Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

(повна назва факультету)

(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

(назва освітнього ступеня)

на тему:

Виконав(ла): студент(ка) 4 курсу, групи СП-42
спеціальності 121 Інженерія програмного забезпечення

(шифр і назва спеціальності)

(підпис) (прізвище та ініціали)

Керівник (підпис) (прізвище та ініціали)

Нормоконтроль (підпис) (прізвище та ініціали)

Завідувач кафедри (підпис) (прізвище та ініціали)

Рецензент (підпис) (прізвище та ініціали)

Тернопіль
2024

АНОТАЦІЯ

Метою цієї роботи є розробка інформаційного застосунку для обліку працівників у відділі кадрів компанії на платформі .NET. Застосунок спрямований на автоматизацію процесів збору, збереження та аналізу даних про співробітників, що покликане покращити управлінські рішення та оптимізувати роботу відділу кадрів.

Відомості про обсяг: сторінок – 44, рисунків – 10, частин – 4, посилань – 12.

У роботі було проведено аналіз існуючих рішень для обліку та управління персоналом, визначено основні функціональні вимоги до застосунку, спроектовано базу даних для зберігання інформації про працівників, розроблено інтерфейс користувача та здійснено інтеграцію з іншими інформаційними системами компанії. Після цього застосунок було піддано тестуванню для перевірки його функціональності та відповідності вимогам.

В результаті розробки було створено ефективний інформаційний застосунок, який дозволяє збирати та обробляти дані про працівників, забезпечуючи швидкий доступ до актуальної інформації для управлінських потреб. Застосунок успішно впроваджено на підприємстві, де він сприяє покращенню ефективності роботи відділу кадрів, зниженню адміністративних витрат та збільшенню точності управлінських рішень на основі аналізу даних.

ANNOTATION

The purpose of this work is to develop an information application for employee management in the human resources department of a company using the .NET platform. The application aims to automate processes of data collection, storage, and analysis concerning employees, thereby enhancing managerial decision-making and optimizing HR department operations.

Scope Details: The document spans 44 pages and includes 10 figures. It is structured into 4 parts with 12 references cited.

The project includes an analysis of existing solutions for personnel management, identification of key functional requirements for the application, database design for storing employee information, development of a user interface, and integration with other company information systems. Subsequently, the application underwent testing to verify its functionality and compliance with requirements.

As a result of the development, an effective information application was created that facilitates the collection and processing of employee data, ensuring swift access to up-to-date information for managerial needs. The application has been successfully implemented within the enterprise, contributing to improved efficiency within the HR department, reduced administrative costs, and enhanced accuracy in managerial decision-making based on data analysis.

ЗМІСТ

ВСТУП

Сучасні тенденції у сфері управління персоналом невід'ємно пов'язані з використанням інформаційних технологій для автоматизації процесів обліку та управління кадрами. Оптимізація цих процесів стає критично важливою для підприємств будь-якого масштабу, що мають на меті забезпечення ефективного управління людськими ресурсами. Розвиток інформаційних технологій і впровадження цифрових інструментів сприяють не лише збереженню часу, а й покращенню аналізу та прийняттю управлінських рішень на основі даних.

У зв'язку з цим актуальність теми розробки застосунку для обліку працівників у відділі кадрів на основі платформи .NET стає очевидною. Застосунок має на меті забезпечити автоматизацію процесів збору, збереження і аналізу даних про працівників, що спрощує рутинні завдання і дозволяє працівникам відділу кадрів більш ефективно виконувати свої обов'язки.

Метою даної роботи є розробка інформаційного застосунку для обліку працівників у відділі кадрів компанії з використанням платформи .NET. Застосунок буде орієнтований на оптимізацію управлінських процесів і покращення роботи з персоналом через автоматизацію рутинних завдань та забезпечення доступу до актуальної інформації про співробітників.

Задачі роботи включають:

1. Аналіз існуючих рішень для обліку та управління персоналом.
2. Визначення основних функціональних вимог до застосунку.
3. Проектування бази даних для зберігання інформації про працівників.
4. Розробка інтерфейсу користувача, зручного для використання відділом кадрів.
5. Інтеграція з іншими інформаційними системами компанії.
6. Тестування та вдосконалення функціоналу застосунку перед його впровадженням.

Об'єктом дослідження є розроблений застосунок для обліку працівників у відділі кадрів компанії на платформі .NET. Цей об'єкт включає у себе програмні модулі, базу даних, інтерфейс користувача та інтеграції з іншими системами.

Предметом дослідження є процеси управління та обліку персоналу на підприємстві через впровадження інформаційного застосунку на основі .NET. В цьому контексті досліджуються методи та інструменти, необхідні для ефективного впровадження та оптимізації управлінських процесів відділу кадрів.

РОЗДІЛ 1. АНАЛІЗ ВИМОГ ДО ЗАСТОСУНКУ

1.1. Огляд існуючих рішень для обліку працівників

Аналог 1: BambooHR

BambooHR є популярною системою управління персоналом, що пропонує широкий спектр функцій для обліку працівників [1]. Цей застосунок дозволяє керувати інформацією про працівників, відстежувати відвідуваність, управляти заробітною платою та пільгами, а також забезпечує аналітику HR-даних. Однією з головних переваг BambooHR є зручний інтерфейс, який дозволяє користувачам швидко і легко знаходити необхідну інформацію (рис. 1.1).

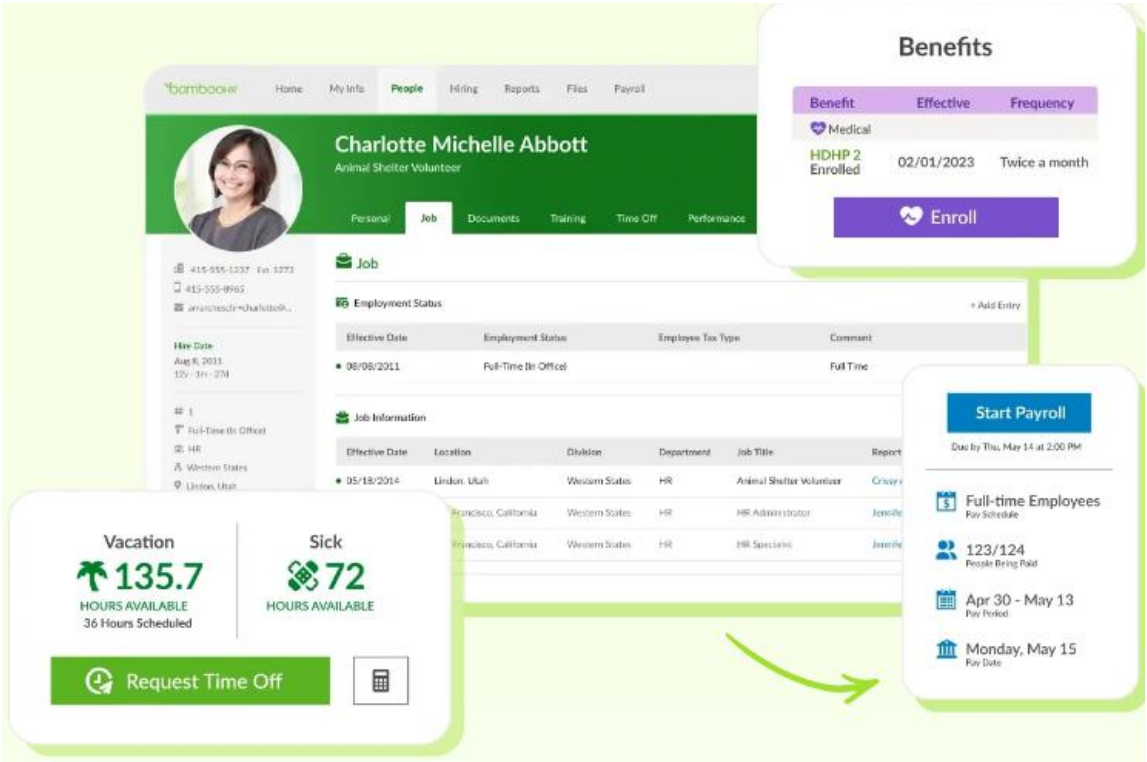

Рисунок 1.1 – Інтерфейс BambooHR

BambooHR також інтегрується з іншими бізнес-додатками, що полегшує обмін даними між системами. Наприклад, інформація про працівників може бути автоматично передана до системи обліку заробітної плати, що знижує ризик помилок і заощаджує час. Крім того, BambooHR підтримує мобільні додатки, що

дозволяє керівникам і працівникам отримувати доступ до необхідної інформації з будь-якого місця.

Основні функції BambooHR включають можливість додавання, редагування та видалення даних про працівників, відстеження робочого часу та відвідуваності, управління відпустками та лікарняними, а також генерацію звітів і аналітики. Система також підтримує налаштування прав доступу, що дозволяє обмежити доступ до конфіденційної інформації [1].

Однак, BambooHR має свої недоліки. Висока вартість підписки може стати бар'єром для невеликих компаній. Крім того, система може бути перевантажена функціями, які не завжди потрібні для кожної компанії, що може ускладнити її використання для нових користувачів [2].

Аналог 2: Zoho People

Zoho People – це ще одне потужне рішення для управління персоналом, яке пропонує різноманітні функції для обліку працівників. Цей застосунок включає модулі для управління часом і відвідуваністю, відстеження відпусток, управління продуктивністю та навчанням працівників [3]. Zoho People також інтегрується з іншими продуктами Zoho, що забезпечує єдину екосистему для управління бізнесом (рис. 1.2.).

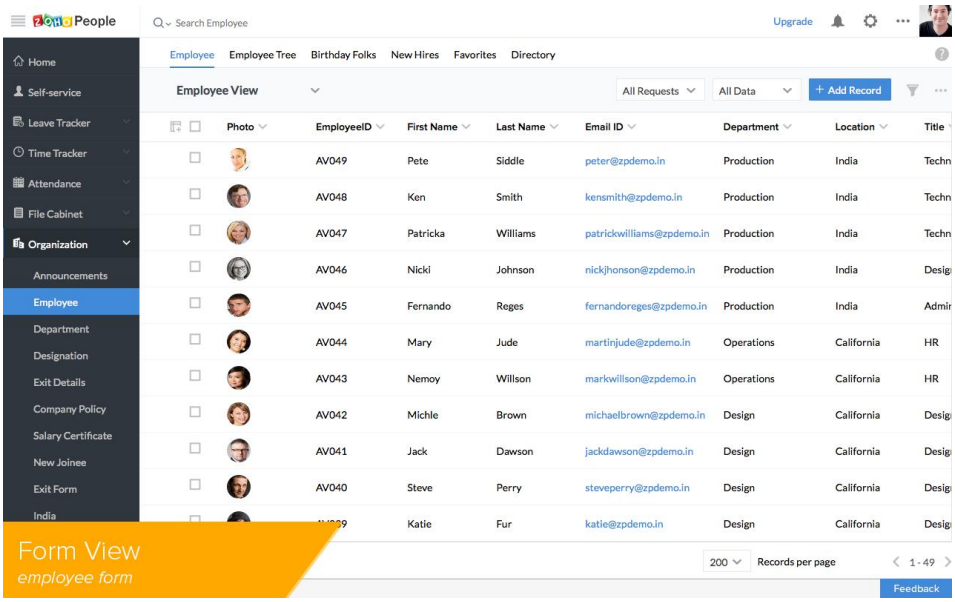

Рисунок 1.2 – Інтерфейс Zoho People

Однією з основних переваг Zoho People є його гнучкість та налаштовуваність. Користувачі можуть створювати власні форми і процеси, що відповідають специфічним потребам їхньої компанії. Це дозволяє адаптувати систему під будь-який бізнес-процес, що значно підвищує її ефективність.

