

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ**

Факультет інформаційних технологій

ПОГОДЖЕНО
Декан факультету

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ
Завідувач кафедри інформаційних
систем і технологій

_____ **Ігор БОЛБОТ**
(підпис)

_____ **Михайло ШВИДЕНКО**
(підпис)

«__» _____ 2026 р.

«__» _____ 2026 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

**«Інформаційна система моделювання процесів управління малим
бізнесом»**

Спеціальність «122 Комп'ютерні науки»

Освітньо-професійна програма «Комп'ютерні науки»

Гарант освітньої програми
д.е.н., професор

_____ **Роман РУДЕНСЬКИЙ**
(підпис)

**Керівник бакалаврської
кваліфікаційної роботи**
науковий ступінь та вчене
звання

_____ **Вікторія СМОЛІЙ**
(підпис)

Виконав

_____ **Денис РОЗДОБУДЬКО**
(підпис)

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ЗАТВЕРДЖУЮ

**Завідувач кафедри інформаційних систем і
технологій**

к.е.н., доцент _____ Михайло
ШВИДЕНКО

(підпис)

« ____ » _____ 2026 р.

ЗАВДАННЯ

ДО ВИКОНАННЯ БАКАЛАВРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ ЗДОБУВАЧУ

Роздобудько Денису Сергійовичу
(прізвище, ім'я, по батькові)

Спеціальність «122 Комп'ютерні науки»

Освітньо-професійна програма «Комп'ютерні науки»

Тема бакалаврської кваліфікаційної роботи

«Інформаційна система моделювання процесів управління малим бізнесом»

затверджена наказом від «6» січня 2026 р. № 46 «С»

Термін подання завершеної роботи на кафедру «22» травня 2026 р.

Вихідні дані до бакалаврської кваліфікаційної роботи (проєкту):

методи аналізу та проєктування інформаційних систем, об'єктно-орієнтований підхід,
UML-моделювання, бази даних, вимоги до інформаційної системи моделювання
процесів управління малим бізнесом.

Перелік питань, що підлягають дослідженню:

аналіз предметної області; визначення вимог до системи; моделювання процесів
управління; проєктування бази даних; вибір технологій; розробка, тестування та
впровадження інформаційної системи.

Дата видачі завдання «6» січня 2026 р.

**Керівник бакалаврської
кваліфікаційної роботи**

(підпис)

Вікторія СМОЛІЙ

Завдання взяв до

виконання

(підпис)

Денис РОЗДОБУДЬКО

Зміст

ВСТУП	4
РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ТЕХНОЛОГІЧНИХ ЗАСОБІВ РОЗРОБКИ ІНФОРМАЦІЙНОЇ СИСТЕМИ	7
1.1. Поняття та особливості малого бізнесу.....	7
1.2. Основні процеси управління малим бізнесом.....	8
1.3. Роль інформаційних систем в управлінні підприємством.....	11
1.4. Аналіз існуючих інформаційних систем для малого бізнесу.....	13
1.5. Необхідність моделювання бізнес-процесів у малому бізнесі.....	16
1.6. Вибір мови програмування, середовища розробки та технологій.....	17
1.7. Висновки до розділу 1.....	19
РОЗДІЛ 2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ МОДЕЛЮВАННЯ ПРОЦЕСІВ УПРАВЛІННЯ	20
2.1. Постановка задачі розробки інформаційної системи.....	20
2.2. Визначення функціональних вимог до системи.....	22
2.3. Визначення нефункціональних вимог до системи.....	25
2.4. Розробка загальної архітектури інформаційної системи.....	26
2.5. Розробка діаграми варіантів використання інформаційної системи.....	28
2.6. Розробка діаграми діяльності основних бізнес-процесів.....	30
2.8. Проєктування бази даних інформаційної системи.....	34
2.9. Розробка ER-діаграми бази даних.....	37
2.10. Проєктування інтерфейсу користувача.....	39
2.11. Висновки до розділу 2.....	40
РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ	41
3.1. Налаштування календарів, валют та основних параметрів системи.....	41
3.2. Налаштування номенклатури товарів і послуг.....	44
3.3. Реалізація модуля обліку клієнтів.....	47

3.4. Реалізація модуля обліку контрагентів.....	51
3.5. Реалізація модуля управління складами.....	55
3.6. Реалізація модуля обліку замовлень і продажів.....	59
3.7. Реалізація модуля фінансового обліку.....	66
3.8. Реалізація модуля моделювання та аналізу бізнес-процесів.....	69
3.9. Реалізація звітності та візуалізації результатів.....	74
3.10. Висновки до розділу 3.....	78
РОЗДІЛ 4 ТЕСТУВАННЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ РОЗРОБЛЕНОЇ СИСТЕМИ.....	79
4.1. Підготовка тестових даних.....	79
4.2. Тестування налаштувань календарів, валют і номенклатури...82	
4.3. Тестування роботи з клієнтами та контрагентами.....	86
4.4. Тестування роботи з кількома замовленнями, складами та товарами.....	88
4.5. Перевірка роботи модулів обліку та управління.....	92
4.6. Тестування модуля моделювання бізнес-процесів.....	93
4.7. Аналіз результатів роботи системи.....	95
4.8. Оцінка ефективності використання системи для малого бізнесу	96
4.9. Висновки до розділу 4.....	98
ВИСНОВКИ.....	99
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	100

ВСТУП

У сучасних умовах розвитку економіки малий бізнес відіграє важливу роль у забезпеченні зайнятості населення, формуванні конкурентного середовища та розвитку локальних ринків товарів і послуг. Невеликі підприємства, магазини, майстерні, сервісні компанії та інші суб'єкти малого бізнесу щоденно виконують значну кількість управлінських операцій, пов'язаних з обліком клієнтів, товарів, послуг, замовлень, фінансових надходжень, витрат, складських залишків та результатів діяльності. Ефективність таких процесів безпосередньо впливає на стабільність роботи підприємства, швидкість прийняття рішень і рівень обслуговування клієнтів.

Однак у багатьох малих підприємствах управління основними бізнес-процесами досі здійснюється за допомогою розрізнених інструментів: електронних таблиць, паперових журналів, окремих файлів або простих облікових програм. Такий підхід часто призводить до дублювання інформації, помилок у розрахунках, втрати даних, складності контролю залишків товарів, затримок у формуванні звітності та ускладнення аналізу поточного стану бізнесу. У результаті керівник або адміністратор малого підприємства витрачає значну кількість часу на рутинні операції замість того, щоб зосередитися на розвитку бізнесу, роботі з клієнтами та прийнятті стратегічних рішень.

Актуальність теми бакалаврської роботи зумовлена необхідністю створення інформаційної системи, яка дозволяє автоматизувати та моделювати основні процеси управління малим бізнесом. Така система має забезпечувати зручне ведення обліку клієнтів, контрагентів, товарів, послуг, замовлень, складів і фінансових операцій, а також надавати можливість аналізувати результати діяльності підприємства за допомогою звітів і візуалізації даних. Особливе значення має саме моделювання бізнес-процесів, оскільки воно дає змогу краще зрозуміти структуру діяльності підприємства, виявити слабкі місця в управлінні та запропонувати шляхи їх оптимізації.

Метою бакалаврської роботи є розробка інформаційної системи моделювання процесів управління малим бізнесом, яка забезпечує автоматизацію основних управлінських операцій, облік ключових даних підприємства, аналіз бізнес-процесів і формування звітної інформації для підтримки прийняття управлінських рішень.

Для досягнення поставленої мети необхідно виконати такі завдання:

1. Проаналізувати предметну область малого бізнесу та визначити основні процеси управління підприємством.
2. Розглянути роль інформаційних систем у підвищенні ефективності управління малим бізнесом.
3. Проаналізувати існуючі програмні рішення для автоматизації малого бізнесу та визначити їх переваги й недоліки.
4. Обґрунтувати необхідність моделювання бізнес-процесів у межах інформаційної системи.
5. Визначити функціональні та нефункціональні вимоги до розроблюваної системи.
6. Спроектувати архітектуру інформаційної системи, структуру бази даних та основні діаграми роботи системи.
7. Розробити програмну реалізацію інформаційної системи з модулями обліку клієнтів, товарів, послуг, замовлень, складів, фінансів і звітності.
8. Провести тестування розробленої системи та оцінити ефективність її використання для малого бізнесу.

Об'єктом дослідження є процеси управління діяльністю малого бізнесу.

Предметом дослідження є методи, моделі та програмні засоби автоматизації і моделювання процесів управління малим бізнесом.

У роботі використовуються методи системного аналізу, моделювання бізнес-процесів, проєктування баз даних, об'єктно-орієнтованого програмування, структурного проєктування інформаційних систем, а також методи тестування програмного забезпечення. Для наочного подання структури системи передбачається використання діаграм варіантів використання, діаграм діяльності, діаграм послідовності, ER-діаграми бази даних та схеми загальної архітектури інформаційної системи.

Практичне значення роботи полягає у створенні програмної системи, яка може бути використана малими підприємствами для спрощення щоденних управлінських операцій. Розроблена система дозволяє централізовано зберігати дані, швидко отримувати інформацію про клієнтів, товари, послуги, замовлення та фінансові показники, а також формувати звіти для аналізу діяльності підприємства. Це сприяє зменшенню кількості помилок,

підвищенню швидкості обробки інформації та покращенню якості управлінських рішень.

Бакалаврська робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел і додатків. У першому розділі розглянуто предметну область малого бізнесу, основні процеси управління, роль інформаційних систем та вибір технологічних засобів розробки. У другому розділі виконано проєктування інформаційної системи, визначено вимоги, розроблено архітектуру, діаграми та структуру бази даних. У третьому розділі описано програмну реалізацію основних модулів системи. У четвертому розділі наведено тестування розробленої системи, аналіз результатів її роботи та оцінку ефективності використання для малого бізнесу.

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ТЕХНОЛОГІЧНИХ ЗАСОБІВ РОЗРОБКИ ІНФОРМАЦІЙНОЇ СИСТЕМИ

1.1. Поняття та особливості малого бізнесу

Малий бізнес є однією з важливих складових сучасної економічної системи, оскільки забезпечує гнучкість ринку, створення робочих місць, розвиток конкуренції та швидке реагування на потреби споживачів. До малого бізнесу зазвичай належать підприємства з невеликою кількістю працівників, обмеженим обсягом ресурсів та відносно невеликими масштабами господарської діяльності. Такі підприємства можуть працювати у сфері торгівлі, послуг, виробництва, ремонту, громадського харчування, доставки, консалтингу та інших напрямках.

Особливістю малого бізнесу є те, що більшість управлінських рішень приймається безпосередньо власником або керівником підприємства. На відміну від великих компаній, де управлінські функції розподілені між окремими відділами, у малому бізнесі одна особа або невелика команда часто відповідає одночасно за продажі, закупівлі, фінанси, роботу з клієнтами, складський облік і планування діяльності. Це робить процес управління більш оперативним, але водночас створює значне навантаження на персонал.

Малий бізнес має низку переваг. До них можна віднести швидкість прийняття рішень, гнучкість у зміні асортименту товарів або послуг, можливість швидко адаптуватися до потреб клієнтів, простішу організаційну структуру та більш тісний контакт із цільовою аудиторією. Завдяки невеликому масштабу діяльності такі підприємства можуть швидше впроваджувати нові підходи до обслуговування, змінювати цінову політику та реагувати на зміни ринку.

Водночас малий бізнес має і певні проблеми. Найчастіше вони пов'язані з обмеженими фінансовими, кадровими та технічними ресурсами. Невеликі підприємства не завжди мають можливість використовувати дорогі корпоративні інформаційні системи, утримувати окремий ІТ-відділ або впроваджувати складні програмні комплекси. Через це облік даних часто ведеться вручну або за допомогою простих електронних таблиць, що знижує точність, швидкість і надійність управлінських процесів.

Для малого бізнесу важливо мати прості, зручні та доступні інструменти автоматизації. Інформаційна система для такого підприємства повинна бути зрозумілою для користувача, не вимагати складного адміністрування, забезпечувати швидкий доступ до основних даних і підтримувати виконання щоденних операцій. До таких операцій можна віднести ведення клієнтської бази, облік товарів і послуг, оформлення замовлень, контроль залишків на складі, облік доходів і витрат, а також формування звітів.

Ще однією важливою особливістю малого бізнесу є залежність ефективності роботи від якості інформації. Якщо дані про клієнтів, замовлення, товари або фінанси зберігаються в різних місцях, керівнику складно швидко оцінити реальний стан підприємства. Наприклад, помилки в обліку складських залишків можуть призвести до нестачі товарів, затримки виконання замовлень або фінансових втрат. Неповна інформація про клієнтів ускладнює повторні продажі та знижує якість обслуговування.

Саме тому для малого бізнесу актуальним є впровадження інформаційних систем, які дозволяють об'єднати основні управлінські процеси в єдиному програмному середовищі. Така система дає змогу зменшити кількість ручних операцій, підвищити точність обліку, прискорити обробку інформації та забезпечити більш якісну підтримку управлінських рішень.

У межах даної бакалаврської роботи малий бізнес розглядається як підприємство, діяльність якого потребує автоматизації основних процесів управління: роботи з клієнтами, обліку товарів і послуг, управління замовленнями, складського контролю, фінансового обліку та аналізу результатів діяльності. Такий підхід дозволяє сформулювати вимоги до інформаційної системи, яка буде орієнтована саме на практичні потреби малого підприємства.

1.2. Основні процеси управління малим бізнесом

Управління малим бізнесом охоплює сукупність взаємопов'язаних процесів, спрямованих на забезпечення стабільної роботи підприємства, ефективне використання ресурсів, задоволення потреб клієнтів і отримання прибутку. Незважаючи на невеликі масштаби діяльності, мале підприємство виконує ті самі базові управлінські функції, що й більші організації: планування, організацію роботи, облік, контроль, аналіз і прийняття рішень.

Одним з основних процесів управління є **планування діяльності підприємства**. Воно передбачає визначення цілей, прогнозування обсягів продажів, планування закупівель, розподіл ресурсів і формування очікуваних фінансових результатів. Для малого бізнесу планування має особливе значення, оскільки навіть незначні помилки у закупівлях, ціноутворенні або розрахунку витрат можуть суттєво вплинути на прибутковість підприємства.

Важливим процесом є **управління клієнтами**. Для малого бізнесу клієнтська база є одним із ключових ресурсів, оскільки постійні клієнти забезпечують повторні продажі та стабільний дохід. Управління клієнтами включає зберігання контактної інформації, історії замовлень, особливостей співпраці, статусу клієнта та можливих заборгованостей. Наявність такої інформації дозволяє краще організувати обслуговування, швидше реагувати на запити та підвищувати лояльність клієнтів.

Наступним важливим напрямом є **управління товарами та послугами**. Підприємство повинно мати актуальну інформацію про асортимент, ціни, характеристики товарів або послуг, наявність залишків і зміни в номенклатурі. Якщо ці дані ведуться вручну, виникає ризик помилок, дублювання позицій або використання неактуальної інформації. Інформаційна система дозволяє централізовано зберігати номенклатуру та швидко використовувати її під час оформлення замовлень, продажів і звітів.

Окреме місце займає **управління замовленнями та продажами**. Цей процес включає створення замовлення, вибір клієнта, додавання товарів або послуг, розрахунок суми, зміну статусу замовлення, контроль виконання та фіксацію оплати. Для малого бізнесу важливо, щоб замовлення оброблялися швидко й без помилок, адже від цього залежить якість обслуговування та репутація підприємства.

Не менш важливим є **складський облік**. Він забезпечує контроль наявності товарів, надходжень, списань, переміщень і залишків. Якщо підприємство продає товари, неправильний складський облік може призвести до ситуацій, коли товар фактично відсутній, але в документах він ще числиться доступним. Це ускладнює виконання замовлень і може спричинити втрату клієнтів. Тому автоматизація складського обліку є однією з ключових функцій інформаційної системи.

Ще одним важливим процесом є **фінансовий облік**. Він включає фіксацію доходів, витрат, оплат від клієнтів, платежів постачальникам, прибутку та

інших фінансових показників. У малому бізнесі фінансовий облік часто ведеться спрощено, однак керівнику все одно необхідно бачити реальну картину: скільки коштів надійшло, які витрати були здійснені, які замовлення оплачені, а які ще очікують оплати. Без систематизованого обліку складно оцінити ефективність діяльності підприємства.

Важливу роль відіграє також **робота з контрагентами**. До контрагентів можуть належати постачальники, партнери, сервісні організації або інші підприємства, з якими здійснюється співпраця. Інформаційна система повинна дозволяти зберігати дані про контрагентів, умови співпраці, поставки, платежі та історію взаємодії. Це спрощує контроль ділових зв'язків і допомагає уникати втрати важливої інформації.

Окремим напрямом управління є **аналіз діяльності підприємства**. Для прийняття обґрунтованих рішень керівнику потрібно мати доступ до звітів про продажі, прибуток, популярні товари або послуги, активність клієнтів, стан складу та фінансові результати. Аналіз дозволяє визначити сильні й слабкі сторони бізнесу, оцінити ефективність окремих процесів і своєчасно вносити зміни в діяльність підприємства.

У межах розроблюваної інформаційної системи основні процеси управління малим бізнесом можна подати як взаємопов'язану структуру, де кожен модуль виконує окрему функцію, але всі дані зберігаються в єдиній базі. Наприклад, замовлення пов'язане з клієнтом, товарами, складськими залишками та фінансовими операціями. Завдяки цьому система дозволяє не лише вести облік, а й моделювати взаємозв'язки між процесами управління.

Таблиця 1.1 – Основні процеси управління малим бізнесом

Процес управління	Зміст процесу	Роль в інформаційній системі
Планування діяльності	Визначення цілей, ресурсів, обсягів продажів і витрат	Формування основи для управлінських рішень
Управління клієнтами	Облік контактів, історії замовлень і взаємодії	Підтримка якісного обслуговування клієнтів
Управління	Ведення номенклатури, цін	Забезпечення

Процес управління	Зміст процесу	Роль в інформаційній системі
товарами послугами	і і характеристик	актуальності даних для продажів
Управління замовленнями	Створення, обробка та контроль виконання замовлень	Автоматизація продажів і взаємодії з клієнтами
Складський облік	Контроль залишків, надходжень і списань	Запобігання нестачі або надлишку товарів
Фінансовий облік	Облік доходів, витрат, оплат і прибутку	Оцінка фінансового стану підприємства
Робота контрагентами	з Облік постачальників і партнерів	Контроль зовнішніх зв'язків підприємства
Аналіз і звітність	Формування звітів візуалізація показників	і Підтримка прийняття управлінських рішень

1.3. Роль інформаційних систем в управлінні підприємством

Інформаційні системи відіграють важливу роль в управлінні сучасним підприємством, оскільки забезпечують збирання, зберігання, обробку, аналіз і передавання даних, необхідних для прийняття управлінських рішень. У діяльності малого бізнесу інформація є одним із головних ресурсів, адже саме на її основі керівник оцінює стан продажів, контролює залишки товарів, аналізує фінансові показники, планує закупівлі та визначає подальші напрями розвитку підприємства.

Інформаційна система підприємства — це сукупність програмних, технічних, інформаційних і організаційних засобів, які використовуються для автоматизації управлінських процесів. Її основне призначення полягає в тому, щоб спростити роботу з даними, зменшити кількість ручних операцій і забезпечити користувача актуальною та достовірною інформацією. Для малого бізнесу це особливо важливо, оскільки обмежена кількість працівників не завжди дозволяє ефективно виконувати всі управлінські функції вручну.

Однією з ключових переваг інформаційних систем є **централізація даних**. Якщо підприємство використовує окремі файли, таблиці або паперові журнали, інформація часто дублюється, втрачається або стає неактуальною. Наприклад, дані про клієнта можуть зберігатися в одному файлі, інформація про замовлення — в іншому, а оплата — у третьому. У такій ситуації складно швидко отримати повну картину взаємодії з клієнтом. Інформаційна система дозволяє зберігати всі пов'язані дані в єдиній базі, що спрощує пошук, обробку та аналіз інформації.

Інформаційні системи також забезпечують **контроль діяльності підприємства**. Керівник може швидко отримати інформацію про поточні замовлення, стан складу, доходи, витрати, заборгованості клієнтів або результати продажів за певний період. Завдяки цьому управлінські рішення приймаються не на основі приблизних оцінок, а на основі конкретних даних. Це підвищує точність планування та дозволяє своєчасно реагувати на проблеми.

Ще однією важливою роллю інформаційних систем є **підтримка аналізу та звітності**. Для малого бізнесу важливо не лише зберігати інформацію, а й бачити результати діяльності у зручному вигляді. Наприклад, система може показувати, які товари продаються найкраще, які клієнти є найбільш активними, які послуги приносять найбільший дохід, які витрати є найбільш значними. Такі звіти допомагають керівнику оцінити ефективність роботи підприємства та прийняти рішення щодо оптимізації бізнес-процесів.

Особливе значення має використання інформаційних систем для **моделювання бізнес-процесів**. Моделювання дозволяє описати послідовність дій, взаємозв'язки між процесами, учасників, інформаційні потоки та результати виконання операцій. Наприклад, процес оформлення замовлення може включати вибір клієнта, додавання товарів або послуг, перевірку наявності на складі, розрахунок вартості, фіксацію оплати та зміну статусу замовлення. Якщо такий процес формалізований у системі, його легше контролювати, аналізувати та покращувати.

Для малого бізнесу інформаційна система повинна бути не лише функціональною, а й зручною у використанні. Надмірно складні корпоративні рішення можуть бути недоцільними для невеликих підприємств, оскільки вони потребують значних витрат на впровадження, навчання персоналу та технічну підтримку. Тому для малого бізнесу важливо розробляти системи, які мають простий інтерфейс, зрозумілу

структуру, базові модулі обліку та можливість швидкого отримання звітної інформації.

У межах даної бакалаврської роботи інформаційна система розглядається як інструмент, що поєднує функції обліку, управління, моделювання та аналізу. Вона повинна забезпечувати роботу з клієнтами, товарами, послугами, контрагентами, замовленнями, складами, фінансовими операціями та звітністю. Завдяки цьому користувач отримує єдине середовище для управління основними процесами малого бізнесу.

