МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
УМАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ САДІВНИЦТВА
Факультет економіки і підприємництва
Кафедра інформаційних технологій

Допущено до захисту
"_____" _____ 2024 р.
Зав. кафедри, к.т.н., доцент
_____ **Ліщук Р.І.**

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ УПРАВЛІННЯ ПЕРСОНАЛОМ

Виконав – студент 2 курсу, групи 21м-з-кн
освітньої програми "Комп'ютерні науки"
спеціальності 122 Комп'ютерні науки
Пінчук Едуард Євгенович

Науковий керівник – к.т.н., доцент
Ліщук Роман Ігорович

Умань – 2024

РЕФЕРАТ

на кваліфікаційну роботу «Розробка інформаційної системи управління персоналом»
студента факультету економіки та підприємництва
Уманського національного університету садівництва
Пінчука Едуарда Євгеновича

Структура та обсяг роботи. Пояснювальна записка кваліфікаційної роботи складається з трьох розділів, містить 94 сторінок, 35 рисунків, 15 таблиць, 2 додатки, 50 джерел.

Кваліфікаційна робота присвячена розробці системи управління персоналом, яка автоматизує ключові HR-процеси та дозволяє ефективно вести облік працівників, організовувати кадрові процеси та формувати звітність. Основними цілями розробки системи є спрощення роботи HR-фахівців, автоматизація збору та обробки інформації про працівників, забезпечення безпеки даних, а також підвищення продуктивності управлінських процесів.

Задачі розробки включали: створення модулів для обліку персональних даних працівників, ведення відпусток, формування звітів по заробітній платі, інтеграцію бази даних з додатком, а також розробку інтерфейсу користувача з можливістю пошуку інформації про працівників. Окрім цього, особлива увага була приділена безпеці зберігання та обробки даних, зокрема, реалізації методів шифрування конфіденційної інформації.

У першому розділі проведено аналіз проблем в управлінні персоналом, розглянуто існуючі системи SAP SuccessFactors, Workday, Zoho People, виявлено їхні недоліки та обґрунтовано необхідність створення нової системи. У другому розділі описано процес вибору технологій для розробки системи, проведено моделювання основних бізнес-процесів і створено архітектуру системи на основі трьохрівневої моделі. Третій розділ присвячений

безпосередній реалізації системи, тестуванню її функціональних можливостей та порівнянню з іншими HR-системами.

У результаті було створено функціональну систему управління персоналом, яка дозволяє автоматизувати процеси кадрового обліку, забезпечує високу продуктивність і простоту використання, що робить її ефективним рішенням для малих та середніх підприємств.

Аннотація

У роботі досліджено комплекс задач, пов'язаних з управлінням персоналом в організаціях різного типу. Розроблено програмне забезпечення для автоматизації кадрових процесів, яке забезпечує облік персоналу, управління трудовими ресурсами та формування звітності. Основними функціями є управління даними працівників, ведення відпусток, формування звітів та забезпечення безпеки даних.

Ключові слова: управління персоналом, програмне забезпечення, база даних, автоматизація, інформаційні технології.

Annotation

The work investigates a set of tasks related to personnel management in various types of organizations. A software system has been developed to automate HR processes, providing personnel accounting, resource management, and reporting. The main features include employee data management, vacation tracking, report generation, and data security.

Key words: personnel management, software, database, automation, information technology.

Зміст

ПЕРЕЛІК ПОЗНАЧОК ТА СКОРОЧЕНЬ 5

ВСТУП 6

1 АНАЛІЗ ПРОБЛЕМАТИКИ ТА ПОСТАНОВКА ЗАДАЧ 10

1.1 Аналіз існуючих проблем в управлінні персоналом 10

1.2 Аналіз існуючих систем управління персоналом. Виявлення недоліків та обмежень в існуючих рішеннях 13

1.3 Обґрунтування необхідності розробки та постановка задач 19

2 ВИБІР ТЕХНОЛОГІЙ ТА МОДЕЛЮВАННЯ СИСТЕМИ 23

2.1 Вибір технологій та середовищ програмування 23

2.2 Моделювання основних бізнес-процесів в системі 29

2.3 Проєктування бази даних та архітектури системи 35

3 РЕАЛІЗАЦІЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ 50

3.1 Розроблення основних компонентів системи 50

3.2 Інтеграція бази даних з додатком 51

3.3 Розроблення модулів системи 55

3.4 Тестування розробленої системи та аналіз отриманих результатів 68

ВИСНОВКИ 79

СПИСОК ДЖЕРЕЛ ІНФОРМАЦІЇ 81

Додаток А Лістинги програми 86

Додаток Б Скрипти створення бази даних 96

ПЕРЕЛІК ПОЗНАЧОК ТА СКОРОЧЕНЬ

ІТ – Інформаційні технології

ПЗ – Програмне забезпечення

СУБД – Система управління базами даних

AES – Стандарт шифрування (Advanced Encryption Standard)

API – Інтерфейс прикладного програмування (Application Programming Interface)

DBMS – Система керування базами даних (Database Management System)

ERP – Система планування ресурсів підприємства (Enterprise Resource Planning)

HR – Управління людськими ресурсами (Human Resources)

IDE – Інтегроване середовище розробки (Integrated Development Environment)

REST – Репрезентативна передача стану (Representational State Transfer)

SQL – Мова структурованих запитів (Structured Query Language)

SSL – Протокол захищених сокетів (Secure Sockets Layer)

UI – Інтерфейс користувача (User Interface)

ВСТУП

В сучасних умовах розвитку економіки та глобалізації ринку праці управління персоналом набуває особливої актуальності, оскільки саме людський ресурс є основним фактором успіху будь-якої організації [1]. Зростаюча конкуренція на ринку змушує компанії шукати нові підходи до організації праці, зокрема, підвищення ефективності використання персоналу, що зумовлює необхідність розробки інноваційних систем управління людськими ресурсами. Розв'язання цих завдань вимагає впровадження комплексних систем управління, що дозволяють не лише забезпечувати виконання основних функцій, таких як підбір, розвиток і мотивація персоналу, але й передбачати та адаптуватися до змін у зовнішньому та внутрішньому середовищі організації.

На сьогоднішній день, традиційні методи управління персоналом часто не відповідають викликам сучасності, особливо у великих корпораціях та організаціях з багаторівневою структурою. Це обумовлено як швидким розвитком інформаційних технологій, так і змінами в підходах до організації праці, що вимагають адаптації управлінських процесів [2]. Більше того, у зв'язку з пандемією COVID-19, багато компаній змушені були переглянути свої стратегії управління персоналом, зокрема, впроваджувати віддалену роботу, нові методи оцінки та розвитку персоналу, а також інноваційні технології для підтримки мотивації та ефективності співробітників [3].

Незважаючи на численні дослідження в галузі управління персоналом, багато організацій все ще стикаються з проблемами ефективного використання людських ресурсів. Існуючі системи управління персоналом часто не враховують специфічних потреб окремих організацій або не забезпечують достатньої гнучкості у вирішенні проблем, пов'язаних з управлінням персоналом. Крім того, багато з цих систем є застарілими і не відповідають сучасним вимогам, що знижує їх ефективність і конкурентоспроможність.

Наукові дослідження у сфері управління персоналом зосереджені на розробці моделей і підходів, які б забезпечували ефективне управління людськими ресурсами в умовах швидких змін. Однак, багато з цих моделей залишаються теоретичними і не знаходять практичного застосування у реальних умовах. Це створює необхідність для розробки нових підходів і методів управління персоналом, які б базувалися на сучасних технологіях та були адаптовані до потреб конкретних організацій.

Актуальність дослідження у даній сфері також обумовлена тим, що Україна перебуває у стадії активного реформування багатьох секторів економіки, що вимагає нових підходів до управління персоналом. Це особливо важливо в умовах зростаючої конкуренції на ринку праці та необхідності підвищення ефективності управління ресурсами у державному та приватному секторах. Крім того, глобалізація і інтеграція української економіки у світовий ринок вимагають від українських компаній відповідати міжнародним стандартам управління персоналом, що також сприяє підвищенню актуальності дослідження у цій галузі.

Мета дослідження полягає у створенні комплексної системи управління персоналом, яка дозволить підвищити ефективність використання людських ресурсів, адаптувати управлінські процеси до сучасних викликів і забезпечити конкурентоспроможність організацій.

Основні задачі дослідження:

– провести аналіз існуючих проблем в управлінні персоналом, включаючи виявлення ключових чинників, що впливають на ефективність роботи персоналу;

– вивчити існуючі системи управління персоналом, визначити їхні недоліки та обмеження, які заважають ефективному управлінню людськими ресурсами;

– розробити концептуальну модель нової системи управління персоналом, яка б усувала виявлені недоліки та враховувала сучасні вимоги до управління;

– реалізувати основні компоненти системи, провести тестування та аналіз результатів з метою оцінки ефективності запропонованого рішення.

Об’єктом дослідження є процеси управління персоналом в організаціях, які визначають ефективність використання людських ресурсів та формують основу для досягнення стратегічних цілей компаній.

Предметом дослідження є методи та засоби, що використовуються в системах управління персоналом для підвищення ефективності управлінських процесів, забезпечення адаптивності до змінних умов середовища та оптимізації використання людських ресурсів.

Для досягнення поставленої мети у дослідженні використано такі методи:

– аналіз літературних джерел та існуючих рішень. Застосовано для дослідження та систематизації знань про існуючі системи управління персоналом, а також для виявлення їхніх недоліків та обмежень;

– порівняльний аналіз. Використано для порівняння різних систем управління персоналом, що дозволило визначити найефективніші підходи та інструменти для їх подальшого впровадження;

– моделювання бізнес-процесів. метод моделювання застосовано для розробки концептуальної моделі нової системи управління персоналом, що включає основні управлінські процеси та їх інтеграцію;

– методи проєктування та розробки програмного забезпечення. Використано для створення архітектури системи, проєктування бази даних та розроблення основних компонентів системи управління персоналом;

– емпіричне тестування. Застосовано для перевірки працездатності розробленої системи, що дозволило оцінити її ефективність та відповідність поставленим вимогам.

Практичне значення одержаних результатів полягає в тому, що розроблена система управління персоналом може бути використана для підвищення ефективності управлінських процесів у різних організаціях. Система готова до впровадження на підприємствах, що потребують оптимізації

використання людських ресурсів, із можливістю подальшого адаптування до специфічних вимог конкретної організації.

1 АНАЛІЗ ПРОБЛЕМАТИКИ ТА ПОСТАНОВКА ЗАДАЧ

1.1 Аналіз існуючих проблем в управлінні персоналом

Успіх сучасного підприємства значною мірою залежить від ефективності управління персоналом, що є однією з найважливіших функціональних підсистем підприємства і ключовим ресурсом для інноваційного розвитку. Людський ресурс відіграє важливу роль у формуванні інтелектуального потенціалу організації та впровадженні новітніх інформаційних технологій у виробничі процеси. Незважаючи на те, що раніше поняття «управління персоналом» не було широко розповсюджене в управлінській практиці, функції управління кадрами все ж виконувались, але перебували переважно в компетенції керівників підрозділів.

Традиційні кадрові служби на підприємствах виконували лише базові функції, такі як планування трудових ресурсів, прийняття та звільнення працівників, організація курсів підвищення кваліфікації та перепідготовки кадрів [4]. Відділи кадрів не були інтегровані з підрозділами, що займалися організацією праці, і мали обмежені можливості щодо оптимального управління персоналом в умовах сучасних інноваційних вимог.

З розвитком ринкової економіки та зниженням рівня централізації управлінських процесів, перед підприємствами постали нові, більш складні завдання в сфері управління трудовими ресурсами. Серед них – необхідність стимулювання працівників, створення конкурентоспроможних компенсаційних пакетів, збереження висококваліфікованих фахівців, які тепер мають ширший вибір щодо місця роботи та рівня винагороди [5]. В умовах високої інфляції підприємства зіткнулися з необхідністю індексації заробітної плати, а також із викликами підвищення продуктивності праці та оптимізації чисельності працівників під тиском конкурентного середовища.

Ці зміни в економічному та соціальному середовищі створили передумови для появи нових підходів до управління персоналом, які вимагали підвищення професіоналізму та інтеграції кадрових служб з іншими

підрозділами підприємства. У відповідь на ці виклики, сучасні підприємства почали активно впроваджувати системи управління персоналом, орієнтовані на довгострокове збереження і розвиток людського капіталу, що стало ключовим фактором їхнього стійкого розвитку в умовах глобалізації та посилення конкуренції.

На сьогодні кадрові служби підприємств стикаються з численними викликами, що вимагають ефективних рішень в управлінні персоналом (рис. 1.1). Незважаючи на важливість людського ресурсу для інноваційного розвитку, існують проблеми, пов’язані з плануванням, підбором, мотивацією та розвитком персоналу. Ці проблеми ускладнюють досягнення оптимальної ефективності управління кадрами, що, як показано на діаграмі, потребує нових підходів та інструментів.



Рис. 1.1. – Основні проблеми в управлінні персоналом

До основних проблем, які потребують уваги, належать такі аспекти:

– планування трудових ресурсів [6]. Визначення потреби в кадрах є ключовим етапом у досягненні стратегічних цілей підприємства. Це передбачає прогнозування майбутніх потреб у персоналі залежно від напрямків розвитку виробництва та визначення необхідної кількості і якості працівників;

– формування резерву персоналу [7]. З метою забезпечення стабільності виробничого процесу, важливо своєчасно здійснювати добір кандидатів та

визначати групи резерву, які можуть бути залучені в разі необхідності. Це забезпечує підприємство кваліфікованими кадрами у випадку появи вакантних посад або при збільшенні обсягів виробництва;

– відбір персоналу із груп резерву [8]. Ефективне управління резервом персоналу передбачає вибір найбільш підходящих кандидатів на певні посади із заздалегідь сформованого резерву, враховуючи їх професійні якості та відповідність вимогам до посади;

– оформлення трудових контрактів [9]. Процедура укладання трудових угод потребує не лише правового підходу, але й врахування інтересів обох сторін. Це забезпечує легітимність взаємовідносин між працівником та роботодавцем і формує основу для подальшої співпраці;

– оцінка рівня трудової діяльності. Оцінка результативності роботи кожного працівника дозволяє визначити його внесок у загальні результати діяльності підприємства, а також виявити сильні та слабкі сторони. Це, у свою чергу, допомагає приймати обґрунтовані рішення щодо кар'єрного зростання або необхідності додаткового навчання;

– переведення, підвищення, пониження, звільнення. В залежності від рівня продуктивності та результатів оцінки працівників, кадрова служба повинна здійснювати переведення на інші посади, підвищення, пониження або звільнення персоналу. Це дозволяє оптимізувати кадровий склад відповідно до потреб підприємства;

– профорієнтація та адаптація [10]. Важливою складовою управління персоналом є залучення нових працівників до колективу та до процесу виробництва, що включає не лише введення в курс справи, але й допомогу в адаптації до нових умов роботи;

– визначення заробітної плати та пільг [11]. Залучення та утримання кваліфікованих кадрів значною мірою залежить від правильного визначення рівня заробітної плати та пільг. Важливо, щоб ці фактори відповідали ринковим умовам і мотивували працівників залишатися в організації;

– навчання кадрів. Підтримання високого рівня кваліфікації працівників є необхідною умовою для успішної діяльності підприємства. Навчання і розвиток кадрів сприяють підвищенню їх продуктивності та здатності вирішувати складні виробничі задачі;

– підготовка керівних кадрів. Управлінська компетентність значною мірою визначає ефективність роботи організації. Тому підготовка та розвиток керівних кадрів є важливою задачею, яка спрямована на формування лідерських якостей, умінь приймати стратегічні рішення та керувати колективом.

Вирішення зазначених завдань вимагає сучасних інноваційних підходів, що виходять за межі традиційних функцій кадрових служб, таких як ведення документації, підготовка наказів, оформлення лікарняних листків та інших технічних аспектів. Для ефективного управління персоналом сьогодні необхідні нові навички, що включають стратегічне планування, використання інформаційних технологій та управління змінами, що дозволяють підвищити ефективність роботи всього колективу та забезпечити успішний розвиток організації в умовах сучасних викликів.

1.2 Аналіз існуючих систем управління персоналом. Виявлення недоліків та обмежень в існуючих рішеннях

На сучасному етапі розвитку бізнесу управління персоналом стало однією з ключових функцій, що забезпечує ефективність діяльності організацій. Існує багато різноманітних систем управління персоналом, які допомагають організаціям керувати своїми людськими ресурсами, забезпечуючи оптимізацію процесів найму, розвитку, мотивації та утримання співробітників. Однак, попри значний прогрес у цій сфері, більшість існуючих систем мають низку недоліків та обмежень, які знижують їхню ефективність та відповідність сучасним вимогам.

Багато сучасних систем управління персоналом орієнтовані на виконання стандартних функцій, таких як ведення обліку працівників, оформлення

документів, обробка заявок на відпустки та інші адміністративні завдання. Проте, ці рішення часто не враховують потреби в гнучкості та адаптивності до змінних умов ринку, що є критично важливим у динамічних бізнес-середовищах. Крім того, існують проблеми з інтеграцією таких систем з іншими управлінськими підсистемами підприємства, що призводить до фрагментації процесів і втрати ефективності.

SAP SuccessFactors – це комплексне рішення для управління людськими ресурсами, що надає широкий спектр інструментів для підбору, розвитку, мотивації та утримання персоналу (рис. 1.2) [12]. Застосунок орієнтований на автоматизацію ключових HR-процесів, включаючи управління талантами, планування кар'єри, навчання, управління ефективністю та компенсаціями.

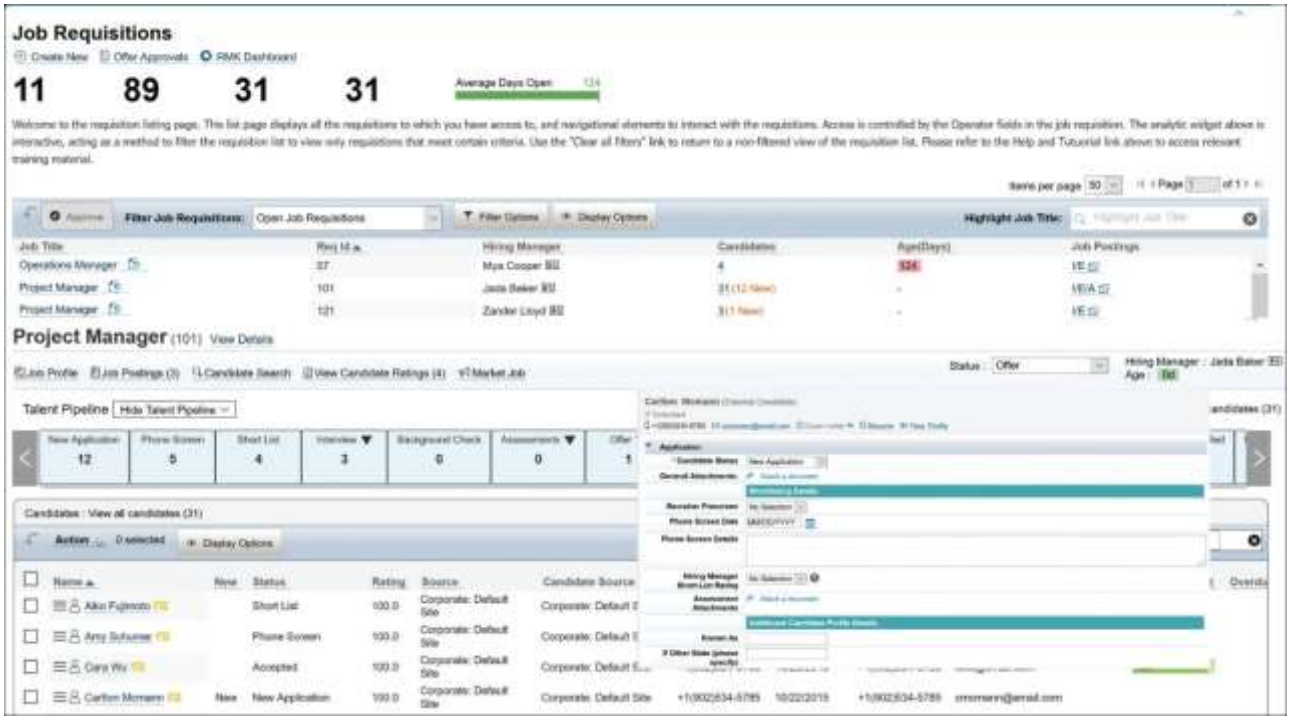


Рис. 1.2. – Приклад інтерфейсу «SAP SuccessFactors»

Архітектура системи базується на хмарній платформі, що забезпечує гнучкість, масштабованість і можливість інтеграції з іншими корпоративними системами. Компоненти SAP SuccessFactors організовані у вигляді модулів, що дозволяє налаштовувати рішення відповідно до специфічних потреб організації. Інтерфейс користувача орієнтований на зручність і простоту, що сприяє легкому освоєнню та використанню.

Переваги SAP SuccessFactors:

– комплексний набір інструментів для управління всіма аспектами HR;

– хмарна архітектура, що забезпечує гнучкість та масштабованість;

– модульна структура, яка дозволяє налаштовувати систему під потреби організації [13];

– зручний і інтуїтивний інтерфейс для користувачів.

Недоліки SAP SuccessFactors:

– висока вартість впровадження та обслуговування;

– складність налаштування та інтеграції з іншими системами без спеціалізованої підтримки [14];

– деякі модулі можуть мати обмежену функціональність у порівнянні з конкурентами;

– необхідність постійного доступу до інтернету для роботи з системою.

Отже, SAP SuccessFactors є потужним і комплексним рішенням для управління персоналом, що підходить для великих організацій з високими вимогами до HR-функцій. Його хмарна архітектура та модульна структура забезпечують гнучкість і масштабованість, однак висока вартість і складність налаштування можуть бути значними перешкодами для впровадження в менш масштабних організаціях.

Workday – це хмарне програмне забезпечення для управління підприємством, яке спеціалізується на управлінні людськими ресурсами, фінансами та аналітикою (рис. 1.3) [15]. Призначення Workday полягає в наданні інструментів для управління персоналом, включаючи найм, розвиток, управління ефективністю, компенсаціями та аналіз даних для підтримки стратегічних рішень в організаціях. Архітектура Workday побудована на базі хмарних технологій, що забезпечує високу гнучкість, масштабованість та інтеграцію з іншими бізнес-системами. Платформа об'єднує всі функції в єдиному середовищі, забезпечуючи користувачам доступ до актуальної

інформації та можливість роботи з будь-якої точки світу через веб-інтерфейс або мобільний додаток.

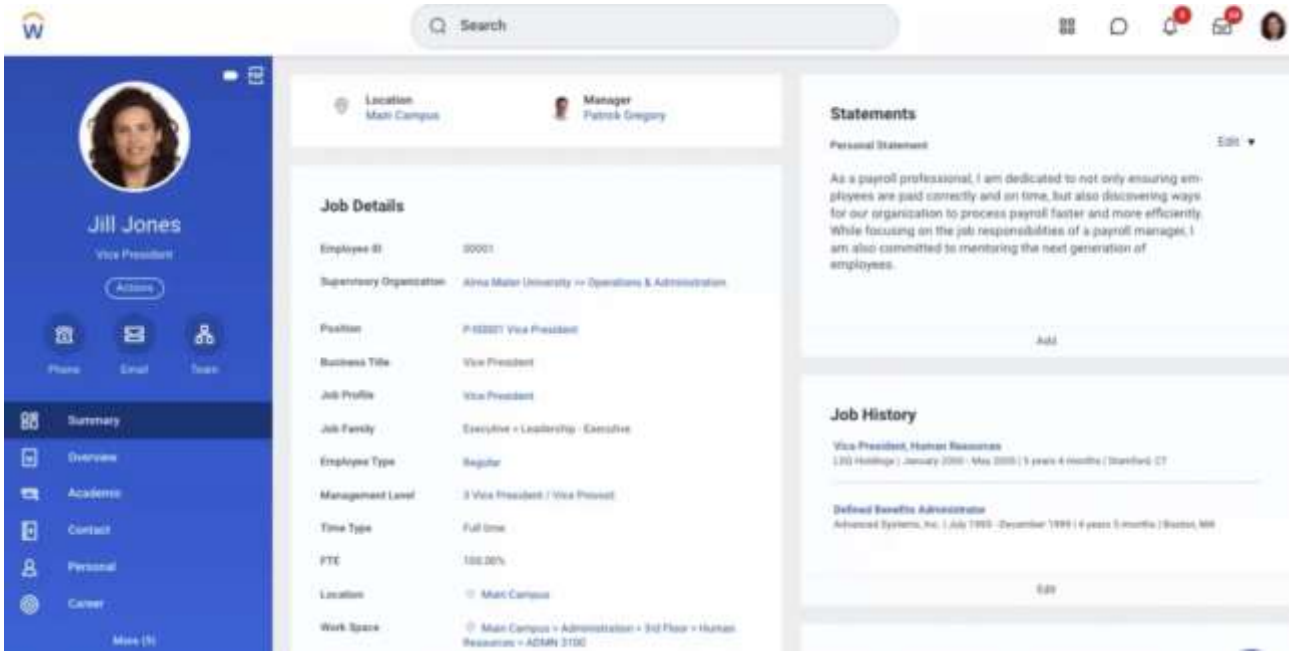


Рис. 1.3. – Приклад інтерфейсу «Workday»

Переваги Workday:

– інтеграція HR, фінансових і аналітичних функцій в одному середовищі;

– хмарна архітектура, що забезпечує доступ з будь-якої точки світу та високу масштабованість [16];

– інтуїтивний інтерфейс користувача, що полегшує використання для працівників різних рівнів;

– регулярні оновлення та вдосконалення, які автоматично впроваджуються без необхідності додаткових зусиль з боку користувача.

Недоліки Workday:

– висока вартість ліцензії та впровадження, що може бути значним бар'єром для малих і середніх підприємств [17];

– обмежені можливості кастомізації, що може не відповідати потребам великих організацій з особливими вимогами;

– потреба в постійному доступі до інтернету для роботи з системою;

– може вимагати значного часу на навчання користувачів через широку функціональність.

