

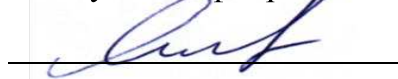
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра інформаційних управляючих систем та технологій

"Допущений до захисту"

Завідувач кафедри ІУСТ



к.т.н, доц. Михелєв І.Л.

" ____ " _____ 2024 р.

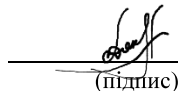
КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти "магістр"

на тему:

Дослідження облікових систем оптової торгівлі

Виконав: студент групи 6141м

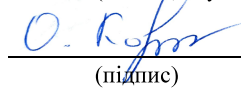

(підпис)

Дякончук О. В.

Керівник роботи:

К.Т.Н, доц.

(посада, науковий ступінь вчене звання)


(підпис)

Гайдаєнко О.В.

м. Миколаїв – 2024 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**Національний університет кораблебудування імені адмірала Макарова**

Навчально-науковий інститут комп'ютерних наук та управління проектами

Кафедра інформаційних управляючих систем та технологійСпеціальність 122 "Комп'ютерні науки"Освітня програма "Комп'ютерні науки"**""ЗАТВЕРДЖУЮ"**

Гарант освітньої програми

В. Ажищев

к.т.н, доц. Ажищев В. Ф.

" ____ " _____ 2024 р.

**ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття ступеня вищої освіти "магістр"**Студента Дякончука Олександра Вікторовича

(прізвище, ім'я, по батькові)

1. Тема роботи: Дослідження облікових систем оптової торгівліКерівник роботи Гайдаєнко Оксана ВолодимирівнаЗатверджені наказом ректора № 1055-уч від "8" жовтня 2024 року.2. Термін подання роботи: 15 грудня 2024 р.3. Вихідні дані до проекту (роботи) — ДСТУ щодо обробки інформації, літературні джерела, технічна документація на існуючі аналоги систем та матеріали наукового стажування.4. Перелік питань, що належать до розробки (найменування розділів): аналіз предметної області та постановка задачі, розробка проекту інформаційної системи для підприємства, що займається оптовою торгівлею медом, реалізація проекту інформаційної системи.

5. Перелік презентаційних матеріалів

6. Консультанти розділів роботи

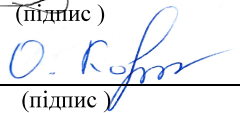
Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Гайдаєнко О.В	05.09.2023	22.12.2023
2	Гайдаєнко О.В	23.12.2023	15.06.2024
3	Гайдаєнко О.В	15.06.2024	25.09.2024
4	Гайдаєнко О.В	25.09.2024	14.11.2024
	Гайдаєнко О.В	25.09.2024	14.11.2024
	Гайдаєнко О.В	25.09.2024	14.11.2024

7. Дата видачі завдання 05 вересня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів роботи	Примітка
1	Аналіз предметної області	22.12.2024	
2	Розробка концепції автоматизованого робочого місця підприємця по оптовій закупівлі та реалізації меду	15.03.2024	
3	Розробка технічного проекту АРМ	15.06.2024	
4	Реалізація проекту АРМ	25.09.2024	
5	Охорона праці та техніка безпеки	14.11.2024	
6	Затвердження теми дипломної роботи	14.11.2024	
7	Подача роботи на анти плагіат	14.11.2024	
8	Затвердження дипломної роботи керівником	16.11.2024	
9	Доповідь та презентація	20.11.2024	
10	Захист дипломної роботи	15.12.2024	

Студент:  Дякончук О. В.
(підпис) (ПІБ)

Керівник роботи:  Гайдаєнко О. В.
(підпис) (ПІБ)

АНОТАЦІЯ

Дослідження облікових систем оптової торгівлі. Кваліфікаційна робота зі спеціальності 122 "Комп'ютерні науки", освітньої програми "Управління проектами". Миколаїв: НУК ім. адм. Макарова, 2024 рік. 91 сторінка

Дана робота спрямована на дослідження архітектури та особливостей корпоративних застосунків, спрямованих на вирішення основних задач бізнесу. Для оцінки було проведено аналіз декількох систем бізнес-обліку та досліджено сучасні технології розробки подібних систем на платформі Java. Також, у даній курсовій роботі розглядаються методи та технології, необхідні для проектування та розробки інформаційної системи підприємства торгівлі медовою продукцією.

Мета роботи полягає у дослідженні існуючих систем обліку та технологій їх розробки на платформі Java, та розробці проєкту інформаційної системи для фізичної особи підприємця (ФОП), що займається оптовою закупівлею продукції бджолярства.

Ключові слова: Java, Jakarta Enterprise Edition; Spring Boot; Spring Framework; операційна система; корпоративний застосунок.

ABSTRACT

Research of wholesale trade accounting systems. Qualification work in specialty 122 "Computer Science", educational program "Project Management". Mykolaiv: NUK named after Adm. Makarov, 2024. 91 pages

This work is aimed at studying the architecture and features of corporate applications aimed at solving the main business tasks. For the assessment, an analysis of several business accounting systems was conducted and modern technologies for developing such systems on the Java platform were investigated. Also, this coursework considers the methods and technologies necessary for designing and developing an information system for a honey products trading enterprise.

The purpose of the work is to study existing accounting systems and technologies for their development on the Java platform, and to develop a project for an information system for an individual entrepreneur (FOP) engaged in wholesale procurement of beekeeping products.

Keywords: Java, Jakarta Enterprise Edition; Spring Boot; Spring Framework; operating system; corporate application.

ЗМІСТ

ВСТУП	7
РОЗДІЛ 1. ДОСЛІДЖЕННЯ ТЕХНОЛОГІЙ РОЗРОБКИ СИСТЕМ БІЗНЕС-ОБЛІКУ ТА ПОСТАНОВКА ЗАВДАННЯ ДЛЯ РОЗРОБКИ ІНФОРМАЦІЙНОЇ ПІДСИСТЕМИ	9
1.1 Аналіз предметної області та інформаційних процесів у сфері торгівлі медовою продукцією	9
1.2 Формування вимог до інформаційної системи	10
1.3 Аналіз можливих рішень серед існуючих інформаційних систем	11
1.4 Дослідження засобів і технологій розробки інформаційних систем	16
1.5 Розробка концепції	25
1.6 Технічні засоби	26
1.7 Програмні засоби	27
1.8 Висновки до розділу 1	28
РОЗДІЛ 2. ПРОЕКТУВАННЯ ПІС ПІДПРИЄМСТВА ОПТОВОЇ ЗАКУПІВЛІ МЕДОВОЇ ПРОДУКЦІЇ	29
2.1 Загальносистемні рішення	29
2.2 Рішення щодо інформаційного забезпечення	36
2.3 Рішення щодо програмного забезпечення	43
2.3.1 Методи розробки програмного забезпечення	47
2.3.2 Вибір операційної системи	48
2.3.3 Типові програмні засоби	49
2.4 Рішення щодо технічного забезпечення	50
РОЗДІЛ 3. РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ПІДПРИЄМСТВА З ОПТОВОЇ ЗАКУПІВЛІ МЕДУ	52

	6
3.1 Інструкція щодо формування та ведення бази даних підприємства.....	54
3.2 Підготовка до роботи з інформаційною системою	55
3.3 Інструкція на випадок виникнення помилок і збоїв у роботі системи..	55
3.4 Висновки	55
РОЗДІЛ 4. ТЕХНІКО-ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ РОЗРОБКИ	56
4.1 Вступ.....	56
4.2 Розрахунок вартості створення інформаційної системи управління контентом web-порталу	57
4.3 Розрахунок економічної ефективності.....	60
4.4 Підсумки	61
Розділ 5. охорона праці.....	63
5.1 Характеристики робочого приміщення	63
5.2 Технічні рішення з гігієни праці та виробничої санітарії.....	64
5.2.1 Мікроклімат	64
5.2.2 Склад повітря робочої зони.....	65
5.2.3 Виробниче освітлення	66
5.2.4 Природне освітлення	67
5.2.5 Штучне освітлення.....	68
РОЗДІЛ 6. ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА	69
ВИСНОВКИ.....	73
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	75

ВСТУП

Тема роботи — "Дослідження облікових систем оптової торгівлі та розробка системи підприємства торгівлі медовою продукцією". Актуальність такої системи обґрунтована існуванням численних підприємств, які не здатні продовжувати своє розвиток через відмову змінювати підхід до роботи та системи автоматизації і обліку, які могли б значно прискорити та спростити весь робочий процес і вирішити багато раніше нездійснених завдань.

Цей дипломний проєкт спрямований на дослідження існуючих рішень серед інформаційних систем бізнес-обліку, їх завдань і особливостей, проведення аналізу їх структури та функціоналу. Також завданням цієї роботи є розгляд і аналіз сучасних рішень для розробки подібних систем на стеку Java і на основі отриманих результатів спроектувати систему підприємства торгівлі продукцією бджільництва.

Перший розділ спрямований на дослідження кількох існуючих систем бізнес-обліку та ERP-систем, короткий опис аналізу їх особливостей, функціоналу, головних переваг та недоліків. Зокрема, буде проведено аналіз і порівняння доступних технологій для розробки корпоративних додатків, наданих Java-платформою.

Будуть описані основні поняття і технології, які використовуються при розробці Java-додатків на стеку Jakarta EE і Spring Framework. Будуть розглянуті ключові характеристики компонентів цих платформ, необхідні для розробки і побудови інформаційних систем.

У другому розділі буде подано детальний опис методів і засобів, що використовуються для розробки інформаційної системи. Також будуть представлені моделі проекрованої бази даних та апаратні засоби, необхідні для використання проекрованої системи.

У третьому розділі будуть описані результати виконаної роботи, надано інформацію про інтерфейс і функціонал веб-додатку, а також надано всі

необхідні інструкції та умови, які потрібно врахувати перед використанням готового АРМ.

У четвертому розділі буде проведена оцінка економічних аспектів дослідження і розробки програмного забезпечення. Буде визначено вартість розробки інформаційної системи та обладнання, необхідного для її роботи.

П'ятий розділ надає інформацію про шкідливі фактори, яким може піддаватися користувач системи, а також специфіку та шкідливі фактори підприємства, що займається закупівлею та збутом меду та детально описує екологічний аспект проведення дослідження та ведення розробки програмного забезпечення, а також вплив ІТ-сфери на навколишнє середовище.

РОЗДІЛ 1. ДОСЛІДЖЕННЯ ТЕХНОЛОГІЙ РОЗРОБКИ СИСТЕМ БІЗНЕС-ОБЛІКУ ТА ПОСТАНОВКА ЗАВДАННЯ ДЛЯ РОЗРОБКИ ІНФОРМАЦІЙНОЇ ПІДСИСТЕМИ

1.1 Аналіз предметної області та інформаційних процесів у сфері торгівлі медовою продукцією

Фізична особа-підприємець (ФОП) — це фізична особа, яка реалізує здатність до праці шляхом самостійної діяльності, на власний ризик, з метою отримання прибутку.

Оптова торгівля — вид торгової діяльності, пов'язаний із придбанням та збутом товару для його використання у підприємницькій діяльності або з іншою метою, не пов'язаною з особистим, сімейним, домашнім чи іншим подібним використанням.

Діяльність підприємця безпосередньо залежить від його можливостей та часу, який він приділяє веденню бізнесу. У цій сфері важливою є необхідність збереження даних про грошові перекази, товарообіг або, наприклад, клієнтську базу.

На основі збережених даних можна вести статистику, розраховувати можливі ризики та аналізувати потенційний прибуток за сезон, що у сфері оптової торгівлі може стати як вагомою перевагою, так і добрим підґрунтям для оптимізації робочого часу.

Подібні підприємства, зазвичай, не мають складної структури підпорядкованості. Більшість роботи виконується самим підприємцем, який є ключовою фігурою та займає як керівну, так і виконавчу посаду.

ФОП самостійно керує діяльністю свого підприємства: він шукає потенційних клієнтів, домовляється про зустрічі з ними, займається закупівлею товарів, шукає ринки збуту, виконуючи посередницьку роботу між постачальниками й виробниками. Часто він також самостійно веде бухгалтерський облік.

Аналіз роботи показав, що весь облік ведеться на папері. Пошук даних і формування вибірок для звітів здійснюються вручну, що підвищує як ризик допущення помилок, так і час, необхідний для обробки даних. Це зменшує час на пошук нових постачальників або точок збуту, що значно знижує ефективність роботи та потенційний дохід.

Можна виділити декілька проблем, які, відповідно до проведеного аналізу, негативно впливають на роботу підприємства:

1. Помилки при заповненні даних клієнтської бази.
2. Можливі помилки під час обробки грошових переказів і заповнення даних про рахунки та кошти.
3. Зберігання некоректної або застарілої інформації.
4. Вплив людського фактору на роботу з великим обсягом збереженої інформації.
5. Тривалий пошук або втрата інформації.
6. Формування некоректних вибірок для створення звітів.

Вирішенням таких проблем може стати використання засобів автоматизації обробки даних.

1.2 Формування вимог до інформаційної системи

Відповідно до проблемних моментів, виділених у попередньому пункті, були сформовані такі вимоги до функціоналу:

1. Створення форм для коректного заповнення даних про клієнтів і угоди.
2. Структурування збереженої інформації.
3. Автоматичний розрахунок виплат за придбану продукцію.
4. Автоматизація обліку збереженої продукції.

Для виконання поставлених функціональних вимог необхідно:

1. Операційна система (ОС).

2. Веб-браузер для можливості використання інформаційної системи та реалізації зручного інтерфейсу.
3. Мови програмування та розмітки для розробки веб-застосунку.
4. Реляційна система управління базами даних (СУБД).

Для розробки проєкту потрібні такі програмні засоби:

1. Інтегроване середовище розробки.
2. Графічний редактор для створення діаграм.
3. Система контролю версій.

1.3 Аналіз можливих рішень серед існуючих інформаційних систем

ERP (Enterprise Resource Planning) — організаційна стратегія інтеграції виробництва та операцій, управління трудовими ресурсами, фінансового менеджменту та управління активами, орієнтована на безперервну балансування та оптимізацію ресурсів підприємства за допомогою спеціалізованого інтегрованого пакета прикладного програмного забезпечення, що забезпечує загальну модель даних і процесів для всіх сфер діяльності [1].

Впровадження ERP-систем вважається фактично необхідною умовою для публічної компанії, починаючи з кінця 1990-х років. ERP-системи, які спочатку впроваджувалися лише на промислових підприємствах, використовуються більшістю великих організацій незалежно від країни, форми власності та галузі.

Завдання ERP — зробити так, щоб усі бізнес-процеси були передбачуваними, контролювати діяльність бізнесу глобально, не концентруючись на окремих аспектах.

Багато хто починає з блокнотів і MS Excel, однак згодом виникають питання, які змушують замислитися: скільки блокнотів і таблиць потрібно для зберігання та обробки даних? Скільки часу займе робота з ними? Як

організувати безпеку даних і обмежити доступ певних співробітників до тієї чи іншої інформації?

Багато компаній стикаються з проблемами масштабування саме тому, що не бажають змінювати свій підхід до роботи та обліку, відмовитися від використання застарілих технологій. Проте бізнес повинен розвиватися, зростати, використовуючи всілякі нововведення, які сприятимуть спрощенню робочого процесу.

Серед сучасних програм обліку можна відзначити Vimp — ERP-систему для малого та середнього бізнесу, завдання якої полягає в автоматизації роботи з контрагентами, товарами, складом, фінансами та звітами. Її успішно використовують підприємства різних галузей, зокрема в логістиці, будівництві, виробництві, сервісних центрах, роздрібній і B2B-торгівлі, а також у готельно-ресторанному бізнесі.

Автором цієї системи є однойменна українська компанія, заснована у 2020 році експертами у сфері фінансів, розробки та впровадження систем бізнес-обліку [2].

Особливість Vimp полягає в тому, що він призначений для вирішення першочергових базових завдань бізнесу, зокрема обліку та планування запасів, фінансового та виробничого обліку.

За допомогою цієї системи значно спрощується та покращується робота з контрагентами (швидкий пошук, прив'язка документації до картки контрагента), процес закупівель (відстеження оплат, формування собівартості, ведення платіжного календаря, нагадування про оплату, контроль статусу заявок), проведення продажів (відстеження статусів продажу й відправлення товару, аналітика), облік складу (контроль руху товарів, інвентаризація, налаштування цін, автоматичний розрахунок собівартості, списання залишків і попередження про них), формування звітності, фінансовий облік (відстеження собівартості, прибутку, руху коштів, взаєморозрахунків, точок росту та повна фінансова звітність), нарахування заробітної плати, ведення роздрібної торгівлі (здійснення та контроль продажів, автоматизація

сканування, продажу й списання товарів, швидке створення місця касира й нової торгової точки тощо) і виробництва (автоматичний розрахунок собівартості для складних продуктів, попередження про низькі залишки сировини, відстеження етапів виробництва).

