

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

ІНСТИТУТ ІНФОРМАТИКИ ТА РАДІОЕЛЕКТРОНІКИ
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК І ТЕХНОЛОГІЙ

(повне найменування інституту, факультету)

КАФЕДРА СИСТЕМНОГО АНАЛІЗУ ТА ОБЧИСЛЮВАЛЬНОЇ
МАТЕМАТИКИ

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавра

(ступінь вищої освіти)

на тему Інформаційна система оптимізації роботи з клієнтами на базі
приватного підприємства в сфері ІТ

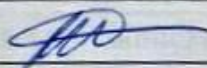
Виконав: студент(ка) 4 курсу, групи КНТ-818сп

Спеціальності 124 – Системний аналіз

(код і найменування спеціальності)

Освітня програма (спеціалізація)

«Інтелектуальні технології та прийняття рішень
в складних системах»



Іванів О.А.

(прізвище та ініціали)

Керівник



Рябенко А.Є.

(прізвище та ініціали)

Рецензент



Пирожок А.В.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет Інформатики та радіоелектроніки, комп'ютерних наук і технологій

Кафедра Системного аналізу та обчислювальної математики

Ступінь вищої освіти бакалавр

Спеціальність 124 – Системний аналіз
(код і найменування)

Освітня програма

(спеціалізація) «Інтелектуальні технології та прийняття рішень в складних системах»

(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри

Г.В. Корніч
«08» червня 2021 року

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Іваніва Олександра Андрійовича

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Інформаційна система оптимізації роботи з клієнтами на базі приватного підприємства в сфері ІТ

керівник проєкту (роботи) Рябенко Антон Євгенович, к.ф.-м.н., доц.,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від «26» травня 2021 року № 215

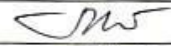
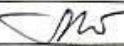
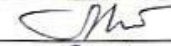
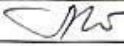
2. Строк подання студентом проєкту (роботи) «08» червня 2021 року

3. Вихідні дані до проєкту (роботи) Умови завдання, перелік літературних джерел за темою дослідження.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) В першому розділі проводиться аналіз предметної області та розглядаються різні поняття на тему диплома. В другому розділі розглядаються різні засоби реалізації програми. В третьому розділі показана програмна реалізація.

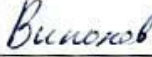
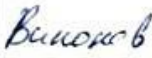
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1	Рябенко А.Є., к.ф.-м.н., доц.		
2	Рябенко А.Є., к.ф.-м.н., доц.		
3	Рябенко А.Є., к.ф.-м.н., доц.		
Нормаконтроль	Ширококорд Д.В., к.ф.-м.н., ст. викл.		

7. Дата видачі завдання « 27 » жовтня 2020 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Сформулювати мету та основні завдання дипломної роботи	27.10.2020 – 03.11.2020	
2	Проаналізувати предметну область	03.11.2020 – 27.11.2020	
3	Проведено аналіз для вибору засобів реалізації програми	27.11.2020 – 03.02.2021	
4	Розроблена програмна реалізація	08.03.2021 – 30.04.2021	
5	Оформлення пояснювальної записки	30.04.2021 – 24.05.2021	
6	Попередній захист дипломної роботи та отримання рецензій.	25.05.2021 – 07.06.2021	
7	Захист дипломної роботи.	08.06.2021	

Студент

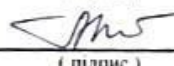


(підпис)

Іванів О.А.

(прізвище та ініціали)

Керівник проєкту (роботи)



(підпис)

Рябенко А.Є.

(прізвище та ініціали)

РЕФЕРАТ

Дипломна робота: 59 с., 12 рис., 1 дод., 17 літературних джерел.

Об'єкт дослідження – принципи і методи побудови інформаційних систем.

Предмет дослідження – представлений аналізом методів і інструментів побудови інформаційних систем і їх використання для розроблення інформаційної системи оптимізації роботи з клієнтами на базі приватного підприємства в сфері ІТ..

Мета роботи – проектування і розроблення інформаційної системи оптимізації роботи з клієнтами на базі приватного підприємства в сфері ІТ.

Методи дослідження – дати характеристику предметної галузі, обрати засоби реалізації, реалізувати інформаційну систему.

В дипломній роботі був проведений аналіз предметної області, також був проведений аналіз вибору мови програмування та засобів реалізації програми. Розроблено програмну реалізацію інформаційної системи для підприємства.

ПРЕДМЕТНА ГАЛУЗЬ, ІНФОРМАЦІЙНА СИСТЕМА, ЗАСОБИ РЕАЛІЗАЦІЇ, ДІАГРАМА, МОВА ПРОГРАМУВАННЯ C#, БАЗА ДАНИХ, СЕРЕДОВИЩА РОЗРОБКИ, СУБД, ІТ-КОМПАНІЯ , АНАЛІЗ РЕЗУЛЬТАТІВ

ЗМІСТ

Завдання	Ошибка! Закладка не определена.
Реферат	4
Вступ.....	6
Розділ 1. Аналіз предметної області.....	7
1.1 ІТ-компанія.....	7
1.2 Інформаційна система	11
1.3 Десктопні застосунки	17
1.4 Тайм-менеджмент.....	25
Розділ 2. Вибір засобів реалізації	27
2.1 Вибір мови програмування.....	27
2.2 Вибір середовища розробки	32
2.3 Вибір СУБД.....	33
Розділ 3. Програмна реалізація	39
3.1 Діаграма варіантів використання.....	39
3.2 Діаграма класів програмного продукту	40
3.3 Структурна діаграма.....	42
3.4 Графічний інтерфейс користувача.....	43
3.5 Керівництво користувача.....	48
3.6 Аналіз отриманих результатів.....	48
Висновки	50
Перелік посилань.....	51
Додаток А. Лістинг програмного коду	53

ВСТУП

В сучасному світі, в якому кожного дня люди створюють терабайти різних даних, щосекундно постає необхідність зберігати, категоризувати, редагувати та переглядати ці дані. Важко уявити собі підприємство, громадську або державну установу, яка не має необхідності у зберіганні великої кількості документації, інформації щодо клієнтів, часу та місць заходів тощо.

В такий важкий, з інформаційної точки зору, час на допомогу людині приходить поняття баз даних. Базы даних покликані спростити роботу з великою кількістю інформації, дати можливість швидко і зрозуміло отримувати з великих архівів даних потрібну інформацію за критеріями.

Виходячи з усього вищесказаного — важко недооцінити актуальність розробки різноманітних інформаційних систем на основі баз даних в різних предметних областях.

Для досягнення поставленої мети слід виконати наступні завдання:

- Дати характеристику предметної галузі.
 - Проаналізувати поняття ІТ-компанії та інформаційної системи.
 - Проаналізувати поняття десктопних додатків.
- Обрати засоби реалізації.
 - Провести огляд мови програмування.
 - Провести огляд системи управління базами даних.
 - Провести огляд середовища розробки.
- Реалізувати інформаційну систему.
 - Створити діаграму варіантів використання.
 - Створити структурну діаграму та діаграму класів.
 - Створити графічний інтерфейс користувача та написати інструкцію користувача.

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 ІТ-компанія

Комп'ютери та інформаційні технології (ІТ) – це використання комп'ютерів для зберігання, отримання, передачі та управління даними чи інформацією. ІТ, як правило, використовується в контексті ділової діяльності, а не особистої або розважальної технології [1]. ІТ визначається в розділі Інформаційно-комунікаційні технології (ІКТ). Системи інформаційних технологій (ІТ-системи), як правило, використовуються невеликою групою користувачів ІТ для інформаційних систем, інформаційних систем зв'язку або комп'ютерних систем, включаючи, наприклад, все обладнання, програмне забезпечення та периферію.

Людство зберігало, отримувало, контролювало і передавало інформацію, оскільки вона була написана месопотамськими шумерами близько 3000 р. до н. е. [1]. Однак термін інформаційні технології в сучасному розумінні вперше з'явився в копії, опублікованій у 1958 році. Автори Гарольд Дж. Левітт і Томас Л. Віслер сказали: «Нова технологія ще не має єдиної усталеної назви. Ми називаємо це Інформаційні технології (ІТ)». Це визначається у трьох категоріях: технічна упаковка, прийняття рішень статистичними та математичними рішеннями.

Цей термін можна використовувати як взаємозамінний з комп'ютерами та комп'ютерними мережами, але він також включає інші технології передачі інформації, такі як телевізори та телефони. Іншим продуктом, який обслуговує економіку, пов'язану з інформаційними технологіями та комп'ютерним обладнанням, є програма mg[4].

Механізм Антикітери бере свій початок на початку першого століття до нашої ери і вважається найвідомішим комп'ютером аналогових машин з новітніми механізмами та редукторами [4]. Вперше такі коробки з'явилися в

Європі в 16 столітті, і до 1645 року перші механічні калькулятори, здатні обробляти чотири основні арифметичні операції.

Електронні комп'ютери з поворотними клапанами почали з'являтися на початку 1940-х рр. Завершений у 1941 р. електронний Zuse Z3 був першим портативним комп'ютером, і його пропонували стати однією з перших машин за сучасними стандартами. Можна налаштувати цифровий електронний комп'ютер для декодування німецьких повідомлень, але він не призначений для виконання однієї функції. Він також не мав можливості підтримувати свій бізнес. Програмне забезпечення в пам'яті – це програмне забезпечення було створено шляхом заміни внутрішньої проводки на вставні, напрямні та перемикачі. [6] Дуже відомий електронний комп'ютер, що зберігає програмне забезпечення «Manchester Baby», червень 1948 р. Перше програмне забезпечення було створено 21-го [6].

Розробка транзисторів в 1940-х і 1940-х роках дозволила створити нове покоління комп'ютерів з меншим енергоспоживанням. Перший комерційно доступний комп'ютер та програмне забезпечення, що зберігається, Ferranti Mark I, мав 4050 клапанів і споживав 25 кВт.

Деякі інші досягнення в галузі напівпровідникових технологій включають інтегральні схеми (IC), розроблені Джеком Кілбі з Техаської документації та Робертом Нойсом з Fairchild Semiconductor в 1959 р., та окислення напівпровідників, розроблене Camp Labs в 195. включає транзистори з фізичним полем (MOSFET). А Тед Хофф, Федеріко Фейджин, Масатосі Сіма та Стенлі Мазоре працювали в Intel у 1971 році. Цей важливий прорив призвів до розвитку персональних комп'ютерів (ПК) та інформативно-комунікаційних технологій (ІКТ) у 1970-х роках [6].

Залежно від використовуваної технології зберігання та обробки ви можете призначити чотири різні фази розвитку ІТ: механічну та електронну [1]. Ця стаття зосереджена на останньому (електронному) періоді.

Технологічні компанії – це електронні компанії, що включають підприємства, пов'язані з послугами, пов'язаними з Інтернетом, такими як

цифрова електроніка, програмне забезпечення та послуги електронної комерції [1-3].

За словами Luck, топ-10 таких технологічних компаній, як 2020, – це Apple Inc., Samsung, Foxconn, Alphabet Inc, Microsoft, Huawei, Dell Technologies, Hitachi, IBM і Sony [2]. Amazon вигідніший за Apple, але його встановлює Luck у роздрібній торгівлі. Найвигіднішими у списку 2020 року є Apple Inc., Microsoft, Alphabet Inc, Intel, Facebook, Samsung та Tencent [2].

Найчастіше називають Apple Inc., Alphabet Inc., Facebook, Microsoft, Amazon.com, Inc. – найбільшими американськими міжнародними технологічними компаніями. Ці п'ять технологічних партнерств керують основними функціональними можливостями, електронною комерцією та інформаційними каналами у всій веб-системі. Станом на 2017 рік загальна вартість Великої п'ятірки становила понад 3,3 мільярда доларів і становить більше 40% від значення індексу Nasdaq 100 [3].

Багато великих технологічних компаній мають репутацію інновацій і щороку витрачають багато грошей на дослідження та розробки. Згідно з рейтингом PwC Global Innovation 1000 Ratings за 2017 рік, технологічні компанії утворюють дев'ять з 20 найбільш інноваційних компаній у світі, за ними слідує Amazon, за нею Alphabet Inc., і один з найбільших користувачів R & D. Одним з них є Intel.