Отже, Workday – це потужне хмарне рішення для управління людськими ресурсами, фінансами та аналітикою, яке забезпечує інтеграцію ключових бізнес-функцій в єдиному середовищі. Завдяки своїй гнучкості та регулярним оновленням, Workday підходить для середніх та великих організацій, які прагнуть покращити ефективність управління. Проте висока вартість та обмежена кастомізація можуть бути перешкодами для деяких компаній, особливо малих і середніх.

Zoho People – це хмарне програмне забезпечення для управління людськими ресурсами, призначене для автоматизації та оптимізації HR-процесів в організаціях різного масштабу (рис. 1.4) [18]. Застосунок надає інструменти для управління даними працівників, відстеження відвідуваності, управління відпустками, оцінки ефективності, навчання та розвитку персоналу.

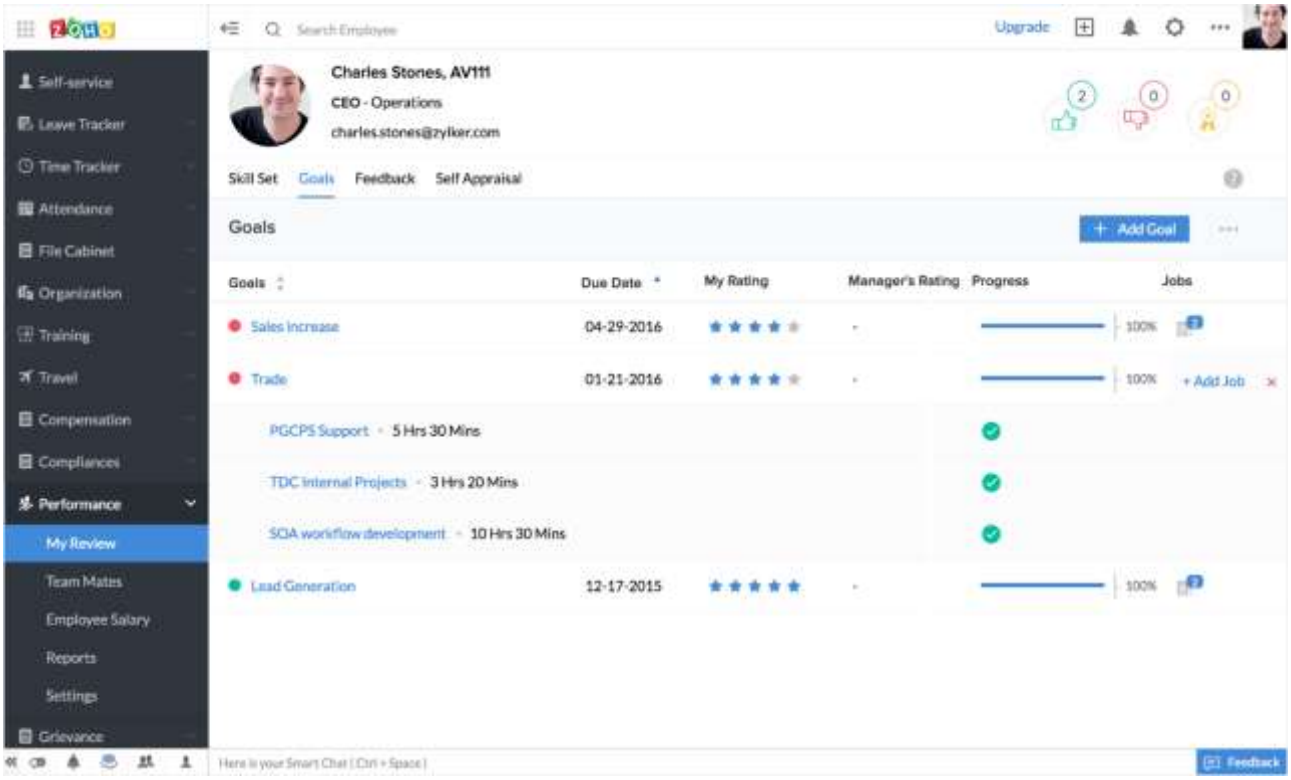


Рис. 1.4. – Приклад інтерфейсу «Zoho People»

Архітектура Zoho People базується на хмарній платформі, що забезпечує легкий доступ до системи з будь-якого місця і пристрою через веб-інтерфейс

або мобільний додаток. Інтеграція з іншими сервісами Zoho та сторонніми додатками дозволяє створити єдине середовище для управління бізнесом, забезпечуючи високу гнучкість і масштабованість системи.

Переваги Zoho People:

– інтуїтивний та зручний інтерфейс, який легко освоюється навіть для користувачів без спеціальної підготовки [19];

– гнучка система налаштувань, що дозволяє адаптувати функціонал до потреб різних організацій;

– інтеграція з іншими продуктами Zoho та сторонніми додатками, що створює єдине середовище для управління бізнесом;

– доступна ціна, що робить його привабливим варіантом для малих та середніх підприємств.

Недоліки Zoho People:

– обмежені можливості для масштабування в дуже великих організаціях з великою кількістю користувачів [20];

– обмежена функціональність у порівнянні з більш дорогими конкурентами, особливо в області аналітики та звітності;

– підтримка користувачів може бути менш оперативною, особливо у випадках складних технічних запитів;

– деякі функції можуть вимагати додаткових модулів або інтеграцій, що збільшує загальну вартість володіння.

Отже, Zoho People є ефективним і доступним рішенням для управління людськими ресурсами, яке підходить для малих та середніх підприємств. Завдяки зручному інтерфейсу та гнучким налаштуванням, система може бути легко адаптована до потреб різних організацій. Однак для великих корпорацій або компаній з високими вимогами до аналітики та масштабованості Zoho People може бути недостатньо потужним і вимагати додаткових рішень.

Виходячи із проведеного аналізу існуючих програмних рішень, стає очевидним, що більшість сучасних систем управління персоналом мають

складний функціонал, високу вартість впровадження та підтримки, що може бути недоступним для малих та середніх підприємств. Ці фактори створюють потребу в розробці нового рішення, яке пропонуватиме простіший і інтуїтивно зрозумілий функціонал, що дозволить організаціям швидко і легко впроваджувати систему без значних витрат на навчання користувачів. Безкоштовна або значно дешевша версія програмного забезпечення забезпечить доступ до основних функцій без додаткових витрат, що зробить його привабливим для компаній з обмеженим бюджетом. Таке рішення матиме потенціал стати ефективним інструментом для управління персоналом, особливо для підприємств, які шукають просте та доступне рішення без надмірного функціонального навантаження.

1.3 Обґрунтування необхідності розробки та постановка задач

Сучасні умови розвитку бізнесу та глобалізації ринку праці ставлять перед підприємствами нові виклики в управлінні персоналом. Існуючі програмні рішення для управління людськими ресурсами, такі як SAP SuccessFactors, Workday, Zoho People та інші, пропонують широкий спектр функціональності, проте вони мають низку обмежень, що ускладнює їх впровадження та використання в невеликих компаніях або організаціях з обмеженим бюджетом. Зокрема, ці системи часто є надто складними для користувачів без спеціальної підготовки, вимагають значних фінансових витрат на ліцензування та підтримку, а також можуть бути важкими для інтеграції з існуючими бізнес-процесами.

Одним із основних недоліків сучасних HR-систем є їх складний і перевантажений функціонал, який далеко не завжди потрібен для невеликих підприємств. Малі та середні компанії часто потребують простого та зрозумілого рішення, яке б дозволяло ефективно керувати базовими аспектами управління персоналом, такими як облік працівників, управління відпустками, планування робочого часу та оцінка ефективності. Надмірна кількість функцій, яка пропонується в існуючих рішеннях, може не тільки збільшувати складність

роботи з програмою, але й відволікати від основних завдань, які потребують вирішення.

Окрім того, висока вартість впровадження та обслуговування сучасних HR-систем є суттєвим бар'єром для багатьох підприємств. Компанії змушені витрачати значні кошти на ліцензування програмного забезпечення, його налаштування, навчання персоналу та підтримку в процесі експлуатації. Це робить такі рішення менш привабливими для організацій з обмеженими фінансовими ресурсами, які шукають ефективні способи оптимізації витрат.

Ще одним важливим аспектом є складність інтеграції сучасних систем з існуючими бізнес-процесами підприємства. Багато організацій вже мають встановлені інфраструктури та інструменти, які вони використовують у своїй щоденній діяльності. Впровадження нового HR-рішення часто вимагає значних змін в існуючій інфраструктурі, що може призвести до збоїв у роботі та додаткових витрат на адаптацію.

Виходячи з вищезазначеного, розробка нового рішення для управління персоналом є актуальною та обґрунтованою. Таке рішення повинно пропонувати простіший і більш інтуїтивний функціонал, що дозволить малим і середнім підприємствам швидко освоїти та впровадити систему без значних витрат. Безкоштовна або значно дешевша версія програмного забезпечення забезпечить доступ до основних функцій без необхідності великих інвестицій, що зробить його привабливим для компаній з обмеженим бюджетом. Крім того, легкість інтеграції з існуючими бізнес-процесами і можливість адаптації до конкретних потреб організації дозволить новому рішенню ефективно вписатися в будь-яке підприємство, незалежно від його розміру та специфіки діяльності.

Задля досягнення цих цілей, під час розробки особлива увага приділяється визначенню функціональних вимог, моделюванню системи, а також проєктуванню бази даних та архітектури системи. Це дозволить створити інструмент, що буде відповідати потребам сучасних організацій, забезпечуючи ефективність управління персоналом та можливість інтеграції з існуючими процесами на підприємстві.

Функціональні вимоги включають наступні компоненти:

– облік співробітників, включаючи додавання, редагування, видалення записів, а також зберігання інформації про особисті дані, контактні дані, історію роботи та інші релевантні дані;

– управління заявками на відпустку, включаючи подання, затвердження та відстеження відпусток;

– створення і підтримка структури організації, включаючи облік відділів;

– облік посад в організації, включаючи опис посадових обов'язків, вимог до кваліфікації та відповідальності, а також зв’язок з відділами та працівниками;

– забезпечення швидкого і зручного пошуку працівників за різними критеріями, такими як відділ, посада, персональні дані та інші параметри;

– генерація звітів по заробітній платі працівників з можливістю вибору періоду, відділу або посади;

– звіти, що відображають інформацію про працівників, зайнятих на конкретних посадах, включаючи дані про заробітну плату, кваліфікацію та інші релевантні показники;

– звіти, що охоплюють дані про всі відділи організації, включаючи кількість працівників, рівень їх кваліфікації, ефективність роботи та інші показники;

– звіти, що відображають інформацію про працівників, які перебувають у відпустці або мають залишки відпусток;

– управління обліковими записами користувачів, включаючи створення, редагування, видалення та контроль доступу до різних модулів системи;

– моніторинг і журналювання системних подій, що дозволяє відслідковувати дії користувачів та зміни в системі;

– можливість зміни даних опікового запису користувача;

– функціонал, що дозволяє змінювати активного користувача без необхідності виходу з системи, що забезпечує зручність для користувачів, які працюють з різними обліковими записами.

Моделювання системи є наступним етапом, на якому необхідно створити концептуальну модель, що відображає основні бізнес-процеси управління персоналом. Це дозволить визначити логіку взаємодії між різними модулями системи та забезпечити цілісність даних у процесі їх обробки.

Проєктування бази даних є важливою частиною роботи, що забезпечує зберігання, доступ та обробку інформації про персонал. База даних повинна бути розроблена таким чином, щоб забезпечити надійне збереження даних, їхню структурованість та можливість швидкого доступу до необхідної інформації.

Архітектура системи повинна забезпечувати ефективну роботу всіх компонентів, зокрема модулів управління обліковими записами, моніторингу системних подій, персоналізації інтерфейсу та зміни користувача. Розподіл функцій між адміністратором та користувачем має бути чітко визначеним для забезпечення безпеки та ефективності використання системи.

Виконання цих задач створює основу для подальшої розробки системи, що буде гнучкою, надійною та відповідатиме сучасним вимогам управління персоналом.

2 ВИБІР ТЕХНОЛОГІЙ ТА МОДЕЛЮВАННЯ СИСТЕМИ

2.1 Вибір технологій та середовищ програмування

Після визначення основних функціональних вимог та архітектури системи управління персоналом, наступним критично важливим кроком є вибір відповідних технологій та середовищ програмування, які забезпечать ефективну реалізацію поставлених задач. Вибір технологій повинен базуватися на декількох ключових критеріях, таких як відповідність вимогам до функціональності, можливість масштабування, легкість інтеграції з іншими системами, а також підтримка сучасних стандартів безпеки та продуктивності.

Середовище програмування, яке буде обране для розробки, повинно забезпечити зручність у використанні, гнучкість у налаштуванні, а також можливість швидкого розгортання і підтримки системи. Крім того, важливим фактором є наявність великої спільноти розробників та широкої бази знань, що дозволить ефективно вирішувати виникаючі технічні питання і забезпечить надійність кінцевого продукту. У даному підрозділі буде розглянуто можливі технології та середовища програмування, а також обґрунтовано вибір оптимального рішення для реалізації даного проєкту.

При виборі мови програмування для розробки системи управління персоналом важливо враховувати кілька факторів, таких як продуктивність, гнучкість, підтримка сучасних технологій, а також наявність інструментів для інтеграції та розширення функціональності. Мова програмування повинна забезпечувати ефективну розробку як серверної, так і клієнтської частин системи, підтримувати об'єктно-орієнтоване програмування та мати широку спільноту для підтримки і розвитку.

Python – це мова програмування високого рівня, відома своєю простотою та читабельністю, що робить її ідеальною для швидкої розробки прототипів і застосунків [21]. Вона широко використовується для веб-розробки, аналізу даних, автоматизації та створення штучного інтелекту завдяки своїй великій кількості бібліотек і фреймворків.

Java – це об'єктно-орієнтована мова програмування, яка забезпечує високу продуктивність і надійність, особливо в масштабних корпоративних додатках [22]. Java має багатий набір бібліотек і фреймворків, що дозволяє створювати різноманітні рішення, від веб-додатків до мобільних і настільних програм.

C# – це мова програмування, розроблена компанією Microsoft, яка є частиною платформи .NET і відома своєю потужністю для розробки як веб-додатків, так і настільних програм [23]. C# забезпечує інтеграцію з широким спектром інструментів і технологій Microsoft, що робить її ідеальним вибором для розробки корпоративних додатків, особливо в середовищах Windows.

У табл. 2.1 проведено порівняльний аналіз мов програмування Python, Java та C#, зосереджений на ключових характеристиках. Цей аналіз допомагає визначити переваги мови C# для розробки системи управління персоналом.

Таблиця 2.1. – Порівняльний аналіз мов програмування

Характеристика	Python	Java	C#
Інтеграція з екосистемою Microsoft	Обмежена підтримка	Обмежена підтримка	Висока інтеграція з інструментами та сервісами Microsoft (Azure, Visual Studio, SQL Server)
Розробка корпоративних додатків	Використовується, але не основний вибір	Популярний вибір, особливо для великих систем	Ідеальний вибір для корпоративних рішень, особливо в середовищі Windows
Підтримка настільних додатків	Обмежена	Можлива, але не найзручніша	Широкі можливості для розробки настільних додатків, особливо з WPF та WinForms
Підтримка асинхронного програмування	Вбудована підтримка, але не завжди найефективніша	Добра підтримка, але потребує додаткових налаштувань	Потужна підтримка асинхронного програмування з async/await та Task паралелізмом
Стабільність та продуктивність	Добра	Висока	Висока стабільність та продуктивність

Виходячи із проведеного аналізу, мова програмування C# є оптимальним вибором для розробки системи управління персоналом. C# забезпечує високу інтеграцію з екосистемою Microsoft, що є важливим фактором для корпоративних рішень, особливо в середовищі Windows. Крім того, потужна підтримка асинхронного програмування дозволяє створювати високопродуктивні та надійні додатки, які можуть ефективно обробляти паралельні завдання. Стабільність, продуктивність та широкі можливості для розробки настільних додатків роблять C# ідеальним інструментом для реалізації даної системи, забезпечуючи її ефективне функціонування та масштабованість.

При виборі системи управління базами даних (СУБД) для розробки системи управління персоналом необхідно враховувати кілька важливих критеріїв, таких як продуктивність, надійність, масштабованість та інтеграція з обраною мовою програмування. Важливо також звернути увагу на підтримку складних запитів, можливість роботи з великими обсягами даних та забезпечення високого рівня безпеки. Обраний інструмент повинен відповідати вимогам сучасних бізнес-додатків і забезпечувати ефективну роботу системи в реальних умовах експлуатації.

MS SQL Server – це потужна реляційна СУБД, розроблена корпорацією Microsoft, яка забезпечує високу продуктивність і надійність при роботі з великими обсягами даних [24]. Вона інтегрується з платформою .NET, що робить її ідеальним вибором для додатків, розроблених на C#, особливо в корпоративному середовищі Windows. MS SQL Server також пропонує широкий набір інструментів для адміністрування баз даних, підтримку складних аналітичних запитів і забезпечення високого рівня безпеки даних [25].

MySQL – це одна з найпопулярніших відкритих реляційних СУБД, яка використовується для розробки веб-додатків і систем з високим навантаженням [26]. MySQL відзначається своєю швидкістю, простотою у використанні та

підтримкою різних операційних систем, що робить її привабливим вибором для різноманітних проектів, особливо тих, що базуються на відкритому коді.

Oracle – це одна з найпотужніших і найбільш надійних СУБД на ринку, відома своїми широкими можливостями для роботи з великими корпоративними даними. Вона забезпечує високу продуктивність, масштабованість і надійність, а також підтримує складні транзакції та розподілені бази даних.

У табл. 2.2 проведено порівняльний аналіз систем управління базами даних MS SQL Server, MySQL та Oracle за ключовими характеристиками. Цей аналіз допомагає обґрунтувати вибір MS SQL Server як найбільш відповідного рішення для розробки системи управління персоналом.

Таблиця 2.2. – Порівняльний аналіз СУБД

Характеристика	MS SQL Server	MySQL	Oracle
Інтеграція з екосистемою Microsoft	Висока інтеграція з інструментами та сервісами Microsoft (Azure, Visual Studio, .NET)	Обмежена інтеграція	Часткова інтеграція через додаткові інструменти
Підтримка складних аналітичних запитів	Відмінна підтримка аналітичних запитів через OLAP та інші сервіси	Обмежена підтримка, потребує додаткових налаштувань	Висока підтримка, але складніша у налаштуванні
Інструменти адміністрування	Потужні вбудовані інструменти для адміністрування та моніторингу	Менше вбудованих інструментів, потрібні сторонні рішення	Складні та багатофункціональні інструменти, які вимагають спеціальних знань
Інтеграція з .NET та C#	Ідеальна інтеграція з платформою .NET та мовою C#	Обмежена інтеграція	Інтеграція можлива, але складніша і потребує додаткових налаштувань
Безпека даних	Вбудовані розширені	Обмежені	Високий рівень

	механізми безпеки та підтримка compliance (HIPAA, GDPR)	вбудовані механізми безпеки	безпеки, але складність налаштування

MS SQL Server є оптимальним вибором для розробки системи управління персоналом завдяки високому рівню інтеграції з екосистемою Microsoft, що забезпечує безшовну роботу з мовою програмування C# та платформою .NET. Розширені інструменти адміністрування та підтримка складних аналітичних запитів дозволяють ефективно керувати великими обсягами даних і забезпечують високий рівень продуктивності. Крім того, MS SQL Server пропонує потужні механізми безпеки, які відповідають сучасним стандартам захисту даних, що є критично важливим для надійного функціонування корпоративних систем. Ці характеристики роблять MS SQL Server ідеальним рішенням для створення стабільної та масштабованої системи управління персоналом.

При виборі середовища розробки для реалізації системи управління персоналом на мові C# важливо враховувати такі фактори, як зручність використання, підтримка інтеграції з іншими інструментами та сервісами, а також можливості розширення та налаштування. Середовище розробки повинно забезпечувати ефективну роботу з кодом, налагодженням і тестуванням, а також підтримувати сучасні стандарти розробки програмного забезпечення. Серед популярних інструментів для розробки на C# розглядаються Visual Studio 2022, Visual Studio Code та Rider, кожне з яких має свої переваги та особливості, які можуть вплинути на вибір залежно від потреб проекту.

Visual Studio 2022 – це потужне середовище розробки, розроблене компанією Microsoft, яке забезпечує широкий набір інструментів для розробки на C#, зокрема інтеграцію з платформою .NET, повноцінне налагодження, тестування та профілювання додатків [28]. Visual Studio 2022 підтримує роботу з великими проєктами, забезпечуючи високу продуктивність і зручність

завдяки своїм інтелектуальним функціям, таким як IntelliSense та інтеграція з Azure.

Visual Studio Code – це легке, безкоштовне та розширюване текстове середовище розробки, яке підтримує широкий спектр мов програмування, включаючи C# [29]. Хоча Visual Studio Code не настільки функціонально насичене, як Visual Studio 2022, воно забезпечує гнучкість через велику кількість розширень, що дозволяють налаштувати середовище під конкретні потреби розробника.

Rider – це середовище розробки від JetBrains, яке спеціалізується на C# та підтримує повну інтеграцію з .NET Framework, .NET Core та Mono [30]. Rider забезпечує високу продуктивність, швидке налагодження та аналіз коду завдяки вбудованим інструментам і можливостям, включаючи рефакторинг, аналіз продуктивності та перевірку коду на льоту.

У табл. 2.3 проведено порівняльний аналіз середовищ розробки Visual Studio 2022, Visual Studio Code та Rider за ключовими характеристиками. Цей аналіз допомагає обґрунтувати вибір Visual Studio 2022 як найбільш відповідного інструменту для розробки системи управління персоналом.

Таблиця – 2.3. Порівняльний аналіз середовищ розробки

Характеристика	Visual Studio 2022	Visual Studio Code	Rider
Інтеграція з платформою .NET	Глибока інтеграція з повною підтримкою всіх функцій .NET	Підтримка через розширення, але з обмеженою функціональністю	Підтримка .NET, але через сторонні рішення
Налагодження та профілювання	Потужні вбудовані інструменти для налагодження, профілювання та аналізу продуктивності	Обмежені можливості налагодження через розширення	Хороші можливості налагодження, але не такі глибокі, як у Visual Studio 2022
Підтримка великих проєктів	Оптимізовано для роботи з великими	Менш підходить для великих проєктів	Добра підтримка великих проєктів,

	проєктами, включаючи функції для управління складними рішеннями	через спрощений функціонал	але може поступатися у продуктивності
Інтелектуальні функції (IntelliSense, рефакторинг)	Розширені інтелектуальні функції	Основні інтелектуальні функції	Добра підтримка рефакторингу та IntelliSense

Visual Studio 2022 є оптимальним вибором для розробки системи управління персоналом завдяки своїй глибокій інтеграції з платформою .NET, що забезпечує повну підтримку всіх функцій C# та інших технологій Microsoft. Це середовище пропонує потужні інструменти для налагодження, профілювання та аналізу продуктивності, що дозволяє ефективно розробляти, тестувати та оптимізувати великі і складні проєкти. Повна інтеграція з Azure надає додаткові можливості для роботи з хмарними сервісами, DevOps та CI/CD, що є важливим для сучасних корпоративних рішень. Розширені інтелектуальні функції, такі як IntelliSense та рефакторинг, підвищують продуктивність розробників, забезпечуючи швидке і точне написання коду. Ці характеристики роблять Visual Studio 2022 ідеальним середовищем для створення масштабованої та ефективної системи управління персоналом.

2.2 Моделювання основних бізнес-процесів в системі

Для створення ефективної системи управління персоналом важливо ретельно змоделювати основні бізнес-процеси, які відбуваються в організації. Це включає визначення ключових функцій, які система повинна підтримувати, а також способи взаємодії користувачів із системою для виконання цих функцій. Моделювання допомагає зрозуміти, як система інтегрується в існуючі робочі процеси, забезпечуючи оптимізацію часу та підвищення ефективності управлінських завдань [31].

Одним з інструментів, що дозволяє описати взаємодію користувачів із системою, є діаграми варіантів використання (use case). Вони дозволяють

наочно представити, які саме дії виконуватимуться користувачами в системі та як ці дії взаємопов'язані між собою. Це включає процеси реєстрації, автентифікації, управління персоналом, відпустками, відділами, посадами, а також пошук та формування звітності.

У табл. 2.4 представлено основні варіанти використання системи, які охоплюють ключові дії користувачів та описують їх функціональне призначення.

Таблиця 2.4. – Опис варіантів використання системи

№	Назва прецеденту	Опис
UC1	Реєстрація користувача	Забезпечує реєстрацію користувача у системі
UC2	Автентифікація в системі	Надає можливість користувачеві пройти автентифікацію в системі
UC3	Вивід каталогу користувачів	Надає системному адміністратору можливість перегляду списку всіх користувачів
UC4	Додати користувача	Дає системному адміністратору можливість внести дані нового користувача до системи
UC5	Редагувати користувача	Дозволяє системному адміністратору оновити інформацію про користувача з каталогу
UC6	Вивід списку персоналу	Надає доступ до перегляду списку всіх працівників у системі
UC7	Додати персонал	Дозволяє системному адміністратору внести дані про нового працівника
UC8	Редагувати персонал	Дозволяє системному адміністратору оновити інформацію про працівника
UC9	Вивід списку відпусток	Надає доступ до перегляду списку всіх відпусток персоналу
UC10	Додати відпустку	Дозволяє адміністратору додати нову відпустку до системи
UC11	Редагувати відпустку	Дозволяє адміністратору оновити інформацію про відпустку
UC12	Вивід списку відділів	Надає доступ до перегляду списку всіх відділів в

		організації
UC13	Додати відділ	Дозволяє адміністратору додати новий відділ до структури організації
UC14	Редагувати відділ	Дозволяє адміністратору оновити інформацію про відділ
UC15	Вивід списку посад	Надає доступ до перегляду списку всіх посад у системі
UC16	Додати посаду	Дозволяє адміністратору додати нову посаду до системи
UC17	Редагувати посаду	Дозволяє адміністратору оновити інформацію про посаду
UC18	Пошук персоналу	Забезпечує можливість пошуку працівників за різними критеріями
UC19	Формування звітності	Забезпечує формування звітів по заробітній платі, посадах, відділах, відпустках

На основі вимог до системи розроблено use-case діаграми прецедентів для двох основних ролей: адміністратора та користувача. Діаграми прецедентів представлені на рис. 2.1 та рис. 2.2 відповідно, де відображені основні сценарії взаємодії цих ролей із системою. Кожна з діаграм описує можливі дії та функціональні можливості, доступні для кожної ролі, забезпечуючи наочне уявлення про взаємодію користувачів із системою.