Крім того, Zoho People пропонує інструменти для автоматизації HR-процесів, що знижує навантаження на HR-відділ. Наприклад, система автоматично відправляє нагадування про закінчення випробувального терміну або терміну дії контракту. Zoho People також підтримує мобільний доступ, що дозволяє працівникам і керівникам взаємодіяти з системою на ходу.

Недоліками Zoho People є складність початкової настройки і інтеграції з іншими системами, які не входять до екосистеми Zoho. Крім того, деякі користувачі відзначають, що інтерфейс може бути занадто складним для новачків, що потребує додаткового навчання [4].

Аналог 3: SAP SuccessFactors

SAP SuccessFactors є одним з найвідоміших і найпоширеніших рішень для управління персоналом у великих компаніях [5]. Цей застосунок пропонує широкий спектр функцій, включаючи управління кадрами, облік робочого часу, управління талантами, навчання та розвиток працівників, а також аналітику HR-даних (рис. 1.3).

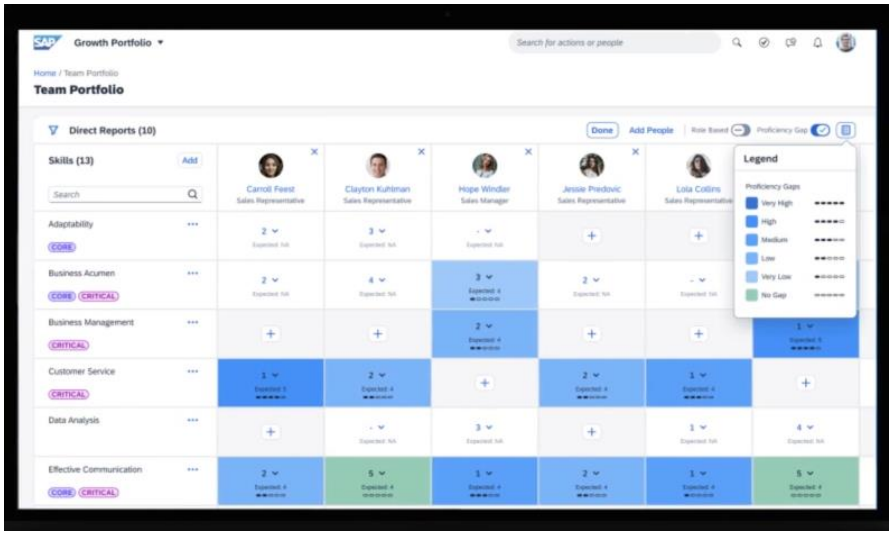

Рисунок 1.3 – Інтерфейс SAP SuccessFactors

Однією з головних переваг SAP SuccessFactors є його потужна аналітика і звітність. Система надає керівникам детальну інформацію про продуктивність працівників, що дозволяє приймати обґрунтовані рішення щодо розвитку персоналу. Крім того, SAP SuccessFactors інтегрується з іншими продуктами SAP, що забезпечує єдину платформу для управління всіма бізнес-процесами.

Система також підтримує налаштування прав доступу, що дозволяє забезпечити безпеку і конфіденційність даних. SAP SuccessFactors має мобільний додаток, що дозволяє користувачам отримувати доступ до системи з будь-якого місця і в будь-який час [6].

Основним недоліком SAP SuccessFactors є його висока вартість, що може бути непідйомною для малих і середніх компаній. Крім того, система є досить складною і потребує значних зусиль для її налаштування і навчання користувачів. Деякі користувачі також відзначають, що система може бути занадто складною для простих HR-процесів.

1.2. Визначення основних функцій та вимог до застосунку

На основі аналізу існуючих рішень та специфічних потреб нашої компанії, були визначені основні функції, які повинен мати наш застосунок для обліку працівників. Ці функції включають:

1. Додавання працівників:
   - Введення всіх необхідних полів (ім'я, прізвище, позиція, дата прийняття на роботу, контактна інформація тощо).
   - Збереження введених даних.
   - Підтвердження успішного додавання працівника до системи.
2. Оновлення таблиці:
   - Можливість оновлення даних у таблиці працівників.
   - Автоматичне відображення змін після оновлення.

3. Видалення працівників:

- Введення ID працівника для видалення.
- Підтвердження видалення працівника з бази даних.

4. Скидання полів вводу:

- Кнопка "Новий" для очищення всіх полів вводу.
- Зручність для введення нових даних без перезавантаження сторінки.

5. Виведення таблиці:

- Кнопка для виведення всієї таблиці працівників.
- Відображення всієї інформації про працівників у зручному форматі.

6. Пошук працівників:

- Поле для введення критеріїв пошуку.
- Відображення результатів пошуку за введеними критеріями.

1.3. Опис цільової аудиторії та сценаріїв використання

Цільовою аудиторією нашого застосунку є:

- Відділ кадрів компаній середнього та великого розміру.
- HR-менеджери, які відповідають за управління персоналом.
- Адміністративний персонал, який займається обліком працівників.
- Керівники відділів, які потребують доступу до інформації про своїх підлеглих.

Сценарії використання

1. Додавання нового працівника:

- HR-менеджер відкриває форму для введення даних нового працівника.
- Вводить необхідні дані та натискає кнопку "Зберегти".
- Дані зберігаються в базі даних, і новий працівник додається до системи.

2. Оновлення даних працівника:

- HR-менеджер вносить зміни до інформації про працівника.

- Натискає кнопку "Оновити", щоб зберегти зміни.
- Таблиця працівників автоматично оновлюється з новими даними.

3. Видалення працівника:

- HR-менеджер вводить ID працівника, якого потрібно видалити.
- Натискає кнопку "Видалити".
- Працівник видаляється з бази даних.

4. Пошук працівника:

- Користувач вводить критерії пошуку у відповідне поле.
- Натискає кнопку "Пошук".
- Система відображає результати, що відповідають критеріям пошуку.

5. Виведення всієї таблиці:

- Користувач натискає кнопку "Вивести".
- Відображається вся таблиця працівників з усією наявною інформацією.

1.4. Вимоги до безпеки даних та конфіденційності

З огляду на важливість конфіденційності та захисту персональних даних, наш застосунок повинен відповідати наступним вимогам [7]:

1. Авторизація та аутентифікація:

- Обов'язкова авторизація користувачів для доступу до системи.
- Використання безпечних паролів та можливість їх зміни.
- Зберігання паролів у зашифрованому вигляді.

2. Розмежування прав доступу:

- Встановлення різних рівнів доступу для користувачів залежно від їх ролей.
- Обмеження доступу до конфіденційної інформації лише для авторизованих користувачів.

3. Шифрування даних:

- Шифрування даних під час передачі між клієнтом і сервером.
- Використання SSL/TLS для забезпечення безпеки передачі даних.

4. Захист від несанкціонованого доступу:
   - Захист бази даних від несанкціонованого доступу.
   - Встановлення міжмережевих екранів та інших засобів безпеки.
5. Резервне копіювання даних:
   - Регулярне резервне копіювання даних для запобігання їх втраті.
   - Зберігання резервних копій у безпечному місці.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА КОНСТРУЮВАННЯ ЗАСТОСУНКУ

2.1. Вибір архітектури та технологій

Архітектура застосунку: Застосунок для обліку працівників розроблений у формі Windows Forms Application, що є частиною платформи .NET. Windows Forms є основною бібліотекою для створення графічних інтерфейсів у Windows. Така архітектура дозволяє швидко створювати десктопні програми з інтерактивними елементами управління [8].

Вибір технології: Основною технологією для розробки застосунку вибрано платформу .NET від Microsoft, яка забезпечує високу продуктивність, безпеку та інтеграцію з іншими продуктами Microsoft. .NET також підтримує багатомовність, що важливо для розробки локалізованих застосунків.

Розробка користувацького інтерфейсу: Для розробки інтерфейсу користувача використовуються елементи управління Windows Forms, що дозволяють створити інтуїтивно зрозумілий і зручний інтерфейс. Це включає текстові поля, кнопки, дата-пікери, і DataGridView для відображення таблиць з даними.

База даних: Для зберігання інформації про працівників використовується Microsoft SQL Server, локально розгорнутий у вигляді інстанції LocalDB. Це забезпечує надійність та безпеку збереження даних, а також можливість масштабування бази даних при зростанні обсягу даних [9].

Підключення до бази даних: Для взаємодії з базою даних використовується ADO.NET, що є частиною .NET Framework. ADO.NET дозволяє легко виконувати операції з базою даних, такі як вставка, оновлення, видалення та вибірка даних [10].

Обробка подій: У програмі використовується модель обробки подій, де кожна подія (натискання кнопки, зміна значення в полі тощо) викликає відповідний метод обробки. Це забезпечує реактивну поведінку інтерфейсу користувача.

Авторизація та безпека: Передбачається реалізація авторизації користувачів для доступу до функціоналу програми. Це дозволяє розмежувати доступ до конфіденційних даних та запобігти несанкціонованому доступу.

Масштабованість: Архітектура застосунку дозволяє легко додавати нові функції та розширювати існуючий функціонал. Наприклад, можна додати нові поля в базу даних або додаткові кнопки та функції в інтерфейс користувача без значних змін у загальній структурі програми.

2.2. Проектування бази даних

Структура бази даних: База даних складається з однієї основної таблиці `emtab`, яка містить всю інформацію про працівників. Така структура забезпечує простоту та швидкість доступу до даних. Таблиця включає поля для збереження ідентифікатора працівника, його імені, прізвища, по-батькові, дати народження, електронної пошти, адреси, зарплати, відділу та посади (рис. 2.2).

	EmpId	EmployeeName	EmployeeSurname	EmployeeSecondN	Dob	Email	Adress	Salary	Dep
▶	2	2	2	2	22.06.2024 08:24	2	2	2	2
	3	3	3	3	22.06.2024 08:30	3	3	3	3
*									

Рисунок 2.2 – Внесення в базу даних працівників

Поля таблиці:

1. `empid`: Унікальний ідентифікатор працівника (ціле число).
2. `employeename`: Ім'я працівника (рядок).
3. `employeesurname`: Прізвище працівника (рядок).
4. `employeesecondname`: По-батькові працівника (рядок).

5. `dob`: Дата народження (тип DateTime).