У той час як у багатьох технологічних компаній та технічних компаній є офіси, розташовані близько один від одного, у різних частинах світу зростає кілька технічних дисциплін. Сюди входять Силіконовий басейн біля затоки Сан-Франциско, Силіконовий док у Дубліні, Силіконова гора в Остіні та Тех-Сіті в Лондоні. Міжнародний технологічний парк в Бангалорі, Чжунгуанкун, Пекін, місто цифрових медіа в Сеулі.

Інформаційні технології (ІТ) та високотехнологічні підприємства є частиною цілого ряду технологічних підприємств.

Компанії з інформаційних технологій часто називають «технологічною галузю» або «технологічною галузю». Ці назви можуть ввести в оману і не

повинні розглядатися як «технічні проекти». Зазвичай це великі компанії, які продають технології та програмне забезпечення для споживачів. Також зауважте, що з галузевої точки зору IT-відділи, як правило, є «центрами витрат». Центр витрат – це відділ або персонал, який несе витрати або «витрати» в межах бізнесу, а не потік прибутку чи доходів. Сучасний бізнес у своїй повсякденній діяльності значною мірою покладається на технологію, а вартість покриття технології, яка робить її більш ефективною, часто вважається «вартістю ведення бізнесу». IT-відділи повинні отримувати фінансування для вищого керівництва та прагнути досягти бажаних результатів у рамках цього бюджету. Варіанти фінансування можуть відрізнятися між урядом та приватним сектором, але принципи однакові або менше. Часто це невідомо для прямого інтересу до автоматизованого штучного інтелекту, але постійний тиск на більш-менш відкриває двері для контролю хоча б невеликих транзакцій у великих корпораціях. Він відкриється автоматично.

Зараз у багатьох компаніях є IT-відділи, які керують комп'ютерами, мережами та іншими технічними сферами бізнесу. Компанії також намагаються інтегрувати IT з результатами бізнесу та приймати рішення з BizOps та бізнес-індустрією.

У контексті бізнесу Американська асоціація інформаційних технологій визначила інформаційні технології як «дослідження, будівництво, розробку, використання, впровадження, підтримку або управління комп'ютерними інформаційними системами». Ці особисті обов'язки полягають у роботі на місцях, включаючи управління мережею, розробку та встановлення програмного забезпечення, планування та управління життєвим циклом технологій організацій, які зберігають, модернізують та замінюють обладнання та програмне забезпечення.

1.2 Інформаційна система

Інформаційна система (ІС) – це система, призначена для збору, зберігання, встановлення та розповсюдження інформації. З соціальної та технічної точки зору ІС має чотири пристрої: ролі, люди, структури, технології. ІС можна визначити як надання інформації, сприяння знанням та інтеграцію обладнання для збору, зберігання та обробки даних, що використовується в цифрових продуктах [1].

Комп'ютерна інформаційна система – це система, що включає людину та комп'ютер, які обробляють або інтерпретують інформацію [2]. Цей термін може також стосуватися лише комп'ютерних систем та встановленого програмного забезпечення.

Інформаційні системи – це вивчення систем, які мають конкретні посилення на інформацію та додаткові мережі обладнання та програмного забезпечення, які приватні особи та організації використовують для збору, фільтрування, обробки, створення та розповсюдження даних. Основна увага приділяється інформаційним системам з вищезазначеними межами, користувачам, процесорам, сховищам, введенням, виходам та комунікаційним мережам [4].

У багатьох організаціях відділ або підрозділ, відповідальний за інформаційні системи та обробку даних, відомий як «інформаційна служба» [5].

Певні інформаційні системи призначені для підтримки операцій, управління та прийняття рішень [6]. Інформаційні системи – це інформаційно-комунікаційні технології (ІКТ), що використовуються організаціями, і є способом, яким люди використовують цю технологію для підтримки своїх бізнес-процесів [8].

Деякі автори чітко розмежовують інформаційні системи, комп'ютерні системи та бізнес-процеси. Інформаційні системи, як правило, мають

компоненти ІКТ, але це стосується не тільки ІКТ, але й кінцевого використання інформаційних технологій. Інформаційні системи також відрізняються від бізнес-процесів. Інформаційні системи допомагають контролювати ефективність бізнес-процесів [9].

Альтер заявляє про переваги розгляду інформаційної системи як особливого типу робочої системи. Робоча система – це система, в якій люди та машини використовують ресурси для здійснення процесів та дій, щоб забезпечити споживачів конкретними продуктами та послугами. Інформаційна система – це функціональна система, призначена для збору, розподілу, зберігання, пошуку, контролю та відображення інформації [8].

Таким чином, інформаційні системи інтегруються із системами даних, з одного боку, та з активними системами, з іншого. [10] Інформаційні системи – це тип комунікаційної системи, в якій дані представляються та обробляються як форма соціальної пам'яті. Інформаційні системи також можна розглядати як неформальні мови, які підтримують прийняття рішень та поведінку людини.

Інформаційні системи є найважливішим полем для читання інформації про організацію [11].

Гроші та інші. (1995) пропонує два погляди на ІС, включаючи програми, пристрої, дані, людей та процеси [11]. Чжен надав інший погляд на інформаційні системи. Це також покращує системні процеси та ключові елементи, такі як контекст, межі, цілі та взаємодія.

Асоціація комп'ютерів чітко дає зрозуміти, що «спеціалісти з інформаційних систем зосереджуються на інтеграції інформаційних технологічних рішень та бізнес-процесів для задоволення інформаційних потреб бізнесу та інших підприємств» [11].

Існують різні типи інформаційних систем, такі як системи обробки закупівель, системи підтримки прийняття рішень, системи управління знаннями, системи управління навчанням, системи управління базами даних та офісні інформаційні системи. В основі багатьох інформаційних систем лежить інформаційна технологія, яка, як правило, розроблена для того, щоб

люди могли виконувати завдання, які людський мозок не може виконувати, такі як обробка великих обсягів інформації, складні розрахунки та керування багатьма одночасними процесами.

Інформаційні технології – це дуже важливий і корисний ресурс, доступний адміністраторам. Багато компаній заснували посаду директора з питань інформації (ІТ). Головний інформаційний директор (CIO) сидить разом з головним виконавчим директором (CEO), фінансовим директором (CFO), головним операційним директором (COO) та головним технічним директором (CTO) виконавчого комітету. Технічний директор може також бути ІТ-директором і навпаки. Директор з інформаційної безпеки (CISO) зосереджується на управлінні інформаційною безпекою.

Шість елементів, які необхідно інтегрувати для формування інформаційної системи, є:

1. Обладнання: Тимчасове обладнання відноситься до механічного обладнання. У сучасних інформаційних системах цей пристрій має власний комп'ютер та всі аксесуари. До допоміжного обладнання належать обладнання для імпорту / експорту, обладнання для зберігання та обладнання зв'язку. Комп'ютерні інформаційні системи дозволяють пристроям містити книги та чорнило.

2. Програмне забезпечення: Термін програмне забезпечення позначає комп'ютерні програми та рекомендації (якщо такі є), які його підтримують. Комп'ютерні програми – це машиночитані інструкції, які посилають цикли на апаратне забезпечення системи для отримання важливої інформації з даних. Зазвичай програмне забезпечення зберігається на спеціальних носіях вводу-виводу, часто на дисках або стрічках. Комп'ютерне програмне забезпечення «програмне забезпечення» перед комп'ютером включало, як користуватися пристроєм (наприклад, заголовки стовпців книги) та як ним користуватися (посібник з каталогу карт).

3. Статистика: Статистика – це факти, які система використовує для отримання важливої інформації. У сучасних інформаційних системах дані, як

правило, зберігаються на диску або на стрічці, доки комп'ютер цього не вимагає. Комп'ютерні інформаційні системи зазвичай зберігають дані у зручному для читання форматі.

4. Процедура: Процедура – це політика, яка керує функціонуванням інформаційної системи. «Процедура для людей полягає в тому, що програмне забезпечення є апаратним забезпеченням» – загальна аналогія, що використовується для пояснення ролі процесів у системі.

5. Люди: У кожній системі люди повинні бути важливими. Часто найбільше ігнорують тих, хто найбільше впливає на успіх чи провал інформаційної системи. Сюди входять не лише користувачі, але й ті, хто управляє комп'ютерами та обслуговує їх, підтримує дані та підтримує комп'ютерні мережі.

6. Зворотній зв'язок: Це ще один компонент ІР, який визначає, чи може ІР надати зворотний зв'язок (хоча цей компонент не потрібний для роботи).

«Класичний» погляд на інформаційні системи, знайдений у підручниках у 1980-х роках, - це піраміда систем, що демонструє корпоративне лідерство, як правило, система обробки закупівель у нижній частині піраміди, за якою слідує система управління інформацією та підтримка рішень. Система продовжувалась. Закінчується вищевказаною інформаційною системою управління. Моделі пірамід були важливі з моменту їх першого створення, але було розроблено кілька нових технологій і з'явилися нові інформаційні системи. Деякі з них нелегко вписуються в оригінальну модель піраміди.

Кілька прикладів такої системи:

- Система підтримки прийняття рішень
- Система соціальних медіа
- Процес управління процесом
- Система управління інформацією
- Психічна система

- Система підприємства
- Інформаційне сховище
- Планування бізнес-ресурсів
- Комп'ютерна платформа
- Експертна система
- Пошукова система
- Геоінформаційна система
- Глобальна інформаційна система
- Медіаінформаційна система тійка
- Автоматизація офісу.

Комп'ютерна інформаційна система – це в основному ІВ, яка використовує комп'ютерні технології для виконання деяких або всіх запланованих завдань. Основними компонентами комп'ютерної інформаційної системи є:

- Пристрій – це периферійний пристрій, який працює в тандемі.
- Програмне забезпечення – це програма або набір програм, що дозволяє пристрою обробляти дані.
- База даних – це структуровані дані, схожі на таблицю.
- Інтернет – це система спільного використання, яка дозволяє вам ділитися ресурсами з іншими комп'ютерами.
- Процедура – це процедура, яка поєднує пристрої для обробки інформації та отримання бажаних результатів.

Перші чотири компоненти (апаратне забезпечення, програмне забезпечення, база даних та мережа) утворюють так звану платформу інформаційних технологій. ІТ-спеціалісти можуть використовувати ці інструменти для створення інформаційних систем, які контролюють заходи безпеки, ризики та управління даними. Ці практики відомі як послуги інформаційних технологій.

Деякі інформаційні системи підтримують частини організації, деякі підтримують цілі організації, а деякі підтримують організаційну групу. Майте на увазі, що кожен відділ або робоча область у вашій організації має власну колекцію програм або інформаційних систем. Ці інформаційні системи на робочому місці (FAIS) є основою стандартної IC, яка є системою та панеллю бізнес-аналітики. Як впливає з назви, кожен FAIS підтримує певні функції у вашій організації, такі як бухгалтерський облік IC, фінансування IC, управління виробничими операціями (ПОМ), продаж IC та людські ресурси. У сфері фінансів та бухгалтерського обліку менеджери використовують IT-системи для прогнозування доходів та операцій бізнесу, визначення найкращих джерел доходів та витрат, проведення аудитів для забезпечення точності організації, а також усіх фінансових звітів та документації. Інші типи організаційних інформаційних систем включають FAIS, системи обробки закупівель, планування ділових ресурсів, автоматизовані офісні системи, інформаційні системи управління, системи підтримки прийняття рішень, експертні системи, інформаційні панелі управління, системи управління ланцюгами поставок та системи електронної комерції. Інформаційні панелі - це особливий тип IP, який підтримує всіх менеджерів у вашій організації. Він забезпечує миттєвий доступ до інформації вчасно і прямий доступ до інформації, організованої за допомогою звітів. Експертні системи прагнуть імітувати роботу професіоналів, використовуючи потенціал міркувань, знань та досвіду в певній галузі.