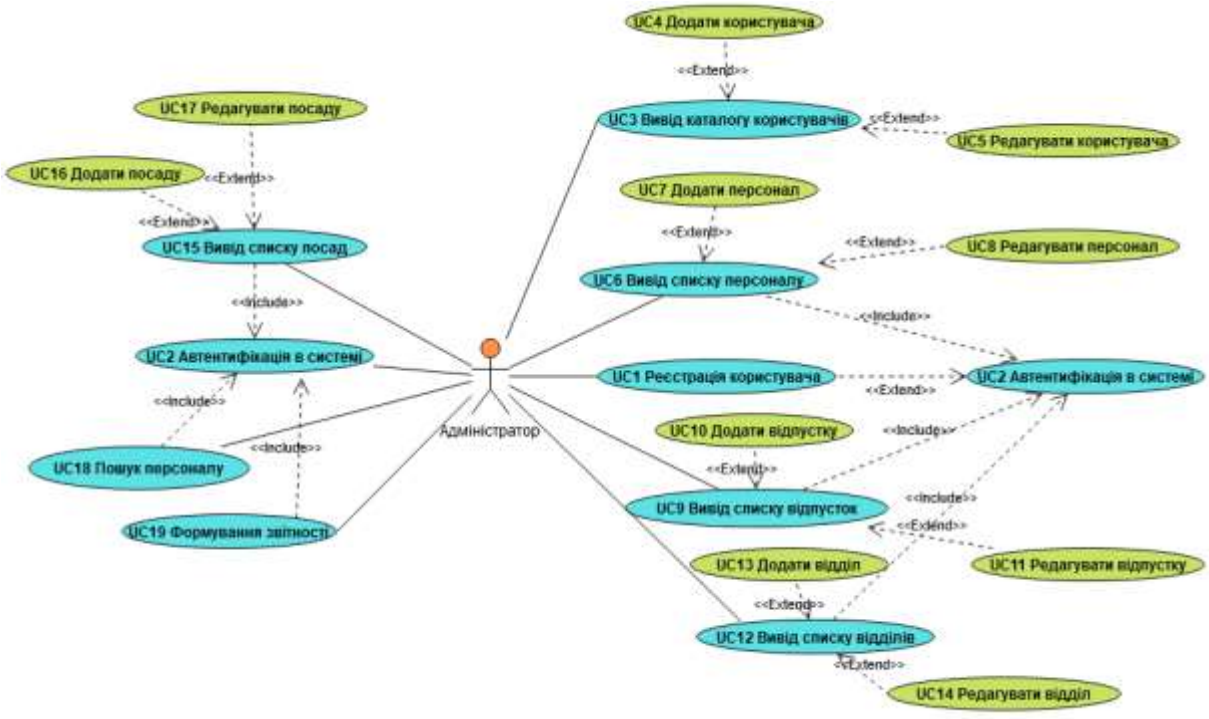


Рис. 2.1. – Діаграма use-case для ролі адміністратора

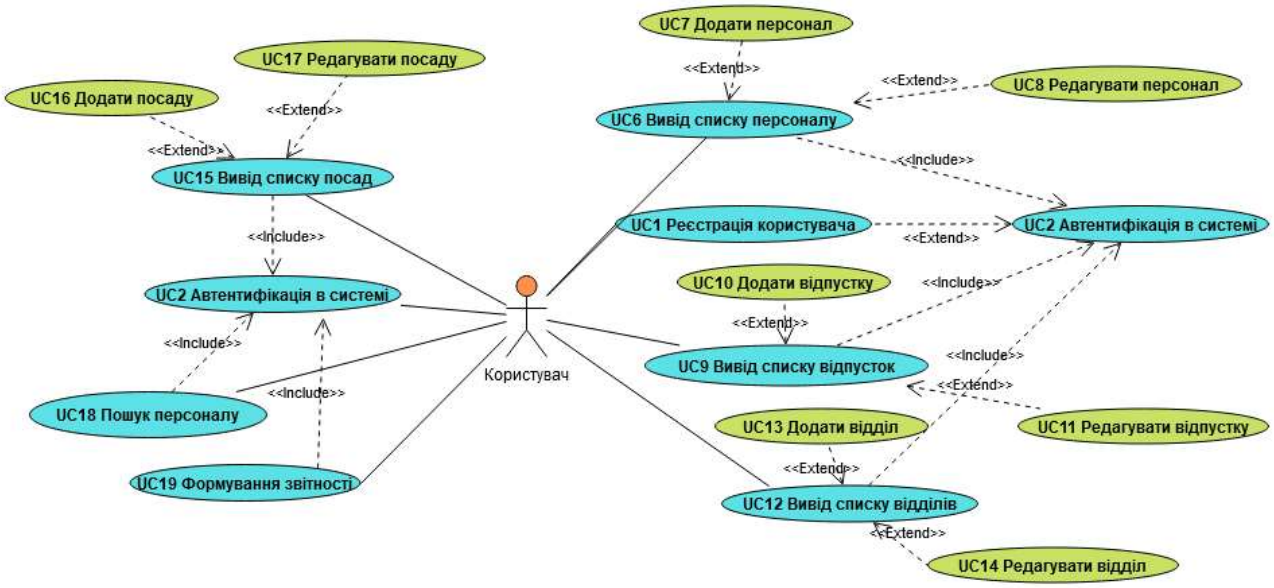


Рис. 2.2. – Діаграма use-case для ролі користувача

Діаграми послідовності відображають порядок взаємодії між різними об'єктами системи під час виконання певних процесів [32]. Вони показують, як відбувається передача повідомлень між об'єктами, та як вони співпрацюють для досягнення конкретної мети. На рис. 2.3 представлено діаграму послідовності процесу додавання інформації про персонал.

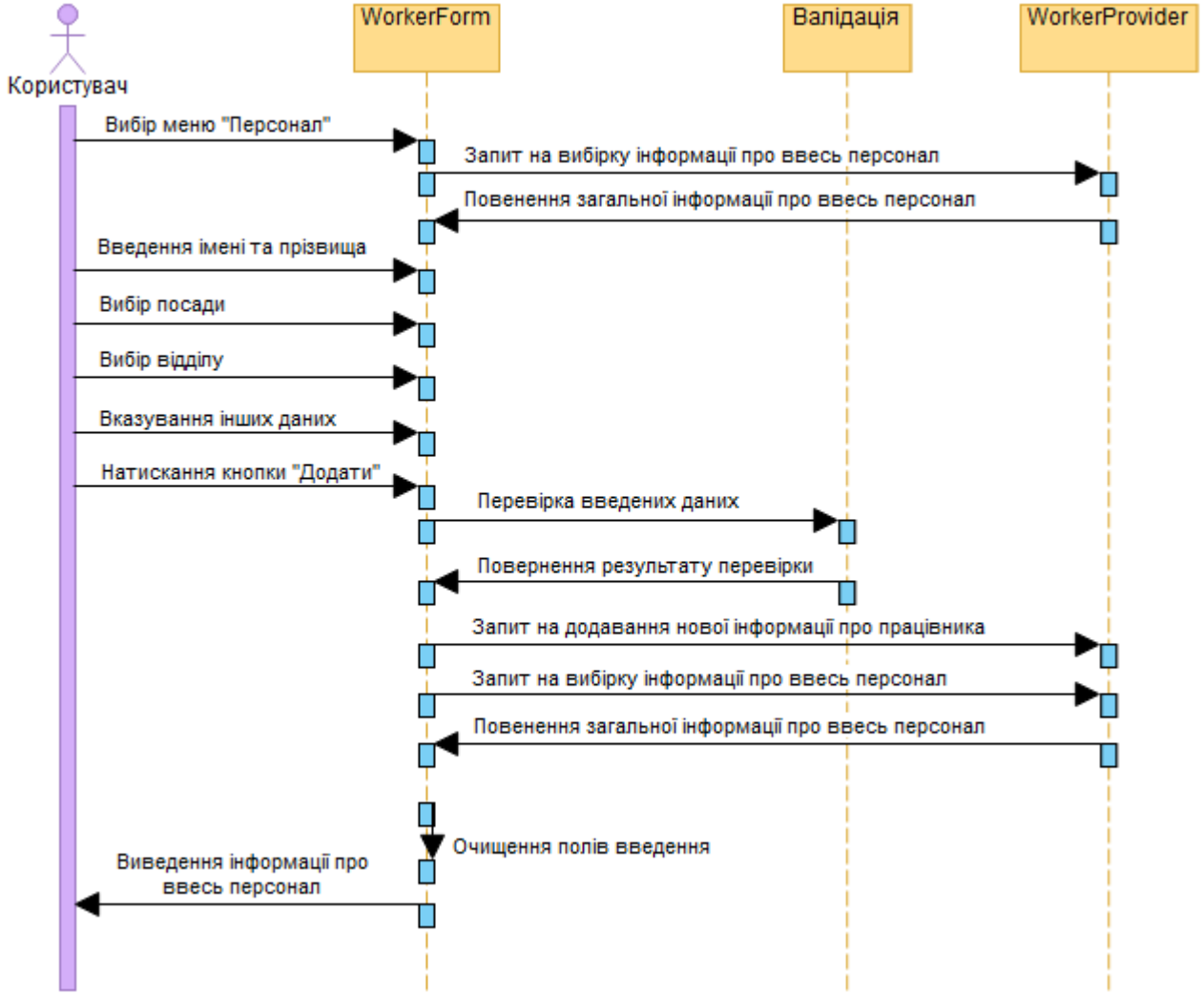


Рис. 2.3. – Діаграма послідовності процесу додавання інформації про персонал

Користувач розпочинає процес, вибираючи меню «Персонал», після чого вводить ім'я та прізвище працівника, обирає посаду, відділ та вказує інші необхідні дані. Після натискання кнопки «Додати», форма WorkerForm надсилає запит на вибірку інформації про весь персонал для перевірки наявності даних у системі. Після цього виконується перевірка введених даних, результати якої повертаються до форми. У разі успішної перевірки, формується запит на додавання нової інформації про працівника. Після успішного додавання, WorkerForm оновлює дані, запитуючи та повертаючи загальну інформацію про весь персонал, і очищає поля введення для можливості подальшого використання. Нарешті, користувачу відображається оновлена інформація про весь персонал, що підтверджує успішне завершення процесу.

На рис. 2.4 представлено діаграму послідовності процесу редагування інформації обраного працівника.

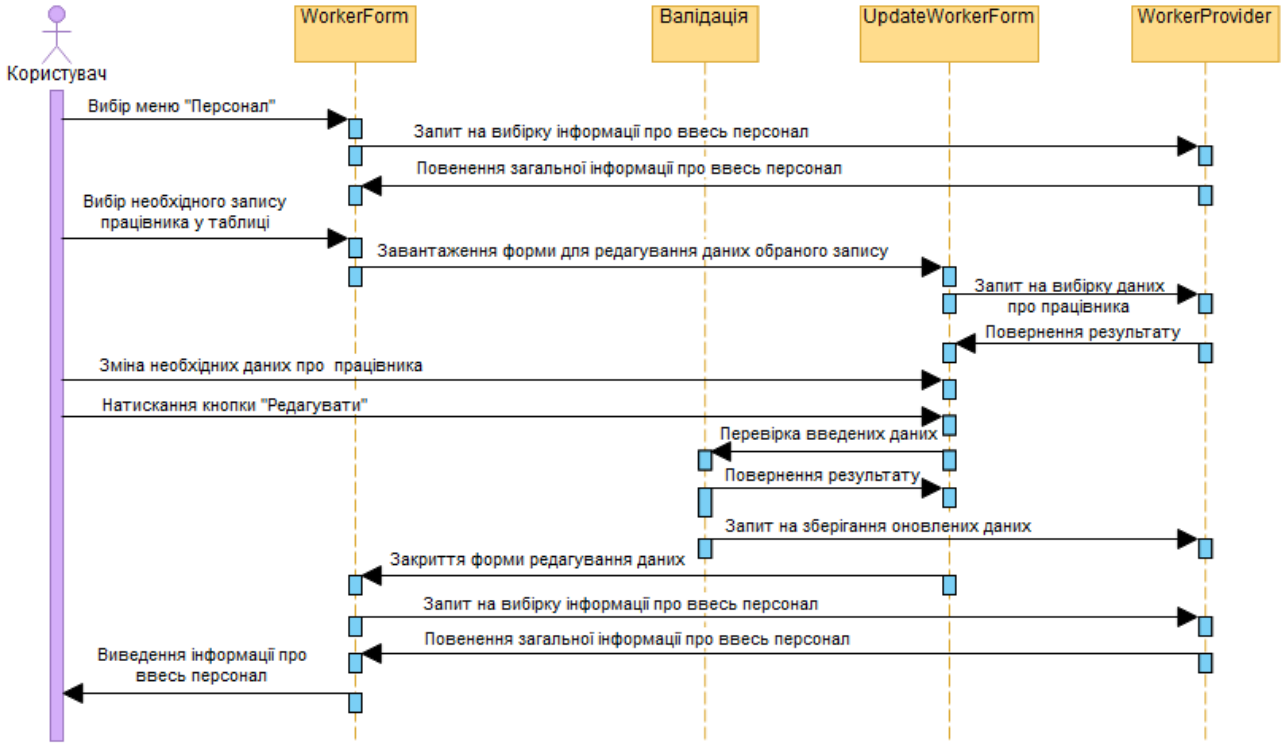


Рис. 2.4. – Діаграма процесу редагування інформації обраного працівника

Користувач розпочинає процес, вибираючи меню «Персонал» та обираючи необхідний запис працівника в таблиці. Після цього WorkerForm ініціює запит на вибірку інформації про весь персонал та повертає загальну інформацію для відображення. Далі завантажується форма для редагування даних обраного працівника. Користувач змінює необхідні дані про працівника та натискає кнопку «Редагувати», після чого здійснюється перевірка введених даних. Результати перевірки повертаються користувачеві, і якщо вони успішні, формується запит на збереження оновлених даних. Після успішного збереження форма редагування даних закривається, і WorkerForm знову запитує та повертає загальну інформацію про весь персонал для відображення оновлених даних.

Користувачу виводиться оновлена інформація про весь персонал, що підтверджує завершення процесу редагування.

2.3 Проєктування бази даних та архітектури системи

Проєктування бази даних є однією з найважливіших складових при створенні ефективної системи управління персоналом. В рамках цього проекту для зберігання та організації даних було обрано сутнісно-зв'язкову модель (ER-модель), яка відображає логічну структуру бази даних та забезпечує ефективне управління інформацією. На рис. 2.5 наведено інформаційну модель бази даних, яка використовується в системі.

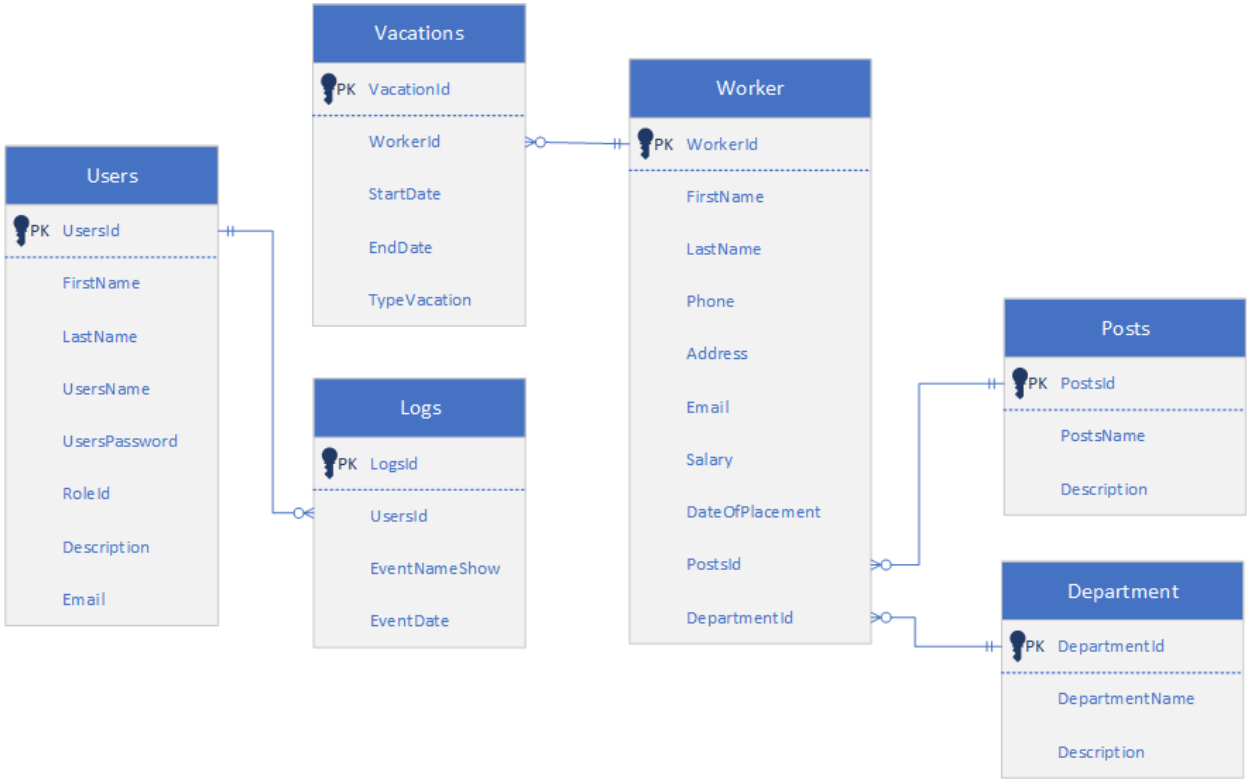


Рис. 2.5. – Інфологічна модель бази даних

Ця модель включає кілька таблиць, кожна з яких відіграє ключову роль у зберіганні та обробці даних, пов'язаних із персоналом, відділами, посадами, відпустками та іншими аспектами управління. Взаємозв'язки між таблицями дозволяють забезпечити цілісність даних та можливість виконання складних запитів, що є критично важливим для підтримки функціонування всієї системи. Ця структура також сприяє ефективній організації даних, забезпечуючи

швидкий доступ до необхідної інформації та підтримку різноманітних операцій і звітів.

Як можна побачити із рис. 2.5 база даних складається із 6-ти сутностей. Нижче розглянуто атрибути та значення для кожної із них.

Таблиця «Users» призначена для зберігання інформації про користувачів системи. Вона містить ключові дані, необхідні для ідентифікації користувачів, а також для управління їхніми обліковими записами та правами доступу в системі (табл. 2.5).

Таблиця 2.5. – Атрибути сутності «Users»

№	Найменування	Тип даних	Призначення
1	UsersId	INT	Унікальний ідентифікатор користувача, використовується як первинний ключ.
2	FirstName	NVARCHAR (50)	Ім'я користувача, дозволяє зберігати особисте ім'я.
3	LastName	NVARCHAR (50)	Прізвище користувача, дозволяє зберігати фамілію користувача.
4	UsersName	NVARCHAR (50)	Унікальне ім'я користувача в системі, використовується для входу.
5	UsersPassword	NVARCHAR (120)	Зашифрований пароль користувача, забезпечує захист даних користувача.
6	RoleId	INT	Ідентифікатор ролі користувача в системі, визначає права доступу користувача.
7	Description	NVARCHAR (1000)	Додатковий опис користувача або його ролі в системі.
8	Email	NVARCHAR (150)	Електронна пошта користувача, використовується для комунікації та відновлення доступу.

Таблиця «Vacations» призначена для зберігання інформації про відпустки працівників. Вона містить дані, які дозволяють фіксувати періоди відсутності працівників на робочому місці та типи відпусток, що надаються персоналу (табл. 2.6).

Таблиця 2.6. – Атрибути сутності «Vacations»

№	Найменування	Тип даних	Призначення
1	VacationId	INT	Унікальний ідентифікатор відпустки, використовується як первинний ключ.
2	WorkerId	INT	Ідентифікатор працівника, що перебуває у відпустці, зв’язаний із таблицею працівників.
3	StartDate	DATETIME	Дата початку відпустки, що дозволяє визначити період відсутності працівника.
4	EndDate	DATETIME	Дата завершення відпустки, що дозволяє відстежувати тривалість відпустки.
5	TypeVacation	NVARCHAR (100)	Тип відпустки, що вказує на характер або причину надання відпустки.

Таблиця «Logs» створена для зберігання інформації про події, що відбуваються в системі, зокрема, про дії користувачів. Вона містить основні дані, які допомагають відстежувати події в системі, що важливо для моніторингу та аудиту дій користувачів (табл. 2.7).

Таблиця 2.7. – Атрибути сутності «Logs»

№	Найменування	Тип даних	Призначення
1	LogsId	INT	Унікальний ідентифікатор події, використовується як первинний ключ.
2	UsersId	INT	Ідентифікатор користувача, що ініціював подію, дозволяє зв’язати подію з користувачем.
3	EventNameShow	NVARCHAR (MAX)	Назва події, що відображається, описує тип або зміст події.
4	EventDate	DATETIME	Дата і час події, що дозволяє відстежувати хронологію подій.

Таблиця «Worker» призначена для зберігання інформації про працівників організації. Вона містить основні дані, необхідні для ідентифікації працівників, а також для управління їхніми контактними та службовими даними, що є критично важливими для ефективного управління персоналом (табл. 2.8).

Таблиця 2.8. – Атрибути сутності «Worker»

№	Найменування	Тип даних	Призначення
1	WorkerId	INT	Унікальний ідентифікатор працівника, використовується як первинний ключ.
2	FirstName	NVARCHAR (50)	Ім'я працівника, що дозволяє зберігати особисте ім'я.
3	LastName	NVARCHAR (50)	Прізвище працівника, що дозволяє зберігати фамілію.
4	Phone	NVARCHAR (25)	Номер телефону працівника для зв’язку.
5	Address	NVARCHAR (MAX)	Адреса проживання працівника.
6	Email	NVARCHAR (250)	Електронна пошта працівника, використовується для комунікації.
7	Salary	FLOAT (53)	Заробітна плата працівника, що відображає його фінансову компенсацію.
8	DateOfPlacement	DATETIME	Дата прийому працівника на роботу, що дозволяє відстежувати термін його служби.
9	PostsId	INT	Ідентифікатор посади, яка займає працівник, зв’язаний із таблицею посад.
10	DepartmentId	INT	Ідентифікатор відділу, до якого належить працівник, зв’язаний із таблицею відділів.

Таблиця «Department» призначена для зберігання інформації про відділи в організації. Вона містить дані, що допомагають структурувати організаційну ієрархію та описувати функції різних відділів (табл. 2.9).

Таблиця 2.9. – Атрибути сутності «Department»

№	Найменування	Тип даних	Призначення
1	DepartmentId	INT	Унікальний ідентифікатор відділу, використовується як первинний ключ.

2	DepartmentName	NVARCHAR (250)	Назва відділу, яка дозволяє ідентифікувати відділ у структурі організації.
3	Description	NVARCHAR (MAX)	Опис відділу, який може включати його функції, завдання та інші характеристики.

Таблиця «Posts» призначена для зберігання інформації про посади в організації. Вона містить дані, які дозволяють описати кожну посаду та її функціональні обов'язки, що є важливою частиною управління людськими ресурсами (табл. 2.10).

Таблиця 2.10. – Атрибути сутності «Posts»

№	Найменування	Тип даних	Призначення
1	PostsId	INT	Унікальний ідентифікатор посади, використовується як первинний ключ.
2	PostsName	NVARCHAR (250)	Назва посади, що ідентифікує її в організаційній структурі.
3	Description	NVARCHAR (MAX)	Опис посади, що включає функціональні обов'язки та вимоги до кандидата.

На рис. 2.7 представлена сутнісно-зв’язкова діаграма, розроблена з урахуванням специфіки системи управління персоналом.

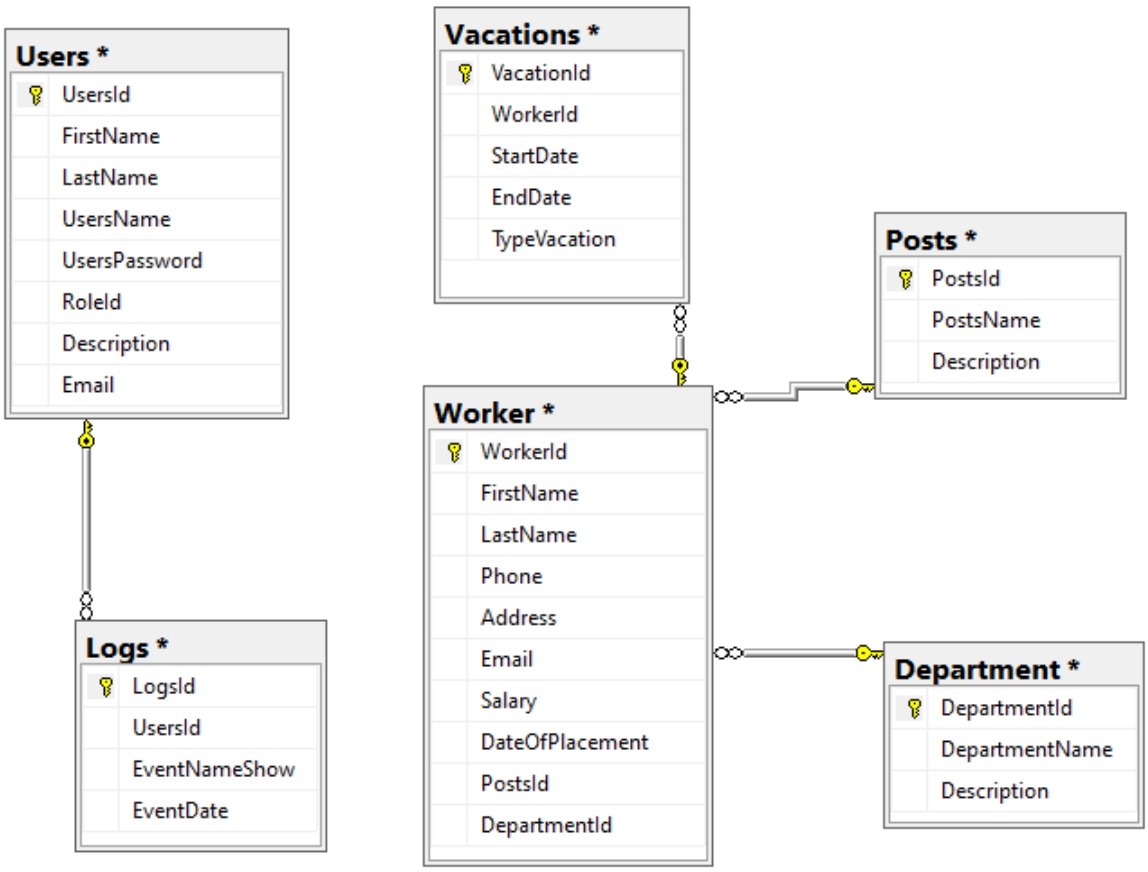


Рис. 2.6. – ERD діаграма бази даних

ER-діаграма забезпечує візуальне представлення структури бази даних, що значно полегшує розуміння взаємозв'язків між різними сутностями та їх атрибутами. Вона виступає ключовим інструментом на етапі проєктування, дозволяючи запобігти потенційним помилкам і забезпечити логічну цілісність даних у процесі розробки системи. Така діаграма є основою для ефективного моделювання складних інформаційних систем, що сприяє їх подальшій реалізації.