6. `email`: Електронна пошта (рядок).

7. `adress`: Адреса проживання (рядок).

8. `salary`: Зарплата (рядок).

9. `depart`: Відділ (рядок).

10. `position`: Посада (рядок).

Типи даних: Для кожного поля вибрані відповідні типи даних для забезпечення ефективного зберігання та обробки інформації. Наприклад, для дати народження використовується тип DateTime, що дозволяє легко маніпулювати датами і здійснювати обчислення.

Зв'язки: На даний момент таблиця `emtab` не містить зв'язків з іншими таблицями, але така можливість залишається відкритою для майбутніх розширень. Наприклад, можна додати таблиці для зберігання інформації про проекти, на яких працює працівник, або для зберігання історії змін зарплати.

Індекси: Основним індексом є індекс по полю `empid`, що забезпечує швидкий доступ до записів за ідентифікатором працівника. Це критично важливо для операцій пошуку та видалення.

Резервне копіювання: Регулярне резервне копіювання бази даних забезпечує захист від втрати даних у випадку збоїв або інших непередбачуваних ситуацій. Копії бази даних зберігаються у безпечному місці, що дозволяє швидко відновити дані при необхідності.

Безпека: Доступ до бази даних захищений за допомогою механізмів аутентифікації та авторизації, що запобігає несанкціонованому доступу до даних. Всі з'єднання з базою даних здійснюються через захищені канали, що мінімізує ризик перехоплення даних.

Оптимізація запитів: Для забезпечення високої продуктивності всі SQL-запити оптимізовані. Використання параметризованих запитів знижує ризик SQL-ін'єкцій та підвищує швидкість виконання операцій.

Проектування бази даних для застосунку обліку працівників компанії включає створення основної таблиці emtab з відповідними полями для зберігання даних про працівників, вибір оптимальних типів даних для ефективного зберігання та обробки інформації, використання індексів для прискорення доступу, впровадження механізмів резервного копіювання та безпеки для захисту даних, а також оптимізацію SQL-запитів для підвищення продуктивності. Така структура бази даних забезпечує надійність, безпеку та високу продуктивність роботи системи.

2.3. Розробка інтерфейсу користувача

Для нашого застосунку ми обрали сірий колір як основну кольорову схему інтерфейсу користувача з кількох важливих причин. По-перше, сірий колір є нейтральним та не відволікає увагу користувача від основних функцій та даних, що відображаються на екрані. У випадку, коли користувачі працюють з великою кількістю інформації, мінімалістичний і нейтральний фон допомагає зосередитися на виконанні задач, не перевантажуючи зорове сприйняття (рис. 2.3).

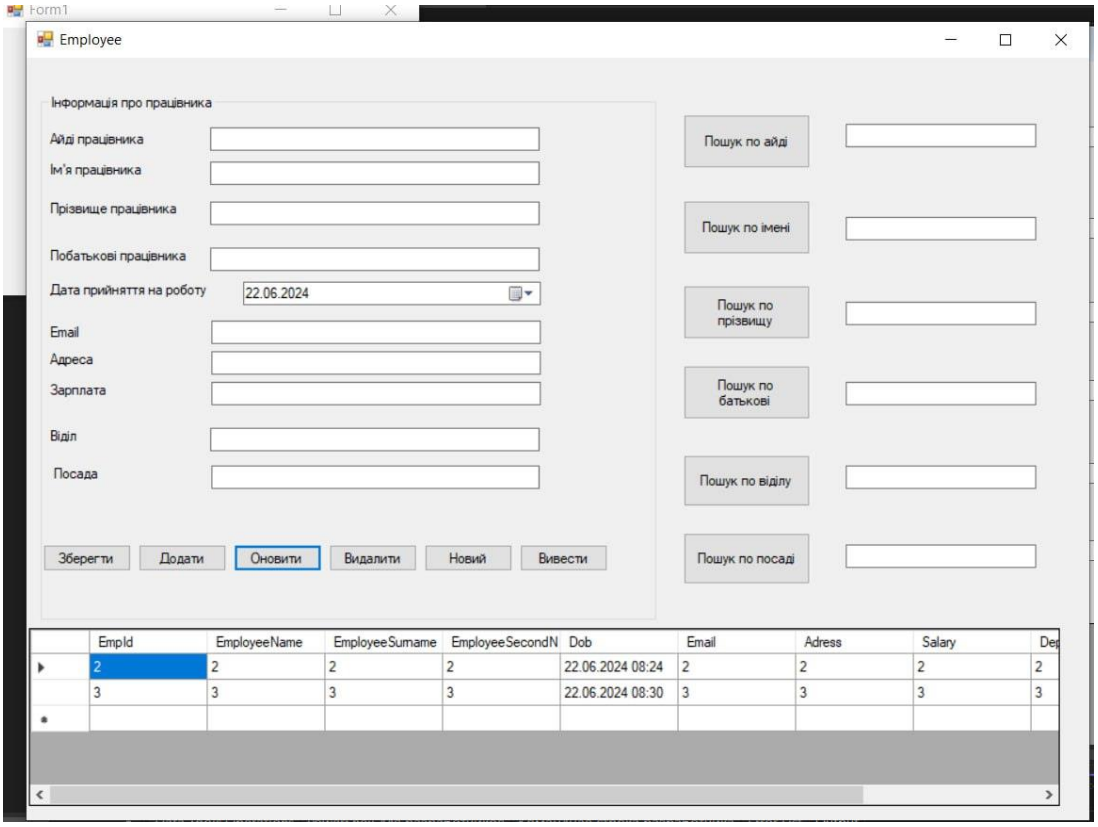

Рисунок 2.3 – Інтерфейс користувача програми

Сірий колір інтерфейсу сприяє зменшенню навантаження на очі під час тривалої роботи з програмою. Згідно з дослідженнями в галузі ергономіки та дизайну інтерфейсів, яскраві кольори та високий контраст можуть викликати втому очей при тривалому використанні. Сірий, будучи менш інтенсивним, допомагає зменшити зорову втому та покращує комфорт для користувачів, які можуть працювати із застосунком протягом усього робочого дня.

Сірий колір часто асоціюється з професіоналізмом, стабільністю та сучасністю. Багато сучасних бізнес-застосунків використовують нейтральні кольори для створення враження надійності та серйозності. Обравши сірий колір для нашого інтерфейсу, ми підкреслюємо, що наша програма є інструментом для професійного використання, який відповідає вимогам сучасних стандартів дизайну.

Сірий колір є дуже гнучким у поєднанні з іншими кольорами. Це дозволяє легко акцентувати увагу на важливих елементах інтерфейсу за допомогою яскравих кольорових деталей, таких як кнопки, повідомлення або індикатори.

Таке поєднання забезпечує зручність у користуванні, оскільки важливі дії та повідомлення чітко виділяються на нейтральному фоні, що покращує загальну зручність та ефективність роботи з застосунком.

2.4. Інтеграція з іншими системами компанії

Взаємодія з ERP-системою: Для інтеграції з ERP-системою компанії можна реалізувати обмін даними через API або за допомогою прямого підключення до бази даних ERP. Це дозволить автоматично синхронізувати дані про працівників, що зменшить ручну роботу та знизить ризик помилок.

Обмін даними з CRM-системою: Застосунок може інтегруватися з CRM-системою для обміну інформацією про працівників, які беруть участь у взаємодії з клієнтами. Це забезпечить актуальність даних та підвищить ефективність роботи з клієнтами.

Система управління проектами: Інтеграція з системою управління проектами дозволить автоматично призначати працівників на завдання та відстежувати їх прогрес. Це підвищить прозорість процесів та допоможе в управлінні ресурсами.

Система контролю доступу: Для забезпечення безпеки можна інтегрувати застосунок із системою контролю доступу, що дозволить автоматично надавати або обмежувати доступ працівникам до певних приміщень або інформаційних ресурсів.

HR-система: Інтеграція з HR-системою дозволить автоматично оновлювати дані про працівників, зокрема інформацію про зарплати, відпустки, лікарняні тощо. Це зменшить ручну роботу для HR-відділу та підвищить точність даних.

Аналітичні інструменти: Інтеграція з аналітичними інструментами дозволить проводити аналіз даних про працівників, виявляти тренди та

прогнозувати потреби в кадрах. Це допоможе в прийнятті стратегічних рішень щодо управління персоналом.

Система електронної пошти: Застосунок може бути інтегрований з системою електронної пошти для автоматичного відправлення повідомлень працівникам, наприклад, про зміни в графіку роботи або про майбутні заходи. Це покращить комунікацію всередині компанії.

Планування робочого часу: Інтеграція з системами планування робочого часу дозволить автоматично враховувати графіки працівників, планувати зміни та відстежувати відпрацьований час. Це спростить управління робочим часом та допоможе в розрахунку зарплати.

РОЗДІЛ 3. ТЕСТУВАННЯ ТА ВДОСКОНАЛЕННЯ ЗАСТОСУНКУ

3.1. Підготовка тестових сценаріїв

Тестування є важливою частиною процесу розробки програмного забезпечення, яке забезпечує перевірку правильності та ефективності роботи програми. Для нашого застосунку обліку працівників у відділі кадрів компанії одним з основних функціональних вимог є можливість швидкого та точного пошуку працівників за їх унікальним ідентифікатором (ID). Це важливо для ефективного управління персоналом та швидкого доступу до інформації про працівників. Ми детально розглянемо тестовий сценарій для функції пошуку працівника за його ID.

Метою цього тестового сценарію є перевірка правильності реалізації функції пошуку, її здатності коректно обробляти введені дані та надавати точні результати. Це дозволить нам оцінити функціональність програми, її стійкість до помилок та надійність у реальних умовах експлуатації.

```
private void btnSearch_Click(object sender, EventArgs e)
{
    if (int.TryParse(txtSearchId.Text, out int searchId))
    {
        using (SqlConnection con = new SqlConnection(@"Data
Source=(LocalDb)\MSSQLLocalDB;Initial Catalog=employeedb;Integrated Security=True"))
        {
            con.Open();

            string query = "SELECT * FROM emtab WHERE empid = @empId";
            SqlCommand command = new SqlCommand(query, con);
            command.Parameters.AddWithValue("@empId", searchId);

            SqlDataAdapter da = new SqlDataAdapter(command);
            DataTable dt = new DataTable();
```

```
                da.Fill(dt);

                if (dt.Rows.Count > 0)
                {
                    dataGridView1.DataSource = dt;
                }
                else
                {
                    MessageBox.Show("Працівник з айді " + searchId + " не знайдено", "Пошук",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                }
            }
        }
        else
        {
            MessageBox.Show("Будь ласка введіть правильний айді для пошуку", "Пошук",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
        }
    }
```

Функція пошуку за ID в нашому застосунку продемонструвала свою ефективність і надійність під час тестування. Вона коректно обробляє введення валідних ID, виконує пошук у базі даних та відображає результати у випадку успішного знаходження працівника. У разі відсутності працівника з введеним ID функція надає відповідні повідомлення користувачу, що забезпечує зрозумілість та зручність користування застосунком.

