Міністерство освіти і науки України
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

РОЗРОБКА СИСТЕМИ ОБЛІКУ ПЕРСОНАЛУ ТА ЗАДАЧ ВИКОРИСТОВУЮЧИ C# ТА РОЛЬОВИЙ КОНТРОЛЬ ДОСТУПУ

DEVELOPMENT OF A PERSONNEL AND TASK ACCOUNTING SYSTEM USING C# AND ROLE-BASED ACCESS CONTROL

спеціальність 121 «Інженерія програмного забезпечення»
освітня програма «Інженерія програмного забезпечення»

Виконав: здобувач вищої освіти
групи ІПЗ-44
Жовнір Максим Вікторович

Керівник:
Цінделіані Давид Мурадович

Кваліфікаційну роботу
допущено до захисту
«___» _____ 20__ р.
Гарант освітньої програми:
к.т.н., доцент
Ліщина Наталія Миколаївна
_____

Луцьк – 2026 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *інженерії програмного забезпечення*
Ступінь вищої освіти *бакалавр*
Галузь знань: *12 «Інформаційні технології»*
Спеціальність: *121 «Інженерія програмного забезпечення»*
Освітня програма: «*Інженерія програмного забезпечення»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____
«___» _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Жовніру Максиму Вікторовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи «Р*озробка системи обліку персоналу та задач використовуючи C# та рольовий контроль доступу»*
Керівник роботи: асистент Цінделіані Д. М.
затверджені наказом закладу вищої освіти від *«31» грудня 2025 р.* *№ 541/01-02*
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи бакалавра
«29» травня 2026 р.
3. Вихідні дані до роботи: *предметна область управління персоналом і задачами, технології розробки програмного забезпечення (C#, .NET), засоби роботи з базами даних (SQLite, Entity Framework Core), вимоги до інформаційних систем та забезпечення інформаційної безпеки.*
4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити): *аналіз сучасного стану проблем автоматизації обліку персоналу та управління задачами, постановка завдання, формування вимог до розроблюваної інформаційної системи, обґрунтування вибору технологій та засобів розробки, практична реалізація об'єкта проєктування, розробка структури бази даних, обґрунтування принципів захисту інформаційно-комп'ютерної системи, тестування та налагодження програмного забезпечення*.
5. Перелік графічного матеріалу: *13 рисунків, 9 лістингів, 2 таблиці.*
_____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз предметної області	*Цінделіані Д. М.*		
Специфікація вимог до розробленої системи	*Цінделіані Д. М.*		
Розробка об’єкта проектування	*Цінделіані Д. М.*		
Нормоконтроль	*Суринович О. М.*		
Гарант ОП	*Ліщина Н. М.*		
Показник запозичень тексту	***___%***		
Академічна доброчесність	*Цінделіані Д. М.*		

7. Дата видачі завдання *«3» лютого 2026 р.*

№	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1	Огляд літературних джерел по темі кваліфікаційної роботи бакалавра	до 14.02.2026 р.	
2	Аналіз проблеми розробки та впровадження об’єкту проектування	до 03.03.2026 р.	
3	Обґрунтування вибору шляхів, технологій і засобів вирішення поставленого завдання	до 14.03.2026 р.	
4	Розробка функціонально-структурної схеми роботи об’єкта проектування та проектування бази даних	до 14.04.2026 р.	
5	Практична реалізація об’єкта проектування та розробка бази даних	до 20.04.2026 р.	
6	Тестування та налагодження об’єкта проектування	до 04.05.2026 р.	
7	Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедру	до 29.05.2026 р.	

Здобувач вищої освіти _____ Максим ЖОВНІР

Керівник кваліфікаційної роботи _____ Давид ЦІНДЕЛІАНІ

АНОТАЦІЯ

Жовнір М. В. Розробка системи обліку персоналу та задач використовуючи C# та рольовий контроль доступу. Рукопис. Кваліфікаційна робота бакалавра ОП «Інженерія програмного забезпечення» спеціальності «Інженерія програмного забезпечення». Луцький національний технічний університет. Луцьк, 2026.

Кваліфікаційна робота бакалавра складається зі вступу, трьох розділів, висновків, списку використаних джерел.

У першому розділі проведено дослідження предметної області, пов’язаної з автоматизацією процесів обліку персоналу та управління задачами, проаналізовано існуючі програмні рішення та виявлено їхні сильні і слабкі сторони, а також обґрунтовано необхідність створення власної системи. У другому розділі сформовано вимоги до програмного продукту, здійснено проєктування системи із використанням UML-нотації та обґрунтовано вибір інструментальних засобів, методів і технологій розробки. У третьому розділі описано процес реалізації інформаційної системи, розроблено структуру бази даних, розглянуто питання забезпечення захисту інформації, а також наведено результати тестування і налагодження програмного забезпечення. У висновках підсумовано отримані результати, підтверджено досягнення мети роботи та повне виконання поставлених завдань.

Ключові слова: інформаційна система, облік персоналу, управління задачами, рольовий контроль доступу, база даних, C#, .NET, Windows Forms, SQLite, Entity Framework Core.

ABSTRACT

Zhovnir M. V. Development of a personnel and task accounting system using C# and role-based access control. Manuscript. Qualification work for bachelor’s degree in Software Engineering, specialty "Software Engineering". Lutsk National Technical University. Lutsk, 2026.

The bachelor’s qualification work consists of an introduction, three chapters, conclusions, a list of references.

In the first chapter, a study of the problem domain related to the automation of personnel accounting and task management processes was conducted. Existing software solutions were analyzed, their strengths and weaknesses were identified, and the necessity of developing a new system was substantiated. In the second chapter, the requirements for the software product were defined, the system was designed using UML notation, and the choice of tools, methods, and development technologies was justified. The third chapter describes the implementation of the information system, including the development of the database structure, consideration of information security mechanisms, as well as the results of software testing and debugging. The conclusions summarize the results obtained, confirm the achievement of the research objective, and verify the complete fulfillment of all defined tasks.

Keywords: information system, personnel management, task management, role-based access control, database, C#, .NET, Windows Forms, SQLite, Entity Framework Core.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасних умовах цифрової трансформації економіки та активного впровадження інформаційних технологій у всі сфери діяльності підприємств особливої ваги набуває автоматизація процесів управління персоналом і контролю виконання робочих задач. Ефективність функціонування організації значною мірою залежить від рівня організації праці, своєчасності виконання завдань та можливості оперативного контролю за діяльністю працівників [1].

На практиці у багатьох малих та середніх підприємствах процеси обліку персоналу та управління задачами залишаються частково або повністю неавтоматизованими. Для цього використовуються електронні таблиці, текстові документи або взагалі паперові носії. Такий підхід призводить до низки проблем, серед яких: складність відстеження актуального стану задач, відсутність централізованого зберігання інформації, висока ймовірність втрати даних, а також ускладнення контролю відповідальності за виконання завдань.

Існуючі програмні рішення, такі як системи управління проєктами або CRM-системи, часто є надмірно складними для впровадження у невеликих організаціях або потребують значних фінансових витрат на ліцензування та підтримку. Крім того, вони можуть містити функціональність, яка не відповідає конкретним потребам користувачів, що ускладнює їх використання у повсякденній діяльності [2-7].

Особливо актуальною є проблема забезпечення контролю доступу до інформації. У межах організації різні категорії користувачів мають різні права та обов’язки, що потребує впровадження механізмів розмежування доступу до функцій системи. Відсутність таких механізмів може призвести до несанкціонованого доступу до даних, їх зміни або видалення.

Таким чином, виникає необхідність у розробці спеціалізованої інформаційної системи, яка б забезпечувала зручний облік персоналу, ефективне управління задачами, контроль їх виконання, а також реалізацію рольового

контролю доступу. Використання сучасних технологій, зокрема платформи .NET та мови програмування C#, дозволяє створити надійний, масштабований та зручний у використанні програмний продукт [8-9].

Отже, тема кваліфікаційної роботи, присвячена розробці системи обліку персоналу та задач із використанням рольового контролю доступу, є актуальною, оскільки спрямована на вирішення практичних задач підвищення ефективності управління організацією та забезпечення безпеки інформаційних ресурсів.

Метою кваліфікаційної роботи є розробка інформаційної системи обліку персоналу та управління задачами з використанням мови програмування C# та реалізацією рольового контролю доступу.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– проаналізувати сучасні підходи до автоматизації управління персоналом;

– визначити вимоги до програмної системи;

– обґрунтувати вибір технологій розробки;

– спроєктувати структуру системи та базу даних;

– реалізувати програмний продукт;

– забезпечити механізми захисту інформації;

– провести тестування та налагодження системи.

Об’єктом дослідження є процес управління персоналом та робочими задачами в організації.

Предметом дослідження є методи та засоби розробки інформаційної системи з використанням технологій платформи .NET.

Практичне значення роботи полягає у створенні програмного продукту, який може бути використаний у малих та середніх організаціях для підвищення ефективності управління персоналом та контролю виконання задач.

РОЗДІЛ 1
АНАЛІЗ МЕТОДІВ РОЗРОБКИ ІНФОРМАЦІЙНИХ СИСТЕМ ТА ПОСТАНОВКА ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

1.1 Аналіз сучасного стану проблеми

У сучасних умовах розвитку інформаційних технологій автоматизація процесів управління персоналом та контролю виконання задач є одним із ключових напрямів підвищення ефективності діяльності організацій. Згідно з сучасними підходами до управління бізнес-процесами, важливим є використання інформаційних систем, що забезпечують централізоване зберігання даних, контроль виконання завдань та підтримку прийняття управлінських рішень.

Аналіз науково-технічної літератури показує, що існує значна кількість підходів до автоматизації управління персоналом та задачами. Зокрема, досліджуються системи управління проєктами, корпоративні інформаційні системи та CRM-рішення, які дозволяють організувати робочі процеси, здійснювати планування та контроль діяльності працівників. У сучасних дослідженнях особлива увага приділяється використанню інформаційних систем для підвищення продуктивності праці, оптимізації ресурсів та забезпечення прозорості управління [1-2].

