МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп’ютерної інженерії

Патра Володимир Миколайович

# Система моніторингу робочого часу працівників IT компанії / The system for monitoring the activities of employees during working hours on the b asis of the IT company

спеціальність: 123 – Комп’ютерна інженерія
освітньо-професійна програма – Комп’ютерна інженерія

Кваліфікаційна

Виконав: студент групи КІ-41
Патра Володимир Миколайович

Науковий Керівник
Дериш Б.Б.

ТЕРНОПІЛЬ-2023

# РЕЗЮМЕ

Кваліфікаційна робота на тему «Система моніторингу діяльності працівників в робочий час на базі IT компанії» зі спеціальності 123 «Комп'ютерна інженерія» освітнього ступеня «бакалавр» містить 108 сторінки, 37 рисунки, 5 таблиць, 4 додаток.

Метою кваліфікаційної роботи є розробка додатку для моніторингу діяльності працівників в робочий час на базі IT компанії.

Методи дослідження включають методи фізичної і логічної структуризації комп'ютерних мереж, методи структурного програмування, теорія графів, елементи математичної логіки.

У проекті надана необхідна інформація та приклад розробленого додатку.

Проведено дослідження сучасних актуальних технологій Які дозволяють створювати додатки, описано архітектуру та процес створення додатку.

Розроблене програмне забезпечення призначено для автоматизації бізнес процесів IT-компанії. Розробку системи було виконано у середовищі Microsoft Visual Studio 2019 при використанні мови програмування C #. Розроблена система повинна значно полегшити роботу для всього керівного складу IT-компанії, а саме тим, тим, що розроблене ПЗ має інтуєтивний інтерфейс, зручний перегляд даних, створення та видалення записів та виконання пошуку і фільтрації потрібних записів за допомогою запитів.

Ключові слова: БАЗА ДАНИХ, ПРОЕКТ, КЛІЄНТ, ІНФОРМАЦІЙНА СИСТЕМА, СЕРВІС, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ.

# RESUME

The qualification work on the topic "System of monitoring the activities of employees during working hours on the basis of the company's IT" from the specialty 123 "Computer engineering" of the educational degree "bachelor" contains 108 pages, 37 figures, 5 tables, 4 appendix.

The purpose of the qualification work is to develop an application for monitoring the activities of employees during working hours on the company's IT base.

Research methods include methods of physical and logical structuring of computer networks, methods of structural programming, graph theory, elements of mathematical logic.

The project provides the necessary information and an example of the developed application.

A study of modern relevant technologies that allow creating applications has been carried out, the architecture and process of application creation have been described.

The developed software is designed to automate hotel business processes. The development of the system was carried out in the Microsoft Visual Studio 2019 environment using the C # programming language. The developed system should greatly facilitate the work of administrators, porters and hotel managers, namely, the fact that the developed software has convenient data viewing, adding and removing records, and searching and filtering the necessary records using queries.

Keywords: DATABASE, PROJECT, CLIENT, INFORMATION SYSTEM, SERVICE, SOFTWARE.

ЗМІСТ



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | КР.КІ. 8351793.00.00.000 ПЗ | | |
| Змн. | Лист | № докум. | Підпис | Дата | | | |
| Розробив | | Патра В.М. | | | СИСТЕМА МОНІТОРИНГУ ДІЯЛЬНОСТІ ПРАЦІВНИКІВ В РОБОЧИЙ ЧАС НА БАЗІ ІТ КОМПАНІЇ | Літ. | Арк. / Акрушів |
| Перевір. | | Мельник Г.М. | | | | | |
| Консульт. | | Савка Н.Я. | | | | ЗУНУ,ФКІТ, КІ-41 | |
| Н. Контр. | | Дериш Б.Б. | | | | | |
| Затвердив | | Дубчак Л.О. | | | | | |

# ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

| | | |
|---|---|---|
| IT | – | Information Technology |
| KPI | – | Key Performance Indicator |
| ERP | – | Enterprise resource planning |
| CRM | – | Customer Relationship Management |
| CI | – | Configuration item |
| SLA | – | Service-level agreement |
| ITSM | – | Information Technology Service Management |
| СУБД | – | Система управління базами даних |
| VBA | – | Visual Basic for Applications |
| IDE | – | Integrated Development Environment |
| MIB | – | Management Information Base |
| SLW | – | Settler LITE Data |
| API | – | Application Programming Interface |
| DAL | – | Data Access Layer |
| BLL | – | Business Logic Layer |
| ERD | – | Entity-Relationship Diagram |
| DFD | – | Data Flow Diagram |
| DSD | – | Data Structure Diagram |
| UI | – | User Interface |

# ВСТУП

Підприємства будь-яких розмірів використовують облік та аналіз робочого дня співробітників для скорочення витрат і раціональнішого використання ресурсів, тому їм необхідно мати точну інформацію про місце знаходження співробітника (на своєму робочому місці або в інших місцях, пов'язаних з виконанням своїх посадових обов'язків).

Аналіз використання робочого дня дозволяє оцінити раціональному використанню трудових ресурсів, і навіть умовам виконання плану з праці. Від раціонального використання робочого часу залежить ефективність роботи підприємства та виконання техніко-економічних показників. Слід зазначити, що аналіз використання робочого дня грає одну з найважливіших ролей у аналітичній роботі підприємства.

В результаті аналізу робочого часу розраховуються такі показники, як:

– середньооблікова чисельність працівників;

– кількість людино-днів, відпрацьованих усіма робітниками протягом року;

– кількість людино-годин, відпрацьованих усіма робітниками протягом року (фонд робочого дня);

– кількість днів, відпрацьованих одним робітником протягом року;

– понаднормово відпрацьований час;

– середня тривалість робочого дня.

Дані показники дозволяють керівництву компанії оцінювати використання робочого часу співробітниками, приймати рішення про преміювання тощо.

У більшості компаній, що займаються розробкою програмного забезпечення, розрахунок заробітної плати проходить окладним способом. Більшість співробітників компанії — аналітики з різною градацією на різних проектах, програмісти, web-розробники, тестувальники, адміністратори БД та

ін. Отже, коли співробітники неправильно документують свій годинник, навмисно чи ні, це може призвести до помилок у розрахунку заробітної плати, формуванні податкової звітності, проблем юридичного характеру. Крім того, недостовірні дані про перероблений час можуть підірвати довіру та моральний дух колег та створити негативну атмосферу всередині організації.

Безпосередньо керівники відділів з різних причин не завжди можуть проконтролювати час перебування на робочому місці співробітників. Крім функції контролю, вищі менеджери виконують інші обов'язки. Під час роботи з'являються несподівані проблеми або умови, не передбачені планом, і робочий процес повністю передбачити і розпланувати не виходить, у тому числі і фаворитам тайм-менеджменту.

Відсутність регламентованого процесу обліку та аналізу робочого часу співробітників в організації тягне за собою низку проблем, які мають економічні, соціальні та організаційні наслідки. Тому доцільним є створення для підприємства системи обліку та аналізу робочого дня.

Метою роботи є розробка системи моніторингу діяльності працівників в робочий час на базі IT компанії.

Для досягнення зазначеної мети було поставлено такі завдання:

– провести дослідження предметної області;

– на основі отриманих в результаті аналізу даних зробити постановку завдання до майбутньої системи;

– здійснити розробку програмного забезпечення;

– провести тестування розробленої системи.

Об'єктом дослідження є програмне забезпечення моніторингу діяльності працівників в робочий час.

Предметом дослідження є методи та засоби моніторингу діяльності працівників в робочий час на базі IT компанії.

# 1 ДОСЛІДЖЕННЯ СТРУКТУРИ ІТ КОМПАНІЇ

## 1.1 Дослідження діяльності та структури ІТ компанії

Системи управління інформаційними технологіями установ та компаній є досить важкими, тому що необхідно враховувати інтереси великої кількості учасників, що формують та експлуатують ІТ-ресурси (замовників створення інформаційної системи (далі ІС), розробників та кінцевих користувачів).

Поняття ІТ (з англ. "Information technology"-інформаційні технології) є загальновживаним), але з іншого боку немає загальновизнаного визначення даного поняття. Отже, інформаційні технології - процеси, методи пошуку, збирання, зберігання, обробки, надання, поширення інформації та способи здійснення таких процесів та методів. Спираючись на дане визначення ІТ менеджмент включає управління комп'ютерними і комунікаційними ресурсами компанії. Його головна мета полягає у побудові, підтримці та покращенні ІТ інфраструктури. Є три рівні ІТ менеджменту: тактичний, операційний та стратегічний. У стратегічному менеджменті відбувається середньострокове та довгострокове планування щодо розвитку ІТ інфраструктури, аналіз існуючих систем для їх подальшого розвитку та інтеграції з бізнесом, а також формується інформаційна політика та потреби бізнесу від ІТ для майбутніх проектів. На операційному та тактичному рівнях реалізується безперебійність та відмовостійкість ІТ інфраструктури.

Створення організації управління ІТ, як і кожної іншої організації управління, передбачає визначення керованих об'єктів і впливів, представлених на рис. 1.1.

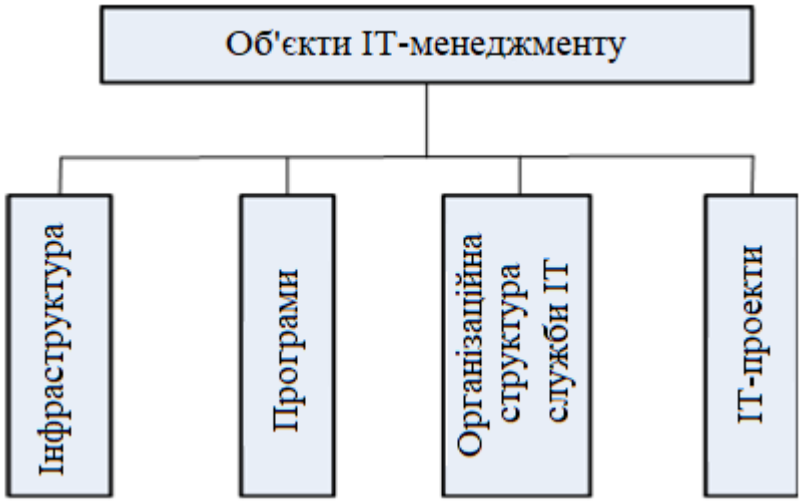


Рисунок 1.1 – Об'єкти IT управління

Об'єктами IT управління є інфраструктура, що містить ПЗ (програмне забезпечення) та технічне забезпечення. ПО включає операційні системи, спеціалізовані середовища розробки, програми з інформаційної безпеки, IT менеджменту та ін. У технічне забезпечення входить вся апаратна частина організації: комп'ютер, сервери, сканери, принтери, факси та ін.

Програми надають підтримку бізнес-процесам компанії та функціонування окремих автоматизованих робочих станцій.

Організаційна структура служби IT встановлює структуру підрозділів, поділ серед них обов'язків та функцій. IT відділ зобов'язаний гарантувати планування, впровадження, пуско-налагодження та експлуатаційний стан інформаційної системи з урахуванням правил та процедур протягом усього життєвого циклу. Проекти - це проекти щодо покращення існуючих інформаційних систем та розробки нових. Причому поліпшення деяких функцій існуючих систем зазвичай не буває спочатку, а потрібні бізнесу вже під час експлуатації інформаційних систем.

На сьогоднішній день бізнес та IT пов'язані набагато сильніше, ніж раніше. Якщо ще 30 років тому IT відділ розглядався виключно як відділ підтримки для бізнесу, то зараз це відділ бізнес-партнер, який відіграє важливу роль у подальшій долі компанії. Зараз бізнес має надвисоку динаміку (зміна цілей, відкриття нових напрямків, модернізація старих), так само як і IT системи знаходяться в умовах постійної зміни, спричинених потребами бізнесу:

– зміни зовні та всередині компанії;

– поява нових технологічних рішень;

– зміна у суспільстві.;

Крім цього, сьогоднішнє становище компаній до IT технологій характеризується досить суворим контролюванням грошових вкладень, що виділяються на IT, і збільшеним вимогам до IT з точки зору окупності інвестицій. З урахуванням цього, бізнес ставить чіткі вимоги до інформаційних систем, щоб IT менеджмент був такого рівня, який дозволяв би синхронно змінювати інформаційні системи у разі зміни потреб бізнесу. сервіс - це IT-послуга, яку компанія надає своїм клієнтам для підтримки їхніх бізнес-процесів.

Термін IT сервіс-менеджмент, з одного боку, використовується як синонім терміну «управління IT-послугами», але, з іншого боку, посилює його, маючи на увазі централізований підхід до менеджменту всією IT-організацією як сучасним сервісним підрозділом, спрямованим на надання послуг бізнес - підрозділи та є невід'ємною ланкою у виробничому процесі.

Системи зберігання даних, електронна пошта, серверна інфраструктура, надання послуг стійкості до відмови систем ІС, бізнес процеси, які реалізуються завдяки інформаційним технологіям, все це IT сервіс.

Для кожної компанії набір IT послуг індивідуальний, та залежить від специфіки галузі бізнесу, стратегії компанії, кваліфікації кадрових ресурсів тощо.

Крім цього, сьогоднішнє становище компаній до IT технологій характеризується досить суворим контролюванням грошових вкладень, що виділяються на IT, і збільшеним вимогам до IT з точки зору окупності інвестицій. З урахуванням цього, бізнес ставить чіткі вимоги до інформаційних систем, щоб IT менеджмент був такого рівня, який дозволяв би синхронно змінювати інформаційні системи у разі зміни потреб бізнесу. сервіс - це IT-послуга, яку компанія надає своїм клієнтам для підтримки їхніх бізнес-процесів.

Термін IT сервіс-менеджмент, з одного боку, використовується як синонім терміну «управління IT-послугами», але, з іншого боку, посилює його, маючи на увазі централізований підхід до менеджменту всією IT-організацією як сучасним сервісним підрозділом, спрямованим на надання послуг бізнес - підрозділи та є невід'ємною ланкою у виробничому процесі.

Системи зберігання даних, електронна пошта, серверна інфраструктура, надання послуг стійкості до відмови систем ІС, бізнес процеси, які реалізуються завдяки інформаційним технологіям, все це IT сервіс.

Для кожної компанії набір IT послуг індивідуальний, та залежить від специфіки галузі бізнесу, стратегії компанії, кваліфікації кадрових ресурсів тощо.

Отже, сервіс IT послуг включає ряд параметрів:

– продуктивність - це здатність інформаційної системи вирішувати певне завдання у певні терміни n разів. Включає ряд параметрів, таких як час реакції, час рішення, пропускна здатність системи;

– витрати - це вартість ресурсів, які витрачаються на обслуговування IT інфраструктури, сюди також включені збитки від простоїв. У витрати включено: програмне забезпечення, комп'ютери, сервери, сканери, принтери, МФУ, зарплата співробітників, комунальні платежі, залучення третіх осіб для надання певних послуг (наприклад, інтернет-провайдер);

– Функціональність визначає поставлене завдання та мета його використання у бізнес-процесі.

– надійність - це середній період між двома збоями;

– конфіденційність прораховує ймовірність несанкціонованого проникнення даних. Як правило, кількісний показник конфіденційності не виводять, а класифікують за ступенем захищеності IT інфраструктури;

– масштаб визначає розмір та складність робіт з підтримки IT інфраструктури. До його показників відносяться кількість апаратних та програмних засобів, користувачів, бізнес-процесів тощо;

– доступність встановлює частину скоординованого часу сервісу, що вимірюється в %, і показує скільки часу IT сервіс доступний;

– час обслуговування - це період часу, коли IT відділ підтримує сервіс. Наприклад, 5 днів на тиждень, з 9.00 до 18.00 або 24/7 цілий рік.

У наш час основою ефективного бізнесу є правильне використання інформаційних технологій, які забезпечують конкурентоспроможність та рентабельність фірми. Головна мета роботи інформаційних систем – це надання покращень бізнес-процесів, їх оптимізація та автоматизація. Завдання IT сервісу компанії надавати якісно послуги з необхідною доступністю та максимальною стійкістю до відмови, згідно з вимогами бізнесу, щоб ІС і вся IT інфраструктури працювали на благо компанії. сервіс виконує низку процесів у своїй діяльності:

– моніторинг – аналіз всіх процесів IT сервісу;

– організаційна діяльність та планування - цей процес відповідає за розробку стратегії, планування необхідних IT ресурсів спільно з бізнесом, ризик менеджмент, та розробка методології для надання якісних послуг;

– впровадження нових інформаційних систем - їх планування, розробка та введення в експлуатацію, використання;

– надання послуг IT сервісу – прописуються вимоги до IT відділу з боку бізнесу, узгодження ресурсів та послуг, які будуть надані кінцевому користувачеві.

Структура IT відділу може виглядати по-різному, залежно від різних факторів:

– від масштабу організації та IT інфраструктури;

– від масштаб IT відділу;

– специфіки підприємства, галузь;

– територіальний розподіл офісів IT компанії.

Приклад структури середнього IT відділу представлений рис. 1.2.


Рисунок 1.2 – Структура середнього IT відділу

Отже, відділ управління розробками, який розробляє та впроваджує інформаційні системи та відділ управління супроводом, що підтримує IT інфраструктуру підприємства, підпорядковуються директору IT компанії. Ці два відділи повинні працювати в синергії, тому що від відділу розробок залежить, як буде реалізований функціонал інформаційної системи, а відділ підтримки відповідальний за те, як кінцеві користувачі працюватимуть з ІС та як ця ІС експлуатуватиметься.

Команда розробників інформаційної системи підпорядковується керівнику проекту. Усередині проекту є розподіл ролей та обов'язків, які закріплюються за кожним співробітником. Над одним проектом працює одна команда. У даному прикладі можна виділити відділ розподільчих систем та відділ офісних додатків, які підпорядковуються управлінню розробки. Перший займається впровадженням таких систем, як ERP, CRM, бухгалтерією та інших. А потім покращенням даних систем, додаванням функціоналу, налаштування звітів тощо. Другий відділ (офісних додатків) надають офісний пакет стандартного програмного забезпечення. Директор управління ставить завдання менеджерам проектів, які транслюють поставлені завдання до проектної команди. Якщо менеджерів проектів немає, то цей функціонал на себе бере директор управління розробками.

В управління супроводу співробітники здебільшого складаються із системних адміністраторів, які мають схожі навички та кваліфікацію. У даному управлінні відповідають за роботу робочих станцій, серверів, IP-телефонії, взаємодію з провайдером інтернету, за безперебійність локальної мережі та багато іншого. Відділ моніторингу (Service Desk) підпорядковується директору IT сервісу та щомісяця надає звіт з виконаних завдань, інцидентів, запитів на обслуговування. У цьому відділі всьому управлінню супроводу розробляється KPI (Key Performance Indicator)- це показник досягнення успіху у певної діяльності чи досягненні певних цілей). Можна сміливо сказати, що KPI - це кількісно вимірний індикатор фактично досягнутих результатів. У цьому відділі аналізують статистичні дані, на основі яких можна оцінити, на якій стадії знаходиться IT сервіс.

Раніше у більшості компаній IT відділ застосовував функціональний підхід управління. Тобто був не IT сервіс, а перелік IT функцій, які були потрібні компанії. Але такий підхід згодом визнали менш ефективним, оскільки IT відділ не брав активної ролі у бізнесі, а був лише відділом підтримки. Сервісний підхід передбачає надання кінцевої користі бізнесу, залишаючи за собою право спільно з бізнесом виробити функціональність, за допомогою якої будуть досягатися поставлені цілі.

Для впровадження IT сервісу в компанію потрібно пройти кілька етапів.

Планування у цьому блоці формуємо стратегію, чіткий план впровадження IT сервісу, описуємо, який функціонал буде покритий, етап застосування, етап пуско-налагодження. Послуга надається в тестовому режимі, етап експлуатації. IT сервіс впроваджений та безперервно покращується.

Ціна послуги (IT сервісу) визначається на стадії планування. На стадії впровадження контролюється бюджет проекту. На етапі пуско-налагодження послуга тестується, а на етапі експлуатації перевіряється витрата коштів на даний IT сервіс, що звіряється із запланованим раніше бюджетом.

Виходить, що між операційною діяльністю IT відділу та параметрами IT сервісу (послугою) немає чіткої відповідності. Одна й та сама функція IT

відділу може бути у кількох ІТ сервісах, і навіть у всіх послугах (ІТ сервісах) компанії. Через це є низка проблем.

Якість ІТ послуги (сервісу), що надається, залежить від декількох функцій ІС, а відповідальність за якість, доступність, надійність має бути закріплена відповідальна особа. У функціональному управлінні ІТ службою цією особою є Директор ІС.

За такої системи, коли за співробітниками закріплені функціональні обов'язки, виникає проблема єдиної точки входу.

Виходить, що функціональний підхід забезпечує поточну діяльність ІТ відділу, а не вирішення всіх стратегічних та управлінських завдань.

Описані вище проблеми може бути подолано процесним підходом. Цей підхід змінює лише управлінські функції ІС. За певним процесом ставиться менеджер, який визначає цілі процеси, його KPI, ролі учасників та правила виконання цього процесу. Таким чином з'являється єдина точка входу, а за якість відповідає конкретна людина, яка розподіляє внутрішню функціональність ІТ сервісу, регламентує весь процес та складові роботи в ньому.

Цей метод не вимагає додаткових трудових ресурсів. Менеджер процесу управляє не підлеглими йому співробітниками, які можуть бути у різних відділах. Сам менеджер також має певну посаду та також виконує операційні завдання.

Дослідження також включає вивчення ролей та обов'язків працівників в різних підрозділах компанії. Це допомагає зрозуміти, які конкретні завдання виконуються працівниками на різних посадах, які навички і знання вимагаються для успішної роботи, та які метрики використовуються для оцінки їхньої продуктивності.

Крім того, проводиться аналіз комунікаційних процесів в компанії. Дослідження включає вивчення внутрішньої комунікації між співробітниками та комунікації з клієнтами або зовнішніми стейкхолдерами. Важливо визначити, які засоби комунікації використовуються, як вони підтримуються технічно та як впливають на ефективність роботи команди.

Ці аналітичні дані про діяльність та структуру ІТ компанії є ключовими для розробки системи моніторингу діяльності працівників. Вони надають важливу інформацію про потенційні проблеми та вузькі місця в бізнес-процесах компанії, а також про основні потреби та вимоги до системи моніторингу. Дослідження діяльності та структури компанії є основою для подальшого розроблення ефективної системи, яка враховує специфіку та потреби самої компанії.