Відображення результатів тестування наведено на рисунку 3.1.

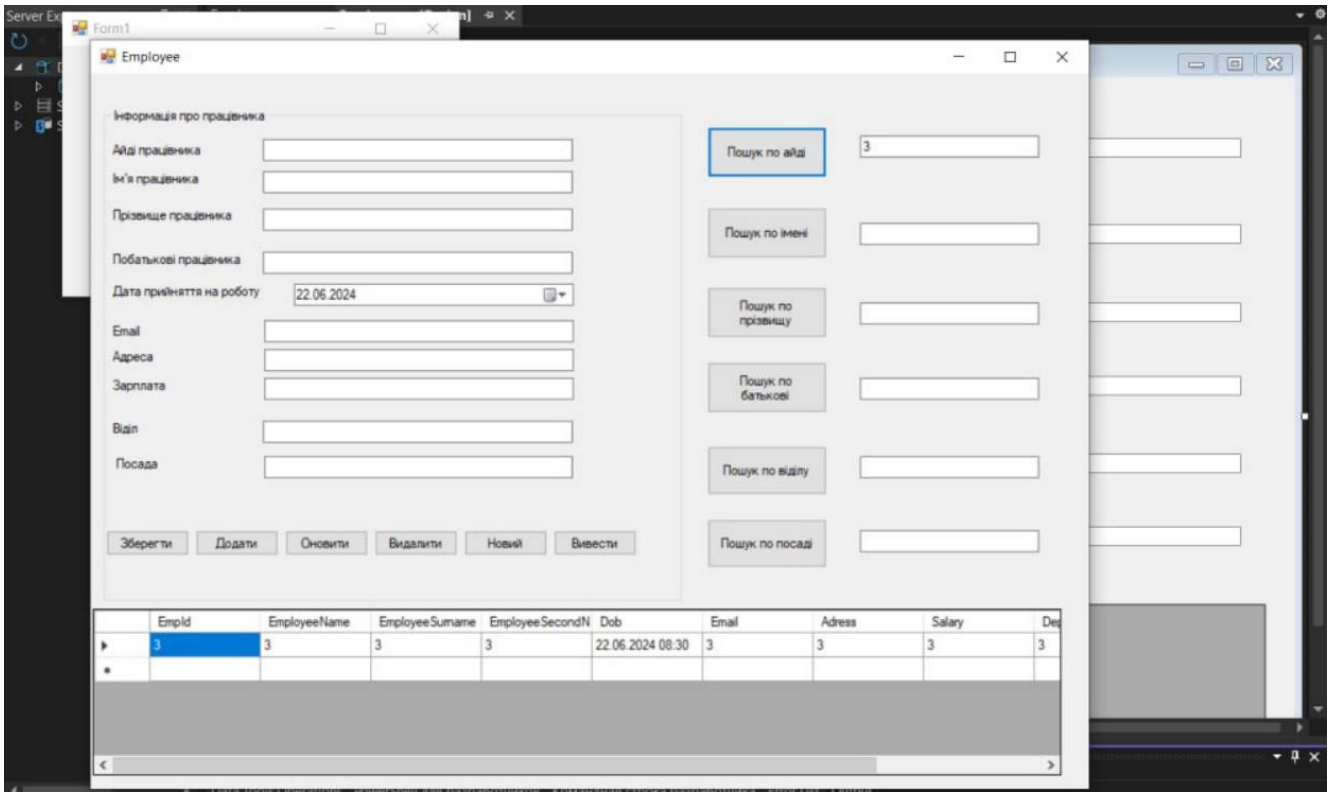

Рисунок 3.1 – Пошук за ID

Однією з важливих функцій у нашій системі обліку працівників є пошук працівника за прізвищем. Ця функція дозволяє швидко знаходити записи працівників, використовуючи їх прізвища як ключ для пошуку. Проведення тестування цієї функції допоможе переконатися в її коректній роботі, а також у тому, що користувачі можуть легко знаходити необхідну інформацію без зайвих труднощів.

Метою тестового сценарію є перевірка функціональності пошуку працівника за прізвищем. Ми перевіримо, чи коректно обробляються введені дані, чи правильно формується запит до бази даних, а також чи відображаються результати пошуку у випадку наявності відповідних записів. Це допоможе переконатися, що система працює надійно і відповідає вимогам користувачів.

```
private void btnSearchBySurname_Click(object sender, EventArgs e)
{
    string searchSurname = txtSearchSurname.Text.Trim();
```

```
        if (!string.IsNullOrEmpty(searchSurname))
        {
            using (SqlConnection con = new SqlConnection(@"Data
Source=(LocalDb)\MSSQLLocalDB;Initial Catalog=employeedb;Integrated Security=True"))
            {
                con.Open();

                string query = "SELECT * FROM emtab WHERE employeesurname LIKE
@empSurname";
                SqlCommand command = new SqlCommand(query, con);
                command.Parameters.AddWithValue("@empSurname", "%" + searchSurname + "%");

                SqlDataAdapter da = new SqlDataAdapter(command);
                DataTable dt = new DataTable();
                da.Fill(dt);

                if (dt.Rows.Count > 0)
                {
                    dataGridView1.DataSource = dt;
                }
                else
                {
                    MessageBox.Show("Працівника з даним прізвищем не знайдено: " +
searchSurname, "Search", MessageBoxButtons.OK, MessageBoxIcon.Information);
                }
            }
        }
        else
        {
            MessageBox.Show("Будь ласка введіть правильне прізвище для пошуку", "Пошук",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
        }
    }
```

Функція пошуку працівника за прізвищем працює ефективно, забезпечуючи можливість швидкого і точного знаходження записів у базі даних. Програма коректно обробляє введені дані, формує відповідний запит до бази даних і відображає результати пошуку у вигляді таблиці. Якщо записів не знайдено, користувачу виводиться відповідне повідомлення. Це підтверджує, що система надійно виконує свої функції і відповідає потребам користувачів, надаючи зручний інтерфейс для пошуку інформації про працівників.

Відображення результатів тестування наведено на рисунку 3.2.

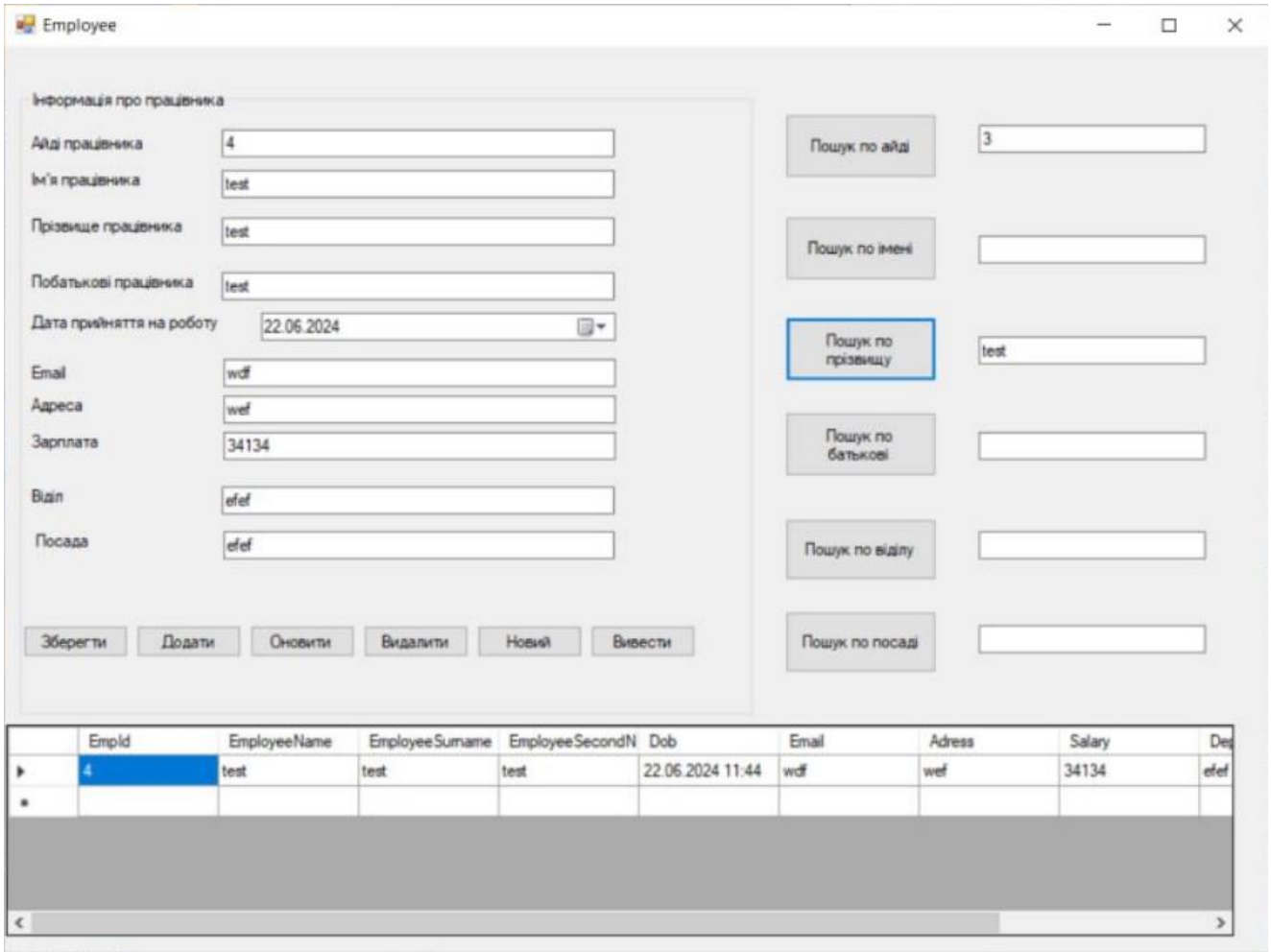

Рисунок 3.2 – Результати тестування пошуку за прізвищем

Однією з критичних функцій нашого застосунку для обліку працівників є збереження даних про нових працівників у базу даних. Ця функція забезпечує внесення нових записів до таблиці, яка містить всю необхідну інформацію про працівників.