Серед існуючих програмних рішень можна виділити такі системи-аналоги:

– системи управління проєктами (Jira, Trello, Asana), які дозволяють створювати та відстежувати задачі, організовувати роботу команд, використовувати гнучкі методології управління проєктами [3-5];

– CRM-системи (Salesforce, HubSpot, Zoho CRM), що забезпечують управління взаємодією з клієнтами та включають функціонал постановки задач і контролю їх виконання [6-8];

– корпоративні системи обліку персоналу, які дозволяють вести кадровий облік, але часто не мають гнучких механізмів управління задачами.

Аналіз зазначених систем дозволяє виділити їх основні переваги:

– широкий функціонал для управління задачами та проєктами;

– можливість роботи в команді та спільного доступу до інформації;

– інтеграція з іншими сервісами та системами;

– підтримка різних ролей користувачів.

Разом з тим, існуючі рішення мають ряд недоліків: надмірна складність інтерфейсу для невеликих організацій; необхідність постійного підключення до Інтернету (для веб-систем); висока вартість ліцензій або обмежений функціонал у безкоштовних версіях; відсутність адаптації під конкретні потреби окремих організацій; складність впровадження та навчання персоналу; обмежені можливості локального використання та автономної роботи.

Крім того, аналіз предметної області показує, що значна кількість малих підприємств і підрозділів продовжує використовувати неефективні інструменти, такі як електронні таблиці або паперовий облік, що ускладнює контроль виконання задач і підвищує ризик втрати даних.

Окремої уваги заслуговує питання інформаційної безпеки та розмежування доступу. У більшості універсальних систем механізми доступу є або занадто складними, або недостатньо гнучкими для реалізації чіткого розподілу ролей між користувачами, що може призводити до порушення принципів безпеки інформації.

Проведений аналіз аналогів та існуючих підходів свідчить про доцільність розробки спеціалізованої інформаційної системи обліку персоналу та задач, яка буде орієнтована на потреби невеликих організацій, забезпечуватиме простоту використання, автономність роботи, ефективний контроль виконання задач, а також реалізацію рольового контролю доступу.

1.2 Постановка завдання на кваліфікаційну роботу бакалавра

Метою кваліфікаційної роботи є розробка інформаційної системи обліку персоналу та управління задачами з використанням мови програмування C# та реалізацією рольового контролю доступу.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– здійснити аналіз сучасного стану проблеми автоматизації управління персоналом та задачами, а також дослідити існуючі програмні рішення та їхні особливості;

– визначити функціональні та нефункціональні вимоги до розроблюваної інформаційної системи;

– обґрунтувати вибір технологій, інструментальних засобів та архітектури програмного забезпечення;

– спроєктувати структуру інформаційної системи, включаючи логічну модель даних та взаємодію компонентів;

– розробити базу даних для зберігання інформації про працівників, користувачів та задачі;

– реалізувати програмний продукт із використанням мови програмування C# та технології Windows Forms із застосуванням ORM-засобів;

– забезпечити реалізацію механізмів рольового контролю доступу, авторизації та захисту даних;

– провести тестування та налагодження розробленої інформаційної системи з метою перевірки її працездатності та відповідності поставленим вимогам.

РОЗДІЛ 2
СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ

2.1 Аналіз, визначення вимог до розроблюваного програмного забезпечення та проєктування програмного забезпечення

Процес розробки інформаційної системи обліку персоналу та управління задачами передбачає формування чітко структурованих вимог до програмного забезпечення, вибір відповідної моделі життєвого циклу, а також застосування методів аналізу та проєктування системи. На початковому етапі важливим є визначення підходу до розробки, який забезпечить гнучкість, можливість поетапного вдосконалення та адаптації до змін вимог користувачів.

Для реалізації поставленого завдання доцільно використовувати ітераційну модель життєвого циклу програмного забезпечення. Такий підхід дозволяє розбити процес розробки на окремі етапи, кожен з яких завершується отриманням працездатного результату. Ітераційна модель забезпечує можливість поступового розширення функціональності системи, а також дозволяє враховувати нові вимоги, що виникають у процесі розробки. Це особливо актуально для інформаційних систем, які орієнтовані на реальні бізнес-процеси та можуть змінюватися в залежності від потреб користувачів.

Визначення вимог до програмного забезпечення здійснювалося на основі аналізу предметної області, дослідження існуючих аналогів та вивчення функціональних потреб потенційних користувачів. При цьому було враховано специфіку організації робочих процесів, що передбачає наявність різних ролей користувачів із відповідним рівнем доступу до інформації. Основними користувачами системи є адміністратор (або менеджер) та працівник, які взаємодіють із системою відповідно до своїх функціональних обов’язків.

Основною метою розроблюваної системи є автоматизація процесів обліку персоналу та управління задачами в організації з можливістю контролю їх виконання та реалізацією рольового контролю доступу. Для досягнення цієї мети система повинна забезпечувати ведення обліку працівників, управління

обліковими записами користувачів, створення та призначення задач, контроль їх виконання, а також збереження історії змін. Важливим аспектом є забезпечення різного рівня доступу до функцій системи залежно від ролі користувача.

Функціональні вимоги до системи визначають її поведінку та набір можливостей, які повинні бути реалізовані. Зокрема, система повинна забезпечувати можливість введення, редагування та видалення інформації про працівників, створення облікових записів користувачів та виконання процедури авторизації. Також передбачено реалізацію механізмів створення задач, їх редагування, призначення відповідальним працівникам, встановлення статусу та пріоритету. Важливим елементом є можливість пошуку, фільтрації та сортування задач, що забезпечує зручність роботи користувачів із системою. Крім того, система повинна підтримувати ведення історії змін задач, що дозволяє здійснювати контроль виконання та відстежувати дії користувачів.

Окрім функціональних вимог, важливу роль відіграють нефункціональні вимоги, які визначають якісні характеристики системи. До таких вимог належать зручність використання, що передбачає інтуїтивно зрозумілий інтерфейс, швидкодія системи при обробці даних, надійність збереження інформації та забезпечення її безпеки. Також важливими є вимоги до масштабованості та можливості подальшого розширення функціональності системи, що дозволить адаптувати її до змін у діяльності організації. Система повинна забезпечувати автономну роботу без необхідності постійного підключення до мережі Інтернет, що є важливою перевагою для локальних інформаційних систем.

У процесі проєктування системи було також розглянуто основні сценарії її використання, які відображають типові дії користувачів. Зокрема, адміністратор має можливість додавати нових працівників, створювати облікові записи користувачів та призначати задачі. Працівник, у свою чергу, отримує доступ лише до власних задач, може переглядати їх та змінювати статус виконання. Такий підхід забезпечує чітке розмежування функцій і відповідальності між користувачами.

Аналіз інформаційних потоків показує, що система забезпечує введення, обробку, збереження та відображення даних. Інформація про працівників і задачі вводиться користувачем через інтерфейс, після чого зберігається у базі даних. За запитом користувача система виконує обробку даних і відображає результати у зручному для сприйняття вигляді. Така організація інформаційних потоків забезпечує ефективну взаємодію користувача із системою та швидкий доступ до необхідної інформації.

Окрему увагу приділено проєктуванню користувацького інтерфейсу, який реалізовано з урахуванням принципів зручності, простоти та логічної структурованості. Інтерфейс системи включає навігаційні елементи, таблиці для відображення даних, форми введення інформації та елементи керування, що забезпечують взаємодію користувача з системою. Використання технології Windows Forms дозволяє створити зручний настільний додаток із зрозумілою логікою роботи.

З метою формалізації структури та поведінки системи було застосовано підхід UML-моделювання, який дозволяє відобразити основні компоненти системи, їх взаємозв’язки та сценарії використання [11]. Використання діаграми варіантів використання дає змогу описати взаємодію користувачів із системою та визначити основні функціональні можливості програмного продукту. Діаграма класів дозволяє представити структуру даних, визначити сутності системи та встановити зв’язки між ними. Для відображення логіки виконання основних процесів застосовано діаграму діяльності, яка демонструє послідовність дій користувачів та системи. Діаграми послідовності використовуються для деталізації взаємодії між окремими компонентами системи під час виконання конкретних операцій, таких як авторизація або зміна стану задачі. Крім того, для відображення архітектури програмного забезпечення використано діаграму компонентів, яка демонструє взаємодію між основними модулями системи. Такий підхід забезпечує комплексне представлення як статичної, так і динамічної структури системи, сприяє глибшому розумінню її

функціонування ще на етапі проєктування та дозволяє зменшити ймовірність виникнення помилок під час реалізації.

Для наочного відображення взаємодії користувачів із розроблюваною системою побудовано діаграму варіантів використання, наведену на рисунку 2.1.

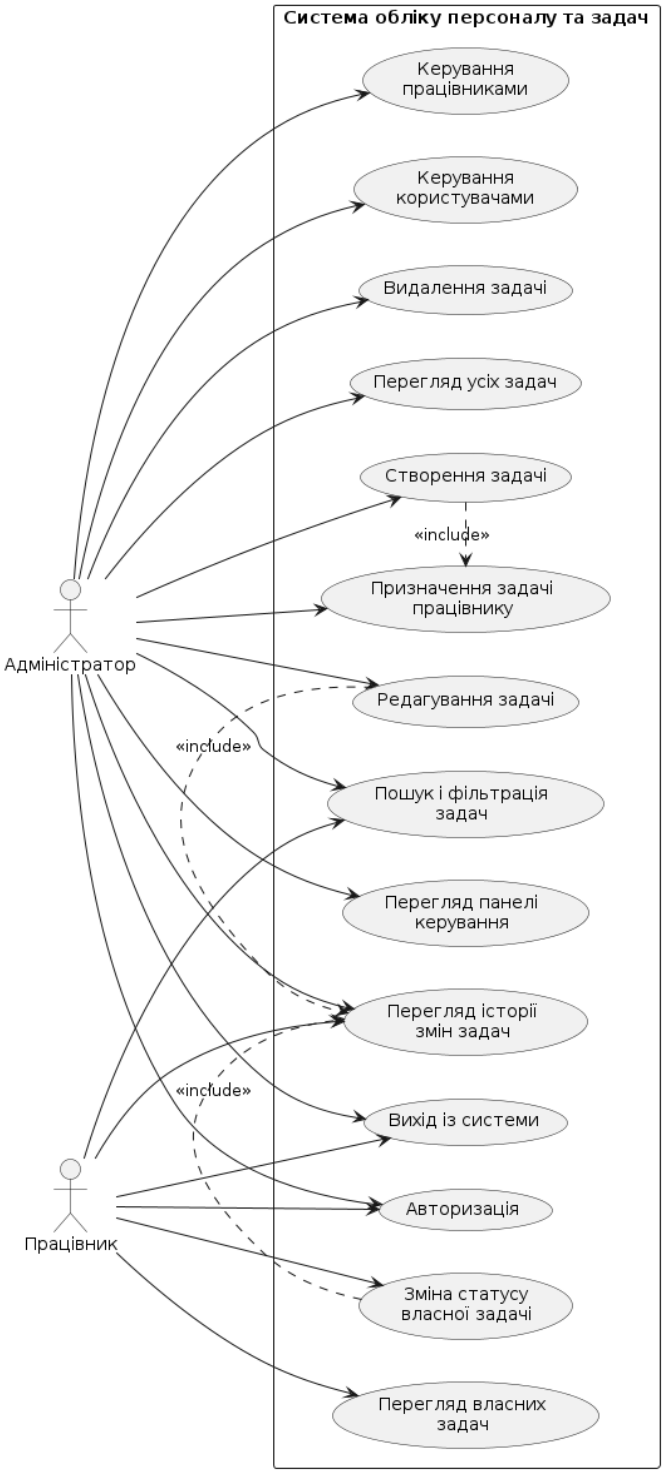

Рисунок 2.1 – Діаграма варіантів використання

Як показано на рисунку 2.1, функціональні можливості системи залежать від ролі користувача, що забезпечує розмежування доступу до її основних модулів.