Одночасно з моделюванням бази даних важливим етапом є проєктування архітектури системи, що безпосередньо впливає на її ефективність, зручність використання та здатність до масштабування. Архітектурне рішення має вирішальне значення, оскільки саме воно визначає, як компоненти системи будуть взаємодіяти між собою і з користувачами, що в свою чергу впливає на загальну продуктивність і стабільність програмного забезпечення. Для цього проєкту було обрано трьохрівневу архітектуру, яка забезпечує розподіл функціональності на три основні рівні: доступ до даних, логіка бізнес-процесів

та інтерфейс користувача [33]. Кожен з цих рівнів виконує свої специфічні функції, забезпечуючи чітку організацію коду та полегшуючи процес розробки та тестування системи.

Зокрема, перший рівень – рівень доступу до даних – був розроблений на початковому етапі проєктування. Цей рівень відповідає за взаємодію з базою даних, здійснюючи операції зберігання, вибірки, оновлення та видалення інформації. Він є критично важливим для забезпечення цілісності та безпеки даних, оскільки гарантує, що всі запити до бази даних виконуються коректно і відповідно до прав доступу користувачів. Завдяки належній реалізації цього рівня система може ефективно працювати з великими обсягами даних, забезпечуючи їхню доступність, актуальність і точність. Ця архітектура також сприяє швидкому виконанню запитів, навіть за умов великої кількості одночасних операцій, що є ключовим для стабільної роботи системи.

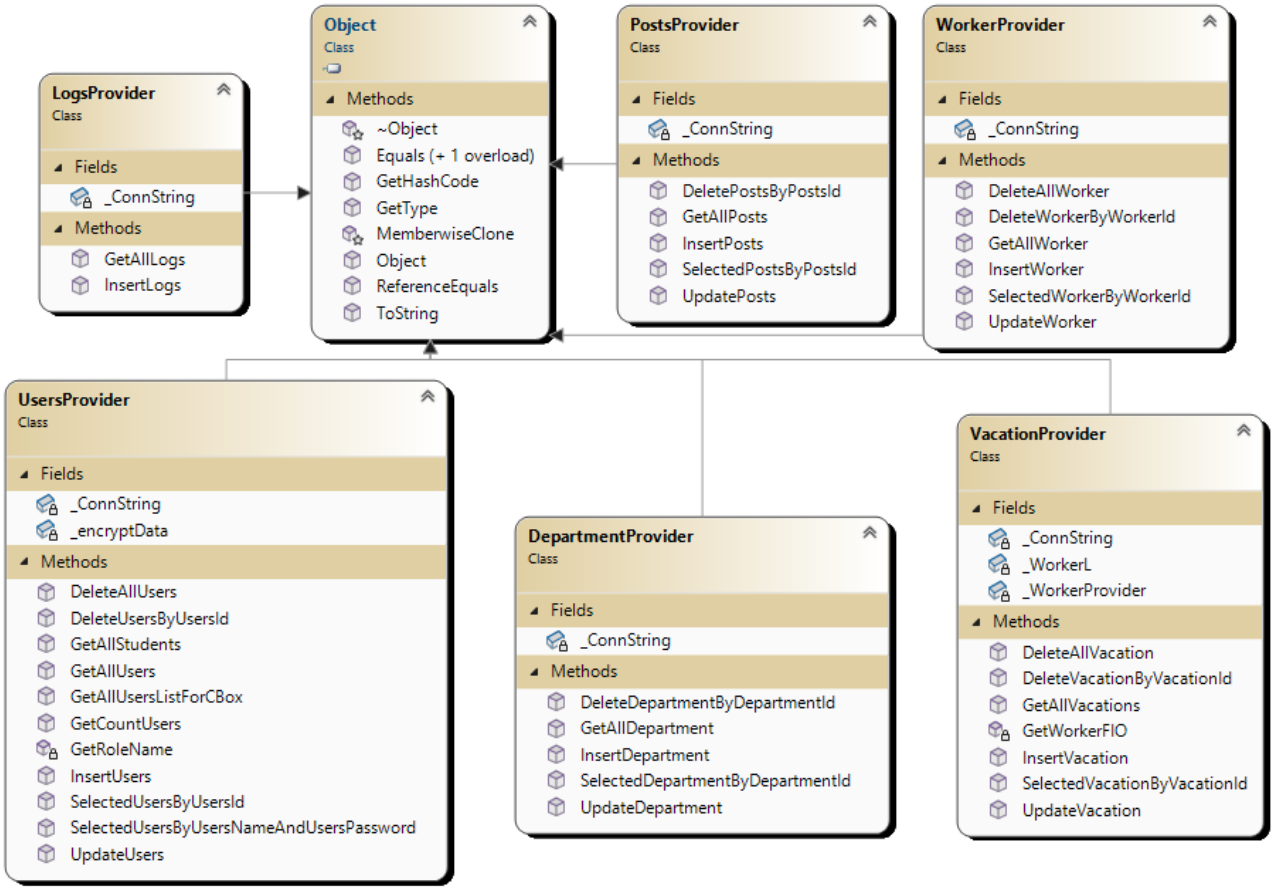


Рис. 2.7. – Діаграма класів рівня даних

У системі управління персоналом класи рівня доступу до даних, що реалізують операції CRUD (створення, читання, оновлення, видалення), відіграють ключову роль у взаємодії з базою даних та забезпеченні ефективного управління інформацією. Ці класи відповідають за коректне та безпечне маніпулювання даними різних сутностей системи. До основних класів цього рівня належать:

– DepartmentProvider –клас, який відповідає за управління даними про відділи підприємства. Він надає можливість додавати нові відділи, отримувати інформацію про існуючі, оновлювати дані щодо структури та характеристик відділів, а також видаляти відділи, що більше не актуальні для підприємства;

– LogsProvider – цей клас призначений для роботи з даними про події системи. За його допомогою здійснюється запис нових подій, отримання журналів дій для моніторингу та аналізу, оновлення інформації про події та видалення застарілих або некоректних записів з логів;

– PostsProvider – клас, що забезпечує управління інформацією про посади персоналу. Він дозволяє створювати нові посади, переглядати деталі існуючих, вносити зміни у відповідні характеристики посад та видаляти посади, які більше не використовуються в організації;

– UsersProvider – цей клас відповідає за маніпуляції з даними користувачів системи. Функціонал включає створення нових облікових записів, отримання інформації про користувачів, оновлення їхніх даних та видалення облікових записів при необхідності, забезпечуючи тим самим ефективне управління доступом до системи;

– VacationProvider – клас, призначений для роботи з даними про відпустки персоналу. Він надає можливість реєструвати нові відпустки, отримувати інформацію про заплановані та минулі відпустки працівників, вносити корективи до деталей відпусток та видаляти записи про відпустки у разі скасування або помилки;

– WorkerProvider – цей клас забезпечує управління даними про персонал підприємства. За його допомогою здійснюється додавання нових працівників до системи, отримання детальної інформації про співробітників, оновлення персональних та професійних даних, а також видалення записів про працівників, які більше не працюють в організації.

Використання цих класів забезпечує структурований та ефективний підхід до управління даними в системі, сприяючи підвищенню надійності та продуктивності програмного забезпечення.

Рівень користувацького інтерфейсу розробленої системи забезпечує зручну та інтуїтивно зрозумілу взаємодію користувачів із функціональними можливостями системи. Інтерфейс орієнтований на спрощення доступу до ключових функцій управління персоналом, забезпечуючи швидкий і ефективний спосіб виконання завдань, таких як управління даними про користувачів, відділи, посади, відпустки та персонал. На рис. 2.8 представлена діаграма класів цього рівня, яка відображає основні елементи інтерфейсу користувача та їх взаємозв'язки.



Рис. 2.8. – Діаграма класів рівня інтерфейсу користувача

Рівень користувацького інтерфейсу розробленої системи також включає класи, які забезпечують генерацію звітності та пошук інформації, що є важливими для аналітики та управління персоналом. Нижче описано відповідні класи, які реалізують функції інтерфейсу користувача:

– DepartmentRaportForm – клас, що відповідає за генерацію звітів по відділах. Він надає інтерфейс для формування звітів, які містять інформацію про структуру, склад та основні показники роботи відділів, що допомагає керівництву оцінювати ефективність кожного підрозділу;

– PostsRaportForm – клас, призначений для створення звітів по посадах. Він дозволяє генерувати детальні звіти про всі посади в організації, включаючи інформацію про зайнятість, вимоги до кваліфікації та інші характеристики, що дозволяє аналізувати розподіл посад та їхню відповідність потребам підприємства;

– SalaryRaportForm – клас, який забезпечує створення звітів по заробітній платі працівників. Цей інтерфейс надає можливість отримувати аналітичні дані про розподіл заробітної плати серед працівників, що дозволяє керівництву приймати зважені рішення щодо компенсаційної політики;

– VocationRaportForm – клас, призначений для формування звітів про працівників, які мають відпустку за обраний період часу. Він надає зручний інтерфейс для отримання інформації про заплановані або минулі відпустки, що дозволяє ефективно управляти графіками відпусток та забезпечувати безперервність робочих процесів;

– WorkersSearchForm – клас, який відповідає за пошук інформації про працівників за різними критеріями, такими як П.І.Б, телефон, посада, електронна адреса, відділ. Він надає інтерфейс для швидкого та зручного пошуку потрібних даних, що полегшує роботу з великою кількістю персоналу та сприяє швидкому доступу до необхідної інформації;

– LoginForm – клас, який відповідає за авторизацію користувачів у системі. Він забезпечує безпечний вхід до системи, запитуючи у користувача ім'я та пароль, та перевіряє їх на відповідність даним у базі;

– PersonalizationForm – клас, що забезпечує можливість персоналізації інтерфейсу користувача. Він дозволяє налаштовувати відображення інтерфейсу відповідно до індивідуальних потреб та уподобань користувача, зберігаючи ці налаштування для подальшого використання;

– SystemLogForm – клас, який відображає журнал подій системи. Цей інтерфейс дозволяє адміністраторам переглядати записи про всі значущі події, що відбувалися в системі, включаючи дії користувачів та системні помилки;

– UpdateUsersForm – клас, призначений для редагування та видалення облікових записів користувачів. Він забезпечує доступ до інформації про користувачів і дозволяє адміністраторам вносити зміни або видаляти облікові записи з системи;

– UsersForm – клас, який надає можливість додавання нових користувачів до системи та перегляду наявних. Він служить основним інтерфейсом для управління користувачами, надаючи зручний доступ до функцій додавання, перегляду та базової обробки облікових записів;

– PersonneMDI – головне вікно системи, яке об’єднує доступ до всіх основних функцій і модулів. Це центральний інтерфейс, з якого користувачі можуть переходити до інших форм і виконувати свої задачі;

– DepartmentForm – клас, що забезпечує інтерфейс для додавання та перегляду відділів підприємства. Він дозволяє користувачам створювати нові відділи, переглядати інформацію про наявні відділи, а також здійснювати базові операції управління;

– UpdateDepartmentForm – клас, який використовується для редагування та видалення відділів підприємства. Він надає інструменти для оновлення інформації про відділи та їхнього видалення у разі необхідності;

– PostsForm – клас, який надає інтерфейс для додавання та перегляду посад у системі. Він дозволяє створювати нові посади, переглядати наявні, а також здійснювати управління даними про посади;

– UpdatePostsForm – клас, призначений для редагування та видалення посад. Він забезпечує зручний інтерфейс для внесення змін до інформації про посади або видалення записів;

– VacationForm – клас, який забезпечує інтерфейс для додавання та перегляду відпусток персоналу. Користувачі можуть реєструвати нові відпустки та переглядати інформацію про вже заплановані або минулі відпустки;

– UpdateVacationForm – клас, призначений для редагування та видалення інформації про відпустки персоналу. Він дозволяє вносити корективи до даних відпусток або видаляти невірні або застарілі записи;

– WorkerForm – клас, що надає можливість додавання нових працівників до системи та перегляду наявних. Інтерфейс дозволяє керувати інформацією про персонал, включаючи додавання нових записів і перегляд детальної інформації;

– UpdateWorkerForm – клас, призначений для редагування та видалення даних про працівників. Він надає зручний інтерфейс для внесення змін до даних про працівників або видалення записів, які більше не актуальні.

Ці класи забезпечують аналітичні можливості системи, допомагаючи керівництву приймати обґрунтовані рішення на основі детальних звітів та точного пошуку інформації про працівників.

Рівень бізнес-логіки розробленої системи виконує ключову роль у забезпеченні коректного виконання бізнес-процесів, що стосуються управління персоналом. Цей рівень відповідає за обробку даних, реалізацію правил і логіки роботи системи, що забезпечує узгодженість та ефективність операцій. На рис. 2.9 представлена діаграма класів цього рівня, що відображає основні елементи бізнес-логіки системи.

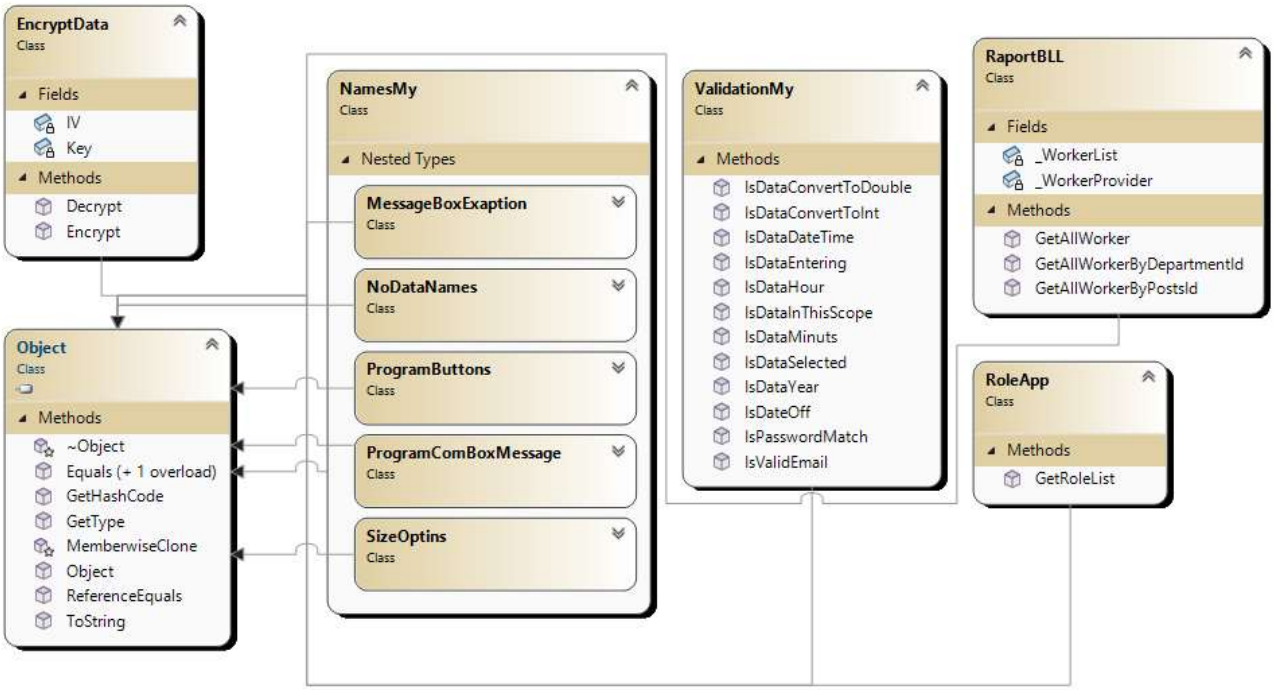


Рис. 2.9. – Діаграма класів рівня бізнес-логіки

Діаграма класів даного рівня складається із:

EncryptData – клас, відповідальний за шифрування та розшифрування даних у системі. Він забезпечує захист конфіденційної інформації, такої як паролі користувачів, через використання криптографічних алгоритмів, що гарантує безпеку зберігання і передачі даних;

NamesMy – клас, який надає допоміжні функції для роботи з іменами та рядками. Він включає методи для нормалізації, форматування та перевірки імен користувачів, працівників та інших сутностей, що допомагає підтримувати узгодженість даних у системі;

RoleApp – клас, що відповідає за управління ролями користувачів у системі. Він забезпечує реалізацію логіки розподілу прав доступу відповідно до ролей, що дозволяє контролювати доступ користувачів до різних функцій і даних системи;

ValidationMy – клас, призначений для перевірки правильності введених даних. Він включає різноманітні методи валідації, які гарантують відповідність даних встановленим правилам і вимогам системи, запобігаючи введенню некоректної інформації;

RaportBLL – клас, який реалізує бізнес-логіку, пов'язану з формуванням звітів. Він обробляє запити на створення різних видів звітності, забезпечуючи правильне збирання, обробку та представлення даних відповідно до вимог користувачів і бізнес-процесів.

Ці класи забезпечують надійну роботу бізнес-логіки системи, дозволяючи ефективно реалізовувати функціональні можливості, необхідні для управління персоналом.

3 РЕАЛІЗАЦІЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

3.1 Розроблення основних компонентів системи

У рамках мови уніфікованого моделювання (UML) діаграма компонентів відіграє важливу роль в ілюстрації архітектури системних рішень. Вона демонструє структуру програмного забезпечення, візуалізуючи зв'язки між окремими компонентами та показуючи, як елементи системи взаємодіють між собою. На цій діаграмі можна відобразити різні компоненти, такі як програмні модулі, бібліотеки, пакети чи навіть цілі системи, які працюють спільно для досягнення загальної мети. Діаграма компонентів допомагає представити складну систему у вигляді сукупності взаємопов'язаних частин, що взаємодіють через чітко визначені інтерфейси. Кожен компонент відповідає за певну частину функціональності системи та може бути розроблений, протестований і впроваджений незалежно від інших, що забезпечує гнучкість архітектури та високу ступінь повторного використання програмного коду.

На рис. 3.1 зображено розроблену діаграму компонентів для системи управління персоналом.

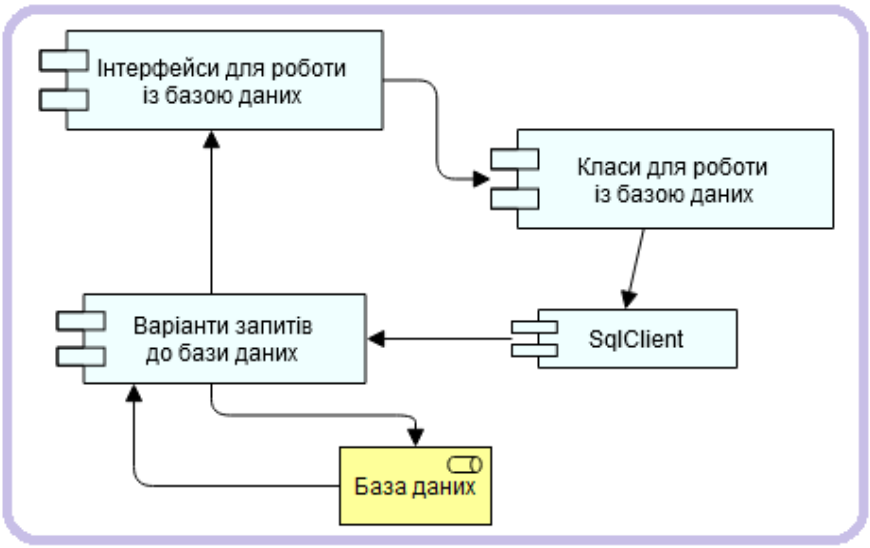


Рис. 3.1. – Діаграма компонентів системи

На представленій діаграмі компонентів виділено такі складові:

– інтерфейси для взаємодії з базою даних. Ці компоненти відповідають за створення абстракцій, що визначають правила доступу до бази даних. Вони

дозволяють іншим елементам системи здійснювати запити до бази, не зважаючи на її конкретну реалізацію.

– класи для роботи з базою даних. Це реалізації зазначених вище інтерфейсів, що забезпечують виконання низькорівневих операцій, таких як обробка запитів, маніпулювання даними та управління транзакціями.

– SqlClient. Цей модуль забезпечує пряме підключення до бази даних та виконання SQL-запитів. Його основними функціями є встановлення з'єднання, виконання запитів і отримання результатів.

– база даних. Фізичне сховище інформації, яке зберігає всі дані системи. Вона може бути реалізована різними способами, включаючи локальні або хмарні технології.

– набір запитів до бази даних. Це колекція SQL-запитів, що включає як прості команди, так і складні процедури та функції, що дозволяють здійснювати операції з даними, такі як читання, оновлення, вставка або видалення.

Стрілки, які ведуть від інтерфейсів і класів до компонента SqlClient, вказують на те, що вони використовують цей модуль для виконання операцій з базою даних. Стрілки від SqlClient до бази даних і набору запитів відображають процес ініціації запитів і взаємодії з базою для виконання необхідних операцій з даними.

Ця діаграма ілюструє модульний підхід до побудови системи, де кожен компонент виконує свою специфічну функцію, забезпечуючи ефективний і скоординований обмін даними.

3.2 Інтеграція бази даних з додатком

Інтеграція бази даних з програмним забезпеченням вимагає від розробників глибокого розуміння того, як взаємодіють компоненти, що відповідають за обмін та обробку інформації. Успішне використання баз даних у системах залежить від правильного налаштування з'єднань, команд та

інструментів для читання або адаптації даних, які формують основу взаємодії між базою даних і програмою. На рис. 3.2 представлена діаграма компонентів, яка демонструє взаємозв'язки між цими елементами в розроблюваній системі.

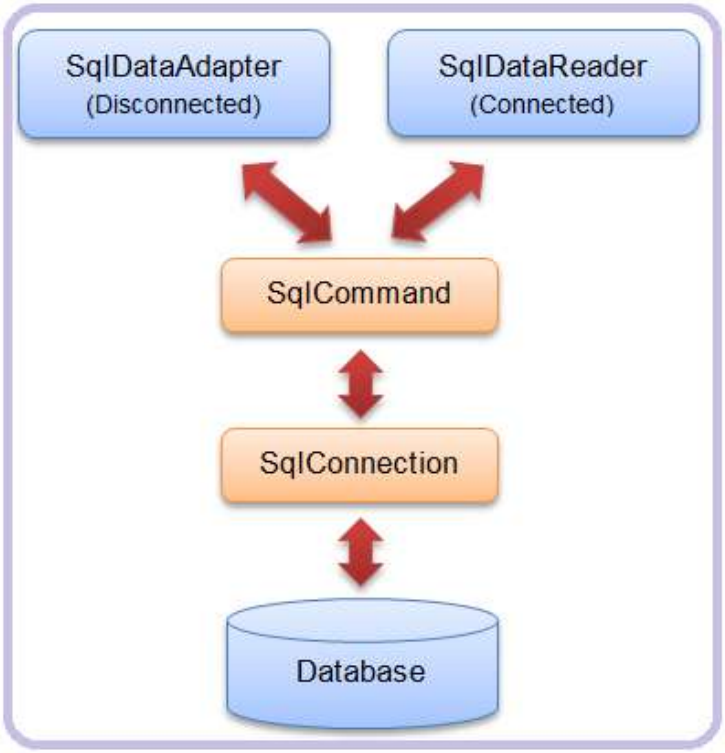


Рис. 3.2. – Схема підключення застосунку до бази даних

Основні компоненти діаграми включають:

– Database – це фізичне сховище, яке містить усі дані, необхідні для роботи додатка;

– SqlConnection – компонент, що встановлює з'єднання між додатком та базою даних, забезпечуючи можливість виконання запитів;

– SqlCommand – інструмент для виконання SQL-запитів через встановлене з'єднання, що дозволяє здійснювати операції з вибірки, вставки, оновлення та видалення даних;

– SqlDataAdapter (Disconnected) – компонент, призначений для роботи з даними у режимі відключеного доступу, дозволяючи маніпулювати даними офлайн та синхронізувати їх з базою даних пізніше;

– SqlDataReader (Connected) – інструмент для негайного доступу до даних з бази в режимі реального часу, що забезпечує ефективне читання інформації відразу після виконання запиту.

Взаємодія між компонентами, відображена на діаграмі, починається з процесу встановлення з'єднання за допомогою компонента SqlConnection. Цей компонент є основоположним для забезпечення зв'язку між додатком та базою даних, оскільки без активного з'єднання неможливо виконати жодні операції з інформацією, що зберігається в базі. Кожен раз, коли додатку необхідно взаємодіяти з базою даних, він спочатку повинен ініціювати з'єднання через SqlConnection. Це з'єднання створює шлях для передачі запитів до бази і отримання відповідей, забезпечуючи прямий канал між програмою та сховищем даних. Стрілка на діаграмі, яка веде від SqlConnection до бази даних, візуально демонструє цей процес встановлення з'єднання, що є першим кроком у будь-якій взаємодії з даними.