Відділи інформаційних технологій великих корпорацій мають значний вплив на розробку, використання та застосування інформаційних технологій у своєму бізнесі. Для полегшення та використання інформаційних систем може бути використано кілька методів та процесів. Багато розробників використовують системні інженерні системи, такі як Система сталого розвитку (SDLC) для розробки інформаційних системних систем. Фазами життєвого циклу розробки системи є планування, аналіз системи та потреб, структура, розробка, інтеграція та тестування, впровадження та експлуатація

та обслуговування. Мета останніх досліджень - забезпечити та виміряти спільний розвиток таких систем в організації та в усіх відносинах. Інформаційні системи можна розробляти вручну (у межах вашої організації) або експортувати. Цього можна досягти, екпортуючи частину або всю систему. Окремим випадком є географічний розподіл команд розробників (шокуючі глобальні інформаційні системи).

Комп'ютерна інформаційна система, що відповідає визначенню Лангефорса, є технічно реалізованим середовищем для:

- Запис, зберігання та розповсюдження мовних виразів
- Також висновки з подібних фраз.

Геоінформаційні системи, інформаційні системи Землі та інформаційні системи про катастрофи є прикладами нових інформаційних систем, але їх можна широко розглядати як просторові інформаційні системи. Розробка системи здійснюється на наступних етапах.

- Діагностуйте та протестуйте проблеми.
- Збір інформації.
- Класифікація нових системних вимог.
- Проектування системи.
- Побудова системи.
- Впровадження системи.
- Моніторинг та обслуговування.

1.3 Десктопні застосунки

Прикладне програмне забезпечення (аббревіатура для використання) - це комп'ютерна програма, призначена для виконання інших функцій, крім функцій самого комп'ютера, як правило, для кінцевого користування.

Прикладами програмного забезпечення є програмне забезпечення для обробки текстів та медіаплеєри. Додаток Joint Nouns зазвичай застосовується до всіх програм. Іншою важливою класифікацією програмного забезпечення є системне програмне забезпечення, пов'язане з продуктивністю та використанням комп'ютера (організація) [1].

Програмне забезпечення може бути інтегровано в комп'ютер із власним системним програмним забезпеченням або надруковано окремо та внесено до списку власних відкритих джерел чи проектів [1]. Додатки, створені для мобільних платформ, називаються мобільними додатками.

В інформаційних технологіях програмне забезпечення (додатки), прикладне програмне забезпечення або програмне забезпечення – це комп'ютерна програма, призначена для допомоги людям у виконанні певних завдань. Додаток може контролювати текст, цифри, звуки, зображення та комбінації цих елементів, залежно від вашої діяльності. Деякі програмні пакети орієнтовані на одну функцію, таку як обробка текстів. Інші, звані інтегрованим програмним забезпеченням, мають кілька програм.

Створене користувачем програмне забезпечення модифікує систему відповідно до потреб конкретного користувача. Створене користувачем програмне забезпечення включає шаблони електронних таблиць, макроси обробки текстів, наукові пояснення, звуки, зображення та анімовані документи. Навіть фільтри електронної пошти є свого роду додатком для користувачів. Користувачі часто самі створюють цю програму і не усвідомлюють її важливості.

Однак різниця між системним програмуванням, таким як операційна система, та програмним забезпеченням є неправильним і може бути суперечливим [3]. Одне з найважливіших питань щодо «Америки проти». Неексклюзивним законом компанії Microsoft було те, чи є браузер Microsoft Internet Explorer частиною операційної системи Windows або іншої програми. Як інший приклад, суперечка щодо імені GNU / Linux частково спричинена розбіжностями щодо зв'язку між ядром Linux та вбудованою в нього

операційною системою. У деяких встановлених системах програмне забезпечення та програмне забезпечення операційної системи не можуть розрізнити користувачів, наприклад, програмне забезпечення, що використовується для управління відеомагнітофонами, DVD-програвачами або мікрохвильовими печами. Наведене вище визначення може виключати інші програми, які розміщені на великих комп'ютерах у великих організаціях.

Слово «додаток», що вживається як прикметник, не зменшує або не стосується значення «додаток» [3]. Наприклад, такі поняття, як програмні модулі (API), сервери програмного забезпечення, оновлення програмного забезпечення та автоматизоване використання системи. Застосовуване програмне забезпечення стосується всіх комп'ютерних програм, а не лише програм.

Деякі програми доступні на різних версіях платформи. Інший працює лише з одним, наприклад, Microsoft Windows Geography App, Android Learning App або Linux Game. Можливо, з'явиться нове та популярне програмне забезпечення, яке працює лише на одній платформі, що збільшує потребу в цій платформі. Його називають вбивцею або вбивцею. Наприклад, VisiCalc був першим і останнім додатком Apple, який допомагав продавати нові персональні комп'ютери в офісі. Для Blackberry це була програма електронної пошти.

В останні роки «використання» (створене до 1981 р. [4]) стало поширеним як аббревіатура для використання мобільних пристроїв, таких як смартфони та планшети .. Донедавна також використовувалася спрощена версія комп'ютерного програмного забезпечення.

Існують різні способи та альтернативи класифікації програмного забезпечення.

З юридичної точки зору, програми визначаються за допомогою чорних ящиків для прав кінцевого користувача або учасника (з використанням проміжних критеріїв реєстрації та множинних внесків).

Програмні додатки також класифікуються за мовою програмування, на якій пишеться або виконується вихідний номер, а також за метою та результатом.

За правами власності та користування

Прикладне програмне забезпечення, як правило, поділяється на два основні класи: програмне забезпечення з відкритим кодом, програмне забезпечення з відкритим кодом та безкоштовне програмне забезпечення.

На захищене авторським правом програмне забезпечення поширюються виключні авторські права, а ліцензії на програми надають обмежені права на використання. Явний принцип стверджує, що програма «може бути відкритою для розширень, а не для змін». Такі запити можуть бути додані до інших.

Безкоштовне програмне забезпечення з відкритим кодом потрібно запускати, розповсюджувати, продавати або розширювати з будь-якою метою, і його слід явно модифікувати або видалити.

Додатки програмного забезпечення FOSS, що випускаються за безкоштовною ліцензією, можуть бути постійними та безкоштовними. Власник, власник або будь-хто інший, хто користується цим правом (авторське право, торгова марка, авторське право чи інша частина права власності), має право додавати до умов ліцензії дискримінацію, обмеження, затримки чи дати закінчення терміну дії. Може.

Програмне забезпечення публічного надбання має форму FOSS, є відкритим або захищеним, може запускатися, розповсюджуватися, модифікуватися, модифікуватися, публікуватися або створюватися похідно, не порушуючи авторські права, і тому може бути скасовано. Ви можете його продати, але ми не передаватимемо загальнодоступне майно будь-якому іншому приватному сектору. Публічно доступне програмне забезпечення може надаватися відповідно до діючої (неліцензійної) програми ліцензування,

яка застосовується до цих умов необмежено (протягом усього життя або назавжди).

За допомогою мови кодування

З часу розвитку та використання Інтернету між веб-додатками, написаними на HTML, JavaScript та іншими фірмовими технологіями, спостерігаються суттєві відмінності. Зазвичай потрібні веб-доступ та запуск браузера. І традиційні рідні програми, написані будь-якою мовою, які можна використовувати на певних типах комп'ютерів. У комп'ютерному співтоваристві існує суперечка з багатьох причин, особливо щодо мобільних пристроїв, таких як смартфони та планшети, щодо використання Інтернету для заміни традиційного програмного забезпечення. Використання Інтернету стало дуже поширеним для різних цілей, але переваги програмного забезпечення рідко втрачаються відразу. Крім того, вони можуть зустрічатися [8-10] і навіть включати.

За призначенням і результатом

Програмування програмного забезпечення також можна вважати горизонтальним або вертикальним [11-12]. Збалансовані програми є найпоширенішими та найпоширенішими, оскільки вони є загальними цілями, такими як текстові процесори та бази даних. Вертикальні додатки - це нішеві продукти, розроблені для певного виду галузі, бізнесу чи відділу у вашій організації. Інтегрований програмний пакет спрямований на вирішення всіх можливих аспектів виробництва, банківської справи, бухгалтерського обліку, обслуговування споживачів тощо.

Існує багато типів програмних програм [6].

- Спектр програм включає широкий спектр програм, що входять до комплексу. Зазвичай вони ведуть діалог із суміжними службами, функціями та

користувачами та можуть взаємодіяти з ними, наприклад, відкривати окремі файли. Ділові програми часто розміщують у кімнатах. Microsoft Office, LibreOffice та iWork включають текстові процесори, електронні таблиці та ін. Зв'яжіться; однак, люкси існують для інших цілей, таких як фотографія та музика.

Бізнес-програмне забезпечення часто відповідає потребам організацій у процесах та потоках даних у різних підрозділах у великих розподілених середовищах. Приклади включають системи планування ресурсів, системи управління взаємовідносинами з клієнтами (CRM) та програмне забезпечення для управління ланцюгами поставок. Відомча програма - це невеликий тип ділової програми, орієнтований на невелику організацію чи групу на велику організацію. (Приклади включають управління витратами на поїздки та бюро підтримки IT)

- Програма інфраструктури бізнесу надає загальні послуги, необхідні для підтримки ділової програми. (Приклади включають бази даних, сервери електронної пошти, мережі та системи управління безпекою.)

Application Platform as a Service (aPaaS) - це служба хмарних обчислень, яка забезпечує розробку та впровадження служб додатків.

- Програмне забезпечення інформаційного працівника дозволяє користувачам створювати та управляти інформацією про окремі проекти всередині відділу, на відміну від управління бізнесом. Прикладами є управління часом, управління ресурсами, аналітика, інструменти співпраці та документація. Текстові процесори, електронні таблиці, електронні листи, клієнти блогів, системи персональної інформації та редактори персональних засобів масової інформації допомагають інформаційним працівникам виконувати різноманітні обов'язки.

- Програмне забезпечення для доступу до вмісту в основному використовується для доступу до вмісту без редагування, але воно може також включати програмне забезпечення, яке дозволяє редагувати вміст. Такі

програми задовольняють потреби окремих людей та груп у використанні цифрових розваг та публікації цифрового вмісту. (Приклади включають медіаплеєри, веб-браузери та асистенти.)

- Освітні програми пов'язані з програмами доступу до вмісту, але містять вміст або послуги, які були модифіковані для використання викладачами чи студентами. Це може включати, наприклад, оцінку (тестування), виявлення значного прогресу та можливості співпраці.

Імітаційне програмне забезпечення імітує фізичні або уявні системи для дослідницьких, навчальних чи розважальних цілей.

- Програмне забезпечення для розробки засобів масової інформації часто створює друковані та електронні носії, які можуть використовуватись іншими в комерційному чи освітньому середовищі. Сюди входять програмне забезпечення для фотографій, настільне видавниче програмне забезпечення, програмне забезпечення для розробки мультимедіа, редактори HTML, редактори цифрової анімації, цифрові пісні та відео пісні.

- Програмне забезпечення для розробки продуктів використовується для виробництва апаратних та програмних продуктів. Сюди входять автоматизовані формати (CAD), автоматизовані формати (CAD), засоби редагування та редагування комп'ютерної мови, інтегровані середовища розробки та інтеграція програмного забезпечення.

- Розважальне програмне забезпечення може включати відеоігри, знімки екрана, фільми чи записану музику та інші розваги, які ви можете відчувати на своєму комп'ютері.

Додатки також можна класифікувати за комп'ютерною платформою, наприклад, за певною операційною системою, мережами презентацій, такими як хмарні програми або програми Web 2.0, та пристроями розгортання, такими як мобільні програмні додатки [13].

Саму операційну систему можна вважати програмним забезпеченням при виконанні простих операцій, таких як обчислення, вимірювання,

візуалізація та обробка слів, які не використовуються для керування пристроями за допомогою командного рядка або графічного інтерфейсу користувача. Сюди не входить програмне забезпечення, яке є частиною операційної системи, наприклад обчислювальне програмне забезпечення та текстові редактори.