## 1.2 Аналіз бізнес процесів ІТ компанії

Ефективність бізнес-процесів організацій сьогодні значною мірою залежить від діяльності ІТ-підрозділу. Технології продовжують удосконалюватися при цьому інфраструктура розширюватиметься. Тому, щоб забезпечити всі потреби бізнесу, ІТ-механізми мають працювати чітко та злагоджено, якісно підтримуючи усі робочі процеси. Для цього було розроблено спеціальний стандарт – ITSM, Information Technology Service Management або «концепція управління якістю інформаційних послуг».

Головна ідея цієї моделі полягає у зміні ролі ІТ-відділу. Раніше цей підрозділ був допоміжним, який відповідав лише за функціональність додатків, робочих комп'ютерів співробітників, серверного та мережевого обладнання. Зараз ІТ-відділ стає повноцінним учасником бізнес-процесів, надаючи конкретні послуги іншим підрозділам і вибудовуючи з ними відносини «постачальник — споживач послуг».

В ідеалі картинка виглядає так. Підрозділи організації, виступаючи у ролі замовника, чітко визначають перелік необхідних ІТ-послуг належної якості, що забезпечує їхню діяльність. Керівник підприємства виділяє на це гроші. ІТ-підрозділ, діючи як виконавець, здійснює підтримку та розвиток ІТ-інфраструктури і тим самим забезпечує якісне надання необхідних послуг.

Щоб реалізувати такий підхід, IT-відділи мають навчитися працювати за нових умов — керувати не ресурсами організації, а послугами на основі даних ресурсів.

Стандарт управління IT базується на трьох принципах:

– детальне опрацювання IT-бізнес-процесів;

– висока кваліфікація айтішників та відповідальність кожного з них за конкретний спектр завдань;

– матеріальна база, що забезпечує якість послуг, що надаються: технології, служби підтримки, контролю, управління, тестування тощо;

Другий і третій елементи також важливі, але визначальним є перший. Без детального опрацювання неможливо правильно скоординувати діяльність організації та отримати ефективні бізнес-результати. Потрібно так розробити IT-процеси, щоб чітко визначити алгоритм дій персоналу у конкретних ситуаціях, узгодити діяльність всього колективу – підрозділів, співробітників та служб, встановити взаємозв'язки між етапами надання послуги. Також необхідно документувати процеси, щоб оцінити їхню продуктивність. Це дозволить забезпечувати відповідну якість послуг, покращувати та змінювати кожен етап, запобігаючи появі збоїв.

Керує всім процесом менеджер – Process Owner. Він призначає виконавців, аналізує вплив IT-процесів на бізнес-діяльність компанії, взаємодіє з керівниками інших підрозділів та організації загалом.

Типову модель IT-процесів в IT компанії можна розділити на декілька категорій. Розглянемо їх нижче.

Управління змінами та конфігурацією. Це головні процеси у структурі управління IT-послугами. Забезпечуючи стабільність IT-системи, вони взаємодіють з рештою елементів Типової моделі. Ці процеси керують змінами інформаційної інфраструктури та її конфігурацією.

Перший, change management, відстежує всі перетворення IT-середовища та контролює внесення змін до IT-системи, узгоджуючи заявки, розставляючи пріоритети, оцінюючи ризики та регулюючи відновлення після збоїв. А

оскільки кожна дія викликає будь-які зміни в інфраструктурі, то вона в будь-якому випадку взаємодіє з процесом управління змінами.

Призначення другого, configuration management, полягає у реєстрації та контролі відомостей про IT-середовищі, точніше, про кожен з елементів конфігурації (CI, configuration item). Процес обробляє дані про атрибути CI (обладнання, софт, співробітники), їх статус (у дії, ремонті) та зв'язки між ними. До процесу управління конфігурацією також звертаються всі складові типової моделі.

Стратегічні бізнес-процеси. Завдяки цим стратегічним розробкам айтішники функціонують як представники бізнесу, а не обслуговуючий персонал. Одні процеси дозволяють проаналізувати ринок IT-послуг та визначити вимоги до IT-підрозділу (business assessment). Інші (customer management) — керувати користувачами, тобто передбачати їх запити, продавати послуги IT, оцінювати рівень задоволеності замовника. Ці два процеси взаємопов'язані. Дані про користувачів використовують щодо ринку та конкурентної обстановки, а всі разом — для розробки IT-стратегії (IT strategy development). У ході останнього процесу перед IT-відділом ставлять конкретні цілі та завдання (а також шляхи їх досягнення та вирішення), визначають бюджет, обирають технології та архітектуру IT-середовища та виконують інші стратегічні дії.

Управління інформаційними послугами. У ході виконання цих процесів загальну стратегію перетворять на детальний план, на основі якого складають конкретні договори IT-обслуговування. Розробка складається з наступних дій.

Планування послуг (Service Planning). Це формування та контроль переліку стандартних послуг, який за потреби змінюється відповідно до специфіки бізнес-підрозділів, а також аналізу можливих збитків.

Управління рівнем послуг (service level management). Конкретні параметри та оцінка вартості стандартної послуги дозволяють визначити рівень обслуговування, достатній для задоволення потреб клієнта. На основі результатів, отриманих після закінчення цих двох процесів, замовник та виконавець укладають угоду SLA.

Управління безпекою (security management). У ході цього процесу визначають захисну політику та доводять її до кожного співробітника організації, забезпечуючи інформаційну безпеку.

Забезпечення готовності послуг (availability management). Це контролює те, наскільки послуга підготовлена для передачі замовнику відповідно до виставлених ним параметрів. Тут, залежно від вимог клієнта, кінцевий результат може змінитися порівняно з тим, що було визначено на стадії планування.

Управління ресурсами (capacity management). Контролює робоче навантаження обладнання відповідно до SLA. Дані, отримані внаслідок цього процесу, допомагають спланувати послуги та керувати їх рівнем.

Зниження видатків (cost management). Процес обчислює реальну вартість IT-послуги, встановлює її бюджет, аналізує використання та є орієнтиром для вдосконалення обслуговування.

Розробка, вдосконалення та впровадження послуг. У ході цих процесів розробляють нові та вдосконалюють діючі послуги, а також тестують їх та інтегрують до IT-системи. Після успішних випробувань компонент повністю впроваджується у інформаційне середовище. Відповідно до завдань процеси діляться на 2 стадії. Build&test реалізує та тестує послугу, перевіряючи її на узгодженість із конкретними стандартами. Release to production, своєю чергою, випускає продуктивну версію з урахуванням результатів першого етапу і вводить їх у дію.

Підтримка інфраструктури. Ці процеси підтримують інфраструктуру, вирішують проблеми, стежать за задоволеністю замовників. Управління операціями (operations management) моніторить стан IT-ресурсів, координує чергу друку та резервування, адмініструє IT-структуру. Служба технічної підтримки (Help Desk) або управління інцидентами (incident management) швидко та якісно обробляє заявки та відновлює готовність послуги у разі виникнення збоїв. Управління ж проблемами (problem management) виступає у вигляді профілактичних заходів, запобігаючи інцидентам на основі інформації, отриманої в результаті попереднього процесу.

Впровадити концепцію управління якістю IT-послуг можна для будь-яких процесів. Ось кілька напрямків, що дозволяють вирішити найболючіші питання. Необхідно розділяти процеси, які керують інцидентами та проблемами. Якщо покласти ці обов'язки на одну людину, то весь свій робочий час вона витрачатиме на вирішення поточних збоїв, а до запобігання проблемам справа не дійде.

Обов'язково розробляти угоду SLA - тільки вона дає можливість замовникам та виконавцям говорити однією мовою. На основі цього перші розуміють, що саме вони отримають, а другі - що конкретно хочуть від них клієнти. Не менш важливо керувати змінами для забезпечення стабільного функціонування IT-інфраструктури.

Також потрібно керувати конфігураціями, щоб оперативно отримувати відомості про її елементи, що зазнають змін. Це дає змогу провести ефективний аналіз можливих ризиків. Приклади нескінченні, кожна організація може навести власний досвід застосування типової моделі.

Отже, концепція управління якістю IT-послуг нова, але необхідна у світлі масштабного розвитку інформаційних технологій. Усі процеси тісно пов'язані один з одним і вибудовують повноцінну стратегію для ефективного управління послугами, а отже, для повноцінного та успішного функціонування IT-компанії.

## 1.3 Обґрунтування проектних рішень

У цьому розділі будуть обґрунтовані проектні рішення для системи моніторингу діяльності працівників в робочий час на базі IT компанії. Будуть розглянуті різні методи та інструменти моніторингу, включаючи використання програмного забезпечення, відеоспостереження, аналіз даних тощо.

Обґрунтування проектних рішень передбачає визначення потенційних переваг, витрат та можливих ризиків впровадження системи моніторингу. Будуть враховані етичні аспекти використання системи моніторингу, зокрема

забезпечення конфіденційності та приватності працівників. Також буде проведений аналіз впливу системи моніторингу на мотивацію працівників та їхнє відчуття приватності.

Важливим фактором, який необхідно врахувати при розробці програмного забезпечення є потреба в обчислювальних ресурсах, які є в агентстві.

Мінімальні вимоги для запуску та функціонування розробленого програмного забезпечення:

- процесор PentiumI V/Xeon 2.4 ГГц;
- оперативна пам'ять: 1024 Мб і вище;
- вільний дисковий простір не менше 120 Мб;
- мережева картка;
- миша;
- клавіатура;
- монітор.

З опису видно, що до технічного забезпечення програми потреби не є високими.

## 1.4 Постановка задачі

Огляд сучасних підходів та засобів до проектування та розроблення програмного забезпечення дозволив обрати для створення власної системи ефективні технології та інструментальні засоби: IDE – Visual Studio 2019; Мова програмування – C#.

На основі виконаного аналізу предметної області можна сформулювати постановку задачі.

Розробити систему, яка буде здійснювати моніторинг діяльності працівників в робочий час на базі IT компанії, дозволить вести облік

співробітників, проектів. Також забезпечити можливість формувати необхідної звітності з метою моніторингу діяльності працівників.

Розробити зручний графічний інтерфейс для роботи з програмою. Основні дії та взаємодія між користувачем та системою повинні супроводжуватися відповідними повідомленнями для користувача.

Створити навігаційне меню для можливості швидкого та зручного отримання доступу до потрібної функції в системі.

Розробити супровідну документацію до створеної системи.

Дипломна робота припускає розробку додатка засобами об'єктно-орієнтованого середовища програмування .

Розробка та реалізація додатку включає 3 основні модулі:

– модуль вводу/редагування інформації;

– модуль пошуку інформації;

– модуль формування звітності

Реалізація додатка виконується з використанням технологій .NET та мови програмування c#. Результат –Windows додаток.

Вона повинна містити в собі сутності, введення, пошук, звітність:

– .програміст (інформація про програміста): прізвище, ім'я, номер телефону, домашня адреса, електронний адрес та посада у компанії;

– замовник (інформація про замовника): прізвище, ім'я, номер телефону, домашня адреса, електронний адрес.

– команди (інформація про команди програмістів): назва та опис;

– посади (інформація про посади): назва та опис;

– проект (інформація про проект): назва проекту, ідентифікатор проекту, ідентифікатор замовника, ідентифікатор команди, початок проекту, закінчення проекту, прибутки;

– тип проекту (інформація про типи проектів): назва та опис;

– задачі проекту (інформація про задачі проекту): назва задачі, ідентифікатор програміста, ідентифікатор проекту, фактичний термін виконання задачі, складність та статус виконання.

– вводиться штат із зазначенням посади;

– фіксується проект (замовник, тип проекту, команда, дата початку, дата здачі, прибуток);

– формується команда, призначається відповідальний виконавець;

– проект розбивається завдання, вказуються терміни їх виконання, виконавці, коефіцієнт складності завдання;

– проектів, у яких брав участь заданий програміст. Вибір усіх завдань, у яких був задіяний вибраний програміст;

– по проекту - кількість поставлених завдань, кількість виконаних завдань, кількість програмістів, відсоток виконання;

– по вибраному програмісту за вказаний період часу проекти, завдання (термін виконання).

Отже, в даному розділі було проведено дослідження діяльності та структури ІТ компанії, також проаналізовано основні бізнес процеси ІТ компанії. Після чого здійснено постановку задачі для розробки системи моніторингу діяльності працівників в робочий час на базі ІТ компанії.

# 2 ВИБІР ТЕХНОЛОГІЙ ПРОЕКТУВАННЯ ТА ОСОБЛИВОСТІ АРХІТЕКТУРИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## 2.1 Вибір технологій

Для побудови користувацького інтерфейсу будемо використовувати середовище розробки Visual Studio 2019 та мову програмування C#.

C# широко використовується професіоналами для розробки великих програмних продуктів завдяки наступним аспектам мови:

- сучасна мова програмування загального призначення;
- підтримка об’єктно-орієнтованої парадигми;
- підтримка компонентно-орієнтованої парадигми;
- мова легка для вивчення навчитися;
- добре структурована;
- дозволяє розробляти ефективні програми;
- мова має підтримку різних комп'ютерних платформах;

C# спроектовано таким чином, що мова відповідає традиційним мовам високого рівня, C та C++ і є об'єктно-орієнтованою мовою програмування. Мова дуже схожа на Java, має численні сильні функції програмування, які роблять його привабливим для багатьох програмістів у всьому світі.

.NET Framework - це платформа для розробників із відкритим кодом, яку можна використовувати для створення широкого кола програм. Цей безкоштовний крос-платформний фреймворк підтримує декілька мов і має великі бібліотеки коду, які спрощують створення додатків для мобільних пристроїв, робочих столів та Інтернету.

Платформа .NET була розроблена для досягнення наступних цілей:

- сумісність;
- підтримка різних платформам;
- мовна незалежність;
- бібліотека базових класів;
- легка розробка;

– безпека.

Для розробки користувацького інтерфейсу платформа .Net має декілька технологій, одна з яких – WinForms. Не дивлячись на те, що ця технологія досить не нова, її важко назвати застарілою. Вона надає широкий спектр різних інструментів для побудови зручного та сучасного інтерфейсу. Крім того, IDE, які підтримують C# та .Net, надають зручний інтерфейс для графічної побудови користувацького застосунку, який розробляється.

Отже, C# та платформа .Net має низку характеристик, які задовольняють вимоги щодо розробки клієнтської частини системи складського обліку. Набір готових класів у стандартній бібліотеці, лаконічний зрозумілий синтаксис мови та зручний конструктор користувацького інтерфейсу зробить розробку зручною та достатньо швидкою. Об’єктно-орієнтована парадигма дозволить спроектувати систему таким чином, що розширення функціоналу буде без накладних розходів ресурсів розробки. Платформа .Net забезпечить безпеку, ефективність програмного забезпечення, а також підтримку декількох платформ.

Для розробки інформаційної бази використовувався Microsoft Access. MS Access – це СУБД, що входить до складу пакету офісних програм Microsoft Office. Дана система управління базами даних має широкий спектр функцій (зв'язні запити, сортування, зв'язки із зовнішніми таблицями та базами даних).

Переваги використання:

– простий інтерфейс користувача, що дозволяє розробляти додатки, використовуючи вбудовані бібліотеки;

– зберігає всі дані у одному файлі;

– пропонує велику кількість «Майстрів», які допомагають уникнути рутинних дій і полегшують роботу досвідченому в програмуванні користувачеві;

– поширеність, яка зумовлена тим, що Access є продуктом компанії Microsoft;

– постійно оновлюється виробником, підтримує безліч мов;

– орієнтованість на користувача з різною професійною підготовкою;

– широкі можливості імпорту/експорту даних у різні формати;

– наявність розвинених інтегрованих засобів розробки додатків. Більшість програм, що розповсюджуються серед користувачів, містить той чи інший обсяг коду VBA (Visual Basic for Applications);

– Наявність вбудованої мови макрокоманд;

Недоліки:

– обмежені можливості щодо забезпечення розрахованого на багато користувачів середовища;

– має нескладні способи захисту з використанням пароля БД (можливе застосування додаткових заходів щодо захисту від несанкціонованого доступу з використанням процедур VBA);

– в питаннях підтримки цілісності даних відповідає лише моделям БД невеликої та середньої складності;

– не розповсюджується безкоштовно.

Отже СУБД Access було мною вибрано туму, що розроблена програма може легко переноситись на будь-які системи сімейства Windows і не потребує додаткового встановлення сервера бази даних. Реляційність бази даних дає можливість ефективно використовувати пам’ять та уникнути дублювання інформації. Завдяки відкритому коду не потрібно витрачати зайві ресурси на різні ліцензії, тобто розробка та використання системи на базі СУБД Access значно дешевша. Підтримка проекту відбувається вже багато років розробниками з усього світу, через що надійність та безпека продукту на високому рівні. Таким чином, СУБД Access повністю покриває вимоги щодо розробленого проекту.

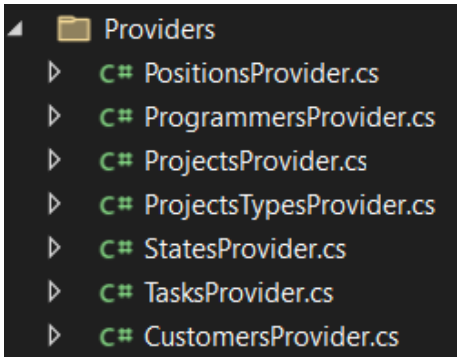

Рисунок 2.1 – Класи для роботи з БД

Для роботи з Базою даних реалізовуємо класи (рис. 2.1). В кожному класі є методи, за допомогою яких можна: вибирати, вставляти, редагувати та видаляти дані. Також є методи, які працюють з вибіркою даних із декількох таблиць.

## 2.2 Середовище розробки Visual Studio, переваги та недоліки

Microsoft Visual Studio - повнофункціональне інтегроване середовище розробки (IDE) за допомогою популярних мов програмування, серед яких C, C++, VB.NET, C#, F#, JavaScript, Python (рис. 2.2).

Функціональність Visual Studio охоплює всі етапи розробки програмного забезпечення, надаючи сучасні інструменти для написання коду, проектування графічних інтерфейсів, складання, налагодження та тестування програм. Можливості Visual Studio можуть бути доповнені шляхом підключення потрібних розширень.

Редактор коду Visual Studio підтримує підсвічування синтаксису, вставку фрагментів коду, відображення структури та пов'язаних функцій. Істотно прискорити роботу допомагає технологія IntelliSense – автозавершення коду у міру введення з клавіатури символів.

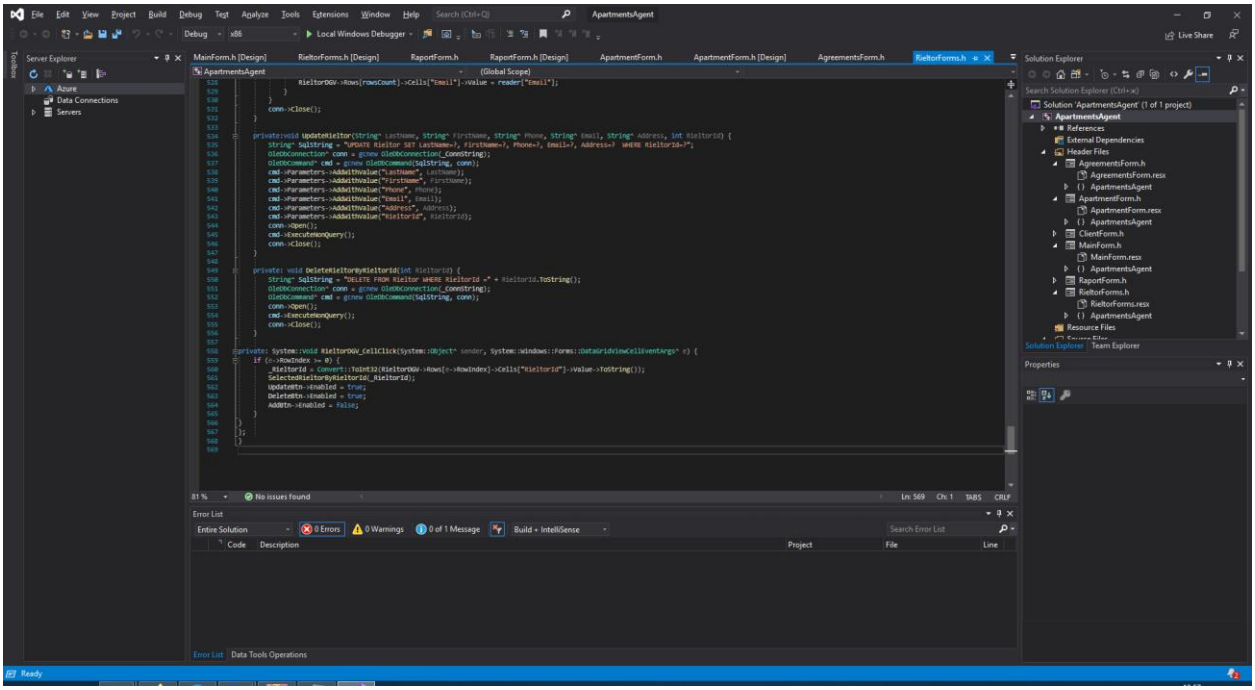

Рисунок 2.2 – Інтерфейс середовища розробки Visual Studio

Вбудований налагоджувач Visual Studio використовується для пошуку та виправлення помилок у вихідному коді, у тому числі на низькому апаратному рівні. Інструменти діагностики дозволяють оцінити якість коду з погляду продуктивності та використання пам'яті.

Дизайнер форм Visual Studio незамінний при розробці програм з графічним інтерфейсом, допомагає спроектувати зовнішній вигляд майбутньої програми та роботу кожного елемента інтерфейсу.

Для командних проектів Visual Studio пропонує підтримку групової роботи, дозволяючи виконувати спільне редагування та налагодження будь-якої частини коду в реальному часі, а як систему управління версіями використовувати Team Foundation або Git.

Основним розширенням файлу, асоційованим з Microsoft Visual Studio, є SLN - Visual Studio Solution File (Файл рішення Visual Studio), при відкритті якого в програму завантажуються всі дані та проекти, пов'язані з програмним рішенням, що розробляється.

Функціональна структура середовища включає:

– редактор вихідного коду, який включає безліч додаткових функцій, як автодоповнення IntelliSense, рефракторинг коду тощо;

– здійснити тестування програмного засобу.налагоджувач коду;

– редактор форм, призначений для спрощеного конструювання графічних інтерфейсів;

– веб-редактор;

– дизайнер класів;

– дизайнерсхем баз даних.