Після успішного встановлення з'єднання з базою даних, наступним кроком стає передача контролю до компонента SqlCommand, який відіграє ключову роль у взаємодії з даними. SqlCommand – це інструмент, що дозволяє додатку генерувати SQL-запити для виконання різних операцій з базою даних. Саме через SqlCommand здійснюються такі важливі дії, як вибірка даних (операція SELECT), внесення нових записів (операція INSERT), оновлення наявної інформації (операція UPDATE), а також видалення непотрібних або застарілих даних (операція DELETE). Усі ці операції передаються в базу даних через вже встановлене з'єднання, що гарантує їхнє виконання з високою ефективністю. Компонент SqlCommand забезпечує гнучкість у формуванні складних запитів, дозволяючи додатку точно вказувати, які дії повинні бути виконані з даними.

Для виконання операцій з даними в рамках системи використовуються два основних компоненти: SqlDataReader і SqlDataAdapter. Кожен із цих компонентів виконує свої специфічні функції у співпраці з SqlCommand, надаючи різні способи доступу та управління даними. SqlDataReader забезпечує

послідовний доступ до даних безпосередньо з бази, що означає, що для його роботи необхідне постійне активне з'єднання з базою даних. Цей компонент використовується в тих випадках, коли необхідно швидко отримати дані та відразу ж обробити їх, без збереження великих обсягів інформації в пам'яті додатку. SqlDataReader дозволяє ефективно працювати з великими обсягами даних, поступово зчитуючи їх з бази, не перевантажуючи систему.

З іншого боку, SqlDataAdapter працює в так званому режимі відключеного доступу (disconnected mode), що означає, що він може завантажувати дані з бази та зберігати їх у локальному середовищі, такому як DataSet або DataTable, без потреби в постійному підключенні до бази. Це особливо зручно у випадках, коли програма повинна мати доступ до даних у автономному режимі або коли взаємодія з базою є тимчасово неможливою. SqlDataAdapter також дозволяє синхронізувати зміни, внесені в локально збережені дані, з базою даних після відновлення з'єднання. Це забезпечує гнучкість у роботі з даними та дозволяє програмі продовжувати функціонувати навіть за відсутності прямого доступу до бази.

Вибір між використанням SqlDataReader або SqlDataAdapter залежить від специфічних вимог додатку. У тих випадках, коли потрібен швидкий і безперервний доступ до даних, зазвичай використовується SqlDataReader, після завершення роботи з яким з'єднання з базою даних закривається. Якщо ж додатку необхідно працювати з даними в режимі відключеного доступу або зберігати їх локально для подальшої обробки, тоді більш доцільним є використання SqlDataAdapter, який дозволяє працювати з даними незалежно від активності з'єднання з базою.

Зв'язок між SqlCommand та двома механізмами доступу до даних – SqlDataReader і SqlDataAdapter – показує, що всі команди, сформовані в SqlCommand, можуть бути виконані через будь-який із цих двох компонентів. Це надає додатку гнучкість у виборі методів роботи з даними, залежно від конкретних потреб і сценаріїв використання.

Для реалізації підключення до бази даних створено змінну «CONNECT» у конфігураційному файлі «App.config», яка містить рядок підключення з необхідними параметрами, детально описаними в документації проекту.

На рис. 3.3 представлено параметри підключення до бази даних системи управління персоналом.

```
<!-- Підключення до бази даних -->
<appSettings>
  <add key="CONNECT" value="Data Source=(LocalDB)\MSSQLLocalDB;
       AttachDbFilename=|DataDirectory|\DB.mdf;Integrated Security=True" />
</appSettings>
```

Рис. 3.3. – Параметри конфігурації змінної бази даних

Код міститься у секції <appSettings> конфігураційного файлу, зокрема:

– «Data Source=(LocalDB)\MSSQLLocalDB» вказує на те, що програма використовує локальну базу даних SQL Server (LocalDB).

– «AttachDbFilename=|DataDirectory|\DB.mdf» визначає шлях до файлу бази даних .mdf, що розташований у директорії програми.

– «Integrated Security=True» означає, що для підключення використовується інтегрована автентифікація Windows, і немає потреби у введенні окремого імені користувача та пароля.

Після встановлення з'єднання відкривається можливість інтеграції таблиць бази даних з формами програми, що забезпечує безперешкодний доступ до управління даними. У середовищі Visual Studio 2022 процес інтеграції бази даних значно спрощено, що сприяє ефективній розробці та легкості внесення змін у дані.

3.3 Розроблення модулів системи

Розроблення модулів системи управління персоналом є ключовим етапом, що визначає функціональність та ефективність роботи всієї системи. На цьому етапі важливо забезпечити не лише коректну роботу основних компонентів, але й можливість їхнього гнучкого налаштування та масштабування відповідно до потреб організації. Система управління персоналом повинна містити модулі для ведення обліку співробітників, їхньої діяльності, а також модулі для

автоматизації кадрових процесів, зокрема нарахування заробітної плати, відстеження відпусток та оцінки продуктивності.

Зі збільшенням кількості персональних даних, що зберігаються в таких системах, питання захисту інформації стає надзвичайно важливим. У сучасних системах управління персоналом безпека даних є одним із пріоритетних аспектів, особливо в умовах постійних кіберзагроз. Важливо не лише захистити доступ до даних через автентифікацію та авторизацію, але й застосувати методи шифрування для збереження конфіденційної інформації. Це забезпечує додатковий рівень захисту даних, навіть у випадку несанкціонованого доступу до бази даних або її зламу.

Одним із найбільш поширених підходів до захисту даних є використання шифрування. Метод шифрування дозволяє перетворювати текстові або інші дані у вигляд, який неможливо прочитати без спеціального ключа. У даній системі управління персоналом реалізовано метод Encrypt, який використовується для шифрування критичних даних, таких як паролі або інша конфіденційна інформація. Код методу Encrypt представлений на рис. 3.4.

```
13  public string Encrypt(string originalString) {
14      // Перевірка, чи вхідний рядок не є null або порожнім
15      if (String.IsNullOrEmpty(originalString)) {
16          throw new ArgumentNullException("The string which needs to be encrypted can not be null.");
17      }
18      // Використання AES шифрування з використанням CryptoServiceProvider
19      using (AesCryptoServiceProvider aes = new AesCryptoServiceProvider()) {
20          // Встановлення ключа та IV для шифрування
21          aes.Key = Key;
22          aes.IV = IV;
23          // Створення потоку в пам'яті для зберігання зашифрованих даних
24          MemoryStream memoryStream = new MemoryStream();
25          // Створення потоку шифрування
26          CryptoStream cryptoStream = new CryptoStream(memoryStream,
27              aes.CreateEncryptor(aes.Key, aes.IV), CryptoStreamMode.Write);
28          // Запис вхідного рядка в потік шифрування
29          StreamWriter writer = new StreamWriter(cryptoStream);
30          writer.Write(originalString);
31          writer.Flush();
32          // Завершення шифрування
33          cryptoStream.FlushFinalBlock();
34          writer.Flush();
35          // Повернення зашифрованих даних у форматі Base64
36          return Convert.ToBase64String(memoryStream.ToArray());
37      }
38  }
```

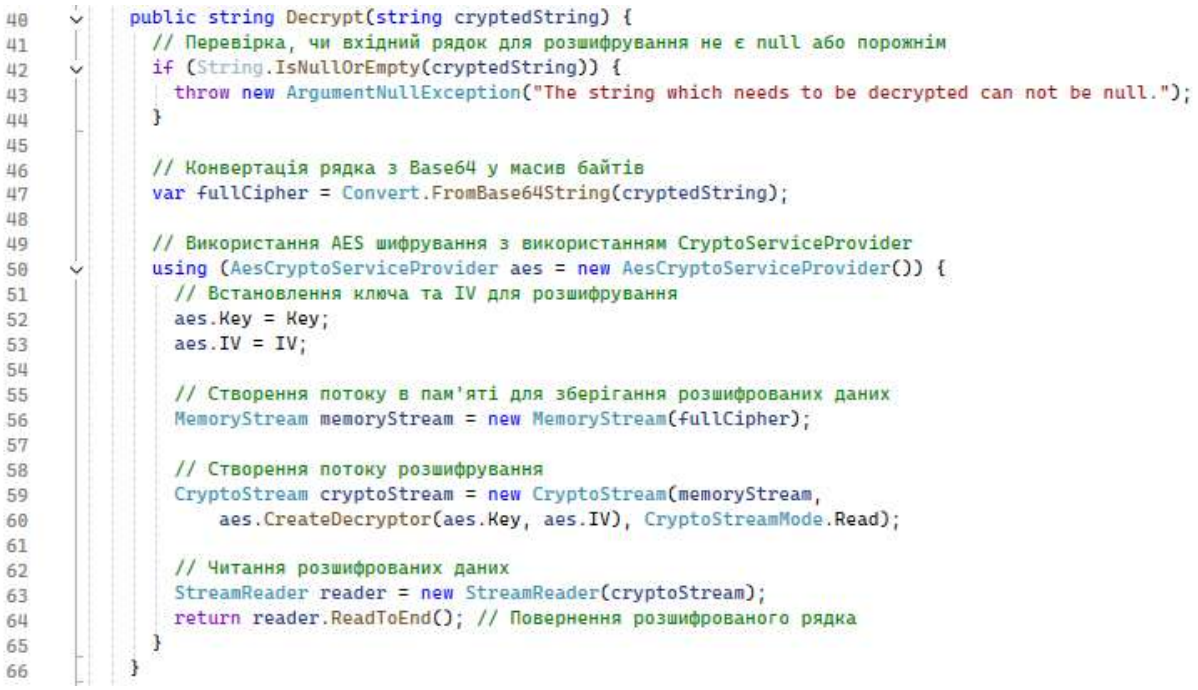
Рис. 3.4. – Реалізація методу для шифрування даних

Метод Encrypt призначений для шифрування текстових рядків, використовуючи алгоритм шифрування AES. Спочатку перевіряється, чи

вхідний рядок не є порожнім або null, щоб уникнути можливих помилок. У випадку, якщо рядок є порожнім, викликається виключення ArgumentNullException, що повідомляє про необхідність передачі значення для шифрування. Після цього використовується об'єкт AesCryptoServiceProvider, який реалізує алгоритм шифрування AES. Для шифрування даних метод задає два ключових параметри – ключ (Key) та ініціалізаційний вектор (IV), які забезпечують безпеку процесу. Далі створюється потік пам'яті (MemoryStream), який слугує сховищем для зашифрованих даних.

Для безпосереднього процесу шифрування створюється CryptoStream, який працює разом із потоком пам'яті та шифрувальним перетворювачем, створеним на основі ключа та IV. Через цей криптопотік записується початковий текстовий рядок, використовуючи StreamWriter, який передає текст для шифрування. Після запису тексту потік шифрування завершують, викликаючи методи FlushFinalBlock і Flush, що забезпечує остаточне формування зашифрованого блоку даних. Зашифровані дані, що містяться у пам'яті, перетворюються у формат Base64 для зручного зберігання або передачі. Наприкінці метод повертає цей Base64-рядок, що є зашифрованою версією початкового тексту.

Метод Decrypt призначений для розшифрування зашифрованого рядка, використовуючи алгоритм AES (рис. 3.5).



```
public string Decrypt(string cryptedString) {
    // Перевірка, чи вхідний рядок для розшифрування не є null або порожнім
    if (String.IsNullOrEmpty(cryptedString)) {
        throw new ArgumentNullException("The string which needs to be decrypted can not be null.");
    }

    // Конвертація рядка з Base64 у масив байтів
    var fullCipher = Convert.FromBase64String(cryptedString);

    // Використання AES шифрування з використанням CryptoServiceProvider
    using (AesCryptoServiceProvider aes = new AesCryptoServiceProvider()) {
        // Встановлення ключа та IV для розшифрування
        aes.Key = Key;
        aes.IV = IV;

        // Створення потоку в пам'яті для зберігання розшифрованих даних
        MemoryStream memoryStream = new MemoryStream(fullCipher);

        // Створення потоку розшифрування
        CryptoStream cryptoStream = new CryptoStream(memoryStream,
            aes.CreateDecryptor(aes.Key, aes.IV), CryptoStreamMode.Read);

        // Читання розшифрованих даних
        StreamReader reader = new StreamReader(cryptoStream);
        return reader.ReadToEnd(); // Повернення розшифрованого рядка
    }
}
```

Рис. 3.5. – Реалізація методу для дешифрування даних

На початку методу перевіряється, чи переданий рядок для розшифрування не є порожнім або null, і якщо він порожній, викликається виключення ArgumentNullException, яке повідомляє про необхідність вказати дійсний рядок для розшифрування. Після цього зашифрований рядок, що передається у форматі Base64, перетворюється у масив байтів за допомогою методу Convert.FromBase64String. Це необхідно для подальшої роботи з шифрованими даними. Далі використовується об'єкт AesCryptoServiceProvider, який реалізує алгоритм шифрування AES. Ключ (Key) та ініціалізаційний вектор (IV) задаються для розшифрування, і з їх допомогою створюється криптопотік.

Для зберігання даних використовується MemoryStream, куди поміщають байти зашифрованого рядка. Потім створюється CryptoStream, який відповідає за розшифрування даних. Він використовує ключ та ініціалізаційний вектор для декодування, після чого потік зчитується за допомогою StreamReader.

Фінальним кроком є читання всіх даних з криптопотоку, що повертає розшифрований текстовий рядок. Цей рядок, який представляє початкові дані до шифрування, повертається як результат роботи методу.

На рис. 3.6 зображено форму авторизації, яка є важливим елементом системи безпеки для доступу до персональних даних користувачів. У формі передбачено введення логіна та пароля, після чого користувач має можливість підтвердити свої дані для входу в систему. Крім того, реалізована опція реєстрації для нових користувачів.

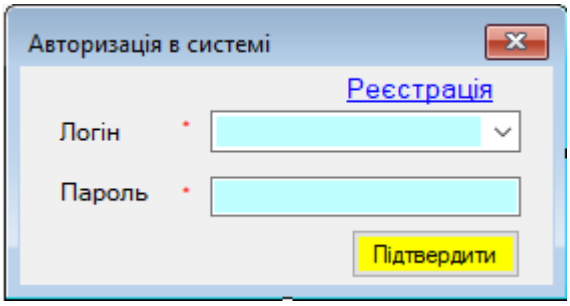


Рис. 3.6. – Форма авторизації користувачів

Для забезпечення безпеки при обробці введених даних, система використовує методи шифрування, зокрема метод Encrypt, який відповідає за шифрування паролів перед їх збереженням у базі даних. Це дозволяє захистити конфіденційну інформацію від несанкціонованого доступу. Для перевірки введених даних під час авторизації застосовується метод Decrypt, який дозволяє розшифрувати паролі для порівняння з тими, що зберігаються у зашифрованому вигляді. Такий підхід підвищує рівень безпеки всієї системи, забезпечуючи надійний захист особистої інформації користувачів.

На рис. 3.7 представлено метод GetSubmitData, який відповідає за обробку даних, які вводяться користувачем під час авторизації. Спочатку він перевіряє коректність введених даних за допомогою методу «IsDataEnteringCorrect». Якщо дані введені правильно, метод звертається до провайдера користувачів (_UserProvider) для отримання списку користувачів, які відповідають введеним логіну та паролю. Дані передаються як параметри методам «SelectedUsersByUsersNameAndUsersPassword», де порівнюються введені значення з тими, що зберігаються у базі.



```
66  private void GetSubmitData() {
67    try {
68      if (IsDataEnteringCorrect()) {
69        List<Users> listUsers = new List<Users>();
70        listUsers =
71          _UserProvider.SelectedUsersByUsersNameAndUsersPassword(UserNameCBox.Text, UserPasswordTbx.Text);
72        if (listUsers.Count > 0) {
73          CurrentUser = listUsers[0];
74          DataLoad();
75        } else {
76          MessageBox.Show(NamesMy.MessageBoxExaption.ThisUserLoginAndUserPasswordNotExistInSystem,
77            NamesMy.MessageBoxExaption.CaptionMessage);
78        }
79      }
80    } catch (Exception ex) {
81      MessageBox.Show(ex.Message);
82    }
83  }
```

Рис. 3.7. – Реалізація методу «GetSubmitData»

Якщо список користувачів не порожній і знайдено відповідний обліковий запис, програма присвоює змінній CurrentUser першого користувача із цього списку, тобто користувача, що пройшов успішну авторизацію. Після цього викликається метод DataLoad, який відповідає за завантаження необхідних даних для поточного користувача.

Якщо ж користувача з такими логіном і паролем не знайдено, програма виводить повідомлення про помилку за допомогою MessageBox.Show, інформуючи, що введені дані не існують у системі. У разі виникнення будь-якої виняткової ситуації в процесі виконання методу, помилка обробляється у блоці catch, і користувачу виводиться відповідне повідомлення з деталями про помилку.

Головна форма системи управління персоналом є ключовим інтерфейсом, який дозволяє користувачам отримати доступ до основних функцій системи, таких як управління персоналом, пошук, звітність та налаштування системи. Форма розроблена таким чином, щоб забезпечити інтуїтивно зрозумілу навігацію та зручний доступ до основних модулів системи. Користувачі можуть легко переходити між різними вкладками, виконуючи необхідні операції з управління даними про співробітників. Окрім функціональних можливостей, дизайн головної форми спрямований на підвищення зручності використання системи.

На рис. 3.8 представлена ця головна форма, яка візуально відображає основну концепцію системи управління персоналом.

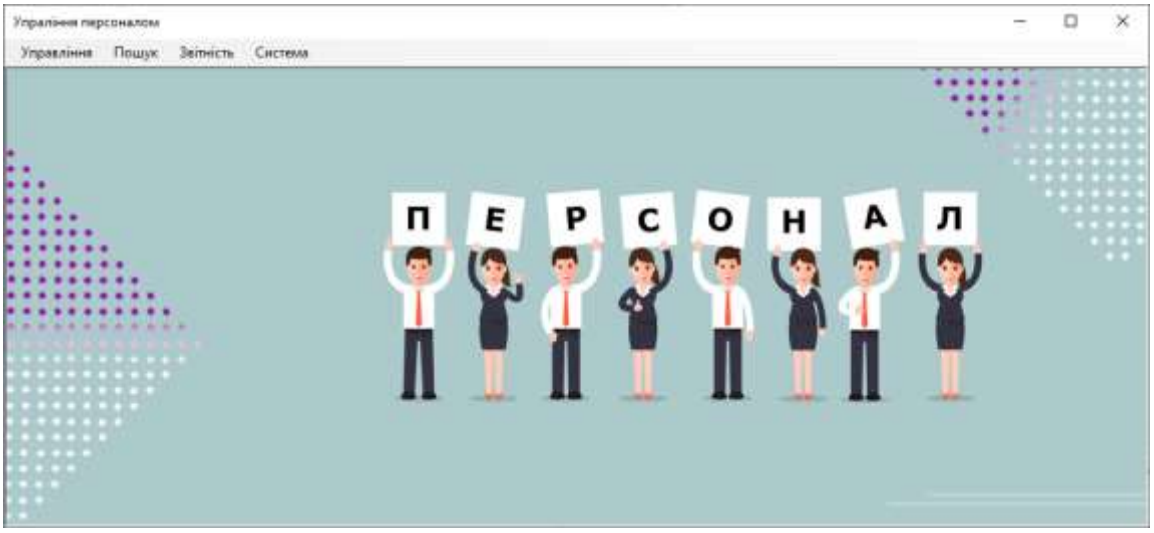


Рис. 3.8. – Реалізація головної форми

Форма для опрацювання даних персоналу дозволяє ефективно керувати інформацією про співробітників. Вона містить поля для введення ключових даних, таких як прізвище, ім'я, номер телефону, e-mail, заробітна плата, посада, відділ, дата працевлаштування та адреса. Інтуїтивний інтерфейс передбачає можливість додавання нових записів, очищення полів форми та виходу з системи. Така структура форми забезпечує зручний спосіб обробки персональних даних співробітників, що дозволяє користувачам швидко й ефективно управляти інформацією. Форма для опрацювання даних персоналу представлена на рис. 3.9.

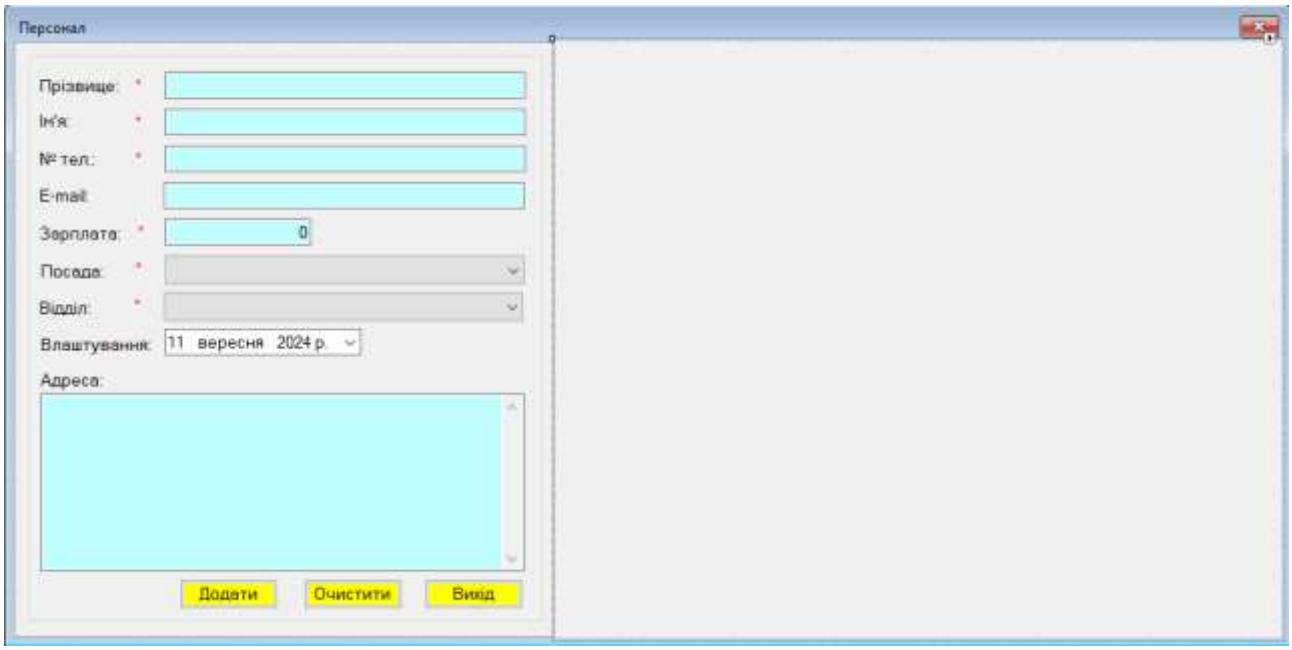


Рис. 3.9. – Форма для опрацювання даних персоналу

Метод LoadAllDate відповідає за завантаження даних для полів вибору, таких як посади та відділи, під час ініціалізації форми (рис. 3.10). Спочатку метод отримує список всіх посад за допомогою провайдера _PostsProvider, що викликає метод GetAllPosts. Отримані дані записуються у змінну _PostsList і прив'язуються до поля вибору PostsCBox. Поле ValueMember налаштовується таким чином, щоб зберігати значення ідентифікаторів посад (PostsId), а DisplayMember відповідає за відображення назв посад (PostsName).

```
30      private void LoadAllDate() {
31          _PostsList = _PostsProvider.GetAllPosts();
32          PostsCBox.DataSource = _PostsList;
33          PostsCBox.ValueMember = "PostsId";
34          PostsCBox.DisplayMember = "PostsName";
35
36          _DepartmentList = _DepartmentProvider.GetAllDepartment();
37          DepartmentCBox.DataSource = _DepartmentList;
38          DepartmentCBox.ValueMember = "DepartmentId";
39          DepartmentCBox.DisplayMember = "DepartmentName";
40      }
```

Рис. 3.10. – Форма для опрацювання даних персоналу

Аналогічно, для відділів використовується провайдер _DepartmentProvider, який викликає метод GetAllDepartment. Отриманий список відділів зберігається у змінну _DepartmentList і також прив'язується до відповідного поля вибору DepartmentCBox. Значенням для ValueMember є ідентифікатори відділів (DepartmentId), а для DisplayMember – назви відділів

(DepartmentName). Цей метод забезпечує автоматичне завантаження та заповнення полів форми необхідними даними під час її ініціалізації.

На рис. 3.11 представлено метод «AddBtn_Click», який виконується при натисканні кнопки «Додати» на формі.

```
42      private void AddBtn_Click(object sender, EventArgs e) {
43          if (IsDataEnteringCorrect()) {
44              _WorkerProvider.InsertWorker(LastNameTBox.Text, FirstNameTBox.Text,
45                  PhoneTBox.Text, AddressTBox.Text, EmailTBox.Text,
46                  Convert.ToDouble(SalaryTBox.Text), DateOfPlacementDTP.Value,
47                  Convert.ToInt32(PostsCBox.SelectedValue),
48                  Convert.ToInt32(DepartmentCBox.SelectedValue));
49              DataLoad();
50              ClearAllControls();
51          }
52      }
```

Рис. 3.11. – Форма для опрацювання даних персоналу

На початку у методі перевіряється коректність введених даних за допомогою методу «IsDataEnteringCorrect». Якщо дані правильні, метод звертається до провайдера _WorkerProvider, щоб додати нового працівника до системи. Для цього використовуються значення, введені в текстові поля форми, такі як прізвище, ім'я, номер телефону, адреса, e-mail, заробітна плата, дата працевлаштування, а також вибрані посада та відділ. Дані про зарплату перетворюються на тип Double, а значення для посади і відділу отримуються з відповідних полів вибору, після чого конвертуються у Int32.

Після успішного додавання працівника викликається метод «DataLoad» для оновлення даних у системі. Після цього всі поля форми очищуються за допомогою методу «ClearAllControls», щоб підготувати форму до введення нових даних.