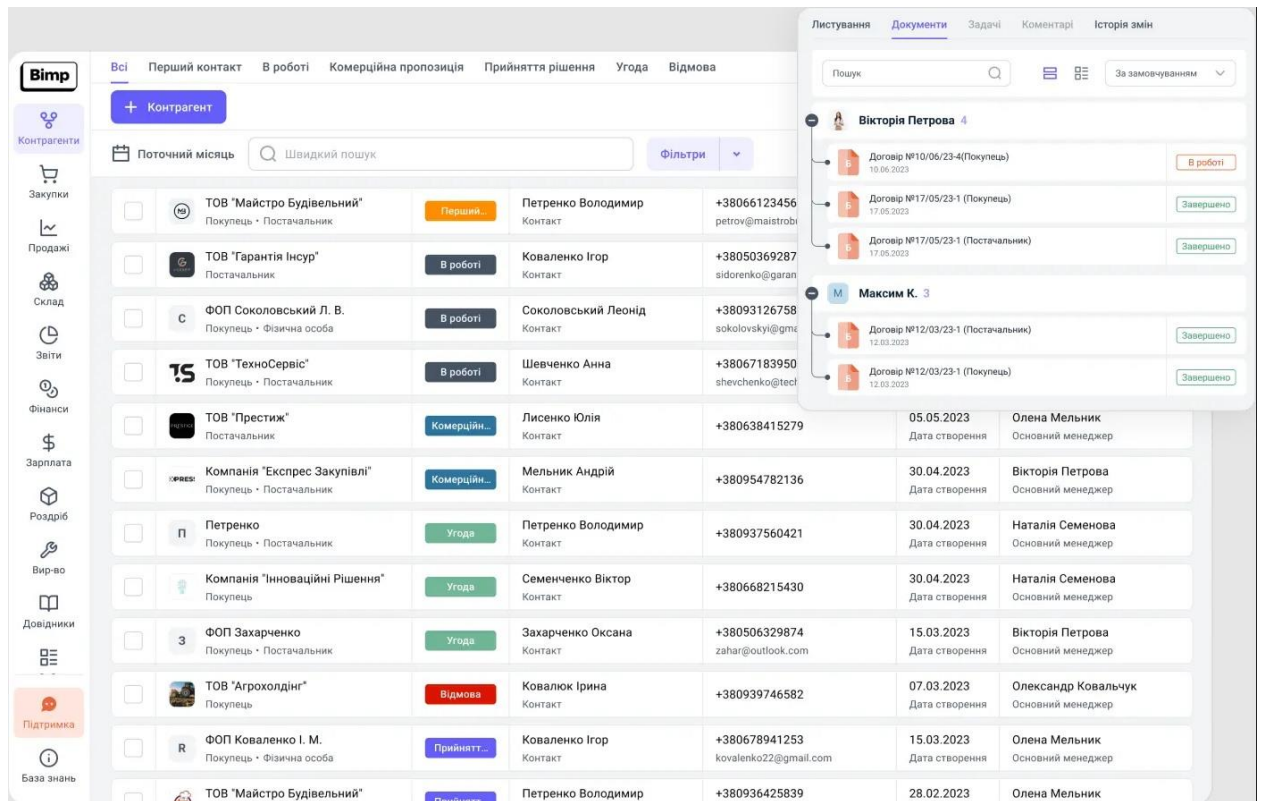


Рисунок 1.1 – Інтерфейс програми Bimp або "Business impulse"

Система пропонує зручний та інтерактивний інтерфейс і функціонал (рисунок 1.1), позиціонуючи себе як заміну застарілим 1С та "Мій склад". Проте для глибшого розуміння варто розглянути саму платформу 1С, яка багато років є лідером у сфері автоматизації бізнесу.

Для клієнта 1С – це система автоматизації, яка може швидко вирішити його бізнес-завдання, незалежно від того, чи це невеликий кіоск, чи великий холдинг. Попри видимі відмінності в потребах цих видів бізнесу, обидва підтримуються єдиною кодовою базою платформи.

1С пропонує рішення, які відповідають на бізнес-завдання шляхом автоматизації. Будь то великий чи малий бізнес, у 1С є рішення для виникаючих проблем. У цьому сценарії ключовими є бізнес-додатки, особливо системи бухгалтерського обліку тощо. Для створення подібних додатків 1С

використовує власну платформу розробки бізнес-додатків, яка адаптована під їхні загальні вимоги:

1. Облік фінансів, товарів і грошового обігу загалом.
2. Проста настройка бізнес-логіки.
3. Широкі можливості інтеграції в гетерогенні IT-ландшафти.

Сама платформа використовує трирівневу архітектуру, в центрі якої знаходиться сервер додатків. Як система управління базами даних використовуються MS SQL або PostgreSQL. Також 1С підтримує Oracle і IBM DB2, які зустрічаються рідше через меншу поширеність на масовому ринку. Користувацький інтерфейс може бути реалізований як веб-додаток або тонкий клієнт, встановлений на комп'ютері користувача [3].

Технологічна платформа "1С:Підприємство" є програмною оболонкою над базою даних. Використовуються бази на основі DBF-файлів у версії 7.7, власний формат 1CD з версії 8.0 або СУБД Microsoft SQL Server на будь-якій із цих версій. Крім того, з версії 8.1 можливе зберігання даних у PostgreSQL та IBM DB2, а з версії 8.2 додано підтримку Oracle. Платформа має власну внутрішню мову програмування, яка, окрім доступу до даних, забезпечує можливість взаємодії з іншими програмами за допомогою OLE та DDE, а у версіях 7.7, 8.0 і 8.1 – через COM-з'єднання.

Проте кожне технологічне рішення має свої межі застосування. Внутрішня мова 1С не є мовою загального призначення, тому вона не може існувати окремо від своєї платформи.

Клієнтська частина платформи функціонує в середовищі Microsoft Windows, а починаючи з версії 8.3, також у середовищах Linux і macOS. Починаючи з версії 8.1, серверна частина платформи в клієнт-серверному варіанті роботи «1С:Підприємство» може працювати на ОС Microsoft Windows і Linux.

В основі лінійки програмних продуктів 1С лежать типові конфігурації. На сайті 1С представлено багато готових коробкових рішень, але основна маса користувачів використовує лише 4 конфігурації:

1. Бухгалтерія підприємства
2. Управління торгівлею
3. Управління виробничим підприємством
4. Заробітна плата й управління персоналом

За більш ніж 20 років свого розвитку 1С досягла значних успіхів у розширенні та вдосконаленні функціоналу своїх застосунків, додавши безліч опцій для модифікації й налаштування платформи, що тільки підвищило її популярність.

Проте 1С не позбавлена й недоліків. Кожна з конфігурацій має ті ж недоліки, що й сама платформа:

1. Відсутність модульності
2. Громіздкість і наявність зайвих функцій
3. Нові баги в нових версіях
4. Непередбачуваний результат оновлень

Відсутність модульності призводить до того, що помилка в роботі однієї функції може впливати на роботу інших.

Практика показує, що у сфері торгівлі люди використовують не більше 30% усього функціоналу, який надає платформа. Подібна ситуація спостерігається й в інших типових конфігураціях. У гонитві за максимальним охопленням можливостей розробники створили дуже громіздкі й складні продукти, у яких усе взаємопов'язано, а тому навіть вимкнення непотрібного функціоналу не завжди можливе.

Таким чином, нагромадження функціоналу створює проблеми не лише для користувачів платформи, а й породжує чимало труднощів у налаштуванні самої платформи.

Однією з сучасних альтернатив, що прийшла на заміну 1С, стала корпоративна розробка BAS. Принаймні як користувачі, так і розробники застосунку представляють її як сучасну зміну застарілій платформі 1С.

1.4 Дослідження засобів і технологій розробки інформаційних систем

Розробка корпоративних систем або систем бізнес-обліку, незважаючи на схожий функціонал, має свою специфіку й вимоги. Проте суть у тому, що для різних проєктів може використовуватися одна архітектура й набір перевірених технологій, наприклад, на базі Spring Framework із можливістю підключення будь-яких технологій або власних реалізацій.

Java дає змогу використовувати величезну кількість бібліотек і фреймворків, що робить її однією з найпопулярніших мов програмування для створення подібних інформаційних систем. Це підкріплюється надзвичайно великим співтовариством, яке постійно розвиває як саму мову, так і технології, що використовуються для прискорення й зручності створення проєктів на Java-стеку, породжуючи різні підходи до їх розробки.

Платформа Java надає справді багато засобів і технологій для розробки.

Java Development Kit (JDK) — це комплект для розробки застосунків мовою Java, який містить різноманітні утиліти та виконуване середовище Java. JDK включає бібліотеки та інструменти, необхідні для створення, компіляції, тестування та виконання застосунків. Однак до пакету JDK не входить інтегроване середовище розробки (IDE), тому розробнику доводиться використовувати зовнішній текстовий редактор і компілювати програми за допомогою утиліт командного рядка [4].

IDE є прямою протилежністю цьому підходу, в якому використовуються несумісні інструменти, такі як текстовий редактор, компілятор тощо. Інтегровані середовища розробки створені для того, щоб максимізувати продуктивність розробника завдяки тісно пов'язаним компонентам із простими користувацькими інтерфейсами. Це зменшує кількість необхідних дій для перемикання між різними режимами роботи. IDE містить багато функцій для створення, зміни, компіляції, розгортання та налагодження програмного забезпечення.

Основним завданням інтегрованого середовища розробки є об'єднання різних утиліт в одному модулі, що дає змогу абстрагуватися від виконання допоміжних завдань, надаючи розробнику можливість зосередитися на розв'язанні основної задачі та уникнути втрати часу під час виконання рутинних технічних дій.

Усі сучасні середовища розробки застосунків Java, такі як JDeveloper, NetBeans, IntelliJ IDEA, Eclipse, Sun Java Studio Creator, базуються на сервісах, які надає JDK. Більшість із них використовують компілятор Javac із комплекту JDK для компіляції Java-програм. Тому середовища розробки або включають одну з версій JDK у комплект постачання, або вимагають попередньої інсталяції JDK на пристрої розробника.

Apache Maven — це інструмент для автоматизації збірки проєктів і управління залежностями на основі опису їхньої структури у файлах Project Object Model (POM) мовою XML. Він базується на плагін-архітектурі, яка дозволяє використовувати плагіни для різних завдань без необхідності їх явної інсталяції.

Jakarta EE (раніше Java Platform, Enterprise Edition або Java EE) — це набір специфікацій і документації для мови Java, що описує архітектуру серверної платформи. Основна мета — забезпечення масштабованості застосунків і цілісності даних під час роботи системи. Java EE API включає технології, які розширюють функціональність Java SE, наприклад, сервлети, JPA (Java Persistence API), EJB (Enterprise JavaBean), JMS (Java Message Service) тощо.

Spring Framework або коротко Spring — це універсальний фреймворк із відкритим вихідним кодом для платформи Java, який забезпечує розв'язання багатьох завдань, із якими стикаються Java-розробники та організації під час створення інформаційних систем на платформі Java. Spring можна розглядати як колекцію менших фреймворків або фреймворків у фреймворку. Більшість із цих фреймворків може працювати незалежно один від одного, але вони забезпечують більшу функціональність у разі спільного використання. Spring

здобув популярність завдяки тому, що став джерелом розв'язань, потрібних для ефективної розробки складних бізнес-застосунків.

JUnit — це фреймворк для автоматизації тестування коду мовою Java. JUnit часто використовується для модульного тестування й належить до сімейства фреймворків xUnit.

Наведені інструменти є лише невеликою частиною засобів і технологій для розробки як масштабних корпоративних інформаційних систем, так і рішень для малого бізнесу. Багатошаровий стек технологій для розробки на Java дозволяє проєктувати складні масштабовані рішення для різних підприємств.

Перш ніж приступити до розробки застосунку, необхідно визначити технології, які будуть використовуватися для реалізації проєкту. Принципово вибір розробника розділений між двома технологіями: Jakarta EE і Spring Framework. Інші технології розробки, такі як Maven, Hibernate чи JUnit, можуть бути інтегровані з будь-якою з них.

Java EE — це платформа, побудована на основі Java SE, яка надає API та середовище виконання для розробки й запуску великомасштабних, багаторівневих, масштабованих, надійних і безпечних корпоративних застосунків (Enterprise applications).

Модель програмування, прийнята в Jakarta EE, дозволяє розробнику зосередитися саме на тому, на чому потрібно — тобто на бізнес-логіці. Більше не потрібно наслідувати класи API; розробник може реалізовувати логіку своєї предметної області звичайною мовою Java і переважно декларативно (за допомогою анотацій) управляти поведінкою сервера застосунків.

Jakarta містить набір технологій і стандартів (Enterprise JavaBeans, JSP, Servlets, Java Persistence API, JMS та інші), які розширюють можливості та забезпечують комплексний підхід до процесу розробки.

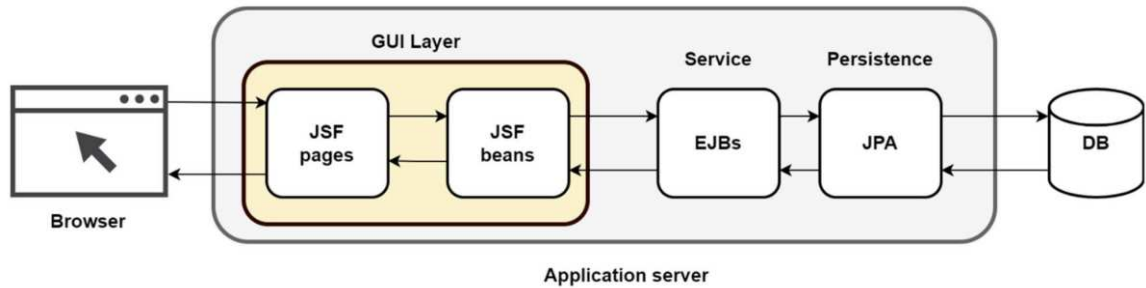


Рисунок 1.2 — Структура застосунку на платформі Java EE

Запит обробляється сторінкою JSF, а потім — біном. Далі управління передається бізнес-логіці в EJB, яка працює з базою даних через JPA (Рисунок 1.1). Важливо зазначити, що зі збільшенням навантаження на застосунок схема роботи системи залишатиметься незмінною. Jakarta дозволяє створювати величезний кластер серверів, а також додавати вузли у разі недостатньої потужності, при цьому зберігаючи роботу в кластерному середовищі без жодних змін у коді.

Завдяки своїм особливостям і структурі стек Jakarta EE рекомендується для розробки застосунків, головною проблемою яких є масштабування проєкту та збільшення навантаження.

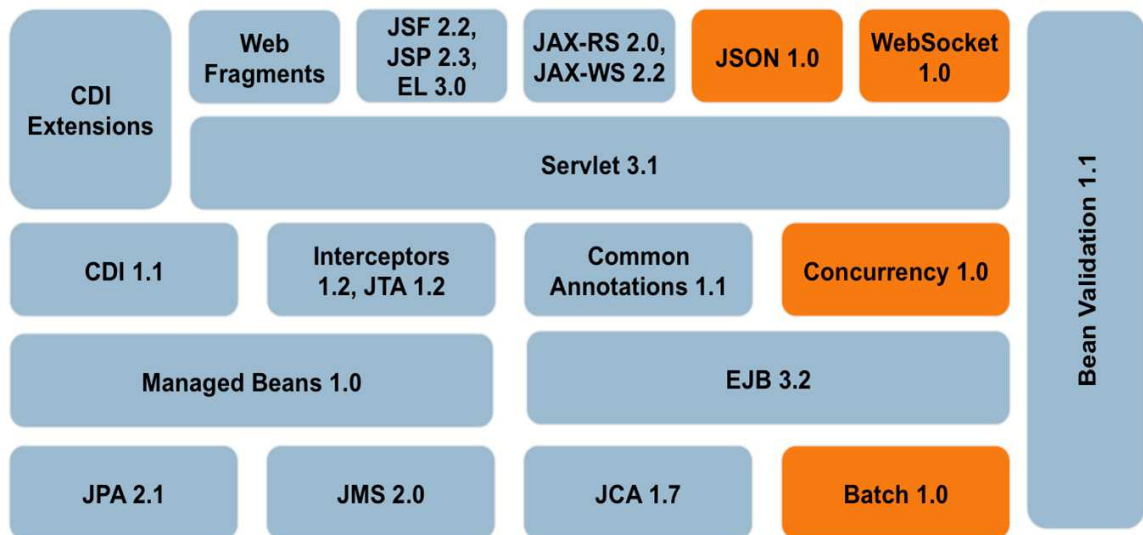


Рисунок 1.3 — Компоненти платформи Jakarta EE з точки зору її документації

Згідно з документацією, Jakarta EE може використовуватися без особливих труднощів. Крім того, увесь необхідний функціонал уже

вбудований у будь-який сервер застосунків (Enterprise Application Server) "прямо з коробки".

Довгі роки Hibernate, а згодом і JPA, були основними технологіями для роботи з реляційними базами даних. Однак поява та стрімкий розвиток NoSQL-рішень наштовхнули інженерів Spring на думку про створення нового, інноваційного підходу для доступу до будь-яких джерел даних. Результатом їхньої праці став репозиторій, який інкапсулював роботу з базою даних і дозволяв працювати з нею, як із звичайною колекцією, незалежно від її фізичної структури чи API.

Spring Framework став гнучким інструментом, який представляє собою контейнер для впровадження залежностей із декількома зручними шарами (наприклад: доступ до бази даних, проксі, аспектно-орієнтоване програмування, RPC, веб-інфраструктура MVC). Це дозволяє швидше й зручніше створювати Java-застосунки.

Ключовим елементом Spring є інфраструктурна підтримка на рівні застосунків: Spring зосереджується на «підключенні» корпоративних застосунків, щоб команди могли фокусуватися на бізнес-логіці, не прив'язуючись до конкретних середовищ розгортання.

Простий застосунок на стеку Spring не сильно відрізняється від архітектури застосунків на стеку Jakarta Enterprise.

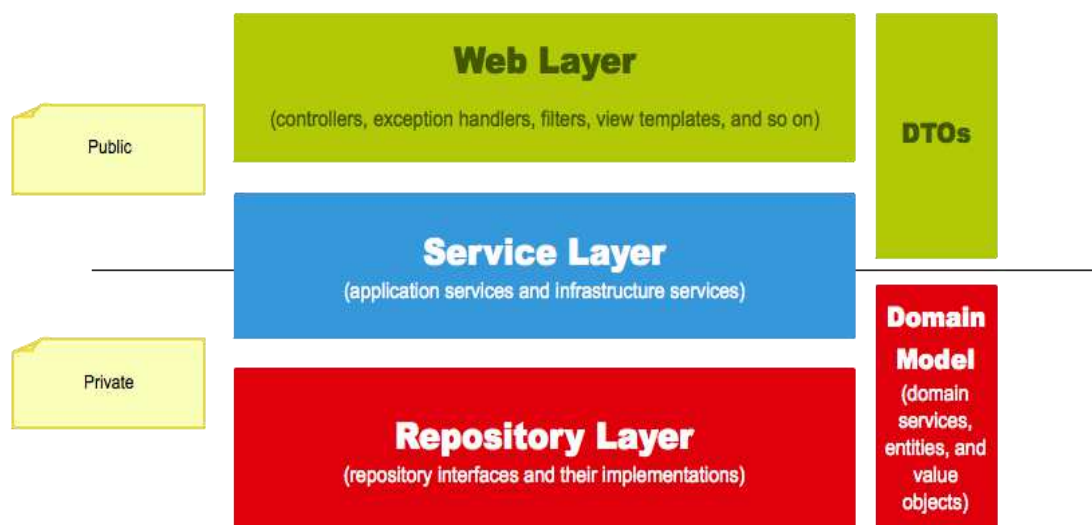


Рисунок 1.4 — Структура застосунку на стеку Spring

Якщо розглядати прості застосунки на стеках Java Enterprise і Spring, архітектура Spring відрізняється лише трохи більшою деталізацією. По суті, єдина вагома відмінність полягає в тому, що Jakarta EE працює лише в рамках корпоративного сервера застосунків (Enterprise Application Server), тоді як застосунок на стеку Spring може працювати на будь-чому: Tomcat, GlassFish або навіть без сервера (оскільки він може запустити сервер усередині себе).

