МІНІСТЕРСТВО ВНУТРІШНІХ СПРАВ УКРАЇНИ
ЛЬВІВСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ ВНУТРІШНІХ СПРАВ
ФАКУЛЬТЕТ №2
Кафедра інформаційних технологій

РОЗРОБЛЕННЯ ВЕБ-СЕРВІСУ З ФОРМУВАННЯ ЗВІТІВ ТА PDF-ДОКУМЕНТІВ ПІДРОЗДІЛІВ ПОЛІЦІЇ ЗА ДОПОМОГОЮ ФРЕЙМВОРКУ BLAZOR

кваліфікаційна робота
здобувача вищої освіти
4 курсу денної форми навчання
Віктор ВІЛЬЧИНСЬКИЙ

Науковий керівник:
доцент, кандидат технічних наук
Тарас РУДИЙ

Рецензент:
доцент, кандидат технічних наук
Світлана ЯЦИШИН

Кваліфікаційна робота допущена до захисту
«___» _____ 2026 р., протокол №____

Завідувач кафедри інформаційних технологій
_____ **Олег ЗАЧЕК**
(підпис)
_____

Львів

2026

АНОТАЦІЯ

ВІЛЬЧИНСЬКИЙ В. І. Розроблення веб-сервісу з формування звітів та pdf-документів підрозділів поліції за допомогою фреймворку blazor. — Рукопис.

Дослідження на здобуття освітнього ступеня «бакалавр» за спеціальністю 126 «Інформаційні системи та технології».-Львівський державний університет внутрішніх справ, МВС України, Львів, 2026.

У кваліфікаційній роботі розроблено веб-сервіс для автоматизації формування звітів та генерації PDF-документів у підрозділах Національної поліції. У процесі дослідження проаналізовано предметну область, недоліки існуючого паперового документообігу та обґрунтовано вибір сучасного технологічного стеку на базі платформи .NET 8. Спроєктовано архітектуру системи за принципами чистої архітектури, інтерфейс користувача за допомогою фреймворку Blazor та розроблено структуру локальної бази даних SQLite. У практичній частині реалізовано логіку доступу до даних, механізм швидкого формування стандартизованих PDF-звітів (рапортів) засобами бібліотеки QuestPDF, а також наведено результати тестування програмного продукту та інструкцію з його розгортання.

Ключові слова: веб-сервіс, Blazor, .NET 8, формування звітів, QuestPDF, SQLite, Національна поліція.

ABSTRACT

VILCHYNSKYI V. I. Development of a web service for generating reports and PDF documents for police departments using the Blazor framework.— Manuscript.

Research for the bachelor's degree in specialty 126 «Information systems and technologies». -Lviv State University of Internal Affairs, MIA of Ukraine, Lviv, 2026.

In the qualification work, a web service was developed to automate the generation of reports and PDF documents in the units of the National Police. During the research, the subject area and the shortcomings of the existing paper document flow were analyzed, and the choice of a modern technology stack based on the .NET 8 platform was justified. The system architecture was designed according to the principles of clean architecture, the user interface using the Blazor framework, and the structure of the local SQLite database was developed. In the practical part, the data access logic and the mechanism for the rapid generation of standardized PDF reports using the QuestPDF library were implemented, along with the results of software testing and deployment instructions.

Keywords: web service, Blazor, .NET 8, report generation, QuestPDF, SQLite, National Police.

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

БД - База даних

ДСТУ - Державний стандарт України

ІПНП - Інформаційний підсистема Національної поліції

ККУ - Кримінальний кодекс України

КУпАП - Кодекс України про адміністративні правопорушення

МВС - Міністерство внутрішніх справ України

НПА - Нормативно-правові акти

НПУ - Національна поліція України

ПІБ - Прізвище, ім’я, по батькові

СЕД - Система електронного документообігу

СУБД - Система управління базами даних

.NET 8 - Програмна платформа компанії Microsoft

API- Application Programming Interface (інтерфейс прикладного програмування

CSS - Cascading Style Sheets (каскадні таблиці стилів)

EF Core (Entity Framework Core) - об’єктно-реляційний засіб доступу до даних для платформи .NET

LINQ - Language Integrated Query (мова інтегрованих запитів)

ORM - Object-Relational Mapping (об'єктно-реляційне відображення)

PDF - Portable Document Format (портативний формат документів)

RBAC - Role-Based Access Control (керування доступом на основі ролей)

UI / UX - User Interface / User Experience (інтерфейс користувача / досвід користувача)

ЗМІСТ

ВСТУП

Актуальність теми. В умовах цифровізації Національної поліції України (НПУ) критично важливим є подолання розриву між зростаючими обсягами інформації та застарілими методами її обробки. Значна частина щоденної звітності досі формується ручним способом у звичайних офісних програмах. Це призводить до помилок ручного введення та низької оперативності управлінських рішень. Розробка спеціалізованого веб-сервісу для автоматизації документообігу є необхідною умовою для оптимізації роботи правоохоронних органів, що й обумовлює актуальність дослідження.

Аналіз останніх досліджень і публікацій. Проблематику використання автоматизованих інформаційних систем у діяльності МВС України, зокрема системи ІПНП, досліджував І. Є. Іванов [6]. Питання інформаційних технологій та документообігу висвітлено в працях О. І. Зачека, В. В. Сеника та ін. [5]. Архітектурні патерни та сучасні підходи до розробки програмного забезпечення на платформі .NET висвітлювали Р. Мартін [11], М. Прайс [12] і Е. Троелсен [9]. Базові вимоги щодо складу та структури реквізитів організаційно-розпорядчої документації визначено ДСТУ 4163:2020 [3], а стандарти захисту даних-відповідним законодавством [2] та працями з інформаційної безпеки В. Б. Вишні [4]. Проте питання створення спеціалізованих веб-сервісів для генерації PDF-звітів «з нуля», з урахуванням специфіки роботи відокремлених підрозділів Національної поліції України, досі залишається невирішеним практичним завданням.

Мета кваліфікаційної роботи полягає у підвищенні ефективності процесів звітування в підрозділах поліції шляхом розробки та розгортання автоматизованого веб-сервісу на базі фреймворку Blazor.

Для досягнення поставленої мети необхідно вирішити такі **завдання**:

1) Проаналізувати процеси документообігу НПУ та недоліки існуючих програмних рішень.

2) Обґрунтувати вибір технологічного стеку (.NET 8, Blazor, SQLite) для реалізації системи.

3) Спроєктувати загальну архітектуру веб-сервісу та алгоритми маршрутизації документів.

4) Розробити структуру локальної реляційної бази даних.

5) Реалізувати підсистему безпеки та рольову модель доступу з урахуванням поліцейської ієрархії.

6) Створити графічний інтерфейс та бізнес-логіку обробки рапортів.

7) Реалізувати механізм швидкої генерації стандартизованих PDF-документів засобами бібліотеки QuestPDF.

8) Провести функціональне тестування розробленого веб-сервісу та скласти інструкцію користувача.

Об'єкт дослідження — процес формування звітної документації в підрозділах Національної поліції України.

Предмет дослідження — методи та програмні засоби автоматизації формування звітів на базі сучасних веб-технологій.

Методи досліджень. У роботі використано комплекс методів: системний аналіз (для дослідження предметної області); об'єктно-орієнтоване проєктування (для побудови архітектури); моделювання даних (для розробки бази даних) та тестування програмного забезпечення.

Практичне значення роботи полягає у розробці діючого прототипу веб-сервісу. Його подальше вдосконалення та використання дозволить скоротити час на підготовку звітів на 60 - 70 %, мінімізувати вплив «людського фактору» та автоматизувати процеси відомчого документообігу.

Структура та обсяг роботи. Робота структурно поділена на вступ, чотири розділи, висновки та список із 22 використаних джерел, що викладені на 63 сторінках основного тексту та доповнені 22 ілюстраціями і 5 таблицями.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ВІДОМИХ ЗАСОБІВ РЕАЛІЗАЦІЇ

1.1. Дослідження процесів звітності та документообігу в підрозділах поліції

Національна поліція України (НПУ) щоденно обробляє великі масиви інформації для забезпечення публічної безпеки та протидії злочинності [1]. Як відзначають дослідники, ефективний документообіг та надійна аналітична підтримка є критично важливими складовими для безперебійної роботи будь-якого правоохоронного підрозділу [6].

Згідно з фаховою літературою, механізм документообігу має декілька стадій: створення документа, його передавання, впорядковане зберігання та відтворення [5]. І якщо етап накопичення та централізованого аналізу даних (у базах ІПНП) чи процеси офіційного пересилання готових документів (через відомчі СЕД) в органах МВС вже достатньо автоматизовані, то найважче піддається автоматизації початкова стадія-безпосереднє створення локальних звітів «з нуля» [5]. Це пояснюється тим, що написання рапортів містить значну творчу та аналітичну складову.

Сьогодні основним інструментом для підготовки юридичних та службових документів на місцях залишається текстовий процесор Microsoft Word [5]. Хоча базові принципи роботи з ним входять до обов'язкового курсу підготовки фахівців, такий ручний підхід до щоденного формування оперативних звітів має кілька суттєвих недоліків:

1. Підвищує ризик виникнення механічних помилок при перенесенні даних («людський фактор»).

2. Ускладнює дотримання єдиної структури ключових реквізитів (грифів, заголовків, підписів), визначеної базовими положеннями ДСТУ 4163:2020 [3].

3. Не забезпечує належного захисту інформації з обмеженим доступом на звичайних комп'ютерах, що прямо суперечить вимогам Закону України «Про

захист інформації в інформаційно- комунікаційних системах» [2] та базовим принципам інформаційної безпеки [4].

Отже, існує гостра необхідність у розробці спеціалізованого локального веб-сервісу. Такий програмний продукт дозволить автоматизувати найскладнішу початкову стадію документообігу (створення звітів) безпосередньо у підрозділі, забезпечить надійний рольовий доступ та швидку генерацію захищених, стандартизованих PDF-документів для їх подальшого завантаження у СЕД.

1.2. Огляд існуючих програмних рішень автоматизації службової діяльності

Процес автоматизації діяльності будь-якого державного підрозділу складається з двох ключових етапів: створення (генерації) звітної документації на основі первинних даних та її подальшого руху (маршрутизації). Тому аналіз існуючих програмних рішень доцільно розділити на огляд систем генерації звітів та систем електронного документообігу (СЕД).

Аналіз систем формування звітів та генерації PDF-документів

На ринку програмного забезпечення існує велика кількість універсальних платформ для побудови звітів. Для порівняльного аналізу обрано чотири найпопулярніші рішення, які найчастіше використовуються в корпоративному секторі:

1) **Microsoft SQL Server Reporting Services (SSRS).** Потужна серверна платформа від Microsoft [19]. Вимагає розгортання важкого сервера баз даних, що є надлишковим для локального підрозділу поліції. Крім того, ліцензування потребує значних фінансових витрат.

2) **FastReport .NET.** Один із найвідоміших генераторів звітів для платформи C# / .NET [17]. Продукт є комерційним. Інтерфейс генерації є занадто універсальним, тому вбудувати в нього специфічну бізнес-логіку

поліції (наприклад, автоматичну підстановку статей ККУ/КУпАП) вкрай складно.

3) **Crystal Reports.** Багаторічний галузевий стандарт для аналітичної звітності [16]. Має застарілу архітектуру, яка дуже важко інтегрується із сучасними веб-фреймворками на кшталт Blazor. Генерація великих масивів PDF-документів відбувається повільніше порівняно з сучасними аналогами.

4) **jsReport.** Сучасний кросплатформний сервер звітів із відкритим вихідним кодом [18]. Генерація PDF відбувається шляхом конвертації HTML-коду. Цей процес є ресурсомістким. Використання JavaScript для шаблонів порушує принцип строгої типізації, який є базовим для захищених застосунків.

Для наочності порівняльний аналіз існуючих універсальних генераторів звітів та розроблюваного у даній роботі спеціалізованого веб-сервісу наведено в табл. 1.1.

Таблиця 1.1

Порівняльна характеристика засобів генерації звітної документації

Програмне рішення	**Ліцензія**	**Основні недоліки для підрозділів НПУ**	**Наявність логіки поліції**
Microsoft SSRS	Комерційна (дорого)	Потребує важкого сервера баз даних та складного адміністрування	Відсутня
FastReport / Crystal Reports	Комерційна	Висока вартість, перевантажений універсальний інтерфейс	Відсутня
jsReport	Безкоштовна	Повільна генерація PDF (через HTML), відсутність строгої типізації	Відсутня
Розроблюваний веб-сервіс	Безкоштовна	Потребує лише початкового ознайомлення персоналу з новим інтерфейсом	**Наявна** (вбудовані довідники ККУ/КУпАП)

Джерело: розроблено автором.

Як видно з даних, наведених у таблиці 1.1, головним недоліком існуючих універсальних генераторів звітів є їхня орієнтація на загальнокорпоративний сектор. Жоден із наведених комерційних чи відкритих

інструментів не задовольняє повною мірою специфічні потреби локального відділу поліції. Вони вимагають високих фінансових витрат на ліцензування, складні у розгортанні та, найголовніше, не містять вбудованої правоохоронної логіки. Натомість концепція розроблюваного веб-сервісу передбачає створення максимально легкого та безкоштовного у розгортанні інструменту, який з коробки міститиме необхідні довідники правопорушень.

Аналіз систем електронного документообігу (СЕД)

Після того як первинний звіт або рапорт згенеровано локально, він має бути офіційно зареєстрований та переданий за належністю. На сучасному етапі цифровізації державного управління України ключову роль у цьому процесі відіграє Система електронної взаємодії органів виконавчої влади (СЕВ ОВВ)- державна телекомунікаційна інфраструктура, призначена для автоматизованого обміну юридично значущими документами із застосуванням кваліфікованого електронного підпису.