Метод «WorkerGridView_CellClick» призначений для обробки події натискання на певний рядок у таблиці даних працівників (рис.3.12). Спочатку перевіряється, чи натиснутий рядок є коректним, тобто його індекс більше або дорівнює нулю, а значення першого стовпця цього рядка не відповідає повідомленню, яке вказує на відсутність даних. Якщо умови виконані, зберігається індекс обраного рядка в змінну _selectedRowIndex.

```
187     private void WorkerGridView_CellClick(object sender, DataGridViewCellEventArgs e) {
188       if (e.RowIndex >= 0 && WorkerGridView[0, e.RowIndex].Value.ToString() != _WorkerList[0].Message) {
189         _selectedRowIndex = e.RowIndex;
190         UpdateWorkerForm updateWorkerForm =
191           new UpdateWorkerForm(Convert.ToInt32(WorkerGridView[0, e.RowIndex].Value.ToString()));
192         updateWorkerForm.ShowDialog();
193         DataLoad();
194       }
195     }
```

Рис. 3.12. – Форма для редагування даних персоналу

Після цього створюється новий екземпляр форми «UpdateWorkerForm», куди передається ідентифікатор працівника, отриманий з першого стовпця обраного рядка (його значення перетворюється на Int32). Ця форма відкривається за допомогою методу «ShowDialog», що дозволяє редагувати дані обраного працівника. Після завершення роботи форми викликається метод «DataLoad», який оновлює дані в таблиці працівників для відображення змін.

Форма редагування даних персоналу, представлена на рис. 3.13, дозволяє користувачу вносити зміни до вже існуючих записів про працівників.

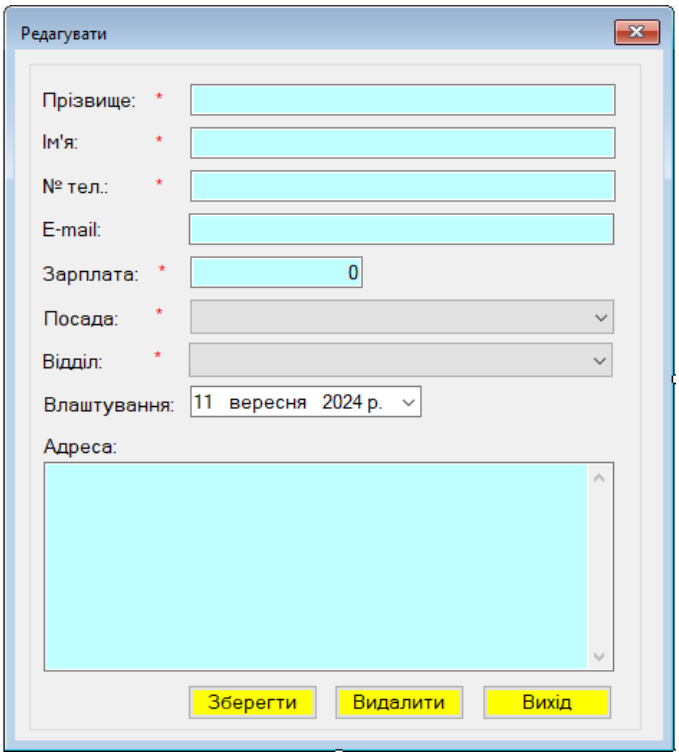


Рис. 3.13. – Форма для редагування даних персоналу

Дана форма також містить ті самі поля для введення інформації, що й при додаванні нових працівників, але додатково пропонує можливість збереження змін або видалення запису. Окрім цього, форма надає можливість корегувати

такі важливі параметри, як посада, відділ, заробітна плата, а також контактні дані.

На рис. 3.14 представлено метод SaveBtn_Click виконується при натисканні кнопки «Зберегти» на формі редагування даних працівника.

```
30      private void SaveBtn_Click(object sender, EventArgs e) {
31          if (IsDataEnteringCorrect()) {
32              _WorkerProvider.UpdateWorker(LastNameTBox.Text, FirstNameTBox.Text,
33                  PhoneTBox.Text, AddressTBox.Text, EmailTBox.Text,
34                  Convert.ToDouble(SalaryTBox.Text), DateOfPlacementDTP.Value,
35                  Convert.ToInt32(PostsCBox.SelectedValue),
36                  Convert.ToInt32(DepartmentCBox.SelectedValue), _WorkerId);
37              this.Close();
38          }
39      }
```

Рис. 3.14. – Збереження відредагованих даних працівника

На початку перевіряється правильність введених даних за допомогою методу «IsDataEnteringCorrect». Якщо введені дані є коректними, метод звертається до провайдера _WorkerProvider для оновлення інформації про працівника. Параметри, що передаються у метод «UpdateWorker», включають прізвище, ім'я, номер телефону, адресу, електронну пошту, заробітну плату, дату працевлаштування, а також значення для полів вибору посади і відділу. Крім цього, передається ідентифікатор працівника (_WorkerId), щоб визначити, який саме запис необхідно оновити. Після успішного оновлення даних форма закривається викликом методу «this.Close».

Також, реалізовано метод «DeleteBtn_Click» виконується при натисканні кнопки «Видалити» на формі редагування даних працівника.

```
30      private void SaveBtn_Click(object sender, EventArgs e) {
31          if (IsDataEnteringCorrect()) {
32              _WorkerProvider.UpdateWorker(LastNameTBox.Text, FirstNameTBox.Text,
33                  PhoneTBox.Text, AddressTBox.Text, EmailTBox.Text,
34                  Convert.ToDouble(SalaryTBox.Text), DateOfPlacementDTP.Value,
35                  Convert.ToInt32(PostsCBox.SelectedValue),
36                  Convert.ToInt32(DepartmentCBox.SelectedValue), _WorkerId);
37              this.Close();
38          }
39      }
```

Рис. 3.15. – Видалення даних працівника

Спочатку метод виводить користувачу підтверджуюче повідомлення за допомогою «MessageBox.Show», в якому запитується, чи дійсно він хоче

видалити цей запис. Якщо користувач підтверджує своє рішення, натиснувши «Yes», викликається метод «DeleteWorkerByWorkerId» з провайдера _WorkerProvider, який видаляє працівника з бази даних за допомогою переданого ідентифікатора працівника (_WorkerId). Після цього форма закривається викликом методу «this.Close».

Також реалізовано декілька методів для формування звітності системи управління персоналом. Наприклад на рис. 3.16 представлено метод для формування зарплатної відомості.

```
22  public void GetSalaryRaport() {
23    int num = 0;
24    double allPrice = 0;
25    _WorkerList = _RaportBLL.GetAllWorker();
26    if (_WorkerList.Count > 0) {
27      RaportTBox.Text = "Звіт по заробітній платі :\r\n";
28      RaportTBox.Text += "-------------------------------------------------------------------------------\r\n";
29
30      RaportTBox.Text += String.Format("{0,3}|{1, -30}|{2, -30}|{3, 12}|\r\n", "№", "П.І.Б.", "№ телефону", "Зарплата");
31      for (int i = 0; i < _WorkerList.Count(); i++) {
32        num++;
33        string raportString = String.Format("{0,3}|{1, -30}|{2, -30}|{3, 12}|\r\n",
34        num,
35        _WorkerList[i].FIO,
36        _WorkerList[i].Phone,
37        _WorkerList[i].Salary);
38        RaportTBox.Text += raportString;
39        allPrice += _WorkerList[i].Salary;
40      }
41      RaportTBox.Text += "-------------------------------------------------------------------------------\r\n";
42      RaportTBox.Text += String.Format("{0,65} {1, 12}\r\n", "Загальна сума: ", allPrice);
43
44    } else {
45      RaportTBox.Text = "Даних не знайдено!";
46    }
47  }
```

Рис. 3.16. – Формування звітності по заробітній платі

Метод «GetSalaryRaport» призначений для створення та відображення звіту по заробітній платі працівників. Спочатку він отримує список всіх працівників за допомогою методу GetAllWorker з класу _RaportBLL і зберігає його у змінну _WorkerList. Якщо список працівників містить записи, вміст текстового поля RaportTBox починає заповнюватися заголовком звіту, який включає назву «Звіт по заробітній платі» та лінію для розділення інформації. Далі додається таблиця, яка містить три колонки: номер працівника, П.І.Б., номер телефону та заробітна плата. У циклі метод проходить по всьому списку працівників, формує рядок з інформацією про кожного працівника і додає цей рядок до текстового поля. Водночас збільшується лічильник працівників та накопичується загальна сума заробітної плати в змінній allPrice.

Після завершення циклу метод додає ще одну лінію розділення та підсумовує загальну суму заробітної плати, яку також відображає в текстовому полі. Якщо ж у списку працівників немає даних, в текстове поле виводиться повідомлення «Даних не знайдено!».

Метод «GetWorkersOnVacation» у свою чергу призначений для отримання списку працівників, які перебувають у відпустці в заданий період часу (рис. 3.17). Метод приймає два параметри – початкову дату startDate та кінцеву дату endDate, і на їх основі будує запит до бази даних. Запит вибирає прізвище, ім'я, номер телефону, посаду та відділ працівників, які перебувають у відпустці, відповідно до заданих дат.

```
23  public List<WorkerInfo> GetWorkersOnVacation(DateTime startDate, DateTime endDate) {
24    var workersOnVacation = new List<WorkerInfo>();
25    string query = @"
26      SELECT W.LastName, W.FirstName, W.Phone, P.PostsName, D.DepartmentName
27      FROM Worker W
28      JOIN Vacations V ON W.WorkerId = V.WorkerId
29      JOIN Posts P ON W.PostsId = P.PostsId
30      JOIN Department D ON W.DepartmentId = D.DepartmentId
31      WHERE V.StartDate <= @EndDate AND V.EndDate >= @StartDate";
32    using (SqlConnection conn = new SqlConnection(_ConnString)) {
33      using (SqlCommand cmd = new SqlCommand(query, conn)) {
34        cmd.Parameters.AddWithValue("@StartDate", startDate);
35        cmd.Parameters.AddWithValue("@EndDate", endDate);
36        conn.Open();
37        using (SqlDataReader reader = cmd.ExecuteReader()) {
38          while (reader.Read()) {
39            var worker = new WorkerInfo {
40              LastName = reader["LastName"].ToString(),
41              FirstName = reader["FirstName"].ToString(),
42              Phone = reader["Phone"].ToString(),
43              PostsName = reader["PostsName"].ToString(),
44              DepartmentName = reader["DepartmentName"].ToString()
45            };
46            workersOnVacation.Add(worker);
47          }
48        }
49      }
50    }
51    return workersOnVacation;
52  }
```

Рис. 3.17. – Формування звітності по відпустці працівників

Для виконання запиту метод використовує об'єкт SqlConnection, який ініціалізується з використанням рядка підключення _ConnString. Після цього створюється SqlCommand, де параметри запиту встановлюються через cmd.Parameters.AddWithValue. Це забезпечує захист від SQL-ін'єкцій і

динамічну передачу дат у запит. Після відкриття з'єднання з базою даних виконується запит через ExecuteReader(), який повертає дані у вигляді SqlDataReader. Метод проходить через кожен запис, отриманий з бази, і створює об'єкти WorkerInfo, що містять інформацію про прізвище, ім'я, номер телефону, посаду та відділ працівників. Ці об'єкти додаються до списку workersOnVacation.Після завершення роботи з даними з'єднання з базою даних закривається, і метод повертає список працівників, які перебувають у відпустці протягом вказаного періоду.

3.4 Тестування розробленої системи та аналіз отриманих результатів

Під час тестування системи управління персоналом, ключовим аспектом було забезпечення надійної та безпечної авторизації користувачів. Особлива увага приділялася перевірці коректного функціонування форми авторизації, оскільки саме вона є початковим етапом роботи користувача з системою. Для цього було проведено кілька тестів, які охоплювали різні сценарії взаємодії з формою: від введення правильних даних до перевірки обробки помилкових або порожніх значень. Результати кожного тесту порівнювалися з очікуваними для підтвердження відповідності роботи форми функціональним вимогам.

У табл. 3.1 наведено тестові сценарії, якими було перевірено форму авторизації користувача. Кожен сценарій містить опис проведеного тесту, вхідні дані, очікуваний та фактично отриманий результат.

Таблиця 3.1 – Тестові сценарії форми авторизації

№	Опис тесту	Вхідні дані	Очікуваний результат	Отриманий результат
1	Авторизація з правильним логіном і паролем	Логін: dholden, Пароль: правильний	Користувач успішно входить в систему	Користувач успішно увійшов до системи
2	Авторизація з неправильним паролем	Логін: dholden, Пароль: неправильний	Виведено повідомлення про невірний пароль	Виведено повідомлення про невірний пароль
3	Авторизація з	Логін: (порожнє),	Виведено	Виведено

	порожніми полями	Пароль: (порожнє)	повідомлення про необхідність заповнення полів	повідомлення про необхідність заповнення полів
4	Перевірка роботи посилання «Реєстрація»	Клік по «Реєстрація»	Відкривається форма реєстрації нового користувача	Відкрилася форма реєстрації нового користувача

Усі проведені тести показали позитивні результати, що підтверджує коректну роботу форми авторизації відповідно до функціональних вимог. Результат тестування форми представлено на рис. 3.18.

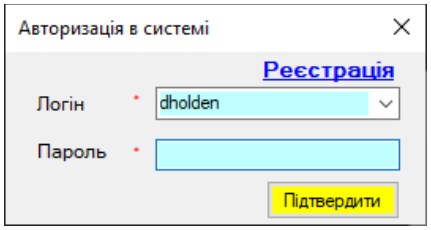


Рис. 3.18. – Тестування форми авторизації користувачів

У табл. 3.2 наведено тестові сценарії, якими було перевірено форму «Відділи». Кожен сценарій містить опис проведеного тесту, вхідні дані, очікуваний та фактично отриманий результат.

Таблиця 3.2 – Тестові сценарії форми «Відділи»

№	Опис тесту	Вхідні дані	Очікуваний результат	Отриманий результат
1	Додавання нового відділу	Назва: Відділ продажів, Опис: Управління продажами	Новий відділ успішно додається до списку відділів	Новий відділ успішно доданий до списку відділів
2	Спроба додати відділ без назви	Назва: (порожнє), Опис: (будь-який)	Виведено повідомлення про необхідність заповнення поля назви	Виведено повідомлення про необхідність заповнення поля назви
3	Очищення полів форми після введення даних	Назва: Відділ маркетингу, Опис: Просування продуктів	Поля назви та опису очищуються після натискання кнопки «Очистити»	Поля успішно очищено після натискання кнопки «Очистити»

4	Перевірка відображення списку відділів	Дані у списку відділів завантажено раніше	Всі відділи відображаються у списку з правильними номерами та назвами	Всі відділи відображаються правильно у списку

Усі проведені тести показали позитивні результати, що підтверджує коректну роботу форми «Відділи» відповідно до функціональних вимог. Результат тестування форми представлено на рис. 3.18.

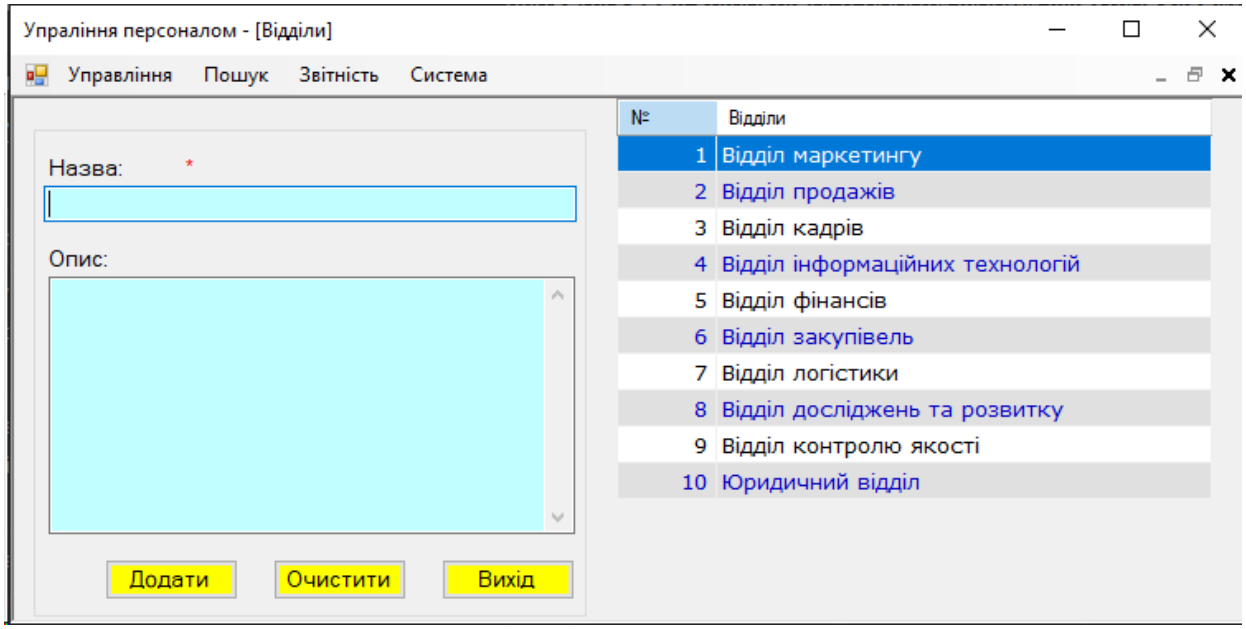


Рис. 3.19. – Тестування форми «Відділи»

У табл. 3.3 наведено тестові сценарії, якими було перевірено форму «Персонал». Кожен сценарій містить опис проведеного тесту, вхідні дані, очікуваний та фактично отриманий результат.

Таблиця 3.3 – Тестові сценарії форми «Персонал»

№	Опис тесту	Вхідні дані	Очікуваний результат	Отриманий результат
1	2	3	4	5
1	Додавання нового працівника	Прізвище: Петренко, Ім'я: Іван, Телефон: (067) 123-45-67, E-mail: test@example.com, Зарплата: 15000, Посада:	Новий працівник успішно додається до списку	Новий працівник успішно доданий до списку

		Бухгалтер, Відділ: Відділ продажів		
2	Спроба додати працівника без обов'язкових полів	Прізвище: (порожнє), Ім'я: (порожнє), Телефон: (порожнє)	Виведено повідомлення про необхідність заповнення обов'язкових полів	Виведено повідомлення про необхідність заповнення обов'язкових полів

Продовження таблиці 3.3

1	2	3	4	5
3	Очищення полів форми після введення даних	Прізвище: Сидоренко, Ім'я: Марія, Телефон: (050) 987-65-43	Поля форми успішно очищаються після натискання кнопки «Очистити»	Поля форми успішно очищені після натискання кнопки «Очистити»
4	Перевірка відображення списку працівників	Дані про працівників завантажено раніше	Усі працівники відображаються у таблиці з правильними даними	Усі працівники відображаються правильно у таблиці

Усі проведені тести показали позитивні результати, що підтверджує коректну роботу форми «Персонал» відповідно до функціональних вимог. Результат тестування форми представлено на рис. 3.19.

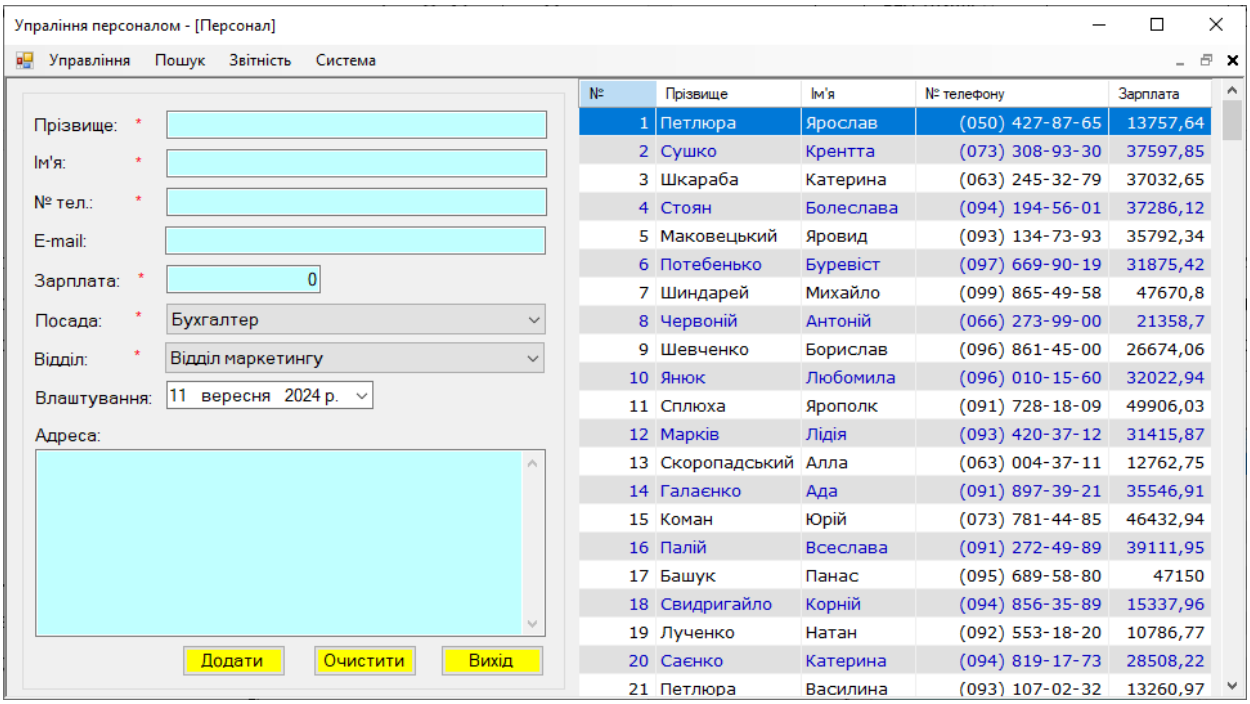


Рис. 3.19. – Тестування форми «Персонал»

Для перевірки коректної роботи форми «Відпустки персоналу» було проведено тестування, спрямоване на перевірку функціоналу додавання та відображення відпусток працівників. Тести охоплювали основні сценарії використання форми, включаючи введення даних про відпустки, очищення форми та правильне відображення інформації у таблиці.

Таблиця 3.4 – Тестові сценарії форми «Відпустки персоналу»

№	Опис тесту	Вхідні дані	Очікуваний результат	Отриманий результат
1	Додавання відпустки для працівника	Працівник: Петлюра Ярослав, Тип: щорічна, Початок: 11.09.2024, Кінець: 12.09.2024	Відпустка успішно додається до списку	Відпустка успішно додана до списку
2	Спроба додати відпустку без вибору працівника	Працівник: (не вибрано), Тип: щорічна, Початок: 11.09.2024, Кінець: 12.09.2024	Виведено повідомлення про необхідність вибору працівника	Виведено повідомлення про необхідність вибору працівника
3	Очищення полів форми після введення даних	Працівник: Ляшук Добромира, Тип: навчальна, Початок:	Поля форми успішно очищені після натискання	Поля успішно очищені після натискання кнопки

		01.09.2022, Кінець: 13.09.2022	кнопки «Очистити»	«Очистити»
4	Перевірка відображення списку відпусток	Дані про відпустки завантажено раніше	Усі відпустки відображаються у таблиці з правильними даними	Усі відпустки відображаються правильно у таблиці

Тестування форми «Відпустки персоналу» підтвердило, що всі функціональні можливості працюють відповідно до вимог. Система коректно обробляє додавання нових записів, перевіряє обов'язкові поля та відображає інформацію у таблиці без помилок. Результат тестування форми представлено на рис. 3.20.

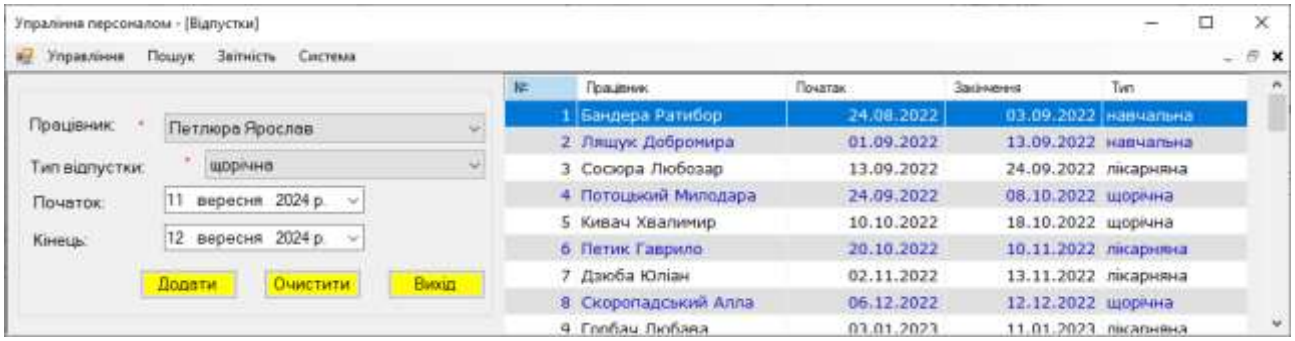


Рис. 3.20. – Тестування форми «Відпустки персоналу»

Для перевірки функціональності форми «Пошук інформації про працівників» було проведено тестування, яке охоплювало різні сценарії пошуку та фільтрації даних працівників за різними критеріями. Тести були спрямовані на перевірку коректності роботи пошукових полів і правильності відображення результатів у табл. 3.5.

Таблиця 3.5 – Тестові сценарії форми «Пошук інформації про працівників»

№	Опис тесту	Вхідні дані	Очікуваний результат	Отриманий результат
1	Пошук працівників за прізвищем	П.І.Б.: «Ла»	У таблиці відображаються працівники з прізвищем, що містять «Ла»	У таблиці відображені працівники з прізвищем «Ла»

2	Пошук за номером телефону	№ тел.: (094) 514-56-01	У таблиці відображається працівник з номером телефону (094) 514-56-01	Відображено працівника з вказаним номером телефону
3	Фільтрація за посадою	Посада: Бухгалтер	У таблиці відображаються всі працівники з посадою Бухгалтер	Відображено працівників з посадою Бухгалтер
4	Пошук працівників по відділу	Відділ: Відділ маркетингу	У таблиці відображаються всі працівники, що належать до Відділу маркетингу	Відображено працівників Відділу маркетингу
5	Очищення полів після пошуку	Пошукові поля заповнені, кнопка «ОК» натиснута	Поля успішно очищені після натискання кнопки «ОК»	Поля очищено, відображено всі записи працівників

Тестування форми «Пошук інформації про працівників» показало, що система коректно виконує пошук і фільтрацію даних за заданими критеріями. Усі тести завершились позитивно, що підтверджує правильну роботу пошукового функціоналу та можливість швидкого доступу до необхідної інформації про працівників. Результат тестування форми представлено на рис. 3.21.

