МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри інтелектуальних
інформаційних систем
_____Юрій КОНДРАТЕНКО
«____»_____2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ МАГІСТРА

ІНТЕЛЕКТУАЛЬНА CRM СИСТЕМА НА ОСНОВІ МЕТОДІВ ЛОГІСТИКИ ТА ABC і XYZ АНАЛІЗУ

Спеціальність 122 Комп'ютерні науки

Освітня програма «Інтелектуальні інформаційні системи»

Здобувач __ Антон САВЧУК
«___»_____ 2024 р.

Керівник канд. фіз.–мат. наук, доцент _____Інесса КУЛАКОВСЬКА
«___»_____ 2024 р.

Миколаїв – 2024

Чорноморський національний університет імені Петра Могили

(повне найменування закладу вищої освіти)

Факультет	Комп’ютерних наук
Кафедра	Інтелектуальних інформаційних систем
Рівень вищої освіти	Другий (магістерський)
Освітній ступень	Магістр
Спеціальність	122 Комп’ютерні науки
Освітня програма	Інтелектуальні інформаційні системи

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних інформаційних систем

_____ Юрій КОНДРАТЕНКО

«____» _____2024 р.

ЗАВДАННЯ

на кваліфікаційну роботу здобувача

Савчука Антона Олександровича

(прізвище, ім’я, по батькові здобувача)

1. Тема кваліфікаційної роботи: «Інтелектуальна CRM система на основі методів логістики та ABC і XYZ аналізу».

Керівник роботи: Кулаковська Інесса Василівна, доцент кафедри ІІС, канд. фіз.–мат. наук. наук.

Затверджена наказом ЧНУ ім. Петра Могили від «03» червня 2024 р. № 140/1.

2. Строк представлення кваліфікаційної роботи «___» _____ 2024 р.

2024 р. Савчук Антон

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні: Створення CRM системи з можливістю використання ABC та XYZ методів логістичного аналізу.

4. Перелік питань, що підлягають розробці: Огляд наявних аналогів та публікацій на тему "CRM система"; Проектування логіки та візуальної складової CRM системи; Проектування бази данних в відповідності до потреб проекту та її наповнення даними; Застосування ABC та XYZ методівлогістичної регресії.

5. Перелік графічних матеріалів: КР – 68 сторінки, 29 рисунки, 2 таблиці, 50 джерел, 1 додаток та презентація.

Керівник роботи _____ Інесса КУЛАКОВСЬКА
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Здобувач _____ Антон САВЧУК
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Дата видачі завдання «07» червня 2024 р.

КАЛЕНДАРНИЙ ПЛАН
кваліфікаційної роботи

Тема: Інтелектуальна CRM система на основі методів логістики та ABC і XYZ аналізу

№	Найменування роботи	Початок	Закінчення	Примітки
1	Отримання завдання на виконання КР	03.06.2024	07.06.2024	Виконано
2	Аналіз предметної області та постановка задачі	10.06.2024	21.06.2024	Виконано
3	Ознайомлення з підприємством. Аналіз існуючих рішень для CRM та закупівельної логістики.	21.06.2024	01.07.2024	Виконано
4	Визначення вимог до CRM–системи, планування проекту.	01.09.2024	25.10.2024	Виконано
5	Розробка ABC–класифікації товарів. Розробка XYZ–класифікації товарів. Розробка графічної оболонки CRM системи. Тестування системи. Виправлення помилок.	26.10.2024	21.11.2024	Виконано
6	Перший попередній захист КР на засіданні комісії кафедри	22.11.2024	22.11.2024	Виконано
7	Корегування роботи за результатами попереднього захисту	23.11.2024	05.12.2024	Виконано
8	Другий попередній захист КР на засіданні комісії кафедри	06.12.2024	06.12.2024	Виконано
9	Доробка та остаточне оформлення КР	07.12.2024	10.12.2024	Виконано
10	Подання КР, її електронної копії та інших документів (відгуку, рецензії) до захисту	16.12.2024	17.12.2024	Виконано

Керівник роботи _____ Інесса КУЛАКОВСЬКА
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Здобувач _____ Антон САВЧУК
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Дата складання календарного плану
«19» червня 2024 р.

АНОТАЦІЯ

до кваліфікаційної роботи
здобувача групи 601м ЧНУ ім. Петра Могили

Савчука Антона Олександровича

на тему: **“ ІНТЕЛЕКТУАЛЬНА CRM СИСТЕМА НА ОСНОВІ МЕТОДІВ ЛОГІСТИКИ ТА ABC і XYZ АНАЛІЗУ”**

Актуальність даного дослідження полягає в необхідності інтеграції логістичних підходів до CRM для підвищення ефективності управлінських рішень та вдосконалення процесів постачання. Використання мови програмування C# дозволяє реалізувати гнучкі та масштабовані рішення, які можуть бути адаптовані під конкретні потреби підприємства.

Об’єктом дослідження є процеси управління взаємовідносинами з клієнтами та логістичними операціями підприємства.

Предметом дослідження є методи та механізми інтеграції закупівельної логістики та ABC і XYZ класифікаційних методів у CRM–систему для оптимізації бізнес–процесів..

Метою є дослідження методів для створення CRM системи з використанням методів закупівельної логістики та впровадження ABC і XYZ класифікаційних методів на мові програмування C#. В рамках роботи будуть розглянуті основи побудови CRM систем, методи аналізу та класифікації товарів і послуг, а також алгоритми, що дозволяють автоматизувати процес управління закупівлями та запасами на підприємстві.

В результаті виконання роботи було досліджено два методи логістичного аналізу (класифікаційні методи ABC і XYZ), досліджено вплив їх внутрішніх параметрів на ефективність роботи алгоритмів, визначено основні переваги та недоліки цих підходів, проведено аналіз подібних програмних продуктів і розроблено програмне забезпечення з реалізацією зазначених методів.

Дана робота складається з трьох розділів. Кожен розділ відповідно присвячений: теоретичним засадам моделювання CRM системи; алгоритмам і методам реалізації додатку; моделюванню і проектуванню CRM системи з методами логістичного аналізу; аналізу отриманих результатів. Загальний обсяг роботи – 85 сторінок. Кваліфікаційна робота містить 2 додатки, 29 рисунків, 2 таблиць і 50 джерел посилання.

Ключові слова: урахування товарів, CRM система, асинхронне програмування, бази даних, фреймворк , XYZ–метод, ABC–метод.

2024 р. Савчук Антон

ABSTRACT

to the qualification work by the student of the group 601m of Petro Mohyla Black Sea National University

Savchuk Anton

“ INTELLIGENT CRM SYSTEM BASED ON LOGISTICS METHODS AND ABC AND XYZ ANALYSIS”

A relevance of this research lies in the need to integrate logistical approaches into CRM systems to improve managerial decision–making efficiency and enhance supply chain processes. The use of the C# programming language enables the implementation of flexible and scalable solutions that can be tailored to the specific needs of enterprises.

An object of research is the processes of managing customer relationships and the logistics operations of an enterprise.

A subject of the research is the methods and mechanisms for integrating procurement logistics and ABC and XYZ classification methods into a CRM system to optimize business processes.

A purpose of the study is research the methods to develop a CRM system utilizing procurement logistics methods and implementing ABC and XYZ classification methods using the C# programming language. The study focuses on the fundamentals of building CRM systems, methods of analyzing and classifying goods and services, and algorithms that enable the automation of procurement and inventory management processes in an enterprise.

As a result of the work, two methods of logistic analysis (ABC and XYZ classification methods) were studied, the impact of their internal parameters on the efficiency of algorithm performance was examined, their main advantages and disadvantages were identified, similar software products were analyzed, and software was developed to implement the specified methods.

This work consists of three sections. Each of them is devoted to: theoretical foundations of CRM system modeling ; algorithms and methods for implementing the application ; Modeling and designing a CRM system with logistics analysis methods and analyzing the obtained results..

The overall scope of the work is 85 pages. Thesis contains 2 applicationi, 29 figures, 2 tables and 50 references in it.

Key words: inventory management, CRM system, asynchronous programming, databases, framework, XYZ method, ABC method.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

CRM - (англ. customer relationship management) Управлі́ння віднóсинами з клієнтами
ABC - метод, який дозволяє класифікувати бізнес-ресурси
XYZ - аналіз, що дозволяє зробити класифікацію ресурсів
VED - директивна сегментація необхідності асортименту
FMR - (англ. Fast moving, Slow moving, Non Moving) аналіз товарного асортименту за частотою застосування/взяття
HML - аналіз заснований на співвідношенні ціна/вага об'єкта за одиницю

ВСТУП

У сучасних умовах ведення бізнесу ефективне управління взаємовідносинами з клієнтами та оптимізація логістичних процесів є одним із ключових факторів успіху підприємств. Конкурентне середовище вимагає від компаній максимальної гнучкості та раціональності в управлінні як клієнтськими даними, так і внутрішніми операціями. Одним із найважливіших інструментів, що дозволяє досягти цих цілей, є CRM система, яка спрямована на управління взаєминами з клієнтами та покращення комунікацій на всіх рівнях бізнес–процесів.

Впровадження логістичних методів у роботу CRM системи дозволяє не тільки покращити взаємодію з клієнтами, але й оптимізувати постачання, управління запасами та розподіл ресурсів. Зокрема, використання методів закупівельної логістики та аналітичних підходів, таких як ABC та XYZ класифікація, дозволяє ефективно управляти ресурсами та скорочувати витрати, розподіляючи товари та послуги за їх значимістю для бізнесу та рівнем передбачуваності споживання.

Метою даної дипломної роботи є розробка CRM системи з використанням методів закупівельної логістики та впровадження ABC і XYZ класифікаційних методів на мові програмування C#. В рамках роботи будуть розглянуті основи побудови CRM систем, методи аналізу та класифікації товарів і послуг, а також алгоритми, що дозволяють автоматизувати процес управління закупівлями та запасами на підприємстві.

Актуальність даного дослідження полягає в необхідності інтеграції логістичних підходів до CRM для підвищення ефективності управлінських рішень та вдосконалення процесів постачання. Використання мови програмування C# дозволяє реалізувати гнучкі та масштабовані рішення, які можуть бути адаптовані під конкретні потреби підприємства.

1 ТЕОРЕТИЧНІ ЗАСАДИ МОДЕЛЮВАННЯ CRM СИСТЕМИ

Комп'ютерний додаток є ключовим елементом сучасного програмного забезпечення. Це набір даних і команд, які створюють основу для взаємодії користувача з комп’ютером. Застосунки виконують широкий спектр завдань – від забезпечення базової роботи пристрою до створення складних професійних рішень.

Кожен додаток має своє конкретне призначення. Наприклад, деякі з них допомагають користувачеві створювати графіку, писати текст чи навіть складати музику. Інші дозволяють впорядковувати файли на жорсткому диску або працювати в Інтернеті. Таке різноманіття програм дозволяє користувачам обирати інструменти, що найкраще відповідають їхнім потребам.

Однією з основних категорій є системні програми. Вони є невидимим фундаментом, що забезпечує функціонування всієї комп’ютерної системи. Наприклад, операційна система – це серце комп’ютера, що з'єднує апаратне забезпечення з програмним. Без неї жоден комп’ютер не зміг би виконати навіть найпростішу команду. До системного програмного забезпечення також відносяться утиліти – невеликі, але надзвичайно корисні програми, які допомагають автоматизувати та оптимізувати роботу комп’ютера. Завдяки утилітам користувач отримує доступ до параметрів, що інакше залишилися б недосяжними.

Інша велика група – прикладні програми, які створені безпосередньо для кінцевого користувача. Саме ці програми забезпечують основну взаємодію людини з комп’ютером. Наприклад, офісні програми стали майже синонімом з повсякденною роботою на комп'ютері. Завдяки текстовим редакторам, електронним таблицям і засобам для створення презентацій користувачі можуть швидко створювати та редагувати документи. Особливу популярність здобув офісний пакет Microsoft Office, який включає такі інструменти, як Word, Excel і PowerPoint, кожен із яких став невід'ємною частиною роботи в багатьох сферах.

Окрім того, існують програми, що допомагають у фінансових обчисленнях. Серед них особливо виділяється 1С: Бухгалтерія – комплекс, який спрощує ведення як робочої, так і домашньої бухгалтерії. В останні роки такі програми стають усе більш популярними навіть серед звичайних користувачів, адже автоматизація фінансів дозволяє заощаджувати час і уникати помилок[1-3].

Окрему групу складають професійні програми, які зазвичай орієнтовані на вузькі спеціалізовані завдання. Наприклад, інструменти для програмування використовуються розробниками для створення нових додатків, тоді як системи проектування, такі як AutoCAD, необхідні інженерам для креслення складних конструкцій.

Не можна обійти увагою і розважальні програми, які приваблюють мільйони користувачів. Це можуть бути відеоігри, що дозволяють зануритися в захопливі віртуальні світи, або освітні програми, які допомагають здобувати нові знання. Мультимедійні програми також є важливою частиною сучасного життя. Вони дають змогу створювати та обробляти відео, аудіо та графіку, відкриваючи перед користувачами величезні творчі можливості.

Усі ці категорії додатків мають спільну мету – зробити роботу з комп’ютером зручнішою, продуктивнішою та цікавішою. Сучасний світ неможливо уявити без цих програм, адже вони проникають у всі сфери людської діяльності, сприяючи її розвитку та вдосконаленню.

1.1 Основні положення використовування CRM систем у суспільстві

Історія управління взаємовідносинами з клієнтами розпочалася задовго до появи комп'ютерних технологій. У часи, коли бізнес–комунікації здійснювалися переважно особисто або через листування, підприємці вели докладні записи про своїх клієнтів у товстих шкіряних реєстрах, записних книжках та картотеках.

Кожен запис був по суті маленьким літописом взаємин: коли клієнт востаннє купував товар, які були його побажання, що подобалося, а що викликало нарікання.

Власники невеликих магазинів та сімейних підприємств знали практично кожного свого клієнта поіменно. Вони пам'ятали звички, сімейний стан, професію. Така персоналізація була природним способом побудови лояльності задовго до того, як з'явилися маркетингові стратегії та терміни на кшталт "клієнтський досвід".

З появою перших електронно–обчислювальних машин розпочалася принципово нова ера обліку та систематизації інформації. Великі корпорації першими усвідомили потенціал комп'ютерів для зберігання та опрацювання величезних масивів даних. Спочатку це були примітивні бази даних, де інформація про клієнтів зберігалася у вигляді простих таблиць: прізвище, контактний телефон, дата останньої покупки.

Справжня революція розпочалася наприкінці 1980–х років, коли з'явилися перші спеціалізовані програмні продукти для управління контактами. Програма ACT!, розроблена компанією Conductor Software, стала справжнім проривом. Вона дозволяла не просто зберігати контактні дані, але й робити нотатки, планувати зустрічі, відстежувати історію комунікацій.

Цей період можна охарактеризувати як перехід від звичайного обліку до усвідомленого управління взаємовідносинами. Бізнес почав розуміти, що клієнт – це не просто запис у базі даних, а жива людина з унікальними потребами та очікуваннями.

CRM (Customer Relationship Management) – це система управління взаємовідносинами з клієнтами, яка є комплексом стратегій, процесів і програмних рішень, що допомагають компаніям організовувати та підтримувати ефективні стосунки з клієнтами. CRM системи дозволяють бізнесам збирати, зберігати й аналізувати дані про клієнтів, що допомагає краще розуміти їхні потреби, підвищувати рівень обслуговування, а також сприяти збільшенню продажів.

Основні функції CRM включають:

- збір та управління інформацією про клієнтів;
- автоматизацію продажів і маркетингу;
- оптимізацію процесів взаємодії з клієнтами (контакт–центри, служби підтримки);
- аналітику і сегментацію клієнтської бази для покращення прийняття рішень.

CRM як концепція виникла у 1980–х роках з метою підвищення якості обслуговування клієнтів та збільшення рівня їхньої лояльності. Спочатку це були прості інструменти для управління контактами та інформацією про клієнтів. Пізніше, у 1990–х роках, CRM системи еволюціонували в більш інтегровані рішення, що охоплюють не лише зберігання даних, але й аналітичні інструменти для прогнозування поведінки клієнтів.

З розвитком інтернету та мережевих технологій CRM–системи почали стрімко трансформуватися. З'явилися перші web–орієнтовані рішення, які дозволяли працювати з клієнтською базою віддалено, без прив'язки до конкретного комп'ютера.

Ключовим моментом став запуск Salesforce.com у 1999 році. Ця платформа фактично створила новий ринок – cloud CRM. Тепер компанії могли орендувати потужний інструмент управління клієнтськими взаємодіями без значних первинних інвестицій у технічну інфраструктуру.

Сучасні CRM системи пропонують інтеграцію з іншими бізнес–додатками, такими як ERP (Enterprise Resource Planning), системи управління постачаннями, обліку та маркетингової автоматизації.

Поділяють три види операційних систем , такі як операційна, аналітична та колаборативна.

Операційна CRM система зосереджується на автоматизації бізнес–процесів, пов'язаних із взаємодією з клієнтами: продажі, маркетинг та підтримка клієнтів.

Основна мета операційної CRM – підвищити ефективність і продуктивність взаємодії з клієнтами.

Основні функції операційної CRM включають:

- управління контактами (contact management);
- управління взаємовідносинами з клієнтами (customer relationship management);
- автоматизація процесів продажу (sales force automation);
- обслуговування клієнтів і підтримка (customer service).

Аналітична CRM орієнтована на збір і аналіз даних про клієнтів з метою оптимізації маркетингових та продажних стратегій. Ця частина системи забезпечує бізнес інформацією для прийняття рішень щодо оптимізації взаємодії з клієнтами.

До основних завдань аналітичної CRM належать:

- сегментація клієнтської бази;
- прогнозування попиту і поведінки клієнтів;
- аналіз ефективності маркетингових кампаній.

Колаборативна CRM спрямована на покращення комунікацій між різними відділами компанії для забезпечення єдиного підходу до клієнта. Вона дозволяє інтегрувати різні канали взаємодії з клієнтами, такі як електронна пошта, телефони, соціальні мережі, що підвищує ефективність обслуговування[4-6].

CRM системи мають значний вплив на підвищення продуктивності та ефективності роботи бізнесу, зокрема:

- покращення взаємовідносин з клієнтами: забезпечують персоналізований підхід до кожного клієнта, що підвищує їх лояльність;
- автоматизація бізнес–процесів: CRM системи автоматизують рутинні завдання, звільняючи час для стратегічних питань;
- підвищення ефективності продажів і маркетингу: надають інструменти для аналізу поведінки клієнтів, що дозволяє зосередитися на найбільш прибуткових сегментах;

– оптимізація комунікацій: CRM об'єднує різні канали комунікацій, дозволяючи відстежувати всю історію взаємодії з клієнтами в одному місці.

CRM системи відіграють важливу роль у логістичній діяльності компаній, забезпечуючи інтеграцію інформаційних потоків щодо клієнтських запитів та замовлень із процесами закупівлі, управління запасами та постачання. Логістична CRM дозволяє автоматизувати процеси управління ланцюгами постачань, координувати взаємодію з постачальниками, контролювати ефективність логістичних операцій.

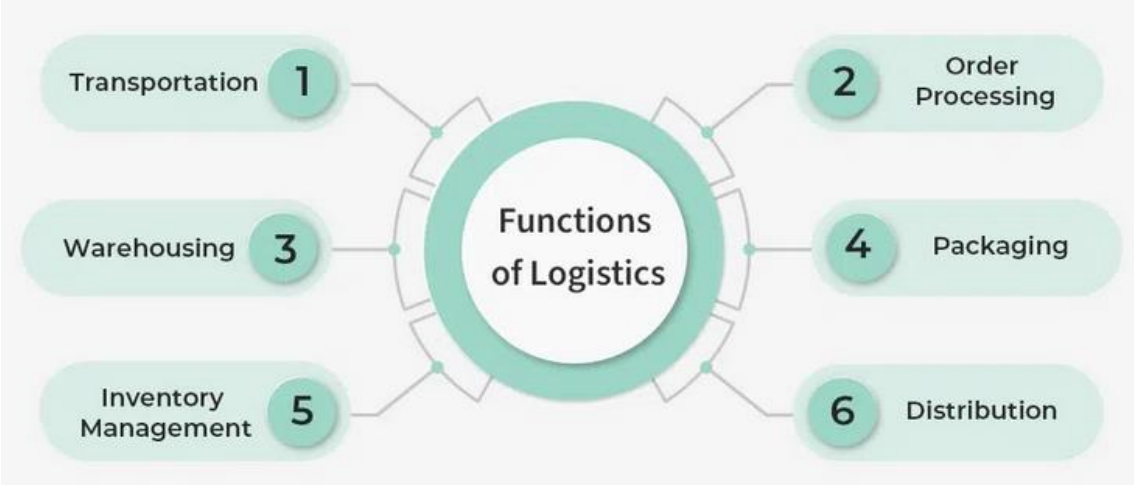

Рисунок 1.1 – Зони впливу логістики

Логістичний аналіз охоплює широкий спектр методів, які допомагають оптимізувати управління запасами, транспортуванням і загальною організацією логістичних процесів. Окрім ABC та XYZ аналізів, існують такі підходи, як VED, FSN, HML, SDE та інші методи, кожен з яких має свої особливості. Наприклад, деякі з них зосереджені на критичності товарів (VED), частоті використання (FSN) або вартості (HML). Ці методи дозволяють аналізувати певні аспекти запасів, які можуть бути важливими в залежності від характеру бізнесу.