Розробка програмного забезпечення – це процес проектування, очищення, проектування, програмування, архівування, тестування та виправлення помилок, пов'язаних із розробкою та обслуговуванням програм, систем чи інших компонентів програмного забезпечення. Розробка програмного забезпечення - це процес написання та підтримання вихідного коду, але в більш широкому розумінні він включає все: від передбачуваної структури програмування до остаточної розробки програмного забезпечення, іноді в процесі, запланованому цією структурою. Отже, розробка програмного забезпечення включає дослідження, нові розробки, створення прототипів, структурування, повторне використання, вдосконалення, обслуговування чи інші заходи, результатом яких є програмні продукти [1].

Додаток має три загальні цілі: задовольнити конкретні потреби клієнтів / бізнесу (для спеціального програмного забезпечення) та задовольнити конкретні потреби ряду потенційних користувачів (для комерційного програмного забезпечення). Його можна розробити для різних цілей. І з відкритим кодом)), або для особистого користування (наприклад, вчені можуть створювати програмне забезпечення для своїх повсякденних операцій). Розробка прошивки, тобто розробка прошивки, що використовується для контролю споживчих товарів, вимагає інтеграції контрольованого процесу розробки продукту та виробництва. Системне програмне забезпечення є ядром програмного забезпечення і самого програмного процесу і часто відкладається.

Дисципліна програмного забезпечення народилася через необхідність кращого контролю за процесом розробки програмного забезпечення.

Призначено використання системного підходу, представленого в інженерній концепції процесу розробки програмного забезпечення.

Проект існує багато способів керувати проектом. Він відомий як модель життєвого циклу розробки програмного забезпечення, техніка, процес або модель. Стил ь водоспаду є традиційною версією, на відміну від нещодавніх інновацій у галузі простої розробки програмного забезпечення [14].

1.4 Тайм-менеджмент

Управління часом – це процес ретельного планування та управління часом, витраченим на певну діяльність, особливо для підвищення ефективності, результативності та продуктивності. Це включає задоволення різноманітних потреб людей, пов'язаних з роботою, соціальним життям, сім'єю, розвагами, особистими інтересами та тимчасовими обов'язками. Управління часом дає людям «вибір» щодо вартості та управління діяльністю ефективно в часі [1]. Управлінню часом може допомогти безліч навичок, інструментів та технік, що використовуються для управління часом на виконання конкретних завдань, проектів та цілей, що відповідають термінам. Раніше управління часом було обмежене бізнесом та роботою, але врешті-решт цей час поширився на особисту діяльність. Система управління часом - це поєднання процесів, інструментів, прийомів та прийомів. Управління часом зазвичай є пріоритетом управління проектами, оскільки воно визначає терміни та обсяг проекту. Важливо також зрозуміти, що існують технічні та структурні відмінності в управлінні часом через різницю в культурних уявленнях про час [15]. Найважливіші теми з літератури з управління часом включають:

- Створення ефективного середовища (прибутковість, якість результатів, час на виконання завдання або проекту)

- Встановіть пріоритети та
- Пов'язані процеси, що скорочують час, витрачений на виконання пріоритетних обов'язків
- Визначення мети.

РОЗДІЛ 2 ВИБІР ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1 Вибір мови програмування

Розглядаючи різні мови програмування було обрані для аналізу та порівняння такі Сі-подібні мови:

- С ++.
- С #.

Проведемо аналіз між цими мовами програмування за такими критеріями:

- Швидкість розробки.

На С # стартувати розробку швидше, що дозволяє швидше отримати прототип рішення. Швидкість розробки проекту на С # значно вище ніж на С ++.

Однак, коли інфраструктура проекту створена, основні підходи і бібліотеки обрані, то швидкість розробки на С ++ і швидкість розробки на С # стають приблизно однаковими.

Таким чином, в коротких малобюджетних проектах С # матиме перевагу в швидкості розробки.

- Кроссплатформеність.

С ++ виявився кроссплатформенним, але з деякими затратами і не з усіма платформами.

С # виявився не кроссплатформенний, незважаючи на існування неофіційних net-адаптацій під різними платформами.

С # спроектований бути кроссплатформенним, однак його розвиток не пішло в цьому напрямку. Тому під Windows утворилася досить повна net-інфраструктура; на інших же платформах рівноцінної інфраструктури не з'явилося.

При цьому для розробки на C ++ склалася практично рівноцінна інфраструктура на більшості існуючих платформ, є маса бібліотек, які скомпільовані або можуть бути скомпільовані під будь-які існуючі платформи. Саме тому існує величезна кількість міжплатформенних додатків і бібліотек різного масштабу, написаних на C ++, на ряду з кросплатформенними бібліотеками є і бібліотеки специфічні для окремих платформ. Все це дає практичний рівноцінний шанси для розвитку додатків на різних платформах.

Тому рекомендувати його для кроссплатформенного використання краще C ++.

- Продуктивність коду і вимогливість до ресурсів.

Можливості по оптимізації unmanaged коду значно ширший, ніж можливості по оптимізації managed коду. Таким чином, пікова продуктивність коду досяжна тільки в unmanaged виконанні, тобто в межі, майже будь-яке завдання на C ++ може бути вирішена з меншими вимогами до ресурсів. Тому в важких завданнях, пов'язаних з обробкою великої кількості даних, C ++ має сильні переваги перед C #.

Але варто розуміти, що при виборі неправильного підходу, на C ++ цілком можна написати код, який буде працювати повільніше коду на C #, що виконує ту ж саму задачу.

Якщо говорити про сукупність суб'єктивних «простоти розробки», «краси коду» і об'єктивної продуктивності, то використовуючи C # простіше написати код, що задовольняє цим критеріям одночасно. Однак це не означає, що продуктивний код на C ++ обов'язково буде страшним або складним для сприйняття, просто при його написанні потрібно більш «творчий» підхід для задоволення перерахованих критеріїв одночасно.

Фундаментальні основи переваг C ++ в можливості писати код, який буде виконуватися безпосередньо процесором, і можливості прямої роботи з пам'яттю. Звичайно, свобода дає більше можливостей створити собі проблеми, але в ряді випадків це краще, ніж неможливість подолання стелі продуктивності. І ця стеля цілком може привести, наприклад, до того, що під

рішення задачі, для якого б вистачило одного хорошого сервера, вам доведеться збирати ферму з декількох серверів, або ж до того, що ваш додаток буде вимагати «топового» заліза на завдання, для яких вистачило б заліза випущеного років 7-10 тому.

- Бібліотеки.

Відмінність асортименту C++ і C# бібліотек в тому, що C++ бібліотек більше, вони мають велику історію, за яку стали непогано налагоджені і оптимізовані, часто кросплатформенних, багато з відкритим кодом. Однак при всіх позитивних сторонах C++ бібліотеки як мають дуже різну, часто навіть архаїчну архітектуру, часто вже не об'єктний, а структурно-процедурний інтерфейс. Пов'язано це з тим, що багато C++ бібліотеки це 3 бібліотеки.

Інша неприємна особливість C++ бібліотек - це створення і перевизначення своїх базових типів. Багато C++ бібліотеки заводять свої типи рядків, контейнерів, скасовують деякі базові типи. Для цього є логічні пояснення (найкраща продуктивність, підтримка кросплатформенних, відсутність відповідних типів на момент написання бібліотеки), проте все це створює певні незручності використання і краси коду. Базові ж C++ бібліотеки дають не так багато, як дають стандартні бібліотеки C#, тому підбір правильних бібліотек для проекту C++ - це завдання, необхідна навіть в порівняно простих проектах.

Однак ж при найближчому розгляді великий шанс, що під вашу специфічне завдання C# бібліотеки не опиняться, більш того, може виявитися, що і вирішувати таке завдання на C# досить неефективно, тому подібної бібліотеки чи не з'явиться і в майбутньому, а якщо і з'явиться, то буде працювати недостатньо швидко.

Друга неприємна особливість бібліотек C# в тому, що багато хто з них є просто обгорткою над unmanaged бібліотеками, що буде завжди приводити до втрат продуктивності на конверсіях типів, і створювати додаткові проблеми налагодження та поширення.

- Зручність налагодження.

Мова C # зручніше налагоджується, якщо це стосується managed коду, а для unmanaged коду налагодження займе більше часу та додасть складнощів в порівнянні зі звичайним unmanaged кодом C ++.

- Мова і Синтаксис.

З першого погляду код C ++ і C # дуже схожий зовні. Але різноманіття коду на C ++ більше, адже C ++ є одночасно і C і C ++ і C ++ 0x і все це ви можете використовувати одночасно (звичайно, якщо це підтримує ваш компілятор).

C # же, це тільки C #, хоча його синтаксис постійно розширюється. Код на C #, як правило, виглядає простіше і лаконічніше, ніж код C ++ (хоча це не завжди можна було сказати про перші версії C #). Мовні конструкції C ++ і C # дуже схожі, однак суттєві відмінності можна знайти в деталях.

Якщо C ++ можна дорікнути за відсутність «в базі» reflection, пізнього зв'язування і збірки сміття. Те C # треба дорікнути за відсутність повноцінних деструкторів, відсутність повноцінних макросів, досить грубу настройку успадкування, відсутність константних методів і членів, відсутність глобальних методів (процедур), дуже обмежену підтримку шаблонів, список можна продовжувати ... Однак жити без всього цього цілком можна як в випадку C ++, так і в разі C #.

Синтаксис C #, мабуть, можна назвати спрощеною версією C ++, таким чином C #, як і будь-яке спрощення, одночасно несе і позитивний і негативний ефекти.

Варто сказати, що більш складний код часто легше пишеться і аналізується, якщо написаний більш простою мовою. З цієї позиції, використовуючи C #, менше шансів допустити помилку в принципово складному коді і більше шансів написати чистий код, володіючи тими ж ресурсами. Це може бути корисно при вирішенні досить складних, але не вимогливих до продуктивності завдань. Однак при цьому більшу кількість

«синтетики» в C # робить менше оцінку продуктивності коду по його «зовнішнім виглядом».

- Вартість підтримки.

В C ++ та C # мають однакову підтримку, але програми, що написані на C # засобами net виправити неможливо, тому в таких моментах вартість підтримки для C # зростає.

- Ризики.

Основний ризик використання C # – це сильна зав'язка на Microsoft. Хоч і Microsoft – це комерційна організація, метою якої є отримання прибутку. Тому в її інтересах є розповсюдження розробки C # і net таким чином, щоб це компанії приносило прибуток.

Ризики використання C ++ також є. Одним із основних ризиків є бінарна сумісність бібліотек в цій мові програмування.

Однако з цими ризиками є ще ризики низькоякісного коду. Для C # ймовірність з'явлення коду низької якості більше в середньому, ніж на C ++, тому що C # менш вимогливий до розробника. Якщо брати C ++, то ситуація з низькоякісним кодом тут де що краще, тому що ймовірність поганого коду в даному випадку значно нище.

- Самодостатність додатків.

Повної самодостатності додатків немає ні у C ++ ні у C #. Для C ++ так чи інакше потрібен runtime, а для C # .net framework.

Однак хотілося б відзначити, що рантайм C ++, як і будь-яка інша бібліотека, може бути статичний лінкуватися в виконуваний модуль, таким чином виконуваний модуль може містити все необхідне для роботи, і за рахунок чого стане самодостатнім, в разі C # таке, стандартними Кошти не піддається реалізації.

- Зручність збірки.

Збірка C ++ проектів помітно складніше збірки проектів C #. Однак варто розуміти, що велика складність надає і додаткову гнучкість, яка рано чи пізно

може стати корисною вам. Правда до цього моменту буде лише збільшувати витрати вашого часу.

- Перспективи.

Міркування про перспективи це завжди спекуляція. На сьогодні і C++ і C# активно розвиваються. Тому перспективи між ними однакові все залежить для потрібно дана мова програмування.

Все це, швидше за все, створить запит на оптимізацію програмних продуктів і запит на паралельні обчислення. Я відчуваю деякий скептицизм щодо того, чи зможе C++ задовольнити всі ці запити.

Тому було обрано C# для програмної реалізації [16].

2.2 Вибір середовища розробки

Проводячи аналіз між різними середовищами розробки було обрано Microsoft Visual Studio тому, що дане середовище більш за все підходить зручного використання та розроблені програм на мові C#.