Структуру основних сутностей програмного забезпечення та зв’язки між ними подано у вигляді діаграми класів на рисунку 2.2.

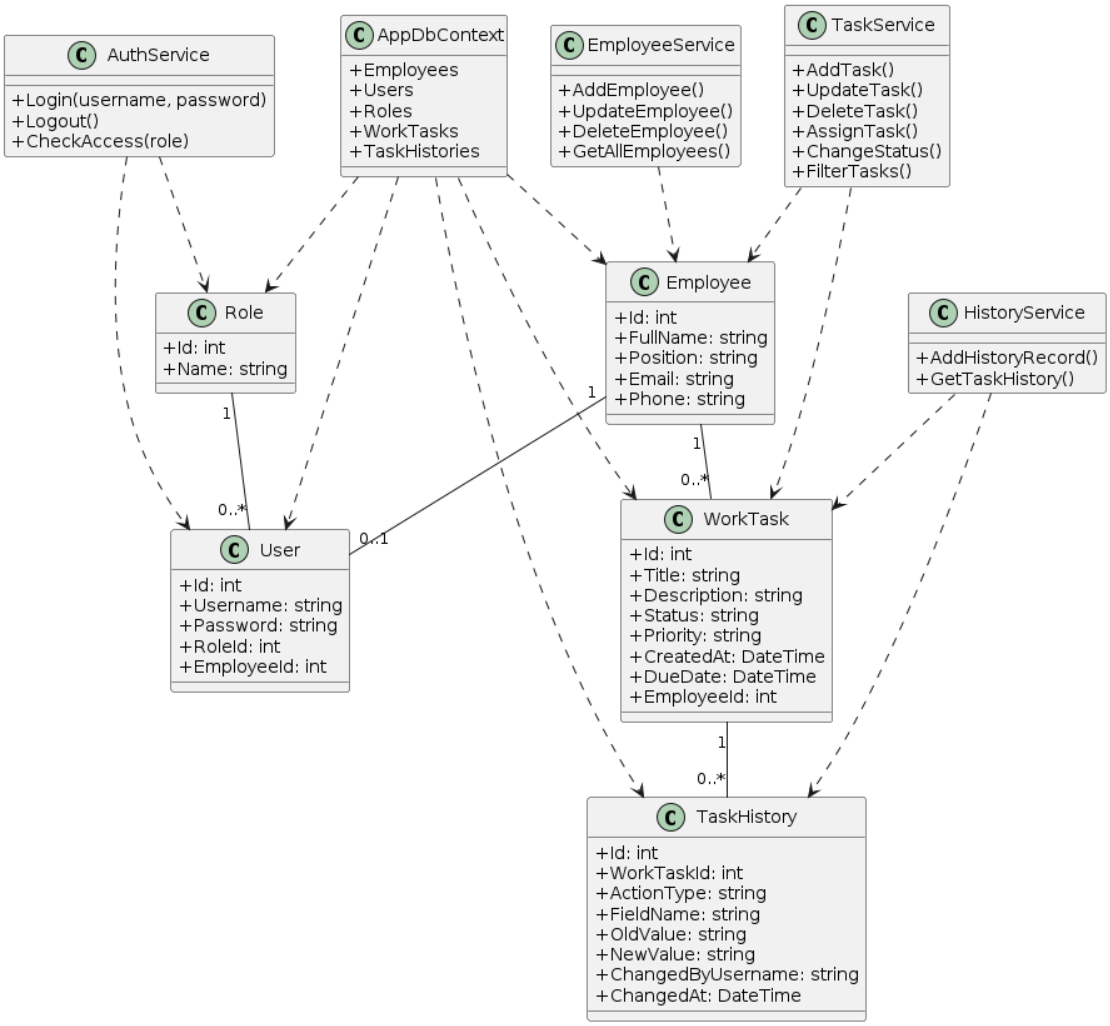

Рисунок 2.2 – Діаграма класів

Відповідно до рисунка 2.2, система включає сутності працівників, користувачів, ролей, задач та історії змін, що формують основу логічної моделі програмного забезпечення.

Послідовність основних дій користувачів у процесі створення та виконання задач доцільно подати за допомогою діаграми діяльності, наведеної на рисунку 2.3.

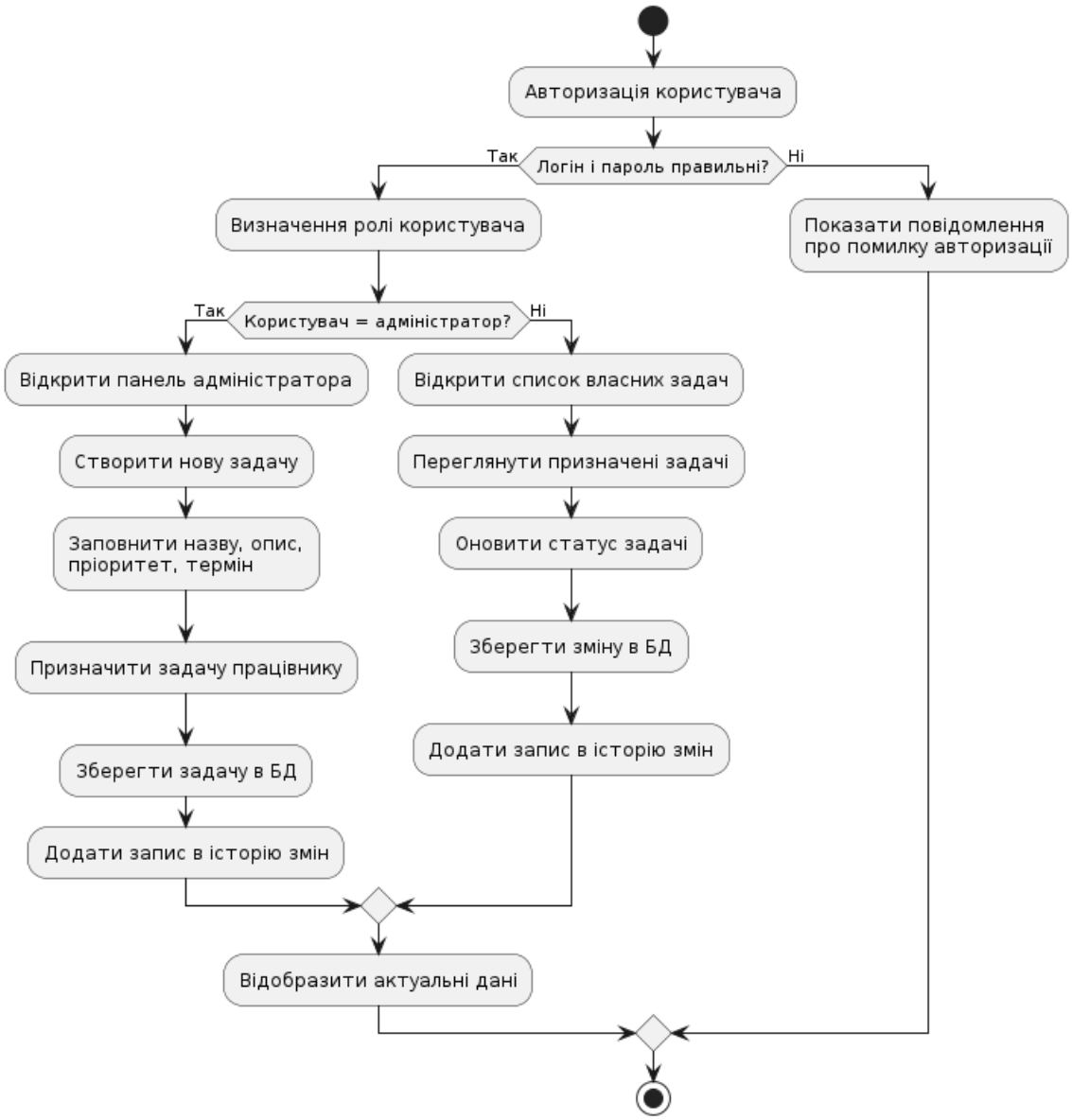

Рисунок 2.3 – Діаграма діяльності

Як видно з рисунка 2.3, робота системи включає етапи авторизації, перевірки ролі користувача, створення або оновлення задачі та збереження змін у базі даних.

Процес взаємодії компонентів системи під час виконання авторизації користувача представлено на діаграмі послідовності, наведеній на рисунку 2.4. Дана діаграма відображає послідовність обміну даними між користувачем, формою авторизації та базою даних під час перевірки облікових даних і визначення ролі користувача в системі.