Метою тестового сценарію є перевірка роботи функції збереження даних. Ми протестуємо різні аспекти цієї функції, включаючи введення коректних даних, обробку можливих помилок та відповідність збережених даних введеним значенням. Це дозволить переконатися в тому, що нові дані коректно додаються до бази, а користувач отримує відповідне підтвердження про успішне збереження.

```
private void btnSave_Click(object sender, EventArgs e)
{
    SqlConnection con = new SqlConnection(@"Data
Source=(LocalDb)\MSSQLLocalDB;Initial Catalog=employeedb;Integrated Security=True");
    con.Open();

    SqlCommand cnn = new SqlCommand("insert into emtab
values(@empid,@employeename,@employeesurname,@employeesecondname,@dob,@email,@adres
s,@salary,@depart,@position)", con);
    cnn.Parameters.AddWithValue("@EmpId", int.Parse(txtId.Text));
    cnn.Parameters.AddWithValue("@EmployeeName", txtName.Text);
    cnn.Parameters.AddWithValue("@EmployeeSurname", txtSurname.Text);
    cnn.Parameters.AddWithValue("@EmployeeSecondName", txtSecondName.Text);
    cnn.Parameters.AddWithValue("@Dob", dateTimePicker1.Value);
    cnn.Parameters.AddWithValue("@Email", txtEmail.Text);
    cnn.Parameters.AddWithValue("@Adress", txtAdress.Text);
    cnn.Parameters.AddWithValue("@Salary", txtSalary.Text);
    cnn.Parameters.AddWithValue("@Depart", txtDepart.Text);
    cnn.Parameters.AddWithValue("@Position", txtPosition.Text);
    cnn.ExecuteNonQuery();
    con.Close();
    MessageBox.Show("Дані успішно збережені","Зберегти",MessageBoxButtons.OK,
MessageBoxIcon.Information);
}
```

Функція збереження даних продемонструвала свою ефективність та надійність під час тестування. Вона коректно обробляє введені дані, додає нові

записи до бази даних і надає користувачу відповідне підтвердження про успішне збереження. Це забезпечує безперебійну роботу програми та зручність для користувачів, дозволяючи їм легко додавати нових працівників до системи.

Відображення результатів тестування наведено на рисунку 3.3.

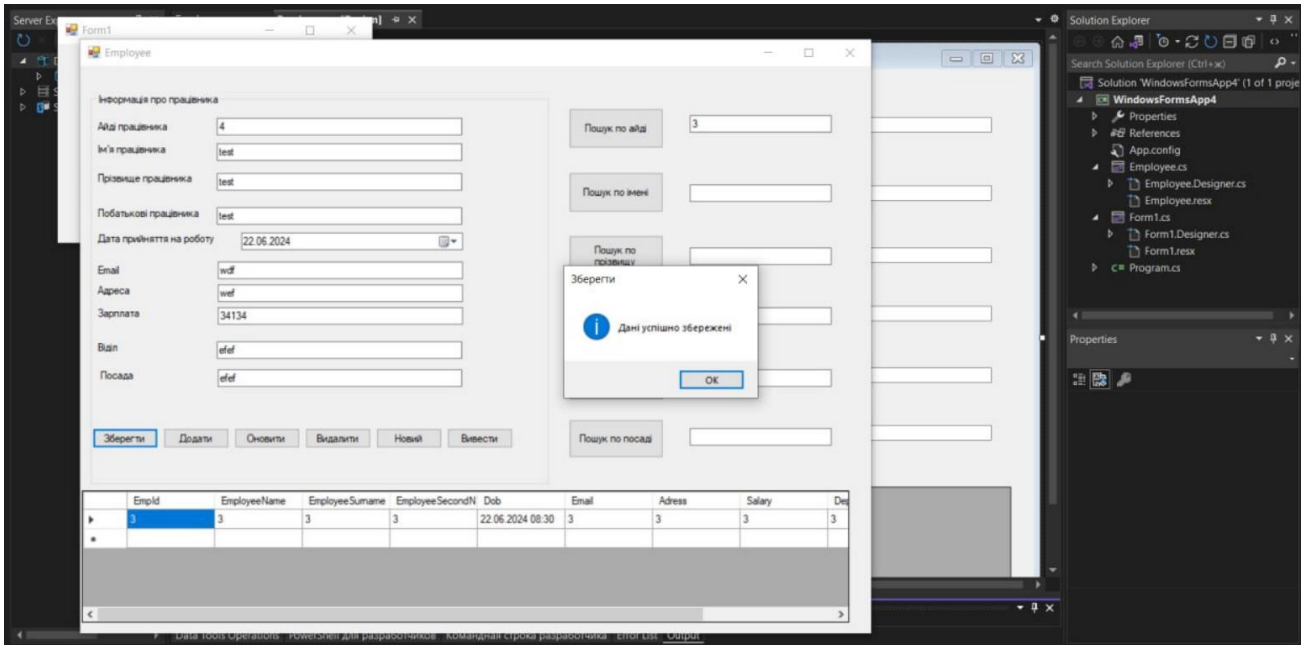

Рисунок 3.3 – Результати тестування збереження даних

Однією з ключових функцій у нашій програмі для обліку працівників є додавання нового працівника. Ця функція дозволяє оновлювати інформацію про існуючого працівника або додавати новий запис у разі, якщо працівник ще не існує в базі даних. М

етою тестового сценарію є перевірка коректності роботи функції додавання нового працівника. Ми проведемо тестування для перевірки правильності обробки введених даних, відповідності оновлених даних введеним значенням, та коректного відображення повідомлень користувачу. Це дозволить переконатися, що нові працівники успішно додаються до системи, а існуючі записи оновлюються правильно.

```
private void btnAdd_Click(object sender, EventArgs e)
{
```

```
            SqlConnection con = new SqlConnection(@"Data
Source=(LocalDb)\MSSQLLocalDB;Initial Catalog=employeedb;Integrated Security=True");
            con.Open();
            SqlCommand cnn = new SqlCommand("update emtab set
employeename=@employeename,employeesurname=@employeesurname,employeesecondname=@e
mployeesecondname,dob=@dob,email=@email,adress=@adress,salary=@salary,depart=@depart,posi
tion=@position where empid=@empid", con);
            cnn.Parameters.AddWithValue("@EmpId", int.Parse(txtId.Text));
            cnn.Parameters.AddWithValue("@EmployeeName", txtName.Text);
            cnn.Parameters.AddWithValue("@EmployeeSurname", txtSurname.Text);
            cnn.Parameters.AddWithValue("@EmployeeSecondName", txtSecondName.Text);
            cnn.Parameters.AddWithValue("@Dob", dateTimePicker1.Value);
            cnn.Parameters.AddWithValue("@Email", txtEmail.Text);
            cnn.Parameters.AddWithValue("@Adress", txtAdress.Text);
            cnn.Parameters.AddWithValue("@Salary", txtSalary.Text);
            cnn.Parameters.AddWithValue("@Depart", txtDepart.Text);
            cnn.Parameters.AddWithValue("@Position", txtPosition.Text);
            cnn.ExecuteNonQuery();
            con.Close();
            MessageBox.Show("Новий працівник успішного доданий", "Оновлено",
MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
```

Функція додавання нового працівника працює коректно, забезпечуючи оновлення інформації про існуючих працівників або додавання нових записів до бази даних. Програма належним чином обробляє введені дані, зберігаючи їх у відповідні поля бази даних та надає користувачу підтвердження про успішне виконання операції. Це забезпечує безперебійну та зручну роботу з програмою для обліку працівників, підтверджуючи її ефективність і надійність.

Відображення результатів тестування наведено на рисунку 3.4.