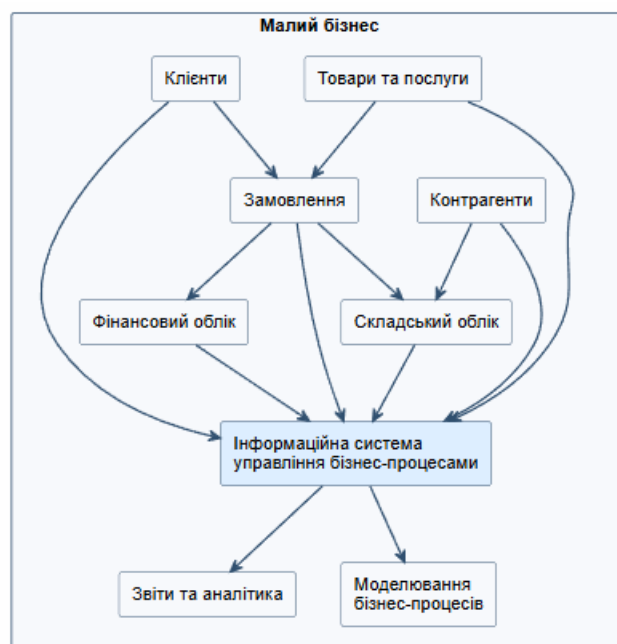


Рисунок 1.1 – Роль інформаційної системи в управлінні малим бізнесом

На цьому рисунку можна показати схему, де в центрі знаходиться **інформаційна система**, а навколо неї розміщені основні процеси: клієнти, товари й послуги, замовлення, склад, фінанси, контрагенти, звіти та аналітика. Стрілками можна показати, що всі ці процеси обмінюються даними через єдину систему.

Отже, інформаційні системи є важливим інструментом підвищення ефективності управління підприємством. Вони дозволяють об'єднати дані в єдиній базі, автоматизувати рутинні операції, покращити контроль, забезпечити швидке формування звітів і створити основу для моделювання бізнес-процесів. Для малого бізнесу це особливо актуально, оскільки правильно підібрана інформаційна система допомагає зменшити витрати часу, уникнути помилок і підвищити якість управлінських рішень.

1.4. Аналіз існуючих інформаційних систем для малого бізнесу

На сучасному ринку програмного забезпечення існує велика кількість інформаційних систем, призначених для автоматизації діяльності підприємств. Такі системи можуть використовуватися для обліку товарів, управління продажами, ведення клієнтської бази, контролю фінансових операцій, формування звітності та організації внутрішніх бізнес-процесів. Для малого бізнесу найбільш актуальними є рішення, які поєднують простоту використання, доступну вартість, базову функціональність і можливість швидкого впровадження.

До поширених типів інформаційних систем для малого бізнесу можна віднести CRM-системи, ERP-системи, складські програми, системи фінансового обліку, онлайн-сервіси для продажів, а також спеціалізовані вебзастосунки для управління замовленнями. Кожен із цих типів має свої переваги та обмеження. Наприклад, CRM-системи добре підходять для роботи з клієнтами, але не завжди забезпечують повноцінний складський або фінансовий облік. ERP-системи мають широку функціональність, проте часто є складними та дорогими для невеликого підприємства.

Для невеликих магазинів і сервісних підприємств часто використовуються прості складські або облікові програми. Вони дозволяють вести номенклатуру товарів, контролювати залишки, оформлювати продажі, фіксувати надходження та списання. Такі програми є корисними для підприємств, які працюють переважно з товарами. Однак їх недоліком може бути обмежена підтримка клієнтської бази, моделювання бізнес-процесів або розширеної аналітики.

Окрему групу становлять хмарні сервіси для малого бізнесу. Їх перевагою є доступ через браузер, відсутність необхідності встановлення складного програмного забезпечення, можливість роботи з різних пристроїв і регулярне оновлення системи. Водночас такі сервіси можуть залежати від стабільності інтернет-з'єднання, мати щомісячну оплату та обмеження в налаштуванні під конкретні потреби підприємства.

Також у малому бізнесі досить часто використовуються електронні таблиці, наприклад Microsoft Excel або Google Sheets. Вони є простими, доступними та зрозумілими для користувачів. За їх допомогою можна вести списки клієнтів, товари, замовлення та фінансові розрахунки. Проте електронні таблиці не є повноцінною інформаційною системою. Вони не забезпечують

належного контролю цілісності даних, прав доступу, автоматичного зв'язку між сутностями та зручного масштабування.

Для порівняння існуючих рішень доцільно розглянути їх за кількома критеріями: функціональність, зручність використання, вартість, можливість адаптації, підтримка звітності та придатність для малого бізнесу.

Таблиця 1.2 – Порівняння існуючих інформаційних систем для малого бізнесу

Тип системи	Основне призначення	Переваги	Недоліки
CRM-система	Управління клієнтами та продажами	Зручна робота з клієнтами, історія взаємодії, контроль заявок	Обмежений складський і фінансовий облік
ERP-система	Комплексне управління підприємством	Широка функціональність, єдина база даних, аналітика	Висока складність, значна вартість, тривале впровадження
Складська система	Облік товарів, залишків і руху продукції	Зручний контроль складу, швидке оформлення операцій	Обмежена робота з клієнтами та бізнес-процесами
Фінансова система	Облік доходів, витрат і платежів	Контроль фінансових показників, звітність	Не охоплює всі управлінські процеси
Хмарний сервіс	Онлайн-управління бізнесом	Доступ з різних пристроїв, швидке впровадження	Залежність від інтернету, регулярна оплата
Електронні	Простий облік	Доступність, простота,	Високий ризик

Тип системи	Основне призначення	Переваги	Недоліки
таблиці	даних	гнучкість	помилки, відсутність повноцінної автоматизації

Аналіз існуючих інформаційних систем показує, що для малого бізнесу важливо знайти баланс між функціональністю та простотою використання. Занадто складні системи можуть бути недоцільними через високу вартість і складність впровадження, а надто прості інструменти не забезпечують належного рівня автоматизації. Саме тому актуальним є створення інформаційної системи, яка буде орієнтована на базові потреби малого підприємства та матиме зрозумілу структуру.

1.5. Необхідність моделювання бізнес-процесів у малому бізнесі

Моделювання бізнес-процесів є важливим етапом удосконалення управління підприємством, оскільки дозволяє описати діяльність організації у вигляді послідовності взаємопов'язаних дій. Для малого бізнесу це особливо актуально, адже багато процесів часто виконуються без чіткої формалізації: частина інформації зберігається в пам'яті працівників, частина — у таблицях або паперових документах, а порядок виконання операцій може залежати від конкретної ситуації. У результаті виникають помилки, затримки та складність контролю.

Бізнес-процес можна розглядати як послідовність дій, спрямованих на досягнення певного результату. Наприклад, процес продажу товару включає отримання звернення від клієнта, вибір товару, перевірку його наявності, створення замовлення, розрахунок вартості, оформлення оплати та оновлення складських залишків. Якщо такий процес не описаний і не автоматизований, працівник може пропустити окремі етапи або внести дані з помилками.

Моделювання також дозволяє виявити дублювання операцій. Наприклад, дані про клієнта можуть вводитися окремо в таблицю замовлень, у файл фінансового обліку та в список контактів. Це збільшує час роботи й підвищує ризик розбіжностей у даних. Якщо процеси описані у вигляді моделі, можна визначити, де саме виникає дублювання, і передбачити єдину базу даних для зберігання інформації.

Ще однією перевагою моделювання є можливість покращити контроль виконання операцій. Якщо в системі визначені етапи роботи із замовленням, користувач може бачити його поточний статус: створене, в обробці, виконане, оплачене або скасоване. Це спрощує контроль і дозволяє швидко визначати, на якому етапі знаходиться конкретне замовлення. Для малого бізнесу це важливо, оскільки навіть невелика кількість пропущених або неправильно оброблених замовлень може призвести до втрати клієнтів.

Моделювання бізнес-процесів також сприяє підвищенню якості управлінських рішень. Коли керівник бачить логіку виконання процесів і має доступ до актуальних даних, він може швидше визначати проблеми та приймати обґрунтовані рішення. Наприклад, якщо звіти показують, що певний товар часто закінчується на складі, можна змінити графік закупівель. Якщо частина замовлень довго залишається в обробці, можна проаналізувати причину затримки та оптимізувати відповідний процес.

У межах інформаційної системи моделювання бізнес-процесів може бути реалізоване через визначення основних сутностей, зв'язків між ними та послідовності виконання дій. Наприклад, клієнт створює або оформлює замовлення, замовлення містить товари або послуги, товари пов'язані зі складом, а оплата пов'язана з фінансовим обліком. Така структура дозволяє не лише зберігати дані, а й відображати реальну логіку роботи підприємства.



Рисунок 1.2 – Узагальнена модель бізнес-процесу обробки замовлення

Моделювання бізнес-процесів є основою для подальшого проектування інформаційної системи. Без попереднього аналізу процесів складно правильно визначити функціональні вимоги, структуру бази даних, модулі програми та логіку роботи інтерфейсу. Тому перед розробкою програмної реалізації необхідно визначити, які саме процеси повинна підтримувати система та які дані мають оброблятися в межах кожного процесу.

1.6. Вибір мови програмування, середовища розробки та технологій

Для реалізації серверної частини системи доцільно використати мову програмування **Python**. Вона є поширеною, зрозумілою та зручною для розробки інформаційних систем, вебзастосунків, обробки даних і формування звітності. Python має велику кількість бібліотек, активну спільноту та простий синтаксис, що дає змогу швидше реалізовувати програмну логіку. Для бакалаврської роботи ця мова є доречною, оскільки дозволяє не лише створити програмну систему, а й зрозуміло описати її структуру та принцип роботи.

Як вебфреймворк можна використати **Flask**. Flask є легким фреймворком для створення вебзастосунків мовою Python. Його перевагою є простота структури, можливість швидкого створення маршрутів, обробки запитів, роботи з шаблонами сторінок і підключення бази даних. На відміну від складніших фреймворків, Flask не перевантажує проєкт зайвими компонентами, що робить його зручним для створення навчальної або бакалаврської програмної реалізації.

Для зберігання даних інформаційної системи може бути використана реляційна база даних **SQLite** або **PostgreSQL**. SQLite є простішим варіантом, оскільки не потребує окремого серверного налаштування та зберігає дані у файлі бази даних. Це зручно для демонстраційної програмної реалізації бакалаврської роботи. PostgreSQL є більш потужною системою керування базами даних і краще підходить для реального використання в бізнесі. У межах даної роботи доцільно використати SQLite для спрощення запуску системи, а в описі зазначити можливість подальшого переходу на PostgreSQL.

Для створення інтерфейсу користувача доцільно використати стандартні вебтехнології: **HTML**, **CSS** та **JavaScript**. HTML забезпечує структуру вебсторінок, CSS відповідає за зовнішній вигляд, а JavaScript може використовуватися для покращення взаємодії користувача із системою.

Наприклад, за допомогою JavaScript можна реалізувати перевірку введених даних, динамічне оновлення елементів сторінки або побудову простих графіків.

Для візуалізації результатів роботи системи можна використати бібліотеку **Chart.js**. Вона дозволяє створювати діаграми та графіки на вебсторінках, що є корисним для модулю звітності. Наприклад, у системі можна відображати графік доходів і витрат, діаграму популярності товарів або послуг, кількість замовлень за період, структуру продажів тощо. Візуалізація даних робить результати аналізу більш зрозумілими для користувача.

Для організації структури програмного коду доцільно використати модульний підхід. Кожен функціональний блок системи може бути реалізований як окремий модуль або набір маршрутів: клієнти, контрагенти, товари і послуги, склади, замовлення, фінанси, звіти та моделювання бізнес-процесів. Такий підхід спрощує розробку, тестування і подальше розширення системи.

Середовищем розробки може бути **Visual Studio Code**, оскільки воно підтримує Python, HTML, CSS, JavaScript, роботу з терміналом, розширення для форматування коду та інтеграцію з системами контролю версій. Це середовище є зручним для створення вебзастосунків і дозволяє працювати з усіма файлами проєкту в одному інтерфейсі.

Таблиця 1.3 – Обрані технології розробки інформаційної системи

Технологія	Призначення	Обґрунтування вибору
Python	Реалізація серверної логіки	Простий синтаксис, велика кількість бібліотек, зручність для веброзробки
Flask	Створення вебзастосунку	Легкий фреймворк, проста структура, швидка реалізація маршрутів
SQLite	Зберігання даних	Просте налаштування, зручність для демонстраційної системи
HTML	Структура вебсторінок	Базова технологія створення вебінтерфейсу
CSS	Оформлення	Дає змогу створити зручний і

Технологія	Призначення	Обґрунтування вибору
	інтерфейсу	зрозумілий вигляд системи
JavaScript	Динамічна взаємодія з користувачем	Дозволяє покращити роботу інтерфейсу
Chart.js	Побудова графіків і діаграм	Зручна візуалізація результатів роботи системи
Visual Studio Code	Середовище розробки	Підтримка Python і вебтехнологій, зручна робота з проєктом
Git	Контроль версій	Збереження історії змін і впорядкування розробки

Таким чином, для розробки інформаційної системи моделювання процесів управління малим бізнесом обрано набір технологій, який поєднує простоту, доступність і достатню функціональність.

1.7. Висновки до розділу 1

У першому розділі бакалаврської роботи було розглянуто предметну область малого бізнесу та визначено основні особливості управління невеликим підприємством. Встановлено, що малий бізнес характеризується гнучкістю, швидким прийняттям рішень, тісною взаємодією з клієнтами та обмеженими ресурсами. Саме через обмеженість фінансових, кадрових і технічних можливостей для таких підприємств особливо важливо використовувати прості, доступні та ефективні інструменти автоматизації.

РОЗДІЛ 2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ МОДЕЛЮВАННЯ ПРОЦЕСІВ УПРАВЛІННЯ

2.1. Постановка задачі розробки інформаційної системи

Після аналізу предметної області малого бізнесу та визначення основних управлінських процесів виникає необхідність у розробці інформаційної системи, яка дозволить автоматизувати облік, контроль, аналіз і моделювання діяльності малого підприємства. Основна проблема, яку має вирішувати така система, полягає у відсутності єдиного програмного середовища для роботи з клієнтами, товарами, послугами, замовленнями, складськими залишками, фінансовими операціями та звітністю.

У багатьох малих підприємствах інформація зберігається у різних джерелах: електронних таблицях, текстових документах, паперових журналах або окремих файлах. Через це виникають труднощі з пошуком актуальних даних, контролем виконання замовлень, аналізом фінансових показників і формуванням звітності. Крім того, ручне ведення обліку збільшує ризик помилок, дублювання інформації та втрати важливих записів.

Задача розробки інформаційної системи полягає у створенні вебзастосунку, який забезпечує централізоване зберігання даних і підтримує основні процеси управління малим бізнесом. Система повинна бути зручною для користувача, мати зрозумілий інтерфейс і не вимагати складного налаштування. Основним користувачем системи може бути власник малого підприємства, адміністратор, менеджер або інший працівник, який відповідає за облік і управління поточними операціями.

Розроблювана інформаційна система повинна забезпечувати виконання таких основних задач:

1. ведення бази клієнтів;
2. облік контрагентів і постачальників;
3. ведення номенклатури товарів і послуг;
4. управління складськими залишками;
5. створення та обробка замовлень;

6. облік продажів і оплат;
7. ведення доходів і витрат;
8. моделювання основних бізнес-процесів;
9. формування звітів;
10. візуалізація результатів діяльності підприємства.

Особливу увагу в системі необхідно приділити зв'язку між окремими процесами. Наприклад, під час створення замовлення користувач повинен мати можливість вибрати клієнта, додати товари або послуги, перевірити їх наявність, розрахувати загальну суму, зафіксувати оплату та оновити складські залишки. Таким чином, система повинна не просто зберігати окремі записи, а відображати реальну логіку роботи малого підприємства.

Загальна постановка задачі передбачає створення інформаційної системи, яка складається з кількох функціональних модулів. Кожен модуль відповідає за окремий напрям діяльності підприємства, але всі модулі повинні працювати з єдиною базою даних. Це забезпечує цілісність інформації та дозволяє уникнути дублювання записів.

Основними модулями системи є:

- модуль управління клієнтами;
- модуль управління контрагентами;
- модуль обліку товарів і послуг;
- модуль складського обліку;
- модуль управління замовленнями;
- модуль фінансового обліку;
- модуль моделювання бізнес-процесів;
- модуль звітності та аналітики.

Очікуваним результатом розробки є програмна система, яка дозволяє користувачу виконувати основні операції управління малим бізнесом у єдиному середовищі. Вона повинна забезпечувати зручне введення, редагування, перегляд і видалення даних, автоматичне формування взаємозв'язків між процесами та побудову звітів для аналізу діяльності підприємства.

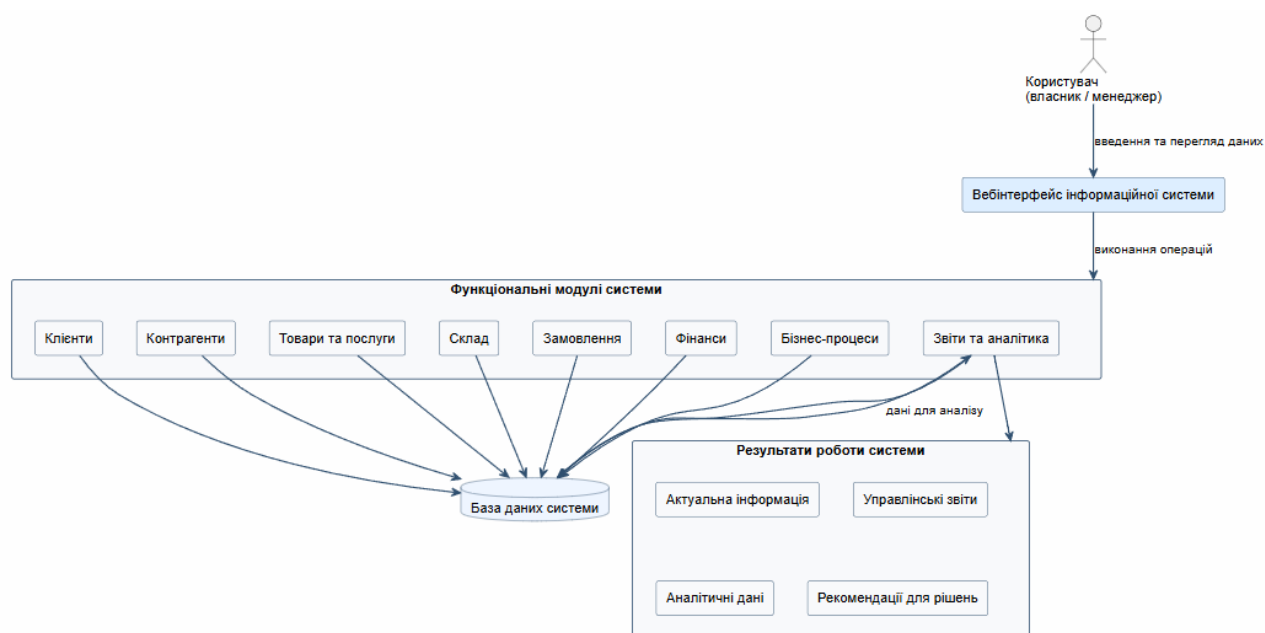


Рисунок 2.1 – Загальна схема задачі розробки інформаційної системи

На рисунку 2.1 подано загальну схему задачі розробки інформаційної системи. Користувач взаємодіє із системою через вебінтерфейс, за допомогою якого здійснюється доступ до основних функціональних модулів: клієнтів, контрагентів, товарів і послуг, складу, замовлень, фінансів, бізнес-процесів, звітів та аналітики.

2.2. Визначення функціональних вимог до системи

Функціональні вимоги визначають, які саме дії повинна виконувати інформаційна система та які можливості мають бути доступні користувачу. Для інформаційної системи моделювання процесів управління малим бізнесом функціональні вимоги формуються на основі основних процесів діяльності підприємства: роботи з клієнтами, контрагентами, товарами, послугами, замовленнями, складом, фінансами, бізнес-процесами та звітністю.

Розроблювана система повинна забезпечувати користувачу можливість виконувати повний цикл управління основними даними підприємства. Це означає, що користувач має мати змогу додавати нові записи, переглядати наявну інформацію, редагувати її, видаляти застарілі або помилкові записи, а також виконувати пошук і фільтрацію даних. Такий підхід дозволяє організувати зручну роботу з інформацією та зменшити залежність підприємства від паперових документів або окремих електронних таблиць.

Окремий функціональний блок повинен забезпечувати **облік товарів і послуг**. Користувач повинен мати можливість додавати нові позиції, вказувати назву, категорію, ціну, одиницю виміру, опис і статус доступності. Для товарів також доцільно передбачити зв'язок зі складськими залишками, а для послуг — можливість використання їх у замовленнях без складського списання.

Важливою функцією системи є **управління складом**. Система повинна дозволяти обліковувати залишки товарів, фіксувати надходження, списання та переміщення. Під час оформлення замовлення система має перевіряти наявність товару на складі та за потреби оновлювати залишки після продажу. Це дозволяє уникати ситуацій, коли товар продається, але фактично відсутній.

Центральним модулем системи є **управління замовленнями**. Користувач повинен мати можливість створити нове замовлення, вибрати клієнта, додати товари або послуги, вказати кількість, розрахувати загальну суму, встановити статус замовлення та зафіксувати оплату. Замовлення повинні мати статуси, наприклад: «нове», «в обробці», «виконане», «оплачене» або «скасоване». Це спрощує контроль за поточними операціями підприємства.

Також система повинна містити **модуль фінансового обліку**. Він має забезпечувати фіксацію доходів, витрат, оплат від клієнтів, платежів постачальникам і загального фінансового результату. Завдяки цьому користувач зможе бачити поточний фінансовий стан підприємства, контролювати прибуток і аналізувати витрати за певний період.

Окремою вимогою є **моделювання бізнес-процесів**. Система повинна дозволяти відображати основні процеси управління як послідовність пов'язаних дій. Наприклад, процес обробки замовлення може включати етапи отримання заявки, вибору клієнта, додавання товарів, перевірки складу, оформлення замовлення, фіксації оплати та формування звіту. Це дозволяє користувачу краще розуміти логіку роботи підприємства.

Важливим функціональним елементом є **формування звітів та аналітики**. Система повинна надавати користувачу можливість переглядати звіти про продажі, доходи, витрати, прибуток, кількість замовлень, популярні товари або послуги, активність клієнтів і стан складських залишків. Для кращого сприйняття інформації доцільно передбачити візуалізацію даних у вигляді графіків або діаграм.