Це робить платформу Spring ідеальним рішенням для реалізації елементів мікросервісної архітектури, але за рахунок відмови від підтримки всіх можливостей корпоративних серверів застосунків. Загалом, кластерний застосунок — це не про платформу Spring, і питання масштабування для Spring-застосунків вирішуються окремо на етапі проєктування.

Стек бібліотек Spring настільки великий, що на офіційному сайті з документацією представлено зображення, яке вже кілька років застаріле, а винести всі його компоненти в одне зображення є досить складним завданням.

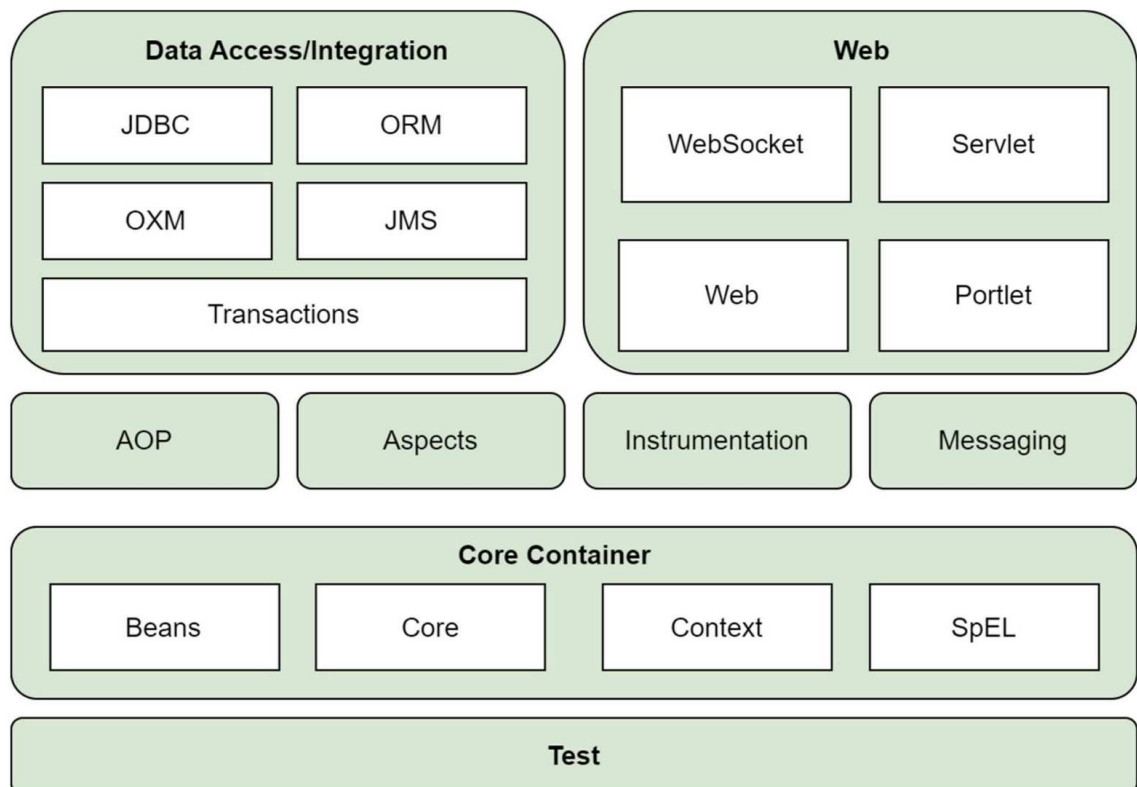


Рисунок 1.5 — Складові та архітектура платформи Spring Framework

Spring Beans and EJB (Enterprise Java Beans)

Jakarta Enterprise Beans (EJB, Enterprise JavaBeans) — це серверний програмний компонент, що інкапсулює бізнес-логіку застосунку. Специфікація EJB не описує, як сервер застосунків забезпечує збереження даних (це завдання делегується специфікації JPA), але натомість описує, як бізнес-логіка може легко інтегруватися з сервісами збереження, які пропонує сервер застосунків. Специфікація детально описує, як сервер застосунків виконує такі завдання:

1. Обробка транзакцій
2. Інтеграція з сервісами збереження, які пропонує Jakarta Persistence (JPA)
3. Управління паралелізмом
4. Програмування з використанням Jakarta Messaging (JMS) та Jakarta Connectors (JCA)
5. Асинхронний виклик методів
6. Планування роботи
7. Сервіси іменування та каталогів через інтерфейс іменування та каталогів Java (JNDI)
8. Міжпроцесна взаємодія за допомогою RMI-IIOP та веб-сервісів
9. Безпека (JCE і JAAS)
10. Розгортання програмних компонентів на сервері застосунків

Spring Beans — це прості об'єкти, якими управляє контейнер Spring, які легко інтегруються в будь-які компоненти застосунку та взаємодіють через введення залежностей.

Підсумовуючи, Spring Beans пропонують простий і гнучкий спосіб управління об'єктами, однак EJB мають ширший функціонал. Enterprise JavaBeans мають вбудовану підтримку розподіленого виконання, включаючи розподілений збирач сміття, автентифікацію, підтримку транзакцій тощо.

Тому вибір технології для розробки залежить від специфіки та вимог до розподіленості об'єктів і масштабування проєкту.

Spring Data and JPA

Jakarta Persistence, також відомий як JPA, є специфікацією інтерфейсу програмування застосунків Jakarta EE, яка використовується для роботи з реляційними базами даних завдяки методам об'єктно-реляційного відображення (Object-Relational Mapping, ORM). Багато ключових аспектів JPA запозичені з ORM-інструментів, таких як Hibernate і TopLink.

Spring Data — це репозиторій, який використовує JPA Entity для взаємодії з базами даних, організуючи їх у репозиторії без прив'язки до конкретної реалізації SQL або JPA. Він спрямований на спрощення роботи з JPA, зменшуючи кількість шаблонного коду в процесі розробки.

На сьогоднішній день платформа Jakarta Enterprise не має завершених аналогів Spring Data, однак із 2022 року ведеться розробка специфікації Jakarta Data у складі оновленої Jakarta EE 11, випуск якої заплановано на 2024 рік, але проєкт усе ще перебуває в розробці.

Enterprise Server and Spring Boot

Spring Boot — це фреймворк із відкритим вихідним кодом, спрямований на прискорення процесу розробки. Він надає різноманітні опції для створення й розгортання веб-застосунків і мікросервісів без необхідності налаштувань конфігурації або написання додаткового коду.

Сервер Jakarta EE — це серверне застосунок, яке реалізує API платформи Jakarta EE та надає стандартні сервіси Jakarta EE. Сервери Jakarta EE іноді називають серверами застосунків, оскільки вони обслуговують дані застосунків для клієнтів, так само як веб-сервери обслуговують веб-сторінки для веб-браузерів.

Гнучкість Spring Boot спрощує розгортання застосунків завдяки уніфікованим інструментам автоматизації, що значно прискорює процес розробки. Однак, на відміну від Spring Boot, Enterprise Server вимагає більше ресурсів і зусиль, але надає повний набір фреймворків і технологій для розробки корпоративних систем.

1.5 Розробка концепції

Відповідно до сформованих вимог була розроблена концепція системи, а згодом, на основі функціональних вимог, — модель розроблюваної системи.

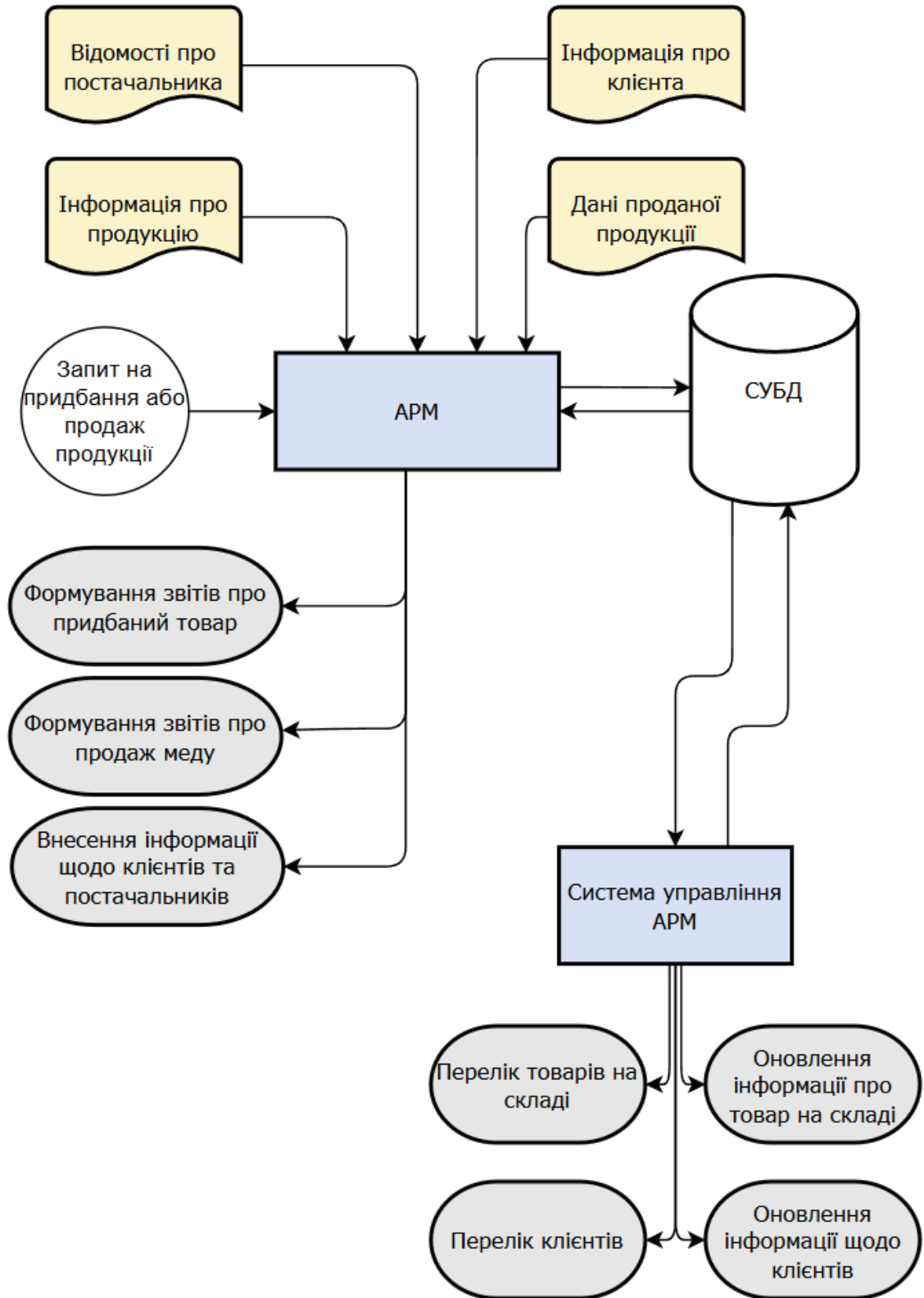


Рисунок 1.6 — Модель інформаційної системи

Модель, зображена на рисунку 1.6, відображає ключові елементи розроблюваної системи:

1. АРМ (автоматизоване робоче місце).
2. База даних (БД) та її система управління.
3. Технічні засоби.
4. Програмні засоби.

Відповідно до створеної моделі, АРМ матиме функціонал, що дозволить вносити, редагувати або видаляти інформацію, яка зберігатиметься в локальній базі даних. Це забезпечить високу швидкість роботи та мінімізує вплив людського фактору під час формування звітів.

Створення подібних систем має відповідати цілям і завданням, дотримуючись основних принципів розробки АРМ, а саме: системності, стійкості, гнучкості та ефективності.

Саме АРМ спростить процес обліку, зберігання та пошуку інформації завдяки автоматизації таких завдань:

1. Зберігання даних клієнтської бази, що значно прискорить пошук і закупівлю необхідної продукції.
2. Формування переліку товарів, їх наявності на складі, а також відповідних даних про ці товари.
3. Формування історії операцій, що дозволить відстежувати рух товарів і грошових коштів.

Збір і аналіз даних дозволять підприємцю відстежувати виникнення потенційних проблем і вживати необхідних заходів для їх усунення. Це також позитивно вплине на швидкість роботи, забезпечуючи високу якість продукції для задоволення потреб клієнтів.

1.6 Технічні засоби

Для забезпечення нормальної роботи АРМ і бази даних користувачеві необхідно мати пристрій із такими характеристиками:

1. Процесор — Intel Core i5 2-го покоління або продуктивніший з інтегрованою графікою.
2. Не менше 8 ГБ оперативної пам'яті.

А також:

1. Комп'ютерну мишу або тачпад.
2. Клавіатуру.
3. Монітор або екран із роздільною здатністю не менше 1440×900 рх.

1.7 Програмні засоби

Однією з переваг проєктованої системи є кросплатформеність. Це забезпечується тим, що АРМ буде спроектоване у вигляді веб-застосунку. Це, у свою чергу, дозволяє використовувати будь-яку операційну систему для ПК із будь-яким веб-браузером, який підтримується цією системою. Однак для коректного відображення інтерфейсу й уникнення багів рекомендується використовувати актуальну версію Google Chrome.

Процес розробки веб-застосунку відбуватиметься з використанням об'єктно-орієнтованої мови програмування Java за допомогою інтегрованого середовища розробки (IDE) IntelliJ IDEA Ultimate Edition.

Відповідно до проведеного дослідження, для розробки було обрано типові рішення для невеликих систем — Spring Framework. Для налаштування конфігурації буде використовуватися один із проєктів Spring — Spring Boot. Як систему контролю версій буде використано Git і веб-сервіс GitHub завдяки вільній інтеграції з середовищем розробки IntelliJ IDEA.

Для проєктування моделей і створення діаграм використовувався графічний редактор Diagrams.net (раніше draw.io).

1.8 Висновки до розділу 1

Підсумовуючи, стосовно двох стеків розробки корпоративного програмного забезпечення, кожна з платформ має свої сильні та слабкі сторони.

Jakarta EE ідеально підходить для розробки великих, добре масштабованих проєктів, хоча її стандарти можуть призвести до меншої гнучкості у виборі й налаштуванні програмного середовища.

Spring Boot, своєю чергою, пропонує більше свободи в налаштуваннях, полегшує роботу з мікросервісною архітектурою, але може вимагати додаткових зусиль для масштабування та підтримки кластерів.

У свою чергу, для реалізації розроблюваного проєкту, з урахуванням його обсягу, обмеженості в ресурсах, необхідності роботи з різними даними та простоти реалізації, було прийнято рішення використовувати для розробки інформаційної системи фреймворк Spring Boot.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ІПС ПІДПРИЄМСТВА ОПТОВОЇ ЗАКУПІВЛІ МЕДОВОЇ ПРОДУКЦІЇ

2.1 Загальносистемні рішення

АРМ призначене для оптимізації робочого процесу та спрощення обліку. Веб-інтерфейс надає користувачеві можливість просто й зручно використовувати весь необхідний функціонал.

Відповідно до вимог до АРМ і сформованої концепції була розроблена модель інформаційної системи, призначеної для невеликого підприємства, яке займається оптовими закупівлями меду.

Згідно з концепцією, єдиним користувачем системи буде саме фізична особа-підприємець, адже він є єдиною людиною, залученою до робочого процесу. Нижче, на діаграмі прецедентів (Use-case diagram) на рисунку 2.1, представлена схема інформаційних процесів і їхньої взаємодії.

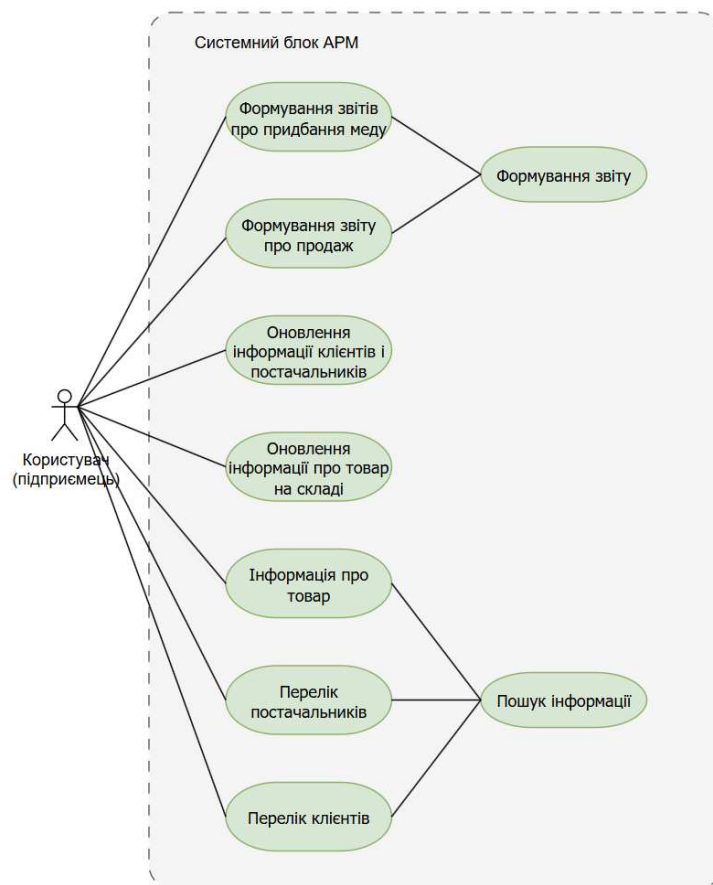


Рисунок 2.1 — Діаграма прецедентів

АРМ буде використовуватися для вирішення таких завдань, автоматизація яких має оптимізувати час робочого процесу для вирішення інших бізнес-завдань, не пов'язаних безпосередньо з АРМ, а саме:

1. Збереження інформації про закупівлі.
2. Збереження інформації про продажі.
3. Ведення обліку товарів на складі.
4. Ведення обліку даних про закупівлю товарів.
5. Ведення обліку продажів.
6. Ведення обліку клієнтської бази.
7. Ведення обліку постачальників продукції.

Далі, у таблицях 2.1–2.5 представлені різні сценарії використання інформаційної системи.

Таблиця 2.1

Таблиця 2.1. Формування звіту про закупівлю

Створення звіту про покупку	
Призначення	Використовується для створення звіту про придбання товару
Користувач	ФОП
Передумова	Необхідність створення звіту про придбання та збереження інформації про клієнта, продукт, збереження інформації про фінанси
Тригер	Натискання кнопки "створити угоду"
Сценарій	Користувач натискає кнопку "створити угоду", у вікні, що з'явиться, вибирає пункт "закупівля меду", заповнює всі необхідні поля та натискає кнопку "створити угоду"
Результат	Буде сформовано звіт про придбання продукції. Вся вказана інформація буде збережена в базі даних.