Однак ABC і XYZ аналізи займають особливе місце через свою універсальність і інтегративність. ABC–аналіз дозволяє зосередитися на фінансовій складовій, виділяючи товари, які найбільше впливають на дохід компанії. XYZ–

аналіз, у свою чергу, акцентує увагу на стабільності та передбачуваності попиту. Разом вони створюють збалансований підхід, який враховує як прибутковість, так і поведінку товарів на ринку. Це дає змогу підприємствам не лише ідентифікувати найважливіші товари, а й вибудувати ефективні стратегії управління запасами.

На відміну від деяких інших методів, які можуть зосереджуватися лише на одному аспекті (наприклад, критичності або вартості), ABC і XYZ дозволяють отримати комплексне уявлення про товари. Вони легко адаптуються до різних типів бізнесу, оскільки їхні принципи базуються на універсальних показниках – фінансовій вигоді та поведінці попиту. Крім того, їх використання є відносно простим у реалізації навіть для компаній, які лише починають впроваджувати аналітичні підходи до управління логістикою.

Перевага ABC і XYZ також у тому, що вони можуть бути інтегровані у сучасні інформаційні системи, включаючи CRM. Завдяки цьому дані для аналізу легко збираються і обробляються автоматично, що дозволяє уникнути ручної роботи та зменшити ризик помилок. У результаті компанії можуть швидко отримувати точні результати, на основі яких ухвалюються управлінські рішення.

Таким чином, хоча інші методи логістичного аналізу мають свої сильні сторони, ABC та XYZ вирізняються своєю простотою, адаптивністю та здатністю працювати разом, забезпечуючи комплексний підхід до управління запасами. Вони допомагають підприємствам не тільки зосереджуватися на найважливіших товарних позиціях, але й ефективно реагувати на зміну попиту, що робить їх ключовими інструментами у сучасній логістиці.

Використання CRM в логістиці дозволяє:

- скоротити час виконання замовлень;
- оптимізувати запаси та постачання;
- підвищити задоволеність клієнтів через прозорість і швидкість процесів.

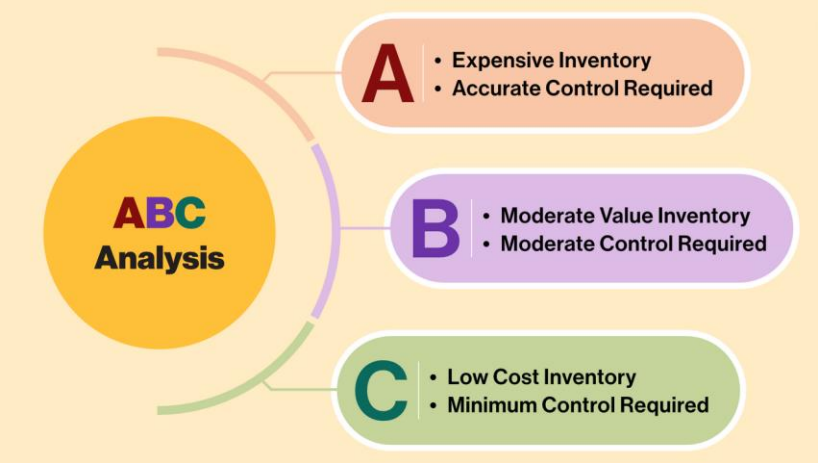

Рисунок 1.2 – Структура ABC аналізу

ABC аналіз – це метод управління запасами, який класифікує товари за їх значущістю для бізнесу. Категорія A – це найважливіші товари, які мають найбільший вплив на прибуток, тоді як категорії B і C представляють товари середньої та низької важливості відповідно.

	A	B	C
X	High Value Fixed Demand Highly Predictable	Average Value Fixed Demand Highly Predictable	Low Value Fixed Demand Highly Predictable
Y	High Value Fluctuating Demand Moderately Predictable	Average Value Fluctuating Demand Moderately Predictable	Low Value Fluctuating Demand Highly Predictable
Z	High Value Uncertain Demand Unpredictable	Average Value Uncertain Demand Unpredictable	Low Value Uncertain Demand Unpredictable

Рисунок 1.3 – Таблиця відповідності XYZ до ABC аналізу

XYZ аналіз – це метод прогнозування, який класифікує товари за рівнем стабільності попиту. Товари групи X мають стабільний і передбачуваний попит, група Y характеризується певними коливаннями, а група Z – непостійним і складно передбачуваним попитом.

Інтеграція ABC і XYZ аналізу в CRM систему дозволяє:

– оптимізувати управління запасами, концентруючи увагу на найважливіших і стабільних товарах;

– знизити витрати на утримання запасів через більш точне прогнозування попиту;

– підвищити ефективність закупівельної логістики, надаючи пріоритет товарам з найбільшим впливом на бізнес.

З розвитком цифрових технологій CRM системи також еволюціонують, інтегруючи нові інструменти для підвищення ефективності:

– штучний інтелект (AI): автоматизація процесів, персоналізація взаємодії з клієнтами, передбачення потреб клієнтів;

– машинне навчання: аналітика великих даних для сегментації клієнтів і покращення точності прогнозів;

– хмарні рішення (Cloud CRM): доступ до даних з будь–якого місця, що забезпечує гнучкість і масштабованість систем;

– мобільні CRM: забезпечують доступ до CRM через мобільні пристрої, що робить процеси більш оперативними і гнучкими.

Сучасні CRM–системи – це вже не просто бази даних, а складні інтелектуальні екосистеми. Вони використовують machine learning для прогнозування поведінки клієнтів, автоматизації маркетингових кампаній, персоналізації комунікацій.

Штучний інтелект дозволяє миттєво аналізувати величезні масиви інформації, виявляти приховані закономірності, передбачати потреби клієнтів навіть раніше, ніж вони їх усвідомлять. Уявіть собі систему, яка не просто зберігає інформацію про попередні покупки, але й може запропонувати найбільш релевантний продукт у конкретний момент часу.

Майбутнє CRM–систем пов'язане з подальшою інтеграцією штучного інтелекту, розширенням можливостей передбачувальної аналітики та створенням максимально персоналізованого клієнтського досвіду.

Вже сьогодні CRM виходять за межі суто продажів, перетворюючись на стратегічний інструмент управління бізнесом. Вони дозволяють не просто

фіксувати взаємодії, але й вибудовувати довгострокові, емоційно забарвлені відносини з клієнтами.

Технології змінюються, але основна мета залишається незмінною – зробити взаємодію між компанією та клієнтом максимально комфортною, передбачуваною та ефективною.

CRM системи є невід’ємною частиною сучасного бізнесу, що сприяє підвищенню ефективності взаємодії з клієнтами, автоматизації процесів і прийняттю обґрунтованих управлінських рішень. Інтеграція логістичних методів, таких як ABC і XYZ аналізи, значно підвищує ефективність управління запасами та закупівлями. У поєднанні з використанням сучасних технологій CRM системи дозволяють компаніям залишатися конкурентоспроможними на ринку та досягати стратегічних цілей.

1.2 Мова програмування C# та WinForms.

Програмування – це процес створення програм, які забезпечують автоматизацію, обробку даних і виконання складних завдань комп'ютером. Воно є невід'ємною частиною розвитку сучасних технологій, охоплюючи широкий спектр напрямків: від простих скриптів для виконання рутинних завдань до розробки складних програмних систем.

Основна мета програмування полягає у створенні алгоритмів, що перетворюють вхідні дані у бажаний результат, при цьому програма повинна бути ефективною, зрозумілою для інших розробників і легко масштабованою. Для цього програміст використовує мову програмування – формальний спосіб запису інструкцій, які комп'ютер може виконати. Високорівневі мови програмування, такі як C#, надають простіший синтаксис, абстрагуючи складні апаратні деталі, що дозволяє розробникам зосередитись на логіці і функціональності програми.

C# – це сучасна мова програмування, створена корпорацією Microsoft як частина платформи .NET. Вона була розроблена для створення надійних, безпечних і продуктивних програм у різних галузях, включаючи розробку настільних додатків, веб–застосунків, мобільних програм, ігор та інших програмних продуктів.

Особливістю C# є її багатопарадигмальність, що дозволяє використовувати різні підходи до розробки, такі як об'єктно–орієнтоване програмування (ООП), функціональне програмування та подійно–орієнтоване програмування. C# також відомий своєю суворою типізацією, яка допомагає уникнути помилок під час компіляції та виконання програм.

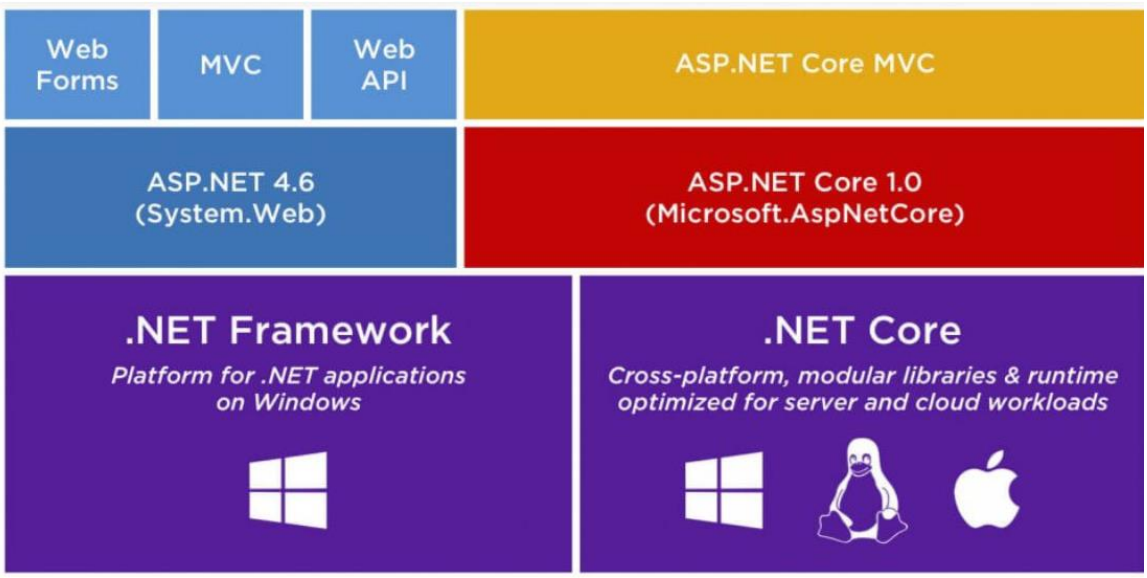

Рисунок 1.4 – Структура платформи .Net

Інтеграція з платформою .NET надає мові доступ до багатої бібліотеки класів, які забезпечують широкий спектр функціональностей, від роботи з файлами та мережею до створення графічного інтерфейсу користувача і забезпечення безпеки даних. Завдяки цьому C# є універсальним інструментом для створення програм будь–якої складності.

Однією з ключових характеристик C# є підтримка об'єктно–орієнтованого програмування, яке є основою для створення модульного, повторно використовуваного та легко масштабованого коду. ООП базується на кількох принципах:

- класи і об'єкти. Клас виступає як шаблон, який описує властивості та поведінку об'єктів. Об'єкт – це конкретна реалізація класу, яка має певний стан і виконує визначені дії.
- інкапсуляція – забезпечує приховування внутрішніх деталей реалізації класу, надаючи доступ до даних і методів лише через визначений інтерфейс. Це зменшує ризик помилок і дозволяє змінювати внутрішню логіку класу без впливу на зовнішній код.
- спадкування – дозволяє створювати нові класи на основі існуючих, успадковуючи їхні властивості й методи. Це сприяє повторному використанню коду та спрощує створення складних ієрархій об'єктів.
- поліморфізм – дає змогу об'єктам виконувати різні дії залежно від їхнього типу або реалізації методу. Це досягається за допомогою перевантаження методів або їх перевизначення у похідних класах.

C# забезпечує сувору типізацію, яка гарантує, що кожна змінна, метод чи параметр має чітко визначений тип. Це дозволяє уникнути багатьох поширених помилок, таких як спроба виконання операцій над несумісними типами, і забезпечує кращу передбачуваність роботи програми.

C# використовує механізм збору сміття (garbage collection), який автоматично видаляє непотрібні об’єкти з пам’яті. Це дозволяє програмісту зосередитися на логіці програми, не турбуючись про управління пам’яттю вручну.

C# підтримує як прості (примітивні) типи даних, такі як числа, символи та логічні значення, так і складні типи, включаючи масиви, списки, словники та класи. Бібліотека LINQ (Language–Integrated Query) дозволяє виконувати запити до даних

у декларативному стилі, незалежно від їхнього джерела – баз даних, масивів або колекцій[7-9].

Мова C# розроблена з урахуванням безпеки коду. Вона включає механізми перевірки типів, контроль доступу до пам’яті та засоби обробки винятків, що дозволяє створювати надійні та захищені програми.

Мова програмування C# та технологія WinForms є важливими складовими платформи .NET, яка широко використовується для розробки різноманітного програмного забезпечення, зокрема настільних додатків. Незважаючи на стрімкий розвиток веб– та мобільних додатків, настільні рішення досі залишаються актуальними у багатьох сферах, таких як корпоративні системи, фінансові додатки, спеціалізовані інструменти для професіоналів тощо. У цьому рефераті розглянуто мову програмування C#, її особливості та переваги, а також технологію Windows Forms, яка використовується для створення графічного інтерфейсу (GUI) у Windows–додатках.

C# була створена компанією Microsoft у 2000 році під керівництвом Андерса Хейлсберга. Вона є частиною платформи .NET і була розроблена з метою створення сучасної, потужної та простої у використанні мови програмування. C# виникла на базі таких мов, як C++ та Java, і поєднує в собі найкращі характеристики обох, а також впроваджує нові, інноваційні рішення.

Спочатку C# була спрямована на розробку програм для Windows, але з розвитком платформи .NET Core (а потім і .NET 5+) C# стала кросплатформенною мовою, що дозволяє створювати програми для Linux, macOS та інших операційних систем.

Основні принципи та особливості мови C#:

– об'єктно–орієнтоване програмування (ООП): C# базується на принципах ООП, що робить її потужним інструментом для розробки складних програм. Вона підтримує такі концепції, як інкапсуляція, успадкування, поліморфізм та

абстракція. Завдяки цим концепціям програмний код є більш зрозумілим, структурованим і легшим для підтримки;

– безпека типів: C# є строго типізованою мовою, що означає, що всі змінні та об'єкти мають чітко визначені типи. Це дозволяє виявляти помилки на етапі компіляції, що підвищує надійність програм та знижує кількість помилок під час виконання;

– автоматичне керування пам'яттю: одна з ключових особливостей C# – це автоматична система керування пам'яттю через механізм збору сміття (Garbage Collection). Завдяки цьому програмісти не повинні вручну вивільняти пам'ять, як це робиться у C++, що значно спрощує розробку та знижує ризик виникнення витоків пам'яті;

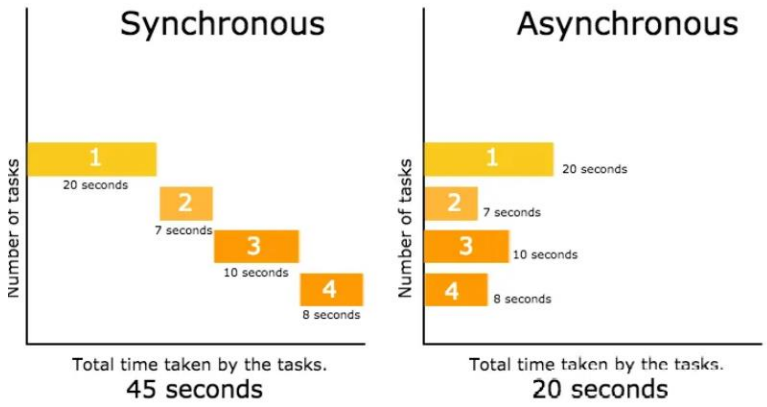

Рисунок 1.5 – Порівняльна діаграма асинхронного програмування

– асинхронне програмування: Завдяки використанню ключових слів async та await, C# підтримує асинхронне програмування, яке дозволяє виконувати операції паралельно або у фоновому режимі без блокування основного потоку. Це особливо важливо для додатків, які працюють з мережею або великими обсягами даних;

– LINQ (Language Integrated Query): C# інтегрує мову запитів безпосередньо в саму мову програмування. Це дозволяє працювати з колекціями

об'єктів, базами даних та іншими джерелами даних за допомогою потужного синтаксису LINQ;

– кросплатформенність: спочатку C# використовувалася тільки для розробки Windows–додатків, але після виходу .NET Core та .NET 5+ стала можливою розробка додатків для різних операційних систем, включаючи Linux і macOS. Це розширило можливості використання C# у сучасному світі програмування.

Windows Forms (WinForms) – це бібліотека для розробки графічного інтерфейсу (GUI) в додатках на платформі Windows. Вона є частиною платформи .NET і використовується для створення настільних додатків із використанням візуальних елементів. WinForms залишається популярним вибором для розробників, які працюють у середовищі Windows, незважаючи на наявність новіших технологій, таких як WPF (Windows Presentation Foundation) і UWP (Universal Windows Platform)[10].

WinForms була вперше випущена у 2002 році як частина першої версії .NET Framework. Її основним завданням було забезпечення швидкого та простого способу створення графічних інтерфейсів для настільних додатків. На той час вона стала заміною для старих технологій на зразок MFC (Microsoft Foundation Classes) та API Windows, які вимагали глибокого знання низькорівневих функцій Windows і були складними у використанні.

У WinForms акцент був зроблений на спрощенні процесу розробки за рахунок надання готових інструментів для роботи з GUI, таких як форми, кнопки, текстові поля, вкладки тощо[11].

У WinForms основним елементом програми є форма – це вікно, яке користувач бачить на екрані. Форми можуть містити різноманітні елементи керування (контролі), такі як:

– кнопки (Button);
– текстові поля (TextBox);

- мітки (Label);
- списки (ListBox);
- меню (MenuStrip);
- таблиці (DataGridView).

Кожен елемент керування може реагувати на події (наприклад, натискання кнопки або введення тексту), що дозволяє програмісту визначати відповідну поведінку програми через обробники подій.

WinForms ґрунтується на подієво–орієнтованій моделі програмування. Це означає, що програма реагує на різні події, такі як дії користувача або системні події, й виконує відповідні дії. Наприклад, подія Click для кнопки визначає, що відбуватиметься після натискання на кнопку.

Однією з найбільших переваг WinForms є інтеграція з IDE Visual Studio, яка надає візуальний редактор для створення інтерфейсу. Розробник може просто перетягувати елементи керування на форму, налаштовувати їхні властивості у спеціальному вікні та додавати події без написання великої кількості коду.

Це дозволяє значно прискорити процес розробки, оскільки більшість візуальних компонентів можна налаштувати за допомогою інтуїтивно зрозумілого інтерфейсу.

Форма – це основне вікно додатка. Вона може бути головним вікном або діалоговим вікном (наприклад, вікно повідомлень або налаштувань). Кожна форма має набір властивостей, таких як розміри, позиція, назва (текст у заголовку), кольори та інші параметри[12-13].