Згідно з даними офіційного реєстру державного підприємства «ДІЯ», на вітчизняному ринку наразі функціонує 39 протестованих систем електронного документообігу [14]. Усі ці платформи технічно здатні інтегруватися з СЕВ ОВВ через відповідні програмні інтерфейси (API), забезпечуючи безперебійний прийом та відправку кореспонденції між відомствами. Водночас результати тестування свідчать, що розширений функціонал, такий як підтримка роботи з нормативно-правовими актами (НПА) на етапі їх погодження, реалізований далеко не в усіх існуючих системах [14].

Серед комерційних рішень найбільшого поширення у державному та приватному секторах набули такі платформи, як «Megapolis.DocNet», «АСКОД» та «FossDoc». Поряд із комерційними продуктами, Міністерство внутрішніх справ активно впроваджує власну спеціалізовану розробку- систему «МІА: Документообіг», створену державним підприємством «ІНФОТЕХ». Запровадження цього продукту дозволяє суттєво зменшити паперове навантаження на апарат міністерства, автоматизувати контроль

виконавської дисципліни та прискорити маршрутизацію відомчих документів [13].

Проте при аналізі існуючих рішень варто розуміти їхнє цільове призначення. Класичні СЕД орієнтовані переважно на маршрутизацію, реєстрацію, накладання резолюцій та архівування **вже готових** документів у масштабах великих установ. Вони не надають гнучкого інструментарію для зручного покрокового створення специфічних галузевих документів "з нуля" безпосередньо виконавцем (наприклад, поліцейським, який складає статистичний звіт чи складний рапорт).

Обґрунтування доцільності власної розробки

Підсумовуючи огляд існуючих програмних рішень, можна стверджувати, що для відокремленого підрозділу поліції критичною проблемою є не стільки пересилання документа глобальними мережами, скільки швидкий і безпомилковий процес його створення.

Класичні системи електронного документообігу (СЕД), такі як державна система «МІА: Документообіг», орієнтовані переважно на маршрутизацію та реєстрацію вже готових документів у масштабах великих установ. Вони не надають гнучкого інструментарію для зручного покрокового створення специфічних галузевих звітів «з нуля» безпосередньо виконавцем.

Тому розробка власного локального веб-сервісу є практично обґрунтованим кроком. Такий продукт міститиме вбудовану логіку правоохоронної діяльності (довідники правопорушень за ККУ та КУпАП), не потребуватиме дорогого ліцензування та забезпечить миттєву генерацію структурованих PDF-звітів. Розроблений веб-сервіс не конкурує з державними СЕД, а доповнює їх, вирішуючи проблему якісної підготовки первинних PDF-документів для їх подальшого завантаження в мережі МВС.

1.3. Обґрунтування вибору технологічного стеку та фреймворку Blazor

Успішна реалізація веб-сервісу для потреб державного правоохоронного органу вимагає підбору надійних інструментів, що гарантують високий рівень інформаційної безпеки та мають тривалий цикл підтримки. З огляду на це, для розробки було обрано екосистему Microsoft .NET.

Платформа .NET 8

Основою програмного продукту виступає платформа .NET 8. Вибір саме 8-ї версії зумовлений її статусом довгострокової підтримки, що гарантує стабільність та отримання критичних оновлень безпеки [22].

Фреймворк Blazor Server та інтерфейс

Для реалізації графічного інтерфейсу користувача (UI) обрано технологію ASP.NET Core Blazor [15] . Вона дозволяє розробляти веб-інтерфейси виключно мовою C#, відмовляючись від традиційного використання JavaScript.

Для забезпечення максимальної безпеки застосовано модель хостингу Blazor Server. Її ключова перевага полягає в тому, що вся бізнес-логіка та обробка чутливої інформації (наприклад, персональних даних фігурантів) відбуваються виключно на захищеному сервері. Клієнтський браузер отримує лише візуальні оновлення інтерфейсу, тому вихідний код і структура бази даних ніколи не завантажуються на комп'ютер кінцевого користувача. Для створення адаптивного та професійного дизайну використано CSS-фреймворк Bootstrap із підтримкою «темної теми», що знижує навантаження на зір працівників під час нічних чергувань [8].

СУБД SQLite та Entity Framework Core

Для збереження оперативних даних інтегровано локальну СУБД SQLite [21]. Цей вибір є оптимальним для відокремленого підрозділу поліції, оскільки SQLite не потребує встановлення та адміністрування важкого сервера, зберігаючи дані у захищеному локальному файлі. Водночас, завдяки використанню технології Entity Framework Core, проєкт не має жорсткої

прив'язки до SQLite. У перспективі систему можна легко масштабувати на потужні промислові бази даних (Microsoft SQL Server або PostgreSQL), змінивши лише конфігураційний файл.

Бібліотека генерації документів QuestPDF

Критично важливим завданням системи є програмне формування базових макетів PDF-документів з урахуванням логіки державних стандартів діловодства (зокрема, ДСТУ 4163:2020) [3]. Для цього використано сучасну нативну .NET-бібліотеку QuestPDF [20]. Завдяки підходу Fluent API вона забезпечує миттєву генерацію звітів та гнучке позиціювання реквізитів безпосередньо у коді C#. Отже, обраний технологічний стек утворює цілісну, продуктивну та безпечну архітектуру, що дозволяє повною мірою реалізувати поставлені перед веб-сервісом завдання.

Висновки до першого розділу

У першому розділі проаналізовано процеси формування звітності в підрозділах Національної поліції України. З'ясовано, що створення щоденної рапортної документації на місцях залишається слабко автоматизованим. Використання стандартних офісних програм знижує оперативність, спричиняє помилки ручного введення та не гарантує належного захисту інформації.

Дослідження існуючих систем електронного документообігу (СЕД) та універсальних генераторів звітів показало, що вони орієнтовані переважно на маршрутизацію вже готових документів або є надто ресурсомісткими. Вони не вирішують локальної проблеми швидкого формування специфічних поліцейських рапортів «з нуля».

Обґрунтовано доцільність розробки власного веб-сервісу та визначено оптимальний технологічний стек для його реалізації: платформу .NET 8 із фреймворком Blazor Server для забезпечення інформаційної безпеки, СУБД SQLite для локального розгортання з можливістю подальшого масштабування (завдяки Entity Framework Core), а також бібліотеку QuestPDF для швидкої генерації PDF-документів.

РОЗДІЛ 2
ПРОЄКТУВАННЯ АРХІТЕКТУРИ ВЕБ-СЕРВІСУ

2.1. Формування вимог до системи та визначення ролей користувачів

Першим і найважливішим етапом розробки будь-якого програмного забезпечення є збір та аналіз вимог. Оскільки розроблюваний веб-сервіс призначений для автоматизації процесів документообігу поліції, критичним аспектом є створення чіткої рольової моделі, щоб кожен працівник мав доступ лише до тієї інформації, яка необхідна йому для роботи.

Визначення рольової моделі користувачів

У системі реалізовано багаторівневу модель управління доступом яка складається з чотирьох базових ролей:

1. Адміністратор: Має найвищий рівень доступу. Відповідає за додавання нових користувачів (редагування посад, звань, жетонів), підтримку користувачів та перегляд системного журналу аудиту.

2. Працівник: Базовий користувач (наприклад, слідчий або дільничний). Може створювати власні рапорти, генерувати їх у PDF та отримувати службові доручення від керівництва.

3. Керівник відділу: Користувач середньої ланки. Бачить структуру свого відділу, може переглядати та попередньо погоджувати рапорти своїх підлеглих, а також видавати їм доручення і переглядати статистику по своєму відділу.

4. Керівник управління: Користувач вищої ланки. Має доступ до загальної статистики підрозділу та наділений виключним правом остаточно затверджувати документи, після чого вони переходять в архів.

Функціональні вимоги

Функціональні вимоги описують, що саме повинна робити програма. На основі аналізу процесів поліцейського документообігу виділено такі основні вимоги:

1) **Управління документами:** Можливість створювати, редагувати та видаляти рапорти (видалення доступне лише для статусу «Чернетка»). Підтримка життєвого циклу документа через статуси («Чернетка», «Погоджено», «Затверджено»).

2) **Генерація PDF:** Вбудований механізм автоматичного формування звітів у форматі PDF (за допомогою бібліотеки QuestPDF).

3) **Система доручень:** Інструмент для керівників, що дозволяє призначати завдання підлеглим із вказанням пріоритету та дедлайну.

4) **Аналітика та пошук:** Наявність інформаційної панелі (Dashboard) для керівництва з підрахунком зареєстрованих правопорушень, а також зручний пошук документів за датою чи автором.

5) **Аудит:** Ведення «Журналу аудиту», де автоматично фіксуються критичні дії користувачів (наприклад, видалення рапортів або доручень).

Нефункціональні вимоги

Ці вимоги формуються на основі міжнародних стандартів якості програмного забезпечення **[10]** та визначають атрибути надійності і безпеки розробленої системи:

1) **Безпека:** Реалізовано захист від підбору паролів (Brute Force)-акаунт блокується після кількох невдалих спроб. Сесія автоматично завершується через 10 хвилин простою.

2) **Швидкодія:** Завдяки використанню технології Blazor Server навігація між сторінками відбувається миттєво, без візуального перезавантаження. Генерація PDF-файлу в пам'яті сервера займає лічені секунди.

3) **Зручність (UI/UX):** Адаптивний інтерфейс підтримує «темну тему» для комфортної роботи вночі. Для мінімізації помилок ПІБ, звання та посада автора автоматично підтягуються у форму рапорту.

4) **Масштабованість:** Використання Entity Framework Core дозволяє, за потреби, швидко перевести систему з локальної бази SQLite на потужніший сервер (наприклад, PostgreSQL).

Моделювання прецедентів (Use Cases) Для візуалізації того, як користувачі взаємодіють із системою, побудовано UML-діаграму прецедентів (Use Case Diagram). Вона наочно демонструє розподіл функціоналу між описаними вище ролями (рис. 2.1).

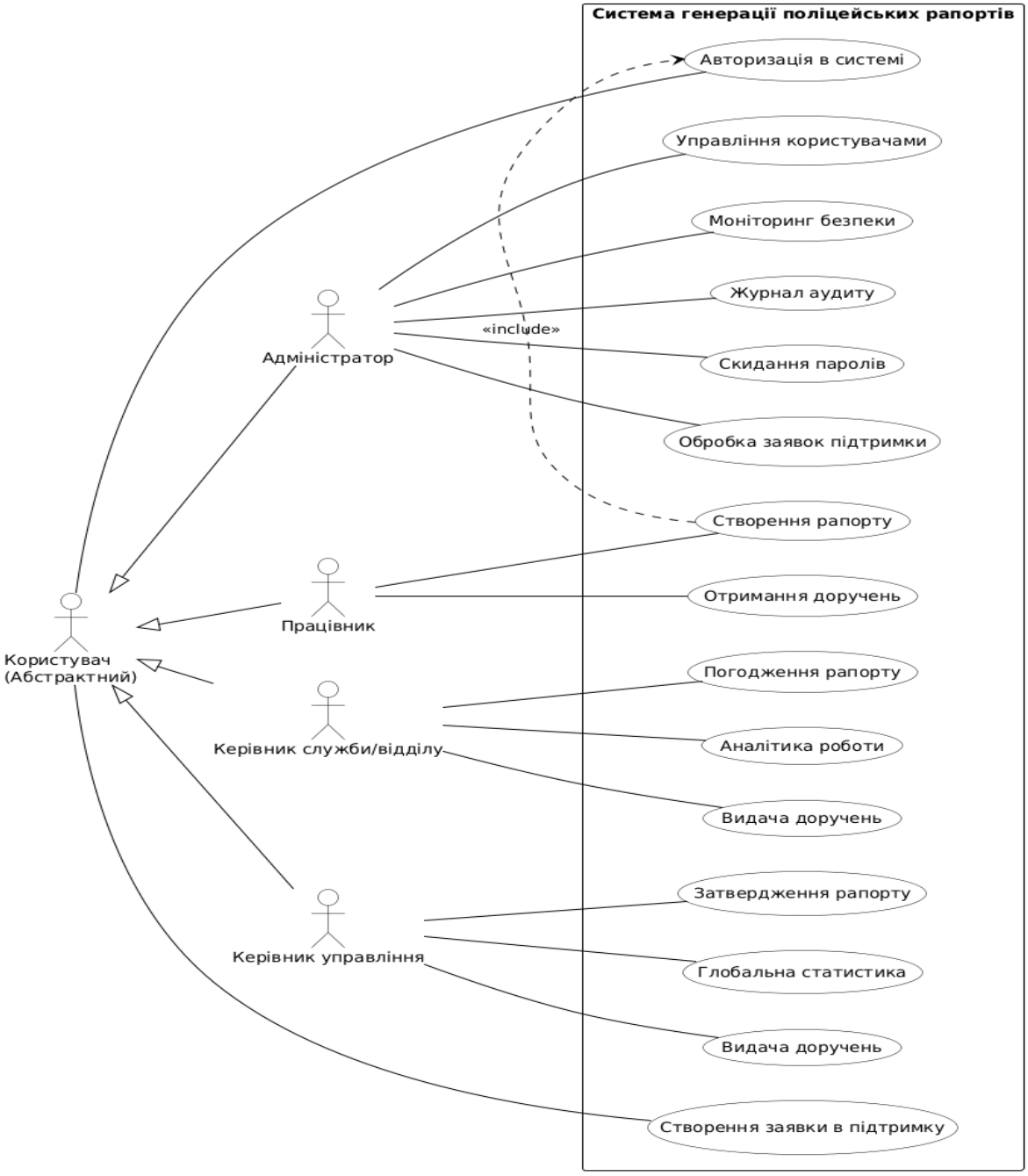


Рис. 2.1. Діаграма прецедентів (Use Case) користувачів веб-сервісу

Для деталізації логіки роботи розроблено специфікації основних сценаріїв використання системи, які наведено в табл. 2.1.