Microsoft Visual Studio – це середовище розробки від Microsoft, який використовується для розроблення програмної реалізації, а також можна писати сайти, додатки, служби та мобільні програми. Розглянуте середовище розробки використовує платформи Microsoft для розробки програмного забезпечення, такі як API Windows, Windows Forms, Foundation Presentation Foundation, Windows Store та Microsoft Silverlight. Завдяки цим платформам можна створити керований код, а бо написати власний.

Visual Studio має редактор та рефакторинг коду, а також приймає різні плагіни, що можуть покращити функціональність. Також один із головних переваг є підтримка 36 різних мов програмування.

Тому для рішення даної задачі було обрано програмне середовище розробки Visual Studio

2.3 Вибір СУБД

Microsoft SQL Server - це система управління базами даних, створена корпорацією Майкрософт. Як сервер баз даних, це програмний продукт, основною функцією якого є зберігання та отримання даних, необхідних іншим програмам, які можуть працювати на вашому комп'ютері або в мережі (Інтернет). Microsoft продає принаймні кілька різних версій Microsoft SQL Server для різних аудиторій та робочих навантажень, від невеликого одноразового використання до широкомасштабного мережевого використання з багатьма одночасними користувачами.

Історія Microsoft SQL Server почалася з SQL Server 1.0, першого продукту Microsoft SQL Server, 16-розрядного сервера для операційної системи OS / 2 в 1989 році, і продовжується донині.

- MS SQL Server OS / 2 був запущений в 1989 році як проект з міграції Sybase SQL Server на OS / 2 від Sybase, Ashton-Tate та Microsoft.
- NT SQL Server 4.2 був випущений в 1993 році. Тобто логін Windows NT.
- SQL Server 6.0 був випущений в 1995 році, і партнерство з Sybase припинилося. Sybase продовжує просувати Sybase Adaptive Server Enterprise, безкоштовну версію SQL Server, що надається корпорацією Майкрософт.
- SQL Server 7.0 був випущений в 1998 році. Тобто вихідний код змінився з C на C ++.
- Випущений у 2005 році, SQL Server 2005 завершує повне перетворення старого коду Sybase на номери Microsoft.
- Випущений у 2012 році, SQL Server 2012 відкриває нову еру онлайн-аналітики, додавши шар пам'яті xVelocity.

- Випущений у 2017 році, SQL Server 2017 додає підтримку Linux для наступних платформ Linux: Red Hat Enterprise Linux, SUSE Linux Enterprise Server, Ubuntu та Docker Engine [2].

Версія SQL Server Express - це безкоштовна версія SQL Server, що включає основний механізм баз даних. Кількість баз даних або користувачів, що підтримуються, не обмежується, але обмежується використанням одного процесора, 1 ГБ пам'яті та 10 ГБ файлів баз даних (4 ГБ файлів баз даних для SQL Server Express 2008 R2) [6]. Він призначений як альтернатива MSDE. Дві додаткові версії мають набір служб, які недоступні в рідній версії Express. Перший - це засіб із підтримкою SQL Server Express, що включає SQL Server Management Studio Basic. Завдяки найсучаснішим послугам SQL Server Express покращує комплексні послуги пошуку тексту та звітування [7].

Зберігання даних - це база даних, яка являє собою набір таблиць із друкованими стовпцями. SQL-сервер має різноманітні дані, включаючи старі типи, такі як Integer, Float, Decimal, Char (включаючи рядки), Varchar (різні шрифти), двійкові (для даних, що не мають BLOB-формату), текст (для текстових даних). типи. І воно запалюється. .. Якщо ви хочете повернути більше значення на число, використовується обчислення симетрії обертання або симетрія обертання (корекція), залежно від аргументу. Коло SELECT (2.5, 0) повертає 3.

Microsoft SQL Server також дозволяє визначати та використовувати стандартні типи форматів (UDT). Він також робить дані сервера доступними у вигляді таблиць та коментарів (так звані «Управління енергією» або DMV). На додаток до таблиць, база даних може також містити інші елементи, такі як коментарі, збережені процедури, індекси та обмеження та журнали транзакцій. База даних SQL Server може містити до 233 елементів і може охоплювати кілька файлів з максимальним розміром файлу 260 ка (1 екзабайт) на рівні ОС [5]. Дані бази даних зберігаються у базовому файлі даних із розширенням .mdf. Вторинні файли даних, ідентифіковані розширенням .ndf, використовуються для того, щоб дозволити розподілити дані з однієї бази даних у декілька файлів

і, необов'язково, у декілька файлових систем. Довідкові файли ідентифікуються за допомогою розширення .ldf [5].

Виділений для бази даних простір для зберігання розділений на сторінки із серійними номерами, кожна з яких має розмір 8 КБ. Сторінки є основним блоком вводу-виводу для операцій SQL Server. Сторінка має 96-байтний заголовок, що зберігає метадані, номер сторінки, тип сторінки, вільний простір на сторінці та ідентифікатор пов'язаного об'єкта для цієї сторінки. Тип сторінки визначає інформацію на сторінці. Ці дані включають дані, що зберігаються в базах даних, індексах та картах розподілу. Він містить інформацію про класифікацію сторінок таблиць та показчиків. Карта змін, що містить інформацію про зміни, внесені на інші сторінки з моменту останнього копіювання або реєстрації, або карта змін, що містить великі типи даних, такі як зображення та текст. Сторінка є основною одиницею операцій вводу-виводу, але фактичним простором управляється з точки зору 8 сторінок, і об'єкт бази даних може займати розмір усіх 8 сторінок («однаковий розмір»), а цей розділ - 7 Інше елементи, якими також можна поділитися («Змішані пікселі»). Рядок у таблиці бази даних не може перевищувати однієї сторінки, тому його розмір обмежений 8 КБ. Однак якщо дані перевищують 8 КБ і містять varchar або варбінарні рівні даних, дані в цих стовпцях будуть перенесені на нову сторінку (або, можливо, шар на сторінці, який називається одиницею розподілу) до індикатора даних [11] Буде замінено.

Для зберігання таблиці рядок розділений на кілька розділів (цифри від 1 до n). Розмір розділу залежить від вас. За замовчуванням всі рядки знаходяться в одному місці. Таблиця розділена на кілька розділів для розподілу бази даних по комп'ютерних групах. Рядки для кожного розділу зберігаються в дереві В або купі. Якщо таблиця містить індекс групи посилань, що дозволяє швидко знаходити рядки, рядки зберігаються в порядку відповідно до значення індексу, а індекс повертає В-дерево. Дані знаходяться на листових вузлах, а деякі вузли зберігають дані індексу для листових даних, доступних на кожному вузлі. Якщо індекс не розміщується в рядку, стовпець не може

використовуватися відповідно до ключа індексу. Перелічені подання мають таку ж структуру сховища, що і таблиця індексу. Таблиці без групових індексів зберігаються у структурі палі. Однак таблиця може містити виключений індекс, що дозволяє швидко знайти стовпець. У деяких випадках оптимістичні структури мають переваги щодо ефективності перед груповими структурами. І купа, і полюс В можуть охоплювати кілька розподілених одиниць [17].

Найважливіший спосіб отримання даних із бази даних SQL Server - це сканування. Запит відображається за допомогою варіанту SQL, який називається T-SQL. T-SQL - це варіант Microsoft SQL Server, який використовується спільно з Legacy та Sybase SQL Server. У запиті чітко вказано, що ви отримаєте. Це обробляє процесор запитів, який визначає послідовність кроків, необхідних для отримання запитуваних даних. Сукупність дій, необхідних для виконання запиту, називається планом дій. Спосіб обробки одного запиту може бути різним. Наприклад, запит, що містить спільний оператор та вибраний оператор, матиме однакові результати, як якщо посилання було виконано в обох таблицях, вибраних у результатах та вибраних з кожної таблиці для створення посилання, але з однаковим результатом. Це інший план реалізації. У цьому випадку SQL Server вибирає програму, яка, як очікується, принесе результати якнайшвидше. Це називається розширенням аргументу і виконується самим процесором аргументів [5].

SQL Server включає оновлення аргументів на основі витрат, які намагаються збільшити витрати на основі ресурсів, необхідних для реалізації запиту. На основі запиту оптимізатор запитів фокусується на схемі бази даних, даних бази даних та системному сховищі на той момент. Потім визначте порядок, у якому ви хочете отримати таблиці, згадані у запиті, порядок, у якому ви виконуєте дії, та спосіб доступу до таблиць. Наприклад, якщо у вашій таблиці є пов'язаний індекс, незалежно від того, використовуєте ви його чи ні, якщо індекс знаходиться у стовпчику, що не є кратним (нижній варіант), можливо, вам не знадобиться індекс, який використовується для доступу до

даних. Нарешті, він одночасно вирішує, чи дозволяти прохання. Одноразова реалізація дорожча за загальний час процесора, але вона може працювати швидше, оскільки реалізація розділена на різні процесори. Коли ви встановлюєте плани запитів, вони тимчасово зберігаються. Програма кешування використовується для наступних дзвінків з тим самим запитом. Невикористані програми з часом будуть знищені [9].

SQL Server також дозволяє визначати збережені процедури. Зберіганий процес - це запит параметру T-SQL, який зберігається на самому сервері (не надається клієнтським програмним забезпеченням, як типовий запит). Збережений процес може приймати значення, надіслане клієнтом, як вхідний параметр і повертати результат як вихідний параметр. Ви можете запам'ятати конкретні завдання, інші збережені процедури та той самий збережений порядок (до певної кількості разів). Вони можуть добровільно надати доступ. На відміну від інших питань, збережені процедури мають пов'язані імена та використовуються під час виконання для вирішення фактичного питання. На додаток до того, що номер не завжди повинен надсилати клієнт (оскільки до нього можна отримати ім'я), це зменшує мережевий трафік і трохи покращує продуктивність. Він також включає процес бронювання, якщо це необхідно.

T-SQL (Transact-SQL) - це ліцензоване розширення мови програмування Microsoft для SQL Server. Він надає інструкції REPL (Read-Eval-Print-Loop), які розширюють стандартний набір керування даними SQL (DML) та інструкціями даних (DDL), а також конфігурацію, захист та управління базами даних SQL Server.

Він надає ключові слова для дій, які може виконувати SQL Server, таких як створення та модифікація схем баз даних, імпорт та редагування даних у базу даних, а також моніторинг та управління самим сервером. Клієнти, які використовують дані або керують сервером, використовують функції SQL Server для надсилання запитів та інформації T-SQL, які обробляються сервером, а результати (або помилки) повертаються до клієнтської програми. Для цього збирайте лише прочитані таблиці, які можуть читати статистику

сервера. Ефективність управління забезпечується збереженими системними процедурами, які можна викликати із запитів T-SQL для виконання операцій управління. Ви також можете створити сервер, який посилається на T-SQL. Підключений сервер може обробляти операції, що виконуються на декількох серверах, з одним запитом [17].

Для цього проекту була обрана система управління базами даних через зручність використання, інтеграцію, побудовану C # Visual Studio, та наявність чудових зображень.

РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Діаграма варіантів використання

Діаграма варіантів або ще так звана діаграма прецедентів – це така діаграма, що показує функціональне призначення системи.

Для більш зручного відображення взаємодії роботи з клієнтами була розроблена діаграма використання. В даній діаграмі є такі компоненти:

- Менеджери.
- Розробники.
- Ліди.
- Замовлення.
- Завдання.
- Клієнти.
- Заявки.

Кожний компонент має такі функції:

- Створення.
- Редагування.
- Видалення.

На рисунку 3.1 зображено діаграму варіантів використання, яка описує можливі дії користувача в системі.