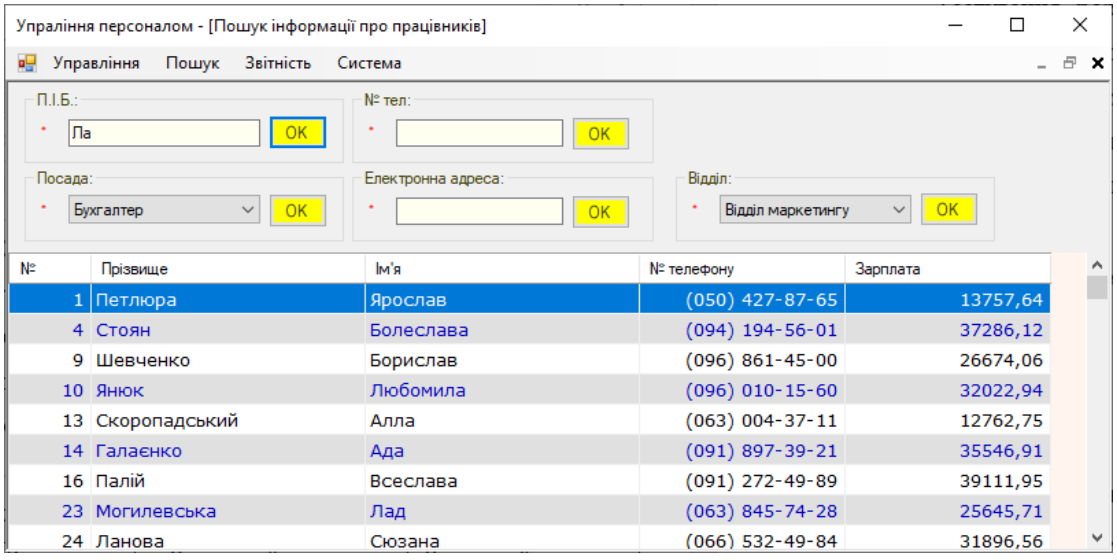


Рис. 3.21. Тестування форми «Пошук інформації про працівників»

Форма «Працівники, які мають відпустку» призначена для відображення списку працівників, що перебувають у відпустці за певний період. Тестування цієї форми було спрямоване на перевірку правильності відображення працівників, які відповідають обраним часовим рамкам, а також на коректність введення та обробки дат.

Таблиця 3.5 – Тестові сценарії форми «Працівники, які мають відпустку»

№	Опис тесту	Вхідні дані	Очікуваний результат	Отриманий результат
1	2	3	4	5
1	Пошук працівників, які мають відпустку	Початок: 01.03.2024, Кінець: 11.09.2024	У текстовому полі відображається список працівників, які мають відпустку в заданий період	Список працівників, які мають відпустку, успішно відображений
2	Пошук без обраного періоду	Початок: (порожнє), Кінець: (порожнє)	Виведено повідомлення про необхідність вибору періоду	Виведено повідомлення про необхідність вибору періоду

Продовження таблиці 3.5

1	2	3	4	5
3	Пошук при однаковій початковій і кінцевій датах	Початок: 01.03.2024, Кінець: 01.03.2024	Виводиться список працівників, які мають відпустку на цю дату	Виведено працівників, що мають відпустку на вказану дату
4	Очищення полів після виконання пошуку	Поля заповнені, натиснуто «ОК»	Поля успішно очищаються після натискання кнопки «ОК»	Поля успішно очищені після натискання кнопки «ОК»

Тестування форми «Працівники, які мають відпустку» підтвердило, що функціонал коректно обробляє пошукові запити відповідно до обраних дат.

Результати показали правильність відображення працівників у текстовому полі згідно з введеними критеріями. Усі тести завершилися позитивно. Результат тестування форми представлено на рис. 3.22.

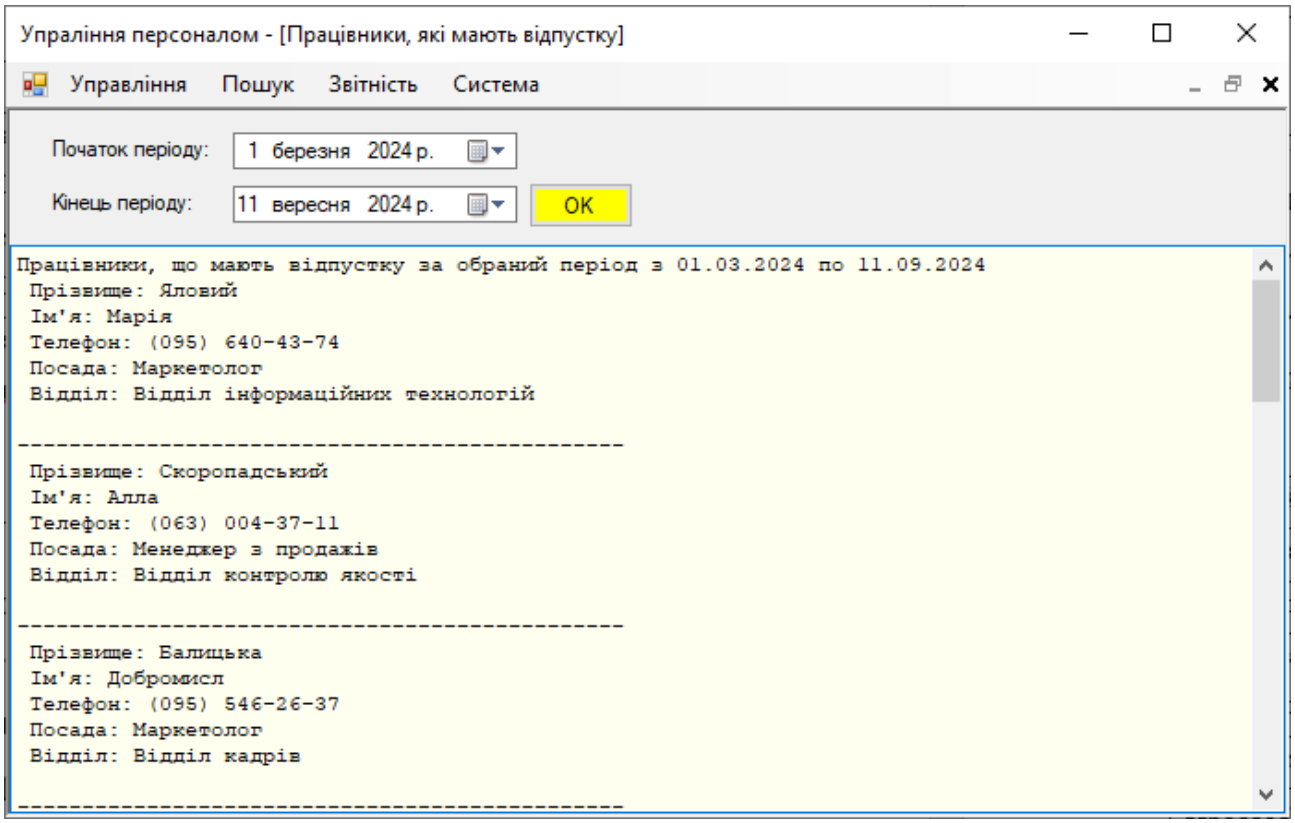


Рис. 3.22. Тестування форми «Працівники, які мають відпустку»

Після проведення тестування розробленої системи управління персоналом, наступним кроком став аналіз її ефективності та порівняння з провідними рішеннями на ринку. Для цього було обрано такі системи, як SAP SuccessFactors, Workday, та Zoho People. Порівняння дозволяє визначити, наскільки розроблене програмне забезпечення відповідає сучасним вимогам ринку та які його ключові переваги й можливі недоліки у порівнянні з іншими системами.

Критеріями для аналізу були обрані такі аспекти: функціональність, інтуїтивність інтерфейсу, гнучкість у налаштуваннях, інтеграція з іншими сервісами, вартість, та простота використання. Ці показники є критично важливими для ефективного управління персоналом у сучасних організаціях різного розміру та структури. Особливу увагу було приділено специфічним

функціям кожної системи, які можуть задовольнити потреби конкретних типів компаній, від невеликих підприємств до великих корпорацій.

SAP SuccessFactors – це одна з найпотужніших платформ для управління людськими ресурсами, яка орієнтована на великий бізнес та пропонує широкий спектр можливостей. Її основною перевагою є глибока інтеграція з іншими корпоративними системами, але вона є досить складною в налаштуванні та використанні. Workday також є популярним рішенням, яке вирізняється своєю гнучкістю та інноваційними інструментами для аналітики даних, але це програмне забезпечення є дорогим і більше орієнтоване на великі організації. Zoho People, у свою чергу, пропонує простий інтерфейс і більше підходить для середніх і малих підприємств, але має обмежену кількість інтеграцій з іншими бізнес-системами.

Для наочності порівняння розробленої системи з іншими рішеннями ринку у контексті її можливостей та переваг, було складено табл. 3.7, яка демонструє ключові характеристики кожного продукту. Це допоможе краще оцінити, як розроблене програмне забезпечення може задовольнити потреби користувачів і які напрямки для його подальшого удосконалення існують.

Таблиця 3.6 – Порівняльний аналіз розробленого ПЗ із аналогами

Характеристика	Розроблене ПЗ	BambooHR	Zoho People	Workday
Українізований інтерфейс	Так	Ні	Ні	Ні
Швидкість обробки запитів	Висока	Середня	Висока	Висока
Можливість роботи без Інтернету	Так	Ні	Ні	Ні
Інтеграція з іншими системами	Обмежена	Висока	Висока	Обмежена
Простота використання	Висока	Середня	Середня	Висока
Гнучкість налаштувань	Висока	Висока	Висока	Середня
Вартість	Доступна/Низька	Висока	Висока	Середня

Аналіз показав, що розроблене програмне забезпечення має низку переваг у порівнянні з аналогами. Наприклад, українізований інтерфейс робить його більш привабливим для локальних компаній, а можливість роботи без Інтернету дозволяє системі ефективно функціонувати навіть в умовах обмеженого доступу до мережі. Крім того, розроблене ПЗ відзначається високою швидкістю обробки запитів і простотою у використанні, що є важливими перевагами для малого та середнього бізнесу.

Разом з тим, деякі недоліки, такі як обмежена інтеграція з іншими системами, можуть бути важливими для великих компаній, які шукають більш комплексні рішення для автоматизації HR-процесів. Таким чином, подальше удосконалення розробленого програмного забезпечення може включати розширення можливостей інтеграції та збільшення гнучкості налаштувань, щоб конкурувати з провідними світовими рішеннями.

ВИСНОВКИ

Дана кваліфікаційна робота присвячена розробці системи управління персоналом, яка покликана вирішити ключові проблеми, що виникають у процесі кадрового менеджменту. Основною метою було створення інтуїтивно зрозумілого і функціонального інструменту для обліку працівників, управління їхньою діяльністю та автоматизації основних HR-процесів, зокрема ведення кадрової документації, планування трудових ресурсів, відбору та адаптації персоналу, а також формування звітності.

У першому розділі була проведена детальна оцінка проблематики управління персоналом. Аналіз показав, що існуючі системи часто мають недоліки в інтеграції з іншими бізнес-системами, є складними у використанні для малих і середніх підприємств або занадто дорогими для впровадження. На основі аналізу провідних рішень на ринку, таких як SAP SuccessFactors, Workday і Zoho People, було виявлено необхідність розробки нового рішення, яке б відповідало сучасним вимогам ринку, забезпечувало простоту використання, високу швидкість обробки даних та низьку вартість впровадження для компаній із обмеженими ресурсами.

Другий розділ висвітлює процес вибору технологій та моделювання системи. Для реалізації проекту було обрано мову програмування C# та середовище розробки Visual Studio 2022. Для роботи з базами даних було прийнято рішення використовувати MS SQL Server, враховуючи його надійність та широкі можливості для інтеграції з .NET-додатками. У розділі також були представлені діаграми бізнес-процесів для ролей адміністратора і користувача, діаграми послідовності та моделювання бази даних, що відображають основні етапи взаємодії користувача з системою.

Третій розділ був присвячений безпосередній реалізації системи та аналізу отриманих результатів. Було розроблено основні компоненти системи, зокрема модулі для шифрування конфіденційних даних, управління інформацією про персонал та формування звітів. Особлива увага приділялася

безпеці зберігання даних, для чого використовувалися методи шифрування паролів та захисту бази даних. У розділі також описано інтеграцію системи з базою даних, що дозволяє ефективно опрацьовувати запити та взаємодіяти з інформацією про працівників.

Проведена робота демонструє застосування сучасних технологій та підходів для автоматизації процесів управління персоналом. Порівняльний аналіз розробленої системи з провідними рішеннями, такими як SAP SuccessFactors, Workday та Zoho People, показав, що запропоноване програмне забезпечення має низку переваг, зокрема українізований інтерфейс, можливість роботи без доступу до Інтернету та високу швидкість обробки запитів. Це робить його привабливим вибором для малих і середніх підприємств, що потребують доступного та функціонального рішення для обліку та управління персоналом. Отримані результати свідчать про успішність реалізації поставлених задач і дозволяють стверджувати, що розроблене програмне забезпечення може бути ефективно використане для автоматизації HR-процесів у різних організаціях.

Розроблена система управління персоналом може бути ефективно використана в різних галузях, де є необхідність в автоматизації кадрових процесів та управлінні персоналом. Зокрема, вона підходить для підприємств малого та середнього бізнесу, освітніх закладів, державних установ, медичних центрів, компаній у сфері послуг, логістики та виробництва. Завдяки своїй гнучкості та функціональності, система може бути адаптована під потреби будь-якої організації, де необхідно вести облік працівників, організовувати трудові процеси, оцінювати продуктивність та формувати звіти по персоналу.

СПИСОК ЛІТЕРАТУРИ

1. Качан Є.П., Шушпанов Д.Г. Управління трудовими ресурсами: Навч. посібник. – К.: Видавничий дім “Юридична книга”, 2015. – 358 с.

2. Уманський О.М., Сумцов В.Г., Гордієнко В.Д. Соціально-трудові відносини: Навч. посібник. – Луганськ: вид-во СНУ ім. В.Даля, 2013. – 472 с.

3. Як пандемія COVID-19 змінила ринок праці в Україні. URL: https://www.ukrinform.ua/rubric-society/3104312-ak-pandemia-covid19-zminila-rinok-praci-v-ukraini.html (дата звернення: 20.08.2024).

4. Гаврилова, В. Планування трудових ресурсів на підприємстві [Текст] / В. Гаврилова; наук. кер. О.І. Карпіщенко // Економічні проблеми сталого розвитку: матеріали Міжнародної науково-практичної конференції імені проф. Балацького О.Ф., м. Суми, 11-12 травня 2016 р.: у 2-х т. / за заг. ред. О.В. Прокопенко. – Суми: СумДУ, 2016. – Т.2. – С. 159-161.

5. Білорус Т.В. Стратегічне управління кадровим потенціалом підприємства: Монографія. – Ірпінь: Національна академія ДПС України, 2007. – 174 с

6. Боков О. В., Вітряк Т. Б. Застосування соціологічних методів у системі прогнозування регіональних ринків праці: практичні аспекти. Ринок праці та зайнятість населення. 2016. №4. С. 33-38.

7. Ткаченко, А. М., & Марченко, К. А. Професійний розвиток персоналу–нагальне завдання сьогодення. Економічний вісник Донбасу. 2014. – С. 194-197.

8. Використання інноваційних методів управління персоналом підприємства. URL: https://chmnu.edu.ua/wp-content/uploads/2019/06/Ekonomika-i-suspilstvo-14-2018.pdf#page=448 (дата звернення: 20.08.2024).

9. Нестеренко О. М. Особливості правового виховання в Європі. Тенденції розвитку юридичної науки в інформаційному суспільстві: матер. міжнар. наук.-практ. конф. (Одеса, 28 груд. 2018 р.). Одеса: Гельветика, 2018. С. 14-17.

10. сова Н. В. Улюблена професія завжди дарує радість / Н. В. Усова // Виховання школярів. – 2011. – № 3. – С. 33-38.

11. Тураєв В. О. Державна система соціальних пільг в Україні та її недоліки, переваги адресного надання пільг / В. О. Тураєв, А. О. Курінна // Вісник Східноукраїнського національного університету імені Володимира Даля. — 2011. — № 17(171). — С. 78—84.

12. SuccessFactors Review (2024): Features, Pros & Cons. URL: https://www.forbes.com/advisor/business/software/successfactors-review/ (дата звернення: 20.08.2024).

13. Overview of SAP SuccessFactors 360 Reviews. URL: https://help.sap.com/docs/SAP_SUCCESSFACTORS_PERFORMANCE_AND_GOALS/f93a40604d3f4d36a5c45560e3e483d9/3e5733b8e6184fd7a3f496c97e7b6dde.html (дата звернення: 20.08.2024).

14. SAP SuccessFactors Review. URL: https://www.pcmag.com/reviews/sap-successfactors (дата звернення: 20.08.2024).

15. SAP SuccessFactors HCM Ratings Overview. URL: https://www.gartner.com/reviews/market/cloud-hcm-suites-for-1000-employees/vendor/sap/product/sap-successfactors-hcm (дата звернення: 20.08.2024).

16. Overview. What is SAP SuccessFactors HCM? URL: https://www.trustradius.com/products/sap-successfactors-hcm/reviews (дата звернення: 20.08.2024).

17. Likes and dislikes about SAP SuccessFactors HCM. URL: https://www.gartner.com/reviews/market/cloud-hcm-suites-for-1000-employees/vendor/sap/product/sap-successfactors-hcm/likes-dislikes (дата звернення: 20.08.2024).

18. Workday HCM Reviews. URL: https://www.softwareadvice.com/psa/workday-hcm-profile/reviews/ (дата звернення: 20.08.2024).

19. Workday Review 2024: Pricing, Pros & Cons. URL: https://www.forbes.com/advisor/business/software/workday-review/ (дата звернення: 20.08.2024).

20. Workday reviews. URL: https://www.glassdoor.com/Reviews/Workday-Reviews-E197851.htm (дата звернення: 20.08.2024).

21. Workday Suite Overview. URL: https://www.g2.com/products/workday-workday/reviews (дата звернення: 20.08.2024).

22. Workday Human Capital Management Ratings Overview. URL: https://www.gartner.com/reviews/market/cloud-hcm-suites-for-1000-employees/vendor/workday-hcm/product/workday-hcm (дата звернення: 20.08.2024).

23. Increase Value: Review and Recommend. URL: https://www.workday.com/content/dam/web/en-us/documents/datasheets/workday-review-recommend-datasheet.pdf (дата звернення: 20.08.2024).

24. Workday HCM Reviews. URL: https://www.getapp.com/hr-employee-management-software/a/workday-hcm/reviews/ (дата звернення: 20.08.2024).

25. Zoho People Ratings Overview. URL: https://www.gartner.com/reviews/market/workforce-management-applications/vendor/zoho/product/zoho-people (дата звернення: 20.08.2024).

26. Honest Zoho People Review 2024: Pros, Cons, Features & Pricing. URL: https://connecteam.com/reviews/zoho-people/ (дата звернення: 20.08.2024).

27. Zoho People Reviews & Product Details. URL: https://www.g2.com/products/zoho-people/reviews (дата звернення: 20.08.2024).

28. Continuous Review. URL: https://www.zoho.com/people/help/adminguide/performance-contreview.html (дата звернення: 20.08.2024).

29. Most Helpful Reviews for Zoho People. URL: https://www.softwareadvice.com/hr/zoho-people-profile/ (дата звернення: 20.08.2024).

30. Comparing Zoho People ratings with its top alternatives. URL: https://www.capterra.com/p/110931/Zoho-People/reviews/ (дата звернення: 20.08.2024).

31. Zoho People Integrations. URL: https://www.capterra.com/p/110931/Zoho-People/ (дата звернення: 20.08.2024).

32. Zoho People Key Features. URL: https://softwareconnect.com/reviews/zoho-people/ (дата звернення: 20.08.2024).

33. Городенко В.П., Марценюк О.П. Програмування на мові Python. Теорія та практика: Навчальний посібник. - К.: Центр навчальної літератури, 2016. - 480с.

34. Карапецький В. П. Побудова графічного контенту додатків з використанням JavaFX і Swing компонентів і даних, взятих із баз даних / В. П. Карапецький. – Науковий вісник НЛТУ. – 2015. – 418 с.

35. Троелсен Е. C# 7.0 та платформа .NET: Посібник для професіоналів. - Львів: Видавництво Львівської політехніки, 2018. - 918 с.

36. Скит Д., C# для професіоналів: тонкощі програмування, Вільямс. 2014. – 608 с.

37. Безменов М.І., Безменова О.М., Калінін Д.В. Основи візуального програмування мовою C#: навч. посіб. для студентів навчально-наукового інституту комп’ютерних наук та інформаційних технологій. – Харків: ФОП Панов А. М., 2023. 648 с.

38. Буй Д.Б., Сільвейструк Л.М. Модель «сутність-зв’язок»: формалізація сутностей та зв’язків // Вісник Київського університету. Серія: фіз.-мат. науки. – 2006. – 254 с.

39. Microsoft SQL Server : веб-сайт. URL: https://www.metabase.com/data_sources/microsoft-sql-server (дата звернення: 20.08.2024).

40. Козловська В. Організація баз даних та знань: конспект лекцій. Одеса, Одеський державний екологічний університет, 2019. – 130с.

41. Дубровський М. І., Зубков А. О. MySQL: навчальний посібник. Львів: Львівська політехніка, 2018. 250 с.

42. Голубчик В., Зайцев А. MySQL трюки: оптимізація, розробка, адміністрування. Київ: Видавництво "Вильямс", 2020. 500 с.

43. Oracle : веб-сайт. URL: https://www.oracle.com/cis/database/ (дата звернення: 20.08.2024).

44. Microsoft Visual Studio Reviews. URL: https://www.softwareadvice.com/game-development/microsoft-visual-studio-profile/reviews/ (дата звернення: 20.08.2024).

45. Visual Studio Code Reviews & Product Details. URL: https://www.g2.com/products/visual-studio-code/reviews (дата звернення: 20.08.2024).

46. Most Helpful Rider Reviews. URL: https://www.gartner.com/reviews/market/integrated-development-environment-ide-software/vendor/jetbrains/product/rider (дата звернення: 20.08.2024).

47. Томашевський О. М., Цегелик Г. Г., Вітер М. Б., Дубук В. І. Інформаційні технології та моделювання бізнес-процесів. Київ : «Центр учбової літератури», 2012. 296 с.

48. Кравець Р. Б., Сєров Ю. О., Марковець О. В. Інформаційні технології організації бізнесу : навчальний посібник. Львів : Львівська політехніка, 2013. 228 с.

49. Пушкар М.С. Ідеальна система обліку: концепція, архітектура, інформація: [монографія] / М.С. Пушкар, М.Г. Чумаченко. – Тернопіль: Карт-бланш, 2011. – 336 с.

50. Гладун А.В. Трьохрівнева архітектура побудови веб-систем. Вісник Національного університету "Львівська політехніка". 2018. – 315 с.