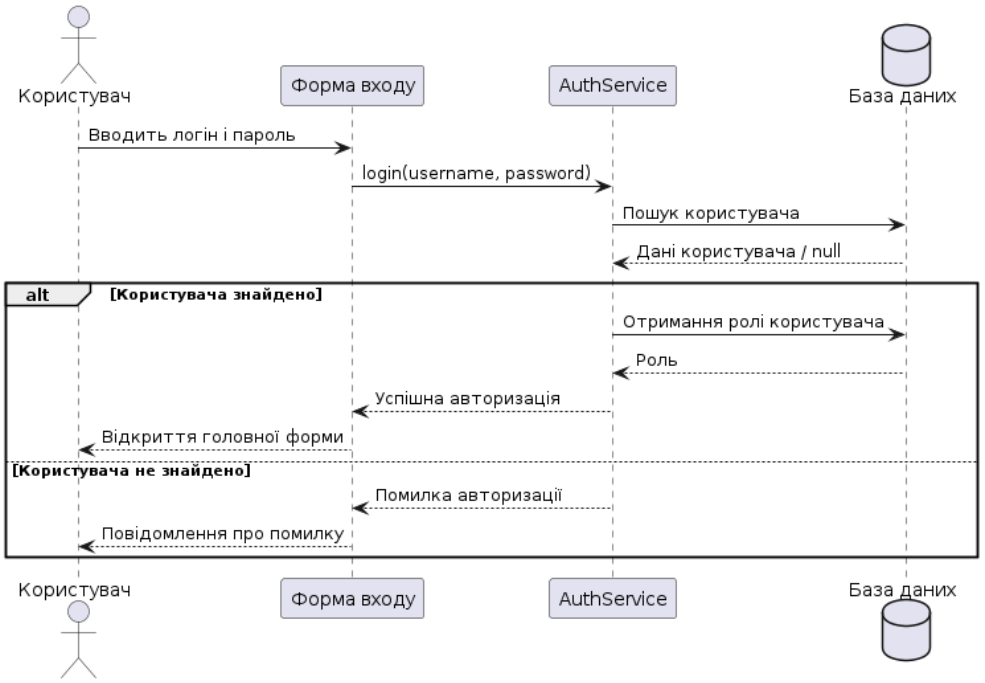

Рисунок 2.4 – Діаграма послідовності авторизації

Згідно з рисунком 2.4, під час авторизації виконується перевірка введених облікових даних, пошук користувача у базі даних та визначення його ролі в системі.

Архітектурну організацію розроблюваної системи та взаємодію її основних програмних компонентів подано на рисунку 2.5.

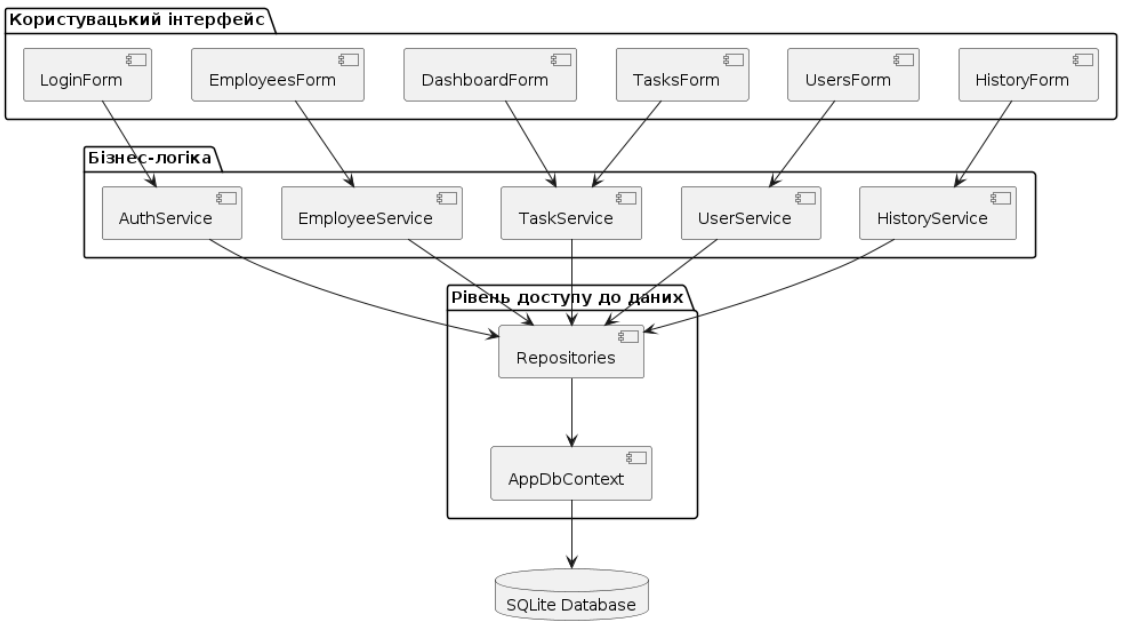

Рисунок 2.5 – Діаграма компонентів

Як видно з рисунка 2.5, система має багаторівневу структуру, що включає інтерфейс користувача, бізнес-логіку, рівень доступу до даних і базу даних.

Проведене моделювання та аналіз вимог дозволили сформувати цілісне уявлення про структуру та функціонування системи.

2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання

Розробка інформаційної системи обліку персоналу та управління задачами передбачає обґрунтований вибір програмних засобів, методів реалізації та алгоритмів обробки даних. Вибір технологічного стеку здійснювався з урахуванням сучасних вимог до програмного забезпечення, зокрема забезпечення надійності, продуктивності, масштабованості, безпеки та зручності використання. Важливим фактором також була можливість швидкої розробки, простоти впровадження та підтримки системи.

У якості основної мови програмування було обрано C#, яка входить до складу платформи .NET та є однією з провідних мов для створення прикладного програмного забезпечення. Мова C# підтримує об’єктно-орієнтовану парадигму програмування, що дозволяє ефективно моделювати предметну область за допомогою класів, об’єктів, наслідування, поліморфізму та інкапсуляції. Це дає змогу створювати структурований, зрозумілий та легко підтримуваний код. Крім того, C# забезпечує автоматичне керування пам’яттю завдяки використанню механізму збирача сміття, що знижує ризик виникнення помилок, пов’язаних із витоками пам’яті [9].

Платформа .NET надає потужну інфраструктуру для розробки програмного забезпечення, включаючи велику кількість бібліотек, засобів роботи з файлами, мережею, базами даних та інтерфейсами користувача. Вона забезпечує кросплатформеність (у випадку використання сучасних версій .NET), високу продуктивність та безпеку виконання програм. Важливою перевагою є

інтеграція з різними інструментами розробки, що значно спрощує процес створення складних інформаційних систем [10].

Середовище розробки Visual Studio було обрано як основний інструмент створення програмного продукту. Воно забезпечує зручний редактор коду з підтримкою підсвічування синтаксису, автоматичного доповнення, рефакторингу та навігації по коду. Крім того, Visual Studio містить вбудовані засоби налагодження, які дозволяють відстежувати виконання програми, знаходити помилки та аналізувати стан змінних у реальному часі. Також середовище підтримує інтеграцію з системами контролю версій, що є важливим для організації процесу розробки.

Для реалізації графічного інтерфейсу користувача було обрано технологію Windows Forms. Вона дозволяє створювати настільні додатки з використанням подієво-орієнтованої моделі програмування, що є зручною для реалізації інтерактивних систем. Windows Forms забезпечує швидку розробку інтерфейсів завдяки використанню візуального конструктора, який дозволяє розміщувати елементи керування, такі як кнопки, текстові поля, таблиці та інші компоненти, без необхідності написання великої кількості коду. Обрана технологія є доцільною для створення локальних інформаційних систем, оскільки не потребує розгортання веб-сервера та забезпечує швидкий доступ до функціональності [12].

У якості альтернативи розглядалися також інші підходи, зокрема веб-орієнтовані технології та фреймворки, такі як ASP.NET або JavaScript-фреймворки. Проте для поставленого завдання було обрано саме настільний додаток, оскільки він краще відповідає вимогам автономності, простоти розгортання та використання у межах локальної інфраструктури організації.

Для організації зберігання даних було обрано систему керування базами даних SQLite [13]. Вона є вбудованою реляційною СКБД, яка не потребує окремого серверного встановлення та конфігурації. Це значно спрощує процес розгортання програмного продукту, оскільки база даних зберігається у вигляді одного файлу. SQLite підтримує стандарт мови SQL, що дозволяє виконувати всі

необхідні операції над даними, включаючи створення таблиць, вставку, оновлення, видалення та вибірку записів. Незважаючи на свою компактність, SQLite забезпечує достатній рівень продуктивності для невеликих і середніх систем, що робить її оптимальним вибором для даного проєкту.

Для взаємодії з базою даних було використано Entity Framework Core, який реалізує концепцію об’єктно-реляційного відображення [14]. Ця технологія дозволяє працювати з даними у вигляді об’єктів мови програмування, що значно спрощує процес розробки та підвищує читабельність коду. Entity Framework Core автоматично виконує перетворення між об’єктами та таблицями бази даних, що дозволяє уникнути написання великої кількості SQL-запитів вручну. Крім того, дана технологія підтримує механізм міграцій, який дозволяє змінювати структуру бази даних без втрати даних, що є важливим при розвитку системи.

У процесі розробки було застосовано об’єктно-орієнтований підхід, який дозволяє розділити систему на логічні компоненти, кожен з яких відповідає за певну функціональність. Це сприяє підвищенню зрозумілості структури програмного забезпечення, спрощує його тестування та підтримку. Також було використано принципи багаторівневої архітектури, що передбачає розділення системи на рівень представлення (інтерфейс користувача), рівень бізнес-логіки та рівень доступу до даних.

Алгоритмічна частина системи включає реалізацію основних функціональних процесів, таких як авторизація користувачів, управління задачами та обробка інформації. Алгоритм авторизації передбачає введення користувачем логіна і пароля, після чого виконується перевірка цих даних у базі. У разі успішної перевірки визначається роль користувача, на основі якої формується доступ до функціональних можливостей системи. Такий підхід дозволяє забезпечити реалізацію рольового контролю доступу та захист інформації.

Алгоритми управління задачами включають створення нових записів, їх редагування, видалення, а також призначення задач конкретним працівникам. При зміні статусу задачі відбувається оновлення відповідного запису у базі даних

та фіксація змін у журналі історії. Це дозволяє забезпечити прозорість роботи системи та можливість відстеження дій користувачів. Для реалізації пошуку та фільтрації задач використовуються алгоритми вибірки даних за заданими критеріями, що забезпечує швидкий доступ до необхідної інформації.