Таблиця 2.2. Формування звіту про продаж продукції

Створення звіту про покупку	
Призначення	Використовується для створення звіту про придбання товару
Користувач	ФОП
Передумова	Продаж товару зі складу з метою отримання вигоди
Тригер	Натискання кнопки "створити угоду"
Сценарій	Користувач натискає кнопку «додати угоду», у вікні, що з'явилося, вибирає пункт «продаж меду», вводить ПІБ клієнта, його номер телефону, та, за можливості, електронну пошту, вносить дані про сорт меду, який було продано, та об'єм товару. Після внесення усіх необхідних даних, користувач натискає кнопку «створити угоду».
Результат	Буде сформовано звіт про продаж продукції. Вся вказана інформація буде збережена в базі даних.

Таблиця 2.3. Заповнення даних клієнтської бази

Заповнення даних клієнтів	
Призначення	Використовується для формування бази даних клієнтів
Користувач	ФОП
Передумова	Продаж товару зі складу з метою отримання вигоди
Тригер	Натискання кнопки "створити угоду"

Продовження таблиці 2.3

Сценарій	Дані про клієнтів заповнюються в тому випадку, коли формується звіт купівлі-продажу, заповнюючи вказані поля у формі створення звіту.
Результат	Будуть створені інформаційні рядки про клієнта, його товар і оплату. Всі введені дані будуть збережені в базі даних.

Таблиця 2.4. Заповнення даних про сорти меду

Заповнення даних про сорти меду	
Призначення	Використовується для формування бази даних клієнтів
Користувач	ФОП
Передумова	Заповнення даних про сорти меду
Тригер	Користувач натискає кнопку "додати сорт"
Сценарій	Натискання кнопки "додати сорт", заповнює вказані поля та зберігає зміни
Результат	В базу даних буде збережена інформація про сорт, з виведенням відповідної інформації в переліку сортів меду для створення звітів.

Таблиця 2.5. Заповнення даних про сорти меду при формуванні звіту
про придбання

Заповнення даних про сорти меду	
Призначення	Використовується для формування бази даних клієнтів
Користувач	ФОП
Передумова	Необхідність придбання сорту меду, інформація про який ще не міститься в базі даних
Тригер	Користувач натискає кнопку "додати сорт" при формуванні звіту про придбання
Сценарій	Користувач натискає кнопку "створити угоду", у формі для заповнення вибирає список сортів меду, там же натискає кнопку "додати сорт", вводить всю необхідну інформацію, зберігає її і продовжує формувати звіт про придбання продукції
Результат	В базу даних буде збережена інформація про сорт, з виведенням відповідної інформації в переліку сортів меду для створення звітів.

Діаграма прецедентів показує, що проєктована система спрямована на максимальну простоту й зручність, надаючи можливість додавати нову інформацію про сорти продукції, якщо таких даних раніше не було в базі, без необхідності скасовувати заповнення форми звіту. Це полегшує роботу підприємця та економить його час.

Для підвищення ефективності роботи інформаційна система має відповідати таким вимогам:

1. Мати простий та інтуїтивно зрозумілий інтерфейс користувача.

2. Бути універсальною у своїй функціональності відповідно до поставлених завдань.
3. Забезпечувати надійність інформаційної системи.
4. Бути легкою у вивченні та використанні.
5. Відповідати загальноприйнятим стандартам розробки.

User-friendly interface — це особливості дизайну програмного забезпечення, вебсайту або іншої технології, які дозволяють користувачам швидко та ефективно взаємодіяти з цими технологіями. Основна мета — забезпечити максимальний комфорт використання технологій та зменшити кількість потенційних помилок, які можуть виникнути під час їх використання.

Простий та інтуїтивно зрозумілий інтерфейс є ключем до забезпечення легкої та ефективної роботи користувача з будь-яким застосунком, без потреби в додатковій допомозі. Це основна умова, яка безпосередньо впливає на робочий процес і швидкість виконання завдань.

Для досягнення цього дизайнер повинен враховувати потреби та вимоги користувачів. Це може включати простоту та ненав'язливість інтерфейсу, легку навігацію, відповідність стандартам і технічним можливостям користувачів, а також використання зручних і привабливих кольорів, шрифтів, зображень та інших елементів дизайну. Як результат, користувач отримує можливість легко взаємодіяти з наданими технологіями, що забезпечує ефективніше та приємніше їх використання. Головне завдання user-friendly інтерфейсу — надати ідеальний досвід використання продукту.

Одним із важливих завдань є забезпечення надійності інформаційної системи та захисту даних. Надійність інформаційної системи — це здатність системи зберігати в межах встановлених значень усі параметри, що характеризують її здатність виконувати потрібні функції в заданих режимах роботи.

Для запобігання витокам даних і забезпечення безпеки важливої інформації необхідно забезпечити надійний захист збережених даних.

Інформація, що міститься в базі даних АРМ, використовується виключно для ведення бізнесу та роботи з клієнтами й постачальниками. Збережені дані мають цінність як об'єкт інформаційної власності, тому забезпечення конфіденційності є пріоритетним завданням, особливо при роботі з людьми, які надають особисту контактну інформацію, таку як номери телефонів або адреси електронної пошти.

Забезпечення та підтримання високого рівня захисту даних значною мірою залежить від самого користувача. Для створення та функціонування розроблюваного проєкту буде використовуватися виключно сертифіковане програмне забезпечення.

Одним із ключових принципів запобігання витокам інформації та подібним сценаріям є усвідомлене використання ПЗ та пристроїв, на яких працює АРМ і зберігаються дані. Уникнення використання піратського ПЗ, своєчасне оновлення сигнатур безпеки (які зазвичай завантажуються разом з оновленнями операційної системи) є важливими факторами, що забезпечують надійність інформаційної системи та збереження даних.

В іншому випадку вся відповідальність за втрату або поширення інформації, що зберігається в базі, цілком і повністю лежить на користувачеві, у цьому випадку на фізичній особі-підприємці, для якого й розробляється ця система.

І, зрештою, доступ до пристрою, на якому встановлено АРМ, має бути суворо обмежений, щоб запобігти впливу третіх осіб на збережені дані або спробам порушення конфіденційності даних, наданих клієнтами та постачальниками продукції.

Використання загальноприйнятих стандартів розробки забезпечує можливість підтримки проєкту іншими розробниками. Ці стандарти є результатом роботи спільноти Java-розробників, що дозволяє іншим ефективно взаємодіяти з проєктом і, у разі необхідності, вносити зміни для модифікації, масштабування або виправлення помилок у розроблюваній інформаційній системі.

Універсальність розроблюваної інформаційної системи полягає в тому, що за умови внесення мінімальних змін, ця система може бути використана для подібних підприємств із подібними вимогами та завданнями в інших сферах бізнесу.

2.2 Рішення щодо інформаційного забезпечення

Розробка структури є важливим етапом проєктування інформаційної системи. Вона містить визначення таблиць, полів, зв'язків та інших об'єктів, необхідних для зберігання та організації даних.

Першим значущим кроком у розробці структури майбутньої бази даних є осмислення основних вимог і правил проєктування. Визначення сутностей включає ідентифікацію основних об'єктів і концепцій, які зберігатимуться в базі даних. Для кожної із цих сутностей мають бути визначені атрибути, які описують значущі характеристики цих сутностей.

Аналіз потенційної взаємодії дозволяє сформувати правильну модель зв'язків і побудувати коректну модель бази даних. Нормалізація покликана усунути дублювання даних і забезпечити їх ефективне зберігання.

Моделювання даних використовується для визначення й аналізу вимог до даних, які будуть необхідні для підтримки бізнес-процесів у організаціях за допомогою певних інформаційних систем. Цей процес є співпрацею фахівців із моделювання даних та зацікавлених сторін бізнесу. У контексті інтеграції бізнес-процесів моделювання даних відіграє одну з ключових ролей, оскільки доповнює моделювання бізнес-процесів і спрощує проєктування майбутньої бази даних.

Концептуальне проєктування — це побудова семантичної моделі предметної області, тобто інформаційної моделі найвищого рівня абстракції. Така модель створюється без орієнтації на конкретну СУБД чи модель даних [5].

На основі вивчення предметної області, вимог до інформаційної системи та потреб замовника була розроблена концептуальна модель даних і змодельована взаємодія об'єктів. Кожен об'єкт у предметній області має атрибути, які відображають його характеристики. Ті атрибути, за якими можна ідентифікувати сам об'єкт, є його ідентифікаторами.

У результаті концептуального проєктування в предметній області проєктованої системи були виділені такі сутності: клієнт, постачальник, мед, звіти про продаж і закупівлю. Було створено модель бази даних АРМ у вигляді ER-діаграми, яка представлена на рисунку 2.2.

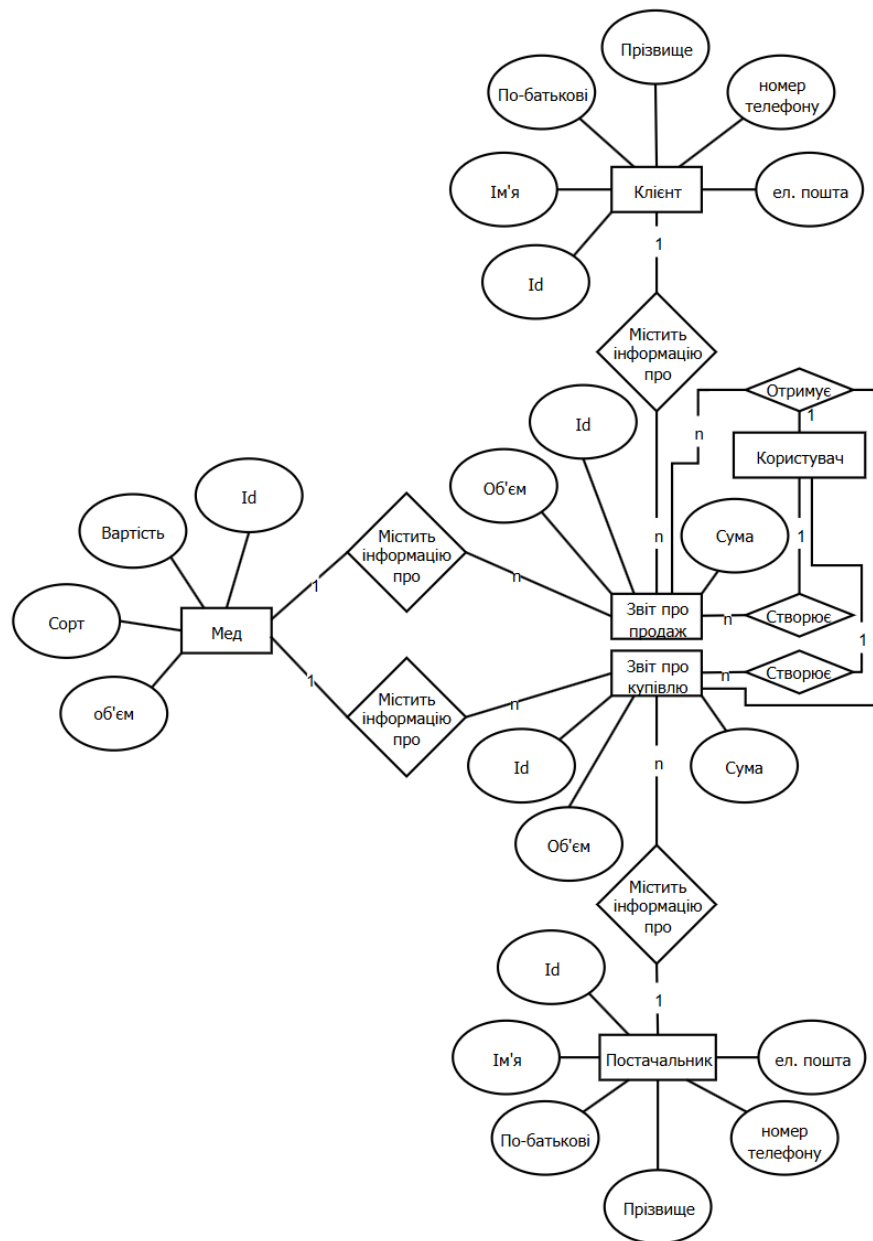


Рисунок 2.2 — Концептуальна модель БД

Термін "логічна модель даних" іноді використовується як синонім "моделі предметної області" або як її альтернатива. Логічна модель даних для проєктованого АРМ підприємства з оптової закупівлі меду представлена на рисунку 2.3.

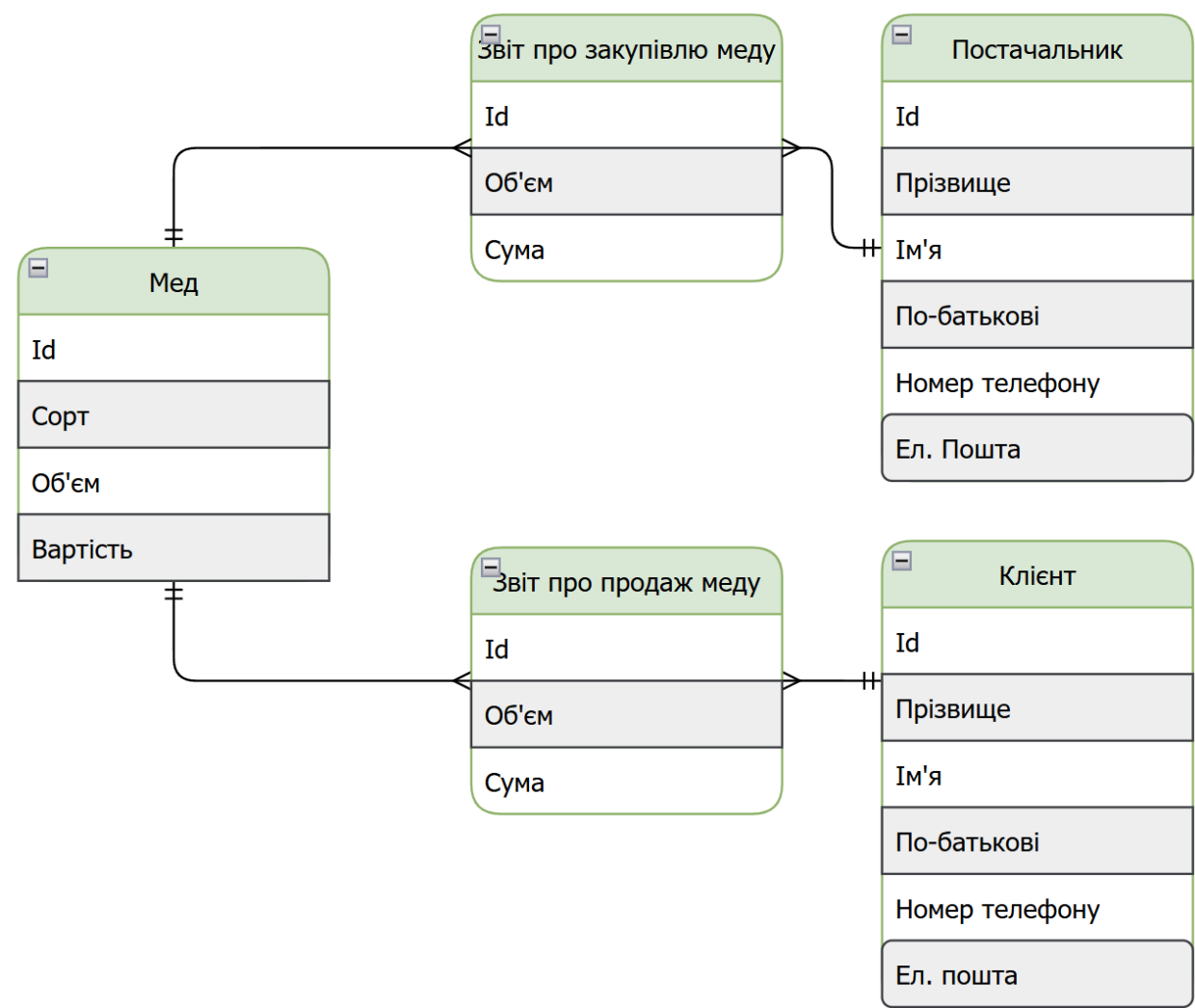


Рисунок 2.3 — Логічна модель бази даних

Ці два поняття тісно пов’язані одне з одним і мають спільні цілі. Модель предметної області більше охоплює поняття предметної області, ніж структуру даних, які пов’язані з цією областю.

Процес розробки встановлює основу, фундамент проєктованої бази, визначаючи її структуру відповідно до обраної предметної області. Логічна структура бази даних має ґрунтуватися на структурах, визначених у концептуальній моделі, оскільки вона описує семантику інформаційного

контексту, який має відображати логічна модель. Ця структура визначається у вигляді реляційних таблиць, включаючи їхні логічні зв'язки та складові.

У процесі формування структури кожної таблиці визначаються поля, які відображають атрибути конкретного об'єкта, його опис, тип даних та інші властивості. Також кожній таблиці присвоюється унікальний ключ, що може містити одне або кілька полів і ідентифікує окремий запис у таблиці.

Фізична структура даних є моделлю даних, яка визначає, яким чином представлені дані, і містить усі деталі, необхідні для створення бази даних у СУБД. Вона включає всі необхідні таблиці, стовпці, зв'язки та властивості бази даних для фізичної реалізації, виключаючи самі дані.

Сам процес фізичного проєктування бази даних нерозривно пов'язаний із конкретною СУБД. Між логічним і фізичним проєктуванням існує постійний взаємозв'язок, оскільки рішення, прийняті на етапі фізичного проєктування з метою оптимізації роботи системи, можуть впливати на структуру самої логічної моделі.

Основною метою фізичного проєктування в реляційній моделі даних є створення набору реляційних таблиць і визначення типу, властивостей та обмежень даних, які визначені на основі інформації, представленої в логічній моделі. Також сюди входить визначення структур зберігання цих даних, методів доступу до них і розробка засобів захисту для проєктованої системи.

Фізична модель даних для проєктованого АРМ підприємства з оптової закупівлі меду представлена на рисунку 2.4.



Рисунок 2.4 — Фізична модель бази даних

У процесі проєктування були визначені поля, первинні ключі, а також створена фізична модель, яка містить структуру бази даних, об'єкти системи, їхні характеристики та взаємозв'язки між ними.