Функціональні вимоги до системи можна подати у вигляді таблиці.

Таблиця 2.1 – Функціональні вимоги до інформаційної системи

№	Функціональна вимога	Опис
1	Управління клієнтами	Додавання, перегляд, редагування, видалення та пошук клієнтів
2	Управління контрагентами	Облік постачальників, партнерів та історії співпраці
3	Облік товарів і послуг	Ведення номенклатури, цін, описів і категорій
4	Складський облік	Контроль залишків, надходжень, списань і переміщень
5	Управління замовленнями	Створення замовлень, вибір клієнта, додавання позицій, зміна статусу
6	Облік оплат	Фіксація оплат за замовленнями та контроль заборгованостей
7	Фінансовий облік	Облік доходів, витрат і прибутку підприємства
8	Моделювання бізнес-процесів	Відображення основних процесів як послідовності дій
9	Формування звітів	Побудова звітів за продажами, фінансами, клієнтами та складом
10	Візуалізація даних	Подання результатів у вигляді графіків і діаграм
11	Пошук і фільтрація	Швидкий пошук записів за ключовими параметрами
12	Перегляд історії операцій	Збереження інформації про виконані дії та зміни статусів

Таким чином, функціональні вимоги до інформаційної системи охоплюють усі основні напрями управління малим бізнесом. Їх реалізація дозволить створити програмний продукт, який забезпечує централізовану

роботу з даними, автоматизацію щоденних операцій, контроль за діяльністю підприємства та підтримку прийняття управлінських рішень.

2.3. Визначення нефункціональних вимог до системи

Нефункціональні вимоги визначають не окремі функції інформаційної системи, а загальні характеристики її роботи. Вони описують, якою повинна бути система з погляду зручності використання, швидкодії, надійності, безпеки, сумісності, масштабованості та можливості подальшого розвитку. Для інформаційної системи моделювання процесів управління малим бізнесом нефункціональні вимоги мають важливе значення, оскільки система повинна бути не лише функціональною, а й зручною для повсякденного використання.

Однією з основних нефункціональних вимог є **зручність інтерфейсу користувача**. Оскільки система орієнтована на малий бізнес, її користувачами можуть бути люди без спеціальної технічної підготовки. Тому інтерфейс повинен бути простим, зрозумілим і логічно структурованим. Основні розділи системи мають бути доступні з головного меню, а дії з додавання, редагування, перегляду та видалення записів повинні виконуватися без зайвих складностей.

Важливою вимогою є **швидкодія системи**. Операції перегляду списків, пошуку клієнтів, створення замовлень, оновлення складських залишків і формування звітів повинні виконуватися за короткий час. Для малого підприємства обсяг даних зазвичай не є надто великим, однак система все одно повинна бути оптимізованою для стабільної роботи з базою даних і швидкого відображення результатів у вебінтерфейсі.

Наступною вимогою є **надійність зберігання даних**. Інформація про клієнтів, замовлення, товари, фінанси та складські залишки є важливою для діяльності підприємства, тому система повинна забезпечувати коректне збереження записів у базі даних. Під час виконання операцій необхідно уникати втрати або пошкодження інформації. Для цього в системі мають використовуватися перевірка введених даних, обмеження цілісності бази даних і логічні зв'язки між таблицями.

Таблиця 2.2 – Нефункціональні вимоги до інформаційної системи

№	Нефункціональна вимога	Опис
1	Зручність використання	Інтерфейс повинен бути простим, зрозумілим і логічно структурованим
2	Швидкодія	Основні операції повинні виконуватися без значних затримок
3	Надійність	Дані повинні коректно зберігатися та оброблятися
4	Безпека	Необхідно передбачити захист даних і базову авторизацію
5	Масштабованість	Система повинна мати можливість подальшого розширення
6	Супроводжуваність	Код повинен бути структурованим і зрозумілим для доопрацювання
7	Сумісність	Система має працювати в сучасних веббраузерах
8	Цілісність даних	Зв'язки між таблицями повинні запобігати некоректним записам
9	Адаптивність інтерфейсу	Сторінки мають коректно відображатися на різних екранах
10	Зрозумілі повідомлення	Система повинна інформувати користувача про помилки та успішні дії

Таким чином, нефункціональні вимоги визначають якісні характеристики інформаційної системи та впливають на зручність її практичного використання.

2.4. Розробка загальної архітектури інформаційної системи

Архітектура інформаційної системи визначає загальну структуру програмного продукту, взаємодію його основних компонентів, принципи обміну даними та логіку роботи користувача із системою. Для

інформаційної системи моделювання процесів управління малим бізнесом доцільно використати клієнт-серверну архітектуру, оскільки вона добре підходить для вебзастосунків і дозволяє розділити інтерфейс користувача, бізнес-логіку та роботу з базою даних.

Розроблювана система має складатися з трьох основних рівнів: рівня користувацького інтерфейсу, рівня серверної логіки та рівня зберігання даних. Такий підхід дозволяє впорядкувати структуру застосунку, полегшити його розробку, тестування та подальше розширення.

Першим рівнем є **рівень користувацького інтерфейсу**. Він відповідає за взаємодію користувача із системою через веббраузер. На цьому рівні розміщуються сторінки для перегляду списків клієнтів, товарів, послуг, замовлень, складів, фінансових операцій і звітів. Користувач може вводити нові дані, редагувати наявні записи, виконувати пошук, застосовувати фільтри та переглядати результати роботи системи.

Другим рівнем є **рівень серверної логіки**. Він реалізується за допомогою мови програмування Python та фреймворку Flask. Серверна частина приймає запити від користувача, обробляє їх, виконує перевірку даних, звертається до бази даних і повертає результат у вигляді вебсторінок або повідомлень. Саме на цьому рівні реалізуються основні правила роботи системи: створення замовлень, перевірка складських залишків, розрахунок суми замовлення, фіксація оплат, формування звітів і моделювання бізнес-процесів.

Третім рівнем є **рівень зберігання даних**. Він представлений базою даних SQLite, у якій зберігається вся основна інформація системи: дані про клієнтів, контрагентів, товари, послуги, склади, замовлення, фінансові операції та звіти. Використання єдиної бази даних дозволяє забезпечити цілісність інформації та зв'язок між різними модулями системи.

Загальна архітектура системи передбачає, що користувач надсилає запит через вебінтерфейс. Серверна частина обробляє цей запит, звертається до відповідного модуля та, за потреби, виконує операції з базою даних. Після цього система повертає користувачу результат: оновлену сторінку, список записів, повідомлення про успішну дію, звіт або графік.

Основними компонентами архітектури інформаційної системи є:

- вебінтерфейс користувача;
- серверна частина застосунку;

- модуль роботи з клієнтами;
- модуль роботи з контрагентами;
- модуль обліку товарів і послуг;
- модуль складського обліку;
- модуль управління замовленнями;
- модуль фінансового обліку;
- модуль моделювання бізнес-процесів;
- модуль звітності та аналітики;
- база даних системи.

Кожен функціональний модуль виконує окрему задачу, однак усі вони взаємодіють через спільну базу даних. Наприклад, модуль замовлень використовує дані з модуля клієнтів, товарів, складу та фінансів. Під час створення замовлення система отримує інформацію про клієнта, перевіряє наявність товарів на складі, розраховує загальну суму та після оплати створює фінансову операцію. Така взаємодія забезпечує цілісну логіку роботи системи.

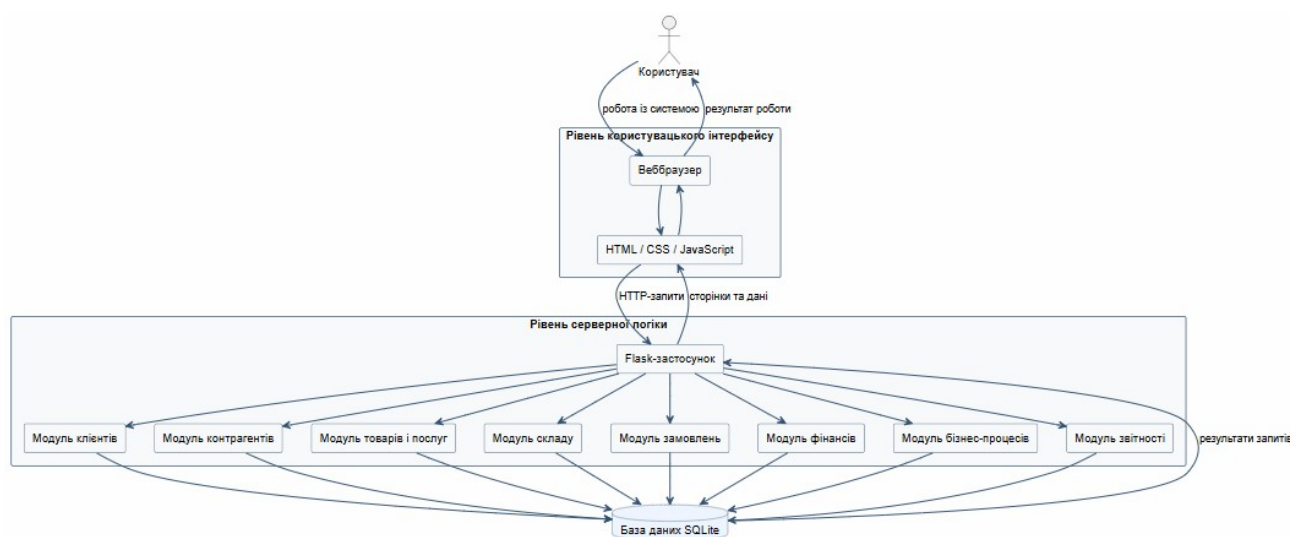


Рисунок 2.2 – Загальна архітектура інформаційної системи

На рисунку 2.2 подано загальну архітектуру інформаційної системи моделювання процесів управління малим бізнесом. Система побудована за клієнт-серверним принципом. Користувач працює із системою через веббраузер, у якому відображається інтерфейс, створений за допомогою HTML, CSS та JavaScript.

2.5. Розробка діаграми варіантів використання інформаційної системи

Діаграма варіантів використання є одним із важливих інструментів проєктування інформаційної системи. Вона дозволяє показати, які користувачі взаємодіють із системою та які основні функції вони можуть виконувати. Така діаграма допомагає визначити межі системи, уточнити її функціональні можливості та краще зрозуміти логіку взаємодії користувача з програмним продуктом.

У межах розроблюваної інформаційної системи основним користувачем є працівник малого підприємства. Це може бути власник бізнесу, адміністратор, менеджер або відповідальна особа, яка виконує щоденні операції з обліку та управління. Користувач взаємодіє з вебінтерфейсом системи та має доступ до основних модулів: клієнтів, контрагентів, товарів і послуг, складу, замовлень, фінансів, бізнес-процесів і звітності.

Основними варіантами використання інформаційної системи є:

- авторизація в системі;
- управління клієнтами;
- управління контрагентами;
- управління товарами та послугами;
- перегляд складських залишків;
- оформлення замовлень;
- зміна статусу замовлень;
- фіксація оплат;
- облік доходів і витрат;
- моделювання бізнес-процесів;
- формування звітів;
- перегляд аналітичних показників.

Центральним сценарієм роботи системи є оформлення замовлення. Для виконання цього сценарію користувач обирає клієнта, додає товари або послуги, перевіряє їх наявність, формує замовлення, фіксує оплату та отримує оновлені дані про склад і фінанси. Саме цей сценарій поєднує

більшість модулів системи та відображає її практичну цінність для малого бізнесу.

Діаграма варіантів використання також дозволяє показати, що окремі дії можуть бути пов'язані між собою. Наприклад, оформлення замовлення включає вибір клієнта, додавання товарів або послуг, перевірку залишків і розрахунок суми. Формування звіту може включати отримання даних про продажі, фінансові операції та складські залишки. Такий підхід допомагає не лише описати функціональність системи, а й показати взаємозв'язок між її процесами.

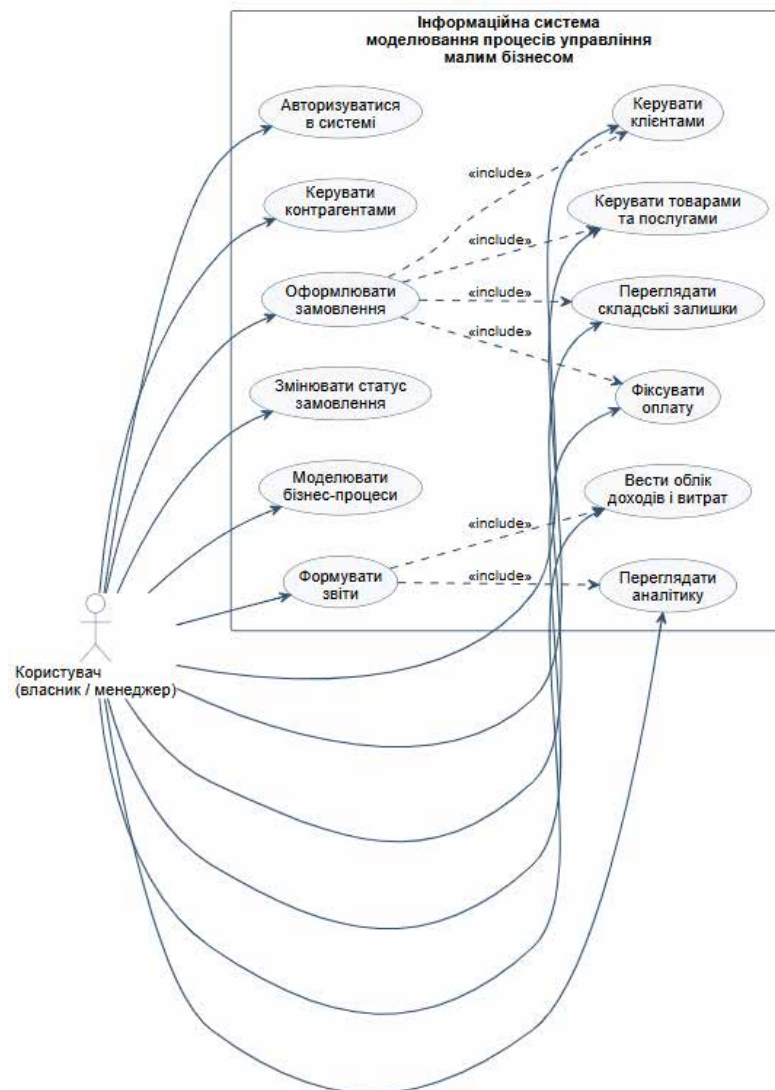


Рисунок 2.3 – Діаграма варіантів використання інформаційної системи

2.6. Розробка діаграми діяльності основних бізнес-процесів

Діаграма діяльності використовується для опису послідовності дій, які виконуються в межах певного бізнес-процесу. Вона дозволяє показати логіку переходу між етапами, умови прийняття рішень, початок і завершення процесу. Для інформаційної системи моделювання процесів управління малим бізнесом така діаграма є важливою, оскільки вона демонструє, як саме користувач працює із системою під час виконання основних операцій.

Одним із ключових бізнес-процесів у малому бізнесі є процес обробки замовлення. Він поєднує роботу з клієнтами, товарами або послугами, складом, оплатою та звітністю. Саме тому цей процес доцільно використати як основу для діаграми діяльності. Його моделювання дозволяє показати, як інформаційна система підтримує повний цикл роботи із замовленням: від створення заявки до оновлення даних і формування результату.

Процес обробки замовлення починається з авторизації користувача в системі. Після входу користувач переходить до модуля замовлень і створює нове замовлення. Далі він обирає клієнта з бази або додає нового клієнта, якщо потрібна інформація ще не збережена в системі. Після цього до замовлення додаються товари або послуги, вказується їх кількість і формується попередня сума.

Наступним етапом є перевірка наявності товарів на складі. Якщо товар є в достатній кількості, система дозволяє продовжити оформлення замовлення. Якщо залишків недостатньо, користувач отримує повідомлення про неможливість виконання замовлення в повному обсязі. У такому випадку він може змінити кількість товару, обрати іншу позицію або скасувати замовлення.

Після успішної перевірки складських залишків система розраховує загальну суму замовлення. Далі користувач фіксує оплату або встановлює відповідний статус, наприклад «очікує оплати». Якщо оплата внесена, система створює фінансовий запис, оновлює залишки товарів на складі та змінює статус замовлення. Якщо оплата ще не виконана, замовлення залишається в системі з відповідним статусом для подальшого контролю.

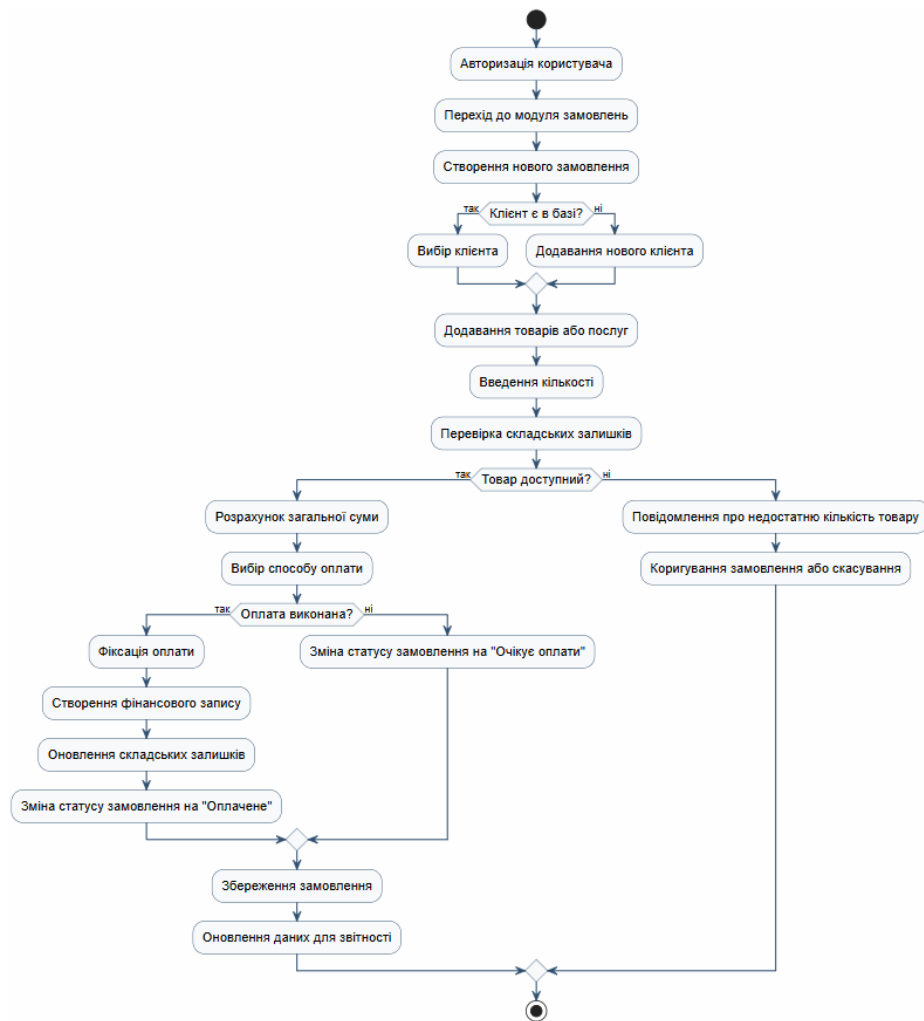


Рисунок 2.4 – Діаграма діяльності процесу обробки замовлення

2.7. Розробка діаграми послідовності роботи користувача із системою

Діаграма послідовності використовується для опису взаємодії між користувачем, інтерфейсом, серверною частиною та базою даних у процесі виконання певної операції. На відміну від діаграми діяльності, яка показує загальну логіку процесу, діаграма послідовності демонструє порядок обміну повідомленнями між компонентами системи.

Для інформаційної системи моделювання процесів управління малим бізнесом доцільно розглянути сценарій створення нового замовлення. Цей сценарій є одним із найважливіших, оскільки він пов'язує між собою клієнтів, товари або послуги, складські залишки, оплату, фінансовий облік і звітність.

У процесі створення замовлення беруть участь такі компоненти:

- користувач;
- вебінтерфейс;
- серверна частина Flask;
- модуль замовлень;
- модуль складу;
- модуль фінансів;
- база даних.

Робота починається з того, що користувач відкриває сторінку створення замовлення у вебінтерфейсі. Інтерфейс надсилає запит до серверної частини, яка отримує з бази даних список клієнтів, товарів і послуг. Після цього користувач заповнює форму замовлення, обирає клієнта, додає потрібні позиції та вказує їх кількість.

Після надсилання форми серверна частина передає дані до модуля замовлень. Модуль замовлень перевіряє правильність введених даних і звертається до модуля складу для перевірки наявності товарів. Якщо товарів достатньо, система створює запис замовлення в базі даних, оновлює складські залишки та, за наявності оплати, створює фінансову операцію. Якщо товарів недостатньо, користувачу повертається повідомлення про помилку.

Далі система оновлює дані для звітності та повертає користувачу повідомлення про успішне створення замовлення. Таким чином, діаграма послідовності дозволяє показати, як саме відбувається обмін даними між основними компонентами системи під час виконання бізнес-операції.

