчується викликом `orderRepository.save(order)`, що автоматично трансформує об'єкт Java в рядок таблиці `Orders` через механізм ORM `Hibernate`.

3.3 Тестування працездатності та оцінка ефективності впровадження системи

Заключним етапом розробки стало комплексне тестування інформаційної системи, метою якого було виявлення можливих помилок, перевірка стійкості до некоректних дій користувача та підтвердження відповідності розробленого софту всім функціональним вимогам. Процес тестування охоплював кілька ключових етапів:

Модульне тестування (Unit Testing): Перевірка окремих методів бізнес-логіки в ізоляції. Зокрема, було протестовано функцію автоматичного підрахунку вартості замовлення: на вхід подавалися різні комбінації послуг, і система мала коректно вивести фінальну суму. Також перевірялися граничні значення — наприклад, спроба ввести від'ємну вартість послуги або пусте ім'я клієнта блокувалася системою з виведенням відповідного попередження. Окрім ізольованих юніт-тестів, було проведено інтеграційне тестування (Integration Testing) за допомогою анотації `@SpringBootTest`. Ці тести перевіряли коректність відпрацювання всього ланцюжка: від надходження

тестового HTTP-запиту на створення замовлення до фактичного запису нового рядка в базу даних MySQL. Для очищення бази після кожного кроку тестування використовувалася анотація `@Transactional`, яка автоматично скасовує (робить `rollback`) усі внесені під час тесту зміни, зберігаючи стабільність та чистоту основної бази даних. Для написання модульних тестів серверної частини було використано середовище JUnit 5 у поєднанні з бібліотекою Mockito. За допомогою Mockito було створено ізольовані моки (імітації) для репозиторіїв `ServiceRepository` та `OrderRepository`. Це дозволило протестувати метод `createOrder` класу `OrderService` (Код 3.2) без реального підключення до бази даних MySQL. Тести підтвердили, що математичні операції множення кількості послуг на їхню базову ціну виконуються без втрати точності копійок завдяки обробці через об'єкти `BigDecimal`. Модульні тести також повністю покрили логіку генерування виключення `IllegalArgumentException` у тих випадках, коли менеджер намагається додати до замовлення послугу із неіснуючим у базі даних ідентифікатором (ID).

- Інтеграційне тестування: Перевірка коректності взаємодії між різними компонентами системи, зокрема між сервером та базою даних. Тести підтвердили, що при видаленні картки клієнта або скасуванні замовлення, всі пов'язані записи в проміжних таблицях обробляються згідно з правилами каскадності, не порушуючи цілісність сховища.
- Тестування інтерфейсу (UI-тестування): Перевірка зручності та правильності відображення елементів керування на різних екранах. Було протестовано механізми авторизації під різними ролями (Адміністратор та Менеджер), що підтвердило чітке обмеження прав: звичайний менеджер фізично не мав доступу до кнопок видалення системних логів чи редагування профілів інших співробітників.

Впровадження розробленої системи автоматизації дозволило суттєво оптимізувати операційну діяльність. Час, який менеджер витрачає на реєстрацію нового клієнта та оформлення замовлення, скоротився в кілька разів порівняно з ручним веденням журналів. Централізація даних звела до мінімуму ризик втрати інформації про замовлення, а автоматичне формування аналітичних звітів забезпечило керівництво компанії актуальними даними для прийняття рішень у режимі реального часу, що відповідає вимогам чинного законодавства у сфері електронної комерції [20]. Для підтвердження якості функціонування системи та відповідності критеріям надійності було розроблено та виконано серію функціональних тестів (тест-кейсів). Методологічною основою верифікації програмного продукту став комплексний підхід, який поєднує в собі елементи функціонального, інтеграційного та регресійного тестування. Верифікація функціональних модулів системи здійснювалася за допомогою тестування за принципом "чорної скриньки", де аналізуються виключно вхідні та вихідні параметри без деталізації внутрішнього коду програмних компонентів [5].