Особлива увага приділяється забезпеченню цілісності та коректності даних. Для цього використовуються обмеження на рівні бази даних, такі як первинні та зовнішні ключі, а також перевірки у програмному коді. Це дозволяє запобігти виникненню логічних помилок та забезпечити узгодженість інформації.

Таким чином, обрані засоби, методи та алгоритми забезпечують ефективну реалізацію інформаційної системи, яка відповідає поставленим вимогам, є зручною у використанні, надійною та придатною до подальшого розвитку і масштабування.

РОЗДІЛ 3
РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ОБЛІКУ ПЕРСОНАЛУ ТА ЗАДАЧ

3.1 Практична реалізація об’єкта проєктування

Практична реалізація інформаційної системи обліку персоналу та задач здійснювалася із використанням мови програмування C# у середовищі розробки Visual Studio. У процесі створення програмного забезпечення було застосовано об’єктно-орієнтований підхід, що дозволило структуровано організувати код, розділити функціональність системи на окремі модулі та забезпечити зручність подальшого супроводу.

Програмний продукт реалізовано у вигляді настільного додатку з використанням технології Windows Forms, що забезпечує зручний графічний інтерфейс користувача та швидку взаємодію із системою. Архітектура додатку побудована за принципом розділення на рівень представлення, бізнес-логіки та доступу до даних, що сприяє підвищенню гнучкості та масштабованості системи.

На початковому етапі реалізації було створено структуру проєкту, яка включає моделі даних, сервіси для роботи з бізнес-логікою та форми для взаємодії з користувачем. Для роботи з базою даних було використано технологію Entity Framework Core, що дозволяє здійснювати доступ до даних через об’єкти мови програмування.

Модель користувача системи реалізована у вигляді класу, який містить основні властивості, необхідні для авторизації та визначення ролі користувача. Приклад реалізації класу користувача наведено у лістингу 3.1.

Лістинг 3.1 – Клас користувача

```
public class User
{    public int Id { get; set; }
    public string Username { get; set; }
    public string Password { get; set; }
    public int RoleId { get; set; }
    public Role Role { get; set; }
    public int? EmployeeId { get; set; }
```

```
    public Employee Employee { get; set; }
}
```

кінець лістингу 3.1

Даний клас відображає структуру таблиці користувачів у базі даних та забезпечує зв’язок із роллю та працівником. Це дозволяє реалізувати механізм рольового доступу та прив’язку облікового запису до конкретного працівника.

Для організації доступу до бази даних було створено контекст даних, який забезпечує взаємодію між програмою та базою даних.

Застосування контексту дозволяє централізовано керувати доступом до даних, а також виконувати операції додавання, оновлення та видалення записів.

Однією з ключових функцій системи є авторизація користувачів. Для цього було реалізовано відповідний алгоритм перевірки облікових даних. Приклад методу авторизації наведено у лістингу 3.2.

Лістинг 3.2 – Метод авторизації користувача

```
public User Login(string username, string password)
{
    using (var context = new AppDbContext())
    {
        var user = context.Users
            .Include(u => u.Role)
            .FirstOrDefault(u => u.Username == username && u.Password == 
password);

        return user;
    }
}
```

кінець лістингу 3.2

У межах розробленого програмного продукту реалізовано базовий механізм авторизації користувачів із перевіркою облікових даних у базі даних. Використано спрощений підхід до зберігання паролів, однак архітектура програмного забезпечення передбачає можливість подальшої інтеграції сучасних засобів криптографічного захисту, зокрема хешування та шифрування паролів.

Для забезпечення управління обліковими записами користувачів реалізовано відповідний модуль. Як видно з рисунка 3.1, користувач має можливість переглядати список користувачів, виконувати пошук, фільтрацію за ролями, а також здійснювати додавання, редагування та видалення записів.

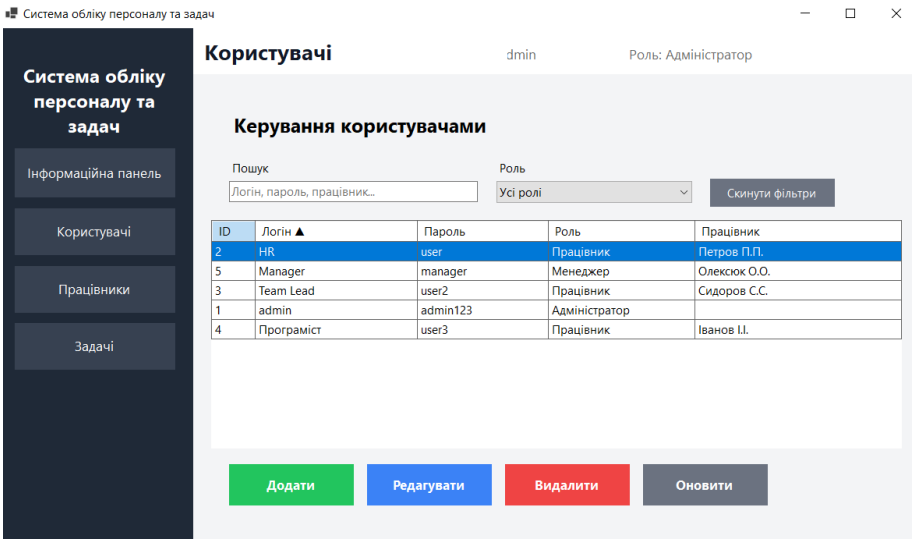

Рисунок 3.1 – Вікно керування користувачами

Для редагування параметрів користувача реалізовано окрему форму введення даних. На рисунку 3.2 представлено діалогове вікно, у якому можна змінити логін, пароль, роль користувача та виконати прив’язку до відповідного працівника, що забезпечує гнучке управління доступом у системі.

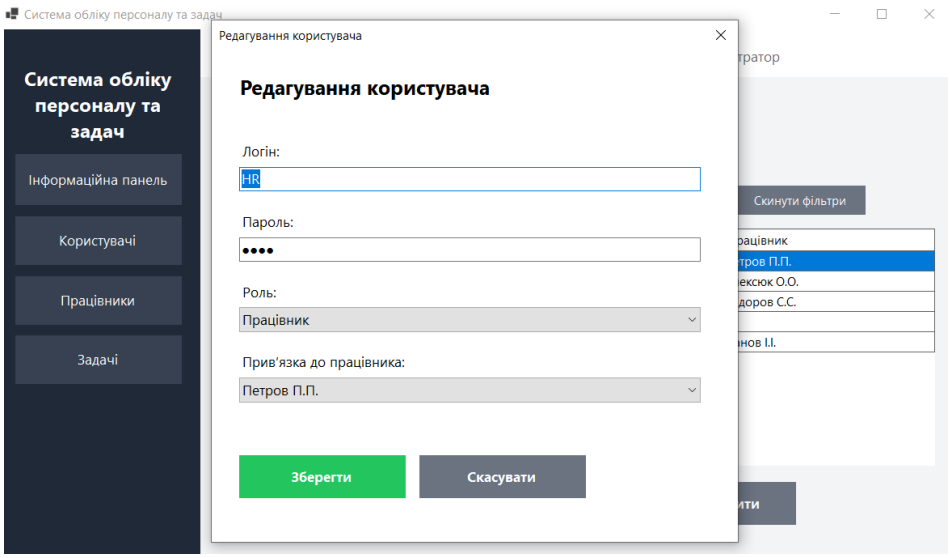

Рисунок 3.2 – Форма редагування користувача

Для обліку персоналу організації реалізовано модуль керування працівниками. Система дозволяє зберігати інформацію про працівників, включаючи ПІБ, посаду, відділ, контактні дані та дату прийому на роботу, а також виконувати операції пошуку, редагування та видалення (рис. 3.3).

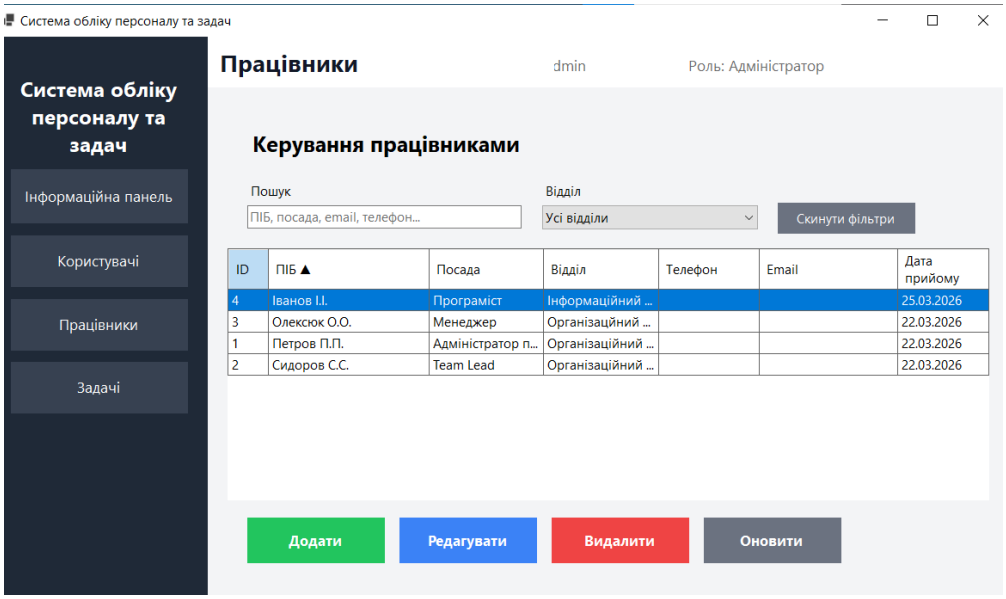

Рисунок 3.3 – Вікно керування працівниками

На рисунку 3.4 показано форму, яка забезпечує введення та зміну персональних даних працівника, що дозволяє підтримувати актуальність інформації в базі даних.