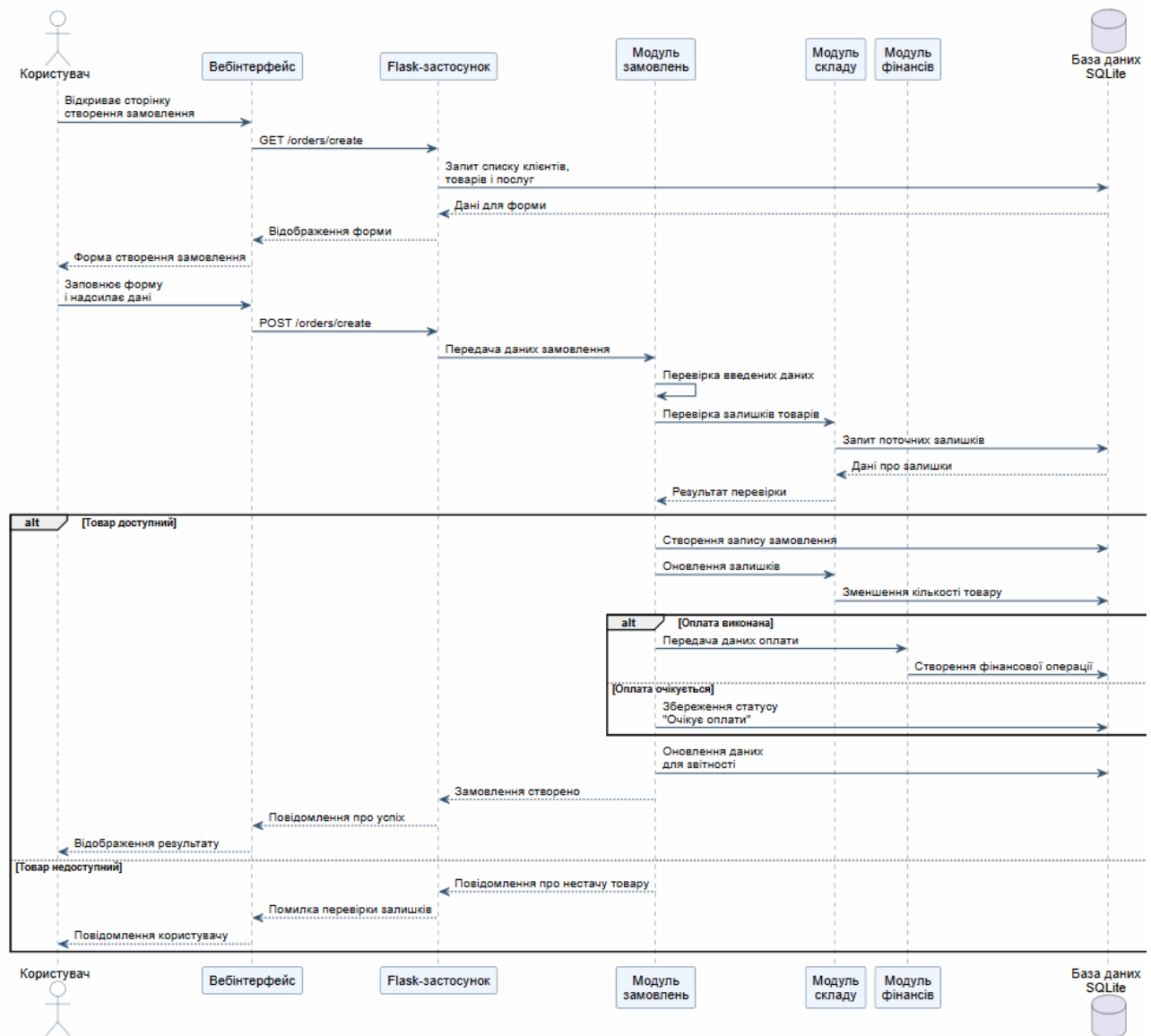


Рисунок 2.5 – Діаграма послідовності створення замовлення в інформаційній системі

2.8. Проектування бази даних інформаційної системи

База даних є одним із ключових компонентів інформаційної системи, оскільки саме в ній зберігається вся основна інформація, необхідна для управління малим бізнесом. Від правильності проектування бази даних залежить стабільність роботи системи, цілісність даних, швидкість пошуку інформації та можливість подальшого розширення програмного продукту.

Для розроблюваної інформаційної системи доцільно використати реляційну модель бази даних. Такий підхід передбачає зберігання інформації у вигляді таблиць, між якими встановлюються логічні зв'язки.

Реляційна модель є зручною для систем, у яких потрібно працювати з клієнтами, товарами, замовленнями, платежами, складами та іншими пов'язаними сутностями.

Основними сутностями бази даних інформаційної системи є:

- користувачі;
- клієнти;
- контрагенти;
- товари та послуги;
- склади;
- складські залишки;
- замовлення;
- позиції замовлення;
- фінансові операції;
- бізнес-процеси;
- звіти.

Сутність **«Користувачі»** призначена для зберігання даних про осіб, які мають доступ до системи. У базовій версії системи може бути реалізований один тип користувача, наприклад адміністратор або менеджер. У подальшому можна додати ролі доступу, щоб розмежувати можливості різних працівників.

Сутність **«Склади»** використовується для зберігання інформації про місця зберігання товарів. Навіть якщо мале підприємство має один склад, виділення цієї таблиці є доцільним, оскільки у майбутньому система може бути розширена для роботи з кількома складами або торговими точками.

Сутність **«Складські залишки»** пов'язує товари зі складами та містить інформацію про кількість конкретного товару на певному складі. Така структура дозволяє контролювати наявність товарів і оновлювати залишки після продажів, надходжень або списань.

Сутність **«Замовлення»** є центральною в базі даних. Вона містить інформацію про клієнта, дату створення, статус замовлення, загальну суму та стан оплати. Одне замовлення може містити кілька товарів або послуг, тому для зберігання складу замовлення використовується окрема таблиця.

Сутність «**Позиції замовлення**» призначена для зберігання окремих товарів або послуг, які входять до конкретного замовлення. У цій таблиці фіксується ідентифікатор замовлення, ідентифікатор товару або послуги, кількість, ціна та сума за позицією. Така структура дозволяє реалізувати зв'язок «одне замовлення — багато позицій».

Сутність «**Фінансові операції**» використовується для обліку доходів і витрат підприємства. Вона може містити тип операції, суму, дату, опис, зв'язок із замовленням або контрагентом. Це дозволяє фіксувати надходження коштів від клієнтів, витрати на закупівлі, оплату послуг та інші фінансові дії.

Сутність «**Бізнес-процеси**» призначена для моделювання основних процесів управління. У ній можуть зберігатися назва процесу, опис, статус, відповідальний користувач, дата створення та пов'язані операції. Це дає змогу формалізувати процеси роботи підприємства та використовувати їх для аналізу.

Для кращого розуміння структури бази даних основні таблиці та їх призначення можна подати у вигляді таблиці.

Таблиця 2.3 – Основні таблиці бази даних інформаційної системи

Назва таблиці	Призначення	Основні поля
users	Зберігання даних користувачів системи	id, username, password, role
clients	Облік клієнтів	id, name, phone, email, address, notes
contractors	Облік контрагентів	id, name, type, phone, email, address
products	Облік товарів і послуг	id, name, category, type, price, unit, description
warehouses	Облік складів	id, name, location, description
stock	Облік складських залишків	id, product_id, warehouse_id, quantity
orders	Облік замовлень	id, client_id, date, status, total_sum, payment_status
order_items	Облік позицій замовлення	id, order_id, product_id, quantity,

Назва таблиці	Призначення	Основні поля
		price, amount
finance_operations	Облік доходів і витрат	id, type, date, amount, description, order_id
business_processes	Опис бізнес-процесів	id, name, description, status, created_at
reports	Збереження параметрів звітів	id, name, type, created_at, description

Таблиця order_items пов'язана з таблицями orders і products, оскільки кожна позиція належить певному замовленню та містить конкретний товар або послугу. Таблиця stock пов'язує товари зі складами, а таблиця finance_operations може бути пов'язана із замовленнями для фіксації оплат.

Основні зв'язки між таблицями можна описати так:

- один клієнт може мати багато замовлень;
- одне замовлення може містити багато позицій;
- один товар або послуга може входити до багатьох позицій замовлень;
- один склад може містити багато складських залишків;
- один товар може бути пов'язаний із багатьма складськими записами;
- одне замовлення може мати одну або кілька фінансових операцій;
- один користувач може створювати або контролювати кілька бізнес-процесів.

2.9. Розробка ER-діаграми бази даних

ER-діаграма бази даних використовується для наочного подання основних сутностей інформаційної системи та зв'язків між ними. Вона дозволяє зрозуміти, які таблиці входять до складу бази даних, які поля вони містять і як саме ці таблиці взаємодіють між собою. Для інформаційної системи моделювання процесів управління малим бізнесом ER-діаграма є важливою частиною проектування, оскільки база даних забезпечує

зберігання всієї інформації про клієнтів, товари, замовлення, склад, фінанси та бізнес-процеси.

Основою ER-діаграми є сутності, що відповідають основним об'єктам предметної області. У межах даної системи такими сутностями є користувачі, клієнти, контрагенти, товари та послуги, склади, складські залишки, замовлення, позиції замовлення, фінансові операції, бізнес-процеси та звіти. Кожна сутність представлена окремою таблицею бази даних.

Найважливішою сутністю є **orders**, тобто таблиця замовлень. Вона пов'язує між собою клієнта, позиції замовлення, оплату та фінансові операції. Одне замовлення належить одному клієнту, але один клієнт може мати багато замовлень. Такий зв'язок має тип **один до багатьох**.

Для зберігання складу замовлення використовується таблиця **order_items**. Вона потрібна тому, що одне замовлення може містити декілька товарів або послуг. Кожна позиція замовлення пов'язана з конкретним замовленням і конкретною позицією з таблиці **products**. Завдяки цьому реалізується зв'язок між замовленнями та товарами.

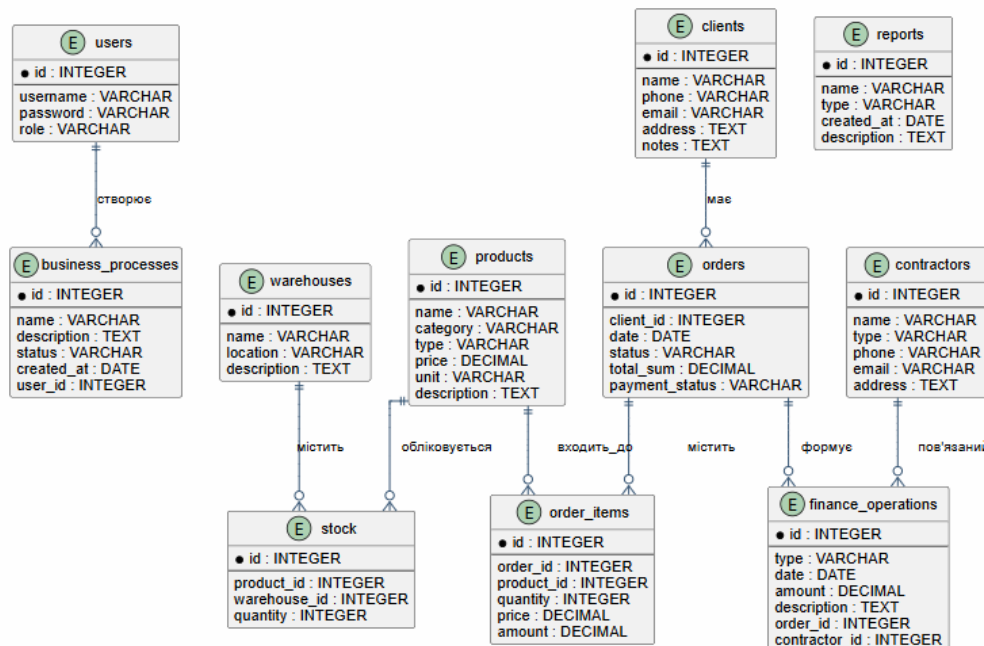


Рисунок 2.6 – ER-діаграма бази даних інформаційної системи

На рисунку 2.6 наведено ER-діаграму бази даних інформаційної системи моделювання процесів управління малим бізнесом. Діаграма містить основні таблиці, необхідні для реалізації функціональності системи: users,

clients, contractors, products, warehouses, stock, orders, order_items, finance_operations, business_processes та reports.

2.10. Проектування інтерфейсу користувача

Проектування інтерфейсу користувача є важливим етапом створення інформаційної системи, оскільки саме через інтерфейс користувач взаємодіє з програмним продуктом. Для інформаційної системи моделювання процесів управління малим бізнесом інтерфейс повинен бути простим, зрозумілим і логічно структурованим. Це особливо важливо, оскільки система орієнтована на власників малого бізнесу, адміністраторів або менеджерів, які можуть не мати спеціальної технічної підготовки.

Основною вимогою до інтерфейсу є зручність виконання щоденних операцій. Користувач повинен швидко знаходити потрібні розділи, додавати нові записи, переглядати дані, редагувати інформацію, створювати замовлення та формувати звіти. Надмірно складна структура меню або велика кількість зайвих елементів може ускладнювати роботу із системою, тому інтерфейс повинен бути мінімалістичним і функціональним.

Головна сторінка системи має виконувати роль інформаційної панелі. На ній доцільно розмістити коротку статистику: кількість клієнтів, кількість активних замовлень, суму доходів, витрат, прибуток, кількість товарів на складі та останні операції. Такий підхід дозволяє користувачу одразу після входу до системи отримати загальне уявлення про поточний стан малого бізнесу.

Навігація системи повинна містити основні розділи:

- головна сторінка;
- клієнти;
- контрагенти;
- товари та послуги;
- склади;
- замовлення;

- фінанси;
- бізнес-процеси;
- звіти;
- налаштування.

Для роботи з довідниками, такими як клієнти, контрагенти, товари та послуги, доцільно використовувати однаковий принцип побудови сторінок. Кожна сторінка повинна містити таблицю зі списком записів, кнопку додавання нового запису, поле пошуку, фільтри та кнопки для редагування або видалення. Це забезпечує єдність інтерфейсу та спрощує навчання користувача.



Рисунок 2.7 – Структура інтерфейсу користувача інформаційної системи

На рисунку 2.7 подано структуру інтерфейсу користувача інформаційної системи. Інтерфейс складається з головної сторінки, розділів для роботи з клієнтами, контрагентами, товарами та послугами, складами, замовленнями, фінансами, бізнес-процесами, звітами та налаштуваннями.

2.11. Висновки до розділу 2

У другому розділі бакалаврської роботи було виконано проектування інформаційної системи моделювання процесів управління малим бізнесом. На основі аналізу предметної області сформульовано постановку задачі розробки системи, визначено її основне призначення, користувачів і функціональні модулі.

Було визначено функціональні вимоги до системи. До них належать управління клієнтами, контрагентами, товарами та послугами, складом, замовленнями, фінансами, бізнес-процесами, звітами й аналітикою.

Реалізація цих вимог дозволяє створити єдине програмне середовище для автоматизації основних процесів малого підприємства.

Також було розглянуто нефункціональні вимоги, які визначають якісні характеристики системи. До них належать зручність використання, швидкодія, надійність, безпека, масштабованість, супроводжуваність, сумісність із сучасними браузерами та цілісність даних. Дотримання цих вимог дозволяє забезпечити стабільну й комфортну роботу користувача із системою.

РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1. Налаштування календарів, валют та основних параметрів системи

Програмна реалізація інформаційної системи моделювання процесів управління малим бізнесом передбачає створення вебзастосунку, який забезпечує роботу з основними управлінськими процесами підприємства. Перед реалізацією окремих модулів необхідно налаштувати базові параметри системи, оскільки саме вони визначають правила подальшої роботи із замовленнями, фінансовими операціями, звітами та аналітичними даними.

До основних параметрів системи належать:

- календарні налаштування;
- валюта обліку;
- формат відображення дат;
- формат фінансових сум;
- статуси замовлень;
- типи фінансових операцій;
- базові категорії товарів і послуг;
- параметри відображення звітності.

Календарні налаштування використовуються для коректного обліку дат створення замовлень, фінансових операцій, надходжень товарів,

формування звітів і аналізу діяльності підприємства за певний період. У системі передбачено використання стандартного календарного формату дати, що дозволяє зручно фільтрувати записи за днем, місяцем або роком.

Для малого бізнесу важливо мати можливість аналізувати діяльність за конкретні періоди. Наприклад, користувач може переглядати продажі за день, доходи за місяць, витрати за квартал або загальний прибуток за рік. Тому в програмній реалізації передбачено збереження дат у базі даних для таких сутностей, як замовлення, фінансові операції, бізнес-процеси та звіти.

Основною валютою системи обрано гривню, оскільки система орієнтована на використання в умовах українського малого бізнесу. Усі суми в замовленнях, фінансових операціях і звітах відображаються у гривнях. При цьому структура системи може бути розширена для підтримки інших валют, якщо підприємство працює з іноземними постачальниками або клієнтами.

У системі передбачено такі типи фінансових операцій:

дохід — надходження коштів від клієнтів за виконані замовлення або продані товари;

витрата — списання коштів на закупівлі, оплату послуг, оренду, логістику або інші потреби підприємства.

Для замовлень передбачено кілька статусів, які дозволяють контролювати процес їх виконання. Основними статусами є:

- нове;
- в обробці;
- виконане;
- очікує оплати;
- оплачене;
- скасоване.

Використання статусів дає змогу користувачу швидко оцінювати поточний стан замовлень. Наприклад, замовлення зі статусом «нове» ще не оброблялося, статус «в обробці» означає, що працівник уже працює із замовленням, а статус «оплачене» свідчить про завершення фінансової частини операції.

Для реалізації базових налаштувань у системі використовується окремий файл конфігурації або відповідний програмний блок. У ньому можуть бути задані значення за замовчуванням: назва системи, основна валюта, формат дати, перелік статусів замовлень і типів фінансових операцій. Це дозволяє централізовано керувати основними параметрами та спрощує подальше доопрацювання програми.

Приклад фрагмента налаштувань системи можна подати так:

```
SYSTEM_NAME = "Business Process Management System"
```

```
DEFAULT_CURRENCY = "UAH"
```

```
DATE_FORMAT = "%d.%m.%Y"
```

```
ORDER_STATUSES = [
```

```
    "Нове",
```

```
    "В обробці",
```

```
    "Очікує оплати",
```

```
    "Оплачене",
```

```
    "Виконане",
```

```
    "Скасоване"
```

```
]
```

```
FINANCE_TYPES = [
```

```
    "Дохід",
```

```
    "Витрата"
```

```
]
```

У цьому фрагменті задається назва системи, основна валюта, формат дати, перелік статусів замовлень і типів фінансових операцій. Такі параметри можуть використовуватися в різних модулях системи, зокрема під час створення замовлень, фіксації оплат і формування звітів.

Для збереження інформації про системні налаштування може використовуватися таблиця settings. Вона дозволяє зберігати параметри у базі даних і змінювати їх без редагування програмного коду.

Таблиця 3.1 – Основні параметри інформаційної системи

Параметр	Значення	Призначення
Назва системи	Business Process Management System	Відображення назви у вебінтерфейсі
Основна	UAH	Облік фінансових

Параметр	Значення	Призначення
валюта		операцій у гривні
Формат дати	день.місяць.рік	Відображення дат у зрозумілому форматі
Типи операцій	Дохід, Витрата	Класифікація фінансових записів
Статуси замовлень	Нове, В обробці, Очікує оплати, Оплачене, Виконане, Скасоване	Контроль етапів виконання замовлення
Періоди звітності	День, місяць, рік	Аналіз результатів діяльності підприємства

У вебінтерфейсі налаштування можуть бути представлені у вигляді окремої сторінки. На цій сторінці користувач може переглядати основні параметри системи, змінювати валюту, формат дати або інші доступні значення. У базовій версії бакалаврської роботи достатньо реалізувати відображення основних параметрів і використання їх у відповідних модулях.

3.2. Налаштування номенклатури товарів і послуг

Номенклатура товарів і послуг є одним із базових довідників інформаційної системи, оскільки саме ці дані використовуються під час створення замовлень, розрахунку вартості продажів, контролю складських залишків і формування звітності. Для малого бізнесу правильне ведення номенклатури має важливе значення, оскільки помилки в назвах, цінах або категоріях можуть призвести до неправильного оформлення замовлень і фінансових розрахунків.

У розроблюваній інформаційній системі номенклатура представлена окремим модулем «**Товари та послуги**». Цей модуль дозволяє користувачу додавати нові позиції, переглядати наявні записи, редагувати інформацію, видаляти застарілі позиції та використовувати товари або послуги під час створення замовлень.

Кожна позиція номенклатури повинна містити основні характеристики, необхідні для її ідентифікації та використання в системі. До таких характеристик належать:

- назва товару або послуги;
- категорія;
- тип позиції;
- ціна;
- одиниця виміру;
- опис;
- статус доступності.

Поле **«Назва»** використовується для відображення товару або послуги в інтерфейсі системи. Воно має бути зрозумілим і коротким, щоб користувач міг швидко знайти потрібну позицію під час оформлення замовлення.

Поле **«Категорія»** дозволяє групувати товари та послуги за певними ознаками. Наприклад, для малого магазину це можуть бути категорії «Товари для дому», «Канцелярія», «Електроніка», «Послуги доставки» або «Сервісне обслуговування». Групування за категоріями спрощує пошук і формування звітів.

Поле **«Тип позиції»** дозволяє розрізняти товари та послуги. Це важливо, оскільки товари зазвичай пов'язані зі складським обліком, а послуги не потребують контролю залишків. Наприклад, якщо користувач додає до замовлення товар, система може перевіряти його наявність на складі. Якщо ж додається послуга, така перевірка не виконується.

Поле **«Ціна»** використовується для автоматичного розрахунку вартості замовлення. Під час додавання позиції до замовлення система множить ціну на кількість і визначає суму за конкретною позицією. Це дозволяє зменшити кількість ручних розрахунків і знизити ймовірність помилки.

Поле **«Одиниця виміру»** потрібне для правильного відображення кількості товарів або послуг. Для товарів це можуть бути одиниці «шт.», «кг», «м», «л», а для послуг — «послуга», «година», «день» тощо.

У базі даних номенклатура зберігається в таблиці `products`. Незважаючи на назву таблиці, у ній можуть зберігатися як товари, так і послуги. Для їх розділення використовується поле `type`.

Приклад структури таблиці `products`:

```
CREATE TABLE products (  
  id INTEGER PRIMARY KEY AUTOINCREMENT,
```

```

    name TEXT NOT NULL,
    category TEXT,
    type TEXT NOT NULL,
    price REAL NOT NULL,
    unit TEXT,
    description TEXT
);

```

У цьому фрагменті SQL-коду створюється таблиця для зберігання номенклатури. Поле `id` є первинним ключем, поле `name` містить назву товару або послуги, `category` визначає категорію, `type` вказує тип позиції, `price` зберігає ціну, `unit` — одиницю виміру, а `description` — додатковий опис.

Для роботи з номенклатурою у вебзастосунку реалізуються відповідні маршрути Flask. Вони забезпечують перегляд списку товарів і послуг, додавання нової позиції, редагування та видалення записів.