Visual Studio також дозволяє створювати та підключати сторонні доповнення (плагіни) для розширення функціональності.

Інтегроване середовище розробки (IDE) Visual Studio пропонує високорівневі функціональні можливості, що виходять за рамки базового керування кодом.

Нижче наведено основні переваги IDE-середовища Visual Studio:

– підтримка багатьох мов під час розробки. Visual Studio дозволяє писати код своєю мовою або будь-якими іншими мовами;

– менше за код для написання. Для створення більшості програм потрібна пристойна кількість стандартного стереотипного коду та Web-сторінки ASP. NET тому не виняток. Наприклад, додавання Web-елемента керування, приєднання обробників подій та коригування форматування потребує встановлення в розмітці сторінки ряду деталей. У Visual Studio такі деталі встановлюються автоматично;

– інтуїтивний стиль кодування. За промовчанням Visual Studio форматує код у міру його введення, автоматично вставляючи необхідні відступи та застосовуючи колірне кодування для виділення елементів типу коментарів;

– висока швидкість розробки. Зручні функції, на зразок функції IntelliSense, функції пошуку та заміни та функції автоматичного додавання та видалення коментарів, дозволяють розробнику працювати швидко та ефективно;

– можливості налагодження. Пропоновані Visual Studio інструменти налагодження є найкращим засобом для відстеження загадкових помилок і діагностування дивної поведінки;

– висока швидкість розробки програм для Microsoft Windows;

– низький поріг входження завдяки простому синтаксису мови;

– можливість компіляції як у машинний код, так і в P-код (на вибір програміста);

– безпека типів забезпечує захист від помилок, пов'язаних із застосуванням покажчиків та доступом до пам'яті;

– можливість використання більшості функцій WinAPI для розширення функціональних можливостей програми. Це питання найбільш повно досліджено Деном Епплманом, який написав книгу "Visual Basic Programmer's Guide to the Win32 API".

Недоліки:

– підтримка операційних систем лише сімейства Windows та Mac OS X (Виняток - VB1 for DOS);

– відсутність повноцінного механізму наслідування реалізації об’єктів. Існуюче у мові успадкування дозволяє успадковувати лише інтерфейси, але з їх реалізацію;

– практично всі вбудовані функції мови реалізовані через бібліотеку часу виконання, що в свою чергу сильно уповільнює швидкість роботи програм.

## 2.3 Аналіз вимог. Use-case діаграми. Основні прецеденти

Згідно з поставленою задачею можна висунути такий список вимог до проекту були створені таблиці 2.1 – Функціональні вимоги до додатку «Моніторинг робочого часу IT компанії», 2.2 – Актори та цілі додатку «Моніторинг робочого часу IT компанії», 2.3 – Опис варіантів використання додатка «Моніторинг робочого часу IT компанії» (див. додаток А таблиці).

З поставлених вимог (див. розділ 1) тепер можна виставити повний опис вимог із сценаріями, що будуть вхідними даними для візуального моделювання мовою UML.

UC1 Вивід каталогу замовників

Актор: користувач.

Ціль актора: вивід інформації про замовників.

Задіяний актор: база даних.

Передумова: користувач натиснув на пункт меню «Замовники».

Післяумова: система відображає екран для виведення списку всіх замовників.

UC2 Додати замовника

Актор: користувач.

Ціль актора: додати нового замовника.

Задіяний актор: база даних.

Передумова: користувач ввівши всю необхідну інформацію про замовника натискає на кнопку «Додати».

Післяумова: система додає нового замовника і відображає екран із списком всіх замовників.

UC3 Редагувати замовника

Актор: користувач.

Ціль актора: редагувати інформацію про вибраного замовника.

Задіяний актор: база даних.

Передумова: користувач вибирає необхідного із списку замовника.

Післяумова: система відображає екран для редагування інформації про вибраного замовника.

UC4 Вивід каталогу програмістів

Актор: користувач.

Ціль актора: вивести інформацію про програмістів.

Задіяний актор: база даних.

Передумова: користувач натиснув на пункт меню «Програмісти».

Післяумова: система відображає екран для виведення списку всіх програмістів.

UC5 Додати програміста

Актор: користувач.

Ціль актора: додати інформацію про нового програміста.

Задіяний актор: база даних.

Передумова: користувач ввівши всю необхідну інформацію про програміста натискає на кнопку «Додати».

Післяумова: система додає нового програміста і відображає екран із списком всіх програмістів.

UC6 Редагувати програміста

Актор: користувач.

Ціль актора: редагувати вибраного із списку програміста.

Задіяний актор: база даних.

Передумова: користувач вибирає необхідної із списку програміста.

Післяумова: система відображає екран для редагування інформації про вибраного програміста.

UC7 Вивід каталогу посад

Актор: користувач.

Ціль актора: вивести інформацію про всі посади.

Задіяний актор: база даних.

Передумова: користувач натиснув на пункт меню «Посади».

Післяумова: система відображає екран для виведення списку всіх посад.

UC8 Додати посаду

Актор: користувач.

Ціль актора: додати нову посаду.

Задіяний актор: база даних.

Передумова: користувач ввівши всю необхідну інформацію про посаду натискає на кнопку «Додати».

Післяумова: система додає нову посаду і відображає екран із списком всіх посад.

UC9 Редагувати посаду

Актор: користувач.

Ціль актора: редагувати вибрану із списку посаду.

Задіяний актор: база даних.

Передумова: користувач вибирає необхідну із списку посаду.

Післяумова: система відображає екран для редагування інформації про вибрану посаду.

UC10 Вивід каталогу команд

Актор: користувач.

Ціль актора: вивести інформацію про всі команди.

Задіяний актор: база даних.

Передумова: користувач натиснув на пункт меню «Команди».

Післяумова: система відображає екран для виведення списку всіх команд.

UC11 Додати команду

Актор: користувач.

Ціль актора: додати нову команду.

Задіяний актор: база даних.

Передумова: користувач ввівши всю необхідну інформацію про команду натискає на кнопку «Додати».

Післяумова: система додає нову команду і відображає екран із списком всіх команд.

UC12 Редагувати команду

Актор: користувач.

Ціль актора: редагувати вибрану із списку команду.

Задіяний актор: база даних.

Передумова: користувач вибирає необхідну із списку команду.

Післяумова: система відображає екран для редагування інформації про вибрану команду.

UC13 Вивід каталогу проектів

Актор: користувач.

Ціль актора: вивести інформацію про всі проекти.

Задіяний актор: база даних.

Передумова: користувач натиснув на пункт меню «Проекти».

Післяумова: система відображає екран для виведення списку всіх проектів.

UC14 Додати проект

Актор: користувач.

Ціль актора: додати новий проект.

Задіяний актор: база даних.

Передумова: користувач ввівши всю необхідну інформацію про проект натискає на кнопку «Додати».

Післяумова: система додає новий проект і відображає екран із списком всіх проектів.

UC15 Редагувати проект

Актор: користувач.

Ціль актора: редагувати вибраний із списку проект.

Задіяний актор: база даних.

Передумова: користувач вибирає необхідний із списку проект.

Післяумова: система відображає екран для редагування інформації про вибраний проект.

UC16 Вивід каталогу типів проекту

Актор: користувач.

Ціль актора: вивести інформацію про всі типи проекту.

Задіяний актор: база даних.

Передумова: користувач натиснув на пункт меню «Типи проекту».

Післяумова: система відображає екран для виведення списку всіх типи проекту.

UC17 Додати проект

Актор: користувач.

Ціль актора: додати новий тип проекту.

Задіяний актор: база даних.

Передумова: користувач ввівши всю необхідну інформацію про тип проекту натискає на кнопку «Додати».

Післяумова: система додає новий тип проекту і відображає екран із списком всіх типів проекту.

UC18 Редагувати тип проекту

Актор: користувач.

Ціль актора: редагувати вибраний із списку тип проекту.

Задіяний актор: база даних.

Передумова: користувач вибирає необхідний із списку тип проекту.

Післяумова: система відображає екран для редагування інформації про вибраний тип проекту.

UC19 Розбити проект на задачі

Актор: користувач.

Ціль актора: розбити вибраний проект на задачі.

Задіяний актор: база даних.

Передумова: користувач натискає на пункт меню «Задачі».

Післяумова: система відображає екран з можливістю розбиття вибраного проекту на задачі.

UC20 Фіксація виконання задачі

Актор: користувач.

Ціль актора: зафіксувати виконання задачі.

Задіяний актор: база даних.

Передумова: користувач натискає на пункт меню «Фіксація виконання задач».

Післяумова: система відображає екран з можливістю фіксації виконання задачі із вказанням фактичної дати виконання.

UC21 Пошук проектів

Актор: користувач.

Ціль актора: здійснити пошук проектів, у яких брав участь заданий програміст. Вибір всіх завдань, у яких він був задіяний.

Задіяний актор: база даних.

Передумова: користувач натискає на пункт меню «По програмісту».

Післяумова: система відображає екран з можливістю здійснити пошук проектів у яких брав участь заданий програміст.

UC22 Звітність за проектами

Актор: користувач.

Ціль актора: створити звіт по вибраному проекту.

Задіяний актор: база даних.

Передумова: користувач вибирає пункт меню «Звіти» -> «По проекту».

Післяумова: система відображає звітність по вибраному із списку проекту (кількість поставлених завдань, кількість виконаних завдань, кількість програмістів, відсоток виконання).

UC23 Звітність по вибраному програмісту

Актор: користувач.

Ціль актора: створити звіт по вибраному програмісту.

Задіяний актор: база даних.

Передумова: користувач вибирає пункт меню «Звітність» -> «По програмісту».

Післяумова: система відображає звітність по вибраному програмісту за вказаний період часу проекти, завдання (термін виконання).

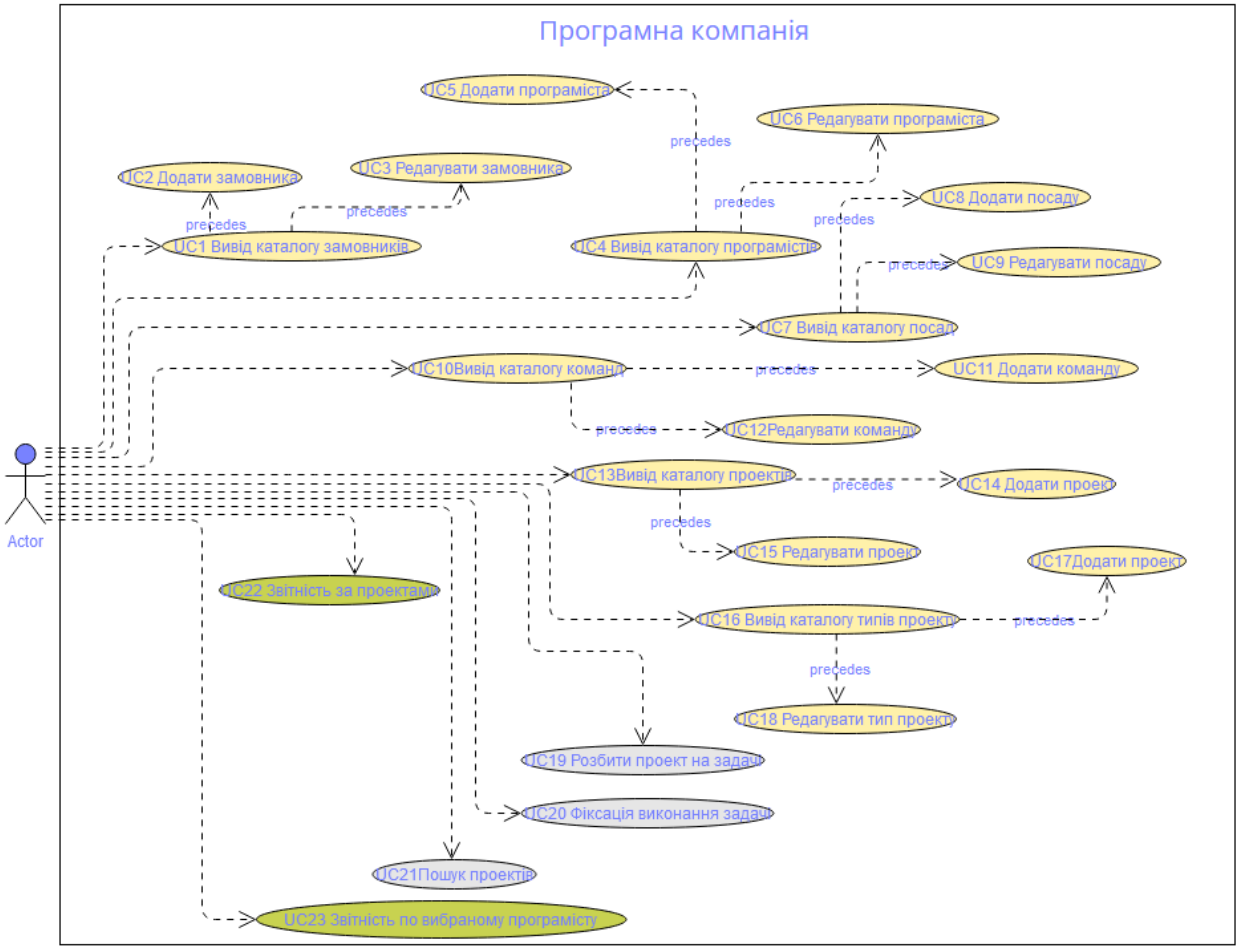


Рисунок 2.3 – Діаграма use-case

## 2.4 Архітектура проекту

Для розробки проекту було використано трирівневу архітектуру. Трирівнева архітектура (трьохланкова архітектура, англ. three-tier) — архітектурна модель програмного комплексу, що передбачає наявність у ньому трьох типів компонентів (рівнів, ланок): клієнтських додатків (з якими працюють користувачі), серверів додатків серверів баз даних (з якими працюють сервери додатків).

Рівень даних (DAL) – це шар доступу до даних. В неї входять 2 метода: «Серіалізація» і «Десеріалізація». Цей шар не знає про інші частини проекту, головна його задача – забезпечити зручну роботу з даними.

Бізнес-логіка (BLL) – це середній шар системи. Містить набір компонентів, які відповідають за обробку отриманих від рівня уявлень даних, реалізує всю необхідну логіку програми, всі обчислення, взаємодіє з базою даних та передає рівню представлення результат обробки.

Інтерфейс(PL) – це шар, що забезпечує візуальне представлення системи. Він посилається на шар логіки, використовуючи її методи при обробці подій вікон. Також цей шар реалізує контроль коректності введених даних.

На рис. 2.4 зображено трьохланкову архітектура додатку.

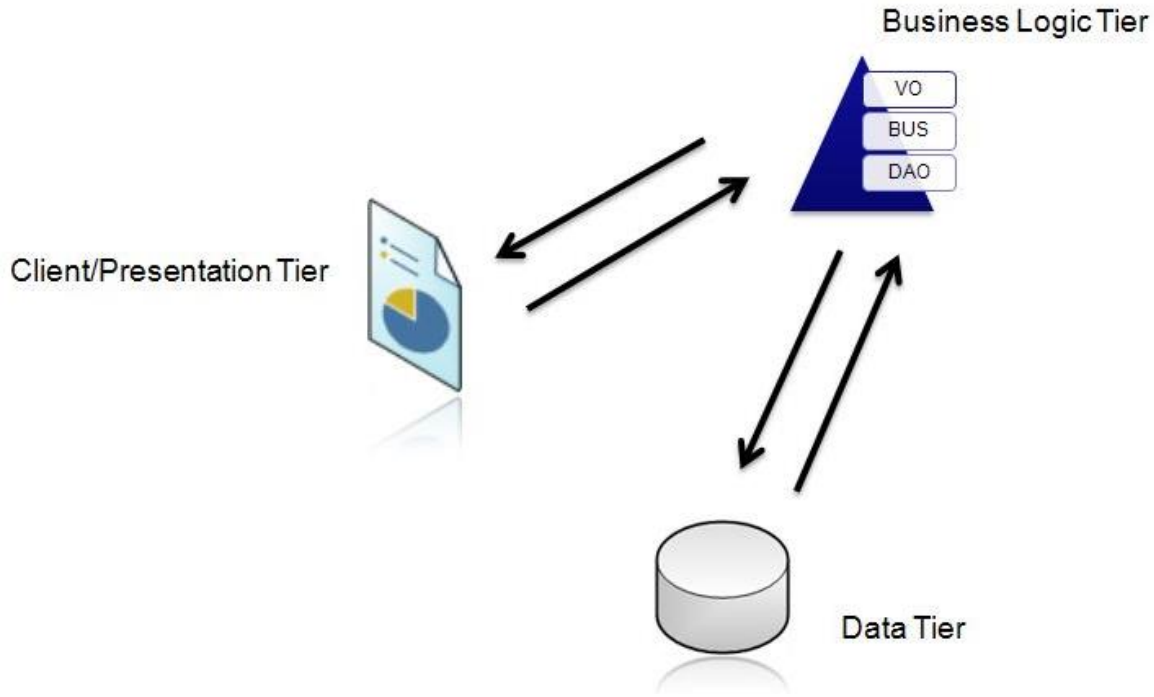


Рисунок 2.4 – Трьохланкова архітектура додатку

Як видно на рисунку 2.4, рівень даних не має контролю над рівнем уявлення, але є проміжний рівень, званий бізнес-рівнем, що несе головну відповідальність за передачу даних з рівня даних на рівень подання і додає задану бізнес-логіку дані.

Якщо виділяти кожен рівень за його функціональністю, то вийде наступний висновок: шар показує логічний поділ компонентів, таке як наявність окремих просторів імен та класів для рівня доступу до бази даних, рівня бізнес-логіки та рівня інтерфейсу користувача.

Рівень це сума всіх фізичних компонентів. Можна виділити три рівні – рівень даних, бізнес-рівень та рівень подання.

Можна сказати, що рис. 2.4 – суміш трирівневої та тришарової архітектури. Тут чітко видно різницю між рівнем та шаром. Оскільки компоненти не залежать один від одного, вони легко супроводжуються без зміни коду.

Цей підхід дійсно дуже важливий, особливо коли кілька розробників працюють над одним і тим самим проектом, і деякі модулі потрібно повторно використовувати в іншому проекті. У певному сенсі можна розподілити роботу між розробниками і супроводжувати її надалі без особливих проблем.

Особливості розробки бази даних. ERD діаграма з описанням сутностей.

Схема "сутність-зв'язок" (також ERD або ER-діаграма) - це різновид блок-схеми, де показано, як різні "сутності" (люди, об'єкти, концепції і так далі) пов'язані між собою всередині системи. ER-діаграми найчастіше застосовуються для проектування та налагодження реляційних баз даних у сфері освіти, дослідження та розробки програмного забезпечення та інформаційних систем для бізнесу.

ER-діаграми (або ER-моделі) покладаються на стандартний набір символів, включаючи прямокутники, ромби, овали та сполучні лінії для відображення сутностей, їх атрибутів та зв'язків. Ці діаграми влаштовані за тим самим принципом, як і граматичні структури: сутності виконують роль іменників, а зв'язку — дієслів.

ER-діаграми - "родичі" схем структури даних (DSD), де замість зв’язків між самими сутностями відображається відношення між елементами всередині них. ER-діаграми часто використовуються у поєднанні з діаграмами DFD, що схематично показують рух потоків інформації в рамках процесу або системи.

Отже, згідно завдання будуємо ERD діаграму бази даних додатку, яка зображена на рис. 2.5.

Як ми бачимо на рис. 2.5  база даних складається із 7 сутностей. Кожна сутність має свою таблицю, а саме:

– таблиця «Positions» – зберігає інформацію про всі посади;

– таблиця «Programmers» - зберігає інформацію про всіх програмістів;

– таблиця «Projects» - зберігає інформацію про проекти, що замовили у компанії;

– таблиця «ProjectsTypes» - зберігає інформацію про всі типи проектів;

– таблиця  «States» - зберігає інформацію про команди компанії;

– таблиця  «Tasks» - зберігає інформацію про задачі, на які розбивають проект;

– таблиця «Customers» - зберігає інформацію про всіх замовників компанії.

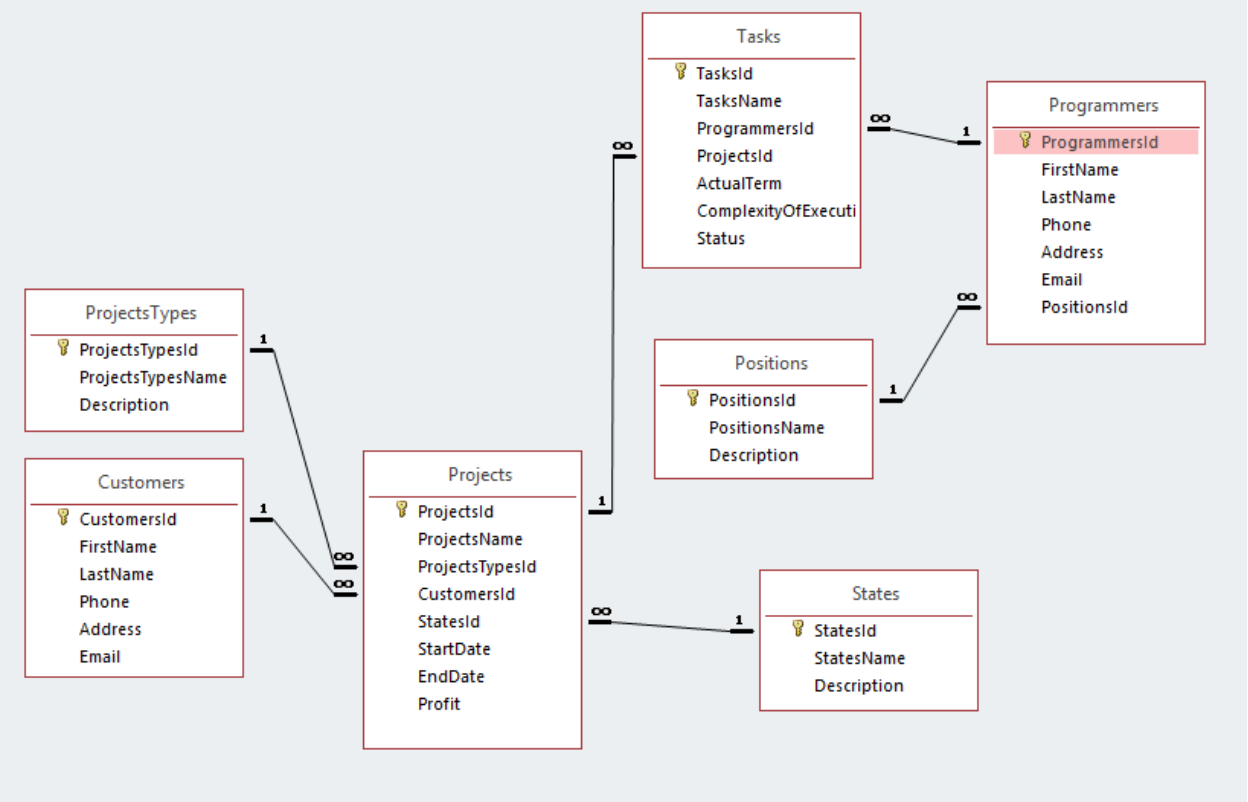


Рисунок 2.5 – ERD діаграма бази даних додатку

Особливості розробки рівня BLL.