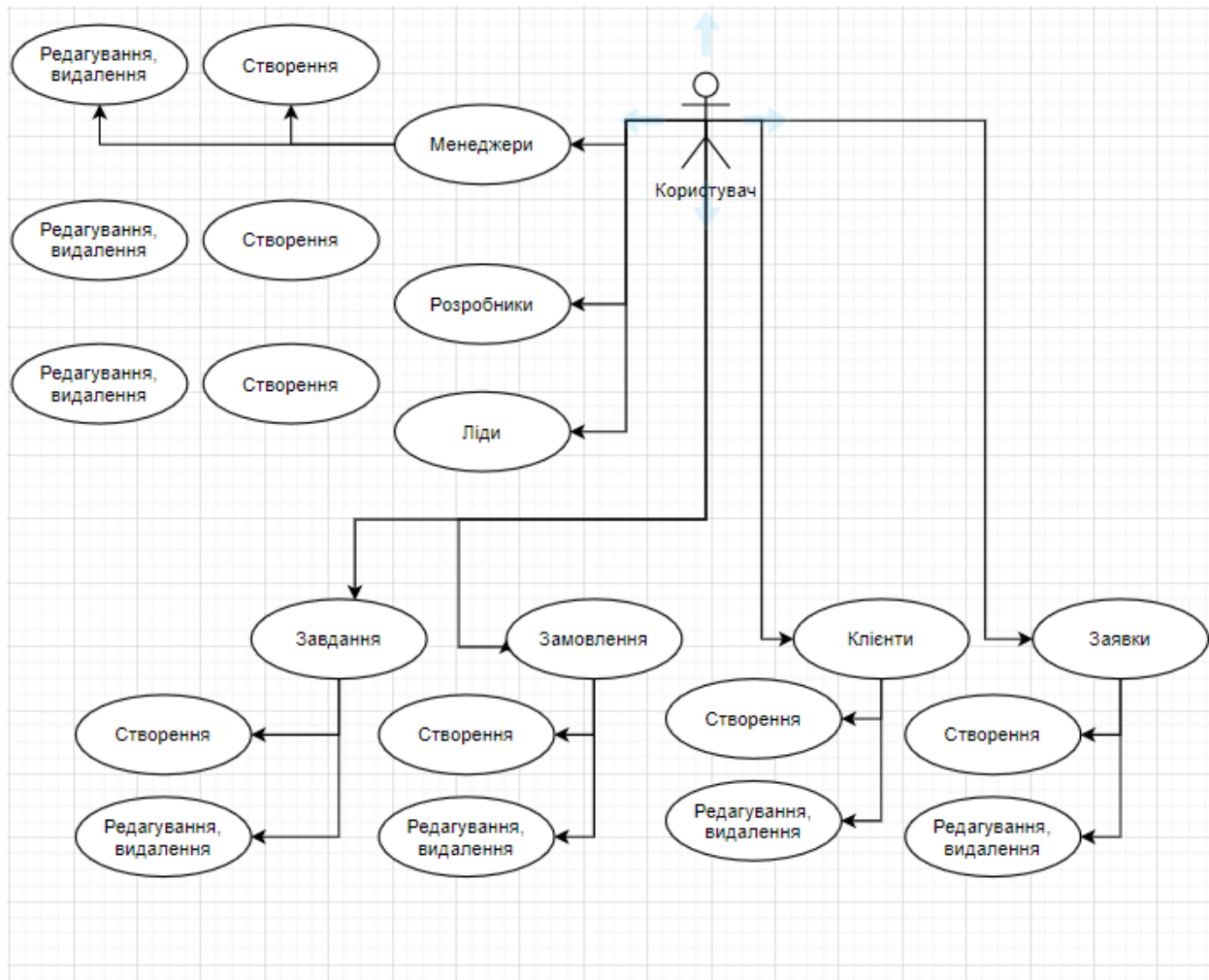


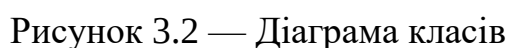
Рисунок 3.1 — Діаграма варіантів використання

3.2 Діаграма класів програмного продукту

Діаграма класів в Unified Modeling Language(UML) – це структурна діаграма статистичного типу, яка описує систему, показуючи її параметри та взаємодії між її компонентами. Такі діаграми використовуються для моделювання структури програми та для того, щоб користувачу було зручніше бачити внутрішній вигляд програми.

При проектуванні діаграми класів потрібно виділити основні аспекти розглянутої системи чи програми.

- Для кожного класу зображені свої параметри та функції (рис. 3.2).



3.3 Структурна діаграма

Також для відображення інформаційної системи було розроблено структурну діаграму, що показує основні модулі системи і взаємодію між ними (рис 3.3).

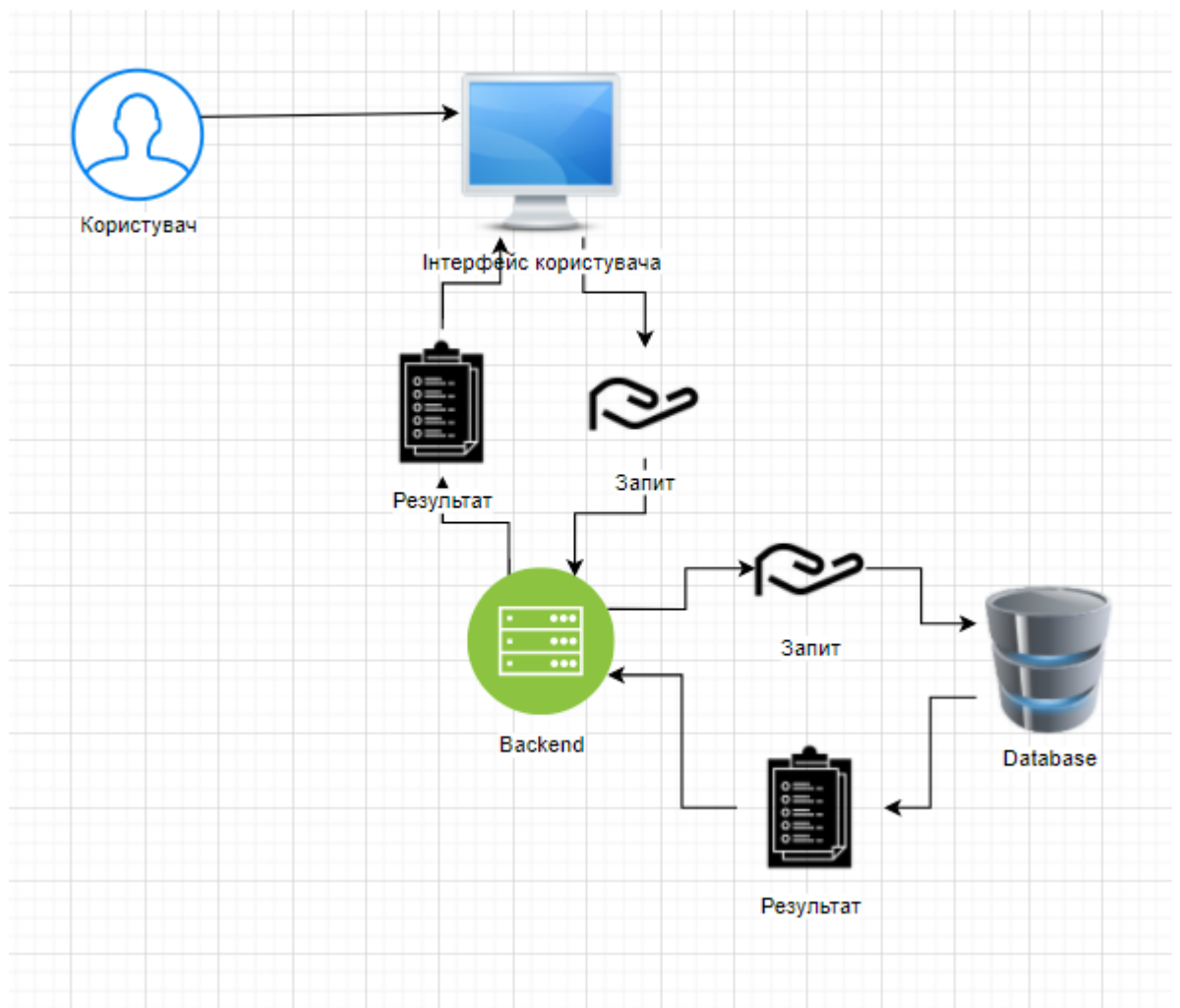


Рисунок 3.3 — Структурна діаграма ІС

3.4 Графічний інтерфейс користувача

Графічний інтерфейс користувача складається з восьми віконних форм, кожна з яких несе в собі частину загального функціоналу, а саме:

- Форма головного меню (рис. 3.4).
- Форма клієнтів (рис. 3.5).
- Форма розробників (рис. 3.6).
- Форма менеджерів (рис. 3.7).
- Форма тімлідів (рис. 3.8).
- Форма заявок (рис. 3.9).
- Форма замовлень (рис. 3.10).
- Форма завдань (рис. 3.11).
- Форма статистики тайм-менеджменту (рис. 3.12).

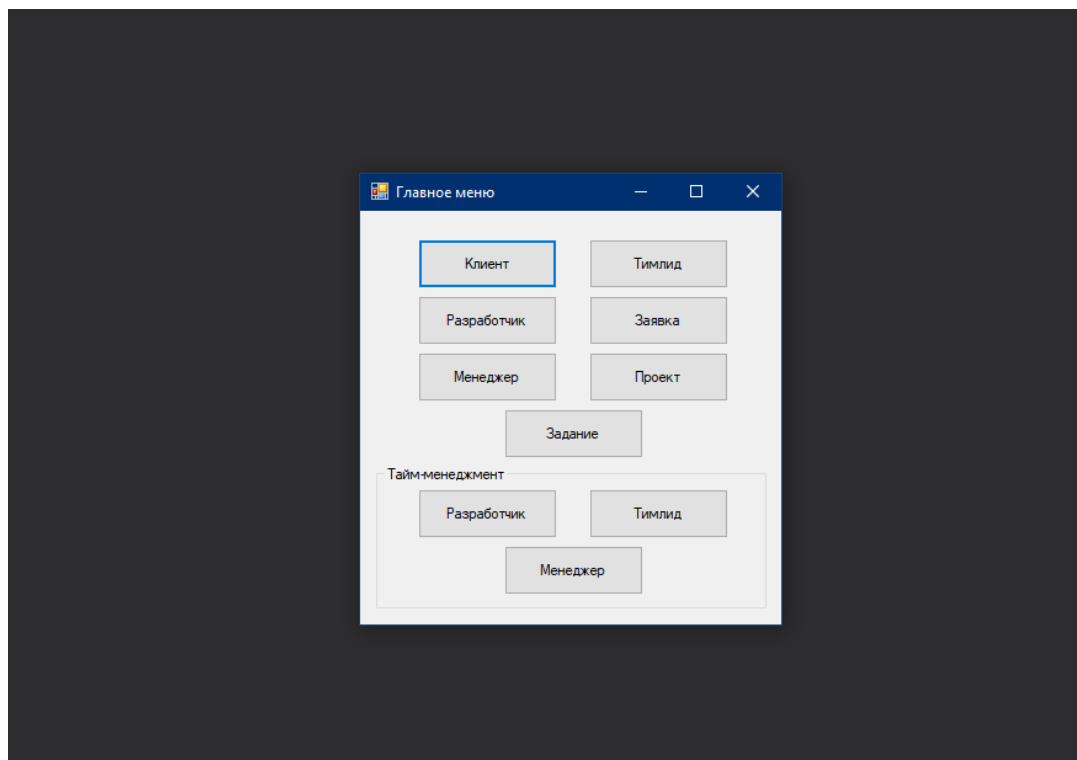
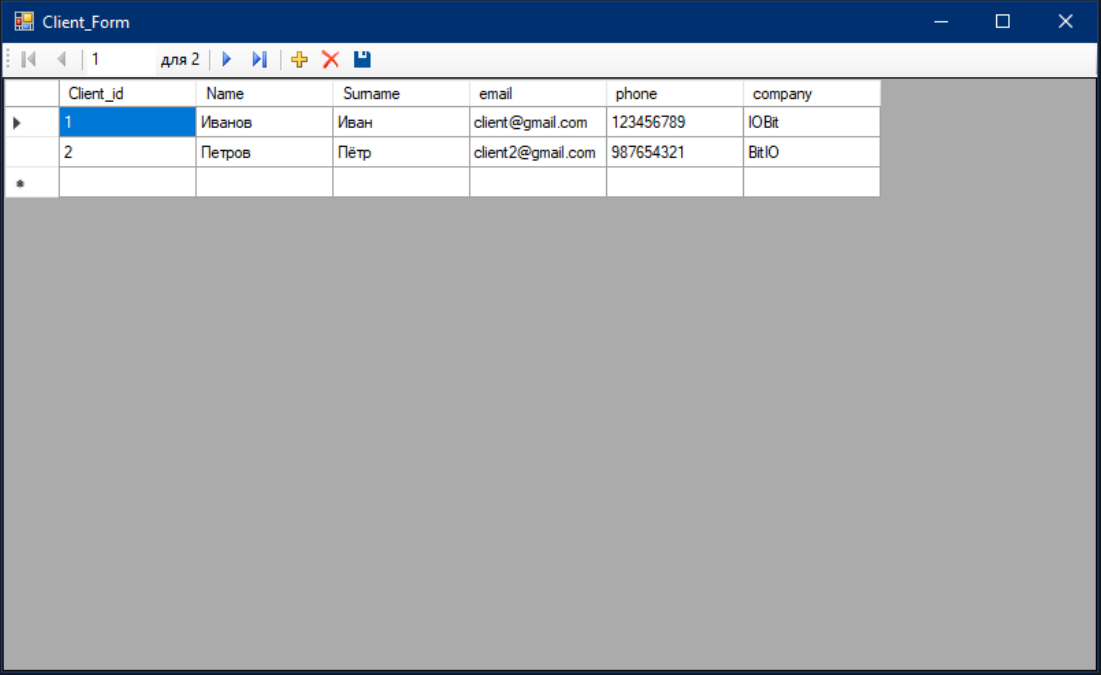