Приклад фрагмента маршруту для відображення номенклатури:

```

@app.route("/products")
def products_list():
    products = Product.query.all()
    return render_template("products.html", products=products)

```

Цей фрагмент коду показує простий маршрут `/products`, який отримує всі записи з таблиці товарів і послуг та передає їх до HTML-шаблону для відображення користувачу.

Для додавання нової позиції може використовуватися форма, у якій користувач вводить назву, категорію, тип, ціну, одиницю виміру та опис. Після надсилання форми система перевіряє правильність введених даних і зберігає запис у базі даних.

Приклад логіки додавання товару або послуги:

```

@app.route("/products/add", methods=["GET", "POST"])
def add_product():
    if request.method == "POST":
        name = request.form["name"]
        category = request.form["category"]
        product_type = request.form["type"]
        price = float(request.form["price"])
        unit = request.form["unit"]
        description = request.form["description"]

```



```

product = Product(
    name=name,
    category=category,
    type=product_type,
    price=price,
    unit=unit,
    description=description
)

db.session.add(product)
db.session.commit()

return redirect(url_for("products_list"))

return render_template("product_form.html")

```

У цьому прикладі користувач заповнює форму, після чого система створює новий об'єкт `Product` і зберігає його в базі даних. Після успішного додавання користувач повертається до списку товарів і послуг.

Для зручності роботи користувача на сторінці номенклатури доцільно реалізувати таблицю зі списком позицій. У ній можуть відображатися назва, категорія, тип, ціна, одиниця виміру та доступні дії. Наприклад, користувач може натиснути кнопку «**Редагувати**» для зміни даних або кнопку «**Видалити**» для видалення позиції.

Таблиця 3.2 – Приклад структури сторінки номенклатури товарів і послуг

Елемент сторінки	Призначення
Кнопка «Додати позицію»	Перехід до форми створення нового товару або послуги
Поле пошуку	Пошук позиції за назвою або категорією
Таблиця номенклатури	Відображення списку товарів і послуг
Фільтр за типом	Відбір тільки товарів або тільки послуг
Кнопка «Редагувати»	Зміна даних вибраної позиції
Кнопка «Видалити»	Видалення застарілої або помилкової позиції

Налаштування номенклатури безпосередньо впливає на роботу інших модулів системи. Наприклад, під час створення замовлення користувач обирає товари або послуги саме з цього довідника. Якщо позиція має тип

«товар», система може перевіряти її наявність на складі. Якщо позиція має тип «послуга», вона додається до замовлення без складського списання.

3.3. Реалізація модуля обліку клієнтів

Модуль обліку клієнтів є одним із основних компонентів інформаційної системи, оскільки клієнти є центральними учасниками процесів продажу, оформлення замовлень і аналізу діяльності малого бізнесу. Наявність структурованої клієнтської бази дозволяє підприємству швидко знаходити потрібну інформацію, переглядати історію взаємодії, контролювати замовлення та покращувати якість обслуговування.

У розроблюваній системі модуль обліку клієнтів забезпечує виконання таких операцій:

- додавання нового клієнта;
- перегляд списку клієнтів;
- редагування даних клієнта;
- видалення клієнта;
- пошук клієнта за ім'ям, телефоном або електронною поштою;
- використання клієнта під час створення замовлення.

Кожен клієнт у системі має набір основних даних, які необхідні для подальшої роботи. До них належать ім'я або назва клієнта, номер телефону, електронна пошта, адреса та додаткові примітки. Поле з примітками може використовуватися для збереження особливостей співпраці, коментарів менеджера або додаткової інформації, яка може бути корисною під час повторної взаємодії.

У базі даних інформація про клієнтів зберігається в таблиці `clients`. Ця таблиця є однією з головних, оскільки вона пов'язана з таблицею замовлень. Один клієнт може мати багато замовлень, тому між таблицями `clients` і `orders` встановлюється зв'язок типу **один до багатьох**.

Приклад SQL-структури таблиці клієнтів:

```
CREATE TABLE clients (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  name TEXT NOT NULL,
  phone TEXT,
```

```

    email TEXT,
    address TEXT,
    notes TEXT
);

```

У цій таблиці поле `id` є унікальним ідентифікатором клієнта. Поле `name` є обов'язковим, оскільки воно використовується для відображення клієнта у списку та під час створення замовлення. Інші поля можуть бути необов'язковими, однак їх наявність дозволяє зберігати більш повну інформацію про клієнта.

Для реалізації модуля клієнтів у Flask-застосунку створюються маршрути для перегляду, додавання, редагування та видалення записів. Основним маршрутом є сторінка зі списком клієнтів. На ній користувач може переглянути всі записи, знайти потрібного клієнта та перейти до форми редагування.

Приклад маршруту для перегляду клієнтів:

```

@app.route("/clients")
def clients_list():
    clients = Client.query.all()
    return render_template("clients.html", clients=clients)

```

Цей маршрут отримує всі записи з таблиці `clients` і передає їх до шаблону `clients.html`. У шаблоні дані відображаються у вигляді таблиці, де кожен рядок відповідає окремому клієнту.

Для додавання нового клієнта використовується окрема форма. Користувач вводить необхідні дані, після чого система перевіряє їх і зберігає новий запис у базі даних.

Приклад маршруту додавання клієнта:

```

@app.route("/clients/add", methods=["GET", "POST"])
def add_client():
    if request.method == "POST":
        name = request.form["name"]
        phone = request.form.get("phone")
        email = request.form.get("email")
        address = request.form.get("address")
        notes = request.form.get("notes")

        client = Client(
            name=name,

```

```

        phone=phone,
        email=email,
        address=address,
        notes=notes
    )

    db.session.add(client)
    db.session.commit()

    return redirect(url_for("clients_list"))

    return render_template("client_form.html")

```

У цьому фрагменті коду після надсилання форми створюється новий об'єкт Client, який додається до бази даних. Після збереження користувач автоматично повертається до сторінки зі списком клієнтів.

Редагування клієнта реалізується через отримання конкретного запису за його ідентифікатором. Якщо користувач змінює дані у формі та натискає кнопку збереження, система оновлює відповідний запис у базі даних.

```

@app.route("/clients/edit/<int:client_id>", methods=["GET", "POST"])
def edit_client(client_id):
    client = Client.query.get_or_404(client_id)

    if request.method == "POST":
        client.name = request.form["name"]
        client.phone = request.form.get("phone")
        client.email = request.form.get("email")
        client.address = request.form.get("address")
        client.notes = request.form.get("notes")

        db.session.commit()

        return redirect(url_for("clients_list"))

    return render_template("client_form.html", client=client)

```

Видалення клієнта повинно виконуватися обережно, оскільки клієнт може бути пов'язаний із замовленнями. Якщо у клієнта вже є замовлення, повне видалення може порушити цілісність даних. Тому в реальній системі доцільно не видаляти клієнта фізично з бази, а змінювати його статус на неактивний. У базовій версії системи можна реалізувати просте

видалення, але в тексті роботи варто зазначити можливість подальшого вдосконалення.

Приклад маршруту видалення клієнта:

```
@app.route("/clients/delete/<int:client_id>", methods=["POST"])
def delete_client(client_id):
    client = Client.query.get_or_404(client_id)

    db.session.delete(client)
    db.session.commit()

    return redirect(url_for("clients_list"))
```

Для зручності користувача сторінка клієнтів повинна мати просту та зрозумілу структуру. Основними елементами сторінки є кнопка додавання нового клієнта, поле пошуку, таблиця клієнтів і кнопки дій.

Таблиця 3.3 – Структура сторінки модуля обліку клієнтів

Елемент сторінки	Призначення
Кнопка «Додати клієнта»	Відкриття форми створення нового клієнта
Поле пошуку	Пошук клієнта за іменем, телефоном або електронною поштою
Таблиця клієнтів	Відображення списку клієнтів
Кнопка «Редагувати»	Перехід до зміни даних клієнта
Кнопка «Видалити»	Видалення клієнта або зміна його статусу
Посилання на замовлення	Перегляд замовлень, пов'язаних із клієнтом

Модуль клієнтів тісно пов'язаний із модулем замовлень. Під час створення нового замовлення користувач обирає клієнта зі списку.

3.4. Реалізація модуля обліку контрагентів

Модуль обліку контрагентів призначений для зберігання та обробки інформації про постачальників, партнерів, підрядників та інші організації,

з якими взаємодіє малий бізнес. У діяльності підприємства контрагенти відіграють важливу роль, оскільки саме через них можуть здійснюватися закупівлі товарів, постачання матеріалів, оплата послуг, виконання робіт або інші господарські операції.

У розроблюваній інформаційній системі модуль контрагентів дозволяє впорядкувати інформацію про зовнішніх учасників бізнес-процесів. Це дає змогу швидко знаходити потрібні контактні дані, переглядати умови співпраці, аналізувати витрати та контролювати взаєморозрахунки.

Основними функціями модуля обліку контрагентів є:

- додавання нового контрагента;
- перегляд списку контрагентів;
- редагування даних контрагента;
- видалення або деактивація запису;
- пошук контрагента за назвою, телефоном або типом;
- використання контрагента у фінансових операціях;
- збереження додаткової інформації про співпрацю.

Кожен контрагент у системі має містити набір основних реквізитів. До них належать назва контрагента, тип співпраці, телефон, електронна пошта, адреса та примітки. Поле «**Тип**» дозволяє визначити, ким саме є контрагент: постачальником, партнером, підрядником або сервісною організацією. Це спрощує фільтрацію та подальший аналіз даних.

У базі даних інформація про контрагентів зберігається в таблиці *contractors*. Ця таблиця може бути пов'язана з фінансовими операціями, оскільки витрати підприємства часто виникають саме в результаті взаємодії з постачальниками або партнерами.

Приклад SQL-структури таблиці контрагентів:

```
CREATE TABLE contractors (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  name TEXT NOT NULL,
  type TEXT,
  phone TEXT,
  email TEXT,
  address TEXT,
```

notes TEXT
);

У цій таблиці поле `id` є первинним ключем, поле `name` містить назву контрагента, поле `type` визначає тип співпраці, а поля `phone`, `email`, `address` і `notes` використовуються для зберігання контактної та додаткової інформації.

Для програмної реалізації модуля контрагентів у Flask-застосунку створюються маршрути для перегляду, додавання, редагування та видалення записів. За логікою роботи цей модуль подібний до модуля клієнтів, однак його дані використовуються переважно у фінансовому обліку та взаємодії з постачальниками.

Приклад маршруту для перегляду списку контрагентів:

```
@app.route("/contractors")
def contractors_list():
    contractors = Contractor.query.all()
    return render_template("contractors.html", contractors=contractors)
```

Цей маршрут отримує всі записи з таблиці `contractors` і передає їх до HTML-шаблону. На сторінці користувач бачить список контрагентів і може виконувати з ними необхідні дії.

Для додавання нового контрагента використовується окрема форма. Користувач вводить назву, тип, контактні дані та примітки, після чого система зберігає запис у базі даних.

```
@app.route("/contractors/add", methods=["GET", "POST"])
def add_contractor():
    if request.method == "POST":
        name = request.form["name"]
        contractor_type = request.form.get("type")
        phone = request.form.get("phone")
        email = request.form.get("email")
        address = request.form.get("address")
        notes = request.form.get("notes")

        contractor = Contractor(
            name=name,
            type=contractor_type,
            phone=phone,
            email=email,
```

```

        address=address,
        notes=notes
    )

    db.session.add(contractor)
    db.session.commit()

    return redirect(url_for("contractors_list"))

    return render_template("contractor_form.html")

```

У цьому прикладі після надсилання форми створюється новий об'єкт Contractor, який зберігається в базі даних. Після успішного додавання користувач повертається до загального списку контрагентів.

Редагування контрагента реалізується шляхом отримання запису за його ідентифікатором. Після зміни даних система оновлює запис у базі даних.

```

@app.route("/contractors/edit/<int:contractor_id>", methods=["GET", "POST"])
def edit_contractor(contractor_id):
    contractor = Contractor.query.get_or_404(contractor_id)

    if request.method == "POST":
        contractor.name = request.form["name"]
        contractor.type = request.form.get("type")
        contractor.phone = request.form.get("phone")
        contractor.email = request.form.get("email")
        contractor.address = request.form.get("address")
        contractor.notes = request.form.get("notes")

    db.session.commit()

    return redirect(url_for("contractors_list"))

    return render_template("contractor_form.html", contractor=contractor)

```

Модуль контрагентів пов'язаний із фінансовим модулем. Наприклад, якщо підприємство здійснює оплату постачальнику, у фінансовій операції можна вказати відповідного контрагента. Це дозволяє не лише фіксувати витрати, а й аналізувати, на яких постачальників або партнерів припадає найбільша частина витрат.

Таблиця 3.4 – Структура сторінки модуля обліку контрагентів

Елемент сторінки	Призначення
Кнопка «Додати контрагента»	Відкриття форми створення нового запису
Поле пошуку	Пошук контрагента за назвою, телефоном або типом
Фільтр за типом	Відбір постачальників, партнерів або підрядників
Таблиця контрагентів	Відображення списку зовнішніх організацій
Кнопка «Редагувати»	Зміна інформації про контрагента
Кнопка «Видалити»	Видалення або деактивація запису
Перегляд операцій	Перехід до фінансових операцій, пов'язаних із контрагентом

Для зручності користувача список контрагентів доцільно подавати у вигляді таблиці. У ній відображаються назва, тип, телефон, електронна пошта, адреса та доступні дії. Така структура дозволяє швидко переглянути основну інформацію й перейти до редагування потрібного запису.

3.5. Реалізація модуля управління складами

Модуль управління складами призначений для контролю місць зберігання товарів, обліку складських залишків, надходжень і списань. Для малого бізнесу складський облік має важливе значення, оскільки від точності інформації про наявність товарів залежить можливість своєчасного виконання замовлень і якість обслуговування клієнтів.

У розроблюваній інформаційній системі модуль складів забезпечує зв'язок між товарами, замовленнями та складськими залишками. Якщо користувач додає товар до замовлення, система повинна перевірити, чи є він у достатній кількості на складі. Після оформлення продажу кількість товару автоматично зменшується, що дозволяє підтримувати актуальний стан залишків.

Основними функціями модуля управління складами є:

- додавання складу або місця зберігання;

- перегляд списку складів;
- редагування інформації про склад;
- перегляд залишків товарів;
- додавання товару на склад;
- оновлення кількості товарів;
- списання товарів після продажу;
- контроль недостатньої кількості товару.

У базовій версії системи може бути реалізований один склад, наприклад «**Основний склад**». Однак структура бази даних передбачає можливість роботи з кількома складами. Це корисно для малого бізнесу, який може мати окремий склад, торгову точку або місце зберігання товарів.

Для реалізації складського обліку використовуються дві основні таблиці: `warehouses` і `stock`. Таблиця `warehouses` зберігає інформацію про склади, а таблиця `stock` містить дані про кількість конкретного товару на конкретному складі.

Приклад SQL-структури таблиці складів:

```
CREATE TABLE warehouses (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  name TEXT NOT NULL,
  location TEXT,
  description TEXT
);
```

Приклад SQL-структури таблиці складських залишків:

```
CREATE TABLE stock (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  product_id INTEGER NOT NULL,
  warehouse_id INTEGER NOT NULL,
  quantity INTEGER NOT NULL DEFAULT 0,
  FOREIGN KEY (product_id) REFERENCES products(id),
  FOREIGN KEY (warehouse_id) REFERENCES warehouses(id)
);
```

У таблиці `warehouses` поле `id` є первинним ключем, `name` містить назву складу, `location` — місце розташування, а `description` — додатковий опис. У

таблиці `stock` зберігається зв'язок між товаром і складом. Поле `product_id` вказує на товар, `warehouse_id` — на склад, а `quantity` містить кількість товару.

Для перегляду складів у Flask-застосунку створюється окремий маршрут. Він отримує записи з таблиці складів і передає їх до HTML-шаблону.

```
@app.route("/warehouses")
def warehouses_list():
    warehouses = Warehouse.query.all()
    return render_template("warehouses.html", warehouses=warehouses)
```

Для додавання нового складу користувач заповнює форму, у якій вказує назву, розташування та опис. Після надсилання форми система створює новий запис у базі даних.

```
@app.route("/warehouses/add", methods=["GET", "POST"])
def add_warehouse():
    if request.method == "POST":
        name = request.form["name"]
        location = request.form.get("location")
        description = request.form.get("description")

        warehouse = Warehouse(
            name=name,
            location=location,
            description=description
        )

        db.session.add(warehouse)
        db.session.commit()

        return redirect(url_for("warehouses_list"))

    return render_template("warehouse_form.html")
```

Окрему роль у системі відіграє сторінка перегляду складських залишків. На ній користувач бачить, які товари зберігаються на складі та в якій кількості. Для зручності можна додати фільтрацію за складом, категорією товару або назвою позиції.

Приклад маршруту для перегляду залишків:

```

@app.route("/stock")
def stock_list():
    stock_items = Stock.query.all()
    return render_template("stock.html", stock_items=stock_items)

```

Для оновлення кількості товару на складі може використовуватися окрема форма. Користувач обирає товар, склад і вказує нову кількість або кількість надходження. Така операція потрібна під час закупівлі товарів або коригування залишків.

```

@app.route("/stock/update", methods=["GET", "POST"])
def update_stock():
    if request.method == "POST":
        product_id = request.form["product_id"]
        warehouse_id = request.form["warehouse_id"]
        quantity = int(request.form["quantity"])

        stock_item = Stock.query.filter_by(
            product_id=product_id,
            warehouse_id=warehouse_id
        ).first()

        if stock_item:
            stock_item.quantity = quantity
        else:
            stock_item = Stock(
                product_id=product_id,
                warehouse_id=warehouse_id,
                quantity=quantity
            )
            db.session.add(stock_item)

        db.session.commit()

    return redirect(url_for("stock_list"))

products = Product.query.filter_by(type="Tovar").all()
warehouses = Warehouse.query.all()
return render_template(
    "stock_form.html",
    products=products,

```

```

        warehouses=warehouses
    )

```

У цьому фрагменті коду система перевіряє, чи існує вже запис про певний товар на певному складі. Якщо запис існує, оновлюється кількість. Якщо запису ще немає, створюється новий складський запис.

Важливою частиною модуля є перевірка залишків під час оформлення замовлення. Якщо користувач додає до замовлення товар, система повинна переконатися, що кількість товару на складі є достатньою.

Приклад логіки перевірки залишків:

```

def check_stock(product_id, warehouse_id, required_quantity):
    stock_item = Stock.query.filter_by(
        product_id=product_id,
        warehouse_id=warehouse_id
    ).first()

    if not stock_item:
        return False

    return stock_item.quantity >= required_quantity

```

Після успішного оформлення замовлення кількість товару на складі повинна зменшуватися. Це дозволяє автоматично підтримувати актуальний стан залишків.

```

def decrease_stock(product_id, warehouse_id, quantity):
    stock_item = Stock.query.filter_by(
        product_id=product_id,
        warehouse_id=warehouse_id
    ).first()

    if stock_item and stock_item.quantity >= quantity:
        stock_item.quantity -= quantity
        db.session.commit()
        return True

    return False

```

Для користувача сторінка складського обліку повинна бути простою та зрозумілою. Основні елементи такої сторінки подано в таблиці 3.5.

Таблиця 3.5 – Структура сторінки модуля управління складами

Елемент сторінки	Призначення
Список складів	Перегляд наявних місць зберігання товарів
Кнопка «Додати склад»	Створення нового складу
Таблиця залишків	Відображення кількості товарів на складах
Фільтр за складом	Перегляд залишків на конкретному складі
Фільтр за товаром	Пошук залишків певного товару
Кнопка «Оновити залишок»	Зміна кількості товару на складі
Попередження про нестачу	Інформування користувача про недостатню кількість товару

Модуль складів тісно пов'язаний із модулем товарів і послуг. У складському обліку використовуються лише ті позиції, які мають тип **«Товар»**. Послуги не потребують зберігання на складі, тому вони не відображаються у складських залишках.

3.6. Реалізація модуля обліку замовлень і продажів

Модуль обліку замовлень і продажів є центральним компонентом інформаційної системи, оскільки саме через нього здійснюється основна взаємодія між клієнтами, товарами, послугами, складом і фінансовим обліком. Для малого бізнесу правильна організація роботи із замовленнями має важливе значення, адже від швидкості та точності обробки замовлень залежить якість обслуговування клієнтів, рівень продажів і фінансовий результат підприємства.

У розроблюваній системі модуль замовлень забезпечує повний цикл роботи із замовленням: створення, наповнення товарами або послугами, розрахунок суми, перевірку складських залишків, зміну статусу, фіксацію оплати та передачу даних до фінансового обліку і звітності.

Основними функціями модуля обліку замовлень і продажів є:

- створення нового замовлення;
- вибір клієнта з бази;

- додавання товарів або послуг до замовлення;
- автоматичний розрахунок суми;
- перевірка наявності товарів на складі;
- зміна статусу замовлення;
- фіксація оплати;
- оновлення складських залишків;
- передача даних до фінансового модуля;
- використання даних замовлень у звітах.

У базі даних для реалізації цього модуля використовуються дві основні таблиці: `orders` і `order_items`. Таблиця `orders` зберігає загальну інформацію про замовлення, а таблиця `order_items` містить окремі товари або послуги, які входять до складу замовлення.

Приклад SQL-структури таблиці замовлень:

```
CREATE TABLE orders (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  client_id INTEGER NOT NULL,
  date TEXT NOT NULL,
  status TEXT NOT NULL,
  total_sum REAL NOT NULL DEFAULT 0,
  payment_status TEXT NOT NULL,
  FOREIGN KEY (client_id) REFERENCES clients(id)
);
```

Приклад SQL-структури таблиці позицій замовлення:

```
CREATE TABLE order_items (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  order_id INTEGER NOT NULL,
  product_id INTEGER NOT NULL,
  quantity INTEGER NOT NULL,
  price REAL NOT NULL,
  amount REAL NOT NULL,
  FOREIGN KEY (order_id) REFERENCES orders(id),
  FOREIGN KEY (product_id) REFERENCES products(id)
);
```

Така структура дозволяє реалізувати зв'язок типу **одне замовлення — багато позицій**. Наприклад, одне замовлення може містити декілька товарів і послуг, а кожна позиція замовлення зберігається окремим записом у таблиці `order_items`.

Під час створення нового замовлення користувач спочатку обирає клієнта. Якщо потрібного клієнта ще немає в базі, його можна додати через модуль обліку клієнтів. Після вибору клієнта користувач додає до замовлення товари або послуги, вказує кількість і переглядає попередній розрахунок суми.