Рівень BLL (Business Logic Layer) є важливою складовою архітектури програмного забезпечення і відповідає за бізнес-логіку додатку. Особливості розробки рівня BLL включають:

Розділення бізнес-логіки: Рівень BLL відокремлює бізнес-логіку від інтерфейсу користувача та доступу до даних. Це дозволяє зберігати код бізнес-логіки незалежним від конкретного типу інтерфейсу або бази даних, що спрощує тестування, підтримку та масштабування додатку.

Обробка бізнес-правил: Рівень BLL включає в себе реалізацію бізнес-правил, які визначають допустимі дії та обмеження в додатку. Це може включати перевірку даних, валідацію, авторизацію, обробку помилок та інші аспекти, необхідні для забезпечення правильної роботи системи.

Керування транзакціями: Рівень BLL може включати механізми керування транзакціями для забезпечення консистентності даних.

Інкапсуляція даних: Рівень BLL визначає, як дані повинні бути оброблені та доступні для інших компонентів системи. Він забезпечує інкапсуляцію

даних, що означає, що доступ до них відбувається через визначені методи та властивості, а не безпосередньо до полів об'єктів.

Повторне використання: Рівень BLL може бути повторно використаним у різних частинах додатку або навіть у різних проектах. Чиста бізнес-логіка може бути розроблена незалежно від конкретного інтерфейсу або платформи, що забезпечує його переносимість та масштабованість.

Враховуючи ці особливості, розробка рівня BLL стає важливим етапом при створенні програмного забезпечення, де бізнес-логіка відіграє ключову роль у забезпеченні правильної та ефективної роботи додатку.

BLL буде обробляти речі, які є частиною бізнес-сфери, а не частиною бази даних і не частиною користувальницького інтерфейсу.

У проекті реалізовано класи для пошуку інформації та формування звітів. Діаграма класів бізнес логіки зображена на рис. 2.6.

Як можна побачити із рис. 2.6, у класі «RaportBLL» реалізовано 3 методи, а саме:

– метод «GetSearchProgrammers» призначений для формування звітності по співробітнику;

– метод «GetProjectRaport» призначений для формування датального протоколу по вибраному проекту IT-компанії.

Особливість реалізації бізнес логіки – діаграма домена.

Будуємо діаграму класів домена проекту. Кожен клас описує конкретну таблицю бази даних для зручного опрацювання даних.

Діаграма домена є однією з особливостей реалізації бізнес-логіки в програмному забезпеченні. Вона використовується для візуалізації ключових понять, об'єктів та їх взаємодії в конкретній області бізнесу.

Особливість діаграми домена включає наступні аспекти.

Візуалізація концепцій: Діаграма домена дозволяє відобразити основні концепції та об'єкти, які існують у конкретній області бізнесу.

Моделювання взаємодії: Діаграма домена відображає взаємодію між об'єктами в системі. Вона показує залежності, асоціації, агрегації та інші типи взаємодій, що допомагає розібратися у структурі та поведінці системи.

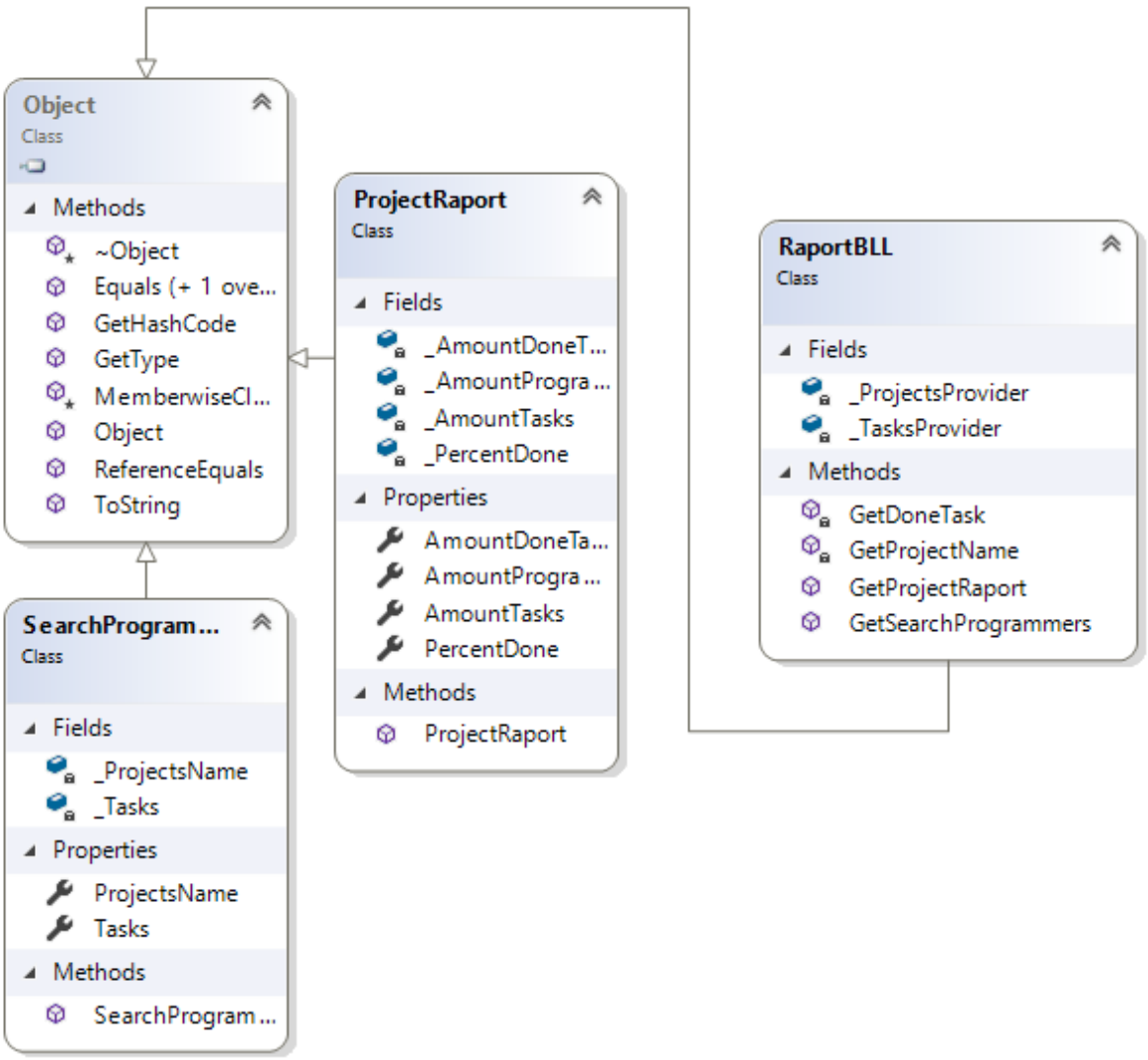


Рисунок 2.6 – Діаграма класу BLL «Моніторинг робочого часу IT компанії»

Зрозумілість та спільне розуміння: Діаграма домена є ефективним засобом комунікації між розробниками, дизайнерами та зацікавленими сторонами. Вона допомагає уникнути непорозумінь та забезпечує спільне розуміння бізнес-логіки та вимог до системи.

Спрощення розробки: Діаграма домена слугує основою для подальшої реалізації бізнес-логіки в програмному забезпеченні. Вона допомагає розробникам уявити структуру системи, спрощує процес проектування та реалізації.

Діаграма домена є потужним інструментом при розробці бізнес-логіки, який допомагає зрозуміти та візуалізувати ключові концепції та взаємодію об'єктів в конкретній області бізнесу.

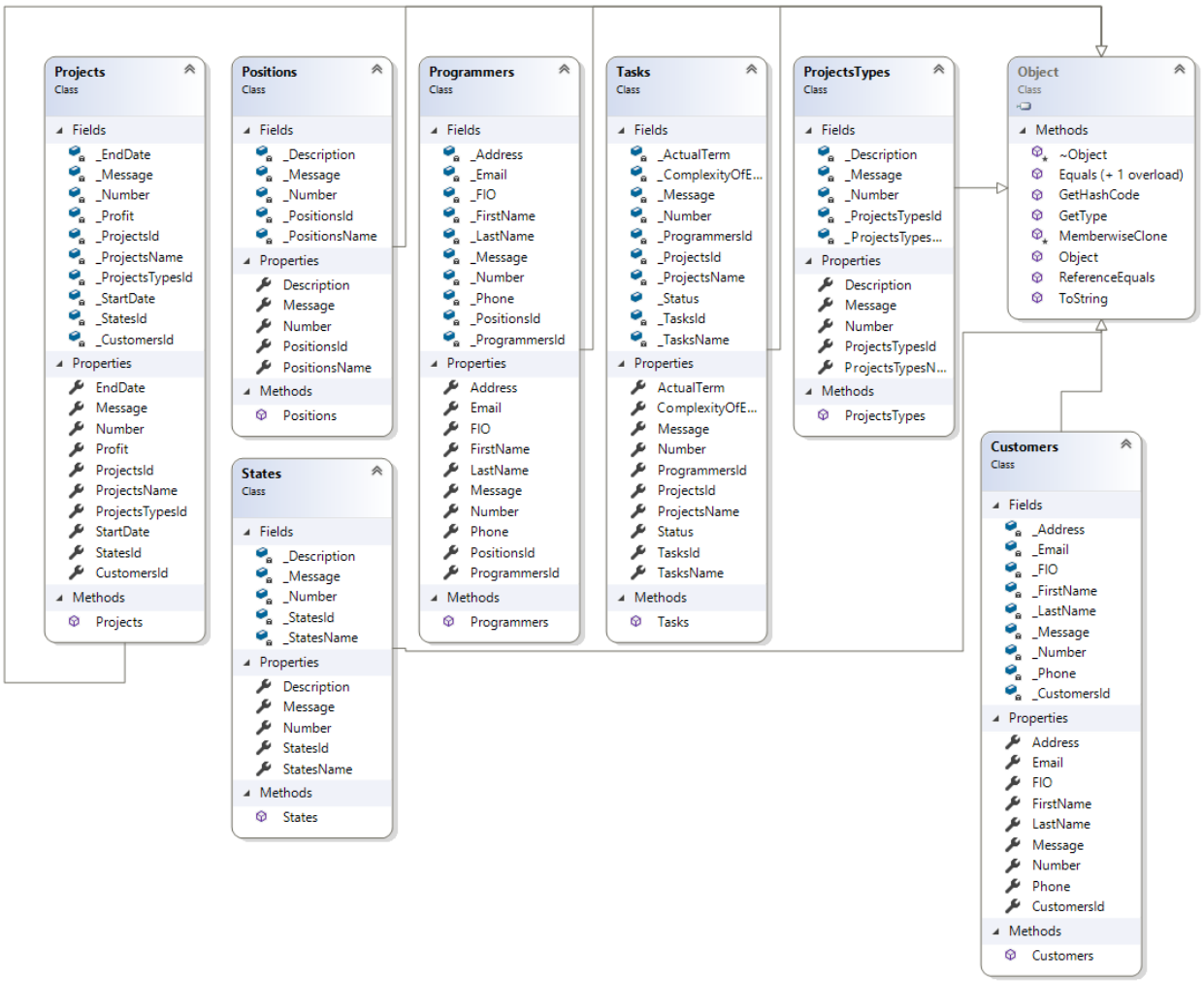


Рисунок 2.7 – Діаграма класу домена

Особливості розробки рівня UI.

В даному проекті для розробки користувацького інтерфейсу я використав Windows Forms.

Windows Forms - це платформа користувача інтерфейсу для створення класичних додатків Windows. Вона забезпечує один з найефективніших способів створення класичних додатків за допомогою візуального конструктора в Visual Studio.

Рівень UI (User Interface) відповідає за взаємодію користувача з програмою. Особливості розробки рівня UI включають:

Візуальний дизайн: Рівень UI вимагає розробки естетичного та зручного для використання інтерфейсу. Це включає в себе вибір кольорів, шрифтів, іконок та компонентів, які створюють приємне та зрозуміле враження для користувачів.

Взаємодія з користувачем: Рівень UI повинен забезпечувати зручну та інтуїтивно зрозумілу взаємодію з користувачем. Це означає використання зручних управлінських елементів, таких як кнопки, поля введення, списки тощо, а також належну обробку введених даних та змін користувача.

Респонсивний дизайн: Розробка UI повинна враховувати різні розміри екранів та пристроїв, що використовуються користувачами. Респонсивний дизайн забезпечує адаптивність інтерфейсу до різних екранних розмірів, що дозволяє користувачам зручно використовувати програму навіть на мобільних пристроях.

Валідація та обробка даних: Рівень UI має забезпечити валідацію введених користувачем даних та їх належну обробку. Це включає перевірку правильності формату, обов'язкових полів, введення числових значень тощо. Додатково, рівень UI може включати механізми обробки помилок та повідомлення користувачам про виниклі проблеми.

Тестування інтерфейсу: Розробка рівня UI вимагає належного тестування, щоб переконатися в правильній роботі всіх компонентів та функціональності. Тестування може включати перевірку правильності розміщення елементів, коректність відображення даних, функціональність кнопок та інші аспекти.

Загалом, розробка рівня UI вимагає уваги до деталей, зручності та ефективності взаємодії з користувачем. Це допомагає створити приємний та продуктивний досвід використання програми.

На рис. 2.8 показана діаграма додатку «Програмна компанія».

Програма складається з дев’ятнадцяти класів рівня UI та є похідними від класу Form, тобто мають графічний інтерфейс.

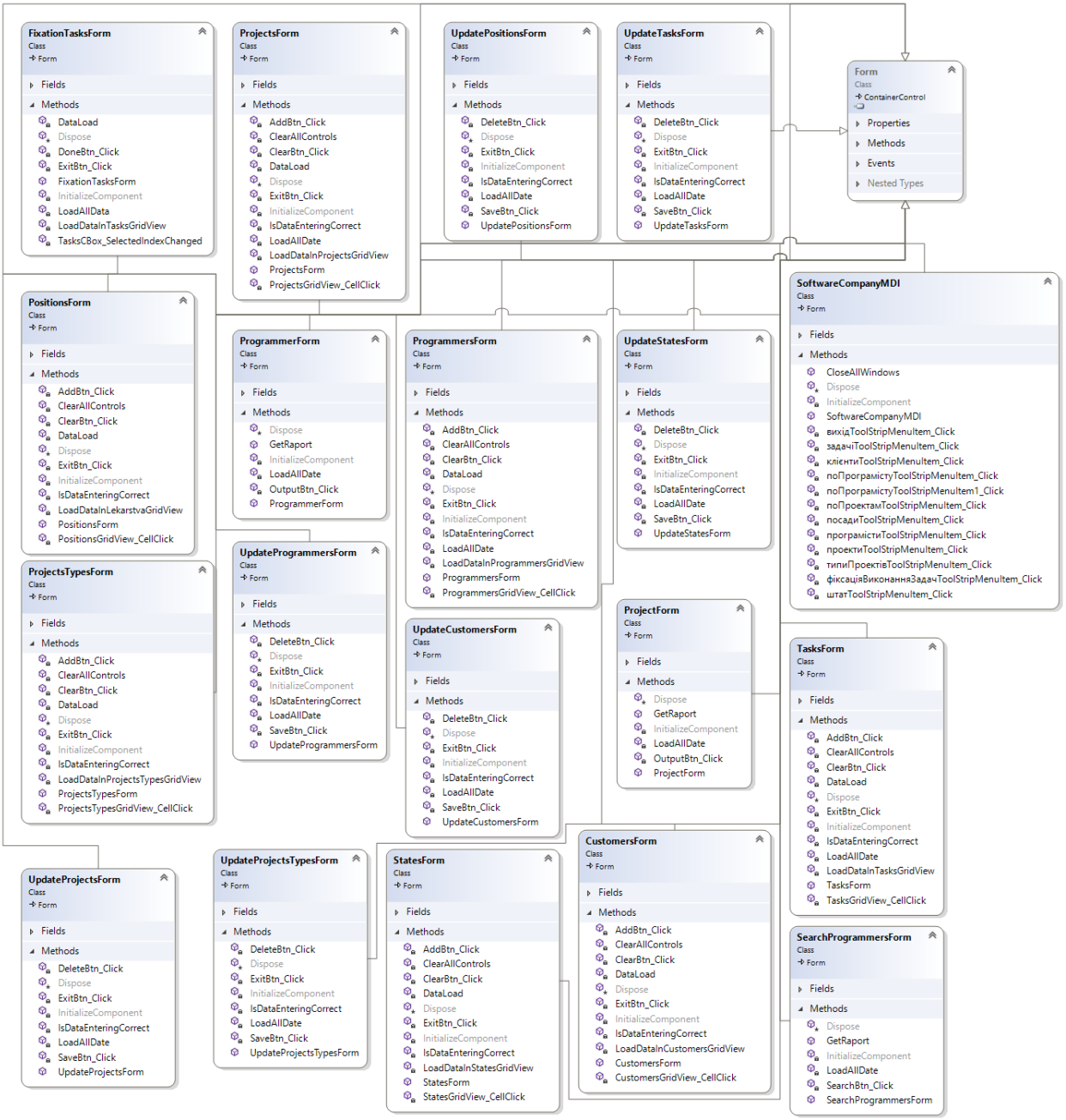


Рисунок 2.8 – Діаграма інтерфейсу додатку «Програмна компанія»

Клас «SoftwareCompanyMDI» становить головне вікно програми. Найважливішим елементом управління в головному вікні є розташування головного меню, за допомогою якого можна відкривати інші форми та вийти з програми.

Клас «PositionsForm» призначений для опрацювання даних про посади у компанії.

Клас «UpdatePositionsForm» призначений для редагування інформації про вибрану посаду.

Клас «ProgrammersForm» призначений для опрацювання даних про програмістів компанії.

Клас «UpdateProgrammersForm» призначений для редагування інформації вибраного програміста.

Клас «StatesForm» призначений для опрацювання даних команд програмістів компанії.

Клас «UpdateStatesForm» призначений для редагування інформації вибраної команди.

Клас «ProjectsTypesForm» призначений для опрацювання даних типів проекту.

Клас «UpdateProjectsTypesForm» призначений для редагування інформації про вибраний тип проекту.

Клас «CustomersForm» призначений для опрацювання даних про замовників.

Клас «UpdatePostsForm» призначений для редагування інформації вибраного замовника.

Клас «ProjectsForm» призначений для опрацювання даних по проектах.

Клас «UpdateProjectsForm» призначений для редагування інформації вибраного із списку проекта.

Клас «TasksForm» призначений для опрацювання даних про задачі проекту.

Клас «UpdateTasksForm» призначений для редагування інформації про вибрану задачу проекту.

Клас «FixationTasksForm» призначений для фіксації виконання поставленої задачі.

Клас «ProgrammerForm» призначений для формування звітності про вибраного програміста.

Клас «ProjectForm» призначений для формування звітності про вибраний проект.

Клас «SearchProgrammersForm» дозволяє користувачу здійснювати пошук проектів, у яких брав участь заданий програміст. Вибір всіх завдань, у яких він був задіяний.

Створення об'єктів інших команд та виклик їх методів відбувається в результаті взаємодії користувача з елементами графічного інтерфейсу програми.

Особливості розробки DAL.

Для роботи з базою даних було реалізовано 7 класів. Назва кожного класу відповідає назві таблиці в базі даних. Для пошуку інформації по певних критеріях та формування звітів були реалізовані методи. Діаграма даного шару розробки показана на рис. 2.9.

DAL (Data Access Layer) відповідає за доступ до даних у програмному забезпеченні. Особливості розробки DAL включають.

Взаємодія з джерелами даних: DAL забезпечує роботу з різними джерелами даних, такими як бази даних, веб-сервіси, файлові системи тощо. Це включає в себе встановлення з'єднання з джерелом даних, виконання запитів і отримання результатів.

Керування транзакціями: DAL може включати механізми керування транзакціями для забезпечення консистентності даних під час взаємодії з джерелами даних. Це важливо, особливо коли виконуються багатоопераційні запити або здійснюється збереження даних в розподілених системах.

Маппінг даних: DAL здійснює маппінг даних з формату, який використовується в джерелі даних, на формат, зрозумілий для решти програми. Це включає в себе перетворення структури даних, типів даних, а також обробку перетворень, необхідних для забезпечення сумісності даних.

Кешування даних: DAL може включати механізми кешування даних для забезпечення швидкодії та ефективності доступу до даних.

Тестування доступу до даних: Розробка DAL вимагає належного тестування, щоб переконатися в правильному доступі до даних, обробці помилок та відповідності до вимог до системи. Тестування може включати

перевірку правильності виконання запитів, валідацію даних та перевірку функціональності.

Розробка DAL вимагає уваги до ефективного доступу до даних, безпеки, керування транзакціями та оптимального використання ресурсів.