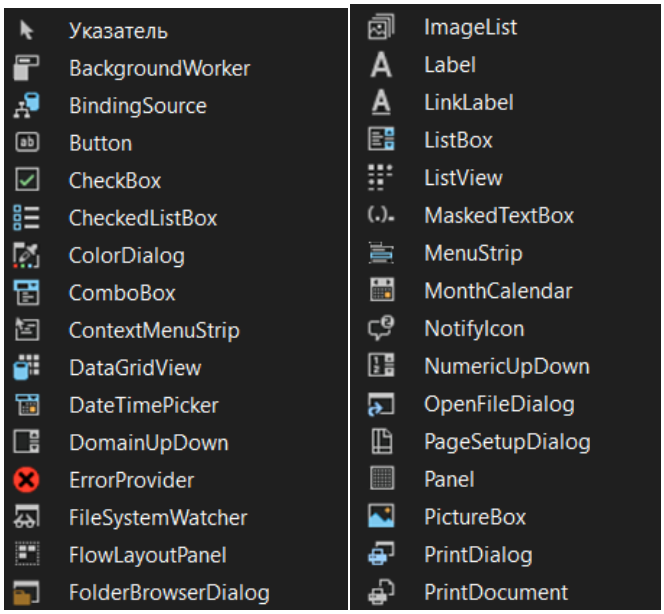

Рисунок 1.6 – Елементи керування у WinForms

WinForms пропонує великий набір стандартних елементів керування, які можна використовувати для взаємодії користувача з додатком. Ось кілька основних:

- **button (Кнопка)**: використовується для виконання певної дії під час натискання;
- **textBox (Текстове поле)**: дозволяє користувачу вводити текст;
- **label (Мітка)**: статичний текст або пояснення для інших елементів керування;
- **comboBox (Випадаючий список)**: поєднання поля введення та списку можливих варіантів;
- **listBox (Список)**: відображає список елементів, з яких користувач може вибрати один або кілька;
- **dataGridView (Таблиця)**: дозволяє відображати та редагувати табличні дані.

Кожен елемент керування може мати прив'язані події, на які він реагує. Наприклад, подія Click для кнопки, TextChanged для текстового поля або

SelectedIndexChanged для списку. Використання подій робить програму інтерактивною та дозволяє обробляти введення користувача[14-17].

Програміст визначає, що саме повинно відбуватися під час виконання певної події. Це може бути як обчислення, так і взаємодія з базами даних або відображення нових форм.

Крім стандартних елементів керування, WinForms дозволяє створювати власні користувацькі елементи або розширювати функціональність існуючих. Це надає додатку унікальний вигляд і дозволяє реалізувати специфічні сценарії роботи, яких може не вистачати у стандартному наборі. Наприклад, можна створити власний компонент для відображення графіків, динамічних звітів або інтерактивних панелей.

Одна з переваг WinForms – це можливість налаштовувати властивості елементів керування безпосередньо через інтерфейс Visual Studio. Панель «Властивості» дозволяє змінювати зовнішній вигляд і поведінку елементів, наприклад, змінювати кольори, розміри, шрифти або додавати інструментальні підказки. Також у програмному коді можна змінювати ці властивості динамічно, залежно від дій користувача або стану програми.

Наприклад, один із популярних сценаріїв – взаємодія з базою даних через DataGridView. Цей компонент дозволяє відображати дані у табличному форматі, надаючи користувачу можливість редагувати записи, сортувати та фільтрувати їх. Через інтеграцію з ADO.NET або Entity Framework дані можуть автоматично завантажуватися в таблицю з бази даних. При цьому програміст може визначати, які стовпці відображати, як форматувати дані, та обробляти події, пов’язані із введенням чи вибором користувача.

Подібним чином обробляються форми введення даних, де елементи, такі як TextBox або ComboBox, дозволяють користувачу вводити інформацію, яка надалі зберігається в базі даних. Поля можуть бути пов’язані з перевіркою введення, наприклад, для забезпечення коректного формату даних (таких як електронна

пошта чи номер телефону). Подія Validating, доступна для багатьох елементів керування, дозволяє реалізувати ці перевірки безпосередньо на рівні інтерфейсу.

Ще однією важливою особливістю WinForms є можливість роботи з кількома формами. Наприклад, головне вікно додатку може бути порталом, через який відкриваються дочірні форми. Кожна така форма відповідає за певний функціонал – управління клієнтами, перегляд звітів чи виконання аналітичних розрахунків. Для переходу між формами використовуються методи Show() або ShowDialog(), залежно від того, чи повинна основна форма залишатися активною під час роботи з дочірньою.[18]

WinForms також підтримує обробку глобальних подій, таких як натискання клавіш або рух миші. Це дозволяє створювати інтуїтивно зрозумілі інтерфейси, наприклад, гарячі клавіші для виконання певних функцій або динамічні меню, які змінюються залежно від поточного контексту.

Що важливо, WinForms інтегрується з іншими технологіями .NET, такими як WCF або ASP.NET, що дозволяє створювати розподілені додатки з доступом до серверів або веб–служб. Це робить WinForms потужним інструментом для створення складних настільних програм, які можуть працювати як автономно, так і в складі більшого програмного середовища.

Таким чином, WinForms пропонує розробнику широкий набір інструментів для створення адаптивного, функціонального і зручного для користувача інтерфейсу, з можливістю масштабування додатка під конкретні бізнес–завдання.

Висновки до розділу 1

У першому розділі було розглянуто основи побудови CRM–систем, їхні види, а також ключові інструменти для їхньої реалізації, зокрема мову програмування C# та технологію Windows Forms. Вивчення теоретичних аспектів

дозволило сформувати цілісне уявлення про роль CRM–систем у сучасному бізнесі, їх функціональність та підходи до реалізації.

Мова програмування C# та технологія Windows Forms є оптимальними інструментами для розробки настільних CRM–додатків завдяки їхній гнучкості, надійності та широким можливостям інтеграції. C# забезпечує розробникам доступ до потужного середовища .NET, що дозволяє легко створювати функціональні додатки з високим рівнем продуктивності. Технологія Windows Forms, у свою чергу, забезпечує створення зручного графічного інтерфейсу користувача, який є важливою складовою для ефективного використання CRM–системи[19-20].

Таким чином, у межах теоретичних засад моделювання CRM–систем були визначені ключові принципи та технології, які формують основу для подальшої розробки програмного продукту. Отримані знання стали базисом для побудови ефективної CRM–системи, яка здатна задовольнити потреби користувачів і забезпечити зручність у використанні.

2 СТВОРЕННЯ АЛГОРИТМІВ ТА МЕТОДІВ РЕАЛІЗАЦІЇ ДОДАТКУ

У попередньому розділі було визначено теоретичні засади CRM системи, яка інтегрує методи закупівельної логістики та алгоритми ABC і XYZ аналізу. Наступним етапом є розробка алгоритмів та методів, які забезпечать реалізацію основних функцій системи на практиці, а також формування структури програмного додатку мовою програмування C#.

Розробка алгоритмів і методів є ключовим етапом, оскільки саме вони визначають функціональність та ефективність роботи CRM системи. Основна задача – створення алгоритмів, які дозволять автоматизувати управління клієнтськими даними, оптимізувати закупівлі та запаси, і проводити аналіз за методами ABC і XYZ. Це включає побудову логіки роботи додатку, яка передбачає обробку великих обсягів даних, їх систематизацію та виконання комплексних операцій на базі інтегрованих моделей.

Мова програмування C# була обрана для реалізації додатку завдяки її можливостям створювати продуктивні, масштабовані та легко підтримувані програми. Вибір C# також обумовлений широким використанням платформи .NET, яка забезпечує необхідну інфраструктуру для реалізації сучасних бізнес–рішень. У цьому розділі ми розглянемо основні етапи створення алгоритмів, включаючи структуру даних, логіку реалізації функцій, а також інтеграцію методів аналізу та закупівельної логістики у єдиній системі.

2.1 Огляд існуючих програмних рішень

На сучасному ринку представлені численні рішення для управління взаємовідносинами з клієнтами (CRM системи), які допомагають бізнесам автоматизувати і вдосконалити процеси комунікації з клієнтами, управління продажами, маркетингом і обслуговуванням клієнтів. Різні CRM платформи

пропонують індивідуальні функції та можливості, орієнтовані на певні сегменти ринку, що дозволяє підприємствам обирати ті рішення, які найбільше відповідають їхнім потребам. У цьому огляді ми розглянемо найпопулярніші CRM системи на ринку, їхні можливості, особливості та технологічні аспекти.

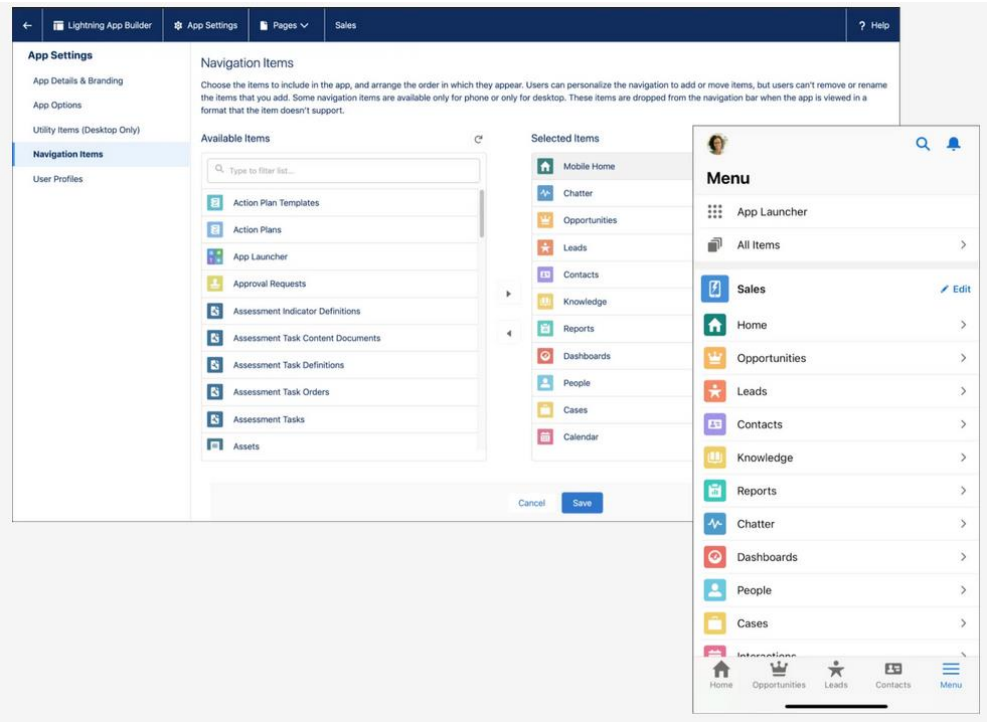

Рисунок 2.1– Salesforce

Salesforce – одна з найвідоміших та найпопулярніших CRM систем у світі, що пропонує хмарне рішення для управління взаємовідносинами з клієнтами. Вона орієнтована на великі підприємства, але також має пропозиції для середнього та малого бізнесу. Основні переваги Salesforce включають:

– широкий набір функцій: управління продажами, маркетингом, підтримкою клієнтів, аналітика і автоматизація бізнес–процесів;

– інтеграції: Salesforce легко інтегрується з іншими популярними інструментами та додатками, такими як ERP системи, сервіси електронної пошти і маркетингу;

– масштабованість: платформа може рости разом із компанією, дозволяючи додавати нові функції та модулі залежно від потреб бізнесу;

– AI технології: інтеграція штучного інтелекту через Salesforce Einstein для аналізу даних і прогнозування поведінки клієнтів.

Недоліки Salesforce:

– висока вартість підписки, що може бути значною для малого бізнесу.

– складність налаштування, що потребує наявності кваліфікованих спеціалістів або додаткових ресурсів на впровадження.

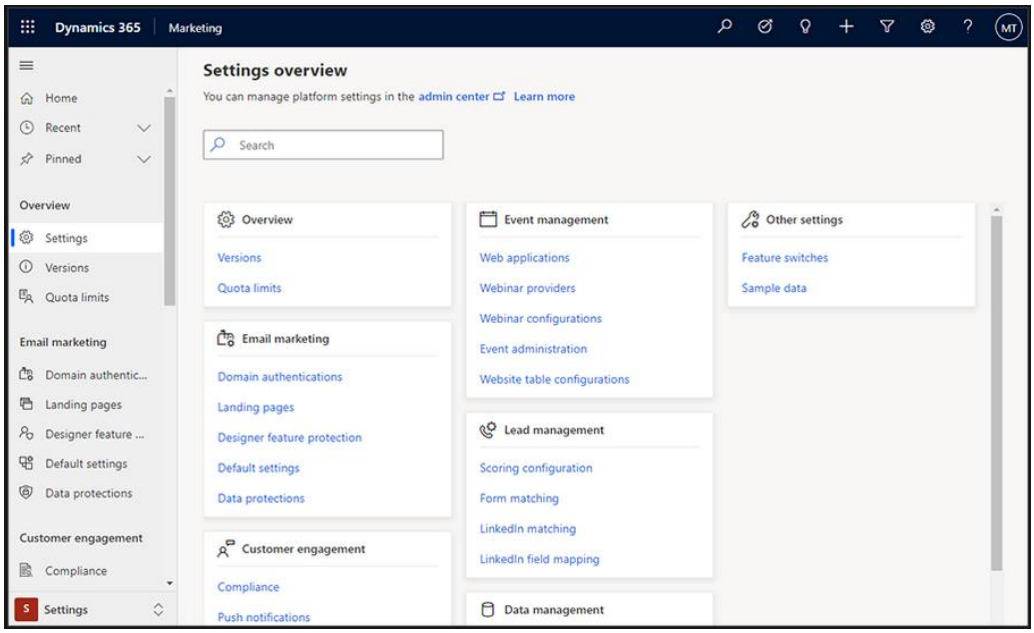

Рисунок 2.2 – Microsoft Dynamics 365

Microsoft Dynamics 365 – це комплексне рішення, яке поєднує можливості CRM та ERP систем. Dynamics 365 пропонує повноцінний набір інструментів для управління клієнтами, автоматизації продажів і маркетингу, а також інтеграцію з іншими продуктами Microsoft, такими як Office 365 та Azure[21-25].

Основні переваги Microsoft Dynamics 365:

– інтеграція з іншими продуктами Microsoft: глибока інтеграція з пакетами Office, що робить роботу з клієнтами більш ефективною;

– аналітика на базі AI: платформа використовує штучний інтелект для прогнозування продажів і надання інсайтів щодо клієнтів;

– модульність: можливість додавати різні модулі для продажів, обслуговування клієнтів, управління проєктами і фінансами.

Недоліки Microsoft Dynamics 365:

– висока вартість для малих підприємств;

– потреба в налаштуванні та навчанні персоналу, що збільшує час впровадження.

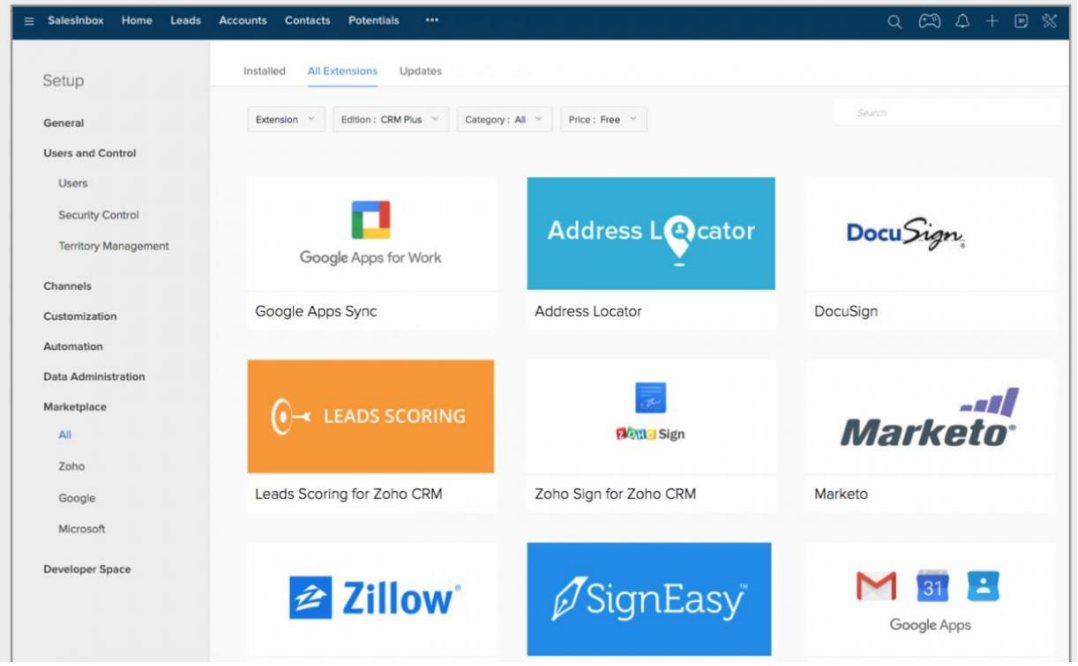

Рисунок 2.3 – Zoho CRM

Zoho CRM – це популярне рішення для малого та середнього бізнесу, яке вирізняється доступністю та функціональністю. Zoho пропонує широкий набір інструментів для управління клієнтами, автоматизації процесів продажу та маркетингу. Платформа відома своїм гнучким інтерфейсом і легкістю налаштування[26].

Основні переваги Zoho CRM:

– доступна ціна: Zoho пропонує більш доступні пакети порівняно з конкурентами, що робить його привабливим для малого бізнесу;

– легкість використання: інтерфейс Zoho простий і зручний, що дозволяє швидко адаптувати систему до потреб користувача;

– можливість інтеграцій: Zoho підтримує інтеграції з іншими популярними інструментами, такими як Google Apps, MailChimp, QuickBooks тощо.

Недоліки Zoho CRM:

– обмежені можливості для великих підприємств, зокрема у сфері аналітики та масштабування;

– можливості кастомізації можуть бути недостатніми для компаній з унікальними бізнес–процесами.

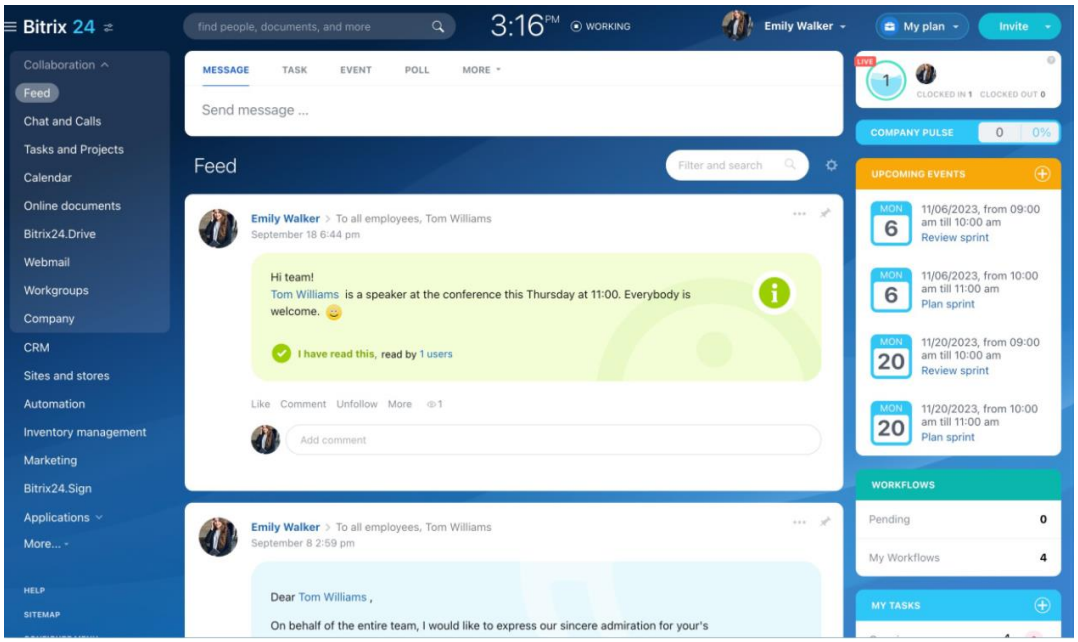

Рисунок 2.4 – Bitrix24

Bitrix24 – це CRM платформа, яка користується популярністю серед малих та середніх підприємств, особливо на ринку СНД. Вона надає безкоштовний тарифний план з базовими функціями CRM і додатковими платними можливостями для великих організацій[27-29].

Основні переваги Bitrix24:

– безкоштовний базовий план: Надає основні CRM функції безкоштовно для невеликих компаній.

– повноцінне управління проектами: Bitrix24 пропонує функціонал для управління проєктами, що інтегрований з CRM;

– інтеграція з соціальними мережами: платформа підтримує інтеграцію з соцмережами та мессенджерами, що дозволяє розширити комунікаційні канали з клієнтами.