Приклад маршруту для перегляду списку замовлень:

```
@app.route("/orders")
def orders_list():
    orders = Order.query.all()
    return render_template("orders.html", orders=orders)
```

Цей маршрут отримує всі замовлення з бази даних і передає їх до HTML-шаблону. На сторінці замовлень користувач може переглянути дату, клієнта, суму, статус замовлення та стан оплати.

Приклад маршруту створення нового замовлення:

```
@app.route("/orders/create", methods=["GET", "POST"])
def create_order():
    if request.method == "POST":
        client_id = request.form["client_id"]
        status = "Нове"
        payment_status = "Очікує оплати"

        order = Order(
            client_id=client_id,
            date=datetime.now().strftime("%d.%m.%Y"),
            status=status,
            total_sum=0,
            payment_status=payment_status
        )

        db.session.add(order)
        db.session.commit()

    return redirect(url_for("edit_order", order_id=order.id))
```



```
clients = Client.query.all()
return render_template("order_create.html", clients=clients)
```

У цьому прикладі після вибору клієнта створюється порожнє замовлення зі статусом «**Нове**». Далі користувач переходить до сторінки редагування замовлення, де може додати товари або послуги.

Додавання позицій до замовлення виконується через окрему форму. Користувач обирає товар або послугу, вказує кількість, після чого система визначає ціну та розраховує суму за позицією.

```
@app.route("/orders/<int:order_id>/add-item", methods=["POST"])
def add_order_item(order_id):
    order = Order.query.get_or_404(order_id)

    product_id = int(request.form["product_id"])
    quantity = int(request.form["quantity"])

    product = Product.query.get_or_404(product_id)
    amount = product.price * quantity

    item = OrderItem(
        order_id=order.id,
        product_id=product.id,
        quantity=quantity,
        price=product.price,
        amount=amount
    )

    db.session.add(item)
    order.total_sum += amount
    db.session.commit()

    return redirect(url_for("edit_order", order_id=order.id))
```

У цьому фрагменті коду система отримує товар, визначає його ціну, множить її на кількість і додає позицію до замовлення. Після цього загальна сума замовлення оновлюється.

Якщо до замовлення додається товар, система повинна перевірити його наявність на складі. Для послуг така перевірка не потрібна, оскільки вони не мають складського залишку.

```
def can_add_to_order(product, quantity, warehouse_id):
    if product.type == "Послуга":
        return True

    stock_item = Stock.query.filter_by(
        product_id=product.id,
        warehouse_id=warehouse_id
    ).first()

    if not stock_item:
        return False

    return stock_item.quantity >= quantity
```

Ця функція перевіряє, чи можна додати товар до замовлення. Якщо позиція має тип **«Послуга»**, система дозволяє додавання без перевірки складу. Якщо позиція є товаром, перевіряється його кількість на складі.

Після оформлення продажу необхідно оновити складські залишки. Це означає, що кількість проданого товару повинна бути списана зі складу.

```
def write_off_stock(product_id, warehouse_id, quantity):
    stock_item = Stock.query.filter_by(
        product_id=product_id,
        warehouse_id=warehouse_id
    ).first()

    if stock_item and stock_item.quantity >= quantity:
        stock_item.quantity -= quantity
        db.session.commit()
        return True

    return False
```

Важливою частиною модуля є зміна статусу замовлення. Статуси дозволяють контролювати етап виконання замовлення. Наприклад, після створення замовлення воно має статус **«Нове»**, після початку обробки — **«В обробці»**, після оплати — **«Оплачене»**, після завершення — **«Виконане»**.

Приклад маршруту зміни статусу замовлення:

```
@app.route("/orders/<int:order_id>/status", methods=["POST"])
def update_order_status(order_id):
```

```
order = Order.query.get_or_404(order_id)
order.status = request.form["status"]
```

```
db.session.commit()
```

```
return redirect(url_for("edit_order", order_id=order.id))
```

Фіксація оплати є ще одним важливим етапом роботи із замовленням. Після оплати система змінює стан оплати замовлення та створює запис у таблиці фінансових операцій.

```
@app.route("/orders/<int:order_id>/pay", methods=["POST"])
```

```
def pay_order(order_id):
```

```
    order = Order.query.get_or_404(order_id)
```

```
    order.payment_status = "Оплачено"
```

```
    order.status = "Оплачено"
```

```
    finance_operation = FinanceOperation(
        type="Дохід",
        date=datetime.now().strftime("%d.%m.%Y"),
        amount=order.total_sum,
        description=f"Оплата замовлення №{order.id}",
        order_id=order.id
    )
```

```
    db.session.add(finance_operation)
```

```
    db.session.commit()
```

```
    return redirect(url_for("edit_order", order_id=order.id))
```

У цьому прикладі після оплати замовлення створюється фінансова операція типу «Дохід». Це дозволяє автоматично передавати інформацію про продаж до фінансового модуля та використовувати її у звітах.

Таблиця 3.6 – Статуси замовлень у системі

Статус замовлення	Значення
Нове	Замовлення створено, але ще не оброблялося
В обробці	Замовлення перебуває в процесі виконання
Очікує оплати	Замовлення оформлене, але оплата ще не внесена

Статус замовлення	Значення
Оплачене	Оплату за замовлення зафіксовано
Виконане	Замовлення повністю завершено
Скасоване	Замовлення скасоване користувачем

Для користувача сторінка замовлення повинна містити основну інформацію про клієнта, дату, статус, стан оплати, список товарів і послуг, загальну суму та доступні дії. Користувач має мати змогу додавати позиції, змінювати статус, фіксувати оплату або переглядати деталі замовлення.

Таблиця 3.7 – Основні елементи сторінки замовлення

Елемент сторінки	Призначення
Дані клієнта	Відображення клієнта, який пов'язаний із замовленням
Список позицій	Перегляд товарів і послуг у складі замовлення
Поле кількості	Введення кількості товару або послуги
Загальна сума	Автоматичний розрахунок вартості замовлення
Статус замовлення	Контроль етапу виконання
Стан оплати	Відображення інформації про оплату
Кнопка «Додати позицію»	Додавання товару або послуги до замовлення
Кнопка «Зафіксувати оплату»	Створення фінансової операції
Кнопка «Зберегти»	Збереження змін у замовленні

Модуль замовлень пов'язаний майже з усіма іншими частинами системи. Він використовує дані клієнтів, номенклатуру товарів і послуг, складські залишки та фінансові операції. Саме тому цей модуль є основним прикладом моделювання бізнес-процесів у розроблюваній інформаційній системі.

3.7. Реалізація модуля фінансового обліку

Модуль фінансового обліку призначений для фіксації, зберігання та аналізу фінансових операцій малого підприємства. У межах інформаційної системи цей модуль забезпечує облік доходів, витрат, оплат за замовленнями, платежів контрагентам і розрахунок фінансового результату діяльності підприємства.

Для малого бізнесу фінансовий облік має важливе значення, оскільки дає змогу контролювати надходження коштів, оцінювати витрати, визначати прибуток і приймати обґрунтовані управлінські рішення. Якщо фінансові дані ведуться вручну або в окремих таблицях, виникає ризик помилок, дублювання записів і складності під час формування звітів. Тому доцільно реалізувати фінансовий облік як окремий модуль інформаційної системи.

Основними функціями модуля фінансового обліку є:

- додавання фінансової операції;
- перегляд списку доходів і витрат;
- редагування фінансових записів;
- видалення помилкових операцій;
- фільтрація операцій за типом і датою;
- зв'язок фінансової операції із замовленням;
- зв'язок витрати з контрагентом;
- розрахунок загальної суми доходів;
- розрахунок загальної суми витрат;
- визначення прибутку за вибраний період.

У системі передбачено два основні типи фінансових операцій: **дохід** і **витрата**. Дохід виникає після оплати замовлення клієнтом. Витрата може бути пов'язана із закупівлею товарів, оплатою послуг, орендою, доставкою, рекламою або іншими потребами підприємства.

У базі даних фінансові операції зберігаються в таблиці `finance_operations`. Вона містить інформацію про тип операції, дату, суму, опис, пов'язане замовлення або контрагента.

Приклад SQL-структури таблиці фінансових операцій:

```
CREATE TABLE finance_operations (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
```

```

type TEXT NOT NULL,
date TEXT NOT NULL,
amount REAL NOT NULL,
description TEXT,
order_id INTEGER,
contractor_id INTEGER,
FOREIGN KEY (order_id) REFERENCES orders(id),
FOREIGN KEY (contractor_id) REFERENCES contractors(id)
);

```

У цій таблиці поле `id` є первинним ключем. Поле `type` визначає тип операції: «Дохід» або «Витрата». Поле `date` зберігає дату операції, `amount` — суму, `description` — пояснення або коментар. Поле `order_id` використовується для зв'язку фінансової операції із замовленням, а `contractor_id` — для зв'язку з контрагентом.

Для перегляду фінансових операцій у Flask-застосунку реалізується окремий маршрут. Він отримує записи з бази даних і передає їх до HTML-шаблону.

```

@app.route("/finance")
def finance_list():
    operations = FinanceOperation.query.all()
    return render_template("finance.html", operations=operations)

```

На сторінці фінансів користувач може переглядати всі доходи та витрати підприємства. Для зручності записи доцільно виводити у вигляді таблиці, де відображаються дата, тип операції, сума, опис, пов'язане замовлення або контрагент.

Додавання фінансової операції виконується через форму. Користувач обирає тип операції, вводить суму, дату, опис і за потреби вибирає контрагента.

```

@app.route("/finance/add", methods=["GET", "POST"])
def add_finance_operation():
    if request.method == "POST":
        operation_type = request.form["type"]
        date = request.form["date"]
        amount = float(request.form["amount"])
        description = request.form.get("description")
        contractor_id = request.form.get("contractor_id")

        operation = FinanceOperation(
            type=operation_type,

```

```

        date=date,
        amount=amount,
        description=description,
        contractor_id=contractor_id if contractor_id else None
    )

    db.session.add(operation)
    db.session.commit()

    return redirect(url_for("finance_list"))

contractors = Contractor.query.all()
return render_template("finance_form.html", contractors=contractors)

```

У цьому фрагменті коду створюється нова фінансова операція. Якщо операція пов'язана з контрагентом, його ідентифікатор також зберігається в базі даних. Це дозволяє надалі аналізувати витрати за постачальниками або партнерами.

Фінансова операція типу «Дохід» може створюватися автоматично після оплати замовлення. Такий підхід уже використовується в модулі замовлень, де після фіксації оплати система створює запис у таблиці `finance_operations`. Завдяки цьому користувачу не потрібно вручну дублювати інформацію про продаж у фінансовому модулі.

Для аналізу фінансового стану підприємства система повинна розраховувати загальні доходи, витрати та прибуток. Прибуток визначається як різниця між доходами та витратами.

```

def calculate_financial_summary():
    income = db.session.query(
        db.func.sum(FinanceOperation.amount)
    ).filter_by(type="Дохід").scalar() or 0

    expenses = db.session.query(
        db.func.sum(FinanceOperation.amount)
    ).filter_by(type="Витрати").scalar() or 0

    profit = income - expenses

    return {
        "income": income,
        "expenses": expenses,
        "profit": profit
    }

```

Ця функція підсумовує всі операції типу «Дохід» і «Витрата», після чого розраховує прибуток. Отримані дані можуть відображатися на головній сторінці системи або в розділі звітності.

Для зручності користувача фінансові операції доцільно фільтрувати за датою, типом і сумою. Наприклад, користувач може переглянути лише доходи за поточний місяць або витрати за певний період. Така можливість є важливою для аналізу діяльності малого бізнесу.

Таблиця 3.8 – Основні елементи сторінки фінансового обліку

Елемент сторінки	Призначення
Кнопка «Додати операцію»	Створення нового фінансового запису
Фільтр за типом	Відбір доходів або витрат
Фільтр за датою	Аналіз операцій за певний період
Таблиця операцій	Перегляд усіх фінансових записів
Поле суми	Відображення розміру доходу або витрати
Опис операції	Пояснення змісту фінансового запису
Пов'язане замовлення	Перегляд продажу, що сформував дохід
Пов'язаний контрагент	Перегляд постачальника або партнера
Підсумковий блок	Відображення доходів, витрат і прибутку

Модуль фінансового обліку тісно пов'язаний із модулем замовлень. Після оплати замовлення система автоматично створює дохідну операцію. Також модуль фінансів пов'язаний із модулем контрагентів, оскільки витрати можуть бути пов'язані з постачальниками або партнерами.

3.8. Реалізація модуля моделювання та аналізу бізнес-процесів

Модуль моделювання та аналізу бізнес-процесів є важливою частиною інформаційної системи, оскільки він дозволяє не лише вести облік даних, а й відображати логіку управління малим бізнесом. Завдяки цьому користувач може бачити, з яких етапів складаються основні процеси підприємства, які дії виконуються в межах кожного процесу та який результат очікується після їх завершення.

У межах розроблюваної системи бізнес-процес розглядається як послідовність взаємопов'язаних дій, спрямованих на виконання певної управлінської задачі. Наприклад, процес обробки замовлення включає вибір клієнта, додавання товарів або послуг, перевірку складських залишків, розрахунок вартості, фіксацію оплати та оновлення звітних даних. Такий підхід дозволяє формалізувати роботу підприємства і зробити її більш зрозумілою для користувача.

Основними функціями модуля моделювання бізнес-процесів є:

- створення нового бізнес-процесу;
- перегляд списку бізнес-процесів;
- редагування опису процесу;
- визначення статусу процесу;
- збереження етапів процесу;
- аналіз виконання процесів;
- зв'язок бізнес-процесів із замовленнями, фінансами або складом;
- використання процесів для формування управлінських висновків.

У базовій версії системи для зберігання бізнес-процесів використовується таблиця `business_processes`. У ній зберігається назва процесу, опис, статус, дата створення та користувач, який створив або контролює процес.

Приклад SQL-структури таблиці бізнес-процесів:

```
CREATE TABLE business_processes (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  name TEXT NOT NULL,
  description TEXT,
  status TEXT NOT NULL,
  created_at TEXT NOT NULL,
  user_id INTEGER,
  FOREIGN KEY (user_id) REFERENCES users(id)
);
```

У цій таблиці поле `id` є первинним ключем. Поле `name` містить назву бізнес-процесу, `description` — його опис, `status` — поточний стан, `created_at` — дату створення, а `user_id` може використовуватися для зв'язку з користувачем системи.

Для більш детального моделювання можна також передбачити таблицю етапів бізнес-процесу. Вона дозволяє зберігати не лише загальний опис процесу, а й окремі кроки його виконання.

```
CREATE TABLE process_steps (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    process_id INTEGER NOT NULL,
    step_name TEXT NOT NULL,
    step_order INTEGER NOT NULL,
    description TEXT,
    FOREIGN KEY (process_id) REFERENCES business_processes(id)
);
```

Таблиця process_steps дозволяє описати послідовність дій у межах конкретного процесу. Наприклад, для процесу «Обробка замовлення» можна створити етапи: «Вибір клієнта», «Додавання товарів», «Перевірка складу», «Фіксація оплати», «Формування звіту».

Приклад маршруту для перегляду бізнес-процесів:

```
@app.route("/processes")
def processes_list():
    processes = BusinessProcess.query.all()
    return render_template("processes.html", processes=processes)
```

Цей маршрут отримує всі бізнес-процеси з бази даних і передає їх до HTML-шаблону. На сторінці користувач може переглянути назву процесу, опис, статус і дату створення.

Додавання нового бізнес-процесу реалізується через окрему форму. Користувач вводить назву, опис і вибирає статус процесу.

```
@app.route("/processes/add", methods=["GET", "POST"])
def add_process():
    if request.method == "POST":
        name = request.form["name"]
        description = request.form.get("description")
        status = request.form["status"]

        process = BusinessProcess(
            name=name,
            description=description,
            status=status,
            created_at=datetime.now().strftime("%d.%m.%Y")
        )

        db.session.add(process)
        db.session.commit()
```

```
return redirect(url_for("processes_list"))
```

```
return render_template("process_form.html")
```

У цьому фрагменті коду після заповнення форми створюється новий об'єкт BusinessProcess, який зберігається в базі даних. Після цього користувач повертається до списку бізнес-процесів.

Для бізнес-процесів доцільно використовувати такі статуси:

Таблиця 3.9 – Статуси бізнес-процесів у системі

Статус	Значення
Запланований	Процес створений, але ще не виконується
Активний	Процес виконується в поточний момент
Завершений	Процес виконано повністю
Призупинений	Процес тимчасово не виконується
Потребує аналізу	Процес завершено, але потребує оцінки ефективності

Для наочного подання бізнес-процесів у системі можна використовувати окрему сторінку, де процеси відображаються у вигляді списку або карток. У кожній картці може бути назва процесу, короткий опис, статус і кнопка для перегляду деталей.

Таблиця 3.10 – Основні елементи сторінки бізнес-процесів

Елемент сторінки	Призначення
Кнопка «Додати процес»	Створення нового бізнес-процесу
Список процесів	Перегляд усіх змодельованих процесів
Статус процесу	Відображення поточного стану процесу
Опис процесу	Пояснення змісту та мети процесу
Етапи процесу	Послідовність дій у межах процесу
Кнопка «Редагувати»	Зміна назви, опису або статусу
Кнопка «Аналізувати»	Перехід до оцінки результатів процесу

Аналіз бізнес-процесів у системі може виконуватися на основі даних інших модулів. Наприклад, ефективність процесу обробки замовлення

можна оцінювати за кількістю виконаних замовлень, сумою продажів, кількістю скасованих замовлень або часткою неоплачених замовлень. Процес складського обліку можна аналізувати за кількістю товарів із низьким залишком або частотою списання товарів.

Для прикладу можна реалізувати функцію аналізу кількості замовлень за статусами:

```
def analyze_orders_by_status():
    result = db.session.query(
        Order.status,
        db.func.count(Order.id)
).group_by(Order.status).all()

    return result
```

Ця функція групує замовлення за статусами та підраховує кількість замовлень у кожній групі. Отримані дані можуть використовуватися для побудови діаграми або короткого аналітичного звіту.

Також можна реалізувати аналіз фінансового результату процесів продажу. Для цього система може підраховувати суму доходів, витрат і прибутку за певний період.

```
def analyze_business_result():
    summary = calculate_financial_summary()

    if summary["profit"] > 0:
        result = "Діяльність підприємства є прибутковою"
    elif summary["profit"] == 0:
        result = "Доходи дорівнюють витратам"
    else:
        result = "Підприємство має збиток"

    return result
```

Цей приклад демонструє просту логіку аналізу фінансового результату. На практиці модуль аналізу може бути розширений за рахунок додаткових показників: середньої суми замовлення, кількості повторних клієнтів, популярності товарів, рівня витрат або динаміки продажів.

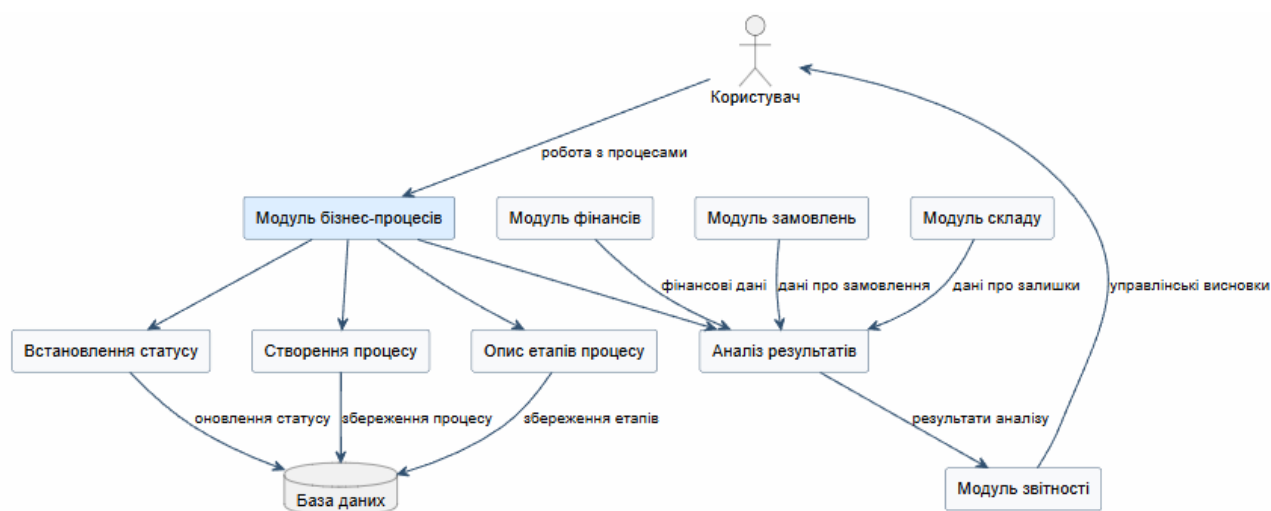


Рисунок 3.1 – Логіка роботи модуля моделювання бізнес-процесів

На рисунку 3.1 подано логіку роботи модуля моделювання бізнес-процесів. Користувач створює бізнес-процес, описує його етапи, встановлює статус і за потреби аналізує результати.

3.9. Реалізація звітності та візуалізації результатів

Модуль звітності та візуалізації результатів призначений для подання узагальненої інформації про діяльність малого бізнесу. Якщо попередні модулі забезпечують введення, зберігання та обробку даних, то модуль звітності дозволяє перетворити ці дані на зрозумілу аналітичну інформацію. Завдяки цьому користувач може оцінити поточний стан підприємства, визначити проблемні місця та приймати обґрунтовані управлінські рішення.

У розроблюваній інформаційній системі звітність формується на основі даних із модулів клієнтів, замовлень, складу, фінансів, товарів і послуг. Основними показниками, які доцільно відображати у звітах, є кількість замовлень, сума продажів, загальні доходи, витрати, прибуток, популярні товари або послуги, активність клієнтів і стан складських залишків.

Основними функціями модуля звітності є:

- перегляд загальної статистики підприємства;
- формування звіту про продажі;
- формування фінансового звіту;
- аналіз популярних товарів і послуг;

- аналіз активності клієнтів;
- перегляд стану складських залишків;
- візуалізація даних у вигляді графіків і діаграм;
- фільтрація звітів за періодом.