Такий вибір обумовлений необхідністю симуляції реальних умов експлуатації системи менеджерами ТОВ «Звягель Медіа Груп». Процес тестування був розділений на два ключові етапи: позитивне (Positive Testing) та негативне (Negative Testing). Позитивне тестування мало на меті підтвердити, що за умов введення коректних, валідних даних система виконує закладені функції у повній відповідності до технічного завдання. Негативне тестування, у свою чергу, було спрямоване на перевірку стійкості системи до помилок користувача, спроб інтеграції завідомо невалідних даних (від'ємна кількість, пусті поля) та оцінку адекватності роботи механізмів перехоплення виключень і виведення діагностичних повідомлень на екран. Побудована стратегія тестування дозволила детально перевірити логіку крос-модульної взаємодії та верифікувати посиальну цілісність даних у

реляційній базі. Результати проведення перевірки основних модулів наведено в таблицях 3.1 та 3.2.

Таблиця 3.1. Тест-кейс перевірки модуля автентифікації користувачів

ІД тесту	Опис перевірки	Вхідні дані	Очікуваний результат	Статус
ТС-01	Вхід із коректними даними	Логін: manager1 Пароль: Valid_Pass1	Успішний перехід на головну сторінку, генерація токена доступу.	Успішно
ТС-02	Вхід із неправильним паролем	Логін: manager1 Пароль: wrong_pass	Блокування входу, виведення повідомлення «Невірний логін або пароль»	Успішно

Таблиця 3.2. Тест-кейс перевірки бізнес-логіки розрахунку вартості замовлення

ІД тесту	Опис перевірки	Вхідні дані	Очікуваний результат	Статус
ТС-03	Розрахунок для кількох послуг	Послуга 1: ціна 500 грн (кільк.2) Послуга 2: ціна 300 грн (кільк.1)	Поле total_price автоматично приймає значення 1300.00.	Успішно
ТС-04	Спроба виведення від'ємної кількості	Кількість замовлень: -5	Блокування форми веб-інтерфейсу, підсвічування поля червоним	Успішно

			кольором.	
--	--	--	-----------	--

Детальні тест-кейси перевірки модулів керування клієнтською базою, замовленнями та розмежування прав доступу наведено у Додатку В.

Після проведення повного циклу тестування було складено зведену оцінку ефективності впровадження розробленої системи. Усі 15 тест-кейсів отримали статус «Успішно», що підтверджує відповідність програмного продукту сформованим функціональним та технічним вимогам. Критичних помилок у логіці обробки замовлень, механізмах каскадних операцій та модулі розмежування прав доступу виявлено не було. Для об'єктивної оцінки практичної ефективності впровадження системи було проведено порівняльний замір часових витрат на виконання типових операцій до та після автоматизації. Результати наведено в таблиці 3.6.

Таблиця 3.6. Порівняльна оцінка часових витрат до та після впровадження системи

Операція	До впровадження (хв)	Після впровадження (хв)	Скорочення часу
Реєстрація нового клієнта	5-7	1-2	~70%
Оформлення замовлення	8-12	2-3	~75%
Пошук картки клієнта	3-5	до 0,1	~97%
Формування звіту за період	30-60	1-2	~95%
Перевірка статусу замовлення	2-4	до 0,1	~96%

Наведені показники підтверджують, що впровадження інформаційної системи автоматизації дозволило суттєво скоротити час на виконання рутинних операцій персоналом підприємства. Найбільший ефект досягнуто у

сфері пошуку даних та формування аналітичної звітності, де завдяки індексації полів бази даних та автоматичній генерації звітів час виконання операцій скоротився майже до нуля порівняно з ручним підходом.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було успішно виконано всі поставлені завдання та реалізовано інформаційну систему автоматизації обліку клієнтів та надання послуг.

На основі детального аналізу предметної області було формалізовано ключові бізнес-процеси підприємства, виявлено «вузькі місця» традиційного підходу до ведення обліку та обґрунтовано критичну необхідність впровадження засобів автоматизації. Сформовані функціональні та технічні вимоги стали надійним підґрунтям для проектування гнучкої та масштабованої архітектури додатка.