Сутності проєктованої системи представлені у вигляді рядків таблиць бази даних і відображаються як об'єкти класів. Зв'язки між класами представлені на діаграмі класів, яка зображена на рисунку 2.5.

Сама діаграма класів — це статичне представлення структури моделі в UML (Unified Modeling Language). Вона призначена для відображення статичних елементів, таких як класи, типи даних, їхній вміст і відносини між ними. Також вона може містити позначення деяких елементів поведінки, однак їхня динаміка розкривається в інших типах діаграм.

Інтерпретація й тип самої діаграми класів можуть залежати від конкретної перспективи та контексту, оскільки класи можуть представляти сутності в аналізі предметної області або частини програмної системи під час реалізації й проєктування.

Відповідно до раніше зроблених висновків, оскільки для зберігання даних використовується реляційна система управління даними PostgreSQL, а реалізація бізнес-логіки виконана на об'єктно-орієнтованій мові програмування Java, для коректної взаємодії цих технологій і оптимізації процесу розробки буде використовуватися методика JPA (Java Persistence API) на основі фреймворка Spring Data JPA.

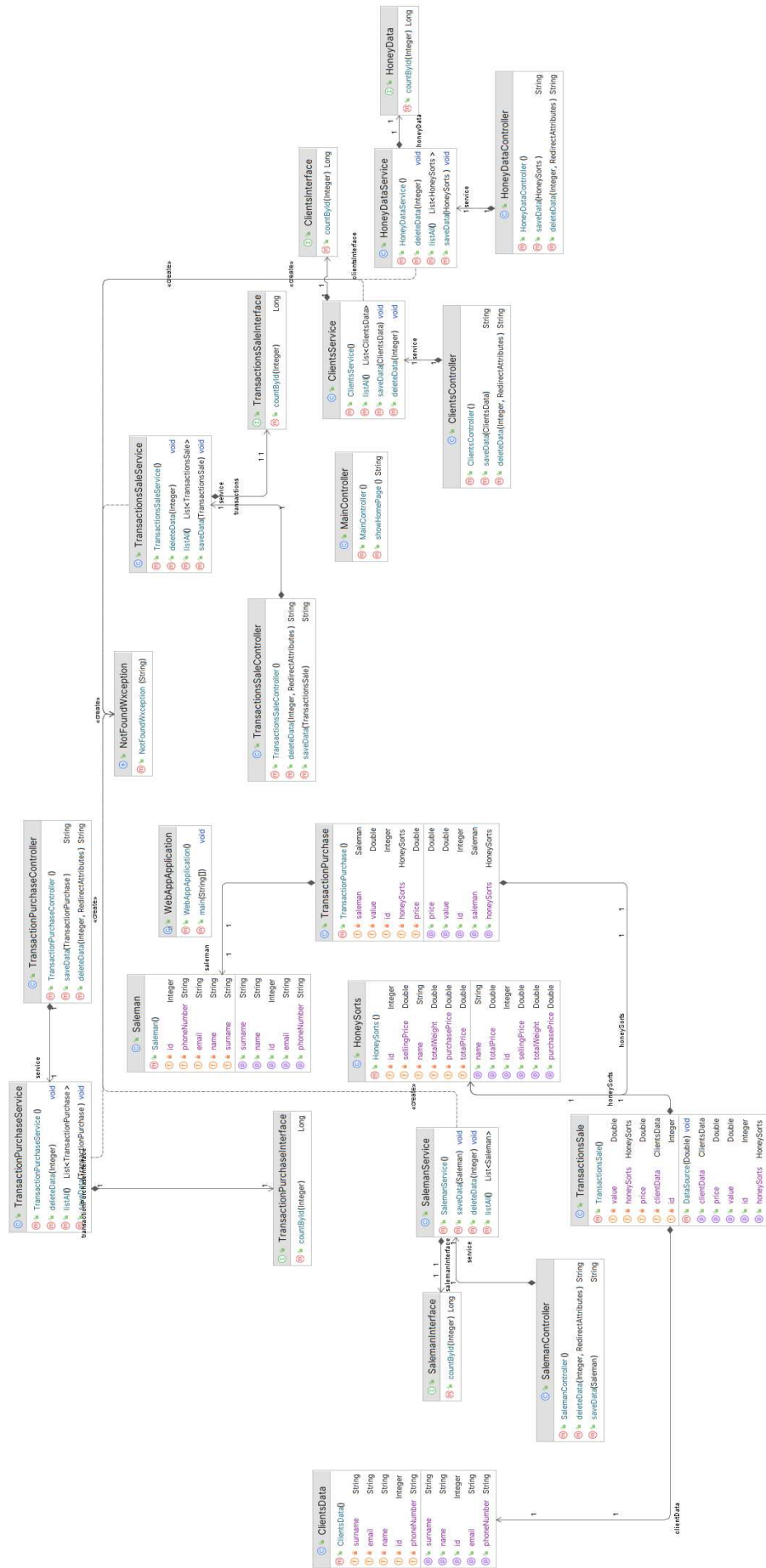


Рисунок 2.5 — Діаграма класів

2.3 Рішення щодо програмного забезпечення

Розробка програмних модулів була виконана за допомогою інтегрованого середовища розробки (IDE) IntelliJ IDEA Ultimate Edition. IntelliJ IDEA — це інтегроване середовище розробки програмного забезпечення для багатьох мов програмування, зокрема Java, JavaScript, Python, розроблене компанією JetBrains [6].

IDE IntelliJ IDEA містить безліч вбудованих інструментів, які дозволяють розробникам швидко й ефективно працювати над реалізацією проєкту. Середовище має багато різноманітних плагінів і налаштувань, спрямованих на спрощення та прискорення процесу розробки, а зручний інтерфейс і редактор коду надають користувачам потужний інструментарій для написання коду.

Головною особливістю IntelliJ IDEA є широкий спектр інструментів, які значно спрощують процес розробки, підвищуючи ефективність розробника й пришвидшуючи цей процес. Додатково, доступний набір плагінів, що спрямовані на різні аспекти розробки програмного забезпечення. Наприклад, дуже зручний модуль взаємодії з базами даних, який дозволяє проєктувати структуру бази даних у вигляді діаграми, налаштовувати зв'язки між таблицями та візуалізувати структуру бази даних у вигляді діаграми.

Середовище розробки також дозволяє:

1. Візуалізувати структуру діаграми класів і фізичну модель бази даних.
2. Взаємодіяти з серверами контролю версій та вести облік змін під час розробки застосунків у команді.

Програмний код написаний об'єктно-орієнтованою мовою програмування Java, яка є однією з найпопулярніших мов, оскільки її універсальність дозволяє вирішувати широкий спектр завдань.

За час свого існування Java перетворилася з простої, але універсальної мови програмування на повноцінну екосистему та платформу, що об'єднала

велике співтовариство. Ця платформа включає різні технології для вирішення широкого спектра завдань і використовується для розробки великих вебпорталів, масштабних корпоративних систем, невеликих застосунків і ігор, а також програмного забезпечення для ПК, смартфонів і побутової техніки.

Java Development Kit (JDK) необхідний для розробки й виконання програм мовою Java. JDK перетворює Java-код на байт-код — набір інструкцій для віртуальної машини Java (JVM), що є частиною середовища виконання Java Runtime Environment (JRE).

Java Runtime Environment — це мінімальна реалізація JVM, необхідна для виконання Java-застосунків. JRE складається з віртуальної машини Java Virtual Machine та бібліотек Java-класів [7], JRE розповсюджується безкоштовно й доступна для завантаження для більшості платформ із сайту Oracle.

Java Virtual Machine (JVM) використовує концепцію проміжного байт-коду для забезпечення переносимості між операційними системами та апаратними платформами. Використання байт-коду дозволяє відокремити компілятор від середовища виконання й виконувати компіляцію з різних мов програмування для виконання на JVM [8].

Це дає змогу запускати Java-код на будь-якій платформі, яка підтримується JVM.

Відповідно до результатів проведеного дослідження, обґрунтованого в першому розділі цієї роботи, для розробки були використані технології Spring Framework.

Spring Framework (або скорочено Spring) — це універсальний фреймворк із відкритим вихідним кодом для платформи Java. Він надає рішення для багатьох завдань, з якими стикаються Java-розробники та організації, що прагнуть створити інформаційну систему на базі платформи Java.

Через широку функціональність складно виділити найбільш значущі структурні елементи, з яких складається фреймворк. Spring не є повністю

прив'язаним до платформи Java Enterprise, незважаючи на його масштабну інтеграцію з нею, що є однією з ключових причин його популярності.

Spring найбільш відомий як джерело розширень (features), необхідних для ефективної розробки складних бізнес-додатків, без необхідності використання важковагових програмних моделей, які історично домінували в індустрії. Ще однією його перевагою є те, що він упровадив раніше невикористовувані функціональні можливості у сучасні методи розробки, навіть за межами платформи Java.

Spring Framework пропонує узгоджену модель, яка застосовна до більшості типів додатків, створених на основі платформи Java. Вважається, що Spring реалізує модель розробки, засновану на найкращих стандартах індустрії, і робить її доступною в багатьох галузях Java [9].

Spring Boot — це Java-фреймворк із відкритим вихідним кодом, призначений для програмування автономних додатків Spring промислового рівня. Він надає набір бібліотек, що спрощують запуск і керування проєктами.

Spring Boot є розширенням підходу "конвенція поверх конфігурації" для платформи Java Spring і допомагає мінімізувати проблеми конфігурації під час створення додатків на основі Spring Framework.

Spring Boot надає широкий функціонал, який дозволяє використовувати фреймворк для створення мікросервісів, вебдодатків та консольних додатків завдяки:

1. Вбудованому серверу вебдодатків (Tomcat, Jetty або Undertow).
2. Наявності обґрунтованих стартових моделей об'єктів проєкту (POM) для інструментів збірки Maven та Gradle.
3. Автоматичному налаштуванню додатків Spring.
4. Готовим функціональним можливостям, таким як метрики, аналіз працездатності та зовнішні конфігурації.
5. Підтримці Kotlin та Apache Groovy на додаток до Java-коду [10].

Важливою частиною цієї роботи стала розробка зручного та зрозумілого графічного інтерфейсу користувача (GUI). Проєктування візуальної

композиції та тимчасової поведінки GUI є важливою частиною програмування додатків у сфері взаємодії людини і комп'ютера.

Його мета — підвищити ефективність та простоту використання базового логічного дизайну програми, що відноситься до дисципліни дизайну, яка називається зручністю користування.

Методи проєктування, орієнтованого на користувача, застосовуються для забезпечення того, щоб візуальна мова, використана в дизайні, добре підходила для виконання завдань.

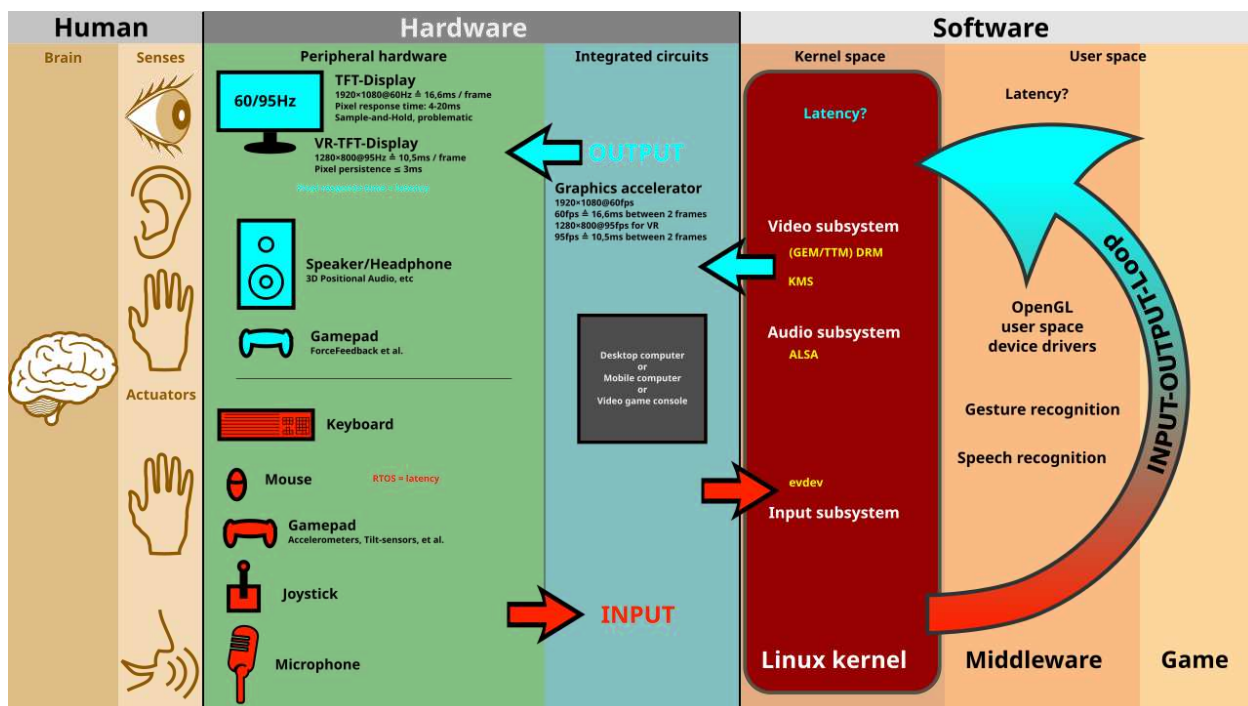


Рисунок 2.6 — Основна концепція роботи GUI

У загальному розумінні GUI — це результат обробки користувацького вводу та, зазвичай, основний інтерфейс для взаємодії людини з пристроєм [11].

Для проєктованої системи інтерфейс користувача реалізований за допомогою мови гіпертекстової розмітки — HTML, яка використовує каскадні таблиці стилів (CSS) для надання інтерфейсу бажаного вигляду.

Веб-дизайн охоплює безліч різних навичок і дисциплін у створенні та обслуговуванні веб-сайтів. Різні сфери веб-дизайну включають веб-графічний дизайн, дизайн користувацького інтерфейсу (UI-дизайн), авторинг, включно зі стандартизованим кодом і спеціалізованим програмним забезпеченням.

Веб-розробка може варіюватися від створення простої статичної сторінки з текстом до складних веб-застосунків, електронної комерції та соціальних мереж. Більш детальний перелік завдань веб-розробки може включати веб-інженерію, веб-дизайн, розробку веб-контенту, роботу з клієнтом, клієнтські/серверні скрипти, налаштування безпеки веб-сервера та розробку електронної комерції [12].

Термін «веб-дизайн» зазвичай використовується для опису процесу проєктування, пов'язаного з дизайном клієнтської частини (front-end) веб-сайту, включно з процесом написання розмітки сторінки [13].

HyperText Markup Language (HTML) — це стандартизована мова гіпертекстової розмітки документів для перегляду веб-сторінок у браузері. Веб-браузери отримують HTML-документ із сервера за протоколами HTTP/HTTPS або відкривають із локального диска, інтерпретуючи код у інтерфейс, який відображається на екрані користувача [14].

Каскадні таблиці стилів (CSS) — формальна мова для опису зовнішнього вигляду документа (вебсторінки), написаного за допомогою мови розмітки (XML, HTML тощо). CSS використовується для налаштування кольорів, шрифтів, стилів, розташування блоків та інших аспектів представлення вебсторінок. CSS дозволяє розділяти логічну структуру вебсторінки та її зовнішній вигляд, що покращує доступність документа, надає гнучкість у його форматуванні та зменшує складність.

Крім того, CSS дозволяє представляти один і той самий документ у різних стилях для екранного перегляду, друку, голосового озвучення чи виводу шрифтом Брайля [15].

2.3.1 Методи розробки програмного забезпечення

АРМ спроектоване з використанням методології об'єктно-орієнтованого програмування (ООП). Ця методологія дозволяє зберігати й обробляти дані як об'єкти, що спрощує й пришвидшує процес розробки. Вона структурує сам

процес розробки, спрощуючи підтримку, оновлення та масштабування проєкту в майбутньому.

Існує багато гідних методологій розробки ПЗ, які є усталеними best practices. Вибір залежить від специфіки проєкту, бюджету та суб'єктивних уподобань.

Для розробки інформаційної системи цього проєкту було обрано інкрементальну модель розробки. Ця модель використовується там, де окремі запити на зміну є чіткими, можуть бути легко формалізовані й реалізовані. Вона дозволяє визначити й систематизувати вимоги до інформаційної системи, а також розробити стратегію розробки та графік впровадження нового функціонала.

Інкрементальна модель дозволяє розділити цикл розробки на невеликі й прості модулі. Кожен із них проходить ретельну перевірку вимог, проєктування, написання коду, впровадження та тестування. Процес розробки починається з випуску базової версії продукту з мінімальним функціоналом. Потім додаються нові функції, які називаються «інкрементами». Робочий процес триває, поки всі заплановані функції не будуть остаточно реалізовані та інтегровані в проєктовану систему [16].

2.3.2 Вибір операційної системи

Вибір операційної системи безпосередньо залежить від уподобань користувача. Linux-системи можуть дозволити зекономити ресурси системи для підвищення продуктивності, але їх налаштування може стати проблемою для користувачів, яким потрібно швидке рішення.

Оптимальним вибором стане використання системи Windows, до якої звикла більшість користувачів. Останньою на цей момент операційною системою Microsoft є Windows 11, представлена 24 червня 2021 року та випущена 5 жовтня того ж року.

Нова версія ОС пропонує оновлений інтерфейс і дизайн порівняно з попередніми версіями, дозволяючи легко налаштовувати систему навіть менш досвідченим користувачам.

Щодо безпеки, Windows 11 працює лише на пристроях із сопроцесором безпеки Trusted Platform Module (TPM) 2.0. У зв'язку з цими нововведеннями системні вимоги Windows 11 були підвищені порівняно з Windows 10. Microsoft офіційно підтримує ОС лише на пристроях із процесором Intel Core восьмого покоління або новіше (за деякими винятками), AMD Ryzen другого покоління або новіше, або на системах на кристалі Qualcomm Snapdragon 850 ARM або новіше, із безпечною завантаженням UEFI і включеним TPM 2.0 [17].

2.3.3 Типові програмні засоби

Створення діаграм під час проєктування й розробки проєкту виконувалося у графічному редакторі Diagrams.net — кросплатформеному програмному забезпеченні для побудови графіків із відкритим вихідним кодом, розробленому на HTML5 і JavaScript. Генерація фізичної моделі бази даних і діаграми класів виконувалася за допомогою вбудованого графічного редактора IntelliJ IDEA.