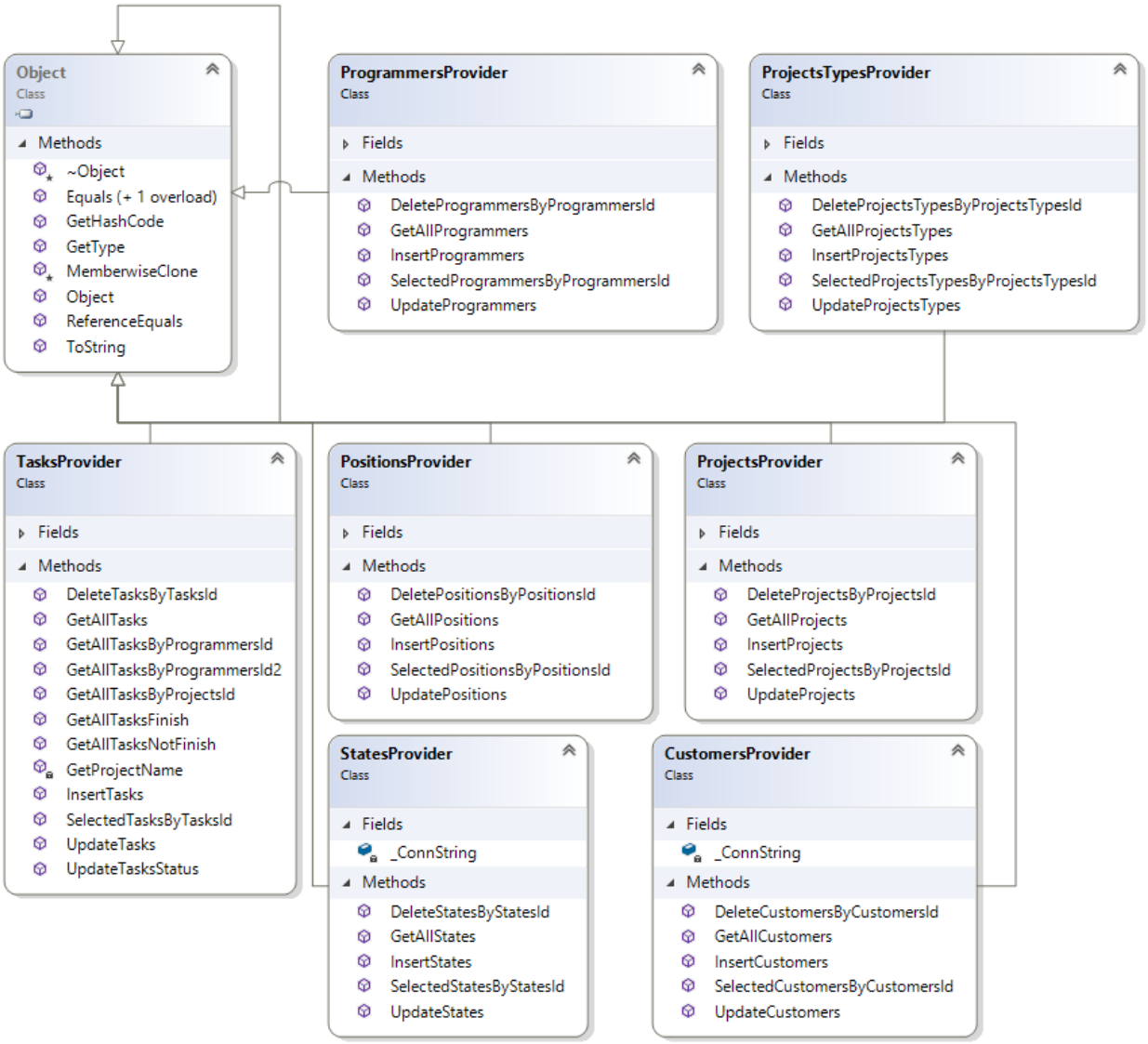


Рисунок 2.9 – Діаграма класів з методами для роботи з базою даних

Доцільним буде провести опис призначення кожного класу:

Клас «PositionsProvider» містить 5 методів для додавання нових позицій у компанії, редагування даних про позицію, вибірки списку всіх позицій із бази та видалення позиції із бази даних.

Клас «ProjectsTypesProvider» містить 5 методів і є аналогічним класу «PositionsProvider».

Клас «CustomersProvider» містить 5 методів і є аналогічним класу «PositionsProvider».

Клас «StatesProvider» містить 5 методів і є аналогічним класу «PositionsProvider».

Клас «ProjectsProvider» містить 5 методів, а саме:

– метод «InsertProjects» призначений для додавання нового проекту в базу даних;

– метод «GetAllProjects» призначений для вибірки всіх проектів із бази даних;

– метод «SelectedProjectsByProjectsId» призначений для вибірки проекту за його ідентифікатором;

– метод «UpdateProjects» призначений для редагування інформації про вибраний проект;

– метод «DeleteProjectsByProjectsId» призначений для видалення інформації про вибраний проект.

Клас «TaskProvider» містить 12 методів, а саме:

– метод «InsertTask» призначений для додавання нової задачі до проекту;

– метод «GetAllTaskNotFinish» призначений для вибірки всіх незакінчених задач;

– метод «GetAllTaskFinish» призначений для вибірки всіх закінчених задач;

– метод «GetAllTask» призначений для вибірки всіх задач;

– метод «GetAllTaskByProgrammersId» призначений для вибірки всіх проектів, у яких брав участь певний програміст;

– метод «GetAllTaskByProgrammersId2» призначений для вибірки всіх задач, у яких брав участь певний програміст;

– метод «GetAllTaskByProjectsId» призначений для вибірки всіх задач одного прокту;

– метод «SelectedTaskByTaskId» призначений для вибірки задачі по її ідентифікатору;

– метод «UpdateTask» призначений для редагування інформації вибраної задачі;

– метод «UpdateTaskStatus» призначений для зміни статусу виконання задачі;

– метод «DeleteTaskByTaskId» призначений для видалення інформації вибраної задачі.

Клас «StatesProvider» містить 5 методів і є аналогічним класу «PositionsProvider».

Клас «ProgrammersProvider» містить 5 методів і є аналогічним класу «PositionsProvider».

Побудова сучасних інформаційних система займає дуже багато часу. Вона починається з аналізу предметної області, класного планування системи, описання вимог, моделювання її поведінки за допомогою uml діаграм. Визначається шаблони, які будуть використовуватись при розробці.

Після планування починається стадія розробки. Розробка починається зі створення користувацького інтерфейсу і закінчується базою даних.

При розробці слід враховувати, що вимоги змінюються швидко і потрібно будувати систему так, щоб вона була гнучкою.

Розробка системи виконується по окремим компонентам. Кожний створений компонент потрібно класно тестувати, щоб мінімізувати помилки на наступних етапах розробки.

Якщо дотримуватися всіх вищеперерахованих вимог, то можна побудувати гнучку і стійку інформаційну систему.

# 3 РОЗРОБКА, ВИКОРИСТАННЯ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## 3.1 Розробка програмних модулів системи

В даному підрозділі наведений код роботи програми, що був використаний для реалізації проекту. Для зменшення кількості коментарів у програмі всі назви класів, об’єктів у програмі мають зрозумілі назви.

Після постановки задачі, маючи всі необхідні дані, проводимо детальний аналіз роботи ІТ компанії. Спочатку засобами СУБД було побудовано модель бази даних

Сучасні бази даних зберігають великі об’єми інформації, тому обробляти їх вручну переглядаючи та редагуючи дані в таблицях, стає дуже важко важко, тому для підвищення ефективності користування базами даних застосовують запити.

Запит– це засіб отримання інформації з бази даних.

Для написання запитів було використано мову SQL (Structured Query Language).

В базі даних, було розроблено необхідні запити на додавання інформації до таблиць, редагування та видалення. Всі запити були реалізовані безпосередньо із розробленого додатку.

Після того, як базу даних було створено під’єднаємо її за допомогою засобів середовища Visual Studio 2019 для розробки подальшої системи. Усі таблиці пізніше будуть підключатись до форм, на яких можливе додавання, редагування та видалення даних.

Для під’єднання БД до середовища Microsoft Visual Studio 2019 у файлі проекту "App.config" створюємо змінну "CONNECT" та задаємо значення параметрів (лістинг 3.1).

Лістинг 3.1 Параметри змінної "CONNECT"

```
<add key="CONNECT" value="Provider=Microsoft.Jet.OLEDB.4.0;Data Source=|DataDirectory|\DataBase.mdb;"/>
```

Для роботи з створеною базою даних використовуємо простір імен System.Data. OleDb.

Для того, щоб створити меню, додаємо елемент menuStrip у головне вікно проекту (рис. 3.1).

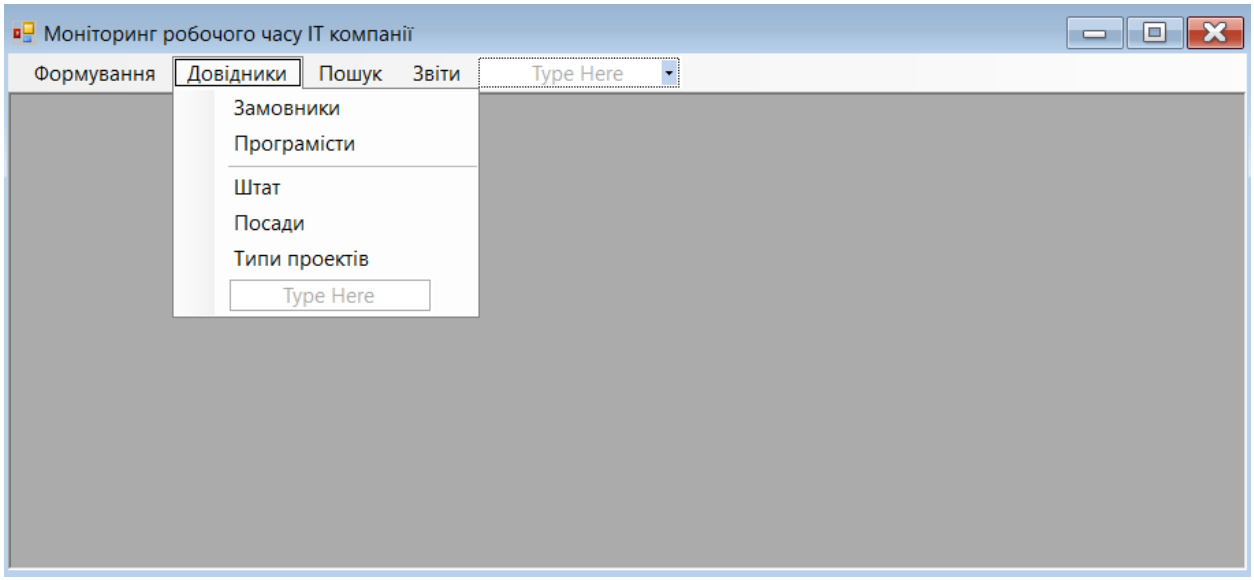


Рисунок 3.1 – Головне меню програми

На кожен із пунктів меню додано певний код, який відповідає за створення екземплярів форм та закриття попередньо відкритого вікна. Аналогічний код застосовуємо для всіх пунктів меню (рис. 3.2).

```
1 reference
private void проектиToolStripMenuItem_Click(object sender, EventArgs e) {
    CloseAllWindows();
    ProjectsForm projectsForm = new ProjectsForm();
    projectsForm.MdiParent = this;
    projectsForm.WindowState = FormWindowState.Maximized;
    projectsForm.Show();
}
```

Рисунок 3.2 – Програмування пунктів меню

Для роботи з базою даних було створено відповідні класи, всі вони розміщені у папці із назвою «Providers» в проекті рішення.

Для з’єднання з базою даних використовується метод «Open» екземпляру класу «OleDbConnection». Для закриття з’єднання з базою даних використовується метод «Close» того ж екземпляру класу.

Для опрацювання даних про проекти був створений клас «ProjectsProvider», який містить в собі 5 публічних методів для опрацювання інформації: додавання інформації про новий проекти, вибірка всіх проектів, вибірка конкретного проекту по його ідентифікатору, редагування вибраного проекту із списку, видалення інформації про проект та ін.

Код методу для додавання проекту представлений на рис. 3.3.

```
1 reference
public void InsertProjects(string ProjectsName, int ProjectsTypesId, int CustomersId,
  int StatesId, DateTime StartDate, DateTime EndDate,
  double Profit) {
  string SqlString = "INSERT INTO Projects (ProjectsName, ProjectsTypesId, CustomersId, " +
    "StatesId, StartDate, EndDate, Profit) " +
    "Values(?, ?, ?, ?, ?, ?, ?)";

  using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
    using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
      cmd.CommandType = CommandType.Text;
      cmd.Parameters.AddWithValue("ProjectsName", ProjectsName);
      cmd.Parameters.AddWithValue("ProjectsTypesId", ProjectsTypesId);
      cmd.Parameters.AddWithValue("CustomersId", CustomersId);
      cmd.Parameters.AddWithValue("StatesId", StatesId);
      cmd.Parameters.AddWithValue("StartDate", StartDate.ToString());
      cmd.Parameters.AddWithValue("EndDate", EndDate.ToString());
      cmd.Parameters.AddWithValue("Profit", Profit);
      conn.Open();
      cmd.ExecuteNonQuery();
      conn.Close();
    }
  }
}
```

Рисунок 3.3 – Метод «InsertProjects» для додавання проекту

Код методу для редагування даних проекту показаний на рис. 3.4.

```
1 reference
public void UpdateProjects(string ProjectsName, int ProjectsTypesId, int CustomersId,
  int StatesId, DateTime StartDate, DateTime EndDate,
  double Profit, int ProjectsId) {
  string SqlString = "UPDATE Projects SET ProjectsName=?, ProjectsTypesId=?, CustomersId=?," +
    " StatesId=?, StartDate=?, EndDate=?, Profit=? " +
    "WHERE ProjectsId=?";

  using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
    using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
      cmd.CommandType = CommandType.Text;
      cmd.Parameters.AddWithValue("ProjectsName", ProjectsName);
      cmd.Parameters.AddWithValue("ProjectsTypesId", ProjectsTypesId);
      cmd.Parameters.AddWithValue("CustomersId", CustomersId);
      cmd.Parameters.AddWithValue("StatesId", StatesId);
      cmd.Parameters.AddWithValue("StartDate", StartDate.ToString());
      cmd.Parameters.AddWithValue("EndDate", EndDate.ToString());
      cmd.Parameters.AddWithValue("Profit", Profit);
      cmd.Parameters.AddWithValue("ProjectsId", ProjectsId);
      conn.Open();
      cmd.ExecuteNonQuery();
      conn.Close();
    }
  }
}
```

Рисунок 3.4 – Метод «UpdateProjects» для редагування інформації проекту

Код методу для вибірки всіх проектів представлений на рис. 3.5.

```
6 references
public List<Projects> GetAllProjects() {
  int i = 0;
  string SqlString = "SELECT * FROM Projects ORDER BY ProjectsName ASC";
  List<Projects> listAllProjects = new List<Projects>();
  using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
    using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
      conn.Open();
      using (OleDbDataReader reader = cmd.ExecuteReader()) {
        while (reader.Read()) {
          Projects oneProjects = new Projects();
          oneProjects.Number = ++i;
          oneProjects.ProjectsId = Convert.ToInt32(reader["ProjectsId"]);
          oneProjects.ProjectsName = reader["ProjectsName"].ToString();
          oneProjects.ProjectsTypesId = Convert.ToInt32(reader["ProjectsTypesId"]);
          oneProjects.CustomersId = Convert.ToInt32(reader["CustomersId"]);
          oneProjects.StatesId = Convert.ToInt32(reader["StatesId"]);
          oneProjects.StartDate = Convert.ToDateTime(reader["StartDate"]);
          oneProjects.EndDate = Convert.ToDateTime(reader["EndDate"]);
          oneProjects.Profit = Convert.ToDouble(reader["Profit"]);
          listAllProjects.Add(oneProjects);
        }
      }
      conn.Close();
    }
  }
  if (listAllProjects.Count == 0) {
    Projects noProjects = new Projects();
    noProjects.ProjectsId = 0;
    noProjects.Message = NamesMy.NoDataNames.NoDataInProjects;
    listAllProjects.Add(noProjects);
  }
  return listAllProjects;
```

Рисунок 3.5 – Метод «GetAllProjects» для вибірки всіх проектів

Код для вибору проекту із списку показаний на рис. 3.6.

```
1 reference
public Projects SelectedProjectsByProjectsId(int ProjectsId) {
  string SqlString = "SELECT * FROM Projects " +
    "Where ProjectsId=" + ProjectsId.ToString();

  Projects oneProjects = new Projects();
  using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
    using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
      conn.Open();
      using (OleDbDataReader reader = cmd.ExecuteReader()) {
        while (reader.Read()) {
          oneProjects.ProjectsId = Convert.ToInt32(reader["ProjectsId"]);
          oneProjects.ProjectsName = reader["ProjectsName"].ToString();
          oneProjects.ProjectsTypesId = Convert.ToInt32(reader["ProjectsTypesId"]);
          oneProjects.CustomersId = Convert.ToInt32(reader["CustomersId"]);
          oneProjects.StatesId = Convert.ToInt32(reader["StatesId"]);
          oneProjects.StartDate = Convert.ToDateTime(reader["StartDate"]);
          oneProjects.EndDate = Convert.ToDateTime(reader["EndDate"]);
          oneProjects.Profit = Convert.ToDouble(reader["Profit"]);
        }
      }
    }
    conn.Close();
  }
  return oneProjects;
}
```

Рисунок 3:6 – Метод «SelectedProjectsByProjectsId» для вибору проекту із списку

Для видалення даних про проекту був розроблений метод «DeleteProjectsByProjectsId», що показаний на рис. 3.7.

```
1 reference
public void DeleteClientByClientId(int ClientId) {
    string SqlString = "DELETE FROM Client WHERE ClientId=" + ClientId.ToString();
    using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
        using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
            conn.Open();
            cmd.ExecuteNonQuery();
            conn.Close();
        }
    }
}
```

Рисунок 3:7 – Метод «DeleteProjectsByProjectsId» для видалення інформації вибраного із списку проекту

Наступним кроком було створення форми для опрацювання інформації про проекти IT-компанії (рис. 3.8).

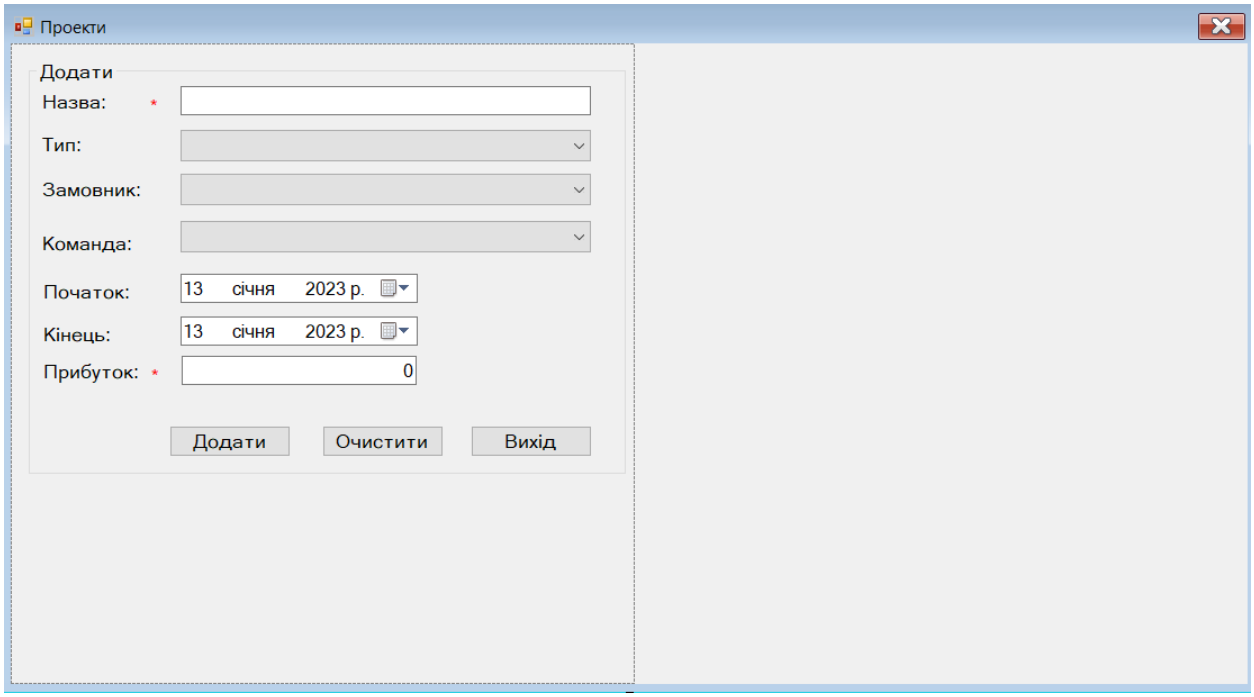


Рисунок 3.8 – Форма для опрацювання інформації про проекти IT-компанії

При завантаженні форми про у конструкторі форми викликається метод «LoadAllDate», який завантажує дані про клієнтів, штати працівників та типи проектів у випадаючі списки (рис. 3.9).

```
1 reference
private void LoadAllDate() {
    _CustomersList = _CustomersProvider.GetAllCustomers();
    CustomersCBox.DataSource = _CustomersList;
    CustomersCBox.ValueMember = "CustomersId";
    CustomersCBox.DisplayMember = "FIO";

    _StatesList = _StatesProvider.GetAllStates();
    StatesCBox.DataSource = _StatesList;
    StatesCBox.ValueMember = "StatesId";
    StatesCBox.DisplayMember = "StatesName";

    _ProjectsTypesList = _ProjectsTypesProvider.GetAllProjectsTypes();
    ProjectsTypesCBox.DataSource = _ProjectsTypesList;
    ProjectsTypesCBox.ValueMember = "ProjectsTypesId";
    ProjectsTypesCBox.DisplayMember = "ProjectsTypesName";
}
```

Рисунок 3.9 – Метод для заповнення даних у випадаючі списки

Для додавання інформацію про новий проект у формі вікна реалізовано метод «AddBtn_Click», що спрацьовує на натискання кнопки «Додати» (рис. 3.10).

```
1 reference
private void AddBtn_Click(object sender, EventArgs e) {
    if (IsDataEnteringCorrect()) {
        _ProjectsProvider.InsertProjects(ProjectsNameTBox.Text,
            Convert.ToInt32(ProjectsTypesCBox.SelectedValue),
            Convert.ToInt32(CustomersCBox.SelectedValue),
            Convert.ToInt32(StatesCBox.SelectedValue),
            StartDateDTP.Value, EndDateDTP.Value,
            Convert.ToDouble(ProfitTBox.Text));
        ClearAllControls();
        DataLoad();
    }
}
```

Рисунок 3.10 – Метод для додавання інформації про новий проект

Як можна побачити, в методі на рис. 3.10 викликається метод «IsDataEnteringCorrect», що призначений для перевірки даних на правильність введення, для подальшого їх зберігання в базі даних. Код методу «IsDataEnteringCorrect» показаний на рис. 3.11.

```
private bool IsDataEnteringCorrect() {
    bool isCorrect = true;
    if (_validation.IsDataEntering(ProjectsNameTBox.Text)) {
        ProjectsNameValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
    } else {
        ProjectsNameValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
    }
    if (_validation.IsDataConvertToDouble(ProfitTBox.Text)) {
        ProfitValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
    } else {
        ProfitValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
    }
    return isCorrect;
}
```

Рисунок 3.11 – Метод  для перевірки введених даних

При потребі  можна видалити інформацію про реєстрацію. За видалення відповідає подія, яка викликається у формі «UpdateProjectsForm», та спрацьовує після натискання лівою кнопкою мишки по кнопці «Видалити». Після вибору запису появляється повідомлення, де необхідно підтвердити видалення (рис. 3.12).