Недоліки Bitrix24:

– обмежені функції для великих компаній і складність у налаштуванні для індивідуальних бізнес–процесів;

– інтерфейс може бути перевантаженим через велику кількість функцій.

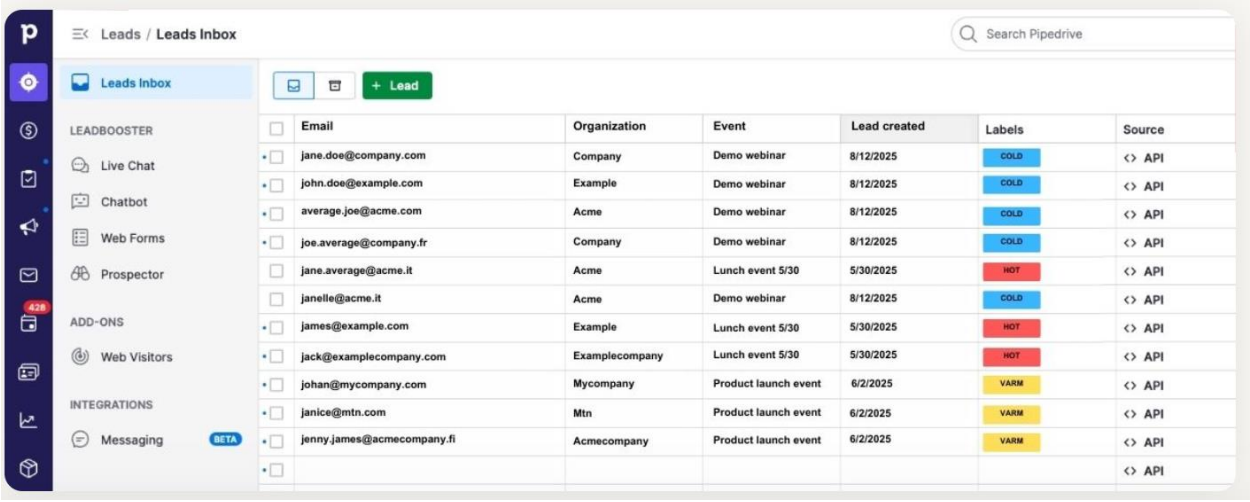

Рисунок. 2.5 – Pipedrive

Pipedrive – це CRM система, орієнтована на малий та середній бізнес, яка допомагає компаніям організувати процеси продажів. Вона відрізняється простим інтерфейсом і функціональністю, спрямованою на управління лідами та угодами[30-33].

Основні переваги Pipedrive:

– простий інтерфейс: легкість використання та управління лідами і угодами роблять Pipedrive популярним серед малих компаній;

– автоматизація продажів: система допомагає автоматизувати рутинні завдання, що дозволяє збільшити ефективність відділу продажів;

– аналітика продажів: Інструменти для аналізу та відстеження ефективності угод у реальному часі.

Недоліки Pipedrive:

– обмежені можливості для великих компаній або складних процесів продажів;

– відсутність глибокої інтеграції з деякими популярними інструментами для маркетингу і підтримки клієнтів.

На ринку CRM систем представлено велику кількість рішень, які можуть відповідати різним потребам компаній – від малого бізнесу до великих корпорацій. Salesforce та Microsoft Dynamics 365 є найбільш масштабованими і функціональними рішеннями, орієнтованими на великі підприємства, тоді як Zoho CRM, Bitrix24 і Pipedrive пропонують доступніші варіанти для малих і середніх компаній. Кожна з платформ має свої переваги і недоліки, тому вибір CRM системи має ґрунтуватися на специфічних вимогах бізнесу, можливості інтеграції з наявними системами та бюджеті[34-35].

Впровадження CRM системи є важливим стратегічним кроком для підприємства, оскільки дозволяє значно підвищити ефективність управління взаємовідносинами з клієнтами, автоматизувати рутинні процеси і покращити загальну продуктивність бізнесу.

2.2 Функціональна модель додатку

Сучасні компанії все частіше стикаються з необхідністю оптимізації взаємовідносин із клієнтами, управління продажами та закупівельними процесами. Для досягнення цих цілей доцільно розробити CRM систему, яка інтегрує методи закупівельної логістики, а також аналітичні підходи ABC і XYZ для класифікації

товарів і ресурсів. Розробка такої системи на мові програмування C# забезпечує гнучкість, надійність і можливість подальшого розширення функціоналу.[36]

Основна мета системи полягає в автоматизації збору та обробки інформації про клієнтів, постачальників і товари, а також у покращенні процесу прийняття рішень щодо закупівель і управління запасами. З точки зору функціональності, система повинна забезпечувати зручне управління клієнтськими даними, продажами, запасами та закупівлями. Однією з ключових функцій є можливість автоматизованого аналізу попиту на товари та їхньої важливості для бізнесу, що здійснюється за допомогою ABC і XYZ методів.

ABC аналіз базується на принципі Парето, згідно з яким 20% товарів приносять 80% прибутку. Основна ідея ABC аналізу полягає в класифікації товарів на три категорії: A, B і C[37-39].

Категорія A – це найцінніші товари, які складають приблизно 20% загального асортименту, але генерують близько 80% доходу.

Категорія B – це товари середньої важливості, що становлять приблизно 30% асортименту і забезпечують 15% доходу.

Категорія C – це товари низької важливості, які складають 50% асортименту, але дають лише 5% доходу.

Для проведення ABC аналізу використовується проста, але ефективна методика:

$$R_i = Q_i \times P_i \tag{2.1}$$

, де R_i– річне споживання товару i,

Q_i – кількість проданого товару i,

P_i – ціна одиниці товару i.

XYZ аналіз класифікує товари на основі стабільності попиту. Він допомагає визначити, які товари мають стабільний попит, які – коливальний, а які – випадковий. Цей метод дозволяє прогнозувати потреби в запасах і приймати відповідні рішення щодо управління ресурсами[40-42].

Категорія X – товари з високою стабільністю попиту, тобто їхній коефіцієнт варіації не перевищує 10–20%.

Категорія Y – товари із середнім рівнем стабільності попиту, для яких коефіцієнт варіації становить 20–50%.

Категорія Z – товари з нестабільним попитом і коефіцієнтом варіації понад 50%. Основою XYZ аналізу є коефіцієнт варіації V, який обчислюється за формулою:

$$V = (\sigma/Q) \times 100 \quad (2.2)$$

,де σ – стандартне відхилення попиту,

Q^{-} – середнє значення попиту на товар.

Розрахунок середнього значення попиту для товару ii:

$$Q_i = \frac{\sum_{j=1}^{m} Q_{ij}}{m} \quad (2.3)$$

,де Q_{ij} – кількість проданих товарів і в період j, а m – кількість періодів.

CRM система повинна надавати зручний інструмент для класифікації товарів за критеріями важливості та стабільності попиту. В цьому контексті, ABC аналіз дозволяє класифікувати товари за їхнім внеском у загальний прибуток підприємства, розподіляючи їх на категорії A, B та C. XYZ аналіз, у свою чергу, класифікує товари за рівнем передбачуваності попиту, що дозволяє краще планувати закупівлі та оптимізувати запаси[43-44].

Закупівельна логістика, як частина функціональної моделі, передбачає ефективне управління постачальниками та облік історії закупівель. Це дозволяє компанії не лише забезпечувати безперервність постачань, але й контролювати витрати та покращувати якість поставок. Інтеграція даних про клієнтів і постачальників у єдину CRM платформу дозволяє створювати комплексні аналітичні звіти для прийняття обґрунтованих рішень.

Важливим аспектом системи є забезпечення автоматизованого моніторингу запасів із можливістю прогнозування потреб на основі історичних даних. Це дозволяє запобігати дефіциту та надлишкам товарів, мінімізуючи витрати на їх

утримання. Для реалізації цієї функції система має забезпечувати контроль мінімальних і максимальних рівнів запасів та автоматичне формування замовлень постачальникам при досягненні критичних значень[45].

Інтеграція методів ABC і XYZ у CRM систему дозволяє ефективніше управляти запасами та оптимізувати закупівельні процеси. Класифікація товарів за категоріями A, B, C та X, Y, Z дозволяє створити матрицю ABC–XYZ для оптимізації управління запасами та планування закупівель.

Приклад матриці ABC–XYZ:

AX: товари з високою значимістю та стабільним попитом. Основна стратегія – підтримка постійного рівня запасів.

AY: товари середньої стабільності з високою значимістю. Рекомендується регулярний моніторинг запасів.

CZ: товари з низькою значимістю та непередбачуваним попитом. Рекомендовано мінімізацію запасів.

Функціональна модель представляє собою абстрактну структуру, яка зосереджує увагу на функціональному аспекті моделювання обраної предметної області. Така модель будується у вигляді ієрархії функцій, що полегшує розуміння взаємодії різних частин системи і логіки її роботи[46-47].

Порівнюючи ці два методи, можна відзначити, що ABC–аналіз зосереджений на фінансових аспектах і прибутковості, тоді як XYZ–аналіз акцентує увагу на поведінці попиту. Разом ці підходи можуть взаємно доповнювати одне одного, створюючи комплексну систему управління запасами. Наприклад, поєднання результатів ABC і XYZ дозволяє визначити товари, які є фінансово значущими (категорія A) та мають стабільний попит (категорія X). У таких випадках підприємство може зосередитися на забезпеченні постійної наявності цих товарів у потрібних обсягах. Водночас товари категорії C із нерегулярним попитом (категорія Z) можуть бути розглянуті як другорядні, що дозволяє зменшити інвестиції в їх запаси.

Важливим аспектом є практична реалізація цих методів. Для ABC–аналізу необхідно мати точні дані про доходи від кожної товарної позиції, тоді як XYZ–аналіз вимагає статистичної обробки даних про продажі та аналізу коливань попиту. У сучасних умовах підприємства можуть автоматизувати ці процеси за допомогою спеціалізованого програмного забезпечення, що знижує трудовитрати і підвищує точність аналізу[48].

Однак при використанні обох методів необхідно враховувати їх обмеження. ABC–аналіз може недооцінити важливість окремих товарів, особливо тих, що входять до категорії C, але мають стратегічне значення для залучення клієнтів. XYZ–аналіз, у свою чергу, може бути неефективним для нових товарів або в умовах швидко змінюваного ринку, де історичних даних для аналізу недостатньо.

Розроблена нами CRM система, як і будь–яка інша програма, має власний алгоритм роботи. Ця система повинна відповідати певним вимогам і завданням, які ми визначаємо на етапі проєктування.

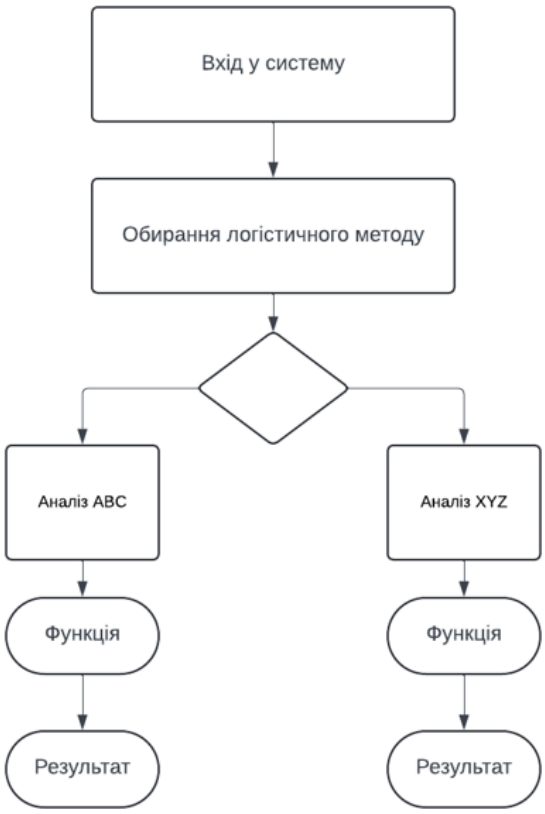

Рисунок 2.6 – Функціональна модель CRM системи

Тут представлено спрощену схему роботи додатку, яка показує основні функціональні частини системи, їхній зв’язок та внутрішні взаємодії. Така модель сприяє кращому розумінню архітектури програми та її функцій.

На основі проведеного дослідження ми можемо визначити можливі недоліки, які необхідно врахувати під час реалізації програми. Це дозволить розробити ефективний алгоритм роботи системи, де кожен етап відповідає діям користувача, а також оптимізувати процеси взаємодії[49].

Рисунок 2.7 – Алгоритм роботи CRM системи

Ключове завдання полягає у розробці зрозумілого та інтуїтивного інтерфейсу для користувача, забезпеченні швидкості роботи додатку, а також оптимізації коду для виключення помилок і неточностей. Потрібно передбачити функціональні підказки та інші допоміжні елементи, що спростять використання системи, зберігаючи при цьому її функціональність на високому рівні.

Вимоги до проекту та його реалізації:

- відповідність поставленим завданням;
- інтеграція логістичних методів ABC і XYZ;
- інтуїтивний і зрозумілий інтерфейс;
- простота навігації;
- наявність підказок і довідкових матеріалів;
- точність і швидкість виконання задач;

– досягнення визначених цілей системи.

У попередньому аналізі було виявлено недоліки деяких схожих систем, що дозволяє нам створити CRM систему, яка поєднує в собі ефективні алгоритми управління логістикою з інтерактивними елементами аналітики та управління. Це дозволить не тільки автоматизувати управління взаємовідносинами з клієнтами, а й оптимізувати логістичні процеси на основі передових аналітичних методів[50].

2.3 Методи для створення CRM системи

При створенні CRM–системи на мові C# важливе місце займає планування архітектури проекту. Сучасні CRM–системи вимагають високої масштабованості, модульності та гнучкості. Тому розробникам важливо обрати відповідні шаблони проектування, такі як MVC (Model–View–Controller) або MVVM (Model–View–ViewModel), які дозволяють чітко розділити функціональність на логічні компоненти. Це не лише полегшує розробку, але й сприяє підтримці та модифікації системи в майбутньому, забезпечуючи структурованість та чітке розмежування відповідальностей між різними модулями.

Одним із важливих аспектів розробки CRM–систем є ефективне управління даними. Для цього зазвичай використовуються ORM (Object–Relational Mapping) інструменти, такі як Entity Framework або Dapper, які забезпечують об’єктно–реляційне відображення бази даних. Ці інструменти дозволяють розробникам працювати з базою даних як з об’єктами, що значно спрощує процес написання коду для CRUD–операцій (створення, читання, оновлення, видалення). За допомогою Entity Framework можна створювати та керувати таблицями для зберігання інформації про клієнтів, замовлення, товари, постачальників та інші важливі елементи. Крім того, ORM дозволяють здійснювати складні SQL–запити та оптимізувати пошукові операції за допомогою індексів, що важливо для швидкого доступу до великої кількості даних.

Однією з ключових складових CRM–системи є база даних, яка забезпечує зберігання, обробку і доступ до інформації про клієнтів, транзакції, продукти та інші важливі дані. У рамках створення CRM–системи на мові C# базу даних було реалізовано за допомогою MSSQL Server, а для інтеграції програмного коду з базою застосовано технологію Entity Framework.

CRM–система працює з великим обсягом даних, які часто є структурованими та мають тісні взаємозв’язки між собою. База даних виконує функцію центрального сховища, де організовано зберігається вся інформація про клієнтів, угоди, замовлення, товари, а також будь–які метадані, пов'язані з роботою системи. Завдяки використанню MSSQL Server база даних стає надійною основою для зберігання інформації, забезпечуючи такі функції, як:

- швидкий доступ до даних через запити;
- забезпечення цілісності та унікальності даних завдяки використанню індексів і ключів;
- підтримка транзакцій для забезпечення надійності операцій з даними;
- резервне копіювання та відновлення, що гарантує збереження даних у разі збою.

Бази даних SQL являють собою потужний та структурований механізм зберігання, обробки та маніпулювання даними, який став стандартом у світі інформаційних технологій. Реляційні бази даних створені на принципах математичної теорії реляційної алгебри, розробленої Едгаром Коддом у 1970 році, та забезпечують надійне та ефективне управління структурованою інформацією.

Архітектура SQL бази даних базується на концепції таблиць, які є основними контейнерами для зберігання даних. Кожна таблиця складається з рядків (записів) та стовпців (полів), де кожен стовпець має чітко визначений тип даних. Типи даних включають числові (integer, decimal, float), текстові (varchar, text, char), часові (date, datetime, timestamp), а також спеціалізовані типи для зберігання унікальних ідентифікаторів, бінарних даних та інших специфічних форматів.

Ключовим елементом реляційних баз даних є система первинних та зовнішніх ключів. Первинний ключ (primary key) унікально ідентифікує кожен запис у таблиці, забезпечуючи цілісність даних. Зовнішній ключ (foreign key) встановлює зв'язок між двома таблицями, дозволяючи створювати складні реляційні структури та підтримувати посилальна цілісність.

SQL (Structured Query Language) – декларативна мова програмування, призначена для управління та маніпулювання реляційними базами даних. Вона дозволяє виконувати широкий спектр операцій: від простого вибору даних до складних агрегацій, групувань та трансформацій. Основні типи операторів SQL включають Data Definition Language (DDL), Data Manipulation Language (DML), Data Control Language (DCL) та Transaction Control Language (TCL).

Join–функції становлять окремий та надзвичайно важливий клас операторів у SQL, які дозволяють об'єднувати записи з різних таблиць на основі певних умов зв'язку. Вони є потужним інструментом для побудови складних запитів та отримання агрегованої інформації з декількох джерел даних.

INNER JOIN є найбільш поширеним типом об'єднання, який повертає лише ті записи, що мають збіги в обох таблицях відповідно до вказаної умови зв'язку. Наприклад, при з'єднанні таблиць "Замовлення" та "Клієнти" за ідентифікатором клієнта, результат міститиме лише ті замовлення, які мають відповідного клієнта.

LEFT JOIN (або LEFT OUTER JOIN) повертає всі записи з лівої таблиці та відповідні записи з правої таблиці. Якщо для запису з лівої таблиці немає збігу в правій, то такі поля будуть заповнені NULL значеннями. Це корисно, коли потрібно зберегти повноту інформації з основної таблиці.

RIGHT JOIN є симетричним до LEFT JOIN і повертає всі записи з правої таблиці та відповідні записи з лівої. Знову ж таки, відсутні збіги доповнюються NULL значеннями.

FULL OUTER JOIN об'єднує характеристики LEFT та RIGHT JOIN, повертаючи всі записи з обох таблиць. Записи без збігів у протилежній таблиці доповнюються NULL значеннями, що дає повну картину даних з обох джерел.

CROSS JOIN створює декартів добуток двох таблиць, тобто генерує всі можливі комбінації рядків з обох таблиць. Цей тип з'єднання рідко використовується безпосередньо, але може бути корисним у певних аналітичних сценаріях.

SELF JOIN є унікальною технікою, коли таблиця з'єднується сама з собою. Це корисно для ієрархічних структур, наприклад, у випадку організаційної ієрархії співробітників або категорій продуктів.

При побудові join–запитів важливо враховувати продуктивність. Неправильно сконструйований join може призвести до повільного виконання запиту, особливо на великих обсягах даних. Тому розробники використовують індекси, оптимізують умови з'єднання та аналізують план виконання запиту.

Сучасні системи управління базами даних, такі як PostgreSQL, MySQL, Microsoft SQL Server та Oracle, пропонують розширені можливості для роботи з join–операціями. Вони підтримують складні синтаксичні конструкції, включаючи підзапити, агрегаційні функції та віконові функції, що значно розширює аналітичні можливості SQL.

Безпека та цілісність даних у SQL базах забезпечується через механізми транзакцій, обмежень цілісності, тригерів та stored procedures. Транзакції гарантують, що набір операцій буде виконаний повністю або не виконаний взагалі, запобігаючи частковим змінам у даних.

Процес створення бази даних розпочався з розробки її логічної моделі, яка відображає всі сутності системи та їхні зв’язки. У контексті CRM–системи ключовими сутностями є клієнти, продукти, угоди, рахунки та співробітники. Наприклад:

Таблиця Customers зберігає інформацію про клієнтів, включаючи їх імена, контактні дані, адреси та статуси.

Таблиця Orders відповідає за облік замовлень, містить зв'язок із клієнтами та включає інформацію про товари.

Таблиця Products зберігає дані про товари чи послуги, які пропонує компанія, зокрема їх ціни, наявність і категорії.