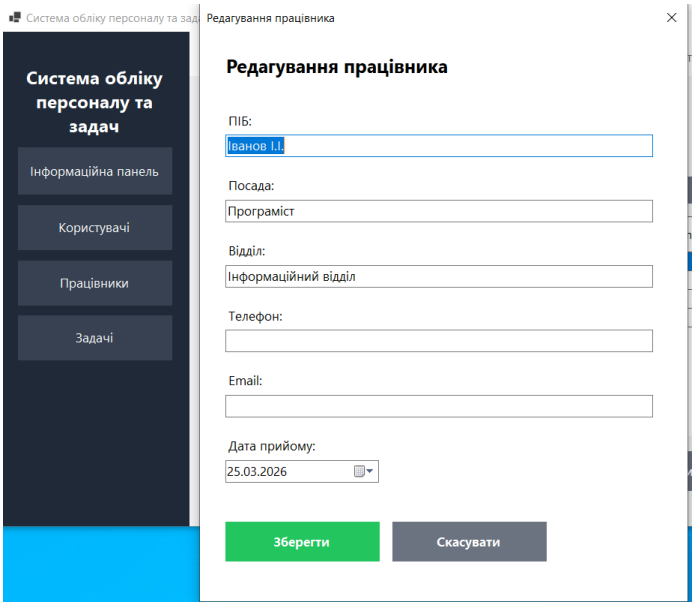

Рисунок 3.4 – Форма редагування працівника

Реалізація управління задачами передбачає створення, редагування та зміну статусу задач. Приклад методу створення задачі наведено у лістингу 3.3.

Лістинг 3.3 – Метод створення задачі

```
public void AddTask(WorkTask task)
{
    using (var context = new AppDbContext())
    {
        context.WorkTasks.Add(task);
        context.SaveChanges();
    }
}
```

кінець лістингу 3.3

Як видно з рисунка 3.5, користувач має змогу переглядати перелік задач, здійснювати фільтрацію за статусом, пріоритетом та працівником, а також контролювати терміни виконання задач.

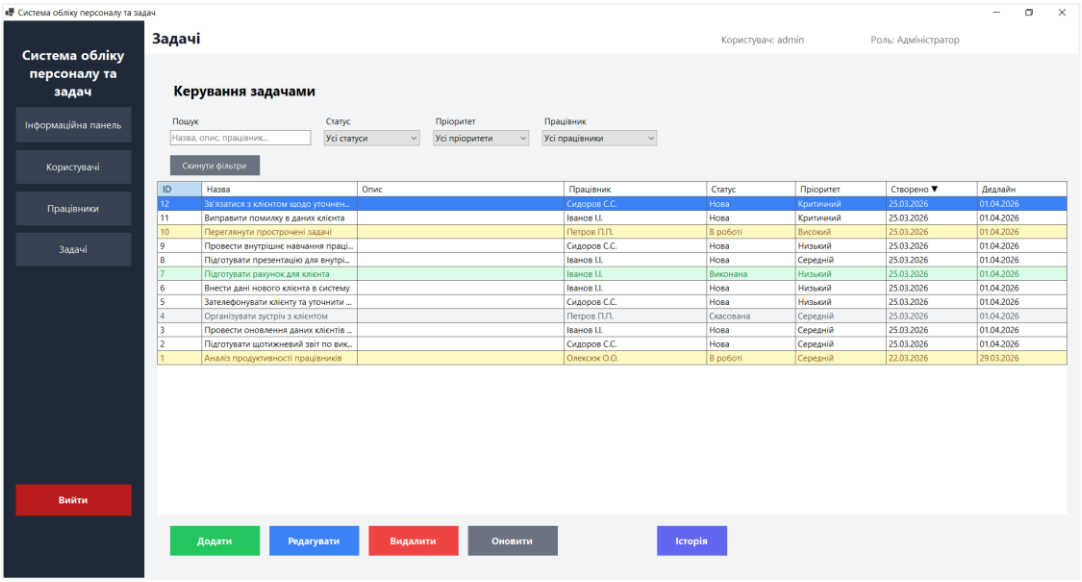

Рисунок 3.5 – Вікно керування задачами

Для створення та редагування задач передбачено відповідне діалогове вікно. На рисунку 3.6 представлено форму, яка дозволяє задавати назву задачі, опис, відповідального працівника, статус, пріоритет, дату створення та дедлайн, що забезпечує ефективне управління процесом виконання робіт.

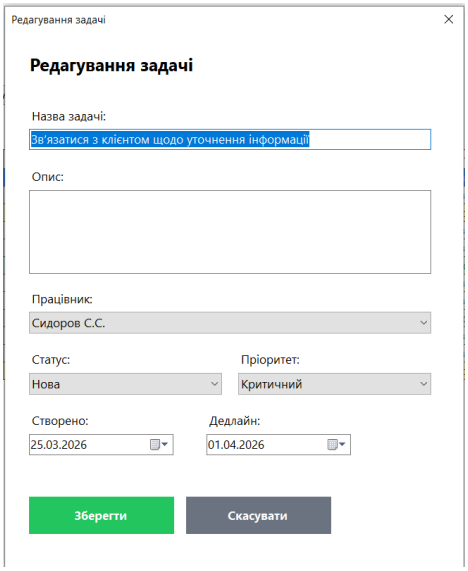

Рисунок 3.6 – Форма редагування задачі

При зміні статусу задачі важливо фіксувати всі зміни для подальшого аналізу. Для цього реалізовано механізм ведення історії змін. Приклад відповідного коду наведено у лістингу 3.4.

Лістинг 3.4 – Збереження історії змін задачі

```
public void AddHistory(int taskId, string field, string oldValue, string
newValue, string username)
{
    using (var context = new AppDbContext())
    {
        var history = new TaskHistory
        {
            WorkTaskId = taskId,
            FieldName = field,
            OldValue = oldValue,
            NewValue = newValue,
            ChangedByUsername = username,
            ChangedAt = DateTime.Now
        };
        context.TaskHistories.Add(history);
        context.SaveChanges();
    }
}
```

кінець лістингу 3.4

Зазначений механізм дозволяє відстежувати всі зміни, що відбуваються із задачами, що підвищує прозорість роботи системи.

Інтерфейс користувача реалізовано за допомогою форм, кожна з яких відповідає за окремий функціональний модуль системи. Наприклад, форма авторизації забезпечує введення логіна і пароля, перевірку даних та перехід до основного вікна програми. Форми управління працівниками та задачами дозволяють виконувати операції додавання, редагування та видалення записів, а також перегляд інформації у вигляді таблиць.

У процесі реалізації було використано елементи керування, такі як DataGridView для відображення даних, TextBox для введення інформації та Button для виконання дій користувача. Обробка подій здійснюється за допомогою обробників подій, що дозволяє реалізувати реакцію системи на дії користувача.

Таким чином, у процесі практичної реалізації було створено повнофункціональну інформаційну систему, яка забезпечує облік персоналу, управління задачами, контроль їх виконання та реалізацію рольового доступу. Використання мови програмування C# та сучасних технологій платформи .NET дозволило реалізувати надійний та ефективний програмний продукт.

3.2 Розробка бази даних

Важливою складовою розробленої інформаційної системи є база даних, яка забезпечує зберігання, оновлення та отримання інформації про працівників, користувачів, задачі та історію змін. Під час проєктування бази даних було враховано функціональні вимоги системи, необхідність забезпечення цілісності даних, зручності доступу до них та можливості подальшого розширення структури програмного продукту.

Для реалізації бази даних у програмному продукті використано SQLite, що є легкою вбудованою системою керування базами даних. Вибір саме цієї СКБД зумовлений тим, що вона не потребує окремого серверного розгортання, забезпечує швидкий доступ до даних та є зручною для використання у локальних

настільних додатках. База даних зберігається у вигляді окремого файла, що спрощує встановлення, перенесення та резервне копіювання системи.

На етапі розробки було визначено основні сутності предметної області, які повинні бути відображені у структурі бази даних. До них належать працівники, ролі користувачів, користувачі системи, робочі задачі та історія змін задач. Відповідно до цього було сформовано логічну модель бази даних, яка забезпечує реалізацію зв’язків між об’єктами та підтримує виконання основних функціональних операцій.

Для відображення структури бази даних та зв’язків між її основними таблицями побудовано ER-діаграму, наведену на рисунку 3.7.

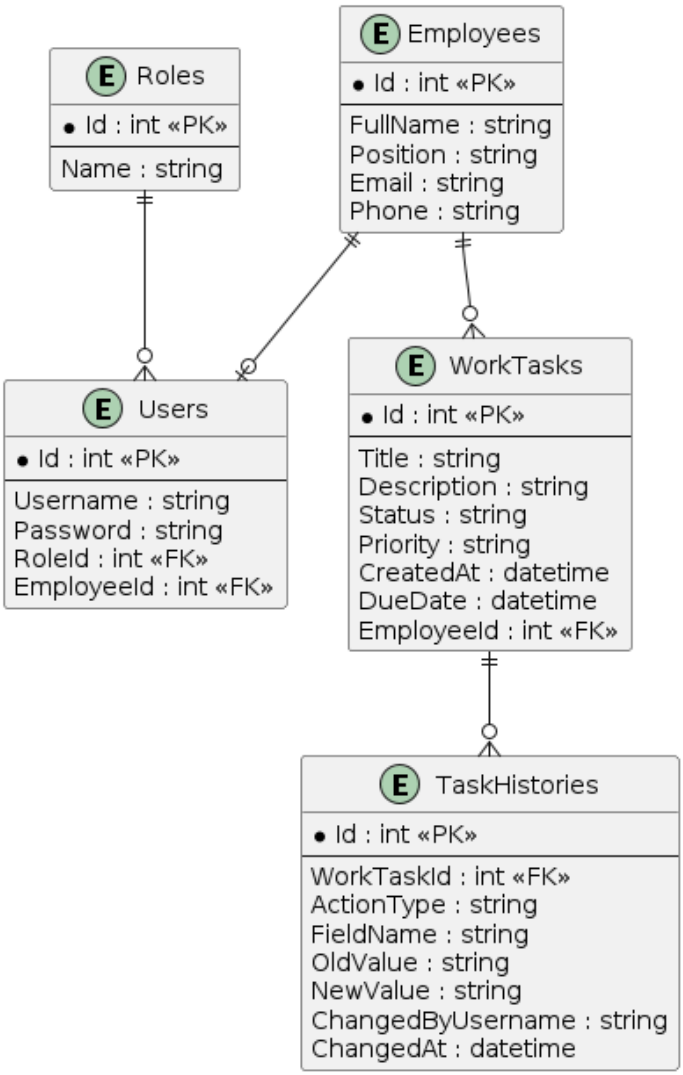

Рисунок 3.7 – ER-діаграма бази даних системи обліку персоналу та задач

Побудована ER-діаграма відображає, що таблиця ролей пов’язана з таблицею користувачів за принципом «один до багатьох», оскільки одна роль може бути призначена кільком користувачам. Таблиця працівників також пов’язана з таблицею користувачів, оскільки кожному працівнику може відповідати обліковий запис у системі. Крім того, таблиця працівників пов’язана із таблицею задач, оскільки одному працівнику може бути призначено декілька задач. Таблиця історії змін пов’язана із таблицею задач і використовується для збереження інформації про всі зміни, які відбувалися під час роботи із задачами. Як видно з рисунка 3.7, така структура дозволяє забезпечити логічну цілісність даних і реалізувати контроль за змінами у системі.