Додаток А

Лістинги програми

Лістинг 1. Код класу «VocationRaportForm»

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Runtime.InteropServices.ComTypes;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace PersonnelManagementApp.Forms.Raport {
  public partial class VocationRaportForm : Form {
    private string _ConnString = System.Configuration.ConfigurationSettings.AppSettings["CONNECT"];

    public VocationRaportForm() {
      InitializeComponent();
    }



    public List<WorkerInfo> GetWorkersOnVacation(DateTime startDate, DateTime endDate) {
      var workersOnVacation = new List<WorkerInfo>();
      string query = @"
        SELECT W.LastName, W.FirstName, W.Phone, P.PostsName, D.DepartmentName
        FROM Worker W
        JOIN Vacations V ON W.WorkerId = V.WorkerId
        JOIN Posts P ON W.PostsId = P.PostsId
        JOIN Department D ON W.DepartmentId = D.DepartmentId
        WHERE V.StartDate <= @EndDate AND V.EndDate >= @StartDate";
      using (SqlConnection conn = new SqlConnection(_ConnString)) {
        using (SqlCommand cmd = new SqlCommand(query, conn)) {
          cmd.Parameters.AddWithValue("@StartDate", startDate);
          cmd.Parameters.AddWithValue("@EndDate", endDate);
          conn.Open();
          using (SqlDataReader reader = cmd.ExecuteReader()) {
            while (reader.Read()) {
              var worker = new WorkerInfo {
                LastName = reader["LastName"].ToString(),
                FirstName = reader["FirstName"].ToString(),
                Phone = reader["Phone"].ToString(),
                PostsName = reader["PostsName"].ToString(),
                DepartmentName = reader["DepartmentName"].ToString()
              };
              workersOnVacation.Add(worker);
            }
          }
        }
      }
      return workersOnVacation;
    }

    private void SearchBtn_Click(object sender, EventArgs e) {
      if (IsDataEnteringCorrect()) {
        RaportTBox.Text = "Працівники, що мають відпустку за обраний період з " +
StartDataDTP.Value.ToShortDateString() + " по " + EndDataDTP.Value.ToShortDateString() + "\r\n";
        List < WorkerInfo > workers = new List<WorkerInfo>();
        workers = GetWorkersOnVacation(StartDataDTP.Value, EndDataDTP.Value);

        // Форматування і встановлення тексту у текстове поле
        StringBuilder sb = new StringBuilder();
        foreach (var worker in workers) {
          sb.AppendLine($" Прізвище: {worker.LastName}\r\n Ім'я: {worker.FirstName}\r\n Телефон:
{worker.Phone}\r\n Посада: {worker.PostsName}\r\n Відділ: {worker.DepartmentName}\r\n");
          sb.AppendLine("-----------------------------------------------");
        }
```

```
        RaportTBox.Text += sb.ToString();
      }
    }
    private bool IsDataEnteringCorrect() {
      bool isCorrect = true;
      if (EndDataDTP.Value <= StartDataDTP.Value) {
        isCorrect = false;
        MessageBox.Show("Початок періоду має бути меншим за кінець", "Увага!");
      }
      return isCorrect;
    }
  }
}
public class WorkerInfo {
  public string LastName { get; set; }
  public string FirstName { get; set; }
  public string Phone { get; set; }
  public string PostsName { get; set; }
  public string DepartmentName { get; set; }
}
```

Лістинг 2. Код класу «PostsRaportForm»

```
using PersonnelManagementApp.AppCode;
using PersonnelManagementApp.Providers;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace PersonnelManagementApp.Forms.Raport {
  public partial class PostsRaportForm : Form {
    private List<Worker> _WorkerList = new List<Worker>();
    private RaportBLL _RaportBLL = new RaportBLL();
    private PostsProvider _PostsProvider = new PostsProvider();
    private List<Posts> _PostsList = new List<Posts>();

    public PostsRaportForm() {
      InitializeComponent();
      LoadAllDate();
    }

    private void LoadAllDate() {
      _PostsList = _PostsProvider.GetAllPosts();
      PostsCBox.DataSource = _PostsList;
      PostsCBox.ValueMember = "PostsId";
      PostsCBox.DisplayMember = "PostsName";
    }

    private void PostsBtn_Click(object sender, EventArgs e) {
      _WorkerList = _RaportBLL.GetAllWorkerByPostsId(Convert.ToInt32(PostsCBox.SelectedValue));
      GetRaport(_WorkerList);
    }

    public void GetRaport(List<Worker> WorkerList) {
      int num = 0;
      double allPrice = 0;
      if (WorkerList.Count > 0) {
        RaportTBox.Text = "Звіт по заробітній платі за посадою :" + PostsCBox.Text + " \r\n";
        RaportTBox.Text += "--------------------------------------------------------------------------------\r\n";

        RaportTBox.Text += String.Format("{0,3}|{1, -30}|{2, -30}|{3, 12}|\r\n", "№", "П.І.Б.", "№ телефону",
"Зарплата");
        for (int i = 0; i < WorkerList.Count(); i++) {
          num++;
          string raportString = String.Format("{0,3}|{1, -30}|{2, -30}|{3, 12}|\r\n",
          num,
          WorkerList[i].FIO,
          WorkerList[i].Phone,
          WorkerList[i].Salary);
          RaportTBox.Text += raportString;
```

```
        allPrice += WorkerList[i].Salary;
       }
       RaportTBox.Text += "-------------------------------------------------------------------------------\r\n";
       RaportTBox.Text += String.Format("{0,65} {1, 12}\r\n", "Загальна сума: ", allPrice);

     } else {
       RaportTBox.Text = "Даних не знайдено!";
     }
   }

  }
}
```

Лістинг 3. Код класу «SalaryRaportForm»

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

namespace PersonnelManagementApp.Forms.Raport {
  public partial class SalaryRaportForm : Form {
    private List<Worker> _WorkerList = new List<Worker>();
    private RaportBLL _RaportBLL = new RaportBLL();

    public SalaryRaportForm() {
      InitializeComponent();
      GetSalaryRaport();
    }

    public void GetSalaryRaport() {
      int num = 0;
      double allPrice = 0;
      _WorkerList = _RaportBLL.GetAllWorker();
      if (_WorkerList.Count > 0) {
        RaportTBox.Text = "Звіт по заробітній платі :\r\n";
        RaportTBox.Text += "-------------------------------------------------------------------------------\r\n";

        RaportTBox.Text += String.Format("{0,3}|{1, -30}|{2, -30}|{3, 12}|\r\n", "№", "П.І.Б.", "№ телефону",
"Зарплата");
        for (int i = 0; i < _WorkerList.Count(); i++) {
          num++;
          string raportString = String.Format("{0,3}|{1, -30}|{2, -30}|{3, 12}|\r\n",
          num,
          _WorkerList[i].FIO,
          _WorkerList[i].Phone,
          _WorkerList[i].Salary);
          RaportTBox.Text += raportString;
          allPrice += _WorkerList[i].Salary;
        }
        RaportTBox.Text += "-------------------------------------------------------------------------------\r\n";
        RaportTBox.Text += String.Format("{0,65} {1, 12}\r\n", "Загальна сума: ", allPrice);

      } else {
        RaportTBox.Text = "Даних не знайдено!";
      }
    }

  }
}
```

Лістинг 4. Код класу «WorkersSearchForm»

```
using PersonnelManagementApp.AppCode;
using PersonnelManagementApp.Providers;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
```

```
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace PersonnelManagementApp.Forms.Seaerch {
  public partial class WorkersSearchForm : Form {
    private int _selectedRowIndex = 0;
    private ValidationMy _validation = new ValidationMy();
    private PostsProvider _PostsProvider = new PostsProvider();
    private List<Posts> _PostsList = new List<Posts>();
    private List<Worker> _SecectedWorkerList = new List<Worker>();
    private WorkerProvider _WorkerProvider = new WorkerProvider();
    private List<Worker> _WorkerList = new List<Worker>();
    private DepartmentProvider _DepartmentProvider = new DepartmentProvider();
    private List<Department> _DepartmentList = new List<Department>();

    public WorkersSearchForm() {
      InitializeComponent();
      LoadAllDate();
      DataLoad();
    }

    private void LoadAllDate() {
      _PostsList = _PostsProvider.GetAllPosts();
      PostsCBox.DataSource = _PostsList;
      PostsCBox.ValueMember = "PostsId";
      PostsCBox.DisplayMember = "PostsName";

      _DepartmentList = _DepartmentProvider.GetAllDepartment();
      DepartmentCBox.DataSource = _DepartmentList;
      DepartmentCBox.ValueMember = "DepartmentId";
      DepartmentCBox.DisplayMember = "DepartmentName";
    }

    private void DataLoad() {
      int firstRowIndex = 0;
      if (WorkerGridView.FirstDisplayedScrollingRowIndex > 0) {
        firstRowIndex = WorkerGridView.FirstDisplayedScrollingRowIndex;
      }
      try {
        _WorkerList = _WorkerProvider.GetAllWorker();
        LoadDataInWorkerGridView(_WorkerList);
        if (_selectedRowIndex == WorkerGridView.Rows.Count) {
          _selectedRowIndex = WorkerGridView.Rows.Count - 1;
        }
        if (_selectedRowIndex >= 0) {
          WorkerGridView.FirstDisplayedScrollingRowIndex = firstRowIndex;
          WorkerGridView.Rows[_selectedRowIndex].Selected = true;
        }
      } catch { }
    }
    private void LoadDataInWorkerGridView(List<Worker> WorkerList) {
      WorkerGridView.DataSource = null;
      WorkerGridView.Columns.Clear();
      WorkerGridView.AutoGenerateColumns = false;
      WorkerGridView.RowHeadersVisible = false;
      WorkerGridView.DataSource = WorkerList;
      if (WorkerList.Count > 0) {
        if (WorkerList[0].Message == NamesMy.NoDataNames.NoDataInWorker) {
          DataGridViewColumn messageColumn = new DataGridViewTextBoxColumn();
          messageColumn.DataPropertyName = "Message";
          messageColumn.Width = WorkerGridView.Width - NamesMy.SizeOptins.MinusSizePanel;
          WorkerGridView.Columns.Add(messageColumn);
        } else {
          DataGridViewColumn DetailIdColumn = new DataGridViewTextBoxColumn();
          DetailIdColumn.DataPropertyName = "WorkerId";
          WorkerGridView.Columns.Add(DetailIdColumn);
          WorkerGridView.Columns[0].Visible = false;

          DataGridViewColumn numberColumn = new DataGridViewTextBoxColumn();
          numberColumn.HeaderText = "№ ";
```

```
      numberColumn.DataPropertyName = "Number";
      numberColumn.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
      numberColumn.Width = NamesMy.SizeOptins.NumberSize;
      WorkerGridView.Columns.Add(numberColumn);

      DataGridViewColumn LastNameColumn = new DataGridViewTextBoxColumn();
      LastNameColumn.HeaderText = "Прізвище";
      LastNameColumn.DataPropertyName = "LastName";
      LastNameColumn.Width = 200;
      WorkerGridView.Columns.Add(LastNameColumn);

      DataGridViewColumn FirstNameColumn = new DataGridViewTextBoxColumn();
      FirstNameColumn.HeaderText = "Ім'я";
      FirstNameColumn.DataPropertyName = "FirstName";
      FirstNameColumn.Width = 200;
      WorkerGridView.Columns.Add(FirstNameColumn);

      DataGridViewColumn PhoneColumn = new DataGridViewTextBoxColumn();
      PhoneColumn.HeaderText = "№ телефону";
      PhoneColumn.DataPropertyName = "Phone";
      PhoneColumn.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
      PhoneColumn.Width = 150;
      WorkerGridView.Columns.Add(PhoneColumn);

      DataGridViewColumn SalaryColumn = new DataGridViewTextBoxColumn();
      SalaryColumn.HeaderText = "Зарплата";
      SalaryColumn.DataPropertyName = "Salary";
      SalaryColumn.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
      SalaryColumn.Width = 150;
      WorkerGridView.Columns.Add(SalaryColumn);
    }
    for (int i = 0; i < WorkerGridView.Columns.Count; i++) {
      WorkerGridView.Columns[i].HeaderCell.Style.BackColor = Color.LightGray;
    }
  }
}

private void FIOBtn_Click(object sender, EventArgs e) {
  _SecectedWorkerList.Clear();
  if (_validation.IsDataEntering(FIOTBox.Text)) {
    for (int i = 0; i < _WorkerList.Count(); i++) {
      if (_WorkerList[i].FIO.ToLower().Contains(FIOTBox.Text.ToLower())) {
        _SecectedWorkerList.Add(_WorkerList[i]);
      }
      FIOValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
    }
    LoadDataInWorkerGridView(_SecectedWorkerList);
  } else {
    FIOValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
  }
}

private void PhoneBtn_Click(object sender, EventArgs e) {
  _SecectedWorkerList.Clear();
  if (_validation.IsDataEntering(PhoneTBox.Text)) {
    for (int i = 0; i < _WorkerList.Count(); i++) {
      if (_WorkerList[i].Phone.ToLower().Contains(PhoneTBox.Text.ToLower())) {
        _SecectedWorkerList.Add(_WorkerList[i]);
      }
      PhoneValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
    }
    LoadDataInWorkerGridView(_SecectedWorkerList);
  } else {
    PhoneValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
  }
}

private void PostsBtn_Click(object sender, EventArgs e) {
  _SecectedWorkerList.Clear();
  if (Convert.ToInt32(PostsCBox.SelectedValue) > 0) {
    for (int i = 0; i < _WorkerList.Count(); i++) {
      if (_WorkerList[i].PostsId == Convert.ToInt32(PostsCBox.SelectedValue)) {
        _SecectedWorkerList.Add(_WorkerList[i]);
```

```
        }
        PostsValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      }
      LoadDataInWorkerGridView(_SecectedWorkerList);
    } else {
      PostsValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
    }
  }

  private void EmailBtn_Click(object sender, EventArgs e) {
    _SecectedWorkerList.Clear();
    if (_validation.IsDataEntering(PhoneTBox.Text)) {
      for (int i = 0; i < _WorkerList.Count(); i++) {
        if (_WorkerList[i].Phone.ToLower().Contains(PhoneTBox.Text.ToLower())) {
          _SecectedWorkerList.Add(_WorkerList[i]);
        }
        PhoneValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      }
      LoadDataInWorkerGridView(_SecectedWorkerList);
    } else {
      PhoneValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
    }
  }

  private void DepartmentBtn_Click(object sender, EventArgs e) {
    _SecectedWorkerList.Clear();
    if (Convert.ToInt32(DepartmentCBox.SelectedValue) > 0) {
      for (int i = 0; i < _WorkerList.Count(); i++) {
        if (_WorkerList[i].DepartmentId == Convert.ToInt32(DepartmentCBox.SelectedValue)) {
          _SecectedWorkerList.Add(_WorkerList[i]);
        }
        DepartmentValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      }
      LoadDataInWorkerGridView(_SecectedWorkerList);
    } else {
      DepartmentValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
    }
  }

  private void WorkerGridView_CellClick(object sender, DataGridViewCellEventArgs e) {
    if (e.RowIndex >= 0 && WorkerGridView[0, e.RowIndex].Value.ToString() != _WorkerList[0].Message) {
      _selectedRowIndex = e.RowIndex;
      UpdateWorkerForm updateWorkerForm = new UpdateWorkerForm(Convert.ToInt32(WorkerGridView[0,
e.RowIndex].Value.ToString()));
      updateWorkerForm.ShowDialog();
      DataLoad();
    }
  }
 }
}
```

Лістинг 5. Код класу «WorkerProvider»

```
using PersonnelManagementApp.AppCode;
using System;
using System.Collections.Generic;
using System.Data.OleDb;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.SqlClient;

namespace PersonnelManagementApp.Providers {
 internal class WorkerProvider {
  private string _ConnString = System.Configuration.ConfigurationSettings.AppSettings["CONNECT"];
  public void InsertWorker(string LastName, string FirstName, string Phone, string Address, string Email, double
Salary, DateTime DateOfPlacement, int PostsId, int DepartmentId) {
    string SqlString = "INSERT INTO Worker (LastName, FirstName, Phone, Address, Email, Salary,
DateOfPlacement, PostsId, DepartmentId) " +
                "Values(@LastName, @FirstName, @Phone, @Address, @Email, @Salary, @DateOfPlacement,
@PostsId, @DepartmentId)";

    using (SqlConnection conn = new SqlConnection(_ConnString)) {
```

```
    using (SqlCommand cmd = new SqlCommand(SqlString, conn)) {
      cmd.CommandType = CommandType.Text;
      cmd.Parameters.AddWithValue("@LastName", LastName);
      cmd.Parameters.AddWithValue("@FirstName", FirstName);
      cmd.Parameters.AddWithValue("@Phone", Phone);
      cmd.Parameters.AddWithValue("@Address", Address);
      cmd.Parameters.AddWithValue("@Email", Email);
      cmd.Parameters.AddWithValue("@Salary", Salary);
      cmd.Parameters.AddWithValue("@DateOfPlacement", DateOfPlacement);
      cmd.Parameters.AddWithValue("@PostsId", PostsId);
      cmd.Parameters.AddWithValue("@DepartmentId", DepartmentId);
      conn.Open();
      cmd.ExecuteNonQuery();
      conn.Close();
    }
  }
}

public List<Worker> GetAllWorker() {
  int i = 0;
  string SqlString = "SELECT * FROM Worker";

  List<Worker> listAllWorker = new List<Worker>();
  using (SqlConnection conn = new SqlConnection(_ConnString)) {
    using (SqlCommand cmd = new SqlCommand(SqlString, conn)) {
      conn.Open();
      using (SqlDataReader reader = cmd.ExecuteReader()) {
        while (reader.Read()) {
          Worker oneWorker = new Worker();
          oneWorker.Number = ++i;
          oneWorker.WorkerId = Convert.ToInt32(reader["WorkerId"]);
          oneWorker.FirstName = reader["FirstName"].ToString();
          oneWorker.LastName = reader["LastName"].ToString();
          oneWorker.FIO = oneWorker.LastName + " " + oneWorker.FirstName;
          oneWorker.Phone = reader["Phone"].ToString();
          oneWorker.Address = reader["Address"].ToString();
          oneWorker.Email = reader["Email"].ToString();
          oneWorker.Salary = Math.Round(Convert.ToDouble(reader["Salary"]),2);
          oneWorker.DateOfPlacement = Convert.ToDateTime(reader["DateOfPlacement"]);
          oneWorker.PostsId = Convert.ToInt32(reader["PostsId"]);
          oneWorker.DepartmentId = Convert.ToInt32(reader["DepartmentId"]);
          listAllWorker.Add(oneWorker);
        }
      }
      conn.Close();
    }
  }

  if (listAllWorker.Count == 0) {
    Worker noWorker = new Worker();
    noWorker.WorkerId = 0;
    noWorker.Message = NamesMy.NoDataNames.NoDataInWorker;
    listAllWorker.Add(noWorker);
  }
  return listAllWorker;
}


public Worker SelectedWorkerByWorkerId(int WorkerId) {
  string SqlString = "SELECT * FROM Worker WHERE WorkerId = @WorkerId";

  Worker oneWorker = new Worker();
  using (SqlConnection conn = new SqlConnection(_ConnString)) {
    using (SqlCommand cmd = new SqlCommand(SqlString, conn)) {
      cmd.Parameters.AddWithValue("@WorkerId", WorkerId);
      conn.Open();
      using (SqlDataReader reader = cmd.ExecuteReader()) {
        while (reader.Read()) {
          oneWorker.WorkerId = Convert.ToInt32(reader["WorkerId"]);
          oneWorker.FirstName = reader["FirstName"].ToString();
          oneWorker.LastName = reader["LastName"].ToString();
          oneWorker.FIO = oneWorker.LastName + " " + oneWorker.FirstName;
          oneWorker.Phone = reader["Phone"].ToString();
```

```
            oneWorker.Address = reader["Address"].ToString();
            oneWorker.Email = reader["Email"].ToString();
            oneWorker.Salary = Math.Round(Convert.ToDouble(reader["Salary"]), 2);
            oneWorker.DateOfPlacement = Convert.ToDateTime(reader["DateOfPlacement"]);
            oneWorker.PostsId = Convert.ToInt32(reader["PostsId"]);
            oneWorker.DepartmentId = Convert.ToInt32(reader["DepartmentId"]);
          }
        }
        conn.Close();
      }
    }
    return oneWorker;
  }


  public void UpdateWorker(string LastName, string FirstName, string Phone, string Address, string Email, double
Salary,
      DateTime DateOfPlacement, int PostsId, int DepartmentId, int WorkerId) {
    string SqlString = "UPDATE Worker SET FirstName = @FirstName, LastName = @LastName, Phone = @Phone,
" +
      "Address = @Address, Email = @Email, Salary = @Salary, DateOfPlacement = @DateOfPlacement, PostsId =
@PostsId, DepartmentId = @DepartmentId WHERE WorkerId = @WorkerId";

    using (SqlConnection conn = new SqlConnection(_ConnString)) {
      using (SqlCommand cmd = new SqlCommand(SqlString, conn)) {
        cmd.CommandType = CommandType.Text;
        cmd.Parameters.AddWithValue("@FirstName", FirstName);
        cmd.Parameters.AddWithValue("@LastName", LastName);
        cmd.Parameters.AddWithValue("@Phone", Phone);
        cmd.Parameters.AddWithValue("@Address", Address);
        cmd.Parameters.AddWithValue("@Email", Email);
        cmd.Parameters.AddWithValue("@Salary", Salary);
        cmd.Parameters.AddWithValue("@DateOfPlacement", DateOfPlacement);
        cmd.Parameters.AddWithValue("@PostsId", PostsId);
        cmd.Parameters.AddWithValue("@DepartmentId", DepartmentId);
        cmd.Parameters.AddWithValue("@WorkerId", WorkerId);
        conn.Open();
        cmd.ExecuteNonQuery();
        conn.Close();
      }
    }
  }

  public void DeleteWorkerByWorkerId(int WorkerId) {
    string SqlString = "DELETE FROM Worker WHERE WorkerId = @WorkerId";

    using (SqlConnection conn = new SqlConnection(_ConnString)) {
      using (SqlCommand cmd = new SqlCommand(SqlString, conn)) {
        cmd.Parameters.AddWithValue("@WorkerId", WorkerId);
        conn.Open();
        cmd.ExecuteNonQuery();
        conn.Close();
      }
    }
  }

  public void DeleteAllWorker() {
    string SqlString = "DELETE FROM Worker";

    using (SqlConnection conn = new SqlConnection(_ConnString)) {
      using (SqlCommand cmd = new SqlCommand(SqlString, conn)) {
        conn.Open();
        cmd.ExecuteNonQuery();
        conn.Close();
      }
    }
  }

 }
}


public class Worker {
```

```
private int _Number;
private int _WorkerId;
private string _FirstName;
private string _LastName;
private string _Phone;
private string _Address;
private string _Email;
private string _FIO;
private double _Salary;
private DateTime _DateOfPlacement;
private int _PostsId;
private int _DepartmentId;
private string _Message;
public Worker() {
  _Number = 0;
  _WorkerId = 0;
  _FirstName = String.Empty;
  _LastName = String.Empty;
  _Phone = String.Empty;
  _Address = String.Empty;
  _Email = String.Empty;
  _FIO = String.Empty;
  _Salary = 0.0;
  _DateOfPlacement = new DateTime();
  _PostsId = 0;
  _DepartmentId = 0;
  _Message = String.Empty;
}
public int Number {
  set { _Number = value; }
  get { return _Number; }
}
public int WorkerId {
  set { _WorkerId = value; }
  get { return _WorkerId; }
}
public string FirstName {
  set { _FirstName = value; }
  get { return _FirstName; }
}
public string LastName {
  set { _LastName = value; }
  get { return _LastName; }
}
public string Phone {
  set { _Phone = value; }
  get { return _Phone; }
}
public string Address {
  set { _Address = value; }
  get { return _Address; }
}
public string Email {
  set { _Email = value; }
  get { return _Email; }
}
public string FIO {
  set { _FIO = value; }
  get { return _FIO; }
}
public double Salary {
  set { _Salary = value; }
  get { return _Salary; }
}
public DateTime DateOfPlacement {
  set { _DateOfPlacement = value; }
  get { return _DateOfPlacement; }
}
public int PostsId {
  set { _PostsId = value; }
  get { return _PostsId; }
}
public int DepartmentId {
```

```
    set { _DepartmentId = value; }
    get { return _DepartmentId; }
  }
  public string Message {
    set { _Message = value; }
    get { return _Message; }
  }
}
```

Додаток Б

Скрипти створення бази даних

```
USE [master]
GO
/****** Object:  Database [DB]    Script Date: 20.08.2024 12:45:00 ******/
CREATE DATABASE [DB]
 CONTAINMENT = NONE
 ON  PRIMARY
( NAME = N'DB', FILENAME = N'C:\Program Files\Microsoft SQL
Server\MSSQL16.MSSQLSERVER\MSSQL\DATA\DB.mdf' , SIZE = 8192KB , MAXSIZE = UNLIMITED,
FILEGROWTH = 65536KB )
 LOG ON
( NAME = N'DB_log', FILENAME = N'C:\Program Files\Microsoft SQL
Server\MSSQL16.MSSQLSERVER\MSSQL\DATA\DB_log.ldf' , SIZE = 8192KB , MAXSIZE = 2048GB ,
FILEGROWTH = 65536KB )
 WITH CATALOG_COLLATION = DATABASE_DEFAULT, LEDGER = OFF
GO
ALTER DATABASE [DB] SET COMPATIBILITY_LEVEL = 160
GO
IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
begin
EXEC [DB].[dbo].[sp_fulltext_database] @action = 'enable'
end
GO
ALTER DATABASE [DB] SET ANSI_NULL_DEFAULT OFF
GO
GO
/****** Object:  Table [dbo].[Department]    Script Date: 20.08.2024 12:45:01 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Department](
	[DepartmentId] [int] IDENTITY(1,1) NOT NULL,
	[DepartmentName] [nvarchar](250) NULL,
	[Description] [nvarchar](max) NULL,
PRIMARY KEY CLUSTERED
(
	[DepartmentId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
/****** Object:  Table [dbo].[Logs]    Script Date: 20.08.2024 12:45:01 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Logs](
	[LogsId] [int] IDENTITY(1,1) NOT NULL,
	[UsersId] [int] NULL,
	[EventNameShow] [nvarchar](max) NULL,
	[EventDate] [datetime] NULL,
PRIMARY KEY CLUSTERED
(
	[LogsId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
```

```
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
/****** Object:  Table [dbo].[Posts]    Script Date: 20.08.2024 12:45:01 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Posts](
	[PostsId] [int] IDENTITY(1,1) NOT NULL,
	[PostsName] [nvarchar](250) NULL,
	[Description] [nvarchar](max) NULL,
PRIMARY KEY CLUSTERED
(
	[PostsId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
/****** Object:  Table [dbo].[Users]    Script Date: 20.08.2024 12:45:01 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Users](
	[UsersId] [int] IDENTITY(1,1) NOT NULL,
	[FirstName] [nvarchar](50) NULL,
	[LastName] [nvarchar](50) NULL,
	[UsersName] [nvarchar](50) NULL,
	[UsersPassword] [nvarchar](120) NULL,
	[RoleId] [int] NULL,
	[Description] [nvarchar](1000) NULL,
	[Email] [nvarchar](150) NULL,
PRIMARY KEY CLUSTERED
(
	[UsersId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
) ON [PRIMARY]
GO
/****** Object:  Table [dbo].[Vacations]    Script Date: 20.08.2024 12:45:01 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Vacations](
	[VacationId] [int] IDENTITY(1,1) NOT NULL,
	[WorkerId] [int] NULL,
	[StartDate] [datetime] NULL,
	[EndDate] [datetime] NULL,
	[TypeVacation] [nvarchar](100) NULL,
PRIMARY KEY CLUSTERED
(
	[VacationId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
) ON [PRIMARY]
GO
/****** Object:  Table [dbo].[Worker]    Script Date: 20.08.2024 12:45:01 ******/
SET ANSI_NULLS ON
```

```
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Worker](
	[WorkerId] [int] IDENTITY(1,1) NOT NULL,
	[FirstName] [nvarchar](50) NULL,
	[LastName] [nvarchar](50) NULL,
	[Phone] [nvarchar](25) NULL,
	[Address] [nvarchar](max) NULL,
	[Email] [nvarchar](250) NULL,
	[Salary] [float] NULL,
	[DateOfPlacement] [datetime] NULL,
	[PostsId] [int] NULL,
	[DepartmentId] [int] NULL,
PRIMARY KEY CLUSTERED
(
	[WorkerId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
USE [master]
GO
ALTER DATABASE [DB] SET  READ_WRITE
GO
```