Кожна таблиця має первинні ключі, які забезпечують унікальність записів, і зовнішні ключі, що створюють зв'язки між таблицями. Наприклад, таблиця Orders містить зовнішній ключ, який зв'язує її з таблицею Customers, що дозволяє відстежувати замовлення кожного клієнта.

Після розробки логічної моделі структура бази даних була створена в MSSQL Server. Цей етап включає визначення таблиць, стовпців, типів даних, а також правил валідації, таких як обмеження NOT NULL для обов'язкових полів і використання унікальних індексів.

Інструмент SQL Server Management Studio (SSMS) значно спростив виконання цих завдань. Він дозволив створити таблиці через графічний інтерфейс, а також за допомогою SQL–запитів. Наприклад, таблиця для зберігання інформації про клієнтів була створена наступним чином:

```
CREATE TABLE Customers(
    CustomerID INT PRIMARY KEY IDENTITY(1,1),
    Name NAVCHAR(100) NOT NULL,
    Email NAVCHAR(100) NOT NULL UNIQUE,
    Phone NAVCAR(15),
    Address NAVCHAR(255),
    Status NAVCHAR(46)
    );
```

Рисунок 2.8 – Приклад SQL запиту

Цей запит визначає таблицю Customers, яка має ідентифікатор клієнта як первинний ключ і забезпечує унікальність електронної пошти.

Для роботи з базою даних із програми на C# було обрано технологію Entity Framework (EF), яка забезпечує об'єктно–реляційне відображення (ORM). EF дозволяє працювати з базою даних, використовуючи об'єкти C# замість написання SQL–запитів вручну. Це значно прискорює процес розробки та зменшує ймовірність помилок.

На етапі інтеграції було застосовано підхід Code–First, за якого модель бази даних визначається у вигляді класів у коді C#. Наприклад, клас для таблиці Customers виглядає так:

```
public class Customer
{
    Ссылок: 0
    public int CustumerID { get; set; }
    Ссылок: 0
    public string Name { get; set; }
    Ссылок: 0
    public string Email { get; set; }
    Ссылок: 0
    public string Phone { get; set; }
    Ссылок: 0
    public string Adress { get; set; }
    Ссылок: 0
    public string Status{ get; set; }
}
```

Рисунок 2.9 – Приклад класу для бази даних

Цей клас визначає структуру таблиці, а EF автоматично створює відповідну базу даних або синхронізує її з кодом, якщо база вже існує.

Під час виконання програми Entity Framework дозволяє легко виконувати CRUD–операції (створення, читання, оновлення, видалення). Наприклад, для додавання нового клієнта до бази даних використовується наступний код:

```
private void button1_Click(object sender, EventArgs e)
{
    Product product = new Product();
    product.Id = 1;
    product.Name=textBox1.Text;
    product.Count = Convert.ToInt32(textBox2.Text);
    product.Buying_price = Convert.ToInt32(textBox4.Text);
    product.Sell_price = Convert.ToInt32(textBox3.Text);
    product.DateTime= DateTime.Now;
    db.Products.Add(product);
    db.SaveChanges();
    Close();
}
```

Рисунок 2.10 – Приклад операції додавання до бази данних

Цей код створює новий об’єкт Customer, додає його до контексту бази даних і зберігає зміни. Подібним чином здійснюється читання та оновлення даних.

Для забезпечення високої продуктивності бази даних використовуються індекси, які прискорюють виконання запитів. Наприклад, індексування стовпця Email у таблиці Customers гарантує швидкий пошук клієнтів за цим параметром.

Щоб підвищити рівень безпеки, використовується аутентифікація на рівні MSSQL Server. Крім того, доступ до бази даних із програми захищений за допомогою рядків підключення (connection strings), що зберігаються у конфігураційних файлах із шифруванням.

На етапі розробки всі операції з базою даних проходять тестування для виявлення помилок та перевірки цілісності даних. У MSSQL Server використовуються тестові запити, а в програмному коді – модульне тестування методів, які працюють із базою. Це дозволяє забезпечити надійну роботу CRM–системи у реальних умовах.

Ми створюємо модельні класи, що відповідають таблицям у базі даних. Наприклад, клас Customer відповідає таблиці клієнтів і містить всі необхідні поля, такі як ім'я, адреса, номер телефону тощо. За допомогою Entity Framework ми можемо використовувати LINQ–запити для доступу до цих об'єктів, що дозволяє писати SQL–запити на рівні C# коду.

CRUD–операції – це основа функціонування будь–якої CRM–системи, адже вони забезпечують базову взаємодію з даними. Назва CRUD є абревіатурою, яка позначає чотири основні дії: створення (Create), читання (Read), оновлення (Update) та видалення (Delete). Саме ці операції дозволяють ефективно працювати з базою даних, забезпечуючи гнучкість і актуальність інформації, яка зберігається у системі.

У CRM–системах дані відіграють ключову роль, адже вони зберігають інформацію про клієнтів, їхні замовлення, історію взаємодій, товари та багато іншого. CRUD–операції дають змогу організовувати цю інформацію, гарантуючи, що вона завжди відповідає актуальному стану бізнесу. Наприклад, коли додається новий клієнт, операція створення забезпечує внесення його даних у базу. Завдяки цьому менеджери можуть мати доступ до цієї інформації в будь–який час.

Читання даних є не менш важливим, оскільки CRM без доступу до наявної інформації втрачає свою ефективність. Це дозволяє співробітникам отримувати необхідні дані для аналізу, прийняття рішень і планування взаємодій із клієнтами. Наприклад, менеджер може переглянути історію покупок клієнта, щоб запропонувати персоналізовану знижку або новий продукт.

Операція оновлення дозволяє підтримувати дані в актуальному стані. У бізнесі обставини змінюються: клієнти оновлюють контактну інформацію, товари змінюють ціни, або замовлення переходять на новий етап обробки. Оновлення даних гарантує, що CRM відображає поточний стан справ, що особливо важливо для збереження довіри клієнтів і оптимізації бізнес–процесів.

Видалення забезпечує очищення системи від застарілих або непотрібних даних, зберігаючи її ефективність і швидкодію. Наприклад, видалення дублікатів клієнтських записів або видалення застарілих продуктів з каталогу дозволяє зберегти базу даних організованою.

З точки зору технічної реалізації, CRUD–операції служать фундаментом для створення динамічного інтерфейсу CRM–системи. За допомогою таких технологій,

як Entity Framework у поєднанні з Windows Forms або іншими інструментами, CRUD дозволяють розробникам швидко налаштовувати взаємодію між користувачем і базою даних. Ці операції також є невід'ємною частиною бізнес–логіки системи, забезпечуючи точність і надійність обробки інформації.

У CRM–системах CRUD–операції відіграють роль фундаменту, на якому будується уся функціональність, що дозволяє компаніям ефективно управляти клієнтами, аналізувати дані й автоматизувати бізнес–процеси. Без них неможливо забезпечити ні зручність роботи користувачів, ні правильне функціонування системи. Вони гарантують, що інформація є доступною, актуальною і керованою, роблячи CRM–систему потужним інструментом для управління бізнесом.

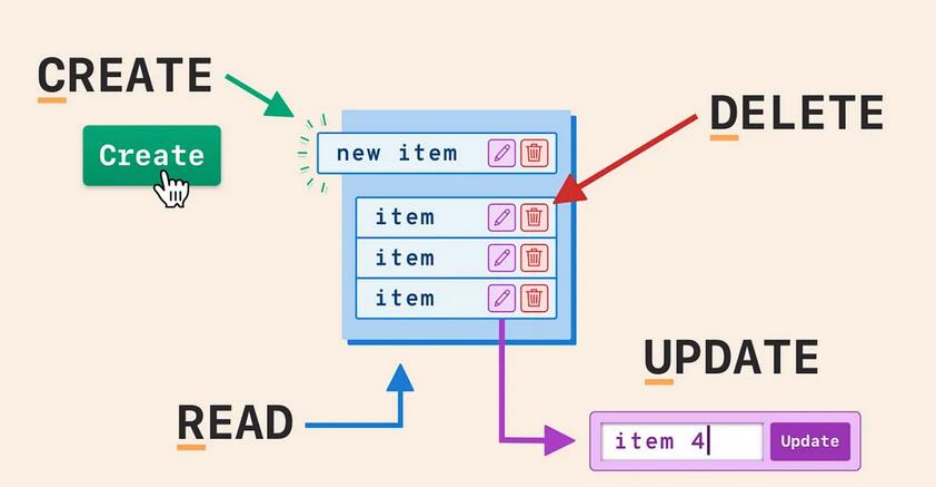

Рисунок 2.11 – Схема CRUD операцій

Що стосується інтеграції CRM–системи з Excel, ми обрали бібліотеку EPPlus для роботи з файлами Excel. Ця бібліотека забезпечує простий та ефективний інтерфейс для створення, читання та модифікації таблиць Excel. Використання Excel стає необхідним, коли потрібно імпортувати велику кількість даних про клієнтів або продукти, отримані з інших джерел, або експортувати результати аналізу та звіти. За допомогою EPPlus ми можемо програмно створювати файли, формувати таблиці, додавати стилі та формули. Наприклад, ми використовуємо це

для автоматизації формування звітів, які містять підсумки ABC та XYZ аналізів, що спрощує роботу менеджерів із великими обсягами даних.

В межах створення CRM–системи для реалізації графічного інтерфейсу ми обрали технологію WinForms. WinForms – це класична платформа для розробки десктопних додатків на мові C#. Вона забезпечує швидку розробку інтерфейсу користувача, пропонуючи великий вибір елементів керування, таких як таблиці, кнопки, текстові поля, комбобокси тощо. Це дозволяє створювати зручні та інтуїтивно зрозумілі форми для менеджерів, які працюють із CRM–системою.

При розробці на WinForms ми створюємо головну форму, яка служить як центральний пункт для доступу до різних розділів CRM–системи. Наприклад, користувач може легко перемикатися між модулями управління клієнтами, замовленнями, звітністю та аналітикою. Для роботи з таблицями ми використовуємо елементи керування типу DataGridView, які дозволяють зручно відображати дані та інтерактивно взаємодіяти з ними (редагування, фільтрація, сортування).

Окрім того, WinForms дозволяє інтегрувати функціональність імпорту та експорту даних з Excel, що дуже зручно для користувачів. Наприклад, ми реалізуємо функцію, яка дозволяє завантажити файл Excel та автоматично імпортувати дані в базу CRM–системи. Це значно полегшує роботу користувачів із великими обсягами інформації та мінімізує ручне введення даних.

Таким чином, інтеграція з базою даних, робота з Excel та використання WinForms дозволяють нам створити надійну та гнучку CRM–систему, яка забезпечує ефективне управління клієнтами та оптимізацію бізнес–процесів. Завдяки можливостям WinForms ми можемо забезпечити користувачам зручний інтерфейс для взаємодії з системою, а робота з базою даних та Excel надає інструменти для роботи з великими обсягами даних та аналітики.

Особливе місце в управлінні запасами займають методи ABC і XYZ класифікації, які широко застосовуються для аналізу асортименту товарів. ABC

аналіз базується на принципі Парето (20/80) та дозволяє класифікувати товари залежно від їхньої значущості для компанії. Товари класу A є найціннішими, і саме вони формують основну частку доходу, тоді як товари класів B і C менш важливі. XYZ аналіз, у свою чергу, використовується для визначення стабільності споживання товарів. Товари класу X характеризуються стабільним попитом, класу Y – середньою стабільністю, а класу Z – нерегулярним або непередбачуваним попитом. Реалізація цих аналізів на C# можлива за допомогою алгоритмів сортування і групування даних. За допомогою LINQ–запитів можна легко впровадити сортування товарів для ABC аналізу та обчислення коефіцієнта варіації для XYZ аналізу.

Асинхронне програмування відіграє ключову роль у створенні продуктивних і ефективних CRM–систем. В умовах, коли система має обробляти великий обсяг даних, виконувати запити до бази даних, інтегруватися із зовнішніми API та виконувати операції з файлами, важливо уникати блокування основного потоку програми. Використовуючи асинхронні методи, ми можемо покращити відгук користувацького інтерфейсу, збільшити продуктивність та оптимізувати розподіл ресурсів, але існують і інші підходи до написання коду, такі як однопоточне та багатопоточне програмування.

Витоки однопоточного програмування сягають далеких 1940–50–х років, коли перші комп'ютери являли собою величезні механічні монстри, здатні виконувати лише одну послідовну операцію за одиницю часу. У ті часи кожна обчислювальна машина нагадувала свого роду інтелектуального гіганта, який міг концентруватися лише на одному завданні, послідовно опрацьовуючи кожну інструкцію. Програмісти того періоду були свого роду диригентами, які вибудовували надзвичайно точні алгоритми виконання завдань.

Однопоточне програмування – це класичний підхід, де виконання програми відбувається строго послідовно, один instruction за одним. Уявіть собі конвеєр, де кожна наступна операція розпочинається лише після повного завершення

попередньої. Такий метод має як очевидні переваги, так і суттєві обмеження. З одного боку, він надзвичайно простий для розуміння та реалізації, з іншого – абсолютно неефективний для складних обчислювальних завдань.

Багатопоточне програмування з'явилося як революційна відповідь на обмеження однопоточної парадигми. Перші експериментальні реалізації багатопоточності з'явилися наприкінці 1960–х років у наукових лабораторіях, де дослідники намагалися паралельно виконувати декілька незалежних завдань. Концепція була простою, але геніальною: замість послідовного виконання операцій, комп'ютер може розподіляти навантаження між декількома потоками виконання.

Уявіть багатопоточність як команду висококваліфікованих робітників, які можуть одночасно працювати над різними ділянками складного проєкту. Один потік опрацьовує графічний інтерфейс, інший – виконує складні математичні обчислення, третій – здійснює мережеві комунікації. Такий підхід дозволяє радикально підвищити продуктивність обчислювальної системи, максимально ефективно використовуючи наявні апаратні ресурси.

У C# однопоточне та багатопоточне програмування реалізуються за допомогою класів та ключових слів простору імен System.Threading. Для створення потоків використовується клас Thread, який дозволяє запускати методи паралельно. Ключове слово async/await спрощує асинхронне програмування, даючи змогу писати синхронний код, який насправді виконується асинхронно.

Для створення нового потоку можна використати конструктор Thread, передавши йому делегат або лямбда–вираз, який визначає код для виконання. Метод Start() запускає потік, а Join() дозволяє очікувати завершення потоку. Клас Task надає більш сучасний і гнучкий підхід до багатопоточності, дозволяючи легко створювати паралельні завдання за допомогою Task.Run().

Синхронізація потоків здійснюється через lock(), Monitor, Mutex та інші примітиви синхронізації, які запобігають одночасному доступу до спільних

ресурсів. Paralell.For та Paralell.ForEach дають змогу легко розпаралелювати цикли, виконуючи ітерації одночасно.

Важливо правильно керувати потоками, уникати взаємних блокувань і забезпечувати потокову безпеку при роботі зі спільними даними, використовуючи механізми синхронізації та атомарні операції.

Асинхронне програмування – найбільш еволюційна та сучасна парадигма, яка народилася з усвідомлення обмежень попередніх підходів. ЇЇ витоки припадають на кінець 1990–х – початок 2000–х років, коли розвиток мережевих технологій та інтернету висунув принципово нові вимоги до обробки інформації. Класичні синхронні моделі виявилися неспроможними ефективно опрацьовувати складні, розподілені операції з високою латентністю.

Концептуальна різниця асинхронного програмування полягає в принципово новому підході до виконання операцій. Замість блокування основного потоку виконання під час очікування результату довготривалої операції, система може продовжувати виконання інших завдань. Уявіть собі офіс, де співробітники не просто чекають, поки щось завантажиться, а продовжують бути продуктивними.

Чому саме асинхронне програмування стає домінуючим підходом при розробці CRM–систем? Відповідь криється в природі сучасного бізнес–середовища. CRM – це не просто база даних, а складна екосистема комунікацій, інтеграцій, паралельних процесів. Уявіть систему, яка одночасно опрацьовує десятки тисяч клієнтських взаємодій, здійснює аналітичні розрахунки, формує звіти та підтримує мережеві комунікації.

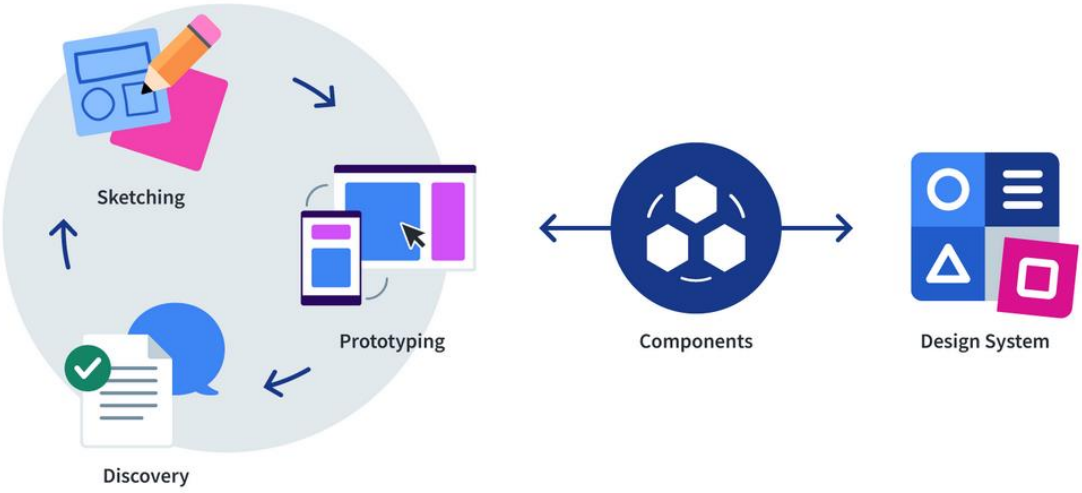

Рисунок 2.12 – Цикл Розробки responsive системи

Асинхронність дозволяє створювати надзвичайно продуктивні та responsive системи. При розробці CRM ми маємо справу з величезними масивами даних, складними аналітичними моделями, інтеграціями з зовнішніми сервісами. Традиційні синхронні підходи миттєво перетворилися б на вузьке горло, яке критично сповільнює роботу всієї системи.

Технологічні гіганти – Google, Facebook, Netflix – вже давно обрали асинхронне програмування магістральним напрямком розвитку своїх платформ. Це не просто технічний вибір, а філософія побудови максимально ефективних, масштабованих та гнучких систем. Асинхронність дозволяє створювати додатки, які практично не мають обмежень продуктивності.

Технологічне майбутнє – за системами, здатними максимально ефективно використовувати наявні обчислювальні потужності, миттєво адаптуватися до мінливих вимог користувачів та забезпечувати непомітну, але надзвичайно потужну роботу backend–інфраструктури. І саме асинхронне програмування є ключем до розв'язання цього амбітного технологічного завдання, тому розберемо у чому полягає використання асинхронності і інші нюанси зв’язані з цим.

Асинхронне програмування в C# реалізується за допомогою ключових слів async та await. Ми оголошуємо метод як асинхронний за допомогою ключового слова async, що дозволяє вказати, що цей метод може виконувати тривалі операції (такі як запити до бази даних або обмін інформацією з веб–сервісами) без блокування потоку. Ключове слово await, у свою чергу, використовується для очікування результату асинхронної операції, дозволяючи програмі продовжувати виконання інших завдань.

У межах розробки CRM–системи ми активно використовуємо асинхронні методи для роботи з базою даних. Наприклад, при запитах до бази даних з використанням Entity Framework ми використовуємо методи ToListAsync(), FirstOrDefaultAsync(), SaveChangesAsync(), що дозволяють виконувати операції без блокування основного потоку. Це особливо важливо, коли користувач взаємодіє з інтерфейсом, оскільки блокування може призвести до зупинки роботи програми та погіршення користувацького досвіду.

Іншим важливим аспектом, де асинхронне програмування є незамінним, є інтеграція з зовнішніми сервісами через API. Наприклад, якщо наша CRM–система має отримувати актуальні дані про курси валют, перевіряти інформацію про клієнтів або надсилати повідомлення, ми використовуємо HTTP–запити за допомогою класу HttpClient. Всі методи HttpClient, такі як GetAsync(), PostAsync(), PutAsync(), працюють асинхронно, що дозволяє уникати блокування під час очікування відповіді від зовнішнього сервісу. Це забезпечує швидку реакцію системи та дозволяє обробляти інші операції паралельно.