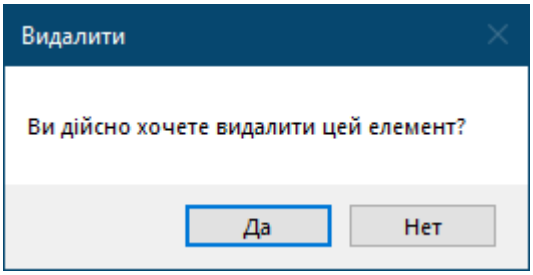


Рисунок 3.12 – Підтвердження видалення

Код події на видалення показаний на рис. 3.13.

```
1 reference
private void DeleteBtn_Click(object sender, EventArgs e) {
    if (MessageBox.Show("Ви дійсно хочете видалити цей елемент?", "Видалити",
        MessageBoxButtons.YesNo) == DialogResult.Yes) {
        _ProjectsProvider.DeleteProjectsByProjectsId(_ProjectsId);
        this.Close();
    }
}
```

Рисунок 3.13 – Подія «DeleteBtn_Click» для видалення проекту

Отже, в даному розділі було проведено частковий опис основним моментів функціонування розробленого програмного забезпечення.

## 3.2 Результати функціонального тестування розробленого додатку

Тестування — це процес аналізу та дослідження, який надає змогу виявити інформацію про якість продукту відносно умов, в яких він буде застосовуватись. Методика тестування також включає в себе процес пошуку дефектів, помилок, несправностей. Також це є випробуванням програмних складових з метою оцінити готовність програмного продукту до використання. Результат тестування оцінюється за наступними критеріями:

– відповідність вимогам, які надавалися розробниками та проектувальниками;

– відповідність вихідних даних;

– прийнятний час виконання функцій;

– практичність;

– відповідність вимогам замовника.

Кількість тестів навіть для простих програмних компонентів може бути ледь не нескінченним, тому тактика тестування має полягати в тому, що будуть проведені тільки необхідні тести з урахуванням доступного часу та ресурсів. Як результат, програмні засоби тестуються стандартним виконанням програми з метою виявлення багів, помилок або інших дефектів.

Існує багато видів тестування: одні зазвичай виконують самі розробники, а інші спеціалізовані групи. В нашому ж випадку буде використовуватись тестування системи.

Тестування системи — це виконання програмного забезпечення в його остаточної конфігурації, інтегрованого з іншими програмними та апаратними системами.

Одним із способів вивчення поставленого питання є дослідження методики «чорної скриньки», Основна роль тестування методів «чорної скриньки» – це інтерфейс програмного забезпечення. Ці тести демонструють:

– як будуть виконуватись функції програми;

– як будуть прийматися вихідні дані;

– як будуть формуватись результати;

– як буде збережена цілісність зовнішньої інформації.

Тестування програмних засобів буде доречно проводити використовуючи методику «чорної скриньки» (див. таблиця 3.1 – Тест-кейси).

Вона базується на використанні шаблонів тестування або ж тест-кейсів. Це означає, що буде створено декілька ситуацій у яких перевіряється працездатність додатку, коректності основних функцій.

## 3.3 Інструкція для користувача програми

Інструкція користувачеві програми є важливим компонентом успішного використання програмного забезпечення. Вона надає користувачам необхідну інформацію, допомагає їм ознайомитися з функціями програми та ефективно виконувати завдання.

Одним із ключових аспектів інструкції користувачеві програми є чіткість та доступність. Інструкція повинна бути написана зрозумілою мовою, уникати технічних термінів або пояснювати їх в разі необхідності. Вона також повинна бути структурованою та логічно побудованою, розподіляючи інформацію на розділи або кроки, що допомагають користувачам в орієнтації.

Інструкція має включати пояснення основних функцій програми, інтерфейсу користувача та доступних опцій. Вона може описувати послідовність дій для виконання конкретних завдань, включаючи клацання кнопок, заповнення форм або вибір пунктів меню. Додаткові матеріали, такі як зображення, відео чи ілюстрації, можуть бути використані для кращого розуміння інструкцій.

Загалом, якісна інструкція користувачеві програми є важливим інструментом для допомоги користувачам в ефективному використанні

програмного забезпечення. Вона забезпечує належне орієнтування, допомагає уникнути помилок та забезпечує задоволення від користування програмою.

Мінімальні вимоги для запуску ПЗ.

Важливим фактором, який необхідно врахувати при розробці програмного забезпечення є потреба в ресурсах.

Мінімальні вимоги до апаратних засобів для запуску та функціонування розробленого програмного забезпечення:

– процесор Pentium IV/Xeon2.4Ггц;

– оперативна пам'ять: 1024 Мб вище;

– вільний дисковий простір менше 120Мб;

– миша;

– клавіатура;

– монітор.

Вимоги до програмних засобів:

– операційна система сімейства Windows: починаючи з Windows 7 – до Windows 10;

– наявність бібліотеки класів .NET framework 4.5 або вище.

Опис процедури розгортання програмного продукту, створеного на платформі .NET

Розгортання програмного продукту на комп'ютері користувача у вигляді автономного застосунку:

– створіть на цільовому диску каталог для застосунку, наприклад «Моніторинг робочого часу IT компанії»;

– скопіюйте каталог «Debug» із проекту;

– для перевірки коректності запуску програми виконайте подвійне клацання лівою кнопкою миші на файлі «SoftwareCompany.exe» (операційна система Windows).

Використання програмного продукту

Дана програма відкриває широкі можливості для організації роботи компаній, що займаються розробкою проектів.

На початку необхідно запустити додаток «Моніторинг робочого часу IT компанії».

Отже, після запуску програми буде відкрите головне вікно програми з основним меню (рис.3.14).

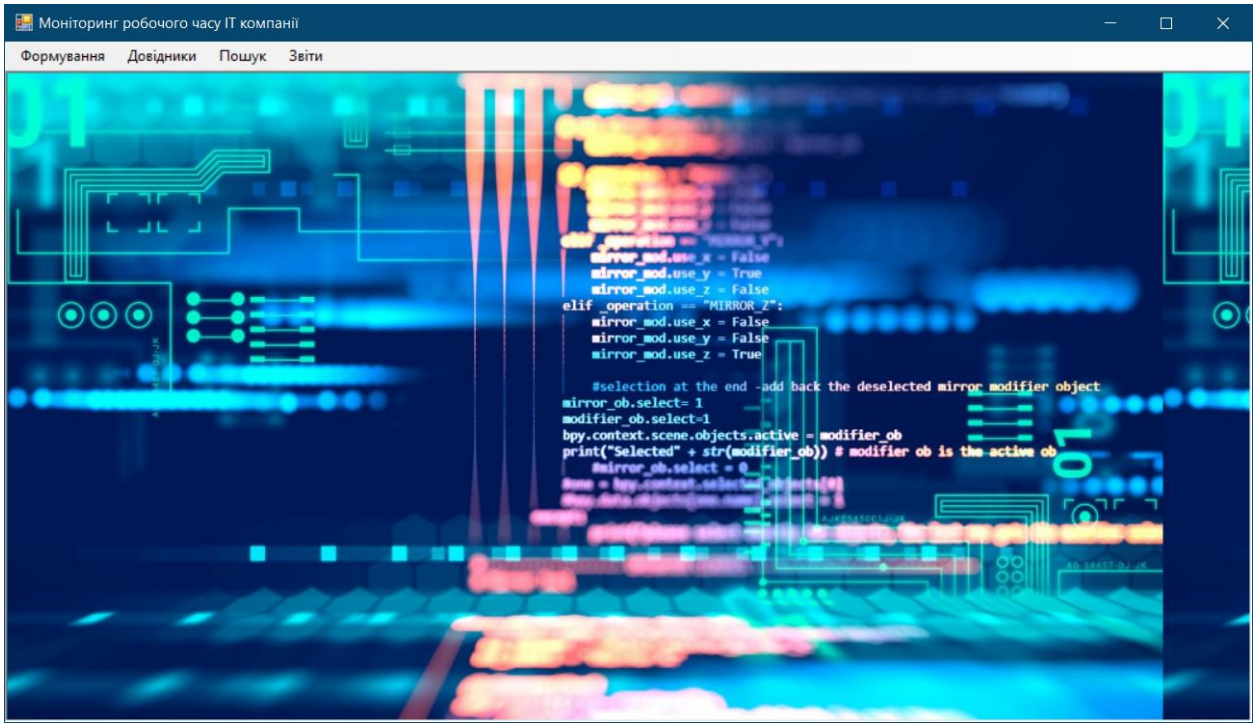


Рисунок 3.14 – Головне меню програми

Для контролю виконаення проектів необхідно наповнити інформаційну систему, а саме додати інформацію про замовників, програмістів, штат, посади та типи проектів.

Програма дозволяє також редагувати та видаляти введені дані про замовників, програмістів, штат, посади та типи проектів.

Для початку потрібно додати інформацію про клієнтів. Для цього необхідно перейти по меню програми «Довідники» –> «Замовники», у відповідному вікні (рис. 3.15) ввести всі необхідні дані про замовника та натиснути кнопку «Додати». Новий замовник з’явиться у правій частині екрану у списку всіх клієнтів.

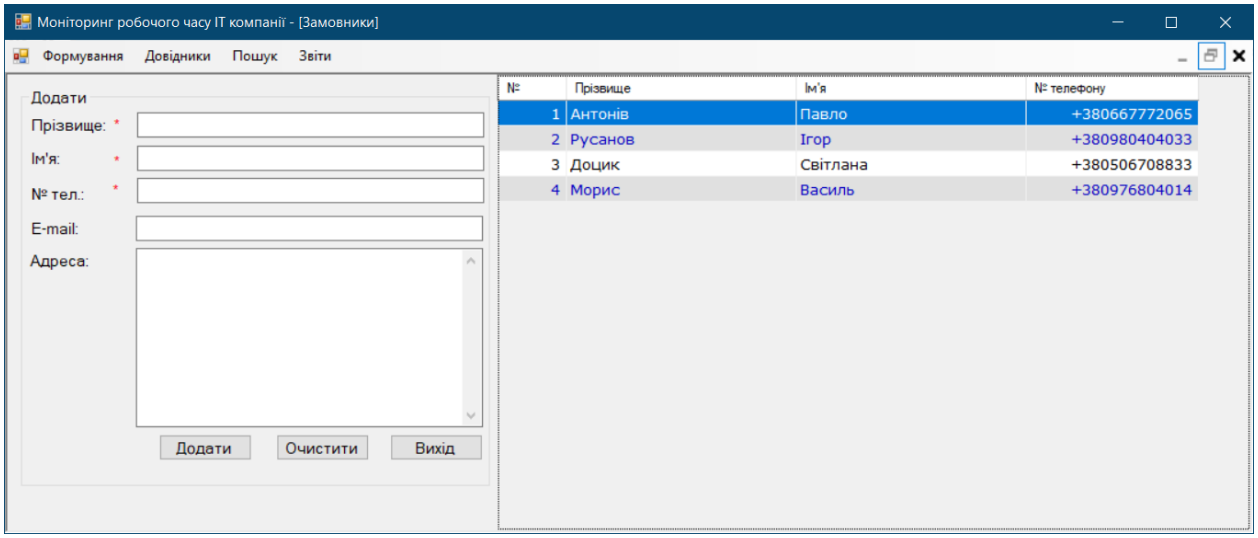


Рисунок 3.15 – Вікно для опрацювання даних про клієнтів

Також реалізована можливість редагування та видалення даних у випадку якщо це є необхідним. На рис 3.16 зображено вікно для редагування даних вибраного із списку клієнта.

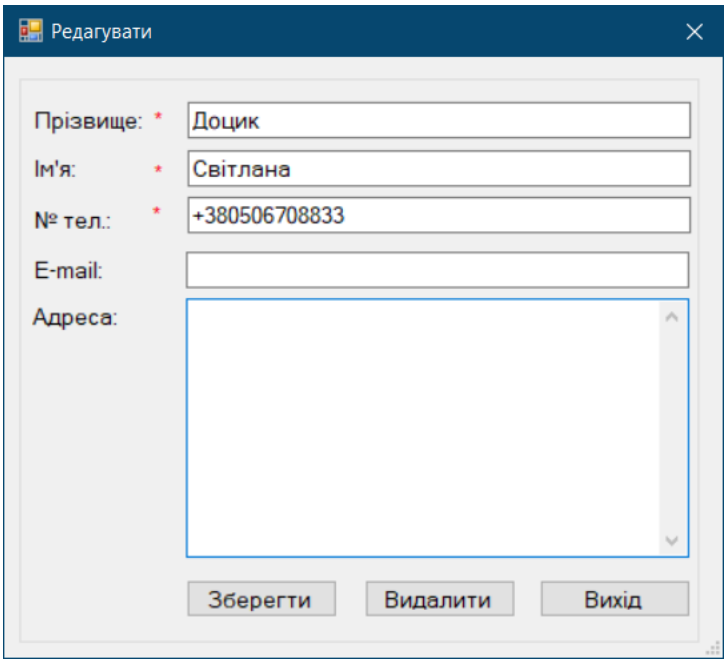


Рисунок 3.16 – Вікно для редагування даних клієнта

Далі необхідно заповнити базу програмістів, вікно для опрацювання даних про програмістів показано на рис. 3.17.

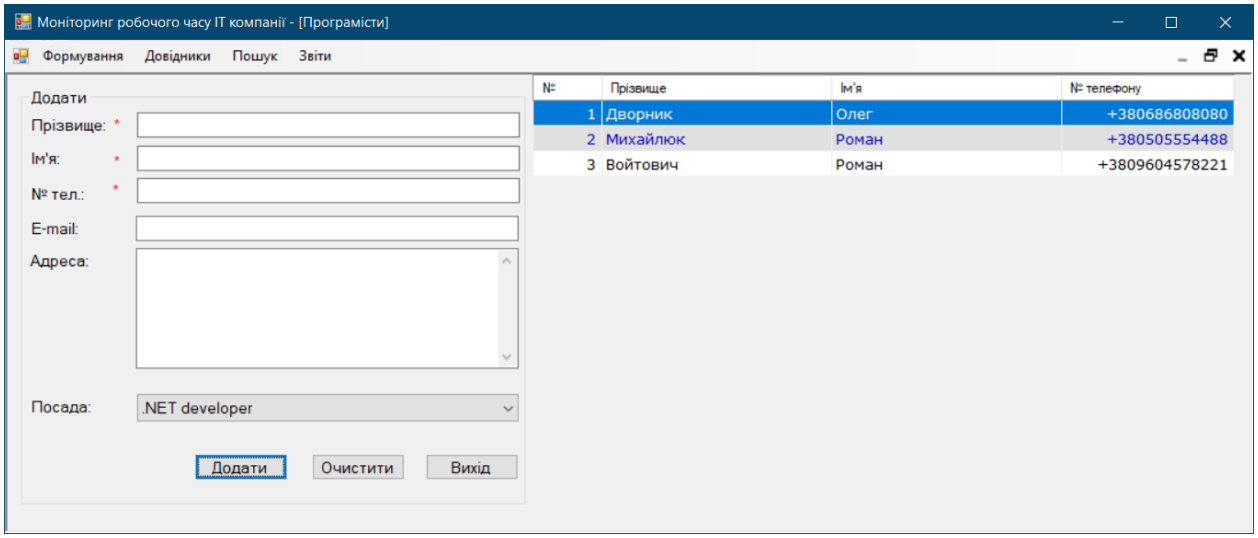


Рисунок 3.17 – Вікно для опрацювання даних про програмістів

Дані про програмістів можна відредагувати, вибравши у правій частині необхідного програміста натисканням лівої кнопки мишки. Після цього з’явиться вікно для редагування даних вибраного програміста (рис. 3.18).

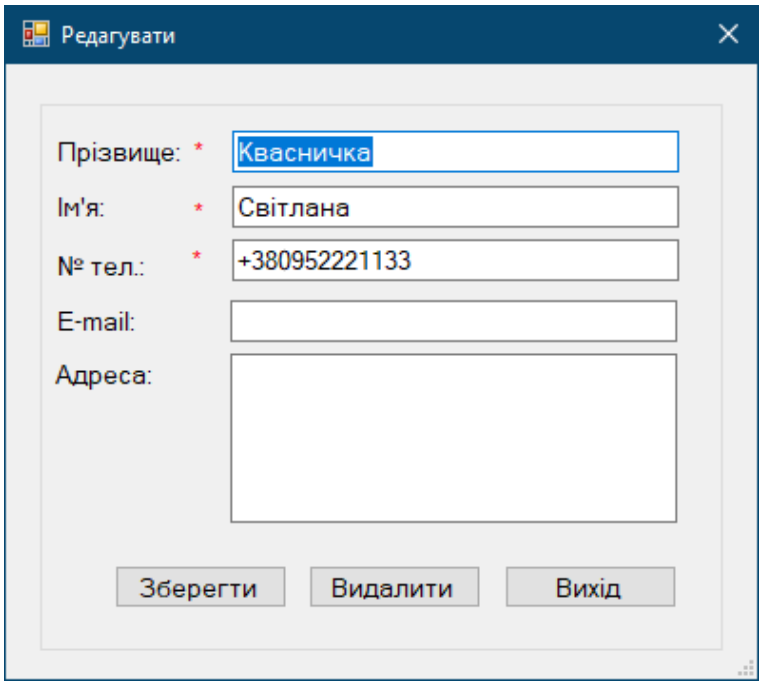


Рисунок 3.18 – Вікно для редагування даних вибраного програміста

Також у програмі є функція додавання нової посади. Для того, щоб додати нову посаду, користувачу програми необхідно перейти до пункту меню «Довідники» –> «Посади», після чого з’явиться відповідне вікно (рис. 3.19).

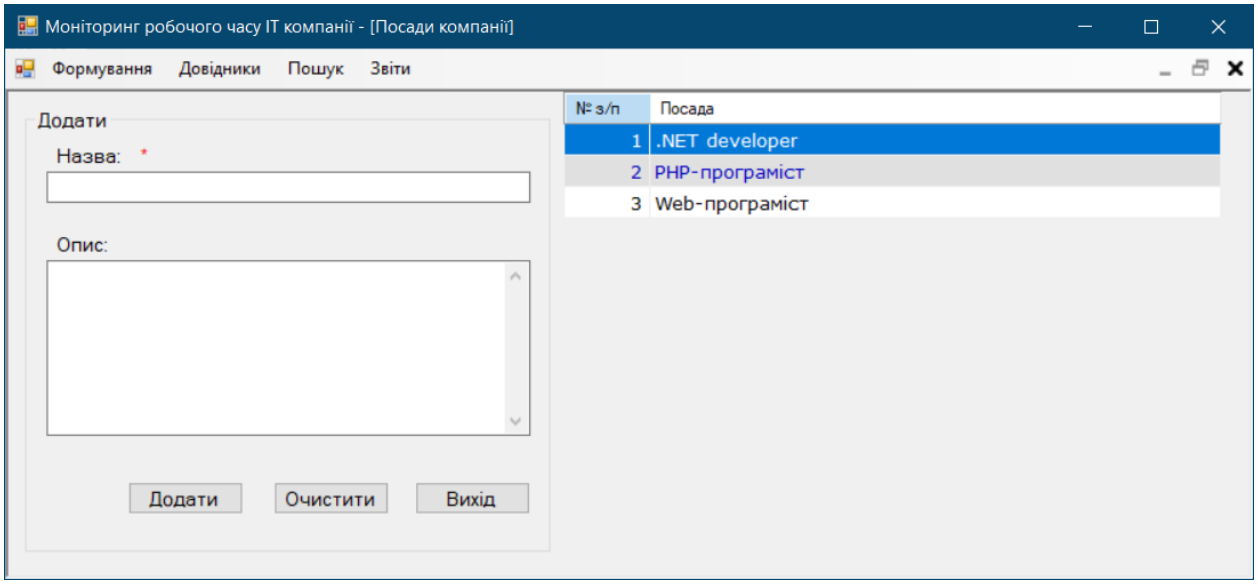


Рисунок 3.19 – Вікно для опрацювання даних про посади

Дані про посади також можна редагувати, для цього необхідно вибрати у правій частині відповідну посаду, після чого зявиться вікно, що показане на рис. 3.20.

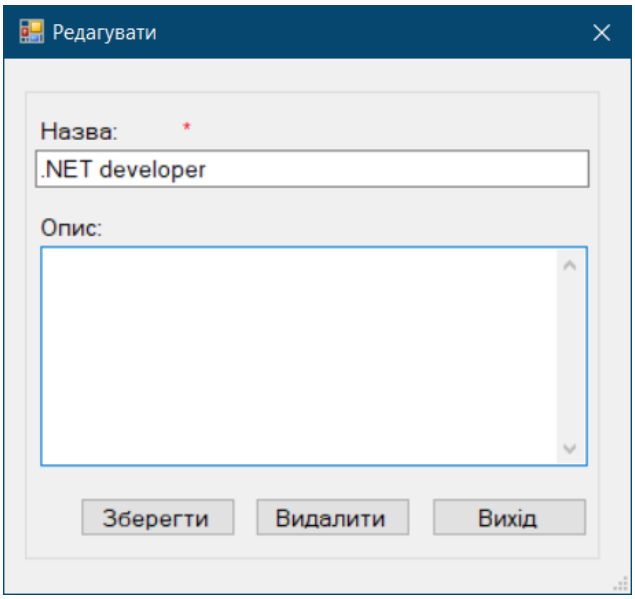


Рисунок 3.20 – Вікно для редагування даних посади

Для реєстрації проекту у IT-компанії необхідно перейти по меню програми «Формування» –> «Проекти», після чого відкриється вінкно, що представлено на рис. 3.21. Для додавання нового проекту необхідно вказати інформацію про нього: назву проекту; тип проекту; замовника; команду, що буде працювати над ним; період часу, протягом якого буде проводитись розробка проекту та грошову суму.