Таблиця 2.1

Основні прецеденти використання системи документообігу

Назва прецеденту	Актор (Роль)	Опис сценарію	Очікуваний результат
Створення рапорту	Працівник	Працівник обирає тип рапорту, заповнює фабулу та зберігає документ. За потреби експортує його в PDF.	Створення запису в базі даних зі статусом «Чернетка». Документ доступний лише автору.
Погодження рапорту	Керівник відділу	Керівник переглядає надісланий рапорт підлеглого та натискає кнопку «Погодити».	Статус змінюється на «Погоджено». Фіксація дії в журналі аудиту. Документ стає видимим для керівника управління.
Затвердження документа	Керівник управління	Керівник управління аналізує погоджений рапорт, перевіряє кримінальну кваліфікацію та натискає «Затвердити».	Документ отримує статус «Архівний». Редагування чи видалення документа повністю блокується.
Відновлення доступу	Працівник	Працівник вводить свій ПІБ та жетон на сторінці входу. Система перевіряє наявність збігів.	При успішному збігу генерується внутрішній запит до панелі Адміністратора.
Перевірка аудиту	Адміністратор	Адміністратор відкриває модуль «Журнал аудиту» для перевірки факту видалення документів посадовими особами.	Виведення таблиці з логами: дата, час, ПІБ порушника, номер видаленого документа.

Джерело: розроблено автором.

Детальне формування функціональних та нефункціональних вимог, а також моделювання сценаріїв взаємодії користувачів, дозволило чітко окреслити межі програмного продукту. Сформована рольова модель та визначені прецеденти є міцним концептуальним підґрунтям для наступного

етапу розробки-безпосереднього проєктування структурної архітектури коду та реляційної бази даних системи.

2.2. Проєктування структурної архітектури системи

Забезпечення надійності, інформаційної безпеки та легкості подальшого супроводу програмного продукту безпосередньо залежить від обраного архітектурного підходу. Під час розробки веб-сервісу було прийнято рішення відмовитися від класичної монолітної архітектури на користь багаторівневої (N-Tier Architecture) із використанням принципів «Чистої архітектури» (Clean Architecture) Роберта Мартіна [16].

Головною парадигмою такого підходу є чітке розділення відповідальності. Логіка відображення інтерфейсу повністю відділена від бізнес-правил, а бізнес-правила не залежать від конкретної СУБД. Це дозволяє масштабувати або тестувати один модуль без ризику порушити працездатність інших.

Структурно архітектура розробленого рішення у середовищі Visual Studio розділена на п'ять незалежних проєктів (рис. 2.2).

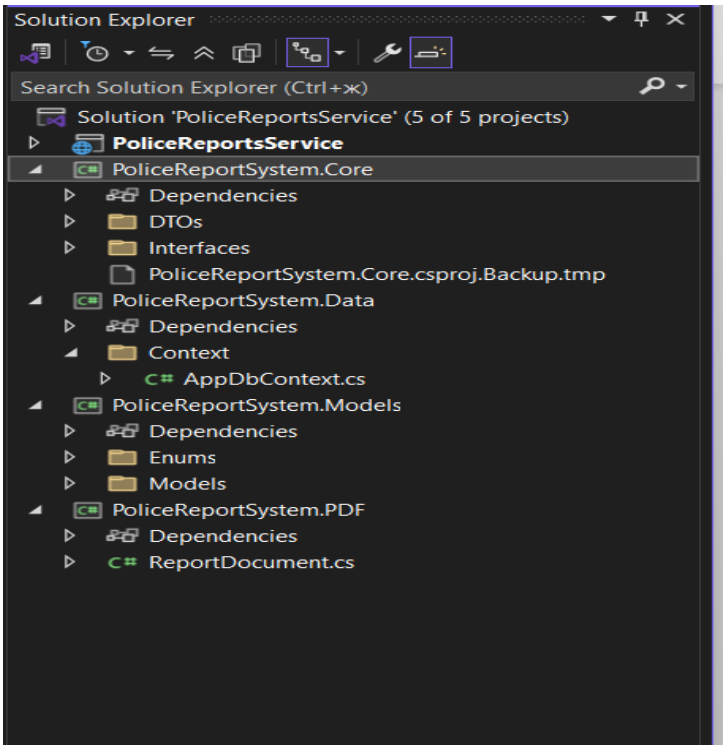


Рис. 2.2. Структурна ієрархія програмного рішення веб-сервісу в середовищі розробки

Детальний аналіз призначення кожного з рівнів системи:

1. Рівень доменних моделей

Центральний, повністю незалежний рівень, що містить опис сутностей предметної області. Включає класи (Officer, Report, AuditLog, Directive), які відображають структуру таблиць бази даних, та строгі перелічення (Enums) для визначення статусів, типів документів і ролей користувачів, що унеможливлює збереження некоректних даних.

2. Інфраструктурний рівень доступу до даних

Відповідає виключно за збереження та вилучення інформації. Головним елементом є клас AppDbContext, реалізований за допомогою ORM Entity Framework Core. Ізоляція логіки роботи з локальною БД SQLite в окремому проєкті дозволяє в майбутньому легко перейти на серверну СУБД (PostgreSQL або MS SQL Server) без зміни бізнес-логіки.

3. Рівень бізнес-логіки

Центральний модуль, що містить правила та алгоритми системи. Складається з об'єктів передачі даних (DTOs) для безпечного обміну інформацією з клієнтом, абстрактних інтерфейсів та класів-сервісів (AuthService, ReportService, StatsService тощо), які реалізують логіку обробки рапортів, доручень, зведення статистики та автентифікації.

4. Інфраструктурний рівень генерації документів

Оскільки процес формування складних PDF-документів є ресурсомістким, цю логіку винесено в ізольований проєкт. Модуль базується на бібліотеці QuestPDF і відповідає за програмну побудову структурованих документів, архітектура яких концептуально спирається на базові вимоги ДСТУ 4163:2020 [3].

5. Презентаційний рівень

Головний вебпроєкт, який забезпечує взаємодію з користувачем за технологією Blazor Server. Включає глобальні макети (Layouts) та Razor-компоненти (Pages), розділені за бізнес-завданнями: сторінки управління звітами, аналітичні дашборди керівника та модулі адміністратора (рис. 2.3).

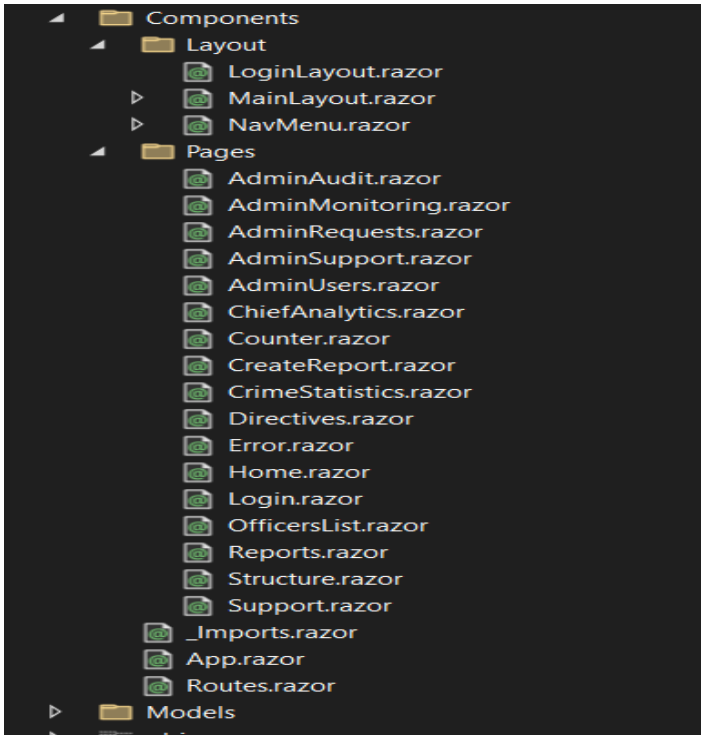


Рис. 2.3. Структура презентаційного рівня (Blazor-компоненти) веб-сервісу

Життєвий цикл обробки запиту та Dependency Injection

Взаємодія між незалежними рівнями відбувається за чітким сценарієм: Презентаційний рівень збирає дані від користувача та передає їх до рівня бізнес-логіки (Core), який валідує інформацію і звертається до рівня Data для операцій з базою даних.

Ключовим механізмом, що об'єднує проєкти в єдину систему, є контейнер впровадження залежностей (Dependency Injection -DI). У файлі Program.cs усі сервіси реєструються із життєвим циклом Scoped. Це гарантує, що компоненти інтерфейсу отримують сервіси автоматично від ядра .NET 8, забезпечуючи надійну ізоляцію оперативних даних кожного окремого користувача (працівника поліції).

Отже, застосування принципів «Чистої архітектури» дозволило створити модульну, захищену та високопродуктивну екосистему, яка повністю відповідає технічним завданням автоматизації процесів Національної поліції.

2.3. Розробка концептуальної та логічної структури бази даних

Основою будь-якої інформаційної системи є надійна, оптимізована та нормалізована база даних. Для збереження оперативної інформації розробленого веб-сервісу було обрано реляційну систему управління базами даних (СУБД) SQLite. Її використання в умовах відокремленого підрозділу поліції є найдоцільнішим, оскільки вона функціонує у режимі Zero-Configuration (не потребує встановлення та адміністрування важкого сервера бази даних) і зберігає всю структуру та дані у єдиному захищеному локальному файлі.

Для проєктування бази даних застосовано сучасний підхід Code-First («Спочатку код») за допомогою технології об'єктно-реляційного відображення Entity Framework Core (EF Core). Цей підхід дозволяє проєктувати структуру таблиць та зв'язки між ними виключно засобами об'єктно-орієнтованої мови

C#, після чого фреймворк автоматично генерує відповідну схему бази даних (міграції). Це виключає необхідність написання SQL-скриптів вручну та мінімізує ризик виникнення синтаксичних помилок.

Логічна модель бази даних (ER-діаграма)

За результатами аналізу предметної області та функціональних вимог було виділено базові інформаційні сутності системи. База даних складається з п'яти основних взаємопов'язаних таблиць: Reports (рапорти), Directives (доручення), AuditLogs (журнал аудиту), ReportStatistic (статистичні дані) та ReportAttachment (додатки до рапорту). Структуру зв'язків між таблицями (Entity-Relationship) відображено на логічній моделі бази даних (рис. 2.4).

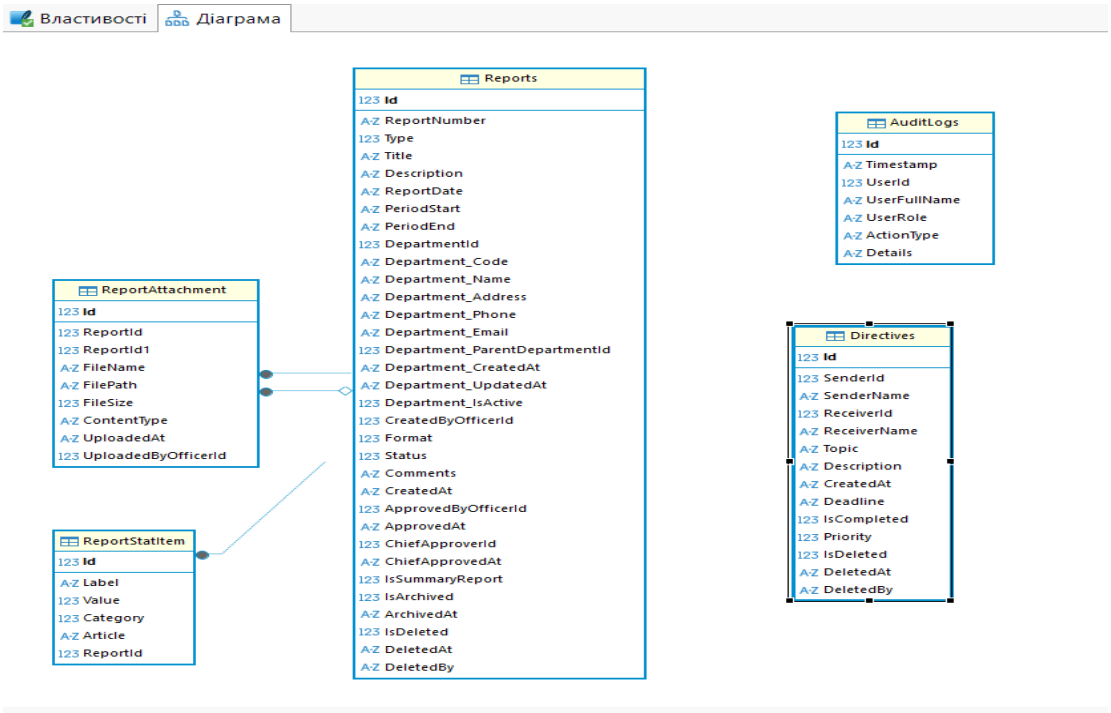
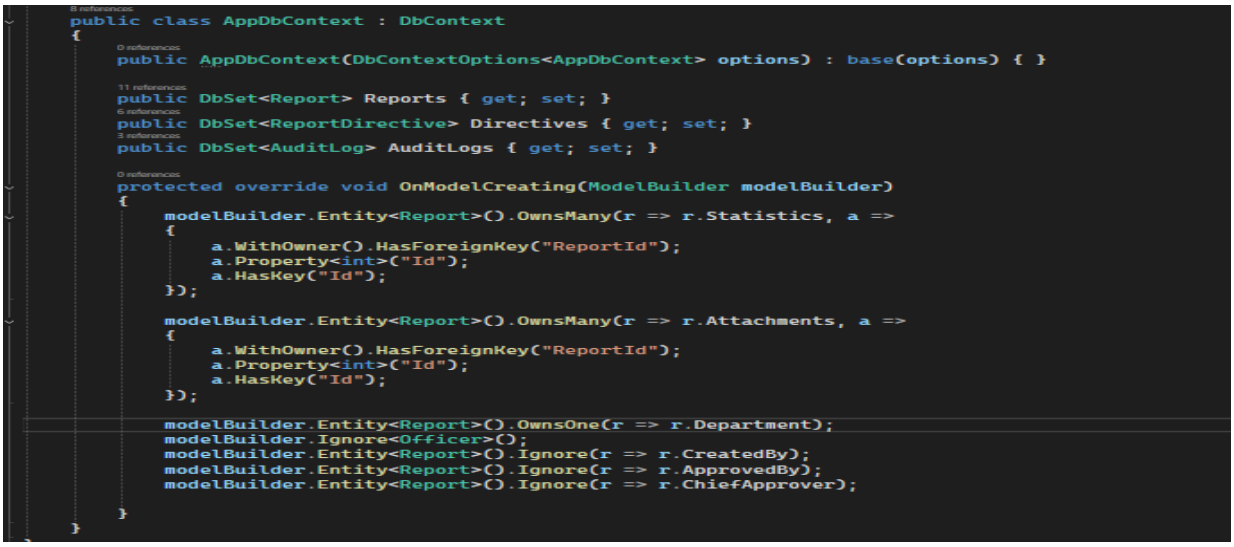