У CRM–системі ми також можемо використовувати асинхронні методи для обробки файлів, таких як Excel або CSV, які містять великі обсяги даних. Наприклад, при імпорті даних ми використовуємо асинхронні методи читання файлів, що забезпечує зручність та швидкість обробки. Це корисно, коли потрібно завантажити або зберегти великий файл без блокування основного потоку, що дозволяє системі залишатися продуктивною та швидкодійною.

Асинхронність у поєднанні з WinForms або WPF дає змогу створювати зручний та відгукливий інтерфейс користувача. Наприклад, якщо ми реалізуємо форму пошуку клієнтів, де користувач вводить ім’я або інші критерії пошуку, асинхронні методи дозволяють виконувати запити до бази даних у фоновому режимі, не блокуючи головну форму. Це дозволяє продовжувати користуватися іншими елементами інтерфейсу, такими як фільтри або кнопки, поки система шукає інформацію.

Асинхронне програмування дозволяє нам також використовувати багатопоточність, особливо під час виконання складних розрахунків або обробки великої кількості даних. На мові C# ми можемо використовувати Task та Task.Run(), щоб запустити обробку даних в окремих потоках. Це особливо важливо при виконанні аналізу ABC і XYZ, де може знадобитися виконання великої кількості обчислень.

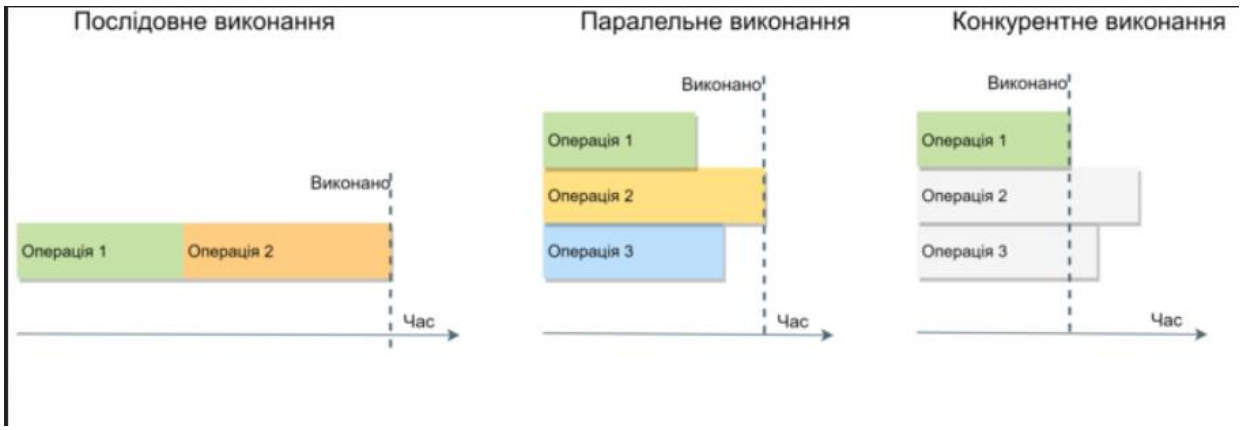

Рисунок 2.13 – Графічне зображення асинхронного програмування

Загалом, використання асинхронного програмування в CRM–системі забезпечує значні переваги, такі як зменшення часу очікування користувачем, підвищення продуктивності та відгуку системи, а також ефективне управління ресурсами. Це дозволяє забезпечити стабільну роботу навіть при великому навантаженні та складних операціях, що є критично важливим для сучасних CRM–рішень.

Інтеграція CRM–систем із Excel відіграє важливу роль у бізнес–аналітиці. Бібліотеки Microsoft.Office.Interop.Excel або EPPlus дозволяють реалізувати імпорт та експорт даних у форматі Excel, що є зручним для користувачів та аналітиків. За допомогою цих бібліотек можна завантажувати великі обсяги даних про клієнтів, товари або транзакції для подальшого аналізу. Крім того, можливість автоматичного формування звітів для менеджерів та відділів закупівель на основі результатів ABC і XYZ аналізу значно спрощує роботу з великими обсягами інформації та підвищує точність прийняття рішень.

Для прогнозування попиту та оптимізації запасів можна застосовувати алгоритми машинного навчання. У контексті C# корисним інструментом є бібліотека ML.NET, яка дозволяє створювати та навчати моделі прогнозування на основі історичних даних. Це відкриває можливості для більш точної оцінки попиту на товари, що, в свою чергу, сприяє оптимізації процесу закупівель та мінімізації ризиків дефіциту чи надлишкових запасів.

Інтеграція CRM–системи з іншими модулями чи системами можлива завдяки реалізації REST API на основі ASP.NET Core. Це дозволяє забезпечити централізоване зберігання даних та ефективний обмін інформацією між різними відділами організації (наприклад, відділами продажу та логістики). REST API також сприяє інтеграції з мобільними додатками чи веб–інтерфейсами, що є важливим для забезпечення доступу до інформації в реальному часі.

Для обробки великих обсягів даних або виконання складних обчислень, пов’язаних із аналізом і оптимізацією, доцільно використовувати багатопоточність. На мові C# це забезпечується використанням класів Task та Parallel, які дозволяють реалізувати паралельну обробку даних. Це корисно під час виконання ABC і XYZ аналізів, оскільки паралельні операції можуть значно скоротити час виконання складних обчислень та підвищити продуктивність системи.

Аналітичні методи ABC і XYZ забезпечують основу для оптимізації процесів закупівельної логістики. На основі отриманих результатів можна застосовувати

алгоритми оптимізації, такі як лінійне програмування або методи планування ресурсів, для зменшення витрат на зберігання запасів та покращення управління ланцюгами постачання. Завдяки аналітиці, яка базується на ABC і XYZ аналізах, компанії можуть краще планувати свої закупівлі, мінімізуючи ризики браку товарів та надлишкових запасів.

Таким чином, розробка CRM–системи на мові програмування C# із впровадженням методів закупівельної логістики та класифікаційних методів ABC і XYZ дозволяє значно підвищити ефективність управління запасами та взаємовідносинами з клієнтами. Комбінація різних методів програмування, інтеграція з базами даних, робота з Excel, використання асинхронності та багатопоточності, а також застосування алгоритмів прогнозування та оптимізації забезпечують надійну основу для створення високопродуктивної та ефективної CRM–системи.

2.4 Методології розробки програмного забезпечення для CRM систем

Розробка сучасних CRM–систем являє собою надзвичайно складний та багатогранний процес, який виходить далеко за межі звичайного програмування. Це унікальна сфера, де перетинаються технологічні інновації, бізнес–аналітика, психологія комунікацій та стратегічне планування. Кожна CRM–система – це по суті живий організм, який постійно еволюціонує, адаптується до мінливих ринкових умов та індивідуальних потреб конкретних підприємств.

Історія становлення методологій розробки програмного забезпечення сягає корінням далеких 1950–60–х років, коли перші електронно–обчислювальні машини лише започатковували свій тріумфальний похід у світ технологій. В ті часи процес створення програмних продуктів нагадував складне мистецтво, де талант окремих розробників важив набагато більше, ніж формалізовані процеси. Програмісти були свого роду технологічними чаклунами, які інтуїтивно відчували

логіку машини та могли перетворювати абстрактні математичні концепції на працюючий код.

Класична модель Waterfall, яка домінувала протягом тривалого періоду технологічного розвитку, являла собою надзвичайно стрункy та логічну послідовність етапів розробки. Уявіть собі величезний водоспад, де кожен наступний каскад народжується лише після повного завершення попереднього. Спочатку детально аналізуються та формалізуються вимоги замовника – створюється свого роду технічна Біблія майбутнього продукту. Потім розробники створюють розгорнуту архітектуру системи, прораховуючи десятки сценаріїв використання та можливих інтеграційних взаємодій. Далі розпочинається безпосередня розробка – процес перетворення абстрактних діаграм та специфікацій на конкретний програмний код.

Для CRM–систем така методологія мала надзвичайно амбівалентну природу. З одного боку, вона дозволяла максимально точно спланувати бюджет, визначити чіткі часові рамки проєкту та мінімізувати ризики непередбачуваних витрат. Замовник міг бути впевнений, що отримає продукт, який максимально точно відповідає первинному технічному завданню. З іншого боку, така модель була надзвичайно ригідною та погано пристосованою до реалій сучасного динамічного бізнесу.

Уявіть собі ситуацію: замовник замовляє CRM–систему, яка розроблятиметься впродовж року. За цей час ринок може кардинально змінитися, з'являться нові технології, трансформуються бізнес–процеси. Але через жорстку структуру Waterfall внесення змін на пізніх етапах розробки перетворюється на надскладне технічне та організаційне завдання. Це все одно, що намагатися змінити архітектуру хмарочоса вже після завершення будівництва.

Поява гнучких методологій, передусім Agile, стала справжньою технологічною революцією. Це була принципово нова філософія створення програмного забезпечення, яка радикально змінила підхід до розробки складних

інформаційних систем. Замість намагання передбачити всі можливі сценарії використання одразу, методологія пропонувала розробляти продукт ітераційно, короткими циклами, постійно отримуючи зворотній зв'язок від замовника.

Agile – це не просто технічний підхід, а цілісна філософія роботи, яка базується на принципах людяності, взаємоповаги та постійного навчання. Команда розробників розглядається не як набір технічних виконавців, а як живий організм, здатний до саморозвитку та адаптації. Кожен член команди несе персональну відповідальність за кінцевий результат, а комунікації відбуваються максимально прозоро та відкрито.

Особливо яскраво переваги Agile проявляються саме при розробці CRM–систем. Жодна, навіть найдосконаліша специфікація не здатна повністю описати всі нюанси бізнес–процесів конкретної компанії. Кожен клієнт – унікальний, зі своєю специфічною корпоративною культурою, неформальними комунікаційними моделями, особливостями взаємодії між підрозділами.

Методологія Scrum, яка є свого роду втіленням Agile – філософії, додатково посилює гнучкість розробки. Вона запроваджує поняття спринтів – коротких, зазвичай двотижневих циклів, протягом яких команда розробників має створити повноцінний, готовий до використання інкремент продукту. Scrum –майстер, який очолює команду, слідкує за дотриманням правил методології та усуває можливі перешкоди на шляху команди.

Для CRM–систем такий підхід є особливо ефективним. По–перше, він дозволяє замовнику бачити проміжні результати та вносити корективи буквально на льоту. По–друге, команда розробників має змогу постійно навчатися, аналізувати власні помилки та покращувати процеси. По–третє, знижуються ризики створення продукту, який не відповідає реальним бізнес–потребам.

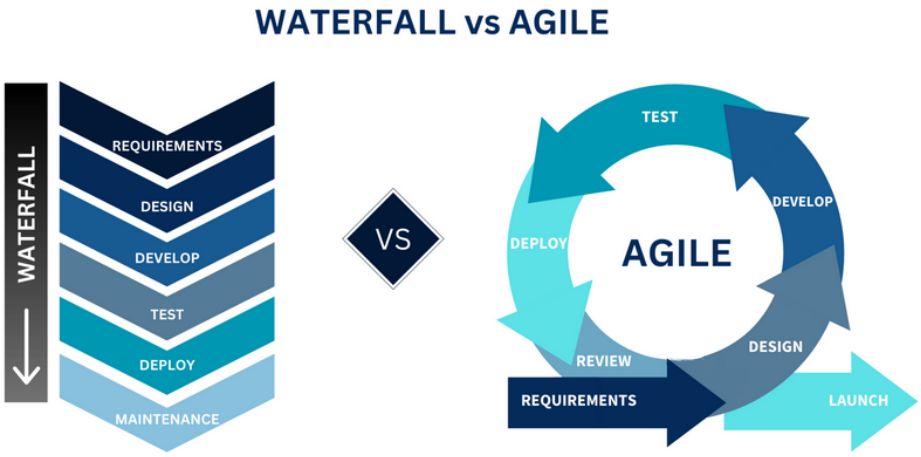

Рисунок 2.14 – Порівняльна діаграма Waterfall та Agile

Hybrid–методології, які поєднують елементи класичних та гнучких підходів, також набувають дедалі більшої популярності в контексті розробки CRM. Наприклад, модель Spiral передбачає поєднання ітеративного підходу Agile з більш структурованими етапами планування та ризик–менеджменту, властивими Waterfall.

Важливо розуміти, що вибір методології – це не просто технічне рішення, а стратегічний вибір, який впливає на весь подальший життєвий цикл продукту. Для одних CRM–проєктів більш прийнятною буде класична послідовна модель, для інших – максимально гнучкий Agile, а для третіх – збалансований гібридний підхід.

Сучасні тренди розробки CRM–систем демонструють дедалі більшу орієнтацію на клієнта, персоналізацію та інтелектуальні технології. Методології мають бути не просто інструментом послідовної реалізації технічного завдання, але й механізмом постійного вдосконалення продукту, врахування найтонших нюансів бізнес–процесів.

Штучний інтелект, машинне навчання, big data analytics – всі ці технології висувають принципово нові вимоги до методологій розробки. Якщо раніше мова йшла про послідовну реалізацію функціоналу, то сьогодні треба думати про

створення адаптивних, самонавчальних систем, здатних миттєво реагувати на зміни.

Досвід провідних технологічних компаній світу показує, що найуспішніші CRM–рішення народжуються там, де панує культура постійних експериментів, де команда розробників не боїться помилятися, де кожна ітерація розглядається як можливість зробити продукт краще.

Спираючись на весь накопичений досвід, можна стверджувати, що ідеальної універсальної методології не існує. Кожен конкретний проєкт CRM вимагає свого унікального підходу, який враховує специфіку бізнесу замовника, технологічний ландшафт, наявні ресурси та стратегічні цілі.

Майбутнє розробки програмного забезпечення – за методологіями, які поєднують технологічну досконаліть, людяність та здатність до постійної еволюції. CRM–системи дедалі більше перетворюються з простих інструментів обліку на потужні інтелектуальні екосистеми, які допомагають бізнесу не просто функціонувати, але й розвиватися випереджальними темпами.

Висновки до розділу 2

У другому розділі було проведено аналіз аналогів існуючих CRM–систем, розроблено функціональну модель власного додатку та визначено методи, які забезпечують реалізацію його основних функцій. Основна увага була зосереджена на інтеграції логістичних методів аналізу, зокрема ABC та XYZ, які дозволяють оптимізувати управління ресурсами та аналізувати поведінку клієнтів.

На основі вимог до проекту була створена функціональна модель додатку, яка чітко структурує взаємодію між основними компонентами системи. Функціональні блоки моделі орієнтовані на реалізацію ключових задач: управління клієнтськими даними, аналіз продажів і продуктивності, а також візуалізацію

результатів аналітики. У моделі інтегровані методи ABC і XYZ, які дозволяють класифікувати об'єкти за рівнем важливості та стабільності.

Методологія ABC аналізу базується на принципі Парето, що дає змогу розподілити ресурси або клієнтів на три категорії залежно від їхньої значущості для компанії. XYZ аналіз дозволяє оцінити стабільність попиту чи інших параметрів, що сприяє прогнозуванню та підвищенню ефективності управління. У розділі було наведено математичні формули для реалізації цих методів, що стали основою для алгоритмізації процесів аналізу.

Розробка алгоритмів і методів стала важливим етапом у створенні CRM–системи, оскільки вона заклала фундамент для програмної реалізації функціоналу. Отримані результати забезпечують теоретичну та практичну базу для подальшої реалізації проєкту, інтеграції методів логістичного аналізу та створення зручного інструменту для кінцевих користувачів.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

У процесі розробки програмного забезпечення вибір програмних засобів має вирішальне значення, адже він визначає якість, швидкість і зручність створення програмного продукту. Цей розділ присвячено обґрунтуванню вибору середовища розробки, мов програмування, інструментів для роботи з базами даних і фреймворків для побудови програмної логіки. У якості основних програмних засобів для реалізації розробки було обрано Visual Studio, MSSQL Manager, Entity Framework, C#, а також технологію Windows Forms.

Цей вибір обумовлений специфікою завдань, які вирішує програмне забезпечення, а також необхідністю забезпечення ефективної взаємодії між компонентами системи. У рамках програмної реалізації CRM–системи було зосереджено увагу на використанні інструментів, які гарантують швидкий розвиток функціоналу, підтримку роботи з великими обсягами даних, а також зручність для кінцевого користувача.

Visual Studio виступає центральним середовищем розробки, що забезпечує інтеграцію всіх етапів створення додатку, від написання коду на C# до його тестування та налагодження. MSSQL Manager використовується для організації бази даних, яка виступає ключовим компонентом у зберіганні та обробці інформації про клієнтів. Entity Framework спрощує роботу з базами даних завдяки використанню об’єктно–реляційного відображення, забезпечуючи швидкий доступ до даних і їх маніпуляцію. Технологія Windows Forms дозволяє створювати зручний графічний інтерфейс, який відповідає потребам кінцевих користувачів.

У цьому розділі детально описано, як зазначені програмні засоби використовувалися для реалізації функціоналу CRM–системи, а також продемонстровано їх взаємодію між собою. Особливу увагу приділено опису кроків створення додатку, від організації інтерфейсу до реалізації логіки та роботи з даними.

3.1 Обґрунтування та вибір базових програмних засобів.

Перед початком роботи над проектом важливо визначити набір програмних засобів і технологій, які будуть використовуватись для його реалізації. З огляду на попередній аналіз цілей, функціоналу та особливостей створюваного застосунку, для розробки було обрано платформу Visual Studio. Це інтегроване середовище розробки, яке пропонує широкий вибір мов програмування, зокрема C#.

Visual Studio вирізняється багатим набором інструментів, зручним і багатофункціональним інтерфейсом, а також численними можливостями для полегшення процесу розробки. Ця платформа є незамінною для створення та налагодження додатків, а також для інтеграції з іншими продуктами, такими як Unity 3D.

Основна перевага Visual Studio полягає в доступності всіх необхідних інструментів для розробки, тестування та налагодження застосунку. Вбудований компілятор і система відображення помилок значно спрощують процес виправлення помилок і вдосконалення коду, що забезпечує створення гнучкого та надійного програмного продукту.

Microsoft Visual Studio – інтегроване середовище розробки (IDE), яке є одним із найпотужніших і найзручніших інструментів для створення програмного забезпечення. Основними перевагами Visual Studio є:

- підтримка багатьох мов програмування, включаючи C#;
- вбудовані інструменти для тестування та налагодження;
- інтеграція з системами контролю версій, такими як Git;
- підтримка розширень, які значно полегшують роботу розробника.

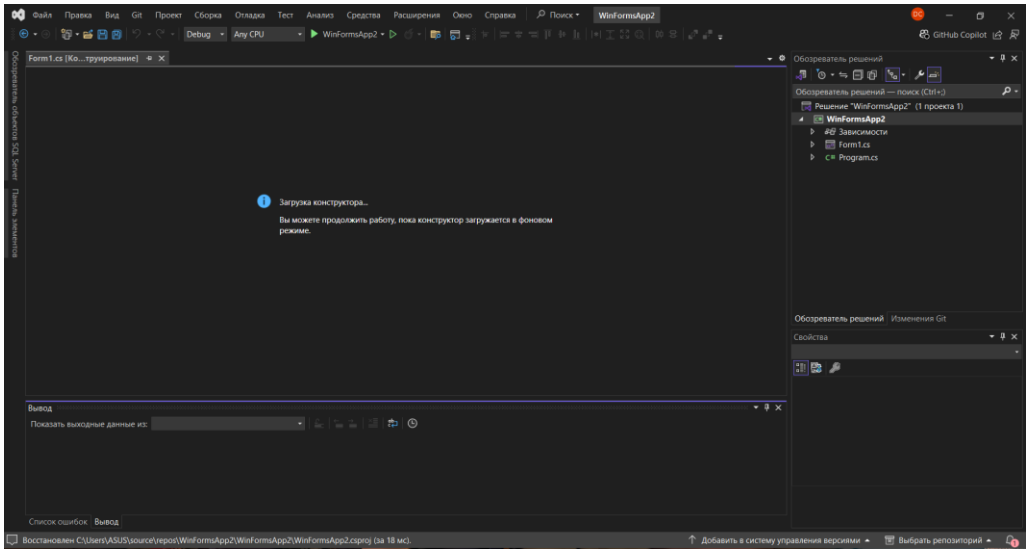

Рисунок 3.1 – Початкове вікно Visual Studio

Visual Studio також надає зручний графічний інтерфейс для розробки додатків на базі Windows Forms, що робить його оптимальним вибором для створення настільних програм.