Для використання інформаційної системи було розроблено веб-застосунок, доступ до якого забезпечується через браузер, наприклад, Microsoft Edge або Google Chrome. Chrome використовує всі новітні технології, дозволяючи створювати й використовувати красивий та інтерактивний користувацький інтерфейс. Він також пропонує гнучку систему налаштувань безпеки та високу швидкість роботи, однак відрізняється "ненажерливістю" під час роботи з багатьма вкладками, що може негативно впливати на слабкі пристрої.

2.4 Рішення щодо технічного забезпечення

Сучасні обчислювальні системи використовують два основних типи носіїв даних: жорсткі магнітні диски (HDD) і твердотільні накопичувачі (SSD).

Жорсткий диск (HDD) — це електромеханічний пристрій для зберігання даних, який зчитує й записує цифрові дані за допомогою магнітного накопичувача з однією або кількома пластинами, що обертаються, покритими магнітним матеріалом [18].

Твердотільний накопичувач (SSD) використовує інтегральні схеми для зберігання даних. SSD-накопичувачі застосовують енергонезалежну флеш-пам'ять NAND, яка зберігає дані в комірках пам'яті. Продуктивність і довговічність SSD залежать від кількості бітів, збережених у комірці, від високопродуктивних одношарових комірок (SLC) до більш доступних, але повільніших чотиришарових (QLC) [19].

Основні переваги SSD:

1. Висока швидкість роботи.
2. Надійність.
3. Відсутність рухомих частин.
4. Компактність, що підвищує енергоефективність.

Технологія NAND — це енергонезалежна флеш-пам'ять, яка використовується в SD-картах, USB-накопичувачах і SSD. Типи пам'яті NAND розрізняються за кількістю збережених у комірці бітів, що впливає на місткість, вартість і довговічність накопичувачів:

1. SLC — один біт даних, висока швидкість, низька місткість.
2. MLC — два біти, менша довговічність порівняно з SLC.
3. TLC — три біти, найбільш поширений тип.
4. QLC — чотири біти, низька ціна, менша довговічність.
5. 3D NAND — багат шарові чипи для збільшення місткості.

Твердотільні накопичувачі подібного типу розраховані на певну кількість записів та стирання даних (P/E). При активному використанні,

частому внесенні змін та перезапису даних, що зберігаються на пристрої, елементи пам'яті починають деградувати, поступово зменшуючи доступний обсяг пам'яті [20].

Формфактор M.2 підтримує NVMe для високошвидкісного зберігання даних і забезпечує сумісність із PCIe [21].

M.2 є одним із найкращих виборів для роботи з носіями даних не тільки в контексті проєктованої інформаційної системи, але й для будь-яких завдань, починаючи з роботи з великими файлами і закінчуючи геймінгом або налаштуванням хмарних сервісів.

Таким чином, для роботи з АРМ підприємцю необхідно придбати портативний пристрій, який дозволить використовувати готове рішення для автоматизації проблемних процесів своєї роботи.

У цьому випадку немає необхідності в організації віддаленого сервера та налаштуванні додаткового обладнання для підтримки роботи бази даних, тому ноутбук стане найкращим рішенням для оперативності та можливості доступу до даних незалежно від місця розташування.

Відповідно до системних вимог, вибір зупинився на ноутбуці Lenovo IdeaPad 1 15AMN7 (82VG00RHRA) Cloud Grey з екраном 15.6" IPS, процесором AMD Ryzen 3 7320U, RAM 16 ГБ та SSD 512 ГБ.

РОЗДІЛ 3. РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ПІДПРИЄМСТВА З ОПТОВОЇ ЗАКУПІВЛІ МЕДУ

В інформаційній системі реалізовано кілька модулів для ведення обліку товару, клієнтської бази, сортів меду та переліку угод. Завдяки інтуїтивно зрозумілому інтерфейсу програмного забезпечення, користувач зможе легко використовувати всі функції АРМ без попередньої підготовки.

Модулі та форми заповнення розроблені так, щоб забезпечити швидкість і зручність формування звітів, роботи з базою даних, а також мінімізувати можливі помилки під час роботи із системою.

Функціонал дозволяє відображати інформацію у вигляді таблиць, які нагадують списки, для кращої візуальної ясності. Сам застосунок виконано в ненав’язливих білих і сірих тонах, а активні вкладки та кнопки виділяються м’яким жовтим кольором на загальному фоні.

Таблиця 3.1. Приклади елементів графічного інтерфейсу

Зовнішній вигляд елемента інтерфейсу	Опис та функціонал
	Приклад того, як підсвічується активна вкладка, на якій у даний момент знаходиться користувач
	Кнопка для формування нових звітів
	Поле форми заповнення з випадаючим списком
	Кнопка додаткових дій для зміни або видалення даних
	Приклад неактивної кнопки форми, в якій не заповнені обов’язкові поля

Додати угоду

×

Тип угоди?*

Закупівля меду

Продаж меду

Інформація клієнта*

Прізвище та ім'я*

Введіть прізвище та ім'я...

Телефон*

+380 _ _ _ _ _

Ел. пошта

Введіть ел. пошту...

Інформація продукту*

Назва меду*

Виберіть назву... ^

Об'єм*

0.00 кг.

Додати сорт

009	Соняшник	320.00 грн. / кг.
008	Ріпак	320.00 грн. / кг.
007	Різнотрав'я	560.00 грн. / кг.
006	Польовий	320.00 грн. / кг.
005	Конюшиний	320.00 грн. / кг.

Не вибрано

0.00 грн.

0.00 кг.

0.00 грн.

Створити угоду +

Рисунок 3.1 — форма для додання нової угоди

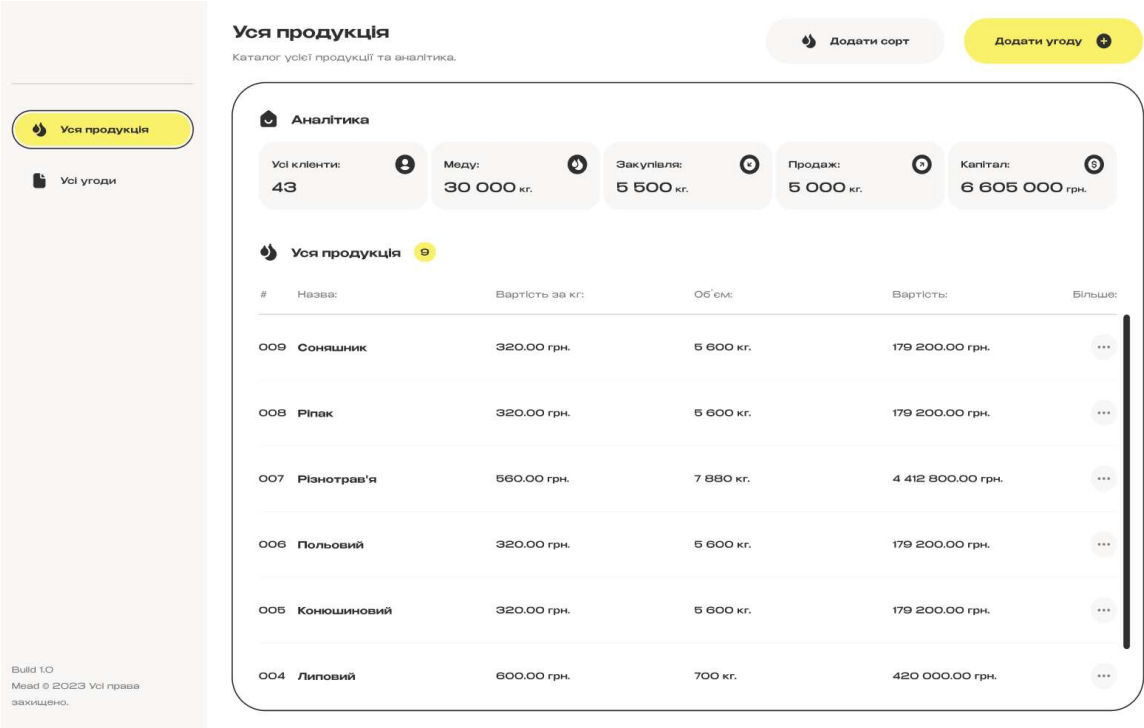


Рисунок 3.1 — Вебінтерфейс для взаємодії користувача з БД

3.1 Інструкція щодо формування та ведення бази даних підприємства

Для формування бази даних користувачеві необхідно зібрати всю інформацію про сорти меду, клієнтів, закупівлі та продажі для її внесення в базу даних інформаційної системи.

База даних цієї системи створюється, налаштовується та управляється за допомогою графічного клієнта PostgreSQL — pgAdmin 4. Використовуючи програмне забезпечення, надане підприємцю, зібрані дані вводяться в систему, формуючи звіти про угоди та виконуючи автоматичний розрахунок залишків товару на складах.

Відповідно до вимог і користувацької угоди, редагувати й видаляти дані може виключно підприємець, оскільки він є єдиним працівником підприємства. Він також несе відповідальність за збереження конфіденційних даних, наданих клієнтами та постачальниками.

Копіювання даних здійснюється за таким графіком:

- 1. Щоденні копії зберігаються протягом тижня.

2. Щотижневі копії зберігаються протягом місяця.
3. Щомісячні копії зберігаються протягом двох років.
4. Щорічні резервні копії зберігаються назавжди.

3.2 Підготовка до роботи з інформаційною системою

Перед початком роботи користувачеві необхідно ознайомитися з інструкцією щодо формування та ведення бази даних. Після налаштування ПК, заповнення бази даних і налаштування резервного копіювання можна приступати до роботи з інформаційною системою.

3.3 Інструкція на випадок виникнення помилок і збоїв у роботі системи

Якщо під час роботи програмного забезпечення або технічних засобів виникла помилка, користувач повинен повідомити про це розробнику для визначення й виправлення можливих помилок.

Якщо проблема пов'язана з технічним обладнанням, необхідно звернутися до сервісного центру для визначення й усунення проблем із «залізом» або периферією, яка використовується для роботи.

У разі небажаного втручання в роботу системи, порушення цілісності файлів або втрати даних, користувачеві необхідно завантажити попередню робочу резервну копію, попередньо створивши ще одну її копію.

3.4 Висновки

Результатом реалізації цього проєкту є засіб для оптимізації робочого часу підприємця, що є зручним інструментом для зберігання й пошуку різних даних, проведення аналітики та прогнозування прибутку.

РОЗДІЛ 4. ТЕХНІКО-ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ РОЗРОБКИ

4.1 Вступ

У сучасному бізнес-середовищі підприємства, що займаються виробництвом та збутом меду, все більше звертаються до впровадження інформаційних систем для автоматизації процесів управління та обліку.

Використання таких систем дозволяє не тільки оптимізувати внутрішні процеси, але й забезпечити ефективну роботу зі збутом продукції, відстежувати запаси, управляти замовленнями та покращувати взаємодію з постачальниками і клієнтами.

Хоча інформаційні системи в агропромисловому секторі, зокрема в сфері збуту меду, вже мають певний досвід застосування, вони тільки починають набирати популярності як важливий інструмент для розвитку бізнесу. З розвитком цифрових технологій і потребою в автоматизації, підприємства з продажу меду все частіше впроваджують сучасні інформаційні системи для покращення управлінських та виробничих процесів.

Створення такої системи потребує значних інвестицій у проектування, закупівлю необхідного обладнання та підтримку її безперебійної роботи. Оцінка економічної вигоди від впровадження інформаційної системи вимагає порівняння витрат на її розробку з отриманими перевагами. Використовуються актуальні тарифні ставки та розрахунки для оцінки окупності інвестицій, що дозволяє зрозуміти доцільність впровадження цієї технології для підприємства по збуту меду.

4.2 Розрахунок вартості створення інформаційної системи управління контентом web-порталу

Загальні витрати на проектування, реалізацію та впровадження інформаційної системи можна обчислити за формулою:

$$G = C_{pr} + C_{hw} + C_{sw} + C_{il}$$

де:

- C_{pr} — витрати на проектування та реалізацію системи;
- C_{hw} — витрати на закупівлю та модернізацію комп'ютерної техніки та периферійних пристроїв;
- C_{sw} — витрати на придбання та установку програмного забезпечення;
- C_{il} — витрати на первинний запуск системи, включаючи навчання персоналу.

Витрати на проектування та реалізацію (C_{pr}):

$$C_{pr} = C_s + C_d + C_{ea} + C_o$$

де:

- C_s — заробітна плата розробників (програмістів, дизайнерів, тестувальників);
- C_d — відрахування на соціальні заходи (47,59% від фонду оплати праці);
- C_{ea} — витрати на експлуатацію та амортизацію обладнання;
- C_o — накладні витрати (50-150% від витрат на оплату праці).

1) Розробка підсистеми вимагає 2 місяці (експертна оцінка, яка базується на досвіді розробки систем такого класу). Заробітна платня розробників за місяць дорівнює 300\$, що в конвертації на українську гривню дорівнює приблизно 12500 грн (відповідно до сучасних розцінок та попиту на спеціалістів такого профілю). Кількість розробників - одна людина.

$$C_s = 1 * 12500 * 2 = 25000 \text{ (грн.)}$$

$$C_d = 25000 * 0,475 = 11875 \text{ (грн.)}$$

2) Витрати на амортизацію ЕОМ, на яких виконується розробка:

$$G_a = \left(\frac{V_{EOM}}{T_s} \right) * \left(\frac{T_{dev}}{12} \right)$$

де:

V_{EOM} = 20000 грн. — балансова вартість ЕОМ

T_s — 5 років - термін служби ЕОМ;

T_{dev} = 0,5 року час необхідний для створення системи виражений в місяцях, і тому ділиться на 12, щоб одержати час, що виміряний в роках.

$$G_a = \left(\frac{20000}{5} \right) * 0,5 = 2000 \text{ (грн.)};$$

3) витрати на експлуатацію персонального комп'ютера, на якому виконується розробка, складаються з оплати споживаної ним електричної енергії:

$$G_e = P_{EOM} * T_{dev} * N_{WD} * N_{HWD} * E_v$$

де:

P_{EOM} — 0,75кВт/год - потужність персонального комп'ютера;

T_{dev} — 2 місяці тривалість розробки;

N_{WD} — 21 день кількість робочих днів у місяці;

N_{HWD} — 8 годин - кількість годин у робочому дні;

E_v — 4.32 грн ціна за 1 кВт електроенергії

$$G_e = 0,75 * 2 * 21 * 8 * 4,32 = 1088,64 \text{ (грн.)}$$

$$C_{ea} = 2000 + 1088,64 = 3088,64 \text{ (грн.)}$$

$$C_o = 25000 * 0,5 = 12500 \text{ (грн.)}$$

$$C_{pr} = 25000 + 11875 + 3088,64 + 12500 = 52463,64 \text{ (грн.)}$$

Витрати на ліцензійне програмне забезпечення не враховуються даному розрахунку, оскільки в роботі системи буде використовуватись вільне, безкоштовне ПЗ з відкритими вихідними кодами Для запровадження системи потрібно враховувати витрати на освіту персоналу та вартість програмного забезпечення.

Витрати на впровадження системи:

Вартість програмного забезпечення — 0 грн (використовуються безкоштовні програми з відкритим кодом);

Вартість підготовки кадрів — 500 грн.

Таким чином витрати на проектування, кодування, реалізацію та впровадження системи складають:

$$G = C_{pr} + C_{hw} + C_{sw} + C_{il} = 52463,64 + 500 = 52963,64 \text{ (грн.)}$$

Витрати на розробку та експлуатацію системи в перший рік експлуатації

$$G_{SEY} = G + G_{TE}$$

де:

G_{TE} — експлуатаційні витрати на апаратне (технічне) забезпечення за рік.

$$G_{TE} = A_a + G_m + G_e$$

де:

A_a — амортизаційні відрахування на обладнання (устаткування);

G_m — витрати на основні і допоміжні матеріали;

G_e — витрати на електроенергію.

Амортизаційні відрахування на обладнання (устаткування) становлять 60% балансової вартості в рік (відповідно до діючого законодавства)

$$A_a = 500 * 0,6 = 300 \text{ (грн.)}$$

В масштабі підприємства річні витрати на основні та допоміжні матеріали (флеш-накопичувачі тощо) визначаються в розмірі 50% вартості основного устаткування

$$G_m = 500 * 0,5 = 250 \text{ (грн.)}$$

Річний обсяг роботи персонального комп'ютера у нормо-годинах

$$W_{PC} = 253,3 * T_d$$

де T_d — середнє добове завантаження обладнання (близько 5 годин);

253,3 — середня кількість робочих днів у році

$$W_{PC} = 253,3 * 5 = 1266,5 \text{ (год.)}$$

Витрати на електроенергію під час 1266,5 годин роботи устаткування в рік при потужності ЕОМ 0,75 кВт/год та ціні 4,32 грн за 1 кВт електроенергії складуть:

$$G_e = 1266,5 * 4,32 * 1,75 = 9574,74 \text{ (грн.)}$$

$$G_{TE} = 300 + 250 + 9574,74 = 10124,74 \text{ (грн.)}$$

Отже у перший рік витрати на створення та експлуатацію системи складуть:

$$G_{SEY} = 52963,64 + 10124,74 = 63088,38 \text{ (грн.)}$$

4.3 Розрахунок економічної ефективності

Економічна доцільність впровадження інформаційної системи базується на наступних перевагах:

- Зниження витрат часу на обслуговування системи завдяки її високій відмовостійкості та автоматичному масштабуванню.
- Можливість надання інформації в оренду користувачам, що дозволяє знизити витрати на її зберігання і обробку.

Ці переваги будуть досягнуті за рахунок полегшення користування системою, підвищення її надійності та стабільності, а також посилення рівня безпеки. Оцінка ефективності показує, що це зниження витрат часу може скласти до 50% від теперішнього рівня.

Хоча важко оцінити фінансово зменшення кількості помилкових або необережних рішень, система також покращує оперативність управління, організацію робочих процесів та своєчасне отримання необхідної інформації для прийняття рішень.

Завдяки впровадженню цієї системи вдасться звільнити 0,1 робочого часу для кожного з 50 користувачів, що еквівалентно вивільненню 5 робітників.