Рис. 2.4. Логічна структура реляційної бази даних веб-сервісу (ER-діаграма)

Налаштування зв'язків та розмежування контекстів

Контекст бази даних реалізовано у класі AppDbContext, де кожній базовій сутності відповідає колекція типу DbSet. Ключовим аспектом проєктування є правильне налаштування відношень між сутностями за допомогою Fluent API у перевизначеному методі OnModelCreating. Фрагмент програмного коду налаштування контексту бази даних наведено на рис. 2.5.



```
public class AppDbContext : DbContext
{
    public AppDbContext(DbContextOptions<AppDbContext> options) : base(options) { }

    public DbSet<Report> Reports { get; set; }
    public DbSet<ReportDirective> Directives { get; set; }
    public DbSet<AuditLog> AuditLogs { get; set; }

    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        modelBuilder.Entity<Report>().OwnsMany(r => r.Statistics, a =>
        {
            a.WithOwner().HasForeignKey("ReportId");
            a.Property<int>("Id");
            a.HasKey("Id");
        });

        modelBuilder.Entity<Report>().OwnsMany(r => r.Attachments, a =>
        {
            a.WithOwner().HasForeignKey("ReportId");
            a.Property<int>("Id");
            a.HasKey("Id");
        });

        modelBuilder.Entity<Report>().OwnsOne(r => r.Department);
        modelBuilder.Ignore<Officer>();
        modelBuilder.Entity<Report>().Ignore(r => r.CreatedBy);
        modelBuilder.Entity<Report>().Ignore(r => r.ApprovedBy);
        modelBuilder.Entity<Report>().Ignore(r => r.ChiefApprover);

    }
}
```

Рис. 2.5. Фрагмент коду налаштування контексту бази даних (клас AppDbContext)

Як видно з рис. 2.5, центральною сутністю системи є рапорт. Оскільки один рапорт може містити декілька статей статистики правопорушень (наприклад, за різними статтями ККУ) та декілька прикріплених файлів, було використано патерн «Належні сутності». Таке рішення вказує EF Core на те, що статистика та додатки не мають власного незалежного життєвого циклу без рапорту. Фреймворк автоматично створює для них окремі підпорядковані таблиці і встановлює зв'язок «один-до-багатьох» через зовнішній ключ. У разі видалення рапорту, каскадно будуть видалені і всі пов'язані з ним підпорядковані дані.

Однією з найважливіших архітектурних особливостей спроєктованої бази даних є ізоляція підсистеми користувачів (Officers) від операційної звітності. У класі `AppDbContext` застосовано явне ігнорування сутності користувача та навігаційних властивостей авторів у рапорті (див. рис. 2.5).

Такий підхід є свідомим використанням принципів предметно-орієнтованого проєктування. Система ідентифікації та управління користувачами винесена в окремий логічний контур. У самій таблиці рапортів зберігаються лише унікальні числові ідентифікатори (ID) авторів та керівників, а детальні дані (ПІБ, посада, звання) підтягуються програмно на рівні бізнес-логіки системи. Це забезпечує слабку зв'язність архітектури: якщо

в майбутньому виникне потреба інтегрувати веб-сервіс із глобальною поліцейською системою авторизації, розробнику не доведеться змінювати структуру таблиць рапортів.

Опис фізичної структури ключових таблиць

Для деталізації структури зберігання інформації розглянемо специфікації основних таблиць бази даних.

Головною таблицею, що акумулює документацію підрозділу, є **Reports** (Рапорти та звіти). Її ключові атрибути:

1) Id (INTEGER, PK) -унікальний первинний ключ документа;

2) Department_Name, Theme, ContentText (TEXT) -назва підрозділу, тема та основна фабула рапорту відповідно;

3) CreationDate (TEXT / DateTime) -дата та час створення;

4) Status, Type (INTEGER) -поточний статус та тип рапорту;

5) CreatorId, ApproverId (INTEGER) -ідентифікатори автора та керівника, який погодив документ.

Поля статусів та типів рапорту зберігаються у вигляді цілих чисел (INTEGER), оскільки на рівні моделей C# вони представлені строгими переліченнями (Enums). Це суттєво зменшує обсяг займаної пам'яті та прискорює пошукові запити у порівнянні зі зберіганням статусу у текстовому форматі.

Критично важливою для забезпечення інформаційної безпеки є таблиця системного аудиту **AuditLogs**, яка містить наступні поля:

1) Id (INTEGER, PK) -унікальний ідентифікатор події;

2) Timestamp (TEXT / DateTime) -точний час фіксації операції;

3) Action (TEXT)-тип виконаної дії (наприклад, "DELETE_REPORT");

4) OfficerId (INTEGER) та OfficerName (TEXT) -ідентифікатор та ПІБ посадової особи;

5) Details (TEXT)-детальний опис операції.

Поле OfficerName навмисно дублює текстові дані (ПІБ особи). Це зроблено для збереження історичної цілісності аудиту: навіть якщо обліковий запис працівника буде назавжди видалено з бази даних, у журналі залишиться чіткий слід про те, хто саме виконував дію в минулому.

Окремий контур бази даних виділено для системи управління завданнями -таблиця **Directives** (Доручення керівництва):

1) Id (INTEGER, PK) -первинний ключ доручення;

2) ReportId (INTEGER, FK) -зовнішній ключ, що посилається на базовий рапорт;

3) ExecutorId (INTEGER)-ідентифікатор працівника-виконавця;

4) Priority (INTEGER) -пріоритет виконання завдання;

5) Deadline (TEXT / DateTime) та TaskDescription (TEXT) -дедлайн і зміст доручення.

Така структура забезпечує повний контроль керівництва над процесом виконання наказів завдяки чіткій прив'язці доручення як до конкретного працівника, так і до первинного документа.

Отже, застосування технології Entity Framework Core дозволило спроєктувати нормалізовану, продуктивну та гнучку базу даних, а використання патернів «Обмеженого контексту» гарантує її адаптивність до можливих змін в IT-інфраструктурі правоохоронних органів у майбутньому.

2.4. Проєктування інтерфейсу користувача (UI/UX)

Проєктування графічного інтерфейсу користувача (User Interface) є одним із найважливіших етапів розробки, оскільки від зручності взаємодії із системою (User Experience) безпосередньо залежить швидкість виконання службових завдань та мінімізація механічних помилок (так званого «людського фактора»).

Враховуючи специфіку роботи поліцейських (зокрема, цілодобові чергування, стресові умови та роботу в приміщеннях зі штучним освітленням),

базовою візуальною концепцією веб-сервісу було обрано «темну тему» (Dark Mode). Такий підхід значно знижує навантаження на зір користувачів під час тривалої роботи з документами. Для реалізації інтерфейсу використано компонентну модель фреймворку Blazor у поєднанні з CSS-бібліотекою Bootstrap. Застосування гнучкої сіткової системи (Grid System) забезпечило повну адаптивність екранів: веб-сервіс коректно відображається як на широкоформатних моніторах у чергових частинах, так і на портативних пристроях (планшетах) керівного складу під час виїздів.

Модуль авторизації та відновлення доступу

Точкою входу до веб-сервісу є модуль авторизації. Екранна форма не містить зайвих елементів, які б відвертали увагу, і складається лише з полів для введення ідентифікаційних даних та кнопки підтвердження. Модуль відновлення доступу спроєктовано у вигляді модального вікна (Modal Dialog). Це дозволяє користувачеві сформувати запит до адміністратора без переходу на інші веб-сторінки, зберігаючи контекст поточної операції.

Робоча панель працівника

Після успішної авторизації працівник (наприклад, слідчий чи інспектор) потрапляє на головну робочу панель. Інтерфейс спроєктовано за принципом інформаційних карток. У лівій частині екрана розміщено блок візуальної ідентифікації з даними авторизованого працівника (ПІБ, посада, жетон, підрозділ). Праворуч розташована «Панель швидких дій» із великими кнопками («Створити рапорт», «Мої звіти»), що дозволяє перейти до найчастіших сценаріїв створення документів в один клік. Для відображення історії документів використано інтерактивні таблиці з кольоровими індикаторами, які допомагають миттєво візуально розрізняти статуси рапортів («Чернетка»-сірий, «На розгляді»-жовтий, «Погоджено»-зелений).

Аналітична панель керівного складу

Інтерфейс користувачів із ролями «Керівник відділу» та «Керівник управління» орієнтований на роботу з великими масивами агрегованих даних. Головним елементом тут виступає аналітичний дашборд. Для зручності

формування звітних періодів інтерфейс оснащено елементами вибору дати (Date Pickers). Статистичні дані візуалізовані у вигляді компактних інформаційних блоків-лічильників, що показують кількість особливо тяжких та тяжких злочинів. Основний масив даних подається у вигляді структурованих таблиць із жорстким розмежуванням правопорушень за статтями Кримінального кодексу (ККУ) та Кодексу про адміністративні правопорушення (КУпАП). Такий підхід забезпечує високу читабельність та дозволяє керівництву швидко оцінювати оперативну обстановку.

Інтерфейс адміністратора

Модуль адміністрування спроєктовано з акцентом на управління списками та моніторинг безпеки. Навігація здійснюється через фіксоване бокове меню. Інтерфейс Журналу аудиту нагадує класичні серверні логи, але адаптовані для комфортного сприйняття: критичні події (наприклад, видалення документів або невдалі спроби входу) виділяються тривожними (червоними) кольорами, що дозволяє адміністратору миттєво реагувати на потенційні загрози.

Підсумовуючи етап проєктування інтерфейсу, варто зазначити, що використання компонентного підходу дозволило створити ергономічний програмний продукт. Обрані візуальні рішення повністю відповідають сучасним вимогам та дозволяють працівникам поліції інтуїтивно працювати з системою без проходження тривалого додаткового навчання.

Висновки до другого розділу

У другому розділі кваліфікаційної роботи здійснено комплексне проєктування архітектури та бази даних розроблюваного інформаційного веб-сервісу для автоматизації процесів звітності у підрозділах Національної поліції України.

На основі аналізу предметної області сформовано детальні функціональні та нефункціональні вимоги до системи. Розроблено багаторівневу рольову модель управління доступом (RBAC), що включає ролі

«Адміністратора», «Працівника», «Керівника відділу» та «Керівника управління». Це забезпечило дотримання принципу найменших привілеїв та ієрархічної підзвітності. Спроєктовано специфікації основних сценаріїв використання (Use Cases), які регламентують життєвий цикл рапорту, процеси погодження та алгоритми безпечного відновлення доступу.

Обґрунтовано використання структурної парадигми «Чистої архітектури» (Clean Architecture). Програмне рішення логічно розділено на п'ять незалежних модулів (доменні моделі, бізнес-логіка, доступ до даних, генерація PDF та презентаційний інтерфейс Blazor Server). Такий підхід забезпечив низьку зв'язність коду, високий рівень захисту від несанкціонованого втручання та підготував систему до можливого подальшого масштабування.

За допомогою технології Entity Framework Core та підходу Code-First спроєктовано оптимізовану реляційну базу даних на платформі SQLite. Застосування патерну «Обмеженого контексту» (Bounded Context) дозволило абстрагувати підсистему авторизації від контуру збереження документів, що гарантує високу адаптивність системи до майбутніх змін в IT-інфраструктурі. Крім того, спроєктовано ергономічний інтерфейс користувача з використанням «темної теми» та адаптивної сітки, що оптимізує щоденну взаємодію поліцейських із системою документообігу.

РОЗДІЛ 3
ПРАКТИЧНА РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОДУКТУ

3.1. Реалізація підсистеми автентифікації та розмежування прав доступу

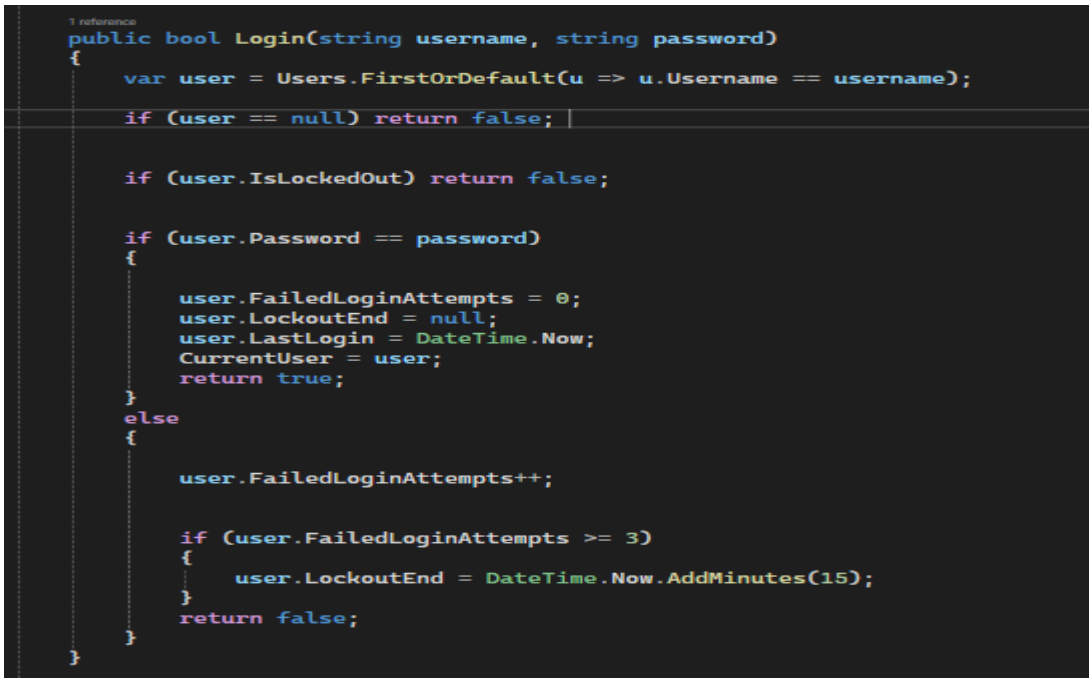
Розробка будь-якої інформаційної системи для потреб правоохоронних органів вимагає впровадження надійних механізмів захисту від несанкціонованого доступу. Згідно з базовими принципами інформаційної безпеки, процес допуску користувача до системи складається з трьох нерозривних етапів: ідентифікації (розпізнавання суб'єкта за його ідентифікатором), автентифікації (перевірки справжності суб'єкта) та авторизації (надання суб'єкту відповідних прав доступу до ресурсів) [4, с. 55].