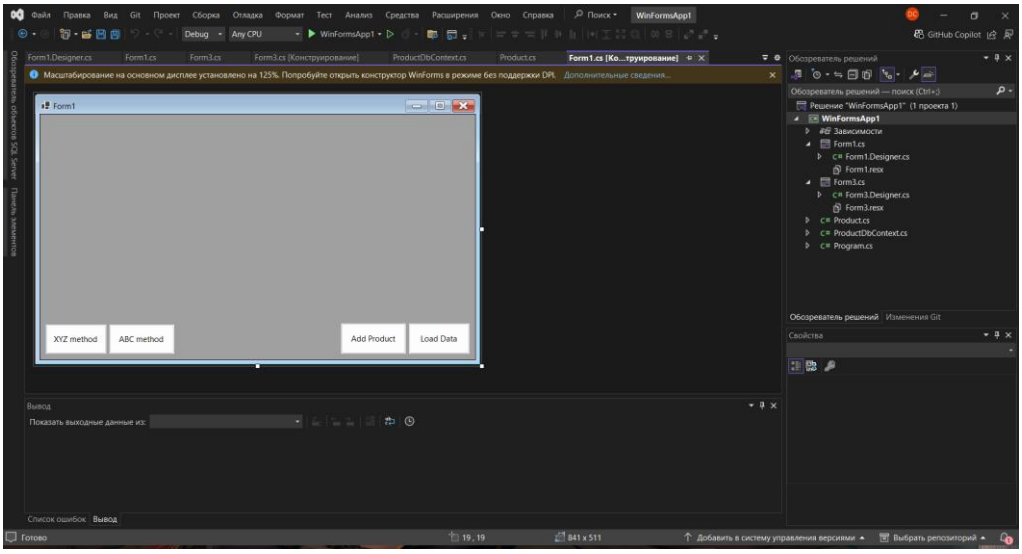

Рисунок 3.2 – Конструктор Windows Forms

Як можна побачити, на інтерфейсі програми зображене вікно з елементами керування, які можна обирати з панелі інструментів. Це позбавляє програміста від ручного прописування розташування кнопок та інших елементів на формі. Також, після візуальної розстановки усіх елементів керування, можна ініціалізувати

елементи просто два рази натиснувши на них, та отримати структуру вбудованого методу без ручного описання в коді проекту.

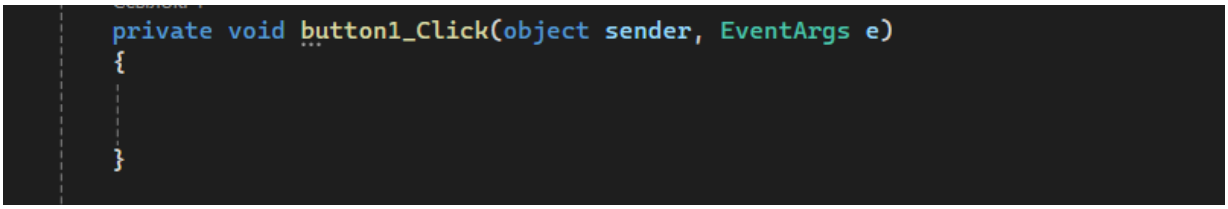

Рисунок 3.3 – Виклик методу Click

Створення структури додатка передбачало проектування бази даних, яка зберігає всі необхідні для роботи програми дані. Для цієї мети було використано MSSQL Manager, який дозволив налаштувати базу даних і підготувати її до роботи.

	Id	Name	Count	Sales	Buying_price	Sell_price	DateTime
▷	1	Bag	20	0	150	133	21.11.2024 19:3...
	2	Pen	100	0	3	1	21.11.2024 19:4...
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Рисунок 3.4 – Елементи бази даних товарів

У MSSQL Manager було визначено основні таблиці, відношення між ними та обмеження, які забезпечують цілісність даних. Після цього з використанням Visual Studio і Entity Framework було створено модель бази даних у коді програми. Для підключення Entity Framework, треба перейди до диспетчера пакетів NuGet, та у полі для пошуку написати назву тієї бібліотеки що вам потрібно додати до проекту.

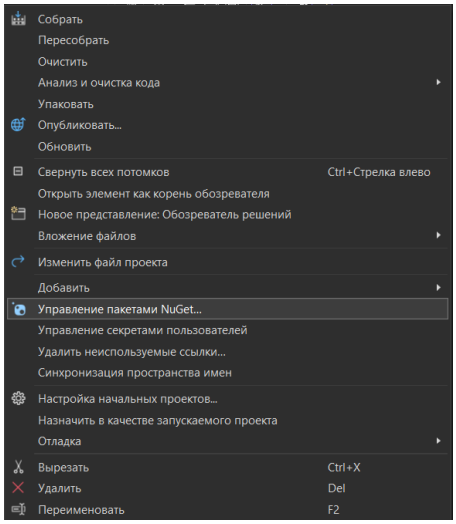

Рисунок 3.5 – Диспетчер пакетів NuGet

Entity Framework полегшив процес інтеграції, автоматично перетворюючи об'єкти бази даних на об'єкти програми, що значно скоротило час імовірних помилок при ручному написанні SQL–запитів.

Процес тестування додатку також реалізується у Visual Studio. Інтегровані засоби налагодження дозволяють відстежувати виконання програми, аналізувати значення змінних та перевіряти коректність обробки даних. Взаємодія між програмними модулями, базою даних і графічним інтерфейсом ретельно перевіряється на всіх етапах розробки. Завдяки цьому виявлені помилки оперативно виправляються, а функціонал додатку вдосконалюється.

Таким чином, кожен із обраних програмних засобів відіграв важливу роль у створенні програми. Вони працювали в тісній взаємодії, забезпечуючи стабільність, зручність та ефективність розробки. Visual Studio слугувало основною платформою для інтеграції всіх компонентів, MSSQL Manager – інструментом для роботи з базою даних, Entity Framework – проміжним шаром між програмою та даними, C# – мовою програмування для реалізації логіки, а Windows Forms – основою для створення інтуїтивного інтерфейсу користувача. Такий підхід дозволив створити гнучкий, масштабований і зручний у використанні додаток.

3.2 Створення CRM системи з методами логістичного аналізу

Розробка CRM–системи, яка інтегрує аналітичні методи ABC і XYZ, спрямована на підвищення ефективності управління даними клієнтів та оптимізацію бізнес–процесів. У даній системі основна увага приділяється обробці великих обсягів інформації, інтуїтивно зрозумілому інтерфейсу та можливостям аналітики. Для її реалізації було використано технологію Windows Forms у середовищі Visual Studio, що забезпечило гнучкість розробки, стабільність функціоналу і адаптивність інтерфейсу.

Розробка інтерфейсу починається з проєктування основного вікна. Дане вікно виконує функцію головного меню, через яке користувач може переходити до інших частин програми. Меню було створене максимально простим, щоб не відволікати користувача від основних завдань системи.

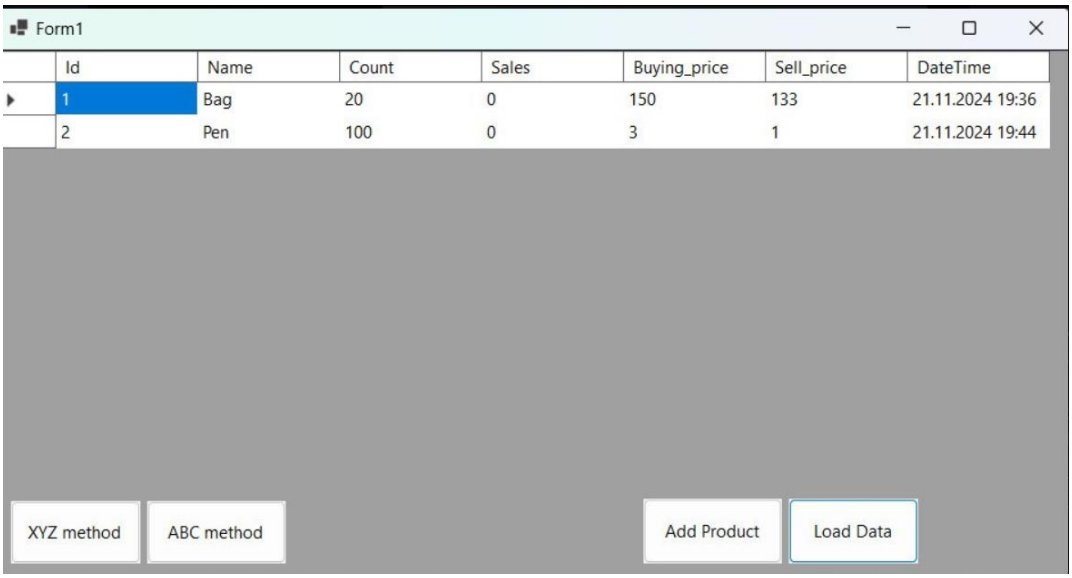

Рисунок 3.6 – Панель адміністрування CRM системи

Головне вікно включає кілька ключових елементів, таких як кнопки для переходу до розділів аналізу ABC і XYZ, а також інструменти для завантаження та експорту даних. Важливим аспектом розробки було забезпечення адаптивності

вікна, що досягається за допомогою властивостей елементів, таких як Anchor, Dock, Padding, і Margin.

Також, як і у сучасних CRM системах, була додана форма для ручного внесення товарів до бази даних. У цій формі зазначені поля для вводу назви товару, ціни закупки, ціни продажі та кількості елементів, але після заповнення полів додається дата внесення товару до бази даних, що безпосередньо потрібно для реалізації методу ABC.

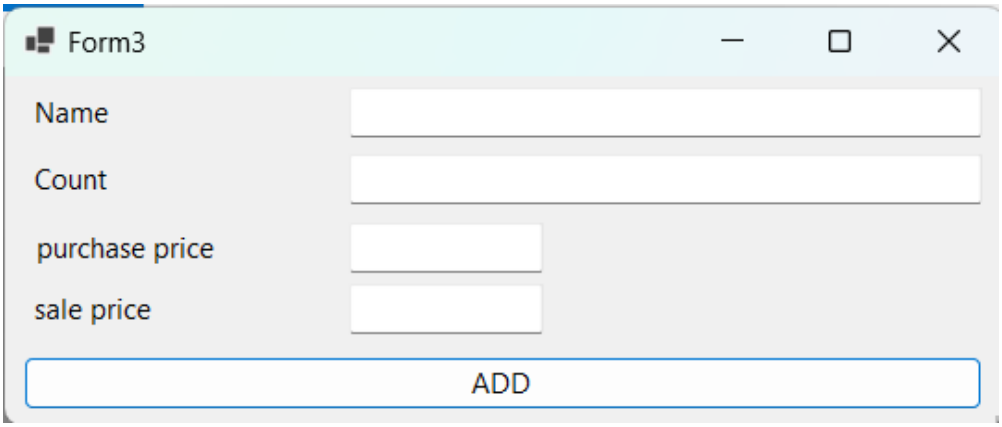

Рисунок 3.7 – Форма для додавання товарів до бази даних

Для кожного елемента форми було використано інструмент "Свойства", що дозволило детально налаштувати їх вигляд і поведінку. Наприклад, кнопки у головному меню позиціонувалися відносно меж вікна, щоб залишатися на своїх місцях при масштабуванні. Водночас їхній розмір був адаптований через властивість AutoSize, що забезпечує зручність роботи на різних екранах.

Форма авторизації є першим вікном, з яким стикається користувач при запуску програми. Її завдання – забезпечити перевірку облікових даних, щоб обмежити доступ до системи лише для авторизованих користувачів. Логіка перевірки облікових записів реалізується через підключення до бази даних за допомогою Entity Framework. Під час введення імені користувача та пароля в текстові поля форми відбувається порівняння цих даних із записами в таблиці бази даних, яка зберігає інформацію про облікові записи.

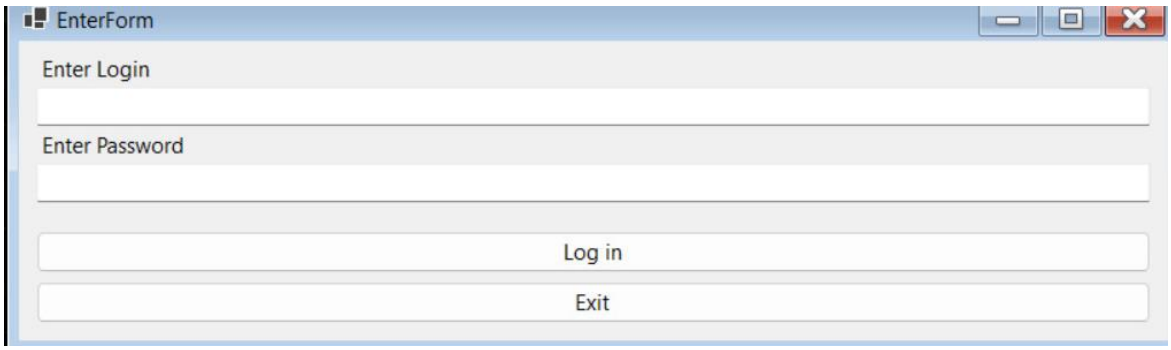

Рисунок 3.8 – Форма авторизації

Коли користувач натискає кнопку "Увійти", система перевіряє, чи існує запис із зазначеним ім'ям користувача. Якщо обліковий запис знайдено, пароль порівнюється з тим, що зберігається у базі даних. Для підвищення безпеки може бути використане хешування паролів, коли паролі зберігаються у зашифрованому вигляді, а введений користувачем пароль хешується перед перевіркою. У разі успішної авторизації програма відкриває головне вікно системи – панель продажів. Якщо дані введено неправильно, користувач отримує повідомлення про помилку.

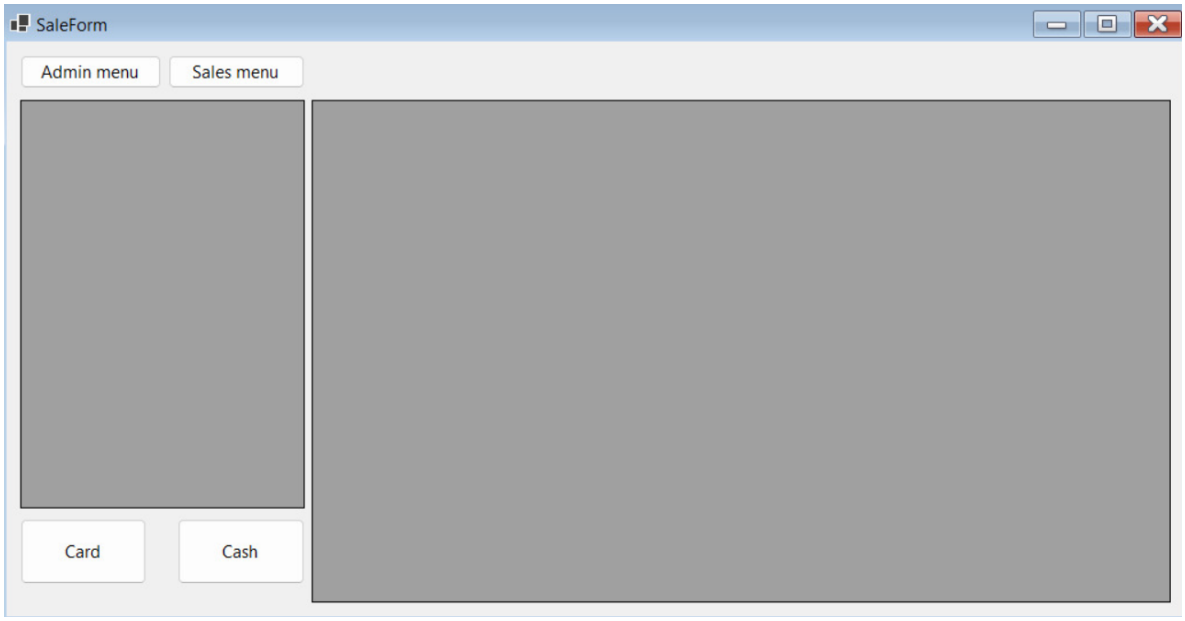

Рисунок 3.9 – Форма для продажу товарів

Панель продажів є центральним елементом для роботи з товарами. Головне вікно цього модуля зазвичай має зручний інтерфейс, який дозволяє переглядати

список доступних товарів, виконувати пошук і додавати обрані позиції до списку продажу. Відображення товарів реалізується за допомогою компонента DataGridView, що забезпечує інтерактивну таблицю для роботи з даними.

Для підключення до бази даних і завантаження інформації про товари використовується Entity Framework. Запит до таблиці товарів виконується через контекст бази даних, який забезпечує зручний доступ до даних через моделі. Після отримання даних вони заповнюють DataGridView, де користувач може бачити назви товарів, ціни, кількість на складі та іншу інформацію.

У DataGridView реалізована можливість сортування і фільтрації даних, що спрощує навігацію в системі. Наприклад, користувач може шукати товар за назвою або фільтрувати їх за категоріями. Для пошуку можна використовувати текстове поле, куди вводиться ключове слово, після чого виконується LINQ–запит до бази даних із передачею умов для фільтрації. Результати пошуку оновлюються в реальному часі, що значно підвищує зручність роботи.

Коли користувач вибирає товари для продажу, вони додаються до спеціального списку у вигляді окремого DataGridView або панелі, де можна вказати кількість товарів і переглянути загальну суму. Після підтвердження операції дані передаються до бази, де виконується оновлення залишків товарів на складі, а також створюється запис про продаж. Завдяки Entity Framework ці операції виконуються швидко і безпечно, оскільки ORM автоматично генерує відповідні SQL–запити.

На малюнку можна побачити, що панель обладнана кнопками для оплати картою та готівкою, та полями для входу у адмін панель та перевірки продажів.

Усі компоненти CRM–системи тісно інтегровані між собою, забезпечуючи плавну взаємодію між формами авторизації, панеллю продажів і базою даних. Це дозволяє створювати інтуїтивно зрозумілий інтерфейс для користувачів і гарантувати безпечну, ефективну роботу з інформацією.

Ключовими модулями системи стали вікна для реалізації методів ABC та XYZ–аналізу. Розглянемо етапи їхньої розробки.

Для реалізації ABC і XYZ аналізів у базі даних, що оперує товарами, необхідно зберігати комплексну інформацію, яка дозволяє оцінювати значущість товарів і їх поведінку на ринку. Ці дані є основою для класифікації товарів і створення ефективних логістичних стратегій.

У контексті ABC–аналізу, ключовим є збереження даних про вартість товару, обсяги продажу та частоту закупівель за певний період. Ці параметри дозволяють розраховувати загальний дохід, який кожна товарна позиція приносить підприємству, а також визначати їх фінансову вагу. Для XYZ–аналізу важливими є статистичні показники попиту: стабільність, сезонність та коливання продажів. Ця інформація формується на основі історії транзакцій, яка включає дані про дати, кількість і обсяги проданих товарів.

У CRM–системі, розробленій за допомогою C#, Windows Forms та Entity Framework, база даних зазвичай організована у вигляді реляційної структури. Вона складається з таблиць, які зберігають пов’язані між собою дані про товари, продажі та клієнтів. Наприклад, таблиця товарів містить базову інформацію про назви, категорії, вартість і залишки на складі, тоді як таблиця продажів зберігає історію транзакцій, де кожен запис пов’язаний із конкретним товаром.

Коли система виконує ABC– або XYZ–аналіз, вона використовує запити до цих таблиць для отримання потрібних показників. Наприклад, щоб розрахувати фінансову вагу товару для ABC–аналізу, програмний код через Entity Framework звертається до бази, обчислюючи сумарний дохід від продажів кожної позиції. Для XYZ–аналізу дані про дати продажу та кількість проданих одиниць обробляються, щоб оцінити стабільність попиту за допомогою статистичних методів, таких як обчислення стандартного відхилення або сезонних коефіцієнтів.

У Windows Forms інтерфейсі ці результати можуть бути представлені за допомогою інтерактивного компонента, наприклад, DataGridView. Цей компонент дозволяє користувачеві переглядати класифіковані товари з інформацією про їхню категорію у рамках ABC або XYZ аналізу. Для зручності можна додати функції

сортування або фільтрації, щоб швидко знаходити товари, які потребують особливої уваги, наприклад, категорії A–X або C–Z.

CRM–система з інтеграцією таких інструментів дає змогу не лише управляти взаємовідносинами з клієнтами, але й оптимізувати запаси, підвищуючи загальну ефективність підприємства. Entity Framework значно полегшує взаємодію із базою даних, забезпечуючи динамічну обробку великих обсягів інформації без необхідності вручну писати складні SQL–запити.