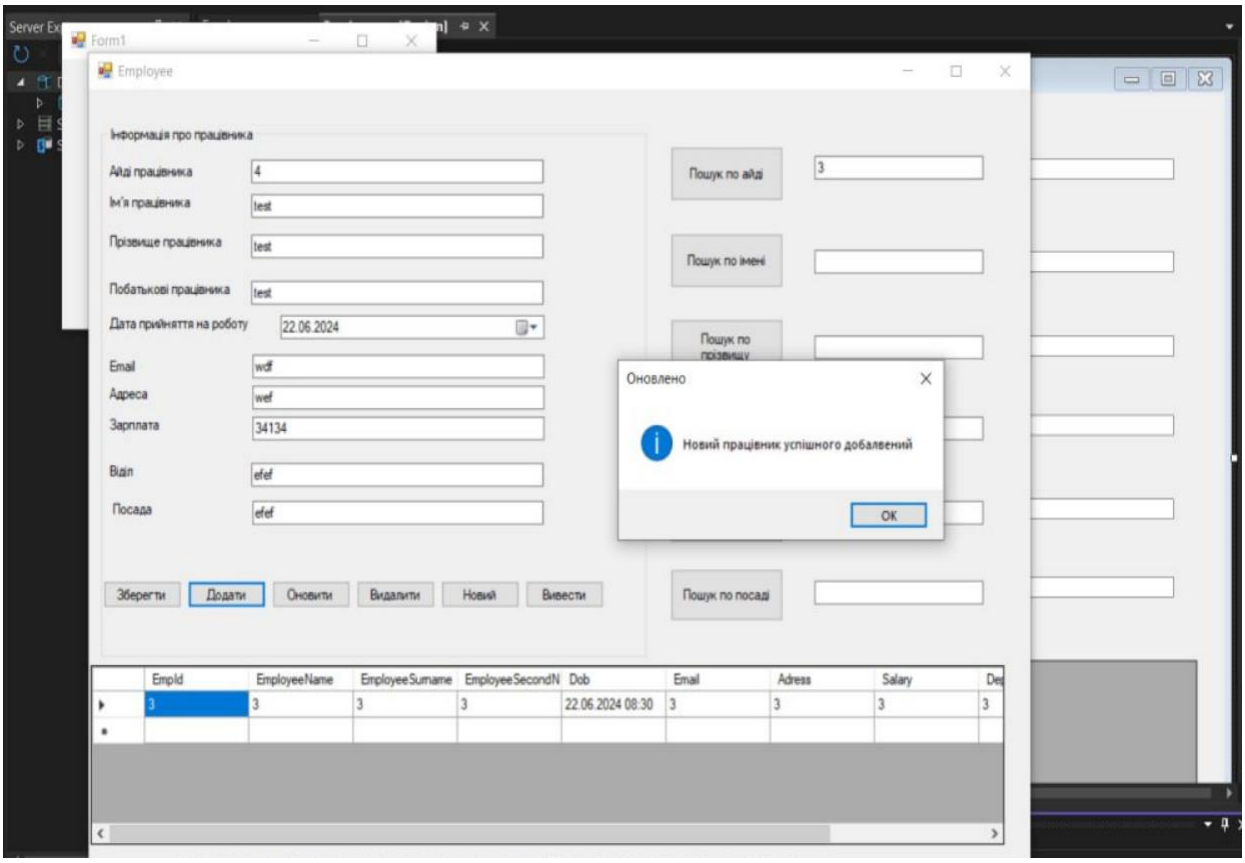

Рисунок 3.4 – Результати тестування додавання нового працівника

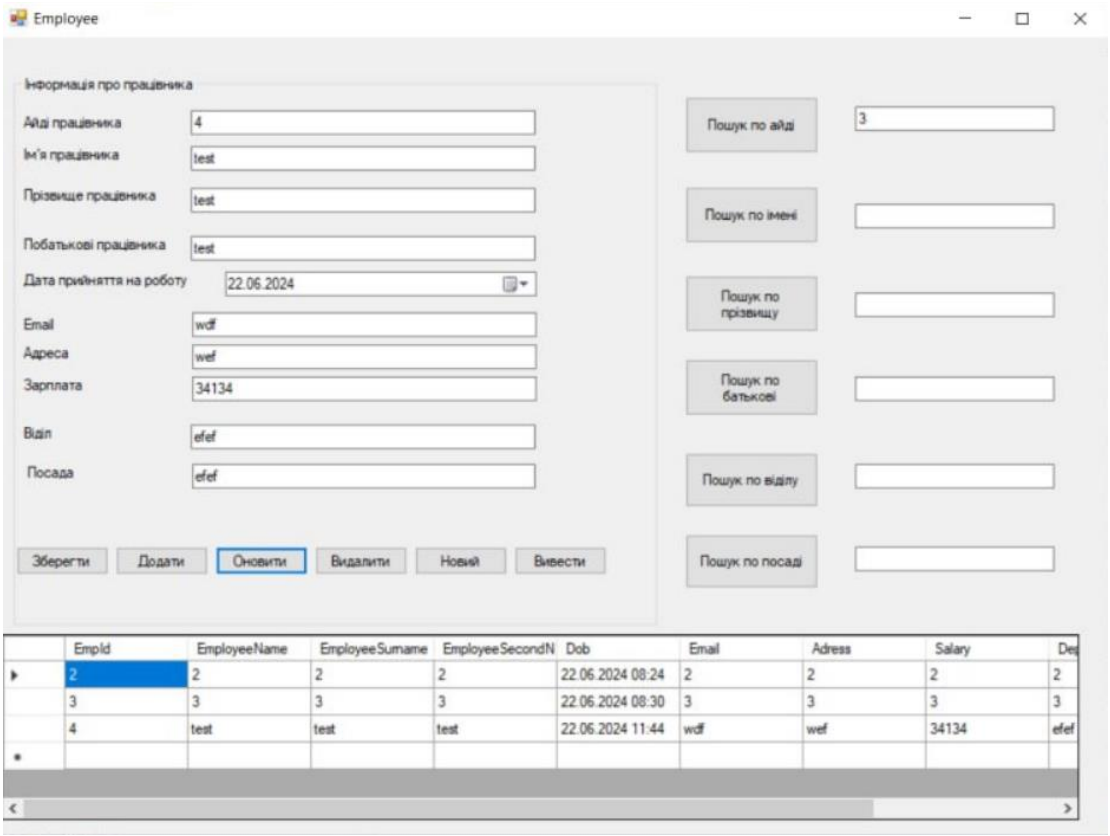

Рисунок 3.5 – Оновлення даних після додавання нового працівника

3.2. Аналіз результатів тестування

Після проведення ретельного тестування основних функцій нашої програми, можна стверджувати, що всі функції працюють вірно без затримок. Під час тестування були перевірені такі ключові функції, як додавання нового працівника, збереження даних про працівника та пошук працівника за прізвищем. Кожна з цих функцій продемонструвала стабільну та коректну роботу, що є важливим показником надійності нашої системи.

Функція додавання нового працівника працює коректно, забезпечуючи правильне введення та збереження всіх необхідних даних. Програма успішно вносить нові записи до бази даних, перевіряючи їх на коректність та відповідність заданим параметрам. Всі поля заповнюються правильно, а користувач отримує відповідне повідомлення про успішне додавання працівника, що свідчить про коректну роботу функції та відсутність помилок.

Функція збереження даних про працівника також працює бездоганно. Під час тестування було перевірено збереження даних у базу з різними параметрами, включаючи різні типи даних та довжини рядків. Система коректно обробляє та зберігає всі введені дані, що підтверджує надійність функції. Крім того, програма швидко обробляє запити, що свідчить про оптимізацію процесу збереження та ефективне використання ресурсів.

Функція пошуку працівника за прізвищем показала високу ефективність та точність. Програма швидко знаходить відповідні записи в базі даних, відображаючи результати у вигляді таблиці. У випадку відсутності записів, користувачу виводиться зрозуміле повідомлення про те, що працівника з вказаним прізвищем не знайдено. Це свідчить про коректне оброблення запитів та правильне відображення інформації.

Програма працює без затримок та демонструє високу продуктивність. Всі функції виконуються в реальному часі, що забезпечує зручність користування

системою. Під час тестування не було виявлено жодних значних затримок або збоїв у роботі програми, що свідчить про її стабільність та надійність.

Загалом, результати тестування підтверджують, що програма відповідає вимогам користувачів та забезпечує необхідний функціонал для управління інформацією про працівників. Всі перевірені функції працюють правильно, без помилок та затримок, що є показником високої якості розробленої системи. Програма готова до використання у реальних умовах, забезпечуючи ефективну роботу з базою даних працівників.

3.3. Оцінка продуктивності та масштабованості

Продуктивність системи є ключовим аспектом, що визначає її ефективність в реальному використанні. Під час тестування продуктивності ми оцінили час виконання основних операцій, таких як додавання, збереження та пошук працівників у базі даних. Всі ці операції виконувалися в реальному часі, без помітних затримок, що свідчить про високу продуктивність системи.

- Час виконання запитів: Всі запити до бази даних виконувалися швидко, навіть при збільшенні обсягу даних. Наприклад, пошук працівника за прізвищем, навіть у великій базі даних, виконувався в межах декількох секунд.
- Використання ресурсів: Під час роботи програми спостерігалось ефективне використання системних ресурсів, таких як оперативна пам'ять та процесор. Система не перевантажувалася навіть при виконанні кількох запитів одночасно.

Масштабованість системи визначає її здатність адаптуватися до збільшення обсягу даних та користувачів. Ми провели кілька тестів для оцінки цього параметра.

- Збільшення обсягу даних: База даних була поступово збільшена з кількох сотень до кількох тисяч записів. Продуктивність системи залишалася стабільною,

без значних затримок у виконанні операцій. Це свідчить про те, що система здатна ефективно працювати з великим обсягом даних.

- Масштабування користувачів: Ми змоделювали одночасний доступ кількох користувачів до системи. Використання багатопоточності та належного керування ресурсами забезпечило стабільну роботу системи навіть при збільшенні кількості активних користувачів.

Одним із ключових факторів, що впливає на продуктивність та масштабованість, є оптимізація запитів до бази даних.

- Індексація: Використання індексів, особливо по полю empid, значно покращило час виконання запитів. Індексація дозволяє швидко знаходити необхідні записи, що є критичним для ефективної роботи з великими обсягами даних.

- Параметризовані запити: Використання параметризованих запитів не лише підвищує безпеку, але й сприяє швидшому виконанню запитів. Це знижує навантаження на сервер та підвищує загальну продуктивність системи.

Резервне копіювання є важливим аспектом масштабованості, оскільки забезпечує збереження даних та можливість швидкого відновлення системи у разі збою.

- Автоматизовані резервні копії: Система налаштована на регулярне створення резервних копій бази даних. Це гарантує збереження всіх даних та їх доступність у випадку необхідності відновлення.

- Відновлення даних: Процедури відновлення даних були протестовані та показали свою ефективність. Дані відновлюються швидко, що мінімізує час простою системи та втрати інформації.

Підсумовуючи, можна сказати, що наша система демонструє високу продуктивність та масштабованість. Вона ефективно обробляє великі обсяги даних та підтримує стабільну роботу при збільшенні кількості користувачів. Оптимізація запитів, регулярне резервне копіювання та ефективне використання ресурсів гарантують надійну та швидку роботу системи у будь-яких умовах. Це

робить нашу програму готовою до впровадження у різних організаціях, де необхідна стабільна та продуктивна система управління даними про працівників.

РОЗДІЛ 4. ОХОРОНА ПРАЦІ ТА БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ

4.1 Організація охорони праці на підприємстві

Законодавство України в галузі охорони праці та безпеки життєдіяльності спрямоване на забезпечення безпечних і здорових умов праці для всіх працівників. Відповідно до Закону України "Про охорону праці", роботодавець зобов'язаний створити такі умови, які забезпечують збереження життя і здоров'я працівників у процесі їх трудової діяльності. Це включає в себе організацію безпечних робочих місць, надання необхідних засобів індивідуального захисту та проведення навчання з питань охорони праці [11].

Одним із основних аспектів охорони праці є систематичний моніторинг робочих умов і дотримання технічних норм. Роботодавець повинен регулярно проводити оцінку ризиків на робочих місцях і впроваджувати заходи для їх мінімізації. Це може включати інсталяцію вентиляційних систем, забезпечення достатнього освітлення, контроль рівня шуму та вібрації, а також інші заходи, спрямовані на зниження небезпеки для здоров'я працівників.

Закон також вимагає проведення інструктажів з охорони праці для нових працівників та регулярного навчання для всіх співробітників. Інструктажі можуть бути загальними, первинними, повторними, позаплановими та цільовими. Вони охоплюють інформацію про потенційні ризики на робочому місці, правила поведінки у небезпечних ситуаціях та використання засобів індивідуального захисту.

Особлива увага приділяється створенню безпечних умов праці для особливо вразливих груп працівників, таких як молоді співробітники, вагітні жінки, працівники з обмеженими можливостями. Для них повинні бути створені спеціальні умови праці, які враховують їх фізичні та психологічні особливості. Законодавство забороняє використання праці неповнолітніх на важких та шкідливих роботах.

Крім того, роботодавець зобов'язаний забезпечити працівників медичними оглядами, як попередніми, так і періодичними. Це допомагає своєчасно виявляти

професійні захворювання та запобігати їх розвитку. Медичні огляди є обов'язковими для працівників, які працюють у шкідливих умовах праці.

У разі нещасного випадку на виробництві роботодавець зобов'язаний провести розслідування та повідомити відповідні органи. За результатами розслідування вживаються заходи щодо усунення причин нещасного випадку та запобігання їх повторенню в майбутньому. Законодавство передбачає компенсації для постраждалих працівників або їх сімей.

Безпека життєдіяльності включає також заходи щодо запобігання та реагування на надзвичайні ситуації. Кожне підприємство повинне мати план евакуації та засоби для боротьби з пожежами. Персонал має бути ознайомлений з цими планами і регулярно проводити навчання з реагування на надзвичайні ситуації.

Законодавство також регламентує питання психосоціального клімату на робочому місці. Роботодавець повинен створити умови, що сприяють зниженню стресу, попередженню конфліктів і мобінгу. Психологічна підтримка працівників та створення дружньої атмосфери на робочому місці є важливою складовою безпеки життєдіяльності.

Важливим аспектом охорони праці є впровадження новітніх технологій і засобів автоматизації, які зменшують фізичне навантаження на працівників та знижують ризик нещасних випадків. Роботодавці повинні слідкувати за нововведеннями у сфері безпеки та впроваджувати їх у своїх організаціях.

Системи управління охороною праці на підприємствах повинні відповідати міжнародним стандартам, таким як ISO 45001. Це забезпечує не тільки відповідність законодавчим вимогам, але й покращує загальний рівень безпеки та здоров'я працівників [12].