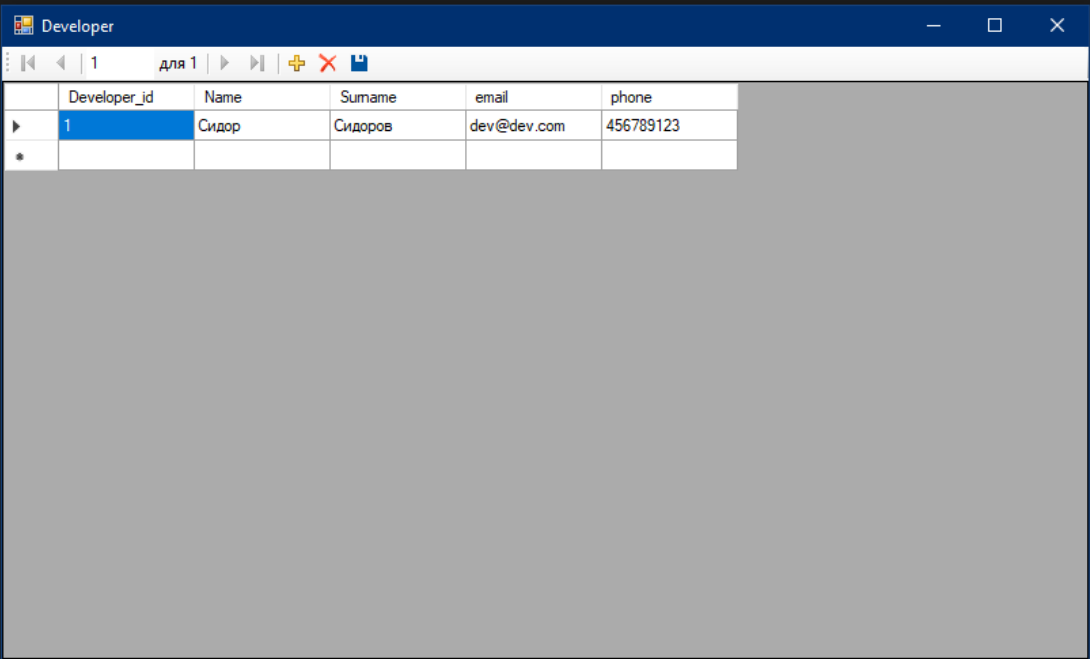
Рисунок 3.4 — Форма головного меню



The screenshot shows a window titled "Client_Form" with a standard Windows interface (minimize, maximize, close buttons). Below the title bar is a toolbar with navigation icons (back, forward, search, etc.) and a status bar indicating "для 2" (for 2). The main area contains a table with the following data:

	Client_id	Name	Surname	email	phone	company
▶	1	Иванов	Иван	client@gmail.com	123456789	IOBit
	2	Петров	Пётр	client2@gmail.com	987654321	BitIO
*						

Рисунок 3.5 — Форма клієнтів



The screenshot shows a window titled "Developer" with a standard Windows interface (minimize, maximize, close buttons). Below the title bar is a toolbar with navigation icons (back, forward, search, etc.) and a status bar indicating "для 1" (for 1). The main area contains a table with the following data:

	Developer_id	Name	Surname	email	phone
▶	1	Сидор	Сидоров	dev@dev.com	456789123
*					

Рисунок 3.6 — Форма розробників

	Manager_id	Name	Surname	email	phone
▶	1	Семён	Завалин	manager@comp....	8524567913
*					

Рисунок 3.7 — Форма менеджерів

	Lead_id	Name	Surname	email	phone
▶	1	Егор	Летов	letov@bk.ru	2584568527
*					

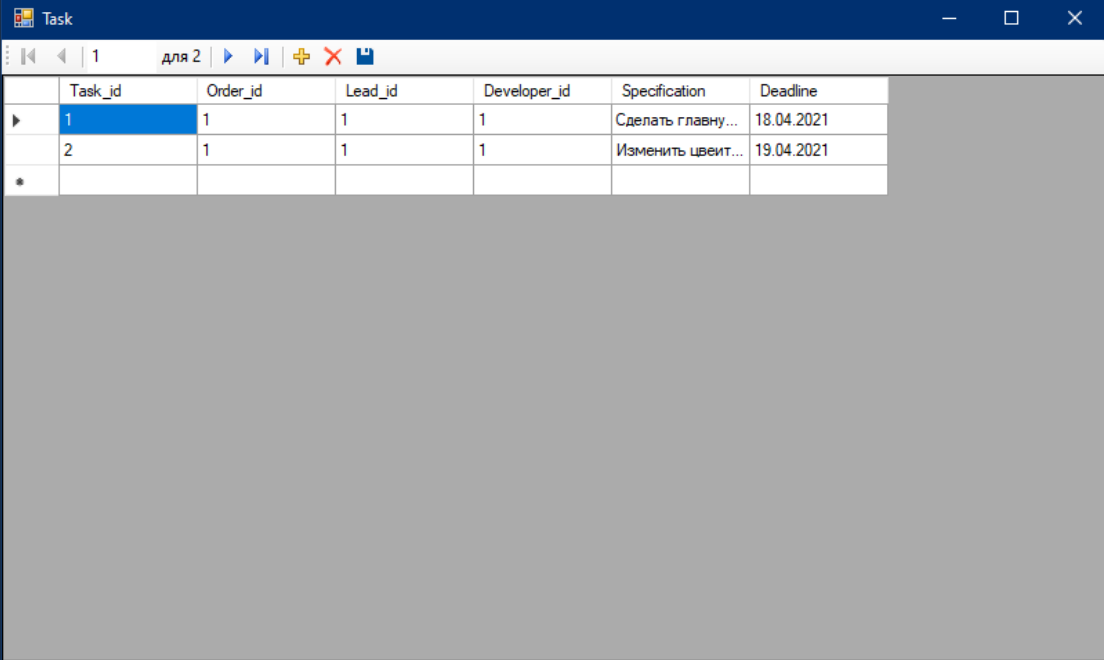
Рисунок 3.8 — Форма тімлідів

	Bid_id	Client	Manager	Bid_date	First_price	Specification	Manager_comment
▶	1	1	1	17.04.2021	1000	Some specification	-
*							

Рисунок 3.9 — Форма заявок

	Order_id	Bid_id	Client_id	Manager_id	Final_price	Manager_comment
▶	1	1	1	1	1500	-
*						

Рисунок 3.10 — Форма замовлень



Task_id	Order_id	Lead_id	Developer_id	Specification	Deadline
1	1	1	1	Сделать главну...	18.04.2021
2	1	1	1	Изменить цветит...	19.04.2021
*					

Рисунок 3.11— Форма завдань

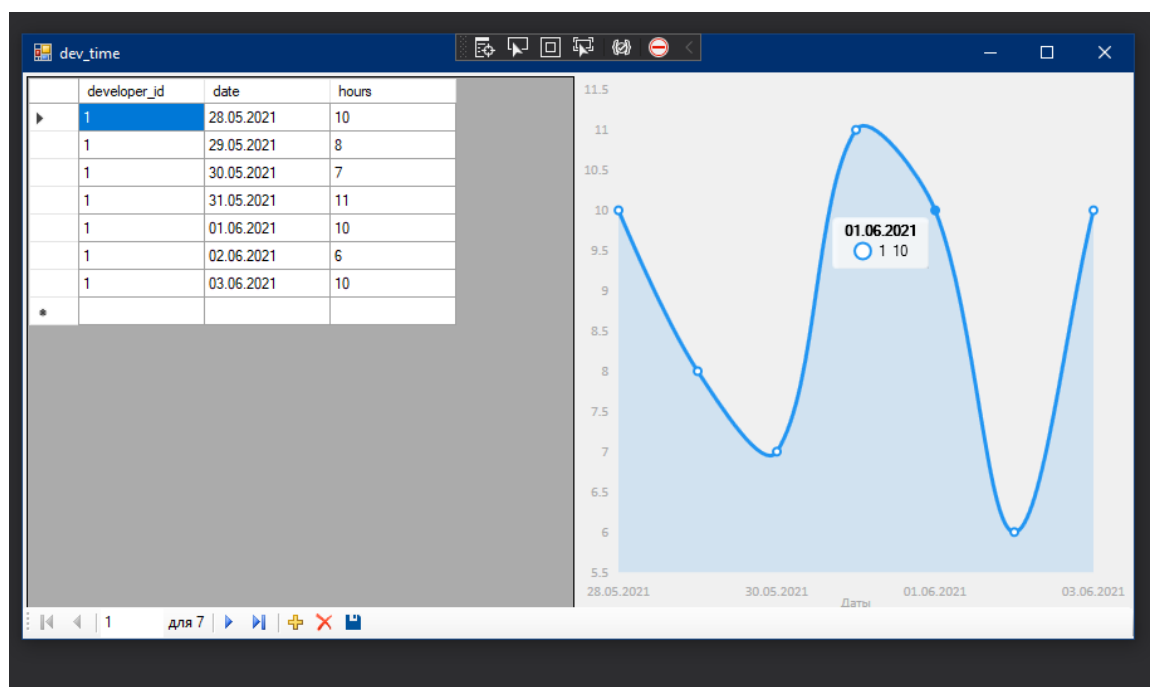


Рисунок 3.12 — Сторінка статистики тайм-менеджменту

3.5 Керівництво користувача

Для роботи з розробленою інформаційною системою не потрібно особливих умінь, проте, порядок дій для виконання деяких задач слід описати.

Порядок дій для додавання нового запису в будь-яку таблицю(розглянемо на прикладі додавання нового менеджера):

- Відкрити додаток.
- В головному меню натиснути кнопку «Менеджер».
- Натиснути кнопку з плюсом в навігаційному меню вгорі.
- В новому рядку, що з'явився, вписати всі необхідні дані.
- Натиснути кнопку «Зберегти» в меню вгорі.

Порядок дій для видалення запису з будь-якої таблиці(розглянемо на прикладі видалення менеджера):

- Відкрити додаток.
- В головному меню натиснути кнопку «Менеджер».
- Натиснути на рядок, в якому знаходиться запис для видалення.
- Натиснути кнопку з хрестиком в навігаційному меню вгорі.

3.6 Аналіз отриманих результатів

В результаті проведеного аналізу предметної області, огляду засобів розробки і проектування програмного продукту було створено інформаційну систему оптимізації роботи з клієнтами на базі приватного підприємства в сфері ІТ.