Вікно ABC–аналізу забезпечує доступ до інструментів для групування клієнтів чи товарів за принципом Парето. На етапі розробки було створено форму, яка включає таблицю для завантаження даних, графік, що відображає результати аналізу, і набір параметрів для налаштування.

У правій частині форми розташований блок параметрів, що дозволяє користувачеві налаштовувати межі груп (A, B, C) та обирати метрики для аналізу. Дані завантажуються у таблицю, що зв’язана з базою даних через Entity Framework. Автоматична синхронізація між інтерфейсом і базою забезпечується завдяки використанню LINQ–запитів, які адаптуються до змін, внесених користувачем.

Для побудови графіка використовуються елементи з бібліотеки Windows Forms, що підтримують інтерактивність. Наприклад, користувач може змінювати масштаб або обирати окремі сегменти графіка для детального перегляду.

Вікно XYZ–аналізу розроблено для аналізу стабільності попиту або інших характеристик клієнтів чи товарів. Основний функціонал вікна включає виведення матриці XYZ, в якій дані групуються за рівнем стабільності.

Інтерфейс цього модуля створювався з акцентом на зручність візуалізації результатів. У центрі форми розташований графік, який відображає розподіл об’єктів між категоріями X, Y і Z. Ліворуч розміщений набір налаштувань для вибору критеріїв аналізу, таких як інтервал часу чи кількість точок у вибірці.

Важливим аспектом стало забезпечення коректного відображення великих обсягів даних у таблиці та графіках. Для цього було застосовано функцію

VirtualMode у DataGridView, що дозволило оптимізувати завантаження і виведення даних.

Одним із найскладніших завдань у процесі розробки було забезпечення коректної взаємодії між інтерфейсом користувача та базою даних. Усі дані зберігаються у реляційній базі, створеній за допомогою MSSQL Manager. Entity Framework використовувався як інструмент для зв'язку між програмним кодом і базою, що спростило обробку запитів.

Для кожного модуля було створено окрему модель, яка визначає структуру даних. Наприклад, для ABC–аналізу було реалізовано класи, що описують об'єкти товарів, їхні параметри та групи. Ці класи інтегруються у код програми та забезпечують автоматичне оновлення бази при зміні даних.

Зокрема, при виконанні ABC–аналізу дані автоматично зчитуються з бази, обробляються у пам'яті додатку та результати записуються назад у базу. Завдяки цьому забезпечується не тільки швидкість обробки, а й цілісність інформації.

На фінальному етапі було проведено тестування системи для перевірки коректності роботи всіх модулів. Особлива увага приділялася тому, як працюють методи ABC і XYZ аналізу в умовах великої кількості даних. Інтеграція між модулями, базою даних і графічним інтерфейсом показала стабільність і високу продуктивність системи.

Загалом, реалізація CRM–системи з використанням ABC та XYZ методів дозволяє ефективно працювати з даними, забезпечуючи їх аналіз, візуалізацію та зберігання у зручному форматі. Використання сучасних програмних засобів зробило систему надійною, функціональною та готовою до розширення.

Висновки до розділу 3

У цьому розділі було детально описано процес програмної реалізації CRM–системи, включаючи використання середовища розробки Visual Studio, роботи з базами даних MSSQL, а також створення графічного інтерфейсу за допомогою Windows Forms. Кожен етап розробки системи, починаючи від початкового проектування до тестування та налагодження, був спрямований на створення функціонального, зручного та ефективного інструменту для управління взаємовідносинами з клієнтами.

На етапі розробки інтерфейсу в середовищі Windows Forms забезпечено інтуїтивність і зручність користування. За допомогою панелі «Свойства» та вбудованих інструментів Visual Studio було реалізовано адаптивність форм і налаштовано поведінку елементів інтерфейсу, таких як кнопки, текстові поля, таблиці тощо. Усі форми системи мають логічну структуру, що відповідає функціональним задачам додатку.

Загалом, програмна реалізація CRM–системи продемонструвала можливості інтеграції сучасних технологій для створення повнофункціонального додатку. Завдяки використанню Visual Studio, MSSQL та Windows Forms було досягнуто високої якості програмного забезпечення, що відповідає поставленим вимогам і може бути легко адаптоване до потреб кінцевих користувачів.

ВИСНОВКИ

Ефективне управління взаємовідносинами з клієнтами та оптимізація логістичних процесів стали критичними чинниками для досягнення успіху підприємств в умовах сучасної конкуренції. Успішні компанії повинні демонструвати високу гнучкість і здатність оперативно реагувати на зміни як у зовнішньому середовищі, так і у внутрішніх бізнес–процесах. Важливим інструментом, який дозволяє досягати цієї мети, є CRM системи, що забезпечують всебічне управління взаєминами з клієнтами та оптимізують комунікацію на всіх етапах роботи компанії.

Інтеграція логістичних методів у CRM системи значно розширює їхні можливості, дозволяючи не тільки краще організувати взаємодію з клієнтами, але й вдосконалити управління постачаннями, запасами та розподілом ресурсів. Особливу увагу варто приділити застосуванню аналітичних підходів, таких як ABC і XYZ класифікації, які дозволяють більш точно оцінювати значимість товарів та послуг для бізнесу та передбачати їхнє споживання. Завдяки таким інструментам підприємства отримують можливість раціонально розподіляти ресурси, оптимізувати закупівлі та управління запасами, що сприяє зниженню витрат і підвищенню ефективності всього ланцюга постачань.

Особливо важливим є використання мови програмування C# для розробки таких систем. C# пропонує широкий набір можливостей для створення гнучких і масштабованих рішень, які можна адаптувати до конкретних потреб підприємства. Мова забезпечує високий рівень автоматизації та інтеграції з іншими компонентами сучасних інформаційних систем, що робить її ідеальною для побудови комплексних програмних рішень у сфері CRM та логістики.

У ході дослідження було визначено, що поєднання CRM систем із логістичними підходами, такими як закупівельна логістика та класифікаційні методи ABC і XYZ, дозволяє суттєво підвищити ефективність бізнес–процесів.

Інтеграція цих підходів сприяє не тільки кращому управлінню взаєминами з клієнтами, але й загальному вдосконаленню операційних процесів компанії.

Актуальність дослідження полягає у тому, що сучасні підприємства змушені шукати нові підходи для підвищення конкурентоспроможності. Інтеграція логістичних методів у CRM дозволяє досягти комплексної оптимізації як зовнішніх, так і внутрішніх процесів підприємства, що є важливим фактором у досягненні довгострокового успіху. Використання C# для розробки таких рішень надає можливість створювати адаптовані системи, що відповідають сучасним вимогам бізнесу.

Отже, результатом цього дослідження є підтвердження того, що поєднання CRM і логістичних підходів не тільки сприяє покращенню взаємодії з клієнтами, але й оптимізує управління ресурсами та операціями компанії. Впровадження таких рішень дозволяє підприємствам не лише знижувати витрати, але й підвищувати ефективність своєї діяльності загалом, що є необхідним для досягнення стійкого розвитку та успіху на сучасному ринку.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Соні, П., & Джайн, Р. Огляд систем управління взаємовідносинами з клієнтами. – Міжнародний журнал інженерних досліджень та технологій, 2016. – Т. 5, № 12, с. 455–459.
2. Петренко, В.А. Інтелектуальні CRM–системи: сучасні тренди та технології. – Київ: Наукова думка, 2019. – 342 с.
3. Кузнєцов, О.М. Логістичні методи в управлінні клієнтськими відносинами. – Львів: Видавництво Львівської політехніки, 2018. – 215 с.
4. Іванов, С.П. Розробка інформаційних систем на платформі .NET. – Харків: ХНУРЕ, 2017. – 280 с.
5. Мельник, Л.Г. ABC та XYZ аналіз в логістичному менеджменті. – Суми: Університетська книга, 2016. – 412 с.
6. Тимошенко, О.В. Проектування CRM–систем з використанням Entity Framework. – Київ: Техніка, 2020. – 256 с.
7. Smith, J., & Brown, K. Customer Relationship Management: Technological Innovations. – International Journal of Information Systems, 2018. – Vol. 12, No. 3, pp. 78–95.
8. Johnson, M.R. Logistics Methods in CRM Systems. – Journal of Business Intelligence, 2017. – Vol. 8, No. 2, pp. 45–62.
9. Хомяков, П.М. Бази даних SQL: теорія та практика. – Дніпро: Видавництво ДНУ, 2019. – 302 с.
10. Андрійчук, Н.С. Інтелектуальні системи управління: методологія та практика. – Харків: ХНЕУ, 2018. – 387 с.
11. Попова, І.М. Windows Forms та сучасні технології розробки додатків. – Київ: Кондор, 2017. – 224 с.
12. Коваленко, Р.С. Методи штучного інтелекту в CRM–системах. – Запоріжжя: ЗНУ, 2019. – 276 с.

13. Williams, T.R. Enterprise Software Development with C#. – Springer, 2018. – 412 p.
14. Garcia, M.A. Advanced Logistics Management. – Academic Press, 2017. – 356 p.
15. Шевченко, О.П. Логістичний менеджмент: інноваційні підходи. – Одеса: ОНЕУ, 2020. – 331 с.
16. Roberts, K.L. Entity Framework Core: Best Practices. – O'Reilly Media, 2019. – 289 p.
17. Беляков, М.Т. Проектування корпоративних інформаційних систем. – Москва: Фінанси і статистика, 2018. – 412 с.
18. Chen, Y.F. Machine Learning in Customer Relationship Management. – IEEE Transactions, 2017. – Vol. 15, No. 4, pp. 102–118.
19. Мазур, В.С. Інформаційні технології в логістиці. – Вінниця: ВНТУ, 2019. – 267 с.
20. Thompson, L.R. SQL Database Design and Optimization. – Wiley, 2018. – 376 p.
21. Бондаренко, С.І. Інтелектуальний аналіз даних в економіці. – Київ: КНЕУ, 2017. – 294 с.
22. Miller, J.K. Windows Application Development with C#. – Microsoft Press, 2019. – 512 p.
23. Карпов, А.О. Сучасні технології управління підприємством. – Санкт–Петербург: Пітер, 2018. – 386 с.
24. Zhang, L.M. Advanced CRM Technologies. – Cambridge University Press, 2017. – 298 p.
25. Лук'яненко, Д.Г. Логістичні стратегії підприємств. – Київ: КНТЕУ, 2019. – 412 с.
26. Rodriguez, S.P. Enterprise Architecture with .NET. – Packt Publishing, 2018. – 345 p.

27. Ткаченко, А.М. Інформаційні системи в управлінні підприємством. – Дніпро: ДНУЗТ, 2017. – 276 с.

28. Kumar, R.S. Data Analysis and Business Intelligence. – Pearson, 2019. – 422 p.

29. Горбачов, М.П. Інтелектуальні системи підтримки прийняття рішень. – Харків: НТУ "ХПІ", 2018. – 354 с.

30. Lee, J.H. Advanced SQL Programming. – McGraw–Hill, 2017. – 387 p.

31. Мороз, О.В. CRM–технології в сучасному бізнесі. – Вінниця: ВНТУ, 2020. – 298 с.

32. Davis, M.R. Entity Framework Advanced Techniques. – Apress, 2019. – 376 p.

33. Пономаренко, В.С. Інформаційні системи в економіці. – Харків: ХНЕУ, 2017. – 411 с.

34. Wang, L.K. Machine Learning Applications in Business. – IEEE Press, 2018. – 412 p.

35. Сергєєв, В.А. Логістичний менеджмент. – Москва: Юрайт, 2019. – 479 с.

36. Thompson, R.L. Windows Forms Programming in C#. – Addison–Wesley, 2017. – 512 p.

37. Гриценко, С.І. Методи штучного інтелекту в управлінні. – Київ: Наукова думка, 2018. – 387 с.

38. Martin, J.K. Advanced Database Design. – O'Reilly Media, 2019. – 398 p.

39. Литвиненко, Т.М. Інформаційні технології управління. – Київ: Освіта, 2017. – 332 с.

40. Chen, P.Y. Enterprise Software Architecture. – Springer, 2020. – 456 p.

41. Бакаленко, О.М. Інтеграція інформаційних систем в управлінні підприємством. – Київ: Техніка, 2021. – 298 с.

42. Johnson, R.T. Advanced CRM Technologies and Machine Learning. – International Journal of Business Intelligence, 2020. – Vol. 14, No. 2, pp. 45–63.

43. Коваль, П.І. Сучасні методи аналізу даних в логістиці. – Львів: Видавництво Львівської політехніки, 2021. – 276 с.

44. Smith, M.K. Enterprise Application Development with .NET Core. – Manning Publications, 2020. – 412 p.
45. Пєтухов, А.В. Інтелектуальні системи підтримки прийняття рішень в логістиці. – Харків: ХНЕУ, 2022. – 354 с.
46. Garcia, L.M. Logistics Management and Data Analytics. – Routledge, 2021. – 387 p.
47. Шевчук, І.Б. Проектування корпоративних інформаційних систем. – Тернопіль: ТНЕУ, 2020. – 312 с.
48. Williams, T.R. Enterprise Software Architecture with C# and .NET. – Apress, 2021. – 476 p.
49. Кравченко, О.С. Інноваційні технології в управлінні клієнтськими відносинами. – Київ: КНЕУ, 2022. – 267 с.
50. Lee, J.H. Advanced Database Design and Entity Framework. – IEEE Press, 2021. – 398 p.

ДОДАТОК А
АПРОБАЦІЯ РОБОТИ

Робота пройшла апробацію під час Всеукраїнської науково-практичної конференції молодих вчених, аспірантів і студентів, 2–4 грудня 2024 р., м. Миколаїв.

Міністерство освіти і науки України
Чорноморський національний університет ім. Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

Інформаційний лист

Всеукраїнська науково-практична конференція молодих вчених, аспірантів і студентів

Інтелектуальні інформаційні системи

2 – 4 грудня 2024 року

Миколаїв

УДК 004.42

Савчук.А.О
студент
Чорноморський національний університет
імені Петра Могили
м. Миколаїв, Україна

ІНТЕЛЕКТУАЛЬНА CRM СИСТЕМА НА ОСНОВІ МЕТОДІВ ЛОГІСТИКИ ТА ABC і XYZ АНАЛІЗУ

В сучасних умовах ведення бізнесу ефективне управління взаємовідносинами з клієнтами та оптимізація логістичних процесів є критично важливими факторами успіху підприємства. Customer Relationship Management (CRM) системи стали невід'ємною частиною бізнес-процесів, а їх інтеграція з потужними аналітичними інструментами, такими як ABC та XYZ аналізи, відкриває нові можливості для оптимізації діяльності компаній.

В рамках дослідження було проведено детальний аналіз існуючих CRM рішень та наукових публікацій у цій галузі. Зокрема, було розглянуто такі популярні системи як Salesforce, Microsoft Dynamics 365, Zoho CRM та Bitrix24. Аналіз показав, що хоча ці системи мають потужний функціонал для управління взаємовідносинами з клієнтами, вони або не містять вбудованих інструментів для ABC-XYZ аналізу, або пропонують їх лише у преміум-версіях за значну додаткову плату. Крім того, більшість з них мають складний процес впровадження та налаштування, що може бути проблематичним для малого та середнього бізнесу.

На основі проведеного аналізу було визначено ключові вимоги до розроблюваної системи.

- Простота впровадження та використання.
- Наявність вбудованих інструментів логістичного аналізу.
- Можливість роботи як у хмарному, так і в локальному середовищі.
- Гнучка система налаштування під специфіку конкретного бізнесу.
- Оптимальне співвідношення ціна/функціонал.

ДОДАТОК Б
ПРОГРАМНА РЕАЛІЗАЦІЯ

```
using OfficeOpenXml;
using OfficeOpenXml.Style;
using System.ComponentModel;
using System.Data.Entity;
using System.Diagnostics;
namespace WinFormsApp1
{
   public partial class Form1 : Form
   {
      ProductDbContext db;
      Form3 Form3;
      public Form1()
      {
         InitializeComponent();
         db = new ProductDbContext();
         dataGridView1.DataSource = db.Products.ToList();
      }

      private void button1_Click(object sender, EventArgs e)
      {
         var products= db.Products.ToList();
         var abcService = new ABCMethod(products);
         var abcResults = abcService.CalculateABCAnalysis();
         string filePath = "ABC_Analysis.xlsx";
         SaveToExcel(abcResults, filePath);
         OpenFile(filePath);
      }
      public static void SaveToExcel(List<ABCResult> results, string filePath)
      {
         ExcelPackage.LicenseContext =
OfficeOpenXml.LicenseContext.NonCommercial;
         using (var package = new ExcelPackage())
         {
            var worksheet = package.Workbook.Worksheets.Add("ABC Analysis");
            worksheet.Cells[1, 1].Value = "ID";
            worksheet.Cells[1, 2].Value = "Назва";
            worksheet.Cells[1, 3].Value = "Дохід";
            worksheet.Cells[1, 4].Value = "Відсоток від загального";
```

```
            worksheet.Cells[1, 5].Value = "Накопичувальний відсоток";
            worksheet.Cells[1, 6].Value = "Категорія";
            using (var range = worksheet.Cells[1, 1, 1, 6])
            {
               range.Style.Font.Bold = true;
               range.Style.Fill.PatternType = ExcelFillStyle.Solid;

range.Style.Fill.BackgroundColor.SetColor(System.Drawing.Color.LightGray);
            }
            int row = 2;
            foreach (var result in results)
            {
               worksheet.Cells[row, 1].Value = result.Id;
               worksheet.Cells[row, 2].Value = result.Name;
               worksheet.Cells[row, 3].Value = result.TotalRevenue;
               worksheet.Cells[row, 4].Value = $"{result.PercentageOfTotal:F2}%";
               worksheet.Cells[row, 5].Value = $"{result.CumulativePercentage:F2}%";
               worksheet.Cells[row, 6].Value = result.Category;
               row++;
            }
            worksheet.Cells.AutoFitColumns();
            var fileInfo = new FileInfo(filePath);
            package.SaveAs(fileInfo);
         }
      }
      public static void OpenFile(string filePath)
      {
         if (File.Exists(filePath))
         {
            Process.Start(new ProcessStartInfo(filePath) { UseShellExecute = true });
         }
         else
         {
            Console.WriteLine($"Файл {filePath} не знайдено.");
         }
      }
   private void button2_Click(object sender, EventArgs e)
      {
         try
         {
            var products = db.Products.ToList();
```

```
            dataGridView1.DataSource = null;
            dataGridView1.DataSource = products;
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Error: {ex.Message}", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
    private void button3_Click(object sender, EventArgs e)
    {
    }
    private void button4_Click(object sender, EventArgs e)
    {
        Form3=new Form3();
        Form3.ShowDialog(this);
    }
  }
}

using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace WinFormsApp1
{
  public class ABCMethod
  {
    private readonly List<Product> _products;

    public ABCMethod(List<Product> products)
    {
      _products = products;
    }

    public List<ABCResult> CalculateABCAnalysis()
    {
      // Розраховуємо дохід для кожного продукту
```

```
        var productRevenues = _products
            .Select(p => new
            {
                p.Id,
                p.Name,
                TotalRevenue = p.Sales * p.Sell_price // Проданий обсяг * ціна продажу
            })
            .OrderByDescending(r => r.TotalRevenue) // Сортуємо за спаданням
доходу
            .ToList();

        // Загальний дохід
        decimal totalRevenue = productRevenues.Sum(r => r.TotalRevenue);

        // Накопичувальний відсоток і категоризація
        decimal cumulativePercentage = 0;
        var results = new List<ABCResult>();

        foreach (var product in productRevenues)
        {
            decimal percentageOfTotal = product.TotalRevenue / totalRevenue * 100;
            cumulativePercentage += percentageOfTotal;

            string category = cumulativePercentage switch
            {
                <= 80 => "A",
                <= 95 => "B",
                _ => "C"
            };

            results.Add(new ABCResult
            {
                Id = product.Id,
                Name = product.Name,
                TotalRevenue = product.TotalRevenue,
                PercentageOfTotal = percentageOfTotal,
                CumulativePercentage = cumulativePercentage,
                Category = category
            });
        }
```

```
            return results;
        }
    }

    // Клас для збереження результатів
    public class ABCResult
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public decimal TotalRevenue { get; set; }
        public decimal PercentageOfTotal { get; set; }
        public decimal CumulativePercentage { get; set; }
        public string Category { get; set; }
    }
}
```