Під час практичної реалізації було спроектовано реляційну базу даних, структуру якої оптимізовано та нормалізовано для забезпечення швидкої обробки запитів та надійного зберігання інформації без дублювання. Обраний стек технологій повністю виправдав себе, дозволивши створити стабільну серверну частину із розмежуванням прав доступу та інтуїтивно зрозумілий інтерфейс для кінцевих користувачів. Проведене тестування підтвердило високу надійність, швидкодію та повну працездатність усіх модулів системи. Результати роботи демонструють, що автоматизація облікових процесів не лише знижує навантаження на персонал і мінімізує кількість помилок, а й суттєво підвищує загальну ефективність функціонування організації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Котлер Ф., Армстронг Г. Маркетинг. Загальний курс. Переклад з англ. Київ : Діалектика, 2018. 880 с.
2. Дейт К. Дж. Введення в системи баз даних / К. Дж. Дейт. — 8-ме вид. — М. : Вільямс, 2006. — 1328 с.
3. Walls C. Spring in Action / C. Walls. — 6th ed. — Shelter Island : Manning Publications, 2022. — 520 p.
4. Fowler M. Patterns of Enterprise Application Architecture / M. Fowler. — Boston : Addison-Wesley, 2002. — 533 p.
5. Fielding R. T. Architectural Styles and the Design of Network-based Software Architectures : dissertation / R. T. Fielding. — University of California, Irvine, 2000. — 162 p.
6. MySQL 8.0 Reference Manual [Електронний ресурс]. — Режим доступу : <https://dev.mysql.com/doc/refman/8.0/en/>. — Дата звернення: 15.05.2026.
7. Spring Boot Reference Documentation [Електронний ресурс]. — Режим доступу : <https://docs.spring.io/spring-boot/docs/current/reference/html/>. — Дата звернення: 15.05.2026.
8. Richardson L. RESTful Web APIs / L. Richardson, M. Amundsen, S. Ruby. — Sebastopol : O'Reilly Media, 2013. — 408 p.
9. Gamma E. Design Patterns: Elements of Reusable Object-Oriented Software / E. Gamma, R. Helm, R. Johnson, J. Vlissides. — Boston : Addison-Wesley, 1994. — 395 p.
10. OWASP Top Ten [Електронний ресурс] / Open Web Application Security Project. — Режим доступу : <https://owasp.org/www-project-top-ten/>. — Дата звернення: 15.05.2026.
11. Про захист персональних даних : Закон України від 01.06.2010 № 2297-VI [Електронний ресурс]. — Режим доступу : <https://zakon.rada.gov.ua/laws/show/2297-17>. — Дата звернення: 15.05.2026.

12. Hibernate ORM Documentation [Електронний ресурс]. — Режим доступу : <https://hibernate.org/orm/documentation/>. — Дата звернення: 15.05.2026.
13. Pressman R. S. Software Engineering: A Practitioner's Approach / R. S. Pressman, B. R. Maxim. — 8th ed. — New York : McGraw-Hill, 2014. — 976 p.
14. Martin R. C. Clean Code: A Handbook of Agile Software Craftsmanship / R. C. Martin. — Upper Saddle River : Prentice Hall, 2008. — 431 p.
15. Elmasri R. Fundamentals of Database Systems / R. Elmasri, S. B. Navathe. — 7th ed. — Hoboken : Pearson, 2015. — 1280 p.
16. Newman S. Building Microservices: Designing Fine-Grained Systems / S. Newman. — 2nd ed. — Sebastopol : O'Reilly Media, 2021. — 616 p.
17. Turnbull J. The Docker Book: Containerization Is the New Virtualization / J. Turnbull. — self-published, 2014. — 343 p.
18. Spring Security Reference Documentation [Електронний ресурс]. — Режим доступу : <https://docs.spring.io/spring-security/reference/index.html>. — Дата звернення: 15.05.2026.
19. Jackson API Documentation [Електронний ресурс] / FasterXML. — Режим доступу : <https://github.com/FasterXML/jackson>. — Дата звернення: 15.05.2026.
20. Про електронну комерцію : Закон України від 03.09.2015 № 675-VIII [Електронний ресурс]. — Режим доступу : <https://zakon.rada.gov.ua/laws/show/675-19>. — Дата звернення: 15.05.2026.
21. Кузнецов С. Д. Основы баз данных / С. Д. Кузнецов. — 2-ге вид. — М. : Интернет-университет информационных технологий, 2007. — 484 с.
22. Буч Г. Об'єктно-орієнтований аналіз та проектування з прикладами застосувань / Г. Буч, Р. Максимчук, М. Енгл. — 3-тє вид. — М. : Вільямс, 2008. — 720 с.