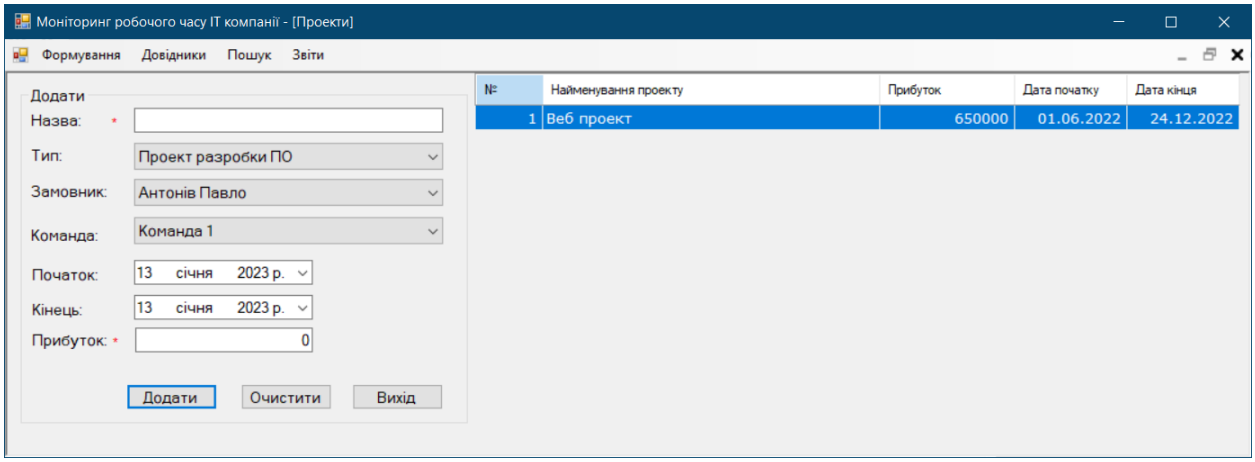


Рисунок 3.21 – Вікно для опрацювання даних про проекти

Вразі допущення помилки, дані про проекти теж можна редагувати. Також, у програмі реалізована можливість розбиття проекту на задачі, для того, щоб можна було працювати більш ефективніше. Для цього користувачу програми необхідно перейти по меню «Формування» –> «Задачі» (рис. 3.22).

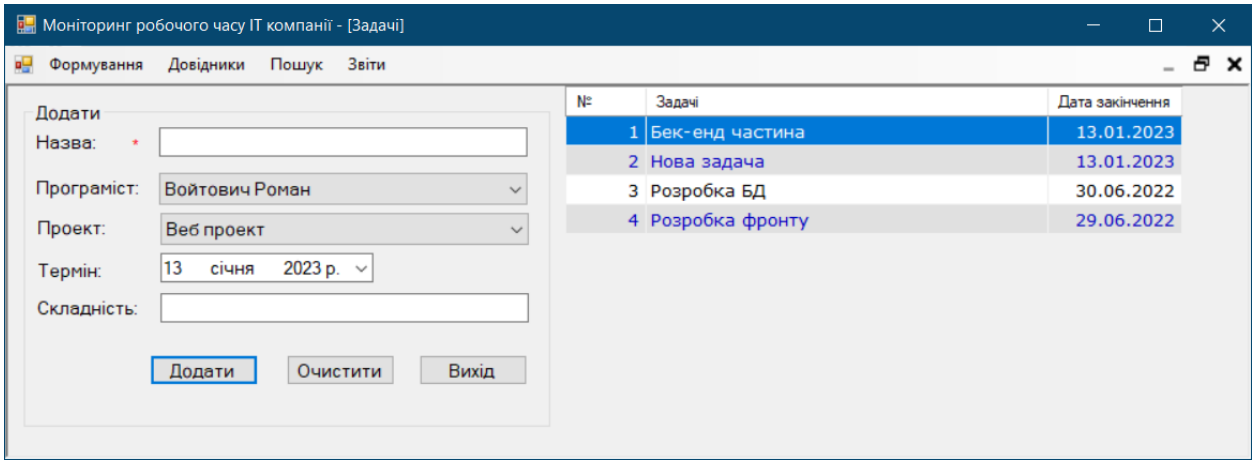


Рисунок 3.22 – Вікно для розбиття проекту на задачі

Для того, щоб розбити проект на задачі необідно вказати: назву задачі, програміста, проект, термін виконання та складність.

Для фіксації виконання задач, необхідно перейти по меню програми «Формування» –> «Фіксація виконання задачі» (рис. 3.23).

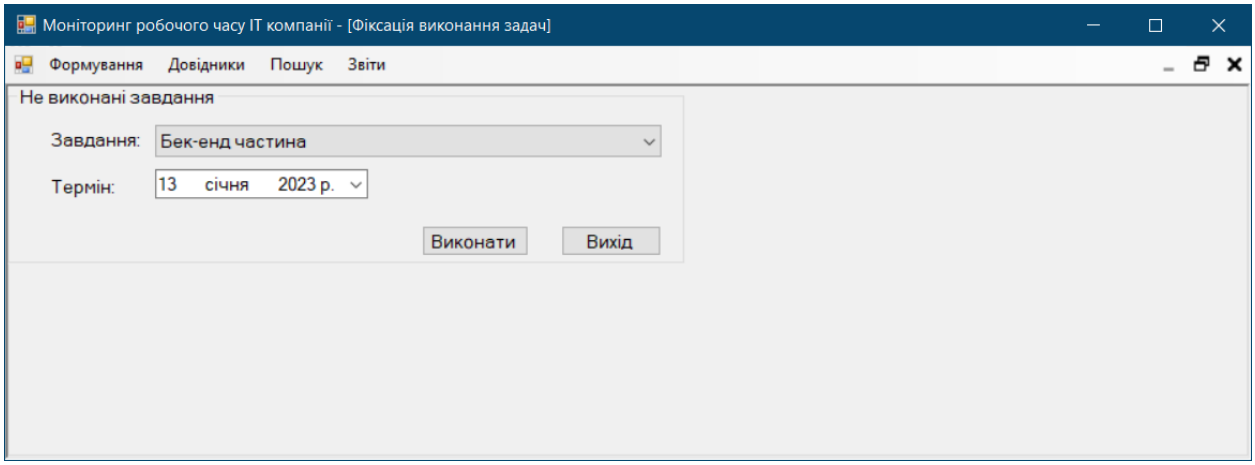


Рисунок 3.23 – Фіксація виконання задачі

Для пошу проектів та задач по вибраному програмісту необхідно перейти по меню програми «Пошук» –> «По програмісту» (рис. 3.24). В даному вікні у необхідно вибрати із списку програміста та натиснути кнопку «Знайти», після чого відповідна інформація буде знайдена.

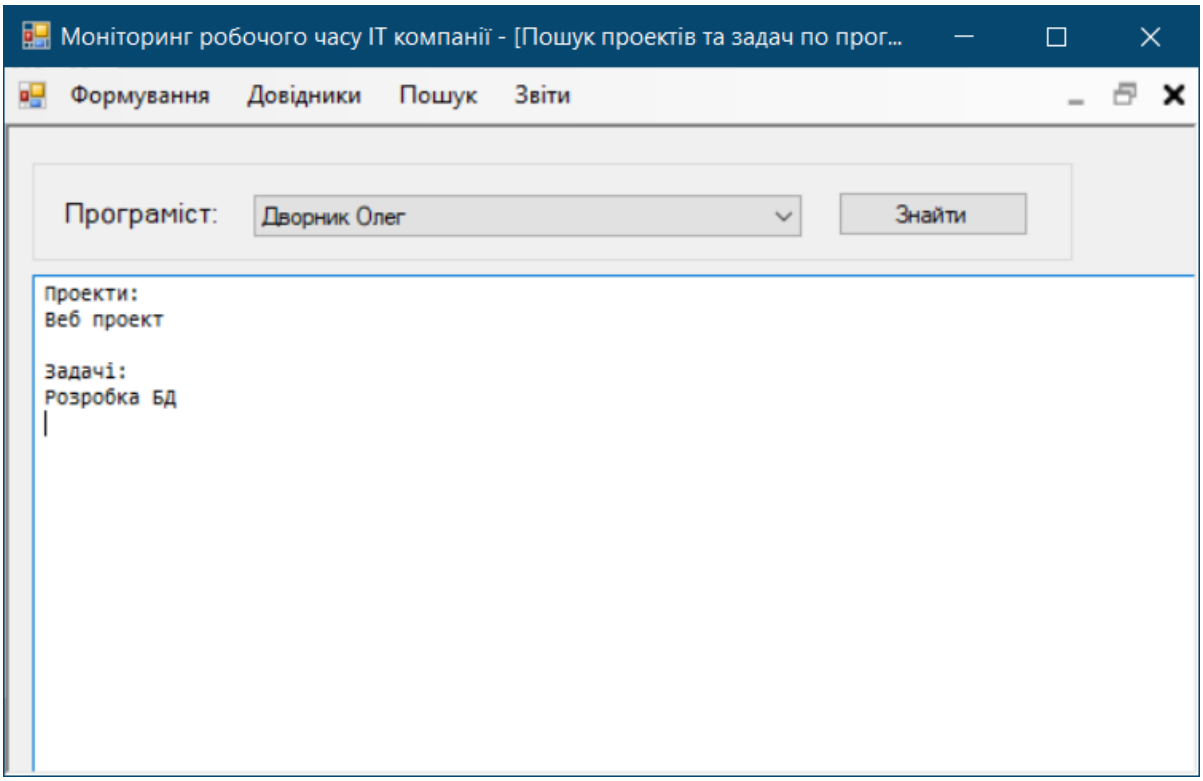


Рисунок 3.24 – Пошук проектів та задач у яких бере участь програміст

Перейшовши по пункті меню «Звіти» –> «По проектам» можна формувати відповідну звітність по вибраному із списку проекті (рис. 3.25).

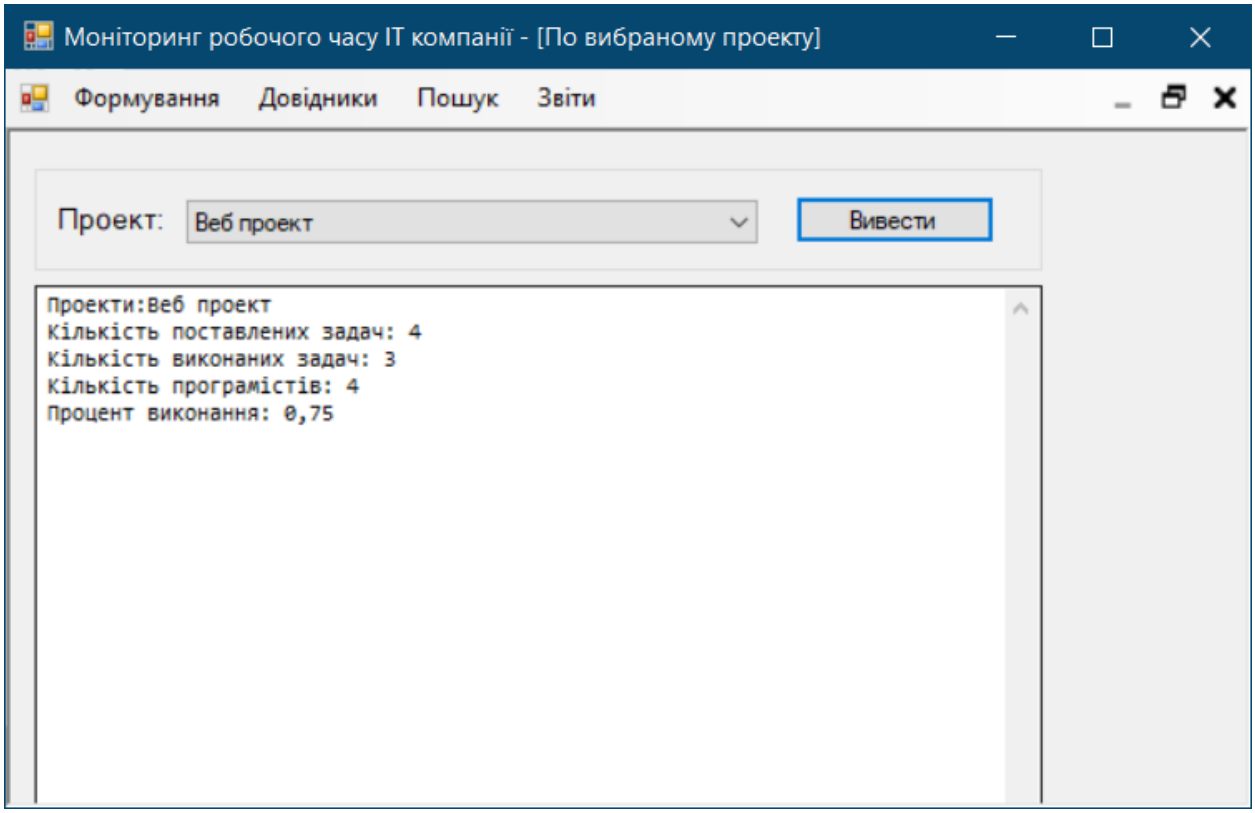


Рисунок 3.25 – Звіт по вибраному проекту

Перейшовши по пункті меню «Звіти» –> «По програмісту» можна формувати відповідну звітність по програмісту за вибраний період часу(рис. 3.26).

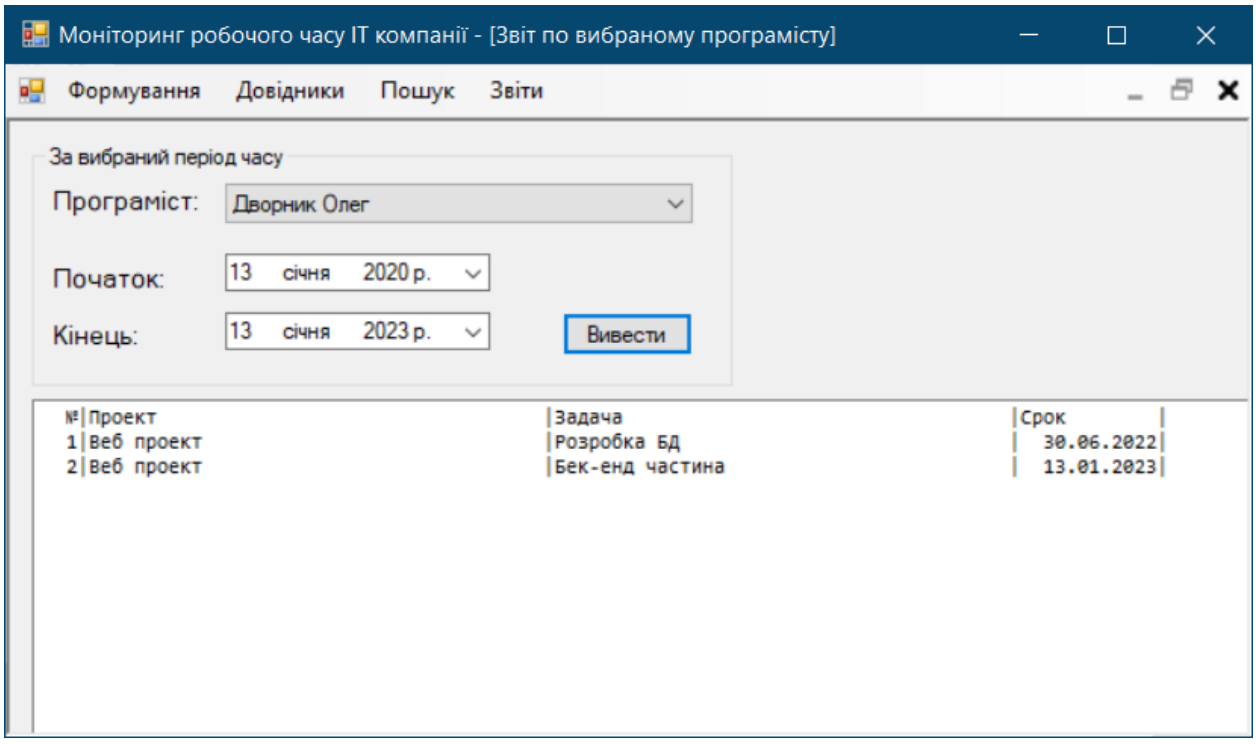


Рисунок 3.26 – Звіт по по програмісту за вибраний період часу

Треба сказати, що дана система є не сильно функціональною, але вона є досить простою в користуванні. ЇЇ інтерфейс є інтуїтивно зрозумілим для користувача.

Тестування програми успішно проведено та не було виявлено жодних помилок системи.

В ході виконання роботи мовою C# в середовищі Visual Studio 2019 реалізовано додаток для моніторингу діяльності працівників в робочий час на базі IT компанії.

Реалізовано такі функціональні вимоги:

- додавати, редагувати та видаляти дані про замовників;
- додавати, редагувати та видаляти дані про програмістів;
- додавати, редагувати та видаляти дані про номери посади працівників;
- додавати, редагувати та видаляти дані про типи проектів;
- можливість формування штату працівників;
- додавати, редагувати та видаляти дані про проекти;
- можливість розбиття проектів на задачі;
- можливість фіксації виконання задач;
- пошук необхідної інформації по вибраному із списку програмісту;
- формування звітності по програмісту за вибраний період часу;
- формування звітності по проекту.

# 4 ТЕХНІКО-ЕКОНОМІЧНИЙ РОЗДІЛ

## 4.1 Етапи технологічного процесу

В наявному розділі дипломної роботи буде проведено техніко-економічне обґрунтування доцільності створення та реалізації програмного забезпечення для моніторингу діяльності працівників на базі ІТ компанії.

У реалізації даного програмного забезпечення беруть участь такі розробники, як: керівник дипломного проекту; студент розробник програмного забезпечення; консультант техніко-економічного розділу.

Почати слід із формулювання загального життєвого циклу створення програмного забезпечення:

– етап прийняття вимог від замовника та планування який має бути кінцевий результат;

– етап задокументовування та збирання всіх вимог та всіх деталів створення програмного продукту;

– етап розробки макету дизайну та проектування графічної структури програмного забезпечення;

– етап розробки програмної частини, тобто логіки продукту;

– етап тестування чи програмний продукт відповідає усім вимогам та чи не виникає дефектів під час роботи. Усування помилок, які були знайдені в процесі тестування;

– етап на якому готове програмне забезпечення надається замовнику;

– етап технічного обслуговування проекту, а саме обновлення та виправлення залишкових помилок.

## 4.2 Визначення необхідного періоду часу та трудомісткості для створення проекту

Трудомісткість можна знайти та обчислити використовуючи формулу:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д\,, \qquad (4.1)$$

$$t = 50 + 13 + 32{,}5 + 32{,}5 + 162{,}5 + 71{,}08 = 361{,}58 \text{ (люд/год)}$$

де $t_o$ - розхід роботи на підготовку й опис поставленої задачі (приймається

50);

$t_u$ - розхід роботи на дослідження алгоритму рішення поставленної задачі;

$t_a$ - розхід роботи на створення блок-схеми алгоритму;

$t_n$ - розхід роботи на програмування по завершеній блок-схемі;

$t_{отл}$ - розхід роботи на налаштування програми на комп'ютері;

$t_д$ - розхід роботи на підготовку документації.

Потрібно визначити умовне число операторів, які розраховуються за формулою:

$$Q = q \cdot C \cdot (1 + p), \qquad (4.2)$$

$$Q = 500 \cdot 0{,}65 \cdot (1 + 1) = 650,$$

$q$ – передбачувана кількість операторів;

$C$ – коефіцієнт складності програми;

$p$ – кількість змін внесених в програму в ході її створення.

Витрати праці на дослідження опису задачі розраховується з урахуванням складності алгоритму та стажу програміста:

$$t_u = \frac{Q \cdot B}{(75..85) \cdot k}\,, \qquad (4.3)$$

$$t_u = \frac{650 \cdot 1,2}{75 \cdot 0,8} = 13$$

де В - значення непередбачуваних витрат праці через недостатньо детальний опис поставленної задачі (в залежності від рівня складності завдання і кількості внесенних змін обирається в діапазоні від 1,2 до 1,5);

k - коефіцієнт рівня кваліфікації працівника, залежить від стажу роботи з даної спеціальності.

Розхід праці на створення алгоритму вирішення поставленої задачі обчислюють за формулою:

$$t_a = \frac{Q}{(20..25) \cdot k}, \quad (4.4)$$

$$t_a = \frac{650}{25 \cdot 0,8} = 32,5$$

Розхід на конструювання програмного забезпечення по завершенній блок-схемі рахують за формулою

$$t_n = \frac{Q}{(20..25) \cdot k}, \quad (4.5)$$

$$t_n = \frac{650}{25 \cdot 0,8} = 32,5$$

Розхід праці на налагодження програми персональному комп'ютері обчислюють:

$$t_{отл} = \frac{Q}{(4..5) \cdot k}, \quad (4.6)$$

$$t_{отл} = \frac{650}{5 \cdot 0,8} = 162,5$$

Розхід праці на підготовку документації обчислюють за формулою:

$$t_д = t_{др} + t_{до}\,, \tag{4.7}$$

$$t_д = 40{,}625 + 30{,}46 = 71{,}08$$

де $t_{др}$ - трудомісткість підготовки матеріалів і рукопису, що розраховують за формулою:

$$t_{др} = \frac{Q}{(15..20) \cdot k}\,, \tag{4.8}$$

$$t_{др} = \frac{650}{20 \cdot 0{,}8} = 40{,}625$$

$t_{до}$ - трудомісткість редагування, друкування й оформлення документації:

$$t_{до} = 0{,}75 \cdot t_{др}\,, \tag{4.9}$$

$$t_{до} = 0{,}75 \cdot 40{,}625 = 30{,}46$$

4.3 Розрахунок витрат на створення програмного забезпечення

В даному випадку під оплатою праці розробника в ролі якого виступає студент беремо розмір стипендії, який складає 2880,9 гривень на місяць. Таким чином річний фонд грошового забезпечення складає 34570,8 гривень.

Кількість робочих годин у році можна обчислити за формулою:

$$N_p = (N - N_n - N_в) \cdot 8\,, \tag{4.10}$$

$$N_p = (365 - 11 - 107) \cdot 8 = 1976$$

звідки, N – сумарне число днів у році (365),

$N_п$ – кількість святкових днів у році (11),

$N_в$ – кількість вихідних днів у році (107).