Для реалізації звітності у системі використовується окрема сторінка **«Звіти та аналітика»**. На цій сторінці користувач може переглянути ключові показники діяльності малого бізнесу. Наприклад, у верхній частині сторінки можуть бути розміщені інформаційні блоки з показниками: загальна кількість замовлень, сума доходів, сума витрат і прибуток.

Приклад функції для формування загальної статистики:

```
def get_dashboard_statistics():
    orders_count = Order.query.count()

    income = db.session.query(
        db.func.sum(FinanceOperation.amount)
    ).filter_by(type="Доход").scalar() or 0

    expenses = db.session.query(
        db.func.sum(FinanceOperation.amount)
    ).filter_by(type="Витрати").scalar() or 0

    profit = income - expenses

    clients_count = Client.query.count()

    return {
        "orders_count": orders_count,
        "income": income,
        "expenses": expenses,
        "profit": profit,
        "clients_count": clients_count
    }
```

Ця функція підраховує основні показники системи: кількість замовлень, доходи, витрати, прибуток і кількість клієнтів. Отримані дані можуть відображатися на головній сторінці або в модулі звітності.

Для перегляду сторінки звітів у Flask-застосунку може використовуватися такий маршрут:

```
@app.route("/reports")
def reports():
    statistics = get_dashboard_statistics()
    return render_template("reports.html", statistics=statistics)
```

У цьому прикладі маршрут /reports отримує статистичні дані та передає їх до шаблону reports.html, де вони відображаються користувачу.

Важливою частиною звітності є аналіз продажів. Система повинна показувати, які товари або послуги користуються найбільшим попитом. Це дозволяє керівнику малого бізнесу визначати найприбутковіші позиції та планувати подальший розвиток асортименту.

Приклад функції аналізу популярних товарів:

```
def get_popular_products():
    result = db.session.query(
        Product.name,
        db.func.sum(OrderItem.quantity).label("total_quantity")
    ).join(OrderItem, Product.id == OrderItem.product_id) \
        .group_by(Product.name) \
        .order_by(db.desc("total_quantity")) \
        .limit(5) \
        .all()

    return result
```

Ця функція підраховує кількість проданих одиниць кожного товару або послуги та повертає п'ять найпопулярніших позицій. Такі дані можуть використовуватися для побудови діаграми популярності товарів.

Також доцільно реалізувати аналіз доходів і витрат. Він дозволяє оцінити фінансовий стан підприємства та визначити, чи є діяльність прибутковою.

```
def get_finance_report():
    operations = FinanceOperation.query.all()

    income = sum(op.amount for op in operations if op.type == "Дохід")
    expenses = sum(op.amount for op in operations if op.type == "Витрати")
    profit = income - expenses

    return {
        "income": income,
        "expenses": expenses,
        "profit": profit
    }
```

Результати фінансового звіту можуть відображатися у вигляді таблиці або діаграми. Наприклад, можна побудувати стовпчикову діаграму, де окремо показано доходи, витрати та прибуток.

Для візуалізації результатів у вебінтерфейсі використовується бібліотека **Chart.js**. Вона дозволяє створювати графіки та діаграми без складної серверної обробки зображень. Дані передаються з Flask-застосунку до HTML-шаблону, після чого JavaScript будує відповідний графік.

Приклад фрагмента HTML-коду для побудови графіка:

```
<canvas id="financeChart"></canvas>

<script>
const ctx = document.getElementById('financeChart');

new Chart(ctx, {
  type: 'bar',
  data: {
    labels: ['Доходи', 'Витрати', 'Прибуток'],
    datasets: [{
      label: 'Фінансові показники',
      data: [incomeValue, expensesValue, profitValue]
    }]
  }
});
</script>
```

У цьому прикладі створюється стовпчикова діаграма фінансових показників. Значення `incomeValue`, `expensesValue` і `profitValue` передаються із серверної частини та використовуються для побудови графіка.

Для звітності доцільно передбачити кілька видів візуалізації:

Таблиця 3.11 – Основні види звітів інформаційної системи

Вид звіту	Джерело даних	Призначення
Звіт про продажі	Замовлення, позиції замовлень	Аналіз кількості та суми продажів
Фінансовий звіт	Фінансові операції	Оцінка доходів, витрат і прибутку
Звіт про популярні товари	Товари, позиції замовлень	Визначення найпопулярніших позицій
Звіт про клієнтів	Клієнти, замовлення	Аналіз активності клієнтів

Вид звіту	Джерело даних	Призначення
Складський звіт	Товари, склади, залишки	Контроль наявності товарів
Звіт про бізнес-процеси	Бізнес-процеси, замовлення, фінанси	Оцінка ефективності процесів

На сторінці звітів користувач повинен мати можливість обрати потрібний тип звіту та період аналізу. Наприклад, можна переглянути продажі за поточний місяць, фінансовий результат за квартал або список популярних товарів за весь період роботи системи.

Таблиця 3.12 – Елементи сторінки звітності та аналітики

Елемент сторінки	Призначення
Блок ключових показників	Відображення кількості замовлень, доходів, витрат і прибутку
Фільтр періоду	Вибір проміжку часу для аналізу
Графік доходів і витрат	Візуальне порівняння фінансових показників
Діаграма популярних товарів	Визначення товарів або послуг з найбільшим попитом
Таблиця звітних даних	Детальний перегляд показників
Кнопка оновлення	Повторне формування звіту після зміни параметрів

3.10. Висновки до розділу 3

У третьому розділі бакалаврської роботи було розглянуто програмну реалізацію інформаційної системи моделювання процесів управління малим бізнесом. Основну увагу приділено опису функціональних модулів системи, їх призначенню, структурі бази даних, прикладам програмного коду та взаємозв'язкам між окремими компонентами.

На початку розділу було описано налаштування календарів, валют та основних параметрів системи. Визначено, що для коректної роботи інформаційної системи необхідно встановити єдині правила відображення дат, фінансових сум, статусів замовлень, типів фінансових операцій і

параметрів звітності. Це забезпечує узгодженість роботи всіх модулів системи.

Далі було розглянуто реалізацію модуля номенклатури товарів і послуг. Цей модуль забезпечує створення, перегляд, редагування та використання позицій номенклатури під час оформлення замовлень. Було визначено, що товари можуть бути пов'язані зі складським обліком, а послуги використовуються у замовленнях без списання зі складу.

РОЗДІЛ 4 ТЕСТУВАННЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ РОЗРОБЛЕНОЇ СИСТЕМИ

4.1. Підготовка тестових даних

Тестування інформаційної системи є важливим етапом розробки, оскільки воно дозволяє перевірити правильність роботи окремих модулів, виявити можливі помилки та оцінити відповідність системи поставленим вимогам. Для інформаційної системи моделювання процесів управління малим бізнесом тестування повинно охоплювати роботу з клієнтами, контрагентами, товарами, послугами, складами, замовленнями, фінансовими операціями, бізнес-процесами та звітністю.

Перед початком тестування необхідно підготувати набір тестових даних. Такі дані повинні імітувати реальну діяльність малого підприємства та дозволяти перевірити основні сценарії роботи системи. До тестових даних належать записи про клієнтів, постачальників, товари, послуги, склади, залишки, замовлення, оплати, витрати та бізнес-процеси.

Для тестування було обрано умовне мале підприємство, яке займається продажем товарів і наданням супутніх послуг. Це дозволяє перевірити роботу системи як з товарами, що потребують складського обліку, так і з послугами, які не списуються зі складу. Такий підхід є зручним для демонстрації основної логіки інформаційної системи.

На першому етапі були підготовлені тестові клієнти. Для кожного клієнта вказується ім'я або назва, телефон, електронна пошта, адреса та примітка. Ці дані використовуються під час створення замовлень і подальшого аналізу активності клієнтів.

Таблиця 4.1 – Тестові дані клієнтів

№	Клієнт	Телефон	Електронна пошта	Адреса
1	Іваненко Олексій	+380501112233	ivanenko@example.com	м. Київ
2	Петренко Марина	+380671234567	petrenko@example.com	м. Львів
3	ТОВ «Комфорт Плюс»	+380931112244	comfort@example.com	м. Дніпро
4	ФОП Савченко Ірина	+380991234321	savchenko@example.com	м. Кропивницький

На другому етапі були підготовлені дані про контрагентів. Вони використовуються для перевірки модуля обліку постачальників і формування витрат у фінансовому модулі.

Таблиця 4.2 – Тестові дані контрагентів

№	Контрагент	Тип	Телефон	Призначення
1	ТОВ «Опт Сервіс»	Постачальник	+380442223344	Постачання товарів
2	ФОП Кравець Андрій	Підрядник	+380672221100	Сервісні роботи
3	ТОВ «Логістика Груп»	Партнер	+380502229988	Доставка замовлень

Наступним етапом є підготовка номенклатури товарів і послуг. Для перевірки системи доцільно використати кілька товарів із різних категорій і кілька послуг. Це дозволяє перевірити, чи правильно система розрізняє товари та послуги під час створення замовлень.

Таблиця 4.3 – Тестова номенклатура товарів і послуг

№	Назва	Категорія	Тип	Ціна, грн	Одиниця
1	Ноутбук офісний	Техніка	Товар	18500	шт.
2	Принтер лазерний	Техніка	Товар	7200	шт.

№	Назва	Категорія	Тип	Ціна, грн	Одиниця
3	Папір офісний А4	Канцелярія	Товар	280	пачка
4	Налаштування обладнання	Послуги	Послуга	900	послуга
5	Консультація спеціаліста	Послуги	Послуга	600	година

Після створення номенклатури необхідно підготувати складські залишки. Для товарів вказується кількість на основному складі. Послуги не додаються до складських залишків, оскільки вони не потребують фізичного зберігання.

Таблиця 4.4 – Тестові складські залишки

№	Товар	Склад	Кількість
1	Ноутбук офісний	Основний склад	8
2	Принтер лазерний	Основний склад	5
3	Папір офісний А4	Основний склад	40

Для перевірки модуля замовлень підготовлено кілька тестових сценаріїв. Вони повинні охоплювати стандартне замовлення, замовлення з послугою, замовлення з оплатою, а також ситуацію, коли кількість товару на складі є недостатньою.

Таблиця 4.5 – Тестові сценарії замовлень

№	Сценарій	Очікуваний результат
1	Створення замовлення з одним товаром	Замовлення створюється, сума розраховується автоматично
2	Додавання товару і послуги до замовлення	Товар перевіряється на складі, послуга додається без перевірки
3	Фіксація оплати за замовленням	Створюється фінансова операція типу «Дохід»
4	Додавання товару в кількості, більшій за залишок	Система повідомляє про недостатню кількість товару

№	Сценарій	Очікуваний результат
5	Скасування замовлення	Статус замовлення змінюється на «Скасоване»

Також необхідно підготувати фінансові операції. Частина доходів формується автоматично після оплати замовлень, а витрати можуть додаватися вручну користувачем. Це дозволяє перевірити розрахунок доходів, витрат і прибутку.

Таблиця 4.6 – Тестові фінансові операції

№	Тип операції	Сума, грн	Опис
1	Дохід	19400	Оплата замовлення з товаром і послугою
2	Дохід	7200	Оплата замовлення за принтер
3	Витрата	5000	Закупівля канцелярських товарів
4	Витрата	1200	Оплата доставки
5	Витрата	3000	Сервісне обслуговування обладнання

Окремо були підготовлені тестові бізнес-процеси. Вони використовуються для перевірки модуля моделювання та аналізу бізнес-процесів.

Таблиця 4.7 – Тестові бізнес-процеси

№	Назва процесу	Статус	Основні етапи
1	Обробка замовлення	Активний	Клієнт → товари → склад → оплата → звіт
2	Закупівля товарів	Запланований	Постачальник → товар → склад → витрата
3	Аналіз продажів	Завершений	Замовлення → фінанси → звіт → висновок

Підготовлений набір тестових даних дозволяє перевірити роботу всіх основних модулів інформаційної системи. Завдяки цим даним можна змодельовати реальні ситуації, які виникають у діяльності малого бізнесу:

створення клієнта, оформлення замовлення, перевірку складу, фіксацію оплати, облік витрат і формування звітів.

4.2. Тестування налаштувань календарів, валют і номенклатури

Тестування налаштувань календарів, валют і номенклатури виконується для перевірки правильності початкових параметрів інформаційної системи. Цей етап є важливим, оскільки базові налаштування використовуються в інших модулях: замовленнях, фінансовому обліку, звітності та складському контролі.

Метою тестування є перевірка того, чи коректно система відображає дати, фінансові суми, валюту обліку, статуси замовлень, типи фінансових операцій, а також чи правильно працює модуль товарів і послуг.

На першому етапі перевіряється відображення календарних даних. Для цього в системі створюються записи із зазначенням дати: замовлення, фінансові операції та бізнес-процеси. Після збереження записів перевіряється, чи правильно дата відображається у списках, формах і звітах.

Таблиця 4.8 – Перевірка календарних налаштувань

№	Тестова дія	Очікуваний результат	Результат
1	Створення нового замовлення	Дата створення зберігається автоматично	Виконано
2	Додавання фінансової операції	Дата операції відображається у встановленому форматі	Виконано
3	Створення бізнес-процесу	Дата створення процесу зберігається в системі	Виконано
4	Перегляд звіту за період	Дані фільтруються відповідно до вибраного періоду	Виконано

Результати тестування показали, що система коректно працює з датами. Дата створення замовлення та бізнес-процесу фіксується автоматично, а фінансові операції можуть створюватися із зазначенням конкретної дати. Це дозволяє використовувати календарні дані для подальшого аналізу діяльності підприємства.

На другому етапі було перевірено відображення валюти та фінансових сум. Оскільки основною валютою системи є гривня, усі суми повинні відображатися у гривнях. Для перевірки було створено кілька товарів, замовлень і фінансових операцій із різними сумами.

Таблиця 4.9 – Перевірка валютних налаштувань

№	Тестова дія	Очікуваний результат	Результат
1	Додавання товару з ціною 18500 грн	Ціна зберігається та відображається коректно	Виконано
2	Створення замовлення з кількома позиціями	Загальна сума розраховується автоматично	Виконано
3	Фіксація оплати замовлення	Сума оплати передається до фінансового модуля	Виконано
4	Перегляд фінансового звіту	Доходи, витрати й прибуток відображаються у гривнях	Виконано

Під час перевірки встановлено, що система коректно обробляє фінансові значення. Ціна товару зберігається в базі даних, загальна сума замовлення розраховується автоматично, а після оплати створюється фінансова операція типу «Дохід». Це підтверджує правильність роботи валютних налаштувань.

На третьому етапі було протестовано модуль номенклатури товарів і послуг. Для цього в систему були додані тестові позиції: товари, які потребують складського обліку, та послуги, які не списуються зі складу. Основна увага приділялася перевірці додавання, редагування, перегляду та використання номенклатури під час створення замовлень.

Таблиця 4.10 – Перевірка модуля номенклатури товарів і послуг

№	Тестова дія	Очікуваний результат	Результат
1	Додавання нового товару	Товар зберігається у списку номенклатури	Виконано
2	Додавання нової послуги	Послуга зберігається та доступна для вибору	Виконано
3	Редагування ціни товару	Нова ціна зберігається в системі	Виконано
4	Видалення тестової	Запис видаляється або стає	Виконано

№	Тестова дія	Очікуваний результат	Результат
	позиції	недоступним	
5	Додавання товару до замовлення	Система перевіряє складський залишок	Виконано
6	Додавання послуги до замовлення	Послуга додається без перевірки складу	Виконано

Результати тестування показали, що модуль номенклатури працює відповідно до поставлених вимог. Користувач може додавати товари й послуги, змінювати їх параметри та використовувати ці позиції під час створення замовлень. Система правильно розрізняє товари та послуги: для товарів виконується перевірка складу, а для послуг така перевірка не застосовується.

Окремо було перевірено правильність автоматичного розрахунку суми позиції замовлення. Для цього до замовлення було додано товар із заданою ціною та кількістю. Система повинна була самостійно розрахувати суму за формулою:

сума позиції = ціна × кількість

Таблиця 4.11 – Перевірка розрахунку суми позиції

Назва позиції	Ціна, грн	Кількість	Очікувана сума, грн	Результат
Ноутбук офісний	18500	1	18500	Виконано
Папір офісний А4	280	5	1400	Виконано
Налаштування обладнання	900	1	900	Виконано

Під час тестування було підтверджено, що система правильно розраховує суму за кожною позицією та додає її до загальної суми замовлення. Це зменшує кількість ручних розрахунків і знижує ймовірність помилок.

Пошук...

Категорія: Усі

Тип: Усі

Статус: Активні

Оновити

№	Назва	Категорія	Тип	Ціна, грн	Одиниця	Статус	Дії
1	Ноутбук офісний	Техніка	Товар	18 500,00	шт.	Активний	Редагувати Видалити
2	Принтер лазерний	Техніка	Товар	7 200,00	шт.	Активний	Редагувати Видалити
3	Папір офісний А4	Канцелярія	Товар	280,00	пачка	Активний	Редагувати Видалити
4	Налаштування обладнання	Послуги	Послуга	900,00	послуга	Активний	Редагувати Видалити
5	Консультація спеціаліста	Послуги	Послуга	600,00	година	Активний	Редагувати Видалити

Показано 1 - 5 з 5 записів

<

1

>

Рисунок 4.1 – Сторінка номенклатури товарів і послуг

На рисунку 4.1 наведено сторінку номенклатури товарів і послуг. На сторінці відображаються основні дані про позиції: назва, категорія, тип, ціна та одиниця виміру. Користувач може додати нову позицію, редагувати існуючу або використати її під час створення замовлення. Така структура сторінки забезпечує зручну роботу з асортиментом малого підприємства.

4.3. Тестування роботи з клієнтами та контрагентами

Тестування роботи з клієнтами та контрагентами виконується для перевірки правильності функціонування довідників, які використовуються в основних бізнес-процесах інформаційної системи. Дані про клієнтів застосовуються під час створення замовлень, а дані про контрагентів — під час обліку витрат, закупівель і взаємодії з постачальниками.

Метою тестування є перевірка можливості додавання, перегляду, редагування, пошуку та видалення записів, а також перевірка правильності використання цих даних в інших модулях системи.

На першому етапі було протестовано модуль обліку клієнтів. Для цього в систему були внесені тестові записи про фізичних осіб, підприємців та організації. Після додавання записів перевірялося, чи коректно вони відображаються у списку клієнтів і чи доступні для вибору під час створення замовлення.

Таблиця 4.12 – Результати тестування модуля клієнтів

№	Тестова дія	Очікуваний результат	Результат
1	Додавання нового клієнта	Клієнт зберігається в базі даних	Виконано
2	Перегляд списку клієнтів	Усі клієнти відображаються в таблиці	Виконано
3	Редагування телефону клієнта	Нові контактні дані зберігаються	Виконано
4	Пошук клієнта за іменем	Система знаходить потрібний запис	Виконано
5	Вибір клієнта під час створення замовлення	Клієнт додається до замовлення	Виконано
6	Видалення тестового клієнта	Запис видаляється або стає недоступним	Виконано

Під час перевірки було встановлено, що модуль клієнтів працює коректно. Після додавання нового клієнта запис одразу відображається у загальному списку. Під час редагування контактної інформації зміни зберігаються в базі даних і відображаються після оновлення сторінки. Пошук дозволяє швидко знайти потрібного клієнта за іменем або контактними даними.

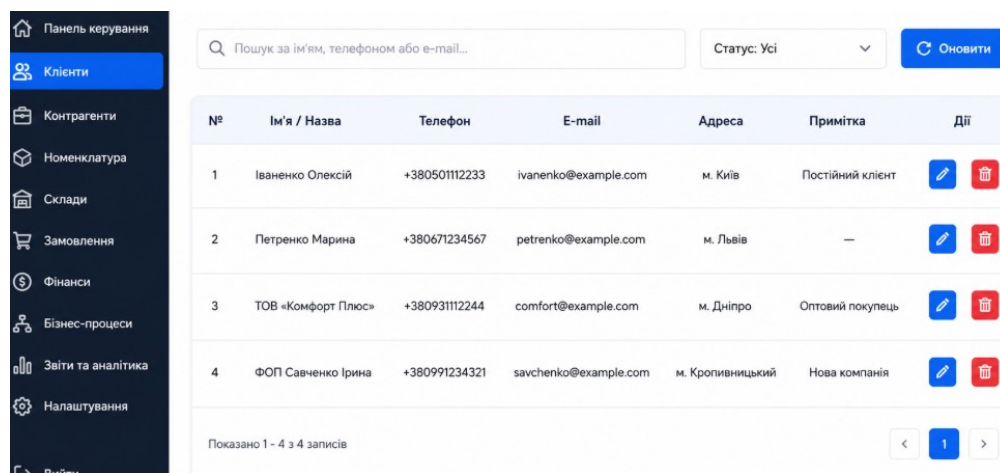


Рисунок 4.3 – Сторінка обліку контрагентів інформаційної системи

Таблиця 4.14 – Перевірка введення обов’язкових полів

№	Тестова дія	Очікуваний результат	Результат
1	Створення клієнта без імені	Система не зберігає порожній запис	Виконано
2	Створення контрагента без назви	Система повідомляє про необхідність заповнення поля	Виконано
3	Введення некоректної електронної пошти	Система дозволяє виправити значення	Виконано
4	Збереження повністю заповненої форми	Запис успішно додається до бази даних	Виконано

Під час тестування було встановлено, що система забезпечує базову перевірку введених даних. Це дозволяє зменшити кількість помилкових записів і підвищити якість інформації, яка зберігається в базі даних.

4.4. Тестування роботи з кількома замовленнями, складами та товарами

Тестування роботи з кількома замовленнями, складами та товарами виконується для перевірки правильності взаємодії між модулями замовлень, номенклатури та складського обліку. Цей етап є важливим, оскільки саме під час оформлення замовлень система повинна коректно використовувати дані про клієнтів, товари, послуги, склади, залишки та оплату.

Метою тестування є перевірка того, чи правильно система обробляє декілька замовлень одночасно, чи коректно розраховує суму продажу, чи оновлює складські залишки після виконання замовлення та чи повідомляє користувача про нестачу товару.

На першому етапі було перевірено створення кількох замовлень для різних клієнтів. Для цього в системі було створено три тестові замовлення з різним складом позицій: одне замовлення містило лише товар, друге — товар і послугу, третє — декілька товарів з різною кількістю.