Центральне місце у структурі бази даних займає таблиця Employees, яка містить відомості про працівників організації. У ній зберігаються ідентифікатор працівника, прізвище, ім’я та по батькові, посада, відділ, номер телефону, електронна адреса та дата прийому на роботу. Ця таблиця використовується як основа для формування кадрового обліку та зв’язку із задачами й користувачами системи.

Таблиця Roles призначена для зберігання переліку ролей користувачів. Її використання дозволяє реалізувати механізм рольового контролю доступу, за якого права користувача визначаються не окремо для кожного запису, а через належність до певної ролі. У розробленій системі передбачено ролі адміністратора, менеджера та працівника, що дає змогу розмежувати функціональні можливості користувачів.

Таблиця Users містить дані, необхідні для авторизації у системі. До її полів належать ідентифікатор користувача, логін, пароль, зовнішній ключ на таблицю ролей та зовнішній ключ на таблицю працівників. Наявність зв’язку між користувачем і працівником дозволяє однозначно ідентифікувати особу, яка працює в системі, а також обмежити доступ працівника лише до власних задач.

Для реалізації механізму управління задачами використовується таблиця WorkTasks, яка містить назву задачі, її опис, статус, пріоритет, дату створення, дедлайн та ідентифікатор працівника, якому вона призначена. Саме ця таблиця

забезпечує підтримку основної бізнес-логіки системи, пов’язаної з постановкою, виконанням та контролем задач.

Таблиця TaskHistories реалізує функцію аудиту змін і використовується для зберігання інформації про зміни, які вносилися до задач. У ній містяться дані про ідентифікатор задачі, тип дії, назву зміненого поля, попереднє та нове значення, ім’я користувача, який виконав зміну, а також дату і час внесення змін. Наявність такої таблиці підвищує прозорість функціонування системи та дозволяє відстежувати історію виконання задач.

Фізична реалізація бази даних у програмному продукті здійснювалася за допомогою Entity Framework Core. Створення таблиць відбувалося на основі класів моделей, які відображають сутності предметної області. Використання технології ORM дозволило скоротити обсяг ручної роботи з SQL-командами, підвищити зручність підтримки коду та забезпечити узгодженість між програмною моделлю і структурою бази даних.

Приклад моделі працівника, яка використовується для формування відповідної таблиці бази даних, наведено у лістингу 3.5.

Лістинг 3.5 – Модель сутності працівника

```
public class Employee
{
    public int Id { get; set; }
    public string FullName { get; set; }
    public string Position { get; set; }
    public string Department { get; set; }
    public string Phone { get; set; }
    public string Email { get; set; }
    public DateTime HireDate { get; set; }

    public ICollection<WorkTask> WorkTasks { get; set; }
    public User User { get; set; }
}
```

кінець лістингу 3.5

Наведений клас відображає структуру таблиці працівників та демонструє використання навігаційних властивостей для встановлення зв’язків із задачами і користувачем. Аналогічно були реалізовані інші сутності системи.

Для централізованого доступу до даних було створено клас контексту бази даних, який містить набори сутностей та забезпечує взаємодію із SQLite. Приклад реалізації контексту наведено у лістингу 3.6.

Лістинг 3.6 – Контекст бази даних

```
public class AppDbContext : DbContext
{
    public DbSet<Employee> Employees { get; set; }
    public DbSet<Role> Roles { get; set; }
    public DbSet<User> Users { get; set; }
    public DbSet<WorkTask> WorkTasks { get; set; }
    public DbSet<TaskHistory> TaskHistories { get; set; }

    protected override void OnConfiguring(DbContextOptionsBuilder
optionsBuilder)
    {
        optionsBuilder.UseSqlite("Data Source=app.db");
    }
}
```

кінець лістингу 3.6

Наявність контексту бази даних дозволяє виконувати операції додавання, редагування, видалення та вибірки інформації через засоби мови програмування C#, що значно спрощує взаємодію з даними.

На рівні структури бази даних важливим є забезпечення цілісності даних. Для цього використовуються первинні ключі, які забезпечують унікальність кожного запису, та зовнішні ключі, що формують зв’язки між таблицями. Таке рішення запобігає появі неузгоджених даних, наприклад, неможливості призначити задачу неіснуючому працівнику або прив’язати користувача до відсутньої ролі.

Крім того, у системі реалізовано логічні перевірки коректності введених даних на рівні програмного коду. Це стосується перевірки обов’язкових полів, допустимості значень статусу та пріоритету задачі, а також узгодженості між роллю користувача та доступними для нього функціями. Поєднання перевірок на рівні коду і на рівні бази даних підвищує надійність функціонування всієї системи.

Таким чином, у процесі розробки бази даних було сформовано логічно цілісну та функціонально повну структуру, яка забезпечує зберігання і обробку інформації, необхідної для роботи системи обліку персоналу та задач. Обрана структура бази даних відповідає функціональним вимогам програмного продукту, підтримує реалізацію рольового контролю доступу, ведення історії змін і може бути розширена у подальших версіях системи.

3.3 Захист інформаційно-комп’ютерної системи

Забезпечення захисту інформаційно-комп’ютерної системи є важливою складовою розробки програмного забезпечення, особливо у випадку систем, що працюють із персональними даними працівників та службовою інформацією. У розробленій системі обліку персоналу та задач реалізовано комплекс заходів, спрямованих на забезпечення конфіденційності, цілісності та доступності даних [15].

Одним із основних механізмів захисту в системі є автентифікація користувачів. Для доступу до функціональних можливостей системи користувач повинен ввести логін і пароль. Перевірка введених даних здійснюється шляхом порівняння з інформацією, що зберігається у базі даних. Таким чином, доступ до системи мають лише зареєстровані користувачі, що дозволяє запобігти несанкціонованому використанню програмного продукту.

Після успішної автентифікації в системі застосовується механізм авторизації, який базується на рольовій моделі доступу. Кожному користувачу призначається певна роль, наприклад адміністратор, менеджер або працівник. Відповідно до цієї ролі визначається перелік доступних функцій. Адміністратор має повний доступ до всіх можливостей системи, включаючи керування користувачами, працівниками та задачами. Менеджер має можливість створювати та контролювати задачі, а працівник – переглядати та виконувати лише ті задачі, які йому призначені. Такий підхід дозволяє ефективно обмежити

доступ до даних і функцій системи відповідно до службових обов’язків користувачів.

У розробленій системі реалізовано базовий механізм авторизації користувачів. Користувач вводить логін і пароль, після чого система виконує перевірку облікових даних у базі даних та визначає роль користувача. Відповідно до ролі відкривається доступ до дозволених функцій системи. Адміністратор має можливість керувати користувачами, працівниками та задачами, а працівник працює переважно з призначеними йому задачами.

Для забезпечення цілісності даних у системі використовуються обмеження на рівні бази даних, зокрема первинні та зовнішні ключі. Вони гарантують, що кожен запис є унікальним, а зв’язки між таблицями залишаються коректними. Наприклад, задача не може бути призначена неіснуючому працівнику, а користувач – неіснуючій ролі. Додатково реалізовано перевірку коректності введених даних на рівні програмного коду, що дозволяє уникнути помилок введення та логічних невідповідностей.

Для перевірки коректності введених даних використовується програмна валідація, приклад якої наведено у лістингу 3.7.

Лістинг 3.7 – Перевірка коректності введених даних

```
public bool ValidateEmployee(Employee employee)
{
    if (string.IsNullOrWhiteSpace(employee.FullName))
        return false;
    if (string.IsNullOrWhiteSpace(employee.Position))
        return false;
    if (employee.Email != null && !employee.Email.Contains("@"))
        return false;
    return true;
}
```

кінець лістингу 3.7

Даний фрагмент коду забезпечує перевірку обов’язкових полів та базову валідацію електронної пошти, що дозволяє уникнути збереження некоректних даних.

У системі також реалізовано механізм журналювання змін, який дозволяє відстежувати всі дії, пов’язані з редагуванням задач. Інформація про зміну значень, користувача, який виконав дію, а також дату і час змін зберігається у відповідній таблиці бази даних. Це забезпечує можливість аудиту дій користувачів, підвищує прозорість роботи системи та дозволяє виявляти можливі помилки або несанкціоновані зміни.

Ще одним важливим елементом захисту є контроль доступу до інтерфейсу користувача. У розробленому програмному забезпеченні реалізовано відображення лише тих елементів інтерфейсу, які доступні для конкретної ролі користувача. Наприклад, кнопки додавання, редагування або видалення даних можуть бути недоступними для користувачів із обмеженими правами. Це дозволяє зменшити ризик випадкового або навмисного порушення роботи системи.

Крім того, у системі передбачено базові заходи захисту від помилок користувача. Зокрема, перед виконанням операцій видалення даних може здійснюватися підтвердження дії, що дозволяє уникнути втрати важливої інформації. Також перевіряється коректність введення даних у полях форм, що запобігає внесенню некоректної або неповної інформації до бази даних.

Для запобігання аварійному завершенню роботи системи реалізовано обробку виняткових ситуацій, приклад якої наведено у лістингу 3.8.

Лістинг 3.8 – Обробка помилок при роботі з базою даних

```
try
{
    _context.WorkTasks.Add(task);
    _context.SaveChanges();
}
catch (Exception ex)
{
    MessageBox.Show("Помилка збереження даних: " + ex.Message);
}
```

кінець лістингу 3.8

Використання конструкції try-catch дозволяє перехоплювати помилки та забезпечує стабільну роботу системи.