У розробленому веб-сервісі підсистему безпеки реалізовано на рівні бізнес-логіки (модуль UserService), яка обробляє запити від презентаційного рівня (Blazor-компонентів).

Захист від атак повного перебору (Brute Force)

Однією з найпоширеніших загроз для веб-застосунків є атаки типу Brute Force (метод повного перебору паролів). Для протидії цьому вектору атак у методі Login реалізовано алгоритм автоматичного тимчасового блокування облікового запису (Account Lockout).

Під час кожної спроби входу система спочатку перевіряє статус блокування користувача. Якщо пароль введено неправильно, інкрементується лічильник невдалих спроб (FailedLoginAttempts). У разі досягнення критичної межі (3 невдалі спроби), обліковий запис автоматично блокується на 15 хвилин, а відповідна подія може бути зафіксована у системному моніторингу для адміністратора. Фрагмент програмного коду, що реалізує дану логіку, наведено на рис. 3.1.



```
1 reference
public bool Login(string username, string password)
{
    var user = Users.FirstOrDefault(u => u.Username == username);

    if (user == null) return false;


    if (user.IsLockedOut) return false;


    if (user.Password == password)
    {

        user.FailedLoginAttempts = 0;
        user.LockoutEnd = null;
        user.LastLogin = DateTime.Now;
        CurrentUser = user;
        return true;
    }
    else
    {

        user.FailedLoginAttempts++;


        if (user.FailedLoginAttempts >= 3)
        {
            user.LockoutEnd = DateTime.Now.AddMinutes(15);
        }
        return false;
    }
}
```

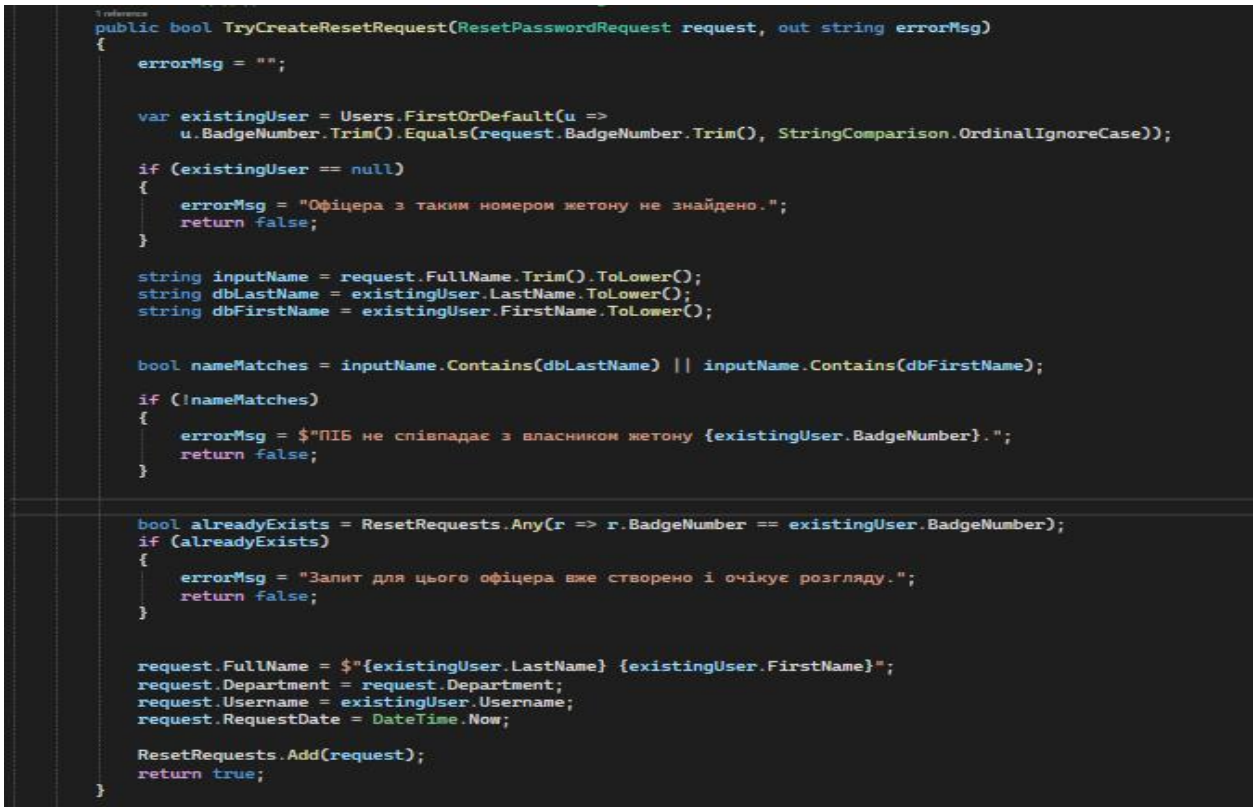
Рис. 3.1. Метод автентифікації із захистом від підбору паролів

У разі легітимного блокування (наприклад, користувач дійсно забув пароль), розроблено механізм ручного розблокування (метод UnlockUser), який доступний виключно користувачам із роллю «Адміністратор».

Безпечний алгоритм відновлення доступу

У випадках втрати автентифікаційних даних користувачем, система не повинна надавати зловмиснику можливість дізнатися, чи існує певний обліковий запис у базі даних (запобігання атакам типу User Enumeration). Тому процес відновлення пароля реалізовано через захищений механізм внутрішніх заявок (Support Tickets), які формуються для адміністратора.

Алгоритм методу TryCreateResetRequest містить комплексну перевірку введених даних. Для успішного формування заявки користувач повинен надати інформацію, яку важко підібрати випадковим чином. Первинна ідентифікація відбувається за унікальним номером жетона поліцейського. Використання методу Trim() та прапорця StringComparison.OrdinalIgnoreCase забезпечує стійкість системи до випадкових пробілів або зміни регістру при введенні (рис. 3.2).



```
public bool TryCreateResetRequest(ResetPasswordRequest request, out string errorMsg)
{
    errorMsg = "";

    var existingUser = Users.FirstOrDefault(u =>
        u.BadgeNumber.Trim().Equals(request.BadgeNumber.Trim(), StringComparison.OrdinalIgnoreCase));

    if (existingUser == null)
    {
        errorMsg = "Офіцера з таким номером жетону не знайдено.";
        return false;
    }

    string inputName = request.FullName.Trim().ToLower();
    string dbLastName = existingUser.LastName.ToLower();
    string dbFirstName = existingUser.FirstName.ToLower();

    bool nameMatches = inputName.Contains(dbLastName) || inputName.Contains(dbFirstName);

    if (!nameMatches)
    {
        errorMsg = $"ПІБ не співпадає з власником жетону {existingUser.BadgeNumber}.";
        return false;
    }

    bool alreadyExists = ResetRequests.Any(r => r.BadgeNumber == existingUser.BadgeNumber);
    if (alreadyExists)
    {
        errorMsg = "Запит для цього офіцера вже створено і очікує розгляду.";
        return false;
    }

    request.FullName = $"{existingUser.LastName} {existingUser.FirstName}";
    request.Department = request.Department;
    request.Username = existingUser.Username;
    request.RequestDate = DateTime.Now;

    ResetRequests.Add(request);
    return true;
}
```

Рис. 3.2. Алгоритм валідації запиту на відновлення пароля

Такий алгоритм є оптимальним з точки зору UX (користувацького досвіду) та безпеки. Гнучка перевірка ПІБ дозволяє прийняти заявку, якщо користувач ввів лише своє прізвище або зробив незначну помилку в ініціалах, але точний збіг жетона гарантує, що запит робить саме власник облікового запису. Крім того, перевірка на дублювання ефективно захищає панель адміністратора від спам-атак.

Взаємодія з інтерфейсом користувача (UI)

Презентаційний рівень підсистеми автентифікації реалізовано у вигляді окремого Razor-компонента, який використовує ізольований макет сторінки (без бокового меню навігації). Візуальне відображення модуля авторизації (рис. 3.3) виконано в мінімалістичному стилі з використанням «темної теми».

Завдяки двосторонньому зв'язуванню даних у фреймворку Blazor, поля форми автоматично синхронізуються з властивостями моделі LoginInputModel.

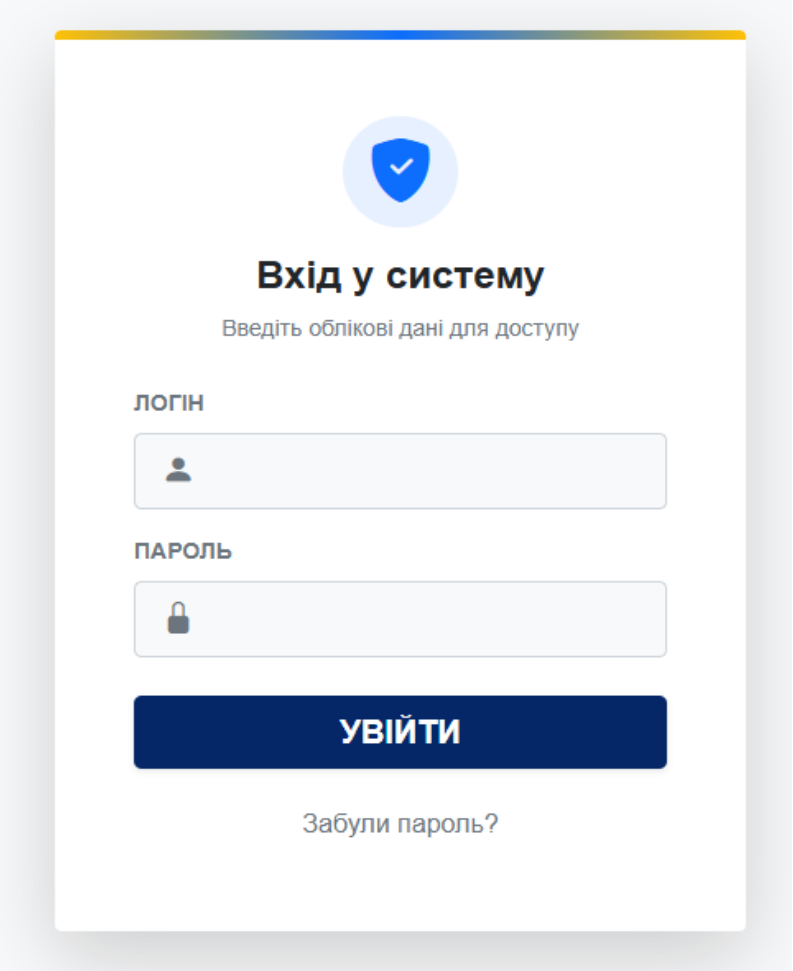


Рис. 3.3. Інтерфейс модуля авторизації користувачів веб-сервісу

Обробка результатів автентифікації відбувається асинхронно. У разі успішного проходження всіх етапів перевірок у сервісі UserService, компонент NavigationManager миттєво перенаправляє користувача на захищену сторінку робочої панелі. У разі виникнення помилок (невірний пароль або активне блокування), система виводить уніфіковане повідомлення «Невірний логін або пароль», не розкриваючи зловмиснику причину збою.

У разі ініціації користувачем процедури відновлення доступу, система не перенаправляє його на іншу сторінку, а викликає інтерактивне модальне вікно (рис. 3.4).