Середньогодинну оплату праці програміста можна обчислити за формулою:

$$C_n = \frac{Ф_p}{N_p}, \quad (4.11)$$

$$C_n = \frac{34570,8}{1976} = 17,5$$

де $Ф_p$ – розмір річної оплати праці

Розхід на оплату праці програміста обчислюється за формулою:

$$B_{оп} = C_n \cdot T_з, \quad (4.12)$$

$$B_{оп} = 17,5 \cdot 361,58 = 6327,65$$

де $T_з$- сумарний час в годинах проведених розробником на проектом

Розхід який пов’язаний із створенням програмного забезпечення обчислюється за формулою:

$$B_{ПК} = T_{ПК} \cdot t_c, \quad (4.13)$$

$$B_{ПК} = 361,58 \cdot 0,09 = 32,54$$

звідки $T_{ПК}$ - термін застосування комп’ютера для створення програмного забезпечення;

Витрати на годину праці за комп'ютером можна обчислити за формулою:

$$C_{ПК} = \frac{B_e}{Ф_{ПК}}, \quad (4.14)$$

$$C_{ПК} = \frac{1450}{1976} = 0,73$$

звідки $B_e$ – розходи на використання персонального комп'ютера терміном на один рік роботи;

$Ф_{ПК}$ – термін часу роботи персонального комп'ютера за рік.

Далі обчислимо кількість часу роботи персонального комп'ютера за рік. Знайшовши фактичний термін роботи комп'ютера у годинах, зможем отримати та оцінити витрати за терміни в годинах роботи комп'ютера.

Фактичні річні витрати часу роботи комп'ютера можна обчислити за формулою:

$$Ф_д = N_p - (Ф_м + Ф_{річ}), \quad (4.15)$$

$$Ф_д = 1976 - (5 + 144) = 1797$$

звідки, $Ф_м$ – термін профілактичних робіт та обслуговування комп'ютера за місяць його експлуатації (в середньому цей період складає 5 годин на місяць);

$Ф_{річ}$ – термін профілактичних робіт та обслуговування комп'ютера за рік його експлуатації (в середньому цей період складає 6 діб на рік (144 години)).

Фактичні розходи в рік на експлуатацію програмного забезпечення обчислюються за формулою:

$$B_{ПК_р} = B_{E_р} + B_{A_р} + B_{РЕМ_р} + B_{ДК_р} + B_{I_р}, \quad (4.16)$$

$$B_{ПК_р} = 139,5 + 3360 + 1344 + 448 + 1120 = 6411,5$$

звідки, $B_{A_р}$ – розхід амортизації за рік;

$В_{Е_р}$ – рохід на використану електро енергію для комп'ютера за рік;

$В_{РЕМ_р}$ – розхід на ремонтне обслуговування комп'ютера за рік;

$В_{ДК_р}$ – розхід на необхідні під час роботи деталі для комп'ютера;

$В_{І_р}$ – інші необхідні розходи;

Розхід на амортизаційні витрати можна обчислити за формулою:

$$В_{А_р} = Ц_{ПК} \cdot Н_А , \tag{4.17}$$

$$В_{А_р} = 22400 \cdot 0,15 = 3360$$

звідки, $Ц_{ПК}$ – балансована ціна робочого комп'ютера;

$Н_А$ – показник амортизаційних розходів (приймається за 15%);

Балансовану ціну персонального комп'ютера обчислюють за формулою:

$$Ц_{ПК} = Ц_р \cdot (1 + К_{УН}) , \tag{4.18}$$

$$Ц_{ПК} = 20000 \cdot (1 + 0,12) = 22400$$

звідки, $Ц_р$- ринкова ціна комп'ютера;

$К_{УН}$ – коефіцієнт, який включає розхід на встановлення й налаштування комп'ютера (вважається за 12%).

Розходи на електро енергію що використовує комп'ютер можна обчислити за формулою:

$$В_{Е_р} = Р_{ПК} \cdot Ф_{ПК} \cdot Ц_Е \cdot К_{ІВ} , \tag{4.19}$$

$$В_{Е_р} = 0,06 \cdot 1976 \cdot 1,68 \cdot 0,7 = 139,5$$

звідки, $Р_{ПК}$ – заявлена виробником паспортна потужність комп'ютера;

$Ф_{ПК}$ – термін часу роботи персонального комп’ютера за рік;

$Ц_{Е}$ – поточна ціна за 1 кВт/год електро енергії;

$К_{ІВ}$ – коефіцієнт що визначає інтенсивність використання комп’ютера (приймається 0,7 - 1).

Отже далі потрібно визначити приблизне значення розходу на електро енергію, що витрачає комп’ютер, складає:

Розходи на поточне обслуговування та профілактичний ремонт (вважається 6% від ціни комп’ютера):

$$В_{РЕМ_р} = Ц_{ПК} \cdot 0{,}06\,, \tag{4.20}$$

$$В_{РЕМ_р} = 22400 \cdot 0{,}06 = 1344$$

Розходи на необхідні під час роботи деталі для комп’ютера (вважаються 2% від ціни комп’ютера)

$$В_{ДК_р} = Ц_{ПК} \cdot 0{,}02\,, \tag{4.21}$$

$$В_{ДК_р} = 22400 \cdot 0{,}02 = 448$$

Інші необхідні розходи, мається на увазі непрямі розходи неодхідні для повноцінної експлуатації комп’ютера (вважається 5-10% від ціни комп’ютера):

$$В_{І_р} = Ц_{ПК} \cdot 0{,}05\,, \tag{4.22}$$

,

$$В_{І_р} = 22400 \cdot 0{,}05 = 1120$$

В процесі створення програмного забезпечення комп’ютер застосовується на наступних стадіях програмування:

– розробка коду програми за створеним алгоритмом;

– налаштування програмного забезпечення на комп'ютері;

– написання документації по закінченому програмному продукту.

Отже, спочатку слід розрахувати затрати часу із використанням компютерної техніки, що обчислюється за формулою:

$$t_{маш} = t_n + t_{отл}^k + t_д, \quad (4.23)$$

$$t_{маш} = 30{,}46 + 130 + 71{,}08 = 231{,}54$$

Розходи на оплату часу роботи комп'ютерної техніки, можна обчислити за формулою:

$$В_{маш} = t_{маш} \cdot С_{ПК}, \quad (4.24)$$

$$В_{маш} = 231{,}54 \cdot 0{,}73 = 169{,}02$$

Сумарний розхід на створення програмного забезпечення, можна розрахувати за формулою:

$$В_{заг} = В_{оп} + В_{маш}, \quad (4.25)$$

$$В_{заг} = 6327{,}65 + 169{,}02 = 6496{,}67$$

## 4.4 Обчислення показників економічної ефективності створення програмного забезпечення

В даній частині техніко економічного розділу буде проведено розрахунки оцінки наскільки доцільно впроваджувати дане програмне забезпечення використовуючи міжнародні стандарти ефективності створення програмного продукту. А саме: показника внутрішньої дохідності, показника інвестиційних вкладень, дохід від розробки, вартість продажу, рентабельність, економічна ефективність та термін за який даний програмний продукт себе окупить.

Спочатку наведу формулу по якій можна обчислити об’єм чистого прибутку і получим показник внутрішньої прибутковості програмного забезпечення.

$$\sum_{i=0}^{T}\frac{Di}{(1+q)^{i}}-\sum_{i=0}^{T}\frac{Ki}{(1+q)^{i}}=0, \qquad (4.26)$$

звідки $D_i$ – дохід за $i$ період;

$K_i$ – маються на увазі інвестиції за $i$ період включаючи інфляцію;

$i$ – термін за який виготовляють проект та реалізують його;

$T$ – весь термін етапів створення проекту;

$q$ – коефіцієнт внутрішнього доходу від проекту.

Коефіцієнт інвестиційних внесків із включенням можливості інфляції можна розрахувати за формулою:

$$K_i = \varphi_i \cdot R_i, \qquad (4.27)$$

$$K_i = 1{,}266 \cdot 6496{,}67 = 8224{,}78$$

звідки $\varphi_i$ – мається на увазі рівень інфляції на даний момен (126,6%);

$R_i$ – капітальні вкладення;

Прибуток від створення прграмного забезпечення за $i$ період можна обчислити за формулою

$$D_i = J_i \cdot (B_i - C_i), \qquad (4.28)$$

$$D_i = 1 \cdot (7406{,}2 - 6496{,}67) = 909{,}53$$

звідки $B_i$ – фактична вартість продажу програмного забезпечення за $i$ період;

$C_i$ – розходи затрачені на розробку програмного забезпечення тобто собівартість;

$J_i$- число проданих зразків програмного забезпечення.

Ціна реалізації на ринку програмного продукту можна обчислити за формулою:

$$B_i = B_{\text{заг}} \cdot (1 + \frac{\text{p}}{100}), \quad (4.29)$$

$$B_i = 6496{,}67 \cdot \left(1 + \frac{14}{100}\right) = 7406{,}2$$

звідки, p – приблизний рівень рентабльності на дани період

Економічну ефективність можна вирахувати за допомогою відношення кінцевого результату створеного програмного забезпечення до витрачених ресурсі:

$$\text{E} = \frac{D_i}{\text{B}_{\text{заг}}}, \quad (4.30)$$

$$\text{E} = \frac{909{,}53}{6496{,}67} = 0{,}14$$

Отже тепер можна обчислити термін окупності створеного програмного забезпеченя за формулою:

$$\text{T} = \frac{1}{\text{E}}, \quad (4.31)$$

$$\text{T} = \frac{1}{0{,}14} = 7{,}14$$

Висновок. В даному розділі було проведено розрахунки часу який буде затрачений на розробку програмного забезпечення, розрахунок витрат на створення проекту та розрахунок показників економічної ефективності. На основі цих данних можна зробити висновок чи являється розробка рентабельною та за скільки часу себе окупить та начне приносити прибуток.

Розробником даного програмного забезпечення виступає студет. З огляду на розрахунки йому знадобиться 361,58 годин на створення проекту. За оплату його праці берем розмір стипендії. Та було проведено обчислення остальних витрати на розробку програмного забезпечення, які сумарно склали 6496,67 гивень. Отримавши ці результати було проведено розрахунок показників економічної ефективності. Судячи із яких можна сказати, що вартість з продажу будескладати 7406,2 гривні за одну копію. Чистий дохід від якої буде складати 909,53 гривні. Тоді економічно ефективність, яка являється відношенням загального прибутку до загальних витрат, рівна 0,14. За допомого якої можна сказати за який термін часу дана розробка себе окупить і цей термін склав 7,14 років. Даний термін часу являється меншим десяти років, що говорить про те що даний програмний продукт являється економічно вигідним, рентабельним та конкурентно здатним на ринку подібний розробок

# ВИСНОВКИ

Даний проект розроблявся для автоматизації бізнес процесів IT-компанії. Розробку інформаційної системи було виконано у середовищі Microsoft Visual Studio 2019 при використанні мови програмування C # та СУБД MS Access. Дана інформаційна система повинна значно полегшити роботу менеджерів та бухгалтерів IT-компанії, а саме тим, тим, що розроблене ПЗ має зручний перегляд даних, додавання та вилучення записів та виконання пошуку і фільтрації потрібних записів за допомогою запитів.

У розробленому продукту реалізовано такі функціональні можливості:

- додавати, редагувати та видаляти дані про замовників;
- додавати, редагувати та видаляти дані про програмістів;
- додавати, редагувати та видаляти дані про номери посади працівників;
- додавати, редагувати та видаляти дані про типи проектів;
- можливість формування штату працівників;
- додавати, редагувати та видаляти дані про проекти;
- можливість розбиття проектів на задачі;
- можливість фіксації виконання задач;
- пошук необхідної інформації по вибраному із списку програмісту;
- формування звітності по програмісту за вибраний період часу;
- формування звітності по проекту.

У першому розділі було проведено дослідження діяльності та структури IT компанії, також проаналізовано основні бізнес процеси IT компанії. Після чого здійснено постановку задачі для розробки системи моніторингу діяльності працівників в робочий час на базі IT компанії.

У другій частині роботи розглянуто дослідження системи та описано її архітектуру. Зроблено аналіз вимог до програмного забезпечення та побудовано use-case діаграми основних прецедентів. Також, розглянуто середовище розробки ПЗ та виділено його основні переваги та недоліки.

У третій частині роботи описується процес розробки програмного забезпечення та проведено його ретельне тестування. Також, було розроблено інструкцію користувача програми. Результати тестування були успішними, помилок не було виявлено.

Четвертий розділ присвячений здійсненню розрахунку економічної ефективності проекту. В процесі якого результати показали, що даний програмний продукт являється економічно вигідним, рентабельним та конкурентно здатним на ринку подібний розробок. Згідно проведеного економічного обґрунтування дане проектне рішення є вигідним для його реалізації на ринку.

В результаті проведеної роботи вирішено актуальне технічне завдання для моніторингу діяльності працівників в робочий час на базі ІТ компанії. У процесі вирішення завдання розроблено інженерну методику автоматизованої процедури управління проектами у ІТ компанії, розбивання його на детальні задачі та розподіл завдань між працівниками і таким чином поставленої мети досягнуто. У ході виконання роботи отримано такі основні наукові та практичні результати:

1. Розроблено гнучку систему, призначену для моніторингу діяльності працівників в робочий час на базі ІТ компанії.

2. Розроблені методи в даній роботі можуть бути використані для широкого класу завдань, тому можливий подальший розвиток розробленого програмного забезпечення.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Патра В.М. Система моніторингу діяльності працівників в робочий час на базі ІТ компанії. VII Науково-практична конференція молодих вчених і студентів «Інтелектуальні комп'ютерні системи та мережі». 23 травня 2023 р. Тернопіль: ЗУНУ, 2023. С. 24

2. Комп'ютерні мережі Частина 1 Навчальний посібник [Електронний ресурс]: навч. посіб. для студ. спеціальності 121 «Інженерія програмного забезпечення» та 126 «Інформаційні системи та технології», спеціалізації «Інженерія програмного забезпечення інформаційно управляючих систем» та «Інформаційне забезпечення робототехнічних систем»/ Б. Ю. Жураковський, І.О. Зенів; КПІ ім. Ігоря Сікорського. Київ : КПІ ім. Ігоря Сікорського, 2020. 336 с. URL: https://ela.kpi.ua/handle/123456789/36615

3. Тарнавський Ю. А., Кузьменко І. М. Організація комп'ютерних мереж: підручник: для студ. спеціальності 121 «Інженерія програмного забезпечення» та 122 «Комп'ютерні науки». КПІ ім. Ігоря Сікорського. Київ: КПІ ім. Ігоря Сікорського, 2018. 259 с. URL: https://ela.kpi.ua/handle/123456789/25156

4. Сайко В.Г., Казіміренко В.Я., Літвінов Ю.М. Мережі бездротового широкосмугового доступу. Навчальний посібник. К.: ДУТ, 2015. 196 с. URL: https://dut.edu.ua/uploads/l_881_80314158.pdf

5. Голь В.Д., Ірха М.С. Телекомунікаційні та інформаційні мережі: навчальний посібник. Київ : ІСЗЗІ КПІ ім. Ігоря Сікорського, 2021. 250 с. URL: https://ela.kpi.ua/bitstream/123456789/45409/1/TIM_navch_posib.pdf

6. The Cisco Learning Network URL: https://learningnetwork.cisco.com

7. Захист інформації в комп'ютерних системах : підручник для студ. спец. 123 «комп'ютерна інженерія» / уклад. О. М. Гапак, С. І. Балога; рец. : М. І. Глебена. – Ужгород: ПП "АУТДОР-ШАРК, 2021. – 184 с. URL: https://dspace.uzhnu.edu.ua/jspui/handle/lib/36506

8. Eksim Ali. Wireless Communications and Networks - Recent Advances. InTech (March, 2012). 596 p. URL: https://www.intechopen.com/books/1637

9. Bonaventure Olivier. Computer Networking : Principles, Protocols and Practice. Saylor, 2022. 278 p.

10. Camisso Jamon. Making Servers Work: A Practical Guide to Linux System Administration. DigitalOcean, 2020. 281 p. URL: https://www.digitalocean.com/community/books/sysadmin-ebook-making-servers-work

11. ДСТУ 3008:2015 Національний стандарт України. Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання. Введ. 01.07.2017. К.: ДП "УкрНДНЦ, 2016. 25 с

12. ДСТУ 8302:2015 Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання. Введ. 01.07.2016. К.: ДП «УкрНДНЦ», 2017. 16 с.

13. Методичні вказівки до випускних кваліфікаційних робіт освітнього рівня “Бакалавр” спеціальності “Комп’ютерна інженерія”/ О.М. Березький, Г.М. Мельник, Л.О.Дубчак, Ю.М. Батько / Під ред. О.М. Березького. Тернопіль: ЗУНУ, 2021. 52 с.

14. Методичні вказівки до виконання практичних робіт з дисципліни «Техніко-економічне обґрунтування розробки комп’ютерних систем”/ Н.Я. Савка, І.Р. Паздрій / Під ред. О.М. Березького. Тернопіль: ТНЕУ, 2019. 40 с.

15. Методичні вказівки до оформлення курсових проектів, звітів про проходження практики, випускних кваліфікаційних робіт для студентів спеціальності «Комп’ютерна інженерія» / І.В. Гураль, Л.О. Дубчак / Під ред. О.М. Березького. Тернопіль: ТНЕУ, 2019. 33 с.

16. Conway K. Software project management: from concept to deployment. – Coriolis Group, 2000. 856 p.

17. Сучасні методи управління проектами [Електронний ресурс] – Режим доступу: URL: https://sgv.in.ua/off-lifaq/25-suchasni-metodi-upravlinnya-proektami

18. Perspectives on Labour and Income: Information technology workers – http://www.statcan.gc.ca/pub/75-001-x/00703/0006579- eng.html

19. Якимчук В. С. Засоби планування та реалізації IT-проектів : дис. канд. техн. наук : 122 «Комп'ютерні науки / Якимчук В. С. – Київ КПІ ім. Ігоря Сікорського, 2018. – 52 с

20. Методи та способи управління проектами у сфері IT URL: https://it-consulting.com.ua/metodi-ta-sposobi-upravlinnya-proektami/

21. Guilford J. P. The structure of intellect / J. P. Guilford // Psychol. Bull. – 1956. – Vol. 5. – P. 267–293.

22. Користувацький інтерфейс операційної системи Windows. URL: https://pidruchniki.com/1402040448804/dokumentoznavstvo/koristuvatskiy_interfeys_operatsiynoyi_sistemi_windows

23. Matt Watson. What is C# used for? [Електронний ресурс] / matt watson // stackify product & company updateS. – 2020. – Режим доступу до ресурсу: https://stackify.com/what-is-c-used-for/ .

24. «.NET Framework Guide» [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/dotnet/framework/

25. MS ACCESS [Електронний ресурс] – Режим доступу до ресурсу: https://studfile.net/preview/5397013/

26. VBA [Електронний ресурс] – Режим доступу до ресурсу: https://www.investopedia.com/terms/v/visual-basic-for-applications-vba.asp

27. Microsoft Visual Studio [Електронний ресурс] – Режим доступу до ресурсу: https://www.visualstudio.com/.

28. Visual Studio Code [Електронний ресурс]. — Режим доступу: https://code.visualstudio.com/

29. IDE [Електронний ресурс] – Режим доступу до ресурсу: https://learn.microsoft.com/uk-ua/visualstudio/extensibility/internals/ide-defined-commands-for-extending-project-systems?view=vs-2019

30. Ben Watson. Writing High-Performance .Net Code / Ben Watson, Leticia Watson - NY : Manning Publications, 2009. – 683 с.

31. Деякі аспекти застосування UML при розробленні складних програмних систем [Електронний ресурс]. — Акименко А.М. — Режим доступу: http://vistnic.stu.cn.ua/index.pl?task=arcl&l=ru&j=8&id=21

32. Методологія IDEF0 [Електронний ресурс] – режим доступу: https://stud.com.ua/87184/ekonomika/metodologiya_idef0

33. Data-Access Layer? [Електронний ресурс]. – 2022. – Режим доступу до ресурсу https://www.geeksforgeeks.org/data-access-layer/

34. Business logic layer [Електронний ресурс] – режим доступу: https://www.geeksforgeeks.org/business-logic-layer/

35. PL/SQL [Електронний ресурс] – режим доступу: https://stackoverflow.com/questions/25076706/pl-sql-wrapper-for-c-sharp-net-example

36. Perkins B. What is ERP? Key features of top enterprise resource planning systems [Електронний ресурс] / Bart Perkins. – 2020. – Режим доступу до ресурсу: https://www.cio.com/article/2439502/what-is-erp-key-features-of-top-enterprise-resourceplanning-systems.html

37. ER diagrams vs. EER diagrams: What's the difference? URL: https://cacoo.com/blog/er-diagrams-vs-eer-diagrams-whats-thedifference .

38. DSD [Електронний ресурс] – режим доступу: https://www.dsd.technology/

39. DFD [Електронний ресурс] – режим доступу: https://www.lucidchart.com/pages/data-flow-diagram

40. BLL [Електронний ресурс] – режим доступу: https://www.integrify.com/what-is-business-logic/ .

41. Windows Forms [Електронний ресурс] – режим доступу: https://www.guru99.com/c-sharp-windows-forms-application.html

42. Visual Studio [Електронний ресурс] – режим доступу: https://vscode.dev/

43. Користувацький інтерфейс операційної системи Windows. URL: https://pidruchniki.com/1402040448804/dokumentoznavstvo/koristuvatskiy_interfeys_operatsiynoyi_sistemi_windows.

44. Етапи розробки користувацького інтерфейсу. URL: https://studopedia.su/12_23303_etapi-rozrobki-koristuvatskogo-interfeysuIteratsiyna-priroda-rozrobki.html.

45. Creating a Data Access Layer [Електронний ресурс]. – 2020. – Режим доступу до ресурсу https://learn.microsoft.com/en-us/aspnet/web-forms/overview/data-access/introduction/creating-a-data-access-layer-cs

46. Datta, A., & Dasgupta, D. (2018). An analysis of employee monitoring using ethical and legal perspectives. Journal of business research.

47. Jensen, C., & Potts, C. (2004). Privacy policies and data collection in employee monitoring. Communications of the ACM.

48. Lim, V. K., Teo, T. S., & Loo, G. L. (2001). Audit trails, accountability, and ethical issues in employee monitoring. Journal of Business Ethics, 30(2).

49. Scott, W. R. (2017). Data privacy in the workplace: An analysis of employee monitoring and surveillance practices. Journal of Business Ethics.

50. Van der Lippe, T., & Lippényi, Z. (2018). Monitoring and autonomy at work: A comparative analysis of employee experiences in Europe. European Sociological Review.