Таблиця 4.15 – Тестування створення кількох замовлень

№	Клієнт	Склад замовлення	Очікуваний результат	Результат
1	Іваненко Олексій	Ноутбук офісний — 1 шт.	Замовлення створено, сума 18500 грн	Виконано
2	Петренко Марина	Принтер лазерний — 1 шт.; налаштування обладнання — 1 послуга	Замовлення створено, сума 8100 грн	Виконано
3	ТОВ «Комфорт Плюс»	Папір офісний А4 — 5 пачок; консультація спеціаліста — 2 години	Замовлення створено, сума 2600 грн	Виконано

За результатами тестування встановлено, що система коректно створює кілька замовлень і зберігає кожне з них як окремий запис у базі даних. Для кожного замовлення правильно визначається клієнт, дата створення, перелік позицій, кількість, ціна та загальна сума.

На другому етапі було перевірено автоматичний розрахунок суми замовлення. Система повинна підраховувати суму кожної позиції за формулою:

сума позиції = ціна × кількість

Після цього всі суми позицій додаються до загальної суми замовлення.

Таблиця 4.16 – Перевірка розрахунку загальної суми замовлення

№	Позиція	Ціна, грн	Кількість	Сума, грн
1	Принтер лазерний	7200	1	7200
2	Налаштування обладнання	900	1	900
	Загальна сума			8100

Результат перевірки показав, що система правильно розраховує вартість кожної позиції та загальну суму замовлення. Це підтверджує коректність роботи логіки продажів і зменшує потребу у ручних розрахунках.

На третьому етапі було перевірено роботу зі складськими залишками. Для цього до замовлення було додано товар, який є на складі в достатній кількості. Після оформлення замовлення система повинна була зменшити залишок товару.

Таблиця 4.17 – Перевірка оновлення складських залишків після продажу

№	Товар	Початковий залишок	Продано	Очікуваний залишок	Результат
1	Ноутбук офісний	8	1	7	Виконано
2	Принтер лазерний	5	1	4	Виконано
3	Папір офісний А4	40	5	35	Виконано

Після виконання тестових замовлень складські залишки були оновлені відповідно до кількості проданих товарів. Це підтверджує правильність зв'язку між модулем замовлень і модулем складського обліку.

Окремо було перевірено ситуацію, коли користувач намагається додати до замовлення товар у кількості, яка перевищує наявний залишок.

Наприклад, на складі є 4 принтери, а користувач намагається додати до замовлення 10 одиниць. У такому випадку система повинна повідомити про недостатню кількість товару та не дозволити оформити замовлення в некоректному вигляді.

Таблиця 4.18 – Перевірка контролю недостатньої кількості товару

№	Товар	Доступно на складі	Запитана кількість	Очікуваний результат	Результат
1	Принтер лазерний	4	10	Повідомлення про нестачу товару	Виконано
2	Ноутбук офісний	7	12	Замовлення не оформлюється	Виконано
3	Папір офісний А4	35	50	Користувач отримує попередження	Виконано

Результати тестування показали, що система правильно контролює наявність товарів на складі. Якщо залишку недостатньо, користувач отримує повідомлення і може змінити кількість товару або обрати іншу позицію. Це дозволяє уникнути помилок під час продажу та підтримувати актуальність складського обліку.

Також було перевірено роботу з послугами. Оскільки послуги не мають фізичного залишку на складі, система повинна дозволяти додавати їх до замовлення без перевірки складської кількості.

Таблиця 4.19 – Перевірка додавання послуг до замовлення

№	Послуга	Кількість	Очікуваний результат	Результат
1	Налаштування обладнання	1	Послуга додається без перевірки складу	Виконано
2	Консультація спеціаліста	2	Сума розраховується за кількістю годин	Виконано

Під час тестування було підтверджено, що система правильно розрізняє товари та послуги. Для товарів виконується перевірка залишків, а послуги додаються до замовлення без складського списання.

Панель керування

Клієнти

Контрагенти

Номенклатура

Склади

Замовлення

Фінанси

Бізнес-процеси

Звіти та аналітика

Налаштування

Пошук товару або складу...

Склад: Усі

Оновити залишок

№	Назва товару	Склад	Категорія	Кількість	Одиниця	Статус	Дії
1	Ноутбук офісний	Основний склад	Техніка	7	шт.	В наявності	
2	Принтер лазерний	Основний склад	Техніка	4	шт.	В наявності	
3	Папір офісний A4	Основний склад	Канцелярія	35	пачка	Достатньо	
4	Клавіатура USB	Резервний склад	Периферія	2	шт.	Мало	
5	Мишка бездротова	Основний склад	Периферія	0	шт.	Немає	
Показано 1 - 5 з 5 записів							

<

1

>

Рисунок 4.5 – Сторінка складських залишків інформаційної системи

4.5. Перевірка роботи модулів обліку та управління

Перевірка роботи модулів обліку та управління виконується для оцінки загальної працездатності інформаційної системи. На цьому етапі аналізується не один окремий модуль, а їх спільна робота в межах загального процесу управління малим бізнесом. Це дозволяє перевірити, чи правильно система об'єднує клієнтів, контрагентів, товари, склади, замовлення, фінанси, бізнес-процеси та звітність.

На першому етапі було перевірено взаємодію модуля клієнтів і модуля замовлень. Для цього було створено нового клієнта, після чого його було використано під час оформлення тестового замовлення.

Таблиця 4.20 – Перевірка взаємодії клієнтів і замовлень

№	Тестова дія	Очікуваний результат	Результат
1	Додавання нового клієнта	Клієнт зберігається в базі даних	Виконано
2	Створення нового замовлення	Клієнт доступний для вибору	Виконано
3	Збереження замовлення	Замовлення пов'язується з клієнтом	Виконано
4	Перегляд замовлень клієнта	Відображається історія замовлень	Виконано

Результати перевірки показали, що зв'язок між клієнтами та замовленнями працює коректно. Система дозволяє використовувати клієнтську базу під час оформлення продажів і зберігати історію взаємодії з кожним клієнтом.

На другому етапі було перевірено взаємодію модулів товарів, складу та замовлень. Для цього в номенклатуру було додано товар, після чого для нього було створено складський залишок. Далі товар було додано до замовлення, а після оформлення продажу перевірено зміну залишку.

Таблиця 4.21 – Перевірка взаємодії товарів, складу та замовлень

№	Тестова дія	Очікуваний результат	Результат
1	Додавання товару до номенклатури	Товар зберігається в системі	Виконано
2	Додавання товару на склад	Залишок товару відображається на сторінці складу	Виконано
3	Додавання товару до замовлення	Система перевіряє наявність товару	Виконано
4	Оформлення продажу	Залишок товару зменшується	Виконано

Перевірка підтвердила, що система правильно обробляє рух товарів. Після продажу кількість товару на складі автоматично оновлюється, що дозволяє підтримувати актуальний облік залишків.

На третьому етапі було перевірено зв'язок між замовленнями та фінансовим обліком. Після фіксації оплати за замовленням система повинна створити фінансову операцію типу «Дохід». Це дозволяє автоматично враховувати продажі у фінансовій частині системи.

Таблиця 4.22 – Перевірка взаємодії замовлень і фінансового обліку

№	Тестова дія	Очікуваний результат	Результат
1	Створення замовлення	Замовлення зберігається зі статусом «Нове»	Виконано
2	Додавання позицій	Загальна сума розраховується автоматично	Виконано
3	Фіксація оплати	Статус змінюється на «Оплачене»	Виконано
4	Перевірка фінансового модуля	Створюється операція типу «Дохід»	Виконано

За результатами перевірки встановлено, що фінансовий модуль коректно отримує дані із замовлень. Це дозволяє уникнути дублювання інформації та зменшує кількість ручних операцій під час ведення обліку.

4.6. Тестування модуля моделювання бізнес-процесів

Тестування модуля моделювання бізнес-процесів виконувалося для перевірки можливості створення, перегляду, редагування та аналізу

процесів управління малим бізнесом. Основна увага приділялася процесу обробки замовлення, оскільки він пов'язаний з клієнтами, товарами, складом, оплатою та звітністю.

Для перевірки було створено тестовий бізнес-процес «**Обробка замовлення**». У ньому були визначені основні етапи: вибір клієнта, додавання товарів або послуг, перевірка складу, оформлення замовлення, фіксація оплати та формування звіту.

Таблиця 4.26 – Тестування модуля бізнес-процесів

№	Тестова дія	Очікуваний результат	Результат
1	Створення бізнес-процесу	Процес додається до системи	Виконано
2	Додавання опису процесу	Опис зберігається	Виконано
3	Встановлення статусу	Статус відображається коректно	Виконано
4	Редагування процесу	Зміни зберігаються	Виконано
5	Перегляд етапів процесу	Етапи відображаються у правильному порядку	Виконано

Результати тестування показали, що модуль бізнес-процесів працює коректно. Користувач може створювати процеси, описувати їхню логіку, змінювати статус і переглядати основні етапи. Це дозволяє використовувати систему не лише для обліку, а й для формалізації управлінських дій.

Також було перевірено зв'язок бізнес-процесів з іншими модулями системи. Наприклад, процес обробки замовлення використовує дані клієнтів, товарів, складу, фінансів і звітності. Завдяки цьому можна оцінити, як окремий процес впливає на роботу малого підприємства.

Таблиця 4.27 – Перевірка зв'язку бізнес-процесу з модулями системи

Модуль	Роль у бізнес-процесі
Клієнти	Вибір клієнта для замовлення
Товари та послуги	Додавання позицій до замовлення
Склад	Перевірка наявності товару

Модуль	Роль у бізнес-процесі
Фінанси	Фіксація оплати
Звітність	Аналіз результату процесу

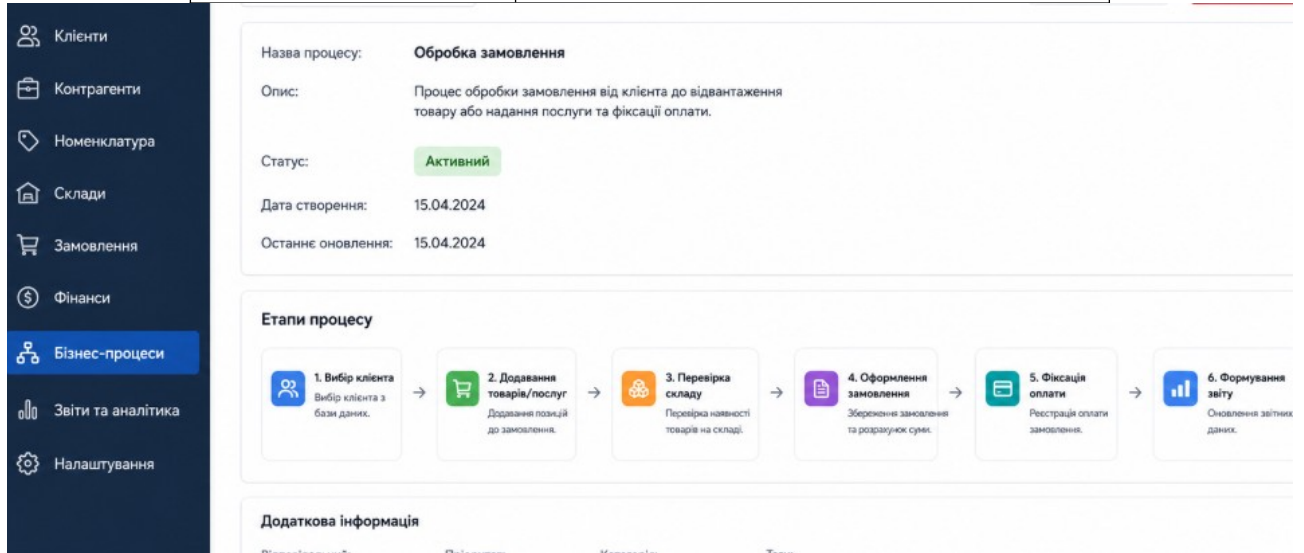


Рисунок 4.6 – Сторінка моделювання бізнес-процесів

На рисунку 4.6 наведено сторінку моделювання бізнес-процесів. На сторінці відображаються назва процесу, його статус, опис та основні етапи виконання. Такий інтерфейс дозволяє користувачу контролювати логіку управлінських процесів і використовувати їх для аналізу діяльності підприємства.

4.7. Аналіз результатів роботи системи

Після тестування основних модулів було виконано аналіз результатів роботи інформаційної системи. Перевірка показала, що система забезпечує централізоване зберігання даних, автоматизацію обліку та підтримку основних процесів малого бізнесу.

У процесі тестування система коректно виконувала такі дії:

- додавання клієнтів і контрагентів;
- ведення номенклатури товарів і послуг;
- облік складських залишків;
- створення замовлень;
- автоматичний розрахунок суми продажу;

- фіксація оплат;
- створення фінансових операцій;
- формування звітних показників;
- моделювання бізнес-процесів.

Таблиця 4.28 – Основні результати роботи системи

Напрямок перевірки	Результат
Облік клієнтів	Дані зберігаються та використовуються в замовленнях
Облік товарів	Номенклатура працює коректно
Складський облік	Залишки оновлюються після продажу
Замовлення	Сума розраховується автоматично
Фінанси	Доходи й витрати враховуються правильно
Звітність	Показники формуються на основі даних системи
Бізнес-процеси	Процеси створюються та аналізуються

Аналіз показав, що система дозволяє зменшити кількість ручних операцій. Наприклад, під час створення замовлення користувачу не потрібно окремо розраховувати суму, перевіряти залишки в іншому файлі або вручну створювати фінансовий запис. Основні дії виконуються в єдиному середовищі.

4.8. Оцінка ефективності використання системи для малого бізнесу

Оцінка ефективності використання інформаційної системи виконується для визначення її практичної користі для малого підприємства. Основними критеріями оцінки є зменшення кількості ручних операцій, прискорення обробки інформації, підвищення точності обліку та покращення контролю за бізнес-процесами.

До впровадження інформаційної системи частина операцій могла виконуватися вручну або за допомогою окремих електронних таблиць. Це ускладнювало пошук даних, контроль замовлень, перевірку залишків і формування звітності. Після використання системи всі основні дані

зберігаються в єдиній базі, а більшість операцій виконується через вебінтерфейс.

Таблиця 4.29 – Порівняння роботи до та після використання системи

Критерій	До використання системи	Після використання системи
Облік клієнтів	Окремі списки або таблиці	Єдина клієнтська база
Облік товарів	Ручне ведення номенклатури	Централізований довідник
Замовлення	Ручне оформлення	Автоматизоване створення
Склад	Окрема перевірка залишків	Автоматична перевірка
Фінанси	Ручний підрахунок	Автоматичний облік доходів і витрат
Звітність	Формування вручну	Автоматичне отримання показників

Використання системи дозволяє скоротити час на виконання типових операцій. Наприклад, створення замовлення виконується швидше, оскільки користувач обирає клієнта й товари зі списку, а система самостійно розраховує суму та перевіряє залишки на складі.

Також підвищується точність обліку. Система автоматично пов'язує замовлення з клієнтом, товари зі складом, оплату з фінансовим модулем, а дані продажів — зі звітністю. Це зменшує ризик помилок і дублювання інформації.

Окремою перевагою є можливість аналізу діяльності підприємства. Користувач може переглядати доходи, витрати, прибуток, кількість замовлень, популярні товари та стан складських залишків. Це дозволяє приймати більш обґрунтовані управлінські рішення.

Таблиця 4.30 – Очікуваний ефект від використання системи

Напрямок	Очікуваний ефект
Облік даних	Зменшення дублювання інформації
Продажі	Прискорення створення замовлень
Склад	Актуальний контроль залишків
Фінанси	Швидкий розрахунок доходів, витрат і прибутку
Звітність	Спрощення аналізу діяльності
Управління	Покращення прийняття рішень

Отже, використання інформаційної системи є ефективним для малого бізнесу, оскільки вона об'єднує основні процеси в одному середовищі, спрощує облік, підвищує точність даних і забезпечує користувача аналітичною інформацією.

4.9. Висновки до розділу 4

У четвертому розділі було проведено тестування розробленої інформаційної системи та оцінено ефективність її використання для малого бізнесу. Для перевірки роботи системи було підготовлено тестові дані: клієнтів, контрагентів, товари, послуги, складські залишки, замовлення, фінансові операції та бізнес-процеси.

Було перевірено роботу базових налаштувань системи, зокрема календарів, валюти, статусів замовлень і номенклатури товарів та послуг. Результати тестування підтвердили, що система коректно працює з датами, фінансовими сумами та типами позицій.

Також було протестовано модулі клієнтів і контрагентів. Перевірка показала, що система дозволяє додавати, редагувати, переглядати та

використовувати ці дані в інших модулях. Дані клієнтів застосовуються під час створення замовлень, а контрагенти використовуються у фінансових операціях.

ВИСНОВКИ

У бакалаврській роботі було розглянуто питання розробки інформаційної системи моделювання процесів управління малим бізнесом. Актуальність теми зумовлена тим, що малі підприємства часто виконують облік клієнтів, товарів, замовлень, складу та фінансів за допомогою окремих таблиць або ручних записів. Такий підхід ускладнює контроль діяльності, збільшує ризик помилок і не дозволяє швидко отримувати аналітичну інформацію для прийняття управлінських рішень.

У першому розділі було проаналізовано предметну область малого бізнесу. Розглянуто поняття малого бізнесу, його особливості, основні процеси управління та роль інформаційних систем у діяльності підприємства. Було встановлено, що для малого бізнесу важливими є простота, доступність і функціональність програмного рішення. Також було виконано аналіз існуючих інформаційних систем і визначено необхідність моделювання бізнес-процесів для підвищення ефективності управління.

У другому розділі було виконано проєктування інформаційної системи. Сформульовано постановку задачі, визначено функціональні та нефункціональні вимоги, розроблено загальну архітектуру системи, діаграму варіантів використання, діаграму діяльності, діаграму послідовності та ER-діаграму бази даних. Також було спроектовано структуру інтерфейсу користувача. У результаті було сформовано логічну основу майбутньої програмної реалізації.

У третьому розділі було описано програмну реалізацію інформаційної системи. Для розробки системи було обрано мову програмування Python, вебфреймворк Flask, базу даних SQLite, а також технології HTML, CSS і JavaScript для створення вебінтерфейсу. У роботі описано реалізацію основних модулів: налаштувань системи, номенклатури товарів і послуг, клієнтів, контрагентів, складів, замовлень, фінансового обліку, бізнес-процесів, звітності та візуалізації результатів.

У четвертому розділі було проведено тестування розробленої системи. Для перевірки були підготовлені тестові дані, які імітують роботу малого підприємства. Було перевірено коректність роботи з клієнтами,

контрагентами, товарами, послугами, складами, замовленнями, фінансовими операціями та бізнес-процесами. Результати тестування показали, що система правильно зберігає дані, розраховує суми замовлень, оновлює складські залишки, фіксує доходи й витрати та формує аналітичні показники.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Гриньова В. М. Управління малим бізнесом: навчальний посібник. Харків: ХНЕУ, 2020. 216 с.
2. Пономаренко В. С. Інформаційні системи в управлінні підприємством: навчальний посібник. Харків: ХНЕУ, 2019. 312 с.
3. Гаврилова Т. А., Хорошевський В. Ф. Базы знань інтелектуальних систем. Київ: Видавничий дім «КМ Академія», 2020. 384 с.
4. Коваленко О. В. Проєктування інформаційних систем: навчальний посібник. Київ: Центр учбової літератури, 2021. 240 с.
5. Бойко В. В., Савінков В. М. Проєктування баз даних інформаційних систем. Київ: КНТЕУ, 2019. 286 с.
6. Литвин В. В. Моделювання бізнес-процесів: навчальний посібник. Львів: Видавництво Львівської політехніки, 2020. 224 с.
7. Fowler M. UML Distilled: A Brief Guide to the Standard Object Modeling Language. 3rd ed. Boston: Addison-Wesley, 2018. 208 p.
8. Date C. J. An Introduction to Database Systems. 8th ed. Boston: Pearson, 2019. 1024 p.
9. Connolly T., Begg C. Database Systems: A Practical Approach to Design, Implementation, and Management. 6th ed. Boston: Pearson, 2020. 1440 p.
10. Python Software Foundation. Python Documentation. Офіційна документація Python. Python є мовою програмування, використаною для реалізації серверної логіки системи.
11. Pallets Projects. Flask Documentation. Офіційна документація Flask. Flask використовується як легкий WSGI-фреймворк для створення вебзастосунку.
12. SQLite Consortium. SQLite Documentation. Офіційна документація SQLite. SQLite використовується як система керування базою даних у програмній реалізації.
13. Mozilla Developer Network. MDN Web Docs. Документація з HTML, CSS, JavaScript та інших вебтехнологій, які використовуються для створення інтерфейсу користувача.

- 14.Booch G., Rumbaugh J., Jacobson I. The Unified Modeling Language User Guide. 2nd ed. Boston: Addison-Wesley, 2019. 496 p.
- 15.Weske M. Business Process Management: Concepts, Languages, Architectures. 3rd ed. Berlin: Springer, 2019. 417 p.

ДОДАТОК

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ

І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

Декларація про академічну доброчесність та використання ШІ

ДЕКЛАРАЦІЯ ЗДОБУВАЧА

Я, Роздобудько Денис, здобувач(ка) НУБіП України, усвідомлюючи відповідальність за порушення академічної доброчесності, цією заявою підтверджую, що:

1. Моя бакалаврська кваліфікаційна робота (проект) на тему «Інформаційна система моделювання процесів управління малим бізнесом» є результатом моєї особистої праці.
2. Усі запозичення з текстів інших авторів мають відповідні посилання згідно з чинними стандартами.
3. Щодо використання інструментів ШІ зазначаю, що (обрати необхідне):
 - ☐ інструменти генеративного ШІ не використовувалися.
 - ☐ ☒ інструменти ШІ (вказати назву: ChatGPT, Gemini, Claude, DeepL, _____ інші (вказати назву)) були використані для:
 - ☐ перекладу іншомовних джерел;
 - ☐ стилістичного редагування та перевірки граматики;
 - ☐ допомоги у написанні/відлагодженні програмного коду;
 - ☐ інше: _____.

Я розумію, що надання недостовірної інформації може призвести до

скасування результатів захисту та відрахування з числа здобувачів НУБіП України.

«___» _____ 2026 _____ **Денис РОЗДОБУДЬКО**
р. (підпис)