Для забезпечення цілісності даних при виконанні комплексних операцій використовується механізм транзакцій, приклад якого наведено у лістингу 3.9.

Лістинг 3.9 – Використання транзакції

```
using (var transaction = _context.Database.BeginTransaction())
{
    try
    {
        _context.WorkTasks.Add(task);
        _context.SaveChanges();
        LogTaskChange(task.Id, "Створення", "", task.Title, currentUser);
        transaction.Commit();
    }
    catch
    {
        transaction.Rollback();
    }
}
```

кінець лістингу 3.9

Такий підхід дозволяє скасувати всі зміни у разі виникнення помилки та запобігти пошкодженню даних.

Важливим аспектом забезпечення безпеки є також фізичний захист даних. Оскільки база даних зберігається у вигляді файлу, необхідно забезпечити обмеження доступу до нього на рівні операційної системи. Це може бути реалізовано шляхом встановлення відповідних прав доступу до файлів або використанням додаткових засобів шифрування.

Для узагальнення основних загроз та засобів їх усунення сформовано таблицю 3.1.

Таблиця 3.1 – Загрози та заходи захисту

Загроза	Засіб захисту
Несанкціонований доступ	Авторизація та ролі
Введення некоректних даних	Валідація даних
Втрата даних	Транзакції
Помилки виконання	Обробка винятків
Несанкціоновані зміни	Журналювання

Структуру розмежування доступу користувачів у системі представлено на рисунку 3.8.

Рисунок 3.8 – Схема контролю доступу в системі

Таким чином, у розробленій інформаційно-комп’ютерній системі реалізовано комплексний підхід до забезпечення захисту даних, який включає автентифікацію, авторизацію, контроль доступу, забезпечення цілісності даних та аудит дій користувачів. Застосовані заходи дозволяють забезпечити надійне функціонування системи та захист інформації від несанкціонованого доступу і модифікації.

3.4 Тестування та налагодження інформаційно-комп’ютерної системи

Тестування є важливим етапом розробки програмного забезпечення, який дозволяє перевірити правильність функціонування системи, виявити помилки та забезпечити її надійність. У процесі розробки інформаційної системи обліку персоналу та задач було проведено комплексне тестування, спрямоване на перевірку відповідності програмного продукту поставленим вимогам.

Тестування системи здійснювалося на різних рівнях, зокрема проводилося функціональне тестування, яке дозволило перевірити коректність реалізації

основних функцій системи, а також перевірка взаємодії між компонентами програмного забезпечення. Особлива увага приділялася перевірці критичних функцій, таких як авторизація користувачів, управління задачами, обробка даних та забезпечення рольового доступу.

У процесі тестування перевірялися основні сценарії роботи користувачів, включаючи створення та редагування записів, зміну статусу задач, пошук інформації та контроль доступу до функцій системи. Результати тестування дозволили підтвердити коректність роботи більшості функцій, а також виявити окремі недоліки, які були усунені на етапі налагодження.

Для систематизації результатів тестування було сформовано таблицю тестових сценаріїв.

Для перевірки функціональності системи сформовано набір тестових сценаріїв, результати яких наведено у таблиці 3.2.

Таблиця 3.2 – Результати функціонального тестування системи

№	Тестовий сценарій	Вхідні дані	Очікуваний результат	Фактичний результат	Статус
1	Авторизація користувача	Логін, пароль	Успішний вхід у систему	Вхід виконано	✓
2	Невірний пароль	Логін, неправильний пароль	Повідомлення про помилку	Виведено повідомлення	✓
3	Додавання працівника	Дані працівника	Запис збережено в БД	Запис додано	✓
4	Редагування працівника	Оновлені дані	Дані оновлено	Дані змінено	✓
5	Видалення працівника	ID працівника	Запис видалено	Запис видалено	✓
6	Створення задачі	Дані задачі	Задача створена	Задача створена	✓
7	Зміна статусу задачі	Новий статус	Статус оновлено	Статус змінено	✓
8	Перегляд задач працівником	Роль «Працівник»	Доступ лише до своїх задач	Доступ обмежено	✓
9	Редагування задач працівником	Роль «Працівник»	Відмова у доступі	Доступ заборонено	✓
10	Пошук задач	Ключове слово	Відображення знайдених задач	Пошук працює	✓

Як видно з таблиці 3.2, усі основні функції системи працюють коректно та відповідають поставленим вимогам.

У процесі тестування було виявлено окремі помилки, пов’язані з обробкою введених даних, відображенням інформації у таблицях та перевіркою прав доступу. Зокрема, на початковому етапі розробки виникали ситуації, коли некоректні дані могли бути збережені у базі даних, а також випадки некоректного відображення списків задач.

Для усунення виявлених недоліків було вдосконалено механізми валідації даних, додано перевірки коректності введення інформації, а також реалізовано обробку виняткових ситуацій. Крім того, було оптимізовано роботу із базою даних, що дозволило підвищити швидкодію системи.

Окрему увагу було приділено перевірці механізму рольового доступу, у результаті чого було забезпечено коректне обмеження функціональних можливостей користувачів залежно від їх ролі.

Таким чином, у процесі тестування та налагодження було підтверджено працездатність розробленої інформаційної системи, її відповідність функціональним вимогам та надійність у роботі. Виявлені недоліки були своєчасно усунені, що дозволило підвищити якість програмного продукту та забезпечити його стабільне функціонування.

ВИСНОВКИ

У кваліфікаційній роботі було досягнуто поставлену мету, яка полягала у розробці інформаційної системи обліку персоналу та управління задачами з використанням мови програмування C# та реалізацією рольового контролю доступу.

Було здійснено аналіз сучасного стану проблеми автоматизації управління персоналом та задачами. Проведено огляд існуючих програмних рішень, таких як системи управління проєктами та CRM-системи, визначено їх переваги та недоліки. На основі цього обґрунтовано доцільність створення власної інформаційної системи, орієнтованої на потреби невеликих організацій.

Визначено функціональні та нефункціональні вимоги до програмного забезпечення. Сформовано перелік основних функцій системи, зокрема управління працівниками, користувачами та задачами, а також вимоги до безпеки, надійності та зручності використання. Це дозволило сформувати чітке технічне бачення майбутнього програмного продукту.

Обґрунтовано вибір технологій, інструментальних засобів та архітектури системи. Обрано мову програмування C#, платформу .NET, технологію Windows Forms для реалізації інтерфейсу користувача, SQLite як систему керування базами даних та Entity Framework Core для взаємодії з базою даних. Такий вибір забезпечив ефективність розробки та стабільність роботи системи.

Виконано проєктування структури інформаційної системи. Побудовано UML-діаграми, зокрема діаграму варіантів використання, діаграму класів та інші, що дозволило формалізувати структуру системи, визначити взаємозв’язки між її компонентами та описати сценарії взаємодії користувачів із програмним продуктом.

Було розроблено базу даних системи. Створено логічну структуру бази даних, яка включає таблиці працівників, користувачів, ролей, задач та історії змін. Реалізовано зв’язки між таблицями за допомогою первинних і зовнішніх ключів, що забезпечує цілісність і узгодженість даних.

Реалізовано програмний продукт із використанням мови програмування C# та технології Windows Forms. Розроблено інтерфейс користувача, реалізовано основні функції системи, включаючи авторизацію, управління даними, обробку інформації та взаємодію з базою даних. У роботі наведено фрагменти програмного коду, що підтверджують практичну реалізацію системи.

Забезпечено реалізацію механізмів захисту інформації. У системі впроваджено автентифікацію користувачів, рольовий контроль доступу, перевірку коректності введених даних, обробку виняткових ситуацій та механізм журналювання змін. Це дозволило підвищити рівень безпеки та надійності програмного продукту.

Проведено тестування та налагодження інформаційної системи. Перевірено коректність роботи основних функцій, виконано тестування різних сценаріїв використання системи, виявлено та усунено помилки. Результати тестування підтвердили працездатність системи та її відповідність поставленим вимогам.

Таким чином, усі поставлені завдання було виконано у повному обсязі, а розроблена інформаційна система може бути використана для автоматизації обліку персоналу та управління задачами в організації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Software Engineering Body of Knowledge (SWEBOK V4). URL: https://www.computer.org/education/bodies-of-knowledge/software-engineering# about (дата звернення: 10.04.2026).

2. Silhavy R. Software Engineering Methods in Systems and Network Systems. Proceedings of 7th Computational Methods in Systems and Software. Vol. 2. 2023. URL: https://link.springer.com/book/10.1007/978-3-031-54813-0 (дата звернення: 21.04.2026).

3. Atlassian. Jira Software Documentation. URL: https://www.atlassian.com/software/jira (дата звернення: 10.04.2026).

4. Atlassian. Trello Guide. URL: https://trello.com/guide (дата звернення: 10.04.2026).

5. Asana. Product Guide. URL: https://asana.com/guide (дата звернення: 10.04.2026).

6. Salesforce. CRM Platform Overview. URL: https://www.salesforce.com (дата звернення: 20.04.2026).

7. HubSpot. CRM Software Guide. URL: https://www.hubspot.com/products/crm (дата звернення: 20.04.2026).

8. Zoho Corporation. Zoho CRM Documentation. URL: https://www.zoho.com/crm/help (дата звернення: 20.04.2026).

9. Microsoft. .NET Documentation. URL: https://learn.microsoft.com/dotnet (дата звернення: 05.05.2026).

10. Microsoft. C# Guide. URL: https://learn.microsoft.com/dotnet/csharp (дата звернення: 20.04.2026).

11. Fowler M. UML Distilled: A Brief Guide to the Standard Object Modeling Language. 4th ed. Addison-Wesley, 2021.

12. Microsoft. Windows Forms Documentation. URL: https://learn.microsoft.com/dotnet/desktop/winforms (дата звернення: 20.04.2026).

13. SQLite Documentation. URL: https://www.sqlite.org/docs.html (дата звернення: 05.05.2026).

14. Microsoft. Entity Framework Core Documentation. URL: https://learn.microsoft.com/ef/core (дата звернення: 05.05.2026).

15. OWASP Foundation. OWASP Top Ten Web Application Security Risks 2021. URL: https://owasp.org (дата звернення: 05.05.2026).