Щоб оцінити прямий економічний ефект, розрахуємо економію від вивільнення 5 робітників. Річна заробітна плата за рахунок звільненого часу складе:

$$\Delta C_n = 5000 * 5 * 12 - 12674,16 = 287325,84 \text{ (грн.)}$$

Далі розраховуємо річний економічний ефект, враховуючи нормативний коефіцієнт ефективності інвестицій у галузь (для обчислювальної техніки він становить 0,5) та додаткові капітальні витрати:

$$E_y = \Delta C_n - E_n * k$$

де:

$$E_y = 287325,84 - 0,5 * 63088,38 = 255781,65 \text{ (грн.)}$$

Тепер визначимо строк окупності системи, поділивши витрати на створення на економічний ефект:

$$T = \frac{k}{\Delta C_n}$$

$$T = \frac{63088,38}{287325,84} = 0,22 \text{ року}$$

Отже, витрати на розробку та впровадження системи для підприємства, що займається оптовою закупівлею медової продукції, окупляться протягом 0,22 року.

4.4 Підсумки

У результаті проведених розрахунків було визначено кілька ключових економічних показників, що підтверджують доцільність створення та впровадження інформаційної системи для позашкільного навчального закладу.

1. Витрати на створення системи:

- Загальні витрати на розробку, проектування та впровадження системи складають 52 963,64 грн.

2. Експлуатаційні витрати:

- Щорічні витрати на підтримку та експлуатацію системи складають 255 781,65 грн.

3. Загальні витрати в перший рік:

- У перший рік після впровадження витрати на модернізацію та експлуатацію системи складатимуть 63 088,38 грн.

4. Вивільнення ресурсів:

- Внаслідок автоматизації процесів відбудеться вивільнення п'яти умовних фахівців.
- Річна сума вивільненої заробітної плати складає 287 325,84 грн.

5. Термін окупності:

- Термін окупності витрат на створення системи складає 2,6 місяці, що є високим показником економічної ефективності.

6. Економічна доцільність:

- Отримані дані про річний економічний ефект та термін окупності дозволяють зробити висновок про економічну доцільність і виправданість розробки та впровадження інформаційної системи в позашкільному навчальному закладі.

7. Можливість підвищення ефективності:

- Для підвищення економічної ефективності системи є можливість пропонувати її як продукт іншим організаціям та користувачам. Це дозволить зменшити час окупності витрат та отримати додатковий прибуток від продажу продукту.

Таким чином, проведені розрахунки свідчать про високу економічну ефективність проекту, що підтверджує доцільність його впровадження, а також відкриває перспективи для подальшого комерційного використання системи.

РОЗДІЛ 5. ОХОРОНА ПРАЦІ

Основна мета охорони праці полягає в захисті працівників від впливу негативних факторів умов праці, зменшенні ризику захворювань і травматизму, пов'язаних з трудовою діяльністю. Загальна соціальна значущість охорони праці проявляється у підвищенні якості та продуктивності праці, покращенні соціально-економічних показників підприємства. З економічної точки зору, дуже важливими є заходи, спрямовані на поліпшення умов праці та підвищення безпеки, що оцінюється з урахуванням змін соціальних показників.

Метою управління охороною праці є забезпечення безпеки, збереження здоров'я та працездатності людини в процесі праці. Ця мета досягається шляхом виконання різних управлінських функцій, точніше комплексу взаємопов'язаних видів діяльності, які проводяться суб'єктом управління цілеспрямовано на об'єкт управління.

5.1 Характеристики робочого приміщення

Планування робочого приміщення передбачає найбільш оптимальне розміщення виробничих компонентів у просторі, включаючи обладнання, технологічне та організаційне обладнання, а також самого працівника. Вимоги до робочих приміщень, які необхідно враховувати, визначені в НПАОП 0.00 - 1.28 - 10 і включають такі параметри:

1. Розмір приміщення не менше 20 м³ на одну людину.
2. Площа приміщення повинна бути не менше 6 м² на одну людину.

У приміщенні буде працювати одна людина, а параметри приміщення наведені в таблиці 5.1.

Таблиця 5.1. Геометричні параметри приміщення

Параметр	Величина
Довжина	3
Ширина	3
Висота	3
Площа	9 м ²
Об'єм	27 м ³

Характер роботи в приміщенні можна охарактеризувати наступним чином:

1. Робота з текстовими даними — 40%
2. Робота з графічними даними — 60%

Робота виконується сидячи і не вимагає значних фізичних навантажень, тому відноситься до категорії робіт "Легка — 1а" [22]. Робочий день, через специфіку підприємства, не обмежує працівника часовими рамками, і він може виконувати свою роботу в будь-який зручний для нього час. Загальні характеристики приміщення відповідають установленим вимогам відповідно до НПАОП 0.00 - 1.28 -10.

5.2 Технічні рішення з гігієни праці та виробничої санітарії

5.2.1 Мікроклімат

Основні показники атмосферного середовища в робочій зоні робочих приміщень — це температура, вологість та рух повітря. Підтримка оптимальних і допустимих значень цих параметрів для робочої зони базується на різних факторах, таких як надлишок тепла, тяжкість виконуваної роботи та пори року.

Три основні параметри повинні відповідати встановленим стандартам. В холодну пору року оптимальна температура коливається від 26 до 28°C, допустима — від 25 до 32°C. В теплу пору року оптимальні значення цих показників становлять 29-31°C, а допустимі — 26-35°C.

Щодо вологості, в холодну пору року оптимальні значення коливаються від 40% до 60%, а допустимі — до 75%. В теплу пору року оптимальні і допустимі значення становлять 40-60% та 65% відповідно.

Для приміщень, що розглядаються в даному дипломному проєкті, характерні наступні аспекти, які можуть негативно впливати на здоров'я людини.

З метою захисту працівника від високих і низьких температур, необхідно передбачити використання спеціальних будівельних заходів, систем опалення та вентиляції. У робочому приміщенні температура повинна підтримуватися в межах $+15^{\circ}\text{C}$ і не перевищувати 26°C . Комфортними умовами для роботи більшості людей визначаються температурою близько 21°C при вологості повітря від 30% до 70%.

Вплив температури на організм зазвичай залежить від вологості повітря, яка відіграє важливу роль в терморегуляції організму. При високій вологості понад 75% і температурах вище 30°C людині може загрожувати перегрів через ускладнене відведення тепла до навколишнього середовища. Оптимальні значення для вологості повітря коливаються в межах 40-60%.

Ще одну важливу роль у терморегуляції відіграє рух повітря. Його наявність сприяє відведенню тепла з поверхні тіла. Рекомендована швидкість руху повітря в невеликому робочому приміщенні коливається від 0.25 до 0.5 м/с.

Для забезпечення вентиляції приміщення використовується система вентиляції, яка базується на використанні системи контролю клімату (кондиціонер). Вона не тільки охолоджує (або обігріває) повітря в приміщенні, але й вентилює його, очищаючи від пилу і видаляючи зайву вологу.

5.2.2 Склад повітря робочої зони.

Для визначення наявності шкідливих речовин у складі повітря необхідно збирати проби повітря в робочій зоні за типових робочих умов,

враховуючи основні технологічні процеси, джерела викидів шкідливих речовин та роботу технологічного обладнання. Протягом всього робочого часу необхідно відібрати достатню кількість проб (не менше п'яти), щоб надати точну гігієнічну оцінку стану атмосферного середовища робочого приміщення.

Під час періодичного санітарного контролю вмісту шкідливих речовин у повітрі робочого місця можливо обмежитись визначенням максимально-разової концентрації. Ефективність поглинання часток фільтром не повинна бути менше 95%. Метод повинен забезпечувати вибіркове визначення складу шкідливих речовин у зібраній пробі повітря на рівні, який не перевищує 0,5 ГДК (Гранично допустима концентрація). Тривалість збору проб при визначенні максимально-разової ГДК не повинна перевищувати 30 хвилин. Метод повинен дозволяти визначення складу шкідливих речовин у повітрі на рівні 0,3 ГДК при необмеженому часі збору проб. Метод визначення має забезпечувати специфічне визначення складу повітря на наявність шкідливих речовин в умовах присутності інших речовин, що на даний момент містяться в повітрі робочої зони.

5.2.3 Виробниче освітлення

Освітлення є невід'ємною частиною будь-якого виробничого процесу, оскільки без нього неможливо забезпечити комфортні умови для роботи. У залежності від типу джерела світла, освітлення можна поділити на два основних види: природне та штучне.

1. Природне освітлення — це освітлення, яке надходить до приміщення через вікна або інші прозорі елементи, використовуючи природні джерела світла, головним з яких є Сонце. Зазвичай воно використовується для створення необхідного рівня освітленості протягом дня, коли природні

умови дозволяють використовувати максимально природне освітлення, яке підлаштовується до змін зовнішніх погодних умов.

2. Штучне освітлення — це освітлення, яке створюється за допомогою електричних приладів, таких як лампи, світильники чи інші джерела світла. Штучне освітлення необхідне для забезпечення стабільної освітленості вночі або в умовах, коли природне освітлення не здатне забезпечити достатній рівень видимості.

Ефективне поєднання обох видів освітлення дозволяє оптимізувати освітлювальну систему в робочих приміщеннях та створити сприятливі умови для виконання трудових завдань.

5.2.4 Природне освітлення

У даному випадку враховується одностороннє бічне природне освітлення з орієнтацією на північ, що забезпечує рівномірне освітлення приміщення без сильних контрастів. Враховуючи, що основною одиницею вимірювання є пікселі, відповідно до цього визначається зорове навантаження, яке класифікується як II. Мінімальні розміри об'єктів, які можуть бути розрізнені, становлять 0,15–0,3 мм. Коефіцієнт світлового клімату для цього варіанту освітлення складає 0,85.

Характеристики вікон:

Довжина вікна: $L = 4$ м

Ширина вікна: $W = 2$ м

Площа віконної поверхні: $S = L * W = 8\text{м}^2$

Розрахунок коефіцієнта освітленості:

$$e_H = e_N t = 1.5 * 0.85$$

Відповідно до ДБН В.2.5-28-2006, нормативне значення коефіцієнта природного освітлення (КПО) має складати 1,5%, що вважається достатнім для забезпечення комфортних умов роботи в приміщенні.

5.2.5 Штучне освітлення

Для забезпечення відповідного рівня освітленості в приміщенні, яке має площу 42 м², планується використання люмінесцентних ламп FН 21 W/840 HE G5, які будуть розміщені у двох паралельних рядах. Згідно з нормами ДБН В.2.5-28-2006, необхідний рівень освітленості при використанні таких ламп повинен складати 300 лк.

Основні технічні параметри ламп:

Потужність: 18 Вт

Тип цоколя: G5

Світловий потік: 2100 лм

Індекс передачі кольору: 85

Кількість ламп у кожному ряді має бути шестикратною для досягнення рівномірного і достатнього освітлення в кімнаті. Таким чином, для оптимальної освітленості буде забезпечено баланс між якістю світла і економією енергії.

РОЗДІЛ 6. ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

Сучасна юридична наука активно розвиває та удосконалює екологічні права й обов'язки, зокрема права суб'єктів, які взаємодіють із природою. Наприклад, у Конституції України зазначено: «Кожен має право на безпечне для життя і здоров'я довкілля, а також на відшкодування шкоди, завданої порушенням цього права. Кожному гарантується право вільно отримувати і поширювати інформацію про стан довкілля, якість харчових продуктів та побутових предметів». Аналогічні положення містяться у Законі України «Про охорону навколишнього природного середовища», адже це право є одним із базових прав людини.

Держава зобов'язана забезпечувати виконання санітарно-гігієнічних заходів, спрямованих на покращення та оздоровлення навколишнього середовища, відповідно до цього права. Усі екологічні права громадян захищаються і можуть бути відновлені у судовому порядку.

Закон «Про охорону навколишнього природного середовища» також визначає обов'язки громадян поряд із їхніми правами. Незалежно від того, чи є люди користувачами природних ресурсів, вони повинні охороняти природу, раціонально використовувати ресурси й уникати завдання шкоди. Окрім цього, громадяни, підприємства, установи та організації як спеціальні користувачі природних ресурсів мають додаткові обов'язки.

Так, плата за спеціальне використання природних ресурсів визначається на основі нормативів і лімітів використання, які враховують розташування, якість, можливість переробки та зберігання ресурсів. Окрім того, суб'єкти спеціального використання природи повинні сплачувати кошти за забруднення довкілля, зокрема за викиди забруднюючих речовин в атмосферу, води чи ґрунт.

Контроль у сфері природокористування та охорони довкілля здійснюється через перевірки, нагляд, інвентаризацію та експертизи. Цей контроль можуть проводити як уповноважені державні органи, так і

громадські організації. Державний контроль покладено на ради народних депутатів, державні адміністрації та Міністерство охорони навколишнього природного середовища разом із його місцевими органами.

Вантажоперевезення, особливо автомобільні та морські, мають значний вплив на екологічний стан планети. Основна шкода пов'язана з викидами парникових газів, таких як вуглекислий газ (CO_2), метан (CH_4) і оксиди азоту (NO_x), які утворюються в процесі спалювання пального. Автотранспорт є головним джерелом забруднення повітря в містах та на автомагістралях. Він сприяє утворенню смогу, підвищує концентрацію дрібнодисперсних частинок і токсичних речовин в атмосфері, що негативно впливає на здоров'я людини та екосистеми. Крім того, інтенсивний рух вантажного транспорту підвищує рівень шумового забруднення, яке чинить стресовий вплив на флору, фауну та людину.

Морські перевезення, попри свою енергоефективність, також завдають значної шкоди екології. Використання важкого пального, що містить велику кількість сірки, призводить до викидів сірчаних сполук, які викликають кислотні дощі та погіршують якість повітря. Крім того, аварії танкерів та витік нафтопродуктів спричиняють довготривалі екологічні катастрофи, забруднюючи океани та знищуючи морську флору і фауну. Вантажоперевезення залізницею вважаються менш шкідливими, але і вони не позбавлені екологічних ризиків, таких як забруднення ґрунту і водойм хімікатами у разі аварій.

Також значний вплив на природу має інфраструктура, пов'язана з вантажоперевезеннями. Будівництво доріг, портів та логістичних хабів призводить до знищення природних екосистем, вирубки лісів та зміни ландшафту. Підвищений попит на перевезення в умовах глобалізації посилює ці процеси. Для пом'якшення наслідків необхідні заходи, такі як перехід на екологічно чисті види пального, розвиток електротранспорту, посилення стандартів на викиди та оптимізація логістичних процесів.

Розробка засобів для зменшення забруднення навколишнього середовища вантажним автотранспортом, який здійснює доставку товарів на склад, включає технічні, організаційні та нормативні підходи.

1. Технічні рішення

1. Ефективні двигуни: Розробка та впровадження двигунів із зниженим споживанням пального та меншим рівнем викидів шкідливих речовин, зокрема дизельних двигунів із системою селективного каталітичного відновлення (SCR) і технологією AdBlue.
2. Електромобілі та гібридні вантажівки: Використання електричних або гібридних транспортних засобів для доставки товарів на склад. Такі транспортні засоби зменшують викиди CO₂ та інших забруднюючих речовин.
3. Альтернативні види пального: Заміна традиційного пального екологічно безпечними варіантами, наприклад, природним газом (CNG/LNG), воднем або біопаливом.
4. Аеродинамічні покращення: Встановлення аеродинамічних обтікачів на вантажівки для зменшення опору повітря, що знижує витрати пального.

2. Організаційні заходи

1. Оптимізація маршрутів: Використання інтелектуальних систем логістики для мінімізації відстаней і часу поїздки. Це знижує витрати пального і, відповідно, рівень забруднення.
2. Спільне використання транспорту: Концепція "вантажного карпулінгу", коли кілька компаній об'єднують свої перевезення для зменшення кількості рейсів.
3. Технічне обслуговування: Регулярний контроль стану транспортних засобів, включно із заміною фільтрів і шин, що сприяє зменшенню викидів.

4. Перевантажувальні центри: Створення проміжних складів для розподілу довгих маршрутів і перевезення вантажів екологічно чистими видами транспорту (наприклад, електричними фургонами).

3. Нормативні та стимулюючі заходи

1. Стимули для екологічного транспорту: Державні субсидії та податкові пільги для компаній, що використовують екологічно безпечний транспорт.
2. Норми викидів: Посилення стандартів викидів для вантажного автотранспорту (наприклад, норми Euro-6).
3. Зони з низьким рівнем викидів: Створення поблизу складів екологічних зон, де дозволено рух лише транспорту з низьким рівнем забруднення.
4. Навчання водіїв: Програми підвищення кваліфікації водіїв, спрямовані на економний стиль водіння (eco-driving), що знижує витрати пального і викиди.

Потенційні перспективи

Інтеграція всіх цих заходів дозволяє мінімізувати екологічний вплив вантажоперевезень, покращити якість повітря та знизити споживання ресурсів. Інноваційні технології у поєднанні з продуманим управлінням логістичними процесами створюють стійку та екологічно безпечну систему доставки товарів.

ВИСНОВКИ

В ході проведеного дослідження були вивчені особливості та завдання таких систем бізнес-обліку, як Вімр та платформи 1С, їх переваги та недоліки. Перш за все, хочеться відзначити універсальність платформи 1С, яка має можливість гнучкого налаштування під конкретні завдання підприємства завдяки наявності вбудованої мови програмування та інтерпретатора. Це дозволяє створювати власні модифікації та налаштування, тому серед різних великих компаній буде важко знайти два схожі рішення.

Вімр же є свіжим сучасним рішенням, яке ще знаходиться в процесі активного розвитку. Він відрізняється сучасним інтерактивним інтуїтивно-зрозумілим інтерфейсом і простотою використання. Однак він все ще обмежений у своїй функціональності порівняно з можливостями платформи 1С.

Java відрізняється своєю популярністю та активним співтовариством, яке постійно розвиває і створює нові технології, завдяки чому сама мова не стоїть на місці.

Для розробки корпоративних рішень платформа Java пропонує вибір між стеками Jakarta Enterprise та Spring Framework. Spring, за довгі роки, став майже єдиною платформою для розробки інформаційних систем і веб-додатків. Він надає прості та ефективні рішення, які дозволяють розробникам швидко збирати та розгортати проекти.

Jakarta EE, на відміну від Spring, не може забезпечити просту реалізацію, але надає більш широкий спектр технологій, однак не відрізняється гнучкістю. Зате вона чудово підходить для проектування великомасштабних корпоративних систем, проблематика яких полягає в збільшенні навантаження та масштабованості.

Проведення аналізу стало важливим кроком у розумінні засобів та технологій, необхідних для проектування інформаційної системи підприємства торгівлі медовою продукцією. На основі отриманих результатів

були вибрані найбільш доречні платформи та методи розробки майбутнього АРМ.