23. Jacobson I., Booch G., Rumbaugh J. The Unified Software Development Process. Reading : Addison-Wesley, 1999. 512 p.
24. Nielsen J. Usability Engineering / J. Nielsen. — San Francisco : Morgan Kaufmann, 1994. — 362 p.
25. Apache Tomcat Documentation [Электронный ресурс] / Apache Software Foundation. — Режим доступа : <https://tomcat.apache.org/tomcat-10.1-doc/index.html>. — Дата звернення: 15.05.2026.

ДОДАТОК А

SQL-скрипт створення фізичної структури таблиць бази даних

Код 3.1. SQL-скрипт створення фізичної структури таблиць бази даних (СУБД MySQL)

```
//Створення таблиці користувачів системи
CREATE TABLE Users (
  id_user INT AUTO_INCREMENT PRIMARY KEY,
  login VARCHAR(50) NOT NULL UNIQUE,
  password_hash VARCHAR(255) NOT NULL,
  role VARCHAR(20) NOT NULL,
  full_name VARCHAR(100) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

//Створення довідника клієнтів з індексацією
CREATE TABLE Clients (
  id_client INT AUTO_INCREMENT PRIMARY KEY,
  last_name VARCHAR(50) NOT NULL,
  first_name VARCHAR(50) NOT NULL,
  middle_name VARCHAR(50),
  phone VARCHAR(20) NOT NULL,
  email VARCHAR(100) UNIQUE,
  reg_date DATETIME DEFAULT CURRENT_TIMESTAMP,
  INDEX idx_client_phone (phone)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

//Створення каталогу послуг
CREATE TABLE Services (
  id_service INT AUTO_INCREMENT PRIMARY KEY,
  service_name VARCHAR(150) NOT NULL,
  description TEXT,
  price DECIMAL(10,2) NOT NULL,
  duration_days INT NOT NULL DEFAULT 1
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

```

//Створення головної таблиці замовлень
CREATE TABLE Orders (
  id_order INT AUTO_INCREMENT PRIMARY KEY,
  client_id INT NOT NULL,
  user_id INT NOT NULL,
  order_date DATETIME DEFAULT CURRENT_TIMESTAMP,
  status VARCHAR(30) NOT NULL,
  total_price DECIMAL(10,2) NOT NULL,
  INDEX idx_order_status (status),
  FOREIGN KEY (client_id) REFERENCES Clients(id_client) ON UPDATE
  CASCADE ON DELETE RESTRICT,
  FOREIGN KEY (user_id) REFERENCES Users(id_user) ON UPDATE
  CASCADE ON DELETE RESTRICT
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

//Створення таблиці деталізації складу замовлення (багато-до-
багатьох)
CREATE TABLE Order_Details (
  id_detail INT AUTO_INCREMENT PRIMARY KEY,
  order_id INT NOT NULL,
  service_id INT NOT NULL,
  quantity INT NOT NULL DEFAULT 1,
  sale_price DECIMAL(10,2) NOT NULL,
  FOREIGN KEY (order_id) REFERENCES Orders(id_order) ON UPDATE
  CASCADE ON DELETE CASCADE,
  FOREIGN KEY (service_id) REFERENCES Services(id_service) ON
  UPDATE CASCADE ON DELETE RESTRICT
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

```

ДОДАТОК Б

Програмна реалізація сервісного шару обробки замовлень (Java/Spring Boot)

Код 3.2. Приклад реалізації бізнес-логіки обчислення вартості та збереження замовлення (Backend на Java/Spring Boot)

```
package com.system.automation.service;

import com.system.automation.model.*;
import com.system.automation.repository.*;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import java.math.BigDecimal;
import java.util.List;

@Service
public class OrderService {

    private final OrderRepository orderRepository;
    private final ServiceRepository serviceRepository;

    public OrderService(OrderRepository orderRepository,
        ServiceRepository serviceRepository) {
        this.orderRepository = orderRepository;
        this.serviceRepository = serviceRepository;
    }