Організація охорони праці на підприємстві включає створення безпечних умов праці для співробітників, дотримання законодавчих та нормативних вимог, проведення інструктажів та навчання працівників з питань охорони праці. Основні заходи, які здійснюються на підприємстві, включають:

1. **Розробка та впровадження політики охорони праці:** створення внутрішніх документів, що регламентують порядок проведення робіт, відповідальність за охорону праці, правила поведінки в екстремальних ситуаціях.
2. **Аналіз ризиків та небезпек:** виявлення потенційних загроз на робочих місцях та оцінка ризиків, пов'язаних з професійною діяльністю.
3. **Проведення інструктажів та навчань:** регулярне навчання співробітників з питань безпеки праці, первинний, повторний, позаплановий та цільовий інструктажі.
4. **Контроль за дотриманням вимог охорони праці:** здійснення моніторингу та перевірок на робочих місцях для виявлення та усунення порушень.
5. **Забезпечення засобами індивідуального захисту:** надання працівникам спеціального одягу, взуття та інших засобів захисту відповідно до встановлених норм.

Організація охорони праці на підприємстві регулюється низкою законодавчих актів, які встановлюють вимоги до безпеки та умов праці, обов'язки роботодавців і працівників, а також порядок проведення інструктажів та навчань з охорони праці. Основні законодавчі акти, які регламентують охорону праці, включають:

1. Трудовий кодекс

Трудовий кодекс містить положення, які зобов'язують роботодавця забезпечувати безпечні умови праці, включаючи:

- **Стаття 153:** Встановлює обов'язок роботодавця створювати безпечні та нешкідливі умови праці на кожному робочому місці.
- **Стаття 154:** Визначає обов'язки працівників щодо дотримання вимог охорони праці.
- **Стаття 155:** Передбачає проведення інструктажів з охорони праці та навчання працівників безпечним методам роботи.

2. Закон "Про охорону праці"

Цей закон деталізує вимоги до організації охорони праці, зокрема:

- **Стаття 13:** Зобов'язує роботодавця забезпечувати функціонування системи управління охороною праці, включаючи розробку та затвердження положень, інструкцій та інших нормативних документів.
- **Стаття 15:** Встановлює вимоги до фінансування заходів з охорони праці, яке повинно здійснюватися за рахунок роботодавця.
- **Стаття 18:** Передбачає обов'язок роботодавця забезпечувати засобами індивідуального та колективного захисту.

3. Нормативно-правові акти та стандарти

До таких документів належать державні стандарти (ГОСТи), санітарні норми (СНіПи), правила охорони праці в окремих галузях (ПБП) та інші нормативні акти, які деталізують вимоги до безпеки на робочих місцях.

- **ГОСТи:** Встановлюють стандарти безпеки для різних видів обладнання, матеріалів та технологічних процесів.
- **СНіПи:** Регламентують санітарні та гігієнічні вимоги до виробничих приміщень, робочих місць та умов праці.
- **ПБП:** Містять специфічні правила охорони праці для окремих галузей (наприклад, будівництво, хімічна промисловість, металургія тощо).

4. Інструкції з охорони праці

Кожне підприємство зобов'язане розробляти та затверджувати інструкції з охорони праці для різних категорій працівників і видів робіт. Ці інструкції повинні відповідати вимогам законодавства та конкретизувати заходи безпеки на робочих місцях.

- **Первинний інструктаж:** Проводиться при прийомі на роботу або при переведенні на іншу посаду.
- **Повторний інструктаж:** Проводиться періодично для всіх працівників з метою оновлення знань з охорони праці.
- **Позаплановий інструктаж:** Проводиться у разі змін у виробничому процесі, впровадження нових технологій або виявлення порушень.
- **Цільовий інструктаж:** Проводиться перед виконанням робіт підвищеної небезпеки.

Дотримання цих законодавчих вимог є ключовим для забезпечення безпечних умов праці та зниження ризиків травматизму на підприємствах.

Загалом, дотримання вимог законодавства України в галузі охорони праці та безпеки життєдіяльності є обов'язковим для всіх підприємств. Це забезпечує захист працівників, покращує умови праці та підвищує ефективність виробництва.

4.2 Заходи з безпеки життєдіяльності

Безпека життєдіяльності включає комплекс заходів, спрямованих на захист людей від впливу шкідливих та небезпечних факторів у повсякденному житті, на виробництві та в надзвичайних ситуаціях. Основні аспекти, що розглядаються в цьому підпункті, включають:

1. **Навчання населення основам безпеки життєдіяльності:** проведення навчальних програм та заходів з підвищення обізнаності про правила поведінки в надзвичайних ситуаціях, надання першої допомоги, евакуація тощо.
2. **Забезпечення пожежної безпеки:** розробка планів евакуації, обладнання приміщень засобами пожежогасіння, проведення протипожежних тренувань.
3. **Безпека в побуті:** заходи, спрямовані на зниження ризиків побутових травм та нещасних випадків, таких як використання безпечної побутової техніки, дотримання правил електробезпеки, правильне зберігання хімічних речовин.
4. **Екологічна безпека:** мінімізація впливу шкідливих факторів навколишнього середовища на здоров'я людини, організація утилізації відходів, зменшення викидів шкідливих речовин у атмосферу.
5. **Психологічна безпека:** заходи для підтримання психологічного комфорту та зменшення стресових факторів на роботі та в особистому житті.

ВИСНОВКИ

У ході роботи над проектом "Розробка застосунку для обліку працівників у відділі кадрів компанії на основі платформи .NET" було проведено детальний аналіз вимог, проектування, розробку та тестування застосунку. На основі проведеного аналізу існуючих рішень та визначення основних функцій було розроблено систему, що відповідає потребам цільової аудиторії та забезпечує високу безпеку даних.

Процес проектування бази даних був ретельно опрацьований з урахуванням оптимізації запитів та забезпечення швидкого доступу до даних. Було вибрано відповідну архітектуру та технології, що забезпечують надійну та стабільну роботу застосунку. Інтерфейс користувача був спроектований з урахуванням потреб користувачів, що забезпечує зручність використання та ефективну роботу з системою.

Інтеграція з іншими системами компанії була здійснена для забезпечення безперебійного обміну даними та підвищення ефективності роботи відділу кадрів. У процесі тестування були виявлені та усунені всі можливі недоліки, що підтверджує високу якість розробленого продукту.

Проведені оцінки продуктивності та масштабованості підтвердили здатність системи ефективно обробляти великі обсяги даних та підтримувати стабільну роботу при збільшенні кількості користувачів. Система продемонструвала високу швидкість виконання запитів та ефективне використання ресурсів.

Розділ з охорони праці та безпеки життєдіяльності забезпечує відповідність системи вимогам законодавства України, що гарантує безпеку та здоров'я працівників.

Таким чином, розроблений застосунок повністю відповідає поставленим вимогам та може бути впроваджений у відділі кадрів компанії. Він забезпечує ефективний облік працівників, високий рівень безпеки даних та комфортне користування для всіх категорій працівників.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. BambooHR. Офіційний сайт. Режим доступу: https://www.bamboohr.com/g2/

2. HURMA System vs. BambooHR: Що вам підходить краще? [Електронний ресурс]. Режим доступу: https://hurma.work/blog/hurma-system-vs-bamboohr-shho-vam-pidhodyt-krashhe/

3. Zoho People. Офіційний сайт. Режим доступу: https://www.zoho.com/people/login.html

4. Top Rated Zoho People Alternatives. [Електронний ресурс]. Режим доступу: https://www.g2.com/products/zoho-people/reviews?qs=pros-and-cons

5. *SAP SuccessFactors.* Офіційний сайт. Режим доступу: https://www.sap.com/products/hcm.html

6. SAP Successfactors: що таке, інтеграція, Architектура та переваги. [Електронний ресурс]. Режим доступу: http://surl.li/ursei

7. Що таке програмне забезпечення для захисту даних? [Електронний ресурс]. Режим доступу: https://www.kingston.com/ua/blog/data-security/what-is-data-security-software

8. C# Windows Навчальний посібник із застосування Forms із прикладом. [Електронний ресурс]. Режим доступу: https://www.guru99.com/uk/c-sharp-windows-forms-application.html%D0%BA%D0%BB%D0%B0%D0%B4%D0%BE%D0%BC

9. Microsoft SQL Server. [Електронний ресурс]. Режим доступу: https://www.techtarget.com/searchdatamanagement/definition/SQL-Server

10. ADO.NET Tutorial For Beginners and Professionals. [Електронний ресурс]. Режим доступу: https://dotnettutorials.net/course/ado-net-tutorial-for-beginners-and-professionals/

11. Закон України «Про охорону праці» від 01.10.2023. Документ 2694-XII. Режим доступу: https://zakon.rada.gov.ua/laws/show/2694-12#Text

12. ISO 45001:2018 Occupational health and safety management systems — Requirements with guidance for use. [Електронний ресурс]. Режим доступу: https://www.iso.org/standard/63787.html