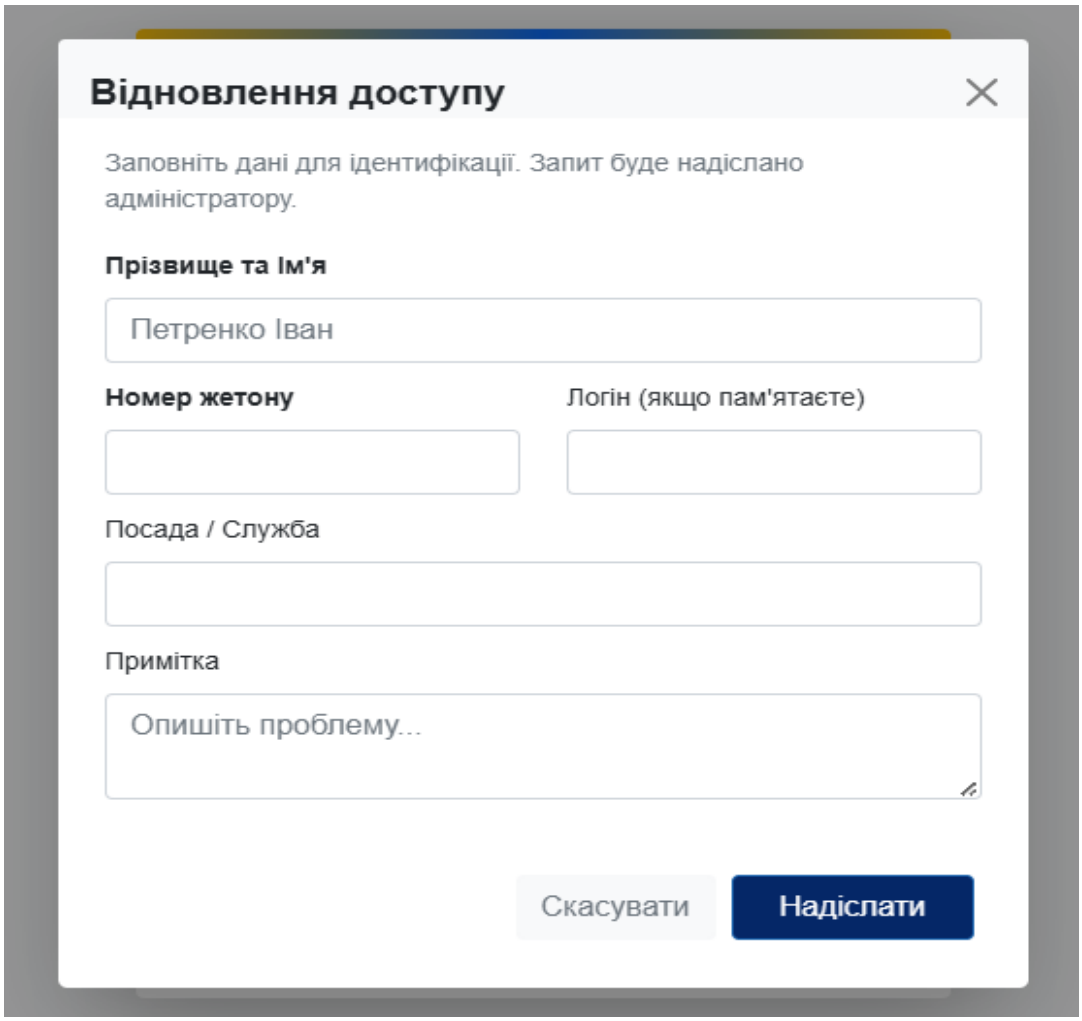


Рис. 3.4. Модальне вікно формування заявки на відновлення доступу

Інтерфейс вікна вимагає введення ідентифікаційних даних (ПІБ, номер жетона, посада), які безпосередньо передаються в описаний раніше алгоритм валідації (метод TryCreateResetRequest). У разі успішної перевірки заявка стає доступною в системному інтерфейсі Адміністратора, а модальне вікно закривається з повідомленням про успішне відправлення запиту.

Реалізація модулів ідентифікації, автентифікації та захищеного відновлення доступу як на рівні серверної бізнес-логіки, так і на рівні клієнтського інтерфейсу, дозволила створити надійний бар'єр, який гарантує конфіденційність та цілісність службових даних у поліцейському веб-сервісі.

3.2. Розробка модуля створення та погодження рапортів із функцією попереднього перегляду

Центральним компонентом розробленого веб-сервісу, який безпосередньо вирішує завдання автоматизації повсякденного діловодства, є модуль створення поліцейських рапортів. Традиційний процес формування звітності передбачає ручне редагування шаблонів у текстових процесорах. Це часто призводить до порушення єдиної візуальної структури документів, людських помилок при копіюванні та значної втрати робочого часу на механічне заповнення багаторазових реквізитів. Для усунення цих недоліків розроблено модуль, який бере на себе завдання автоматизованого збору даних, їх структуризації та швидкої програмної генерації кінцевого документа [18].

Інтерфейс створення документа (UI) та Live Preview

Клієнтську частину модуля реалізовано у компоненті CreateReport.razor за допомогою вбудованого у фреймворк Blazor механізму EditForm, який забезпечує миттєву валідацію введених даних. Інтерфейс форми створення рапорту спроєктовано у вигляді зручного двопанельного редактора з функцією попереднього перегляду в реальному часі (Live Preview) (рис. 3.5).

Значною ергономічною перевагою розробленого модуля є те, що всі ідентифікаційні та системні поля (ПІБ автора, спеціальне звання, найменування підрозділу) заповнюються системою автоматично на основі сесії авторизованого користувача. Працівнику поліції для формування рапорту необхідно заповнити лише ключові атрибути події:

1) **Тип документу:** Обирається з переліченння ReportType (Денний звіт оперативний, Тижневий, Місячний, Квартальний, Річний, Інцидент, Статистична довідка або Подія).

2) **Формат подання:** Визначає структуру документа згідно з переліченням ReportFormat (Письмовий текстовий, Статистичний або Комбінований).

3) **Дата складання, Тема рапорту та Текст:** Текстові блоки для опису безпосередньої фабули правопорушення або результатів службової діяльності.

Завдяки реактивній природі Blazor, будь-які зміни, внесені працівником у поля форми зліва, миттєво відображаються на інтерактивній панелі «Попередній перегляд (PDF)» справа. Це дозволяє користувачеві візуально контролювати форматування та розміщення реквізитів ще до збереження документа в базу даних.

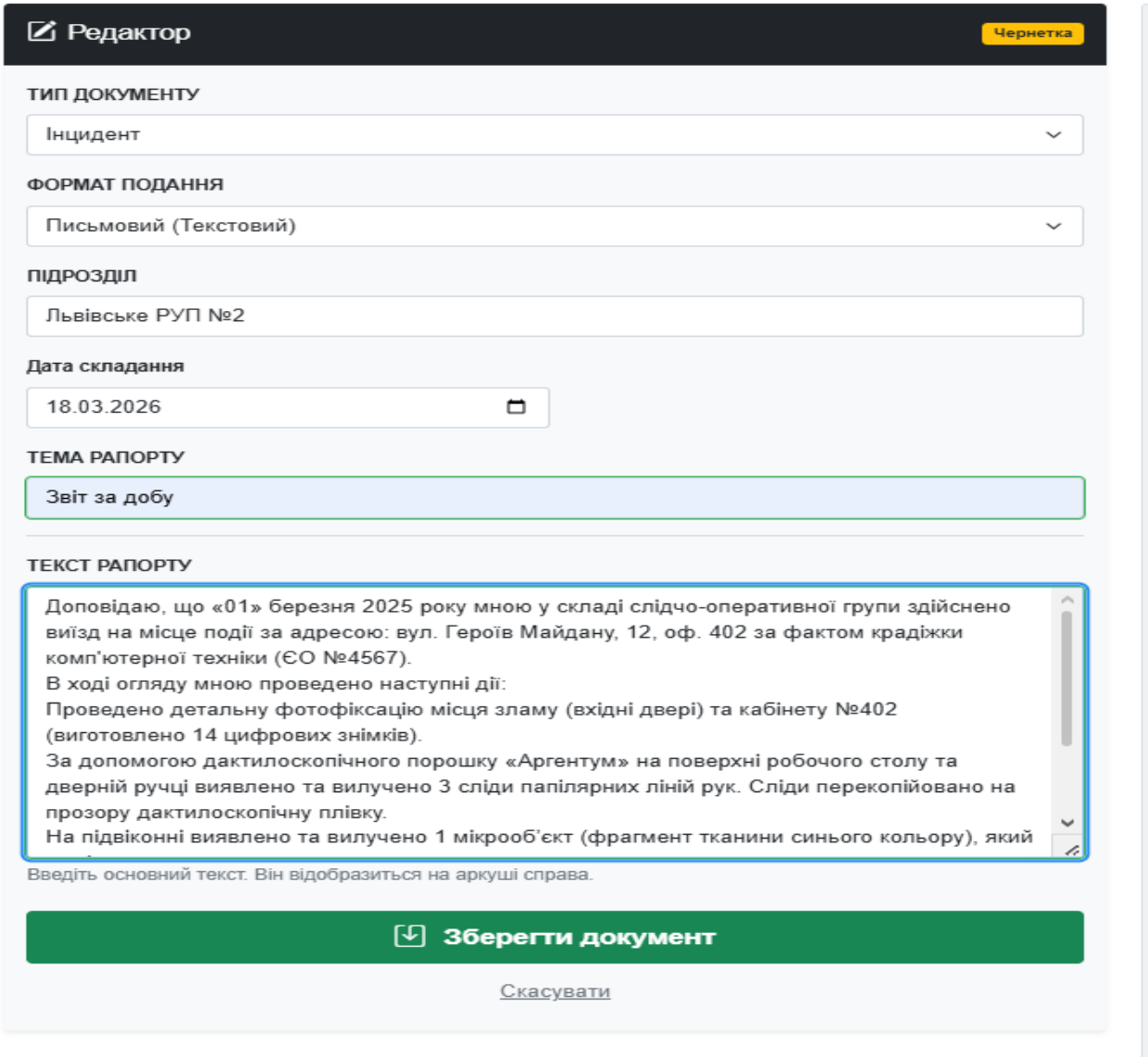


Рис. 3.5. Інтерфейс форми створення нового поліцейського рапорту

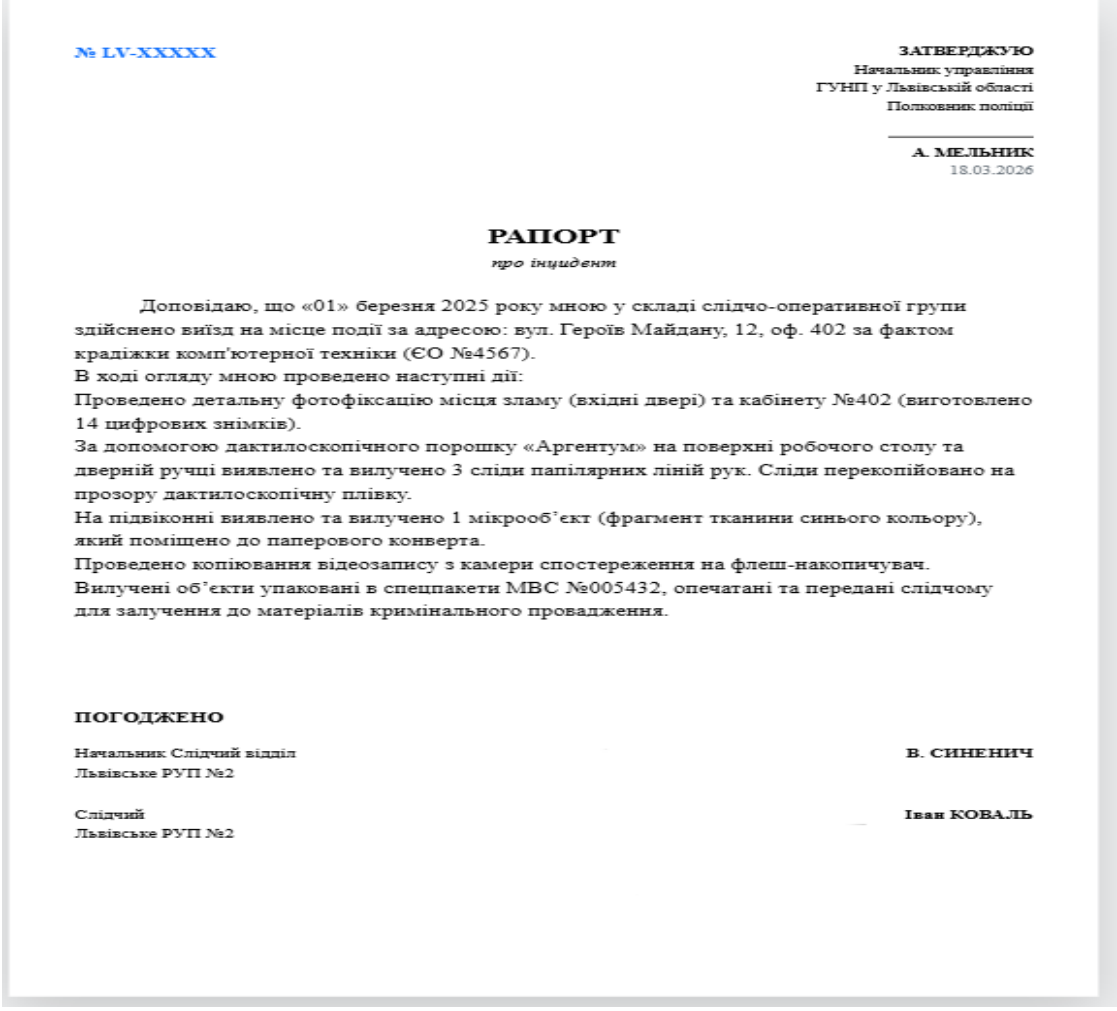
Логіка модуля також враховує ієрархічну належність автора: якщо рапорт створює керівник (IsChief = true), документ одразу після збереження

отримує статус Approved (Затверджено). Якщо ж документ створює звичайний працівник, йому автоматично присвоюється первинний статус.

Функція попереднього перегляду (Live Preview)

Інноваційною особливістю розробленого модуля є наявність панелі попереднього перегляду (Live Preview), яка розташована праворуч від форми вводу. Генерація макета відбувається повністю в оперативній пам'яті сервера без створення тимчасових файлів на диску.

Завдяки реактивній природі фреймворку Blazor, працівник може візуально оцінити згенерований макет документа безпосередньо в інтерфейсі веб-сервісу ще до його остаточного збереження або відправки керівництву (рис. 3.6). Це дозволяє завчасно виявити орфографічні помилки або некоректне форматування тексту.



№ LV-XXXXX

ЗАТВЕРДЖУЮ
Начальник управління
ГУНП у Львівській області
Полковник поліції

_____
А. МЕЛЬНИК
18.03.2026

РАПОРТ
про інцидент

Доповідаю, що «01» березня 2025 року мною у складі слідчо-оперативної групи здійснено виїзд на місце події за адресою: вул. Героїв Майдану, 12, оф. 402 за фактом крадіжки комп'ютерної техніки (ЄО №4567).
В ході огляду мною проведено наступні дії:
Проведено детальну фотофіксацію місця зламу (вхідні двері) та кабінету №402 (виготовлено 14 цифрових знімків).
За допомогою дактилоскопічного порошку «Аргентум» на поверхні робочого столу та дверній ручці виявлено та вилучено 3 сліди папілярних ліній рук. Сліди перекопійовано на прозору дактилоскопічну плівку.
На підвіконні виявлено та вилучено 1 мікрооб'єкт (фрагмент тканини синього кольору), який поміщено до паперового конверта.
Проведено копіювання відеозапису з камери спостереження на флеш-накопичувач.
Вилучені об'єкти упаковані в спецпакети МВС №005432, опечатані та передані слідчому для залучення до матеріалів кримінального провадження.