Основне питання, яке вирішує дане програмне рішення — це питання збереження, категоризацій та вдалого відображення даних ІТ компанії щодо її взаємодії з клієнтами.

Найголовніші критерії, на які спирається вся розробка — це швидкість відгуку програми на дії користувача та можливий обсяг даних, які можна зберегти в системі.

Для збереження текстових даних в системі використовується тип даних `varchar`, який займає 8000 байт, або, приблизно 8 кілобайт.

Середній запис в таблиці системи складається з 6 строкових полів. Тобто, один запис в системі еквівалентний 48 кілобайтам, або 0,0468 мегабайту.

Для порівняння, проект системи в середовищі Visual Studio займає 2,2 мегабайти, що еквівалентно 48 записам в системі.

Максимальний розмір бази даних систему управління базами даних Microsoft SQL Server становить 10 гігабайтів. 10 гігібайтів становить 10240 мегабайт або 10485760 кілобайт. Оскільки один запис займає приблизно 48 кілобайт, можна зробити висновок, що бази даних вистачить на 249660 записів в таблиці, тобто, майже 300 тисяч повноцінних записів.

Оскільки подібного роду клієнти рідко міщують об'єм, більший за 5-7 записів на день. З цього виходить, що функціонування даного клієнту з точки зору місця вистачить приблизно на 35665 днів або на 97,7 років, чого більш ніж достатньо для технологічного рішення.

ВИСНОВКИ

В процесі роботи над даним проектом були виконані наступні завдання:

- Дати характеристику предметної галузі
- Проаналізувати поняття IT-компанії
- Проаналізувати поняття інформаційної системи
- Проаналізувати поняття десктопних додатків
- Обрати засоби реалізації
- Провести огляд мови програмування
- Провести огляд системи управління базами даних
- Провести огляд середовища розробки
- Реалізувати інформаційну систему
- Створити діаграму варіантів використання
- Створити структурну діаграму
- Створити діаграму класів
- Створити графічний інтерфейс користувача
- Написати інструкцію користувача

Завдяки чіткому виконанню завдань, поставлених на початку роботи, в результаті було отримано повноцінну інформаційну систему, яку можна використовувати в реальних умовах при деяких доробках.

ПЕРЕЛІК ПОСИЛАНЬ

1. **Ryan, T.** Caffeine and computer screens: student programmers endure weekend long appathon [Електронний ресурс] / Thorne Ryan // The Arbiter. – 2015. – Режим доступу до ресурсу: <https://web.archive.org/web/20160709054212/https://arbiteronline.com/2013/03/14/caffeine-and-computer-screens-student-programmers-endure-weekend-long-appathon/>.
2. **Paul, E.** Ceruzzi. A History of Modern Computing / Paul E. Ceruzzi. – Вашингтон: Smithsonian Institution's National Air, 2005. – 406 с.
3. **Ulrich, W.** Application Package Software: The Promise Vs. Reality [Електронний ресурс] / William Ulrich // Business Technology & Digital Transformation Strategies Cutter Benchmark Review. – 2006. – Режим доступу до ресурсу: <https://www.cutter.com/article/application-package-software-promise-vs-reality-393871>. Application Package Software: The Promise Vs. Reality
4. **Gassée, J.** The silly web vs. native apps debate [Електронний ресурс] / Jean-Louis Gassée. – 2012. – Режим доступу до ресурсу: <https://web.archive.org/web/20160415200141/http://www.thisurlisfalse.com/the-silly-web-vs-native-apps-debate/>.
5. **Frechette, C.** What journalists need to know about the difference between Web apps and native apps [Електронний ресурс] / Casey Frechette // Poynter. – 2013. – Режим доступу до ресурсу: <https://www.poynter.org/reporting-editing/2013/what-journalists-need-to-know-about-the-difference-between-web-apps-and-native-apps/>.
6. **Valums, A.** Web apps vs desktop apps [Електронний ресурс] / Andrew Valums // Valums. – 2010. – Режим доступу до ресурсу: <https://web.archive.org/web/20130402210926/http://valums.com/web-apps/>.
7. **Campbell-Kelly, M.** Computer: A History of the Information Machine / M. Campbell-Kelly, W. Aspray. – New York: Basic Books, 1996. – 243 с.

8. **Piccolo, G.** Information systems for managers: with cases / G. Piccolo, F. Pigni. – Burlington: Prospect Press, 2018. – 28 с. – (Prospect Press).
9. **O'Hara, M.** Managing the three Levels of Change / M. O'Hara, R. Watson, B. Cavan // Information Systems Management / M. O'Hara, R. Watson, B. Cavan. – Washington: DBLP, 1999. – С. 63–70.
10. **Zwass, V.** Information system [Электронный ресурс] / Vladimir Zwass // Britannica. – 2016. – Режим доступа до ресурсу: <https://global.britannica.com/topic/information-system>.
11. **D'Atri, A.** Interdisciplinary Aspects Of Information Systems Studies / A. D'Atri, M. De Marco, N. Casalino. – Springer: Physica-Verlag, 2008. – 416 с.
12. **Jessup, L.** Information Systems Today: Managing in the Digital World / L. Jessup, J. Valacich. – Washington: Washington State University, 2008. – 416 с.
13. **Bulgacs, S.** The first phase of creating a standardised international innovative technological implementation framework/software application [Электронный ресурс] / Simon Bulgacs // International Journal of Business and Systems Research. – 2013. – Режим доступа до ресурсу: <https://www.inderscience.com/info/inarticle.php?artid=55312>.
14. **O'Brien, J. A.** Introduction to Information Systems: Essentials for the E-Business Enterprise / O'Brien. – Boston: McGraw-Hill, 2003. – 576 с.
15. **Beynon-Davies, P.** Enlarge Business Information Systems / Beynon-Davies. – Basingstoke: Red Globe Press, 2009. – 510 с.
16. **Дейт, К.Дж.** SQL и реляционная теория. Как грамотно писать код на SQL. – Пер. с англ. – СПб.: Символ-Плюс, 2010. – 480 с., ил.
17. **Васильев, А.** Программирование на C# для начинающих. Основные сведения / А. Васильев. – Москва : Эксмо, 2018. – 592 с. – (Российский компьютерный бестселлер).

ДОДАТОК А

Лістинг програмного коду

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Advertising_Application
{
    public partial class Bid : Form
    {
        public Bid()
        {
            InitializeComponent();

            private void bidBindingNavigatorSaveItem_Click(object sender, EventArgs e)
            {
                this.Validate();
                this.bidBindingSource.EndEdit();
                this.tableAdapterManager.UpdateAll(this.advertisingDBDataSet);

            }

            private void Bid_Load(object sender, EventArgs e)
            {
                // TODO: данная строка кода позволяет загрузить данные в таблицу
                "advertisingDBDataSet.Bid". При необходимости она может быть перемещена или удалена.
                this.bidTableAdapter.Fill(this.advertisingDBDataSet.Bid);

            }
        }
    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

```

```

namespace Advertising_Application
{
    public partial class Client_Form : Form
    {
        public Client_Form()
        {
            InitializeComponent();
        }

        private void clientBindingNavigatorSaveItem_Click(object sender, EventArgs e)
        {
            this.Validate();
            this.clientBindingSource.EndEdit();
            this.tableAdapterManager.UpdateAll(this.advertisingDBDataSet);
        }

        private void Client_Form_Load(object sender, EventArgs e)
        {
            // TODO: данная строка кода позволяет загрузить данные в таблицу
            "advertisingDBDataSet.Client". При необходимости она может быть перемещена или удалена.
            this.clientTableAdapter.Fill(this.advertisingDBDataSet.Client);
        }
    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Advertising_Application
{
    public partial class Developer : Form
    {
        public Developer()
        {
            InitializeComponent();
        }

        private void developersBindingNavigatorSaveItem_Click(object sender, EventArgs e)
        {
            this.Validate();
            this.developersBindingSource.EndEdit();
            this.tableAdapterManager.UpdateAll(this.advertisingDBDataSet);
        }
    }
}

```

```

    }

    private void Developer_Load(object sender, EventArgs e)
    {
        // TODO: данная строка кода позволяет загрузить данные в таблицу
        "advertisingDBDataSet.Developers". При необходимости она может быть перемещена или удалена.
        this.developersTableAdapter.Fill(this.advertisingDBDataSet.Developers);

    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Advertising_Application
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            Client_Form newForm = new Client_Form();
            this.Hide();
            newForm.ShowDialog();
            this.Show();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            Developer newForm = new Developer();
            this.Hide();
            newForm.ShowDialog();
            this.Show();
        }

        private void button3_Click(object sender, EventArgs e)
        {
            Manager newForm = new Manager();
            this.Hide();
            newForm.ShowDialog();
        }
    }
}

```

```

        this.Show();
    }

    private void button4_Click(object sender, EventArgs e)
    {
        Teamlead newForm = new Teamlead();
        this.Hide();
        newForm.ShowDialog();
        this.Show();
    }

    private void button5_Click(object sender, EventArgs e)
    {
        Bid newForm = new Bid();
        this.Hide();
        newForm.ShowDialog();
        this.Show();
    }

    private void button6_Click(object sender, EventArgs e)
    {
        Order newForm = new Order();
        this.Hide();
        newForm.ShowDialog();
        this.Show();
    }

    private void button7_Click(object sender, EventArgs e)
    {
        Task newForm = new Task();
        this.Hide();
        newForm.ShowDialog();
        this.Show();
    }
}

} using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Advertising_Application
{
    public partial class Manager : Form
    {
        public Manager()
        {

```

```

        InitializeComponent();
    }

    private void managersBindingNavigatorSaveItem_Click(object sender, EventArgs e)
    {
        this.Validate();
        this.managersBindingSource.EndEdit();
        this.tableAdapterManager.UpdateAll(this.advertisingDBDataSet);

    }

    private void Manager_Load(object sender, EventArgs e)
    {
        // TODO: данная строка кода позволяет загрузить данные в таблицу
        "advertisingDBDataSet.Managers". При необходимости она может быть перемещена или удалена.
        this.managersTableAdapter.Fill(this.advertisingDBDataSet.Managers);

    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Advertising_Application
{
    public partial class Order : Form
    {
        public Order()
        {
            InitializeComponent();
        }

        private void ordersBindingNavigatorSaveItem_Click(object sender, EventArgs e)
        {
            this.Validate();
            this.ordersBindingSource.EndEdit();
            this.tableAdapterManager.UpdateAll(this.advertisingDBDataSet);

        }

        private void Order_Load(object sender, EventArgs e)
        {
            // TODO: данная строка кода позволяет загрузить данные в таблицу
            "advertisingDBDataSet.Orders". При необходимости она может быть перемещена или удалена.

```

```

        this.ordersTableAdapter.Fill(this.advertisingDBDataSet.Orders);

    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Advertising_Application
{
    public partial class Task : Form
    {
        public Task()
        {
            InitializeComponent();

            private void taskBindingNavigatorSaveItem_Click(object sender, EventArgs e)
            {
                this.Validate();
                this.taskBindingSource.EndEdit();
                this.tableAdapterManager.UpdateAll(this.advertisingDBDataSet);

            }

            private void Task_Load(object sender, EventArgs e)
            {
                // TODO: данная строка кода позволяет загрузить данные в таблицу
                "advertisingDBDataSet.Task". При необходимости она может быть перемещена или удалена.
                this.taskTableAdapter.Fill(this.advertisingDBDataSet.Task);

            }
        }
    }

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

```

```

namespace Advertising_Application
{
    public partial class Teamlead : Form
    {
        public Teamlead()
        {
            InitializeComponent();

            private void leadBindingNavigatorSaveItem_Click(object sender, EventArgs e)
            {
                this.Validate();
                this.leadBindingSource.EndEdit();
                this.tableAdapterManager.UpdateAll(this.advertisingDBDataSet);

            }

            private void Teamlead_Load(object sender, EventArgs e)
            {
                // TODO: данная строка кода позволяет загрузить данные в таблицу
                "advertisingDBDataSet.Lead". При необходимости она может быть перемещена или удалена.
                this.leadTableAdapter.Fill(this.advertisingDBDataSet.Lead);
            }
        }
    }
}

```