Отримані результати можуть бути використані як рекомендації для розробки інших подібних систем, а розроблений додаток, згідно з результатами тестування, досягнув мети оптимізації робочого процесу та раціоналізації витрат часу на роботу з даними.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Enterprise resource planning (ERP) [Електронний ресурс] — URL: <https://ru.wikipedia.org/wiki/ERP> (дата звернення 21.09.2024)
2. Система обліку Business Impulse (Bimp) — URL: <https://bimpsoft.com/> (дата звернення 24.09.2024)
3. 1C:Enterprise for In-House Software Development. Strengths and Weaknesses of the platform [Блог] — URL: <https://medium.com/@evilbeaver/1c-platform-for-in-house-software-development-strengths-and-weaknesses-of-1c-enterprise-5ecbc3e3b32a> (дата звернення 22.10.2024)
4. Java Development Kit (JDK) [Електронний ресурс] — URL: https://en.wikipedia.org/wiki/Java_Development_Kit (дата звернення 05.10.2024)
5. Проектування бази даних [Електронний ресурс] — URL: https://uk.wikipedia.org/wiki/Проектування_бази_даних (дата звернення 05.10.2024)
6. IntelliJ IDEA (IDE) [Електронний ресурс] — URL: https://en.wikipedia.org/wiki/IntelliJ_IDEA (дата звернення 20.10.2024)
7. Java Runtime Environment (JRE) [Електронний ресурс] — URL: https://ru.wikipedia.org/wiki/Java_Runtime_Environment (дата звернення 20.10.2024)
8. Создаем байт-код для JVM без компилятора [Блог] — URL: <https://habr.com/ru/companies/otus/articles/726610/> (дата звернення 29.10.2024)
9. Spring Framework [Електронний ресурс] — URL: https://ru.wikipedia.org/wiki/Spring_Framework (дата звернення 05.10.2024)
10. Spring Boot [Електронний ресурс] — URL: https://en.wikipedia.org/wiki/Spring_Boot (дата звернення 06.10.2024)
11. Graphical user interface (GUI) [Електронний ресурс] — URL: https://en.wikipedia.org/wiki/Graphical_user_interface (дата звернення 01.11.2024)
12. Web development [Електронний ресурс] — URL: https://en.wikipedia.org/wiki/Web_development (дата звернення 01.11.2024)

13. Web design [Електронний ресурс] — URL: https://en.wikipedia.org/wiki/Web_design (дата звернення 01.11.2024)
14. HyperText Markup Language (HTML) [Електронний ресурс] — URL: <https://ru.wikipedia.org/wiki/HTML> (дата звернення 01.11.2024)
15. Cascading Style Sheets (CSS) [Електронний ресурс] — URL: <https://ru.wikipedia.org/wiki/CSS> (дата звернення 01.11.2024)
16. Другие модели процессов разработки ПО [Блог] — URL: <https://javarush.com/quests/lectures/questservlets.level15.lecture06> (дата звернення 01.11.2024)
17. Windows 11 [Електронний ресурс] — URL: https://ru.wikipedia.org/wiki/Windows_11 (дата звернення 01.11.2024)
18. Hard disk drive (HDD) [Електронний ресурс] — URL: https://en.wikipedia.org/wiki/Hard_disk_drive (дата звернення 01.11.2024)
19. Solid-state drive (SSD) [Електронний ресурс] — URL: https://en.wikipedia.org/wiki/Solid-state_drive (дата звернення 01.11.2024)
20. Что надёжнее, SSD или HDD? [Блог] — URL: <https://habr.com/ru/companies/timeweb/articles/733886/> (дата звернення 01.11.2024)
21. M.2 (Next Generation Form Factor) [Електронний ресурс] — URL: <https://en.wikipedia.org/wiki/M.2> (дата звернення 01.11.2024)
22. Гігієнічні вимоги до організації роботи з візуальними дисплейними терміналами електронно-обчислювальних машин — URL: <https://zakon.rada.gov.ua/rada/show/v0007282-98#Text> (дата звернення 01.11.2024)

ДОДАТОК А

Сутності



ClientsData.java

```
package com.mycompany.entities;

import jakarta.persistence.*;

@Entity
@Table (name = "clients")
public class ClientsData {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;
    @Column (nullable = false, unique = false, length = 15)
    private String name;
    @Column (nullable = false, unique = true, length = 15)
    private String surname;
    @Column (nullable = false, unique = true, length = 15)
    private String phoneNumber;
    @Column (nullable = true, unique = true, length = 20)
    private String email;

    public ClientsData() {
    }

    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getSurname() {
        return surname;
    }

    public void setSurname(String surname) {
        this.surname = surname;
    }

    public String getPhoneNumber() {
        return phoneNumber;
    }

    public void setPhoneNumber(String phoneNumber) {
```

```

        this.phoneNumber = phoneNumber;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }
}

```



HoneySorts.java

```

package com.mycompany.entities;

import jakarta.persistence.*;

@Entity
@Table (name = "honey_sorts")
public class HoneySorts {
    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public Double getPurchasePrice() {
        return purchasePrice;
    }

    public void setPurchasePrice(Double purchasePrice) {
        this.purchasePrice = purchasePrice;
    }

    public Double getSellingPrice() {
        return sellingPrice;
    }

    public void setSellingPrice(Double sellingPrice) {
        this.sellingPrice = sellingPrice;
    }

    public Double getTotalWeight() {
        return totalWeight;
    }

    public void setTotalWeight(Double totalWeight) {

```

```

        this.totalWeight = totalWeight;
    }

    public Double getTotalPrice() {
        return totalPrice;
    }

    public void setTotalPrice(Double totalPrice) {
        this.totalPrice = totalPrice;
    }

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;
    @Column (nullable = false, unique = true, length = 15)
    private String name;
    @Column (nullable = false, name = "purchase_price")
    private Double purchasePrice;
    @Column (nullable = false, name = "selling_price")
    private Double sellingPrice;
    private Double totalWeight;
    private Double totalPrice;
}

```



Saleman.java

```

package com.mycompany.entities;

import jakarta.persistence.*;

@Entity
@Table (name = "saleman")
public class Saleman {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;
    @Column (nullable = false, unique = false, length = 15)
    private String name;
    @Column (nullable = false, unique = true, length = 15)
    private String surname;
    @Column (nullable = false, unique = true, length = 15)
    private String phoneNumber;
    @Column (nullable = true, unique = true, length = 20)
    private String email;

    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {

```

```

        this.name = name;
    }

    public String getSurname() {
        return surname;
    }

    public void setSurname(String surname) {
        this.surname = surname;
    }

    public String getPhoneNumber() {
        return phoneNumber;
    }

    public void setPhoneNumber(String phoneNumber) {
        this.phoneNumber = phoneNumber;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }
}

```



TransactionPurchase.java

```

package com.mycompany.entities;

import jakarta.persistence.*;

@Entity
@Table (name = "transactions_purchase")
public class TransactionPurchase {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;
    @Column (nullable = false, unique = false, length = 20)
    private Double value;

    @Column (nullable = false, unique = false, length = 20)
    private Double price;

    @ManyToOne
    private Saleman salesman;
    @ManyToOne
    private HoneySorts honeySorts;

    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }
}

```

```

    public Double getValue() {
        return value;
    }

    public void setValue(Double value) {
        this.value = value;
    }

    public Double getPrice() {
        return price;
    }

    public void setPrice(Double price) {
        this.price = price;
    }

    public Saleman getSaleman() {
        return saleman;
    }

    public void setSaleman(Saleman saleman) {
        this.saleman = saleman;
    }

    public HoneySorts getHoneySorts() {
        return honeySorts;
    }

    public void setHoneySorts(HoneySorts honeySorts) {
        this.honeySorts = honeySorts;
    }
}

```



TransactionsSale.java

```

package com.mycompany.entities;

import jakarta.persistence.*;

@Entity
@Table (name = "transactions_sale")
public class TransactionsSale {
    @Id
    @GeneratedValue (strategy = GenerationType.IDENTITY)
    private Integer id;
    @Column (nullable = false, unique = false, length = 20)
    private Double value;

    @Column (nullable = false, unique = false, length = 20)
    private Double price;

    @ManyToOne
    private ClientsData clientData;
    @ManyToOne
    private HoneySorts honeySorts;

    public Integer getId() {

```



```

        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public Double getValue() {
        return value;
    }

    public void setValue(Double value) {
        this.value = value;
    }

    public void DataSource(Double val) {
        this.value = val;
    }

    public Double getPrice() {
        return price;
    }

    public void setPrice(Double price) {
        this.price = price;
    }

    public ClientsData getClientData() {
        return clientData;
    }

    public void setClientData(ClientsData clientData) {
        this.clientData = clientData;
    }

    public HoneySorts getHoneySorts() {
        return honeySorts;
    }

    public void setHoneySorts(HoneySorts honeySorts) {
        this.honeySorts = honeySorts;
    }
}

```

Controllers



ClientsController.java

```

package com.mycompany.controllers;

import com.mycompany.entities.ClientsData;
import com.mycompany.services.ClientsService;
import com.mycompany.entities.NotFoundWxception;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;

```

```

import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

@Controller
public class ClientsController {
    @Autowired
    ClientsService service;

    @PostMapping("/save-client")
    public String saveData(ClientsData clientsData) {
        service.saveData(clientsData);
        return "redirect:/index";
    }

    @GetMapping("/delete-client/{id}")
    public String deleteData(@PathVariable("id") Integer id,
        RedirectAttributes redirectAttributes) {
        try {
            service.deleteData(id);

            return "index";
        } catch (NotFoundException e) {
            redirectAttributes.addFlashAttribute("message",
                e.getMessage());
            return "redirect:/all-transactions-sale";
        }
    }

    @GetMapping("/edit/{id}")
    public String showEditForm(@PathVariable("id") Integer id, Model
        model, RedirectAttributes redirectAttributes) {
        try {
            ClientsService clients = service.get(id);
            model.addAttribute("client", clients);
            return "index";
        } catch (NotFoundException e) {
            return "redirect:/all-transactions-sale";
        }
    }
}

```



HoneyDataController.java

```

package com.mycompany.controllers;

import com.mycompany.services.ClientsService;
import com.mycompany.services.HoneyDataService;
import com.mycompany.entities.HoneySorts;
import com.mycompany.entities.NotFoundWxception;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

@Controller

```

```

public class HoneyDataController {
    @Autowired
    HoneyDataService service;

    @PostMapping("/save")
    public String saveData(HoneySorts sorts) {
        service.saveData(sorts);
        return "redirect:/index";
    }

    @GetMapping("/delete/{id}")
    public String deleteData(@PathVariable("id") Integer id,
        RedirectAttributes redirectAttributes) {
        try {
            service.deleteData(id);

            return "index";
        } catch (NotFoundException e) {
            redirectAttributes.addFlashAttribute("message",
                e.getMessage());
            return "redirect:/index";
        }
    }

    @GetMapping("/edit/{id}")
    public String showEditForm(@PathVariable("id") Integer id, Model
        model, RedirectAttributes redirectAttributes) {
        try {
            HoneyDataService honey = service.get(id);
            model.addAttribute("client", honey);
            return "index";
        } catch (NotFoundException e) {
            return "redirect:/all-transactions-sale";
        }
    }
}

```



SalemanController.java

```

package com.mycompany.controllers;

import com.mycompany.entities.NotFoundWxception;
import com.mycompany.entities.Saleman;
import com.mycompany.services.HoneyDataService;
import com.mycompany.services.SalemanService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

@Controller
public class SalemanController {
    @Autowired

```

```

SalemanService service;

@PostMapping("/save-client")
public String saveData(Saleman saleman) {
    service.saveData(saleman);
    return "redirect:/index";
}

@GetMapping("/delete-client/{id}")
public String deleteData(@PathVariable("id") Integer id,
RedirectAttributes redirectAttributes) {
    try {
        service.deleteData(id);

        return "index";
    } catch (NotFoundException e) {
        redirectAttributes.addFlashAttribute("message",
e.getMessage());
        return "redirect:/all-transactions-sale";
    }
}

@GetMapping("/edit/{id}")
public String showEditForm(@PathVariable("id") Integer id, Model
model, RedirectAttributes redirectAttributes) {
    try {
        SalemanService saleman = service.get(id);
        model.addAttribute("client", saleman);
        return "index";
    } catch (NotFoundException e) {
        return "redirect:/all-transactions-sale";
    }
}
}
}

```



TransactionPurchaseController.java

```

package com.mycompany.controllers;

import com.mycompany.entities.NotFoundWxception;
import com.mycompany.entities.TransactionPurchase;
import com.mycompany.services.HoneyDataService;
import com.mycompany.services.TransactionPurchaseService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

@Controller
public class TransactionPurchaseController {
    @Autowired
    TransactionPurchaseService service;
    @PostMapping("/save-transaction")
    public String saveData(TransactionPurchase

```

```

transactionPurchase) {
    service.saveData(transactionPurchase);
    return "redirect:/index";
}

@GetMapping("/delete-transaction/{id}")
public String deleteData(@PathVariable("id") Integer id,
RedirectAttributes redirectAttributes) {
    try {
        service.deleteData(id);

        return "index";
    } catch (NotFoundException e) {
        redirectAttributes.addFlashAttribute("message",
e.getMessage());
        return "redirect:/all-transactions";
    }
}

@GetMapping("/edit/{id}")
public String showEditForm(@PathVariable("id") Integer id, Model
model, RedirectAttributes redirectAttributes) {
    try {
        TransactionPurchaseService purchaseService =
service.get(id);
        model.addAttribute("client", purchaseService);
        return "index";
    } catch (NotFoundException e) {
        return "redirect:/all-transactions-sale";
    }
}
}
}

```



TransactionsSaleController.java

```

package com.mycompany.controllers;

import com.mycompany.entities.NotFoundWxception;
import com.mycompany.entities.TransactionsSale;
import com.mycompany.services.HoneyDataService;
import com.mycompany.services.TransactionsSaleService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

@Controller
public class TransactionsSaleController {

    @Autowired
    TransactionsSaleService service;
    @PostMapping("/save-transaction")
    public String saveData(TransactionsSale transactionsSale) {
        service.saveData(transactionsSale);
    }
}

```

```

        return "redirect:/index";
    }

    @GetMapping("/delete-transaction/{id}")
    public String deleteData(@PathVariable("id") Integer id,
        RedirectAttributes redirectAttributes) {
        try {
            service.deleteData(id);

            return "index";
        } catch (NotFoundException e) {
            redirectAttributes.addFlashAttribute("message",
                e.getMessage());
            return "redirect:/all-transactions";
        }
    }

    @GetMapping("/edit/{id}")
    public String showEditForm(@PathVariable("id") Integer id, Model
        model, RedirectAttributes redirectAttributes) {
        try {
            TransactionsSaleService sale = service.get(id);
            model.addAttribute("client", sale);
            return "index";
        } catch (NotFoundException e) {
            return "redirect:/all-transactions-sale";
        }
    }
}
}
}

```

Services



ClientsService.java

```

package com.mycompany.services;

import com.mycompany.entities.ClientsData;
import com.mycompany.entities.NotFoundWxception;
import com.mycompany.interfaces.ClientsInterface;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class ClientsService {
    @Autowired
    ClientsInterface clientsInterface;

    public List<ClientsData> listAll() {
        return (List<ClientsData>) clientsInterface.findAll();
    }

    public void saveData(ClientsData clientsData) {
        clientsInterface.save(clientsData);
    }

    public void deleteData(Integer id) throws NotFoundException {

```

```

        Long counter = clientsInterface.countById(id);
        if (counter == null || counter == 0) {
            throw new NotFoundException("Unable to find data by this
ID " + id);
        }
        clientsInterface.deleteById(id);
    }
}

```



HoneyDataService.java

```

package com.mycompany.services;

import com.mycompany.entities.HoneySorts;
import com.mycompany.entities.NotFoundWxception;
import com.mycompany.interfaces.HoneyData;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class HoneyDataService {
    @Autowired
    HoneyData honeyData;

    public List<HoneySorts> listAll() {
        return (List<HoneySorts>) honeyData.findAll();
    }

    public void saveData(HoneySorts sorts) {
        honeyData.save(sorts);
    }

    public void deleteData(Integer id) throws NotFoundException {
        Long counter = honeyData.countById(id);
        if (counter == null || counter == 0) {
            throw new NotFoundException("Could not find any data about
sort with this ID " + id);
        }
        honeyData.deleteById(id);
    }
}

```



SalemanService.java

```

package com.mycompany.services;

import com.mycompany.entities.NotFoundWxception;
import com.mycompany.entities.Saleman;
import com.mycompany.interfaces.SalemanInterface;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service

```

```

public class SalemanService {
    @Autowired
    SalemanInterface salemanInterface;

    public List<Saleman> listAll() {
        return (List<Saleman>) salemanInterface.findAll();
    }

    public void saveData(Saleman saleman) {
        salemanInterface.save(saleman);
    }

    public void deleteData(Integer id) throws NotFoundException {
        Long counter = salemanInterface.countById(id);
        if (counter == null || counter == 0) {
            throw new NotFoundException("Unable to find data by this
ID " + id);
        }
        salemanInterface.deleteById(id);
    }
}

```



TransactionPurchaseService.java

```

package com.mycompany.services;

import com.mycompany.entities.NotFoundWxception;
import com.mycompany.interfaces.TransactionPurchaseInterface;
import com.mycompany.entities.TransactionPurchase;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class TransactionPurchaseService {
    @Autowired
    TransactionPurchaseInterface transactionPurchaseInterface;

    public List<TransactionPurchase> listAll() {
        return (List<TransactionPurchase>)
transactionPurchaseInterface.findAll();
    }

    public void saveData(TransactionPurchase transData) {
        transactionPurchaseInterface.save(transData);
    }

    public void deleteData(Integer id) throws NotFoundException {
        Long counter = transactionPurchaseInterface.countById(id);
        if (counter == null || counter == 0) {
            throw new NotFoundException("Unable to find data by this
ID " + id);
        }
        transactionPurchaseInterface.deleteById(id);
    }
}

```




TransactionsSaleService.java

```
package com.mycompany.services;

import com.mycompany.entities.NotFoundWxception;
import com.mycompany.interfaces.TransactionsSaleInterface;
import com.mycompany.entities.TransactionsSale;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import java.util.List;

@Service
public class TransactionsSaleService {
    @Autowired
    TransactionsSaleInterface transactions;

    public List<TransactionsSale> listAll() {
        return (List<TransactionsSale>) transactions.findAll();
    }

    public void saveData(TransactionsSale transData) {
        transactions.save(transData);
    }

    public void deleteData(Integer id) throws NotFoundWxception {
        Long counter = transactions.countById(id);
        if (counter == null || counter == 0) {
            throw new NotFoundWxception("Unable to find data by this
ID " + id);
        }
        transactions.deleteById(id);
    }
}
```