ПОГОДЖЕНО

Начальник Слідчий відділ
Львівське РУП №2 — В. СИНЕНИЧ

Слідчий
Львівське РУП №2 — Іван КОВАЛЬ

Рис. 3.6. Інтерфейс попереднього перегляду (Live Preview) згенерованого макета

Програмна генерація PDF та завантаження на клієнті

Найбільш ресурсомісткою з інженерної точки зору задачею є трансформація введеного тексту в офіційний PDF-документ. Замість використання застарілих інструментів конвертації HTML-коду, у проєкті застосовано нативну C# бібліотеку QuestPDF. Вона використовує підхід Fluent API, що дозволяє описувати макет документа (розміри сторінки, відступи, шрифти) суворим програмним кодом.

Фрагмент реального коду класу ReportDocument.cs, який реалізує інтерфейс IDocument та формує структуру сторінки, наведено на рис. 3.7.

```
public class ReportDocument : IDocument
{
    public Report ReportModel { get; }

    public ReportDocument(Report report)
    {
        ReportModel = report;
    }

    public byte[] GetPdfBytes() => this.GeneratePdf();

    public DocumentMetadata GetMetadata() => DocumentMetadata.Default;

    public void Compose(IDocumentContainer container)
    {
        QuestPDF.Settings.License = LicenseType.Community;

        container.Page(page =>
        {
            page.Margin(40);
            page.DefaultTextStyle(x => x.FontSize(12).FontFamily(Fonts.TimesNewRoman));

            page.Header().Element(ComposeHeader);
            page.Content().Element(ComposeContent);
            page.Footer().AlignCenter().Text(x =>
            {
                x.Span("Стор. ");
                x.CurrentPageNumber();
            });
        });
    }
```

Рис. 3.7. Програмна генерація макета документа засобами QuestPDF

Для передачі кінцевого масиву байтів у браузер користувача використано механізм взаємодії Blazor із мовою JavaScript, що показано на рис. 3.8.

```
private async Task DownloadPdf(Report report)
{
    try
    {
        var pdfBytes = ReportService.GenerateReportPdf(report.Id);
        using var fileStream = new MemoryStream(pdfBytes);
        using var streamRef = new DotNetStreamReference(fileStream);
        await JS.InvokeVoidAsync("downloadFileFromStream", $"{report.ReportNumber}.pdf", streamRef);
    }
    catch (Exception ex)
    {
        await JS.InvokeVoidAsync("alert", $"Помилка PDF: {ex.Message}");
    }
}
```

Рис. 3.8. Асинхронне завантаження PDF-файлу на клієнтський пристрій

Сформований за цим алгоритмом кінцевий PDF-документ має чітку структуру, містить усі необхідні реквізити, резолюції керівництва та повністю готовий до друку або подальшого цифрового обігу.

Варто зазначити, що реальна документація підрозділів Національної поліції України (рапорти, фабули справ, статистика) містить службову інформацію та персональні дані з обмеженим доступом. З огляду на це, усі наведені в даній кваліфікаційній роботі візуальні зразки згенерованих PDF-документів (зокрема, на рис. 3.9) є абстрактними ілюстративними макетами. Дані, що використовуються для їх формування (ПІБ офіцерів, фабули інцидентів, назви підрозділів), є повністю фіктивними і створені виключно з метою демонстрації коректності роботи алгоритмів веб-сервісу.

Рис. 3.9. Результат програмної генерації: кінцевий PDF-документ поліцейського рапорту

Маршрутизація та зміна статусів (Workflow)

Життєвий цикл документа в системі суворо регламентований. У стані «Чернетка» автор має повні права на редагування тексту. Після натискання кнопки подачі, статус запису в базі даних змінюється на «Очікує погодження». З цього моменту документ блокується для редагування автором і стає видимим у робочій панелі керівника відповідного підрозділу, який має змогу його перевірити, погодити та передати на остаточне затвердження начальнику управління. Усі етапи зміни статусів автоматично фіксуються у журналі аудиту.

3.3. Реалізація підсистеми контролю доручень та системного журналу аудиту

Сучасні інформаційні системи правоохоронних органів вимагають не лише автоматизації створення звітності, але й забезпечення надійних інструментів для внутрішньої комунікації, постановки завдань та жорсткого контролю за діями персоналу. Для вирішення цих завдань у розробленому веб-сервісі реалізовано модуль системного аудиту з використанням патерну «м'якого видалення» та підсистему електронних службових доручень.

Системний аудит та безпечне видалення даних (Soft Delete)

Критичною вимогою до поліцейських інформаційних систем є неможливість безповоротного знищення електронних доказів або службових документів. Навіть якщо документ має статус «Чернетка», фізичне видалення запису з бази даних оператором (командою DELETE в SQL) є порушенням протоколів безпеки.

Для вирішення цієї проблеми у системі реалізовано архітектурний патерн «М'якого видалення» (Soft Deletion). Замість фізичного видалення рядка з таблиці SQLite, система використовує спеціальні аудиторські поля: IsDeleted, DeletedAt та DeletedBy.

Коли користувач ініціює видалення документа, він зникає з його робочого екрана, проте інформація залишається в базі даних. Користувач із роллю «Адміністратор» має доступ до спеціального модуля «Панель журналу аудиту безпеки» (рис. 3.10).

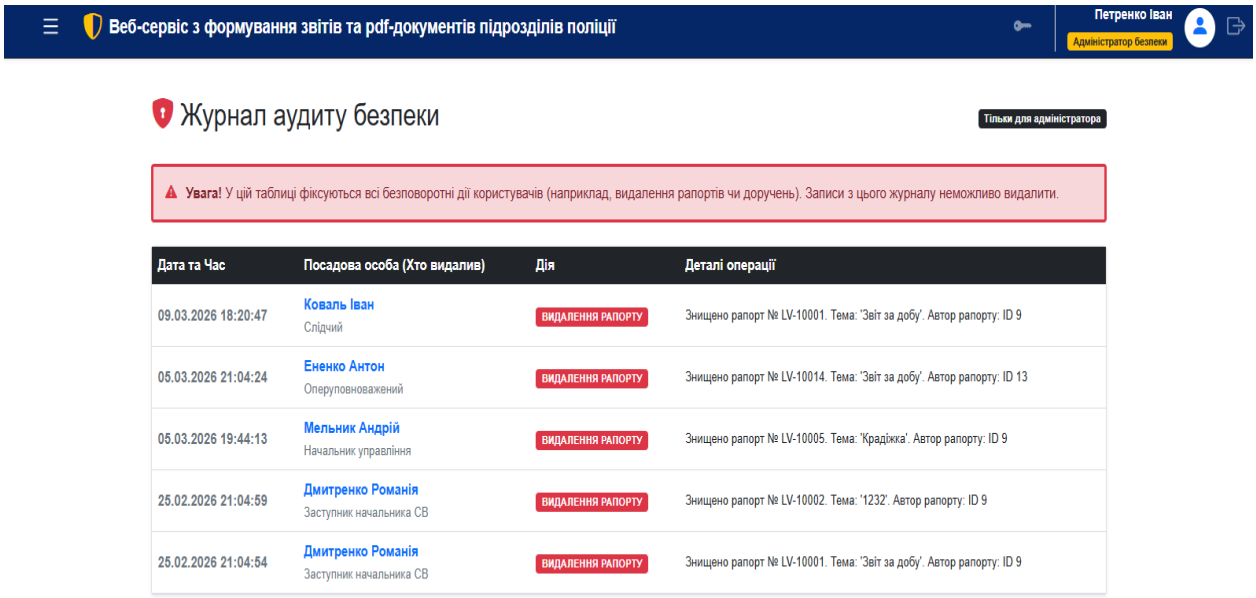


Рис. 3.10. Інтерфейс модуля системного аудиту та моніторингу безпеки

Цей інтерфейс виводить зведену таблицю всіх прихованих записів із зазначенням точної дати, ПІБ та посади особи, яка здійснила операцію, а також детальний опис видаленого об'єкта (наприклад, тема рапорту або доручення). Цей механізм гарантує повну підзвітність дій персоналу та захищає систему від випадкового або навмисного знищення службової інформації.

Архітектура та ієрархія підсистеми доручень

Для забезпечення вертикальної комунікації між керівним складом та підлеглими розроблено модуль доручень. Інформаційна сутність доручення реалізована у класі ReportDirective, який містить дані ініціатора, виконавця, терміни виконання, текстовий опис та рівень пріоритетності (DirectivePriority: Звичайний, Важливий, Терміново).

Особливістю розробленого модуля є алгоритмічне дотримання суворої субординації Національної поліції. Логіку формування списку підлеглих реалізовано на рівні презентаційного компонента Directives.razor (рис. 3.11).

```
private void LoadData()
{
    var user = UserService.CurrentUser;
    if (user == null) return;


    Inbox = DirectiveService.GetInboxForUser(user.Id);
    if (CanGiveDirectives)
    {
        SentDirectives = DirectiveService.GetSentByUser(user.Id);

        if (user.IsChief)
        {
            Subordinates = UserService.Users.Where(u => u.CanApprove && !u.IsChief).ToList();
        }
        else if (user.CanApprove)
        {
            Subordinates = UserService.Users.Where(u => u.Unit == user.Unit && !u.CanApprove).ToList();
        }
    }
}
```

Рис. 3.11. Алгоритм розмежування підлеглих згідно з посадовою ієрархією

Завдяки цьому алгоритму начальник управління не призначає завдання напряму рядовим слідчим (минаючи їхнього безпосереднього керівника), а керівник сектору не має доступу до працівників інших структурних підрозділів.

Клієнтський інтерфейс управління завданнями

Робочий простір модуля доручень (рис. 3.12) розділено на два основні візуальні блоки: «Вхідні завдання» (для виконання) та «Видані доручення» (для контролю підлеглих). Кожне завдання відображається у вигляді інформаційної картки, що містить деталізовану інформацію про ініціатора, дедлайн та кольорові індикатори пріоритетності (наприклад, статус «ТЕРМІНОВО» виділяється червоним кольором та спеціальною іконкою для максимального привернення уваги).

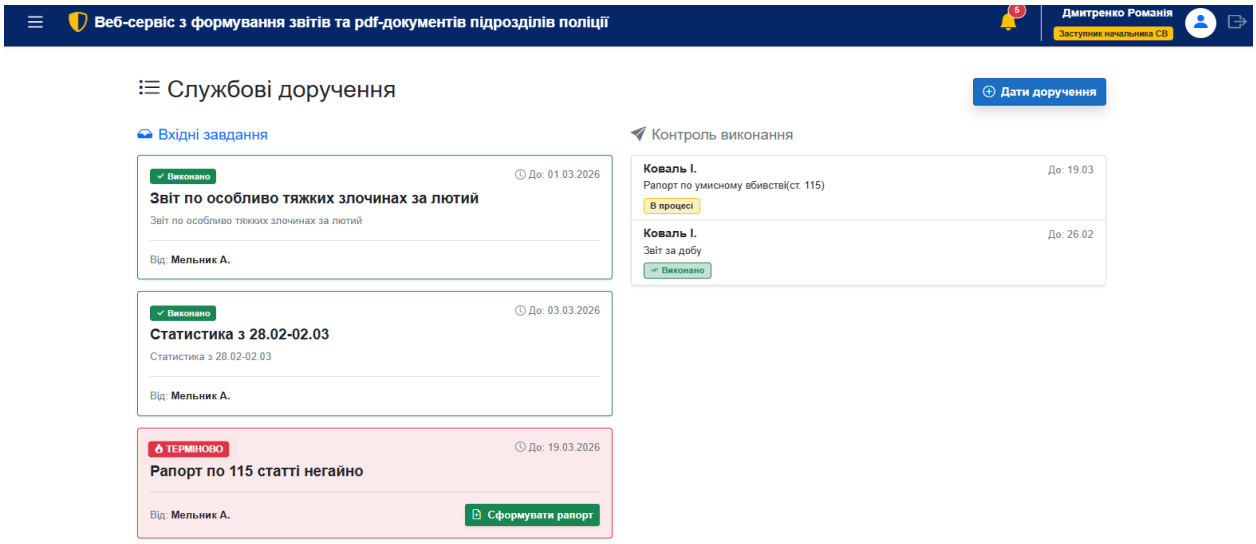


Рис. 3.12. Робоча панель підсистеми контролю службових доручень

Процес створення нового доручення керівником відбувається у модальному вікні (рис. 3.13). Форма вимагає вибору виконавця з відфільтрованого списку підлеглих, встановлення дедлайну, пріоритету та детального опису завдання. Використання модального вікна дозволяє швидко сформувати наказ без переходу на інші сторінки системи.