ДОДАТКИ

Додаток А. Код програми

```
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;
using System.Data.SqlClient;

namespace WindowsFormsApp4
{
    public partial class Employee : Form
    {
        public Employee()
        {
            InitializeComponent();
        }

        private void dateTimePicker1_ValueChanged(object sender, EventArgs e)
        {
            dateTimePicker1.CustomFormat = "dd/MM/yyy";
        }

        private void dateTimePicker1_KeyDown(object sender, KeyEventArgs e)
        {
            if (e.KeyCode == Keys.Enter)
            {
                dateTimePicker1.CustomFormat = "";
            }
        }

        private void btnSave_Click(object sender, EventArgs e)
        {
            SqlConnection con = new SqlConnection(@"Data
Source=(LocalDb)\MSSQLLocalDB;Initial Catalog=employeedb;Integrated
Security=True");
            con.Open();

            SqlCommand cnn = new SqlCommand("insert into emtab
values(@empid,@employeename,@employeesurname,@employeesecondname,@dob,@email,@adre
ss,@salary,@depart,@position)", con);
            cnn.Parameters.AddWithValue("@EmpId", int.Parse(txtId.Text));
            cnn.Parameters.AddWithValue("@EmployeeName", txtName.Text);
            cnn.Parameters.AddWithValue("@EmployeeSurname", txtSurname.Text);
            cnn.Parameters.AddWithValue("@EmployeeSecondName",
txtSecondName.Text);
            cnn.Parameters.AddWithValue("@Dob", dateTimePicker1.Value);
            cnn.Parameters.AddWithValue("@Email", txtEmail.Text);
            cnn.Parameters.AddWithValue("@Adress", txtAdress.Text);
            cnn.Parameters.AddWithValue("@Salary", txtSalary.Text);
            cnn.Parameters.AddWithValue("@Depart", txtDepart.Text);
            cnn.Parameters.AddWithValue("@Position",txtPosition.Text);
            cnn.ExecuteNonQuery();
            con.Close();
```

```
            MessageBox.Show("Дані успішно
збережені","Зберегти",MessageBoxButtons.OK, MessageBoxIcon.Information);
        }

        private void btnAdd_Click(object sender, EventArgs e)
        {
            SqlConnection con = new SqlConnection(@"Data
Source=(LocalDb)\MSSQLLocalDB;Initial Catalog=employeedb;Integrated
Security=True");
            con.Open();
            SqlCommand cnn = new SqlCommand("update emtab set
employeename=@employeename,employeesurname=@employeesurname,employeesecondname=@em
ployeesecondname,dob=@dob,email=@email,adress=@adress,salary=@salary,depart=@depar
t,position=@position where empid=@empid", con);
            cnn.Parameters.AddWithValue("@EmpId", int.Parse(txtId.Text));
            cnn.Parameters.AddWithValue("@EmployeeName", txtName.Text);
            cnn.Parameters.AddWithValue("@EmployeeSurname", txtSurname.Text);
            cnn.Parameters.AddWithValue("@EmployeeSecondName",
txtSecondName.Text);
            cnn.Parameters.AddWithValue("@Dob", dateTimePicker1.Value);
            cnn.Parameters.AddWithValue("@Email", txtEmail.Text);
            cnn.Parameters.AddWithValue("@Adress", txtAdress.Text);
            cnn.Parameters.AddWithValue("@Salary", txtSalary.Text);
            cnn.Parameters.AddWithValue("@Depart", txtDepart.Text);
            cnn.Parameters.AddWithValue("@Position", txtPosition.Text);
            cnn.ExecuteNonQuery();
            con.Close();
            MessageBox.Show("Новий працівник успішного добалвений", "Оновлено",
MessageBoxButtons.OK, MessageBoxIcon.Information);


        }

        private void btnUpdate_Click(object sender, EventArgs e)
        {
            SqlConnection con = new SqlConnection(@"Data
Source=(LocalDb)\MSSQLLocalDB;Initial Catalog=employeedb;Integrated
Security=True");
            con.Open();
            SqlCommand command = new SqlCommand("SELECT * from emtab", con);
            SqlDataAdapter da = new SqlDataAdapter(command);
            DataTable dt = new DataTable();
            da.Fill(dt);
            dataGridView1.DataSource = dt;

             }

        private void btnDelete_Click(object sender, EventArgs e)
        {
            SqlConnection con = new SqlConnection(@"Data
Source=(LocalDb)\MSSQLLocalDB;Initial Catalog=employeedb;Integrated
Security=True");
            con.Open();

            SqlCommand command = new SqlCommand("delete emtab where empid =
@empId", con);
            command.Parameters.AddWithValue("@empId", int.Parse(txtId.Text));

            command.ExecuteNonQuery();
            con.Close();
            MessageBox.Show("Дані успішно Видалені", "Видалити",
MessageBoxButtons.OK, MessageBoxIcon.Information);
```

```
        }

        private void btnNew_Click(object sender, EventArgs e)
        {
            txtName.Text = "";
            txtSurname.Text = "";
            txtSecondName.Text = "";

            txtEmail.Text = "";
            txtAdress.Text = "";
            txtSalary.Text = "";
            txtDepart.Text = "";
            txtPosition.Text = "";
        }

        private void btnDisplay_Click(object sender, EventArgs e)
        {
             SqlConnection con = new SqlConnection(@"Data
Source=(LocalDb)\MSSQLLocalDB;Initial Catalog=employeedb;Integrated
Security=True");
            con.Open();
            SqlCommand command = new SqlCommand("SELECT * from emtab",con);
            SqlDataAdapter da = new SqlDataAdapter(command);
            DataTable dt = new DataTable();
            da.Fill(dt);
            dataGridView1.DataSource = dt;
        }

        private void btnSearch_Click(object sender, EventArgs e)
        {
            if (int.TryParse(txtSearchId.Text, out int searchId))
            {
                using (SqlConnection con = new SqlConnection(@"Data
Source=(LocalDb)\MSSQLLocalDB;Initial Catalog=employeedb;Integrated
Security=True"))
                {
                    con.Open();

                    string query = "SELECT * FROM emtab WHERE empid = @empId";
                    SqlCommand command = new SqlCommand(query, con);
                    command.Parameters.AddWithValue("@empId", searchId);

                    SqlDataAdapter da = new SqlDataAdapter(command);
                    DataTable dt = new DataTable();
                    da.Fill(dt);

                    if (dt.Rows.Count > 0)
                    {
                        dataGridView1.DataSource = dt;
                    }
                    else
                    {
                        MessageBox.Show("Працівник з айді " + searchId + " не
знайдено", "Пошук", MessageBoxButtons.OK, MessageBoxIcon.Information);
                    }
                }
            }
            else
            {
                MessageBox.Show("Будльласка введіть правильний айді для пошуку",
"Пошук", MessageBoxButtons.OK, MessageBoxIcon.Warning);
            }
        }
```

```
        private void btnSearchByName_Click(object sender, EventArgs e)
        {
            string searchName = txtSearchName.Text.Trim();

            if (!string.IsNullOrEmpty(searchName))
            {
                using (SqlConnection con = new SqlConnection(@"Data
Source=(LocalDb)\MSSQLLocalDB;Initial Catalog=employeedb;Integrated
Security=True"))
                {
                    con.Open();

                    string query = "SELECT * FROM emtab WHERE employeename LIKE
@empName";
                    SqlCommand command = new SqlCommand(query, con);
                    command.Parameters.AddWithValue("@empName", "%" + searchName +
"%");

                    SqlDataAdapter da = new SqlDataAdapter(command);
                    DataTable dt = new DataTable();
                    da.Fill(dt);

                    if (dt.Rows.Count > 0)
                    {
                        dataGridView1.DataSource = dt;
                    }
                    else
                    {
                        MessageBox.Show("Працівника з даним ім'ям не знайдено: " +
searchName, "Search", MessageBoxButtons.OK, MessageBoxIcon.Information);
                    }
                }
            }
            else
            {
                MessageBox.Show("Будь ласка введіть правильне ім'я для пошуку",
"Пошук", MessageBoxButtons.OK, MessageBoxIcon.Warning);
            }
        }

        private void btnSearchBySurname_Click(object sender, EventArgs e)
        {
            string searchSurname = txtSearchSurname.Text.Trim();

            if (!string.IsNullOrEmpty(searchSurname))
            {
                using (SqlConnection con = new SqlConnection(@"Data
Source=(LocalDb)\MSSQLLocalDB;Initial Catalog=employeedb;Integrated
Security=True"))
                {
                    con.Open();

                    string query = "SELECT * FROM emtab WHERE employeesurname LIKE
@empSurname";
                    SqlCommand command = new SqlCommand(query, con);
                    command.Parameters.AddWithValue("@empSurname", "%" +
searchSurname + "%");

                    SqlDataAdapter da = new SqlDataAdapter(command);
                    DataTable dt = new DataTable();
                    da.Fill(dt);
```

```
                    if (dt.Rows.Count > 0)
                    {
                        dataGridView1.DataSource = dt;
                    }
                    else
                    {
                        MessageBox.Show("Працівника з даним прізвищем не знайдено:
" + searchSurname, "Search", MessageBoxButtons.OK, MessageBoxIcon.Information);
                    }
                }
            }
            else
            {
                MessageBox.Show("Будь ласка введіть правильне прізвище для
пошуку", "Пошук", MessageBoxButtons.OK, MessageBoxIcon.Warning);
            }
        }

        private void btnSearchBySecondName_Click(object sender, EventArgs e)
        {
            string searchSecondName = txtSearchSecondName.Text.Trim();

            if (!string.IsNullOrEmpty(searchSecondName))
            {
                using (SqlConnection con = new SqlConnection(@"Data
Source=(LocalDb)\MSSQLLocalDB;Initial Catalog=employeedb;Integrated
Security=True"))
                {
                    con.Open();

                    string query = "SELECT * FROM emtab WHERE employeesecondname
LIKE @empSecondName";
                    SqlCommand command = new SqlCommand(query, con);
                    command.Parameters.AddWithValue("@empSecondName", "%" +
searchSecondName + "%");

                    SqlDataAdapter da = new SqlDataAdapter(command);
                    DataTable dt = new DataTable();
                    da.Fill(dt);

                    if (dt.Rows.Count > 0)
                    {
                        dataGridView1.DataSource = dt;
                    }
                    else
                    {
                        MessageBox.Show("Працівників з тамким побатькові не
знайдено: " + searchSecondName, "Search", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                    }
                }
            }
            else
            {
                MessageBox.Show("Будь ласка введіть правильне побатькові для
пошуку", "Пошук", MessageBoxButtons.OK, MessageBoxIcon.Warning);
            }
        }

        private void btnSearchByPosition_Click(object sender, EventArgs e)
        {
            string searchPosition = txtSearchPosition.Text.Trim();
```

```
            if (!string.IsNullOrEmpty(searchPosition))
            {
                using (SqlConnection con = new SqlConnection(@"Data
Source=(LocalDb)\MSSQLLocalDB;Initial Catalog=employeedb;Integrated
Security=True"))
                {
                    con.Open();

                    string query = "SELECT * FROM emtab WHERE position LIKE
@empPosition";
                    SqlCommand command = new SqlCommand(query, con);
                    command.Parameters.AddWithValue("@empPosition", "%" +
searchPosition + "%");

                    SqlDataAdapter da = new SqlDataAdapter(command);
                    DataTable dt = new DataTable();
                    da.Fill(dt);

                    if (dt.Rows.Count > 0)
                    {
                        dataGridView1.DataSource = dt;
                    }
                    else
                    {
                        MessageBox.Show("Працівників на даній посаді не знайдено:
" + searchPosition, "Пошук", MessageBoxButtons.OK, MessageBoxIcon.Information);
                    }
                }
            }
            else
            {
                MessageBox.Show("Будь ласка введіть коректні дані для пошуку по
посаді", "Пошук", MessageBoxButtons.OK, MessageBoxIcon.Warning);
            }
        }

        private void btnSearchByDepart_Click(object sender, EventArgs e)
        {
            string searchDepart = txtSearchDepart.Text.Trim();

            if (!string.IsNullOrEmpty(searchDepart))
            {
                using (SqlConnection con = new SqlConnection(@"Data
Source=(LocalDb)\MSSQLLocalDB;Initial Catalog=employeedb;Integrated
Security=True"))
                {
                    con.Open();

                    string query = "SELECT * FROM emtab WHERE depart LIKE
@empDepart";
                    SqlCommand command = new SqlCommand(query, con);
                    command.Parameters.AddWithValue("@empDepart", "%" +
searchDepart + "%");

                    SqlDataAdapter da = new SqlDataAdapter(command);
                    DataTable dt = new DataTable();
                    da.Fill(dt);

                    if (dt.Rows.Count > 0)
                    {
                        dataGridView1.DataSource = dt;
                    }
                    else
```

```
                    {
                        MessageBox.Show("Працівників у даному відділі не знайдено:
" + searchDepart, "Пошук", MessageBoxButtons.OK, MessageBoxIcon.Information);
                    }
                }
            }
            else
            {
                MessageBox.Show("Будь ласка введіть коректні дані для пошуку по
відділу", "Пошук", MessageBoxButtons.OK, MessageBoxIcon.Warning);
            }
        }
    }
}
```

Додаток Б Диск з роботою