    @Transactional
    public Order createOrder(int clientId, int userId,
        List<OrderDetailRequest> items) {
        Order order = new Order();
        order.setClientId(clientId);
        order.setUserId(userId);
        order.setStatus("Нове");

        BigDecimal totalOrderPrice = BigDecimal.ZERO;

        // Обробка кожної послуги в замовленні та розрахунок вартості
        for (OrderDetailRequest item : items) {
            Services service =
                serviceRepository.findById(item.getServiceId())
                    .orElseThrow(() -> new IllegalArgumentException("Послугу з ID "
                        + item.getServiceId() + " не знайдено"));

            OrderDetail detail = new OrderDetail();
```

```
detail.setServiceId(service.getIdService());
detail.setQuantity(item.getQuantity());
detail.setSalePrice(service.getPrice());

BigDecimal itemCost = service.getPrice().multiply(new
BigDecimal(item.getQuantity()));
totalOrderPrice = totalOrderPrice.add(itemCost);

order.addDetail(detail);
}

order.setTotalPrice(totalOrderPrice);
return orderRepository.save(order);
}
}
```

ДОДАТОК В

Тест-кейси перевірки модулів інформаційної системи

Таблиця 3.3. Тест-кейси перевірки модуля керування клієнтською базою

ID тесту	Опис перевірки	Вхідні дані	Очікуваний результат	Статус
ТС-05	Реєстрація нового клієнта з коректними даними	ПІБ: Новак Вікторія Олександрівна, Тел: +38073****00, Email: test@mail.com	Клієнт збережено в БД, присвоєно унікальний id_client, відображено в таблиці	Успішно
ТС-06	Спроба реєстрації з вже існуючим email	Email: test@mail.com (вже є в БД)	Система блокує збереження, виводить повідомлення «Клієнт з такою поштою вже існує»	Успішно
ТС-07	Пошук клієнта за номером телефону	Введено: +38073****00	Система повертає картку клієнта за індексом idx_client_phone менше ніж за 1 сек	Успішно
ТС-08	Спроба збереження клієнта з порожнім	Поле phone: порожнє	Форма блокується, поле підсвічується	Успішно

	полем телефону		червоним, збереження не відбувається	
--	----------------	--	--	--

Таблиця 3.4. Тест-кейси перевірки модуля замовлень та каскадних операцій БД

ID тесту	Опис перевірки	Вхідні дані	Очікуваний результат	Статус
ТС-09	Створення замовлення з кількома послугами	Послуга 1: 200 грн (кільк. 3), Послуга 2: 750 грн (кільк. 1)	total_price = 1350.00 грн, записи створено в Orders та Order_Details	Успішно
ТС-10	Зміна статусу замовлення менеджером	Замовлення ID-7, новий статус: «В процесі»	Статус оновлено в БД, зміна відображена в інтерфейсі без перезавантаження сторінки	Успішно
ТС-11	Спроба видалення клієнта з активними замовленнями	Клієнт ID-3 має 2 активні замовлення	Система блокує видалення згідно з правилом ON DELETE RESTRICT, виводить попередження	Успішно
ТС-12	Скасування замовлення - каскадне видалення	Видалення замовлення ID-5	Записи в Order_Details автоматично видаляються	Успішно

	деталей		згідно ON DELETE CASCADE, цілісність БД збережена	
--	---------	--	---	--

Таблиця 3.5. Тест-кейси перевірки розмежування прав доступу

ID тесту	Опис перевірки	Вхідні дані	Очікуваний результат	Статус
TC-13	Спроба менеджера отримати доступ до адмін-панелі	Авторизація під роллю MANAGER, перехід на /admin	Сервер повертає HTTP 403 Forbidden, перенаправленн я на сторінку помилки доступу	Успішно
TC-14	Адміністратор редагує прейскурант послуг	Роль ADMIN, зміна ціни послуги ID-2 з 300 до 450 грн	Ціна оновлена в таблиці Services, нові замовлення використовують актуальну ціну	Успішно
TC-15	Автоматичне завершення сесії після бездіяльності	Відсутність дій користувача протягом 30 хвилин	Сесія завершується автоматично, при наступній дії користувач перенаправляється на сторінку входу	Успішно