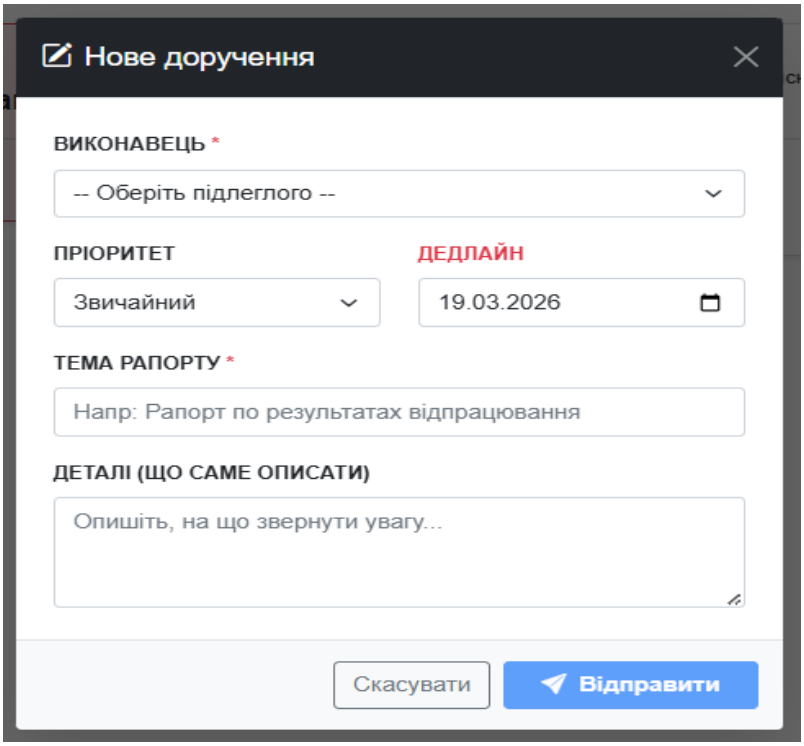


Рис. 3.13. Інтерфейс форми створення нового доручення підлеглому

Життєвий цикл завдання жорстко контролюється системою статусів. Коли працівник отримує вхідне доручення, він має змогу ознайомитися з його деталями та, за необхідності, одразу перейти до створення відповідного рапорту за допомогою кнопки «Написати рапорт». Після фактичного виконання поставленої задачі, підлеглий натискає кнопку «Виконано». У цей момент на рівні бази даних булевий прапорець IsCompleted змінюється на true, а саме доручення візуально переміщується до архіву виконаних завдань. Така архітектура модуля забезпечує прозорий контроль виконавчої дисципліни та повністю виключає можливість втрати службових наказів.

3.4. Розробка модулів оперативної статистики та аналітики ефективності персоналу

Ефективне управління підрозділами Національної поліції неможливе без оперативного аналізу великих масивів даних. Для прийняття обґрунтованих управлінських рішень керівний склад повинен мати миттєвий доступ до актуальної статистики злочинності та об'єктивних показників ефективності роботи підлеглих. Для забезпечення цієї потреби у веб-сервісі розроблено два незалежні аналітичні дашборди.

Аналітика ефективності персоналу

Модуль аналітики роботи персоналу призначений для візуалізації показників документообігу та контролю навантаження на працівників. Інтерфейс екрана (рис. 3.14) містить інформаційні картки із загальною кількістю створених та погоджених рапортів, а також детальну таблицю індивідуальної ефективності кожного офіцера.

Таблиця ефективності автоматично групує відфільтровані звіти за ідентифікатором автора та підраховує кількість документів у стані «Відхилено», «На розгляді» та «Затверджено».

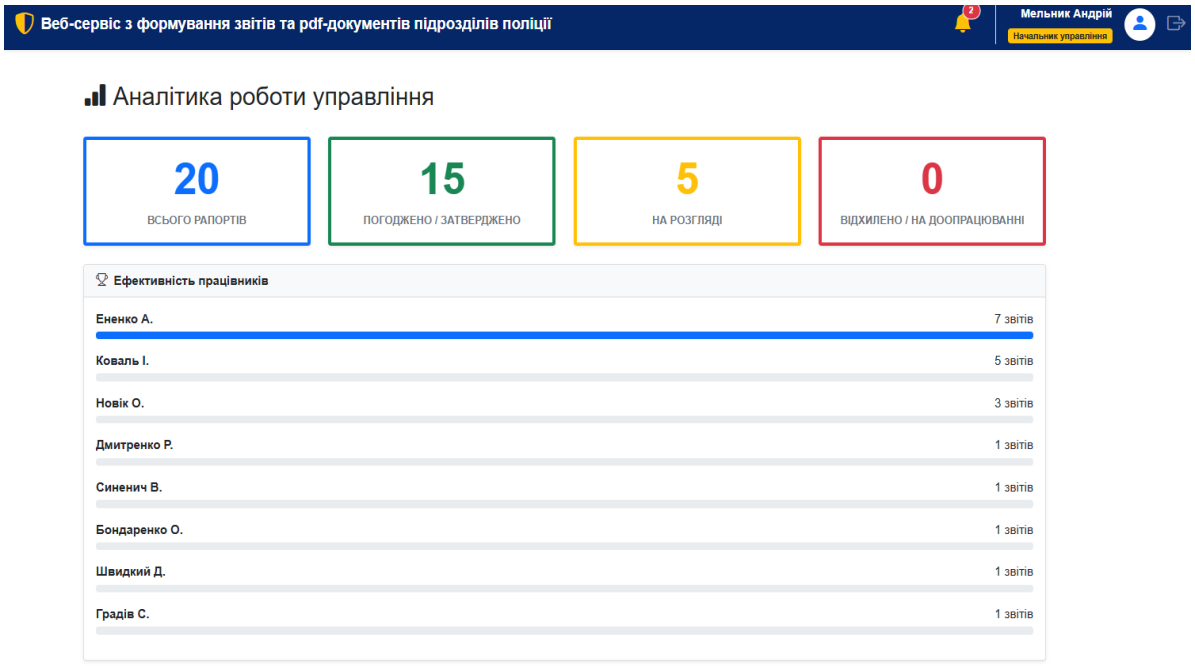


Рис. 3.14. Інтерфейс модуля аналітики ефективності роботи персоналу

Важливою архітектурною особливістю цього модуля є алгоритмічне дотримання поліцейської ієрархії при вибірці даних. Для фільтрації масивів інформації використано технологію інтегрованих запитів LINQ (Language-Integrated Query) [9]. Залежно від ролі авторизованого користувача, система динамічно формує набір даних: керівник управління отримує повний зріз по всій базі даних, тоді як керівник відділу бачить статистику виключно своїх підлеглих. Фрагмент коду, що реалізує цю логіку, наведено на рис. 3.15.

```
if (IsBigBoss)
{

    FilteredReports = allReports;
}
else
{
    FilteredReports = allReports.Where(r =>
    {

        var author = UserService.Users.FirstOrDefault(u => u.Id == r.CreatedByOfficerId);
        return author != null && author.Unit == user.Unit;
    }).ToList();
}
```

Рис. 3.15. Фільтрація звітності згідно з посадовою ієрархією засобами LINQ

Інтерактивний дашборд оперативної статистики

Другим критично важливим інструментом для керівництва є модуль оперативної статистики правопорушень (рис. 3.16). Коли працівники поліції

створюють статистичні рапорти, вони вносять кількісні показники за певними статтями (Кримінального кодексу України або КУпАП). Усі ці записи зберігаються у базі даних (у підпорядкованій таблиці ReportStatistic) із прив'язкою до дати події.

Інтерфейс дашборду дозволяє керівнику обрати будь-який часовий проміжок за допомогою фільтрів «Період з» та «Період по». При зміні дат система миттєво агрегує дані з усіх наявних звітів та виводить зведену інформацію (наприклад, загальну кількість розкритих особливо тяжких злочинів або складених адмінпротоколів за місяць).

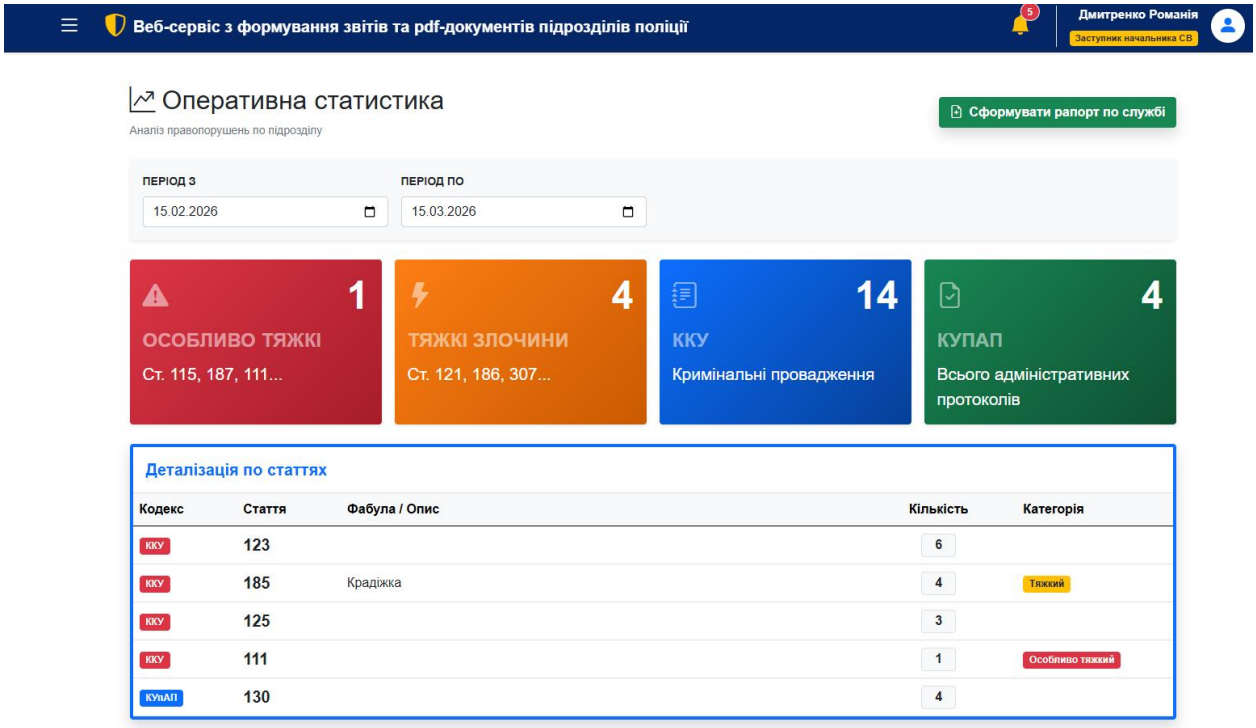


Рис. 3.16. Аналітичний дашборд оперативної статистики правопорушень

Зручною функцією цього модуля є можливість автоматичної генерації зведеного звіту. Натисканням кнопки «Сформувати рапорт по службі» керівник ініціює створення нового офіційного документа, який заповнюється підрахованою на екрані статистикою. Для забезпечення цілісності бази даних та уникнення задвоєння статистичних показників, такому зведеному рапорту програмно присвоюється маркер. Цей підхід гарантує, що під час формування річних звітів статистика зі зведених документів не буде додана до первинних показників.

Впровадження аналітичних дашбордів дозволило перетворити систему документообігу на повноцінний інструмент підтримки прийняття управлінських рішень, що суттєво підвищує загальну ефективність роботи поліцейського підрозділу.

Висновки до третього розділу

У третьому розділі висвітлено процес практичної реалізації поліцейського веб-сервісу на платформі .NET 8 та Blazor Server. Розроблено надійну підсистему автентифікації, яка включає захист від атак повного перебору (Brute Force) та безпечний механізм відновлення доступу за унікальним номером жетона працівника.

Реалізовано центральний модуль електронного документообігу для автоматизованого створення рапортів. Інтеграція бібліотеки QuestPDF дозволила генерувати уніфіковані макети PDF-документів на основі введених користувачем даних. Крім того, розроблено функцію інтерактивного попереднього перегляду (Live Preview), що відображає документ у реальному часі без необхідності збереження тимчасових файлів на сервері.

Впроваджено підсистему службових доручень із вбудованим контролем поліцейської ієрархії та патерном «м'якого видалення» (Soft Delete) для забезпечення безперервного системного аудиту. Крім того, розроблено інтерактивні аналітичні дашборди для керівництва, де за допомогою технології LINQ реалізовано агрегацію статистики правопорушень (за ККУ та КУпАП) і оцінку ефективності персоналу з можливістю автоматичного формування зведених звітів.

РОЗДІЛ 4
ТЕСТУВАННЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ РОБОТИ СИСТЕМИ

4.1. Вибір методів та середовища тестування

Етап тестування є критично важливою складовою розробки, головною метою якого є перевірка відповідності розробленого веб-сервісу висунутим вимогам та виявлення дефектів.

Методологія тестування

Враховуючи архітектурні особливості веб-сервісу(Blazor Server та рольова модель доступу), для комплексної перевірки було обрано такі методи:

1. **Функціональне тестування («чорний ящик»).** Тестувальник взаємодіє виключно з графічним інтерфейсом, перевіряючи коректність бізнес-логіки: створення рапортів, генерації PDF-документів та роботи дашбордів.
2. **Тестування безпеки та контролю доступу.** Перевірка ієрархічної рольової моделі (RBAC) на предмет неможливості несанкціонованого доступу до даних інших відділів, а також тестування алгоритмів захисту від підбору пароля (Brute Force).
3. **Тестування сумісності.** Перевірка коректності відображення адаптивного інтерфейсу на різних пристроях (від ПК до планшетів).

Опис середовища тестування

Для забезпечення чистоти експерименту та ізоляції тестових даних було налаштовано локальне тестове середовище на базі ОС Windows 11 із вбудованим веб-сервером .NET 8 Runtime.

База даних: Для тестування використовувався ізольований файл БД SQLite (police_test.db), автоматично згенерований механізмами Entity Framework Core. Для якісної перевірки аналітичних модулів до бази було завантажено тестовий набір: 5 облікових записів офіцерів різних рангів та 50 згенерованих рапортів.

Клієнтське середовище: Перевірка коректності завантаження згенерованих PDF-файлів та загальної адаптивності проводилася у найпопулярніших сучасних веб-браузерах: Google Chrome, Microsoft Edge, Mozilla Firefox.

Обрані методи та налаштоване середовище дозволили провести всебічну перевірку розробленого продукту, імітуючи реальні умови його експлуатації в підрозділах Національної поліції України.

4.2. Функціональне тестування основних модулів веб-сервісу

Для підтвердження працездатності розробленого програмного продукту та його готовності до впровадження в реальні умови діяльності підрозділів Національної поліції, було проведено серію функціональних тестів (Black-Box Testing). Основна увага приділялася перевірці критичних бізнес-процесів: підсистемі автентифікації, генерації звітності та модулю дотримання поліцейської ієрархії при роботі з дорученнями [7].

Процес тестування формалізовано за допомогою сценаріїв використання (Test Cases). Кожен сценарій містить опис ініціюючої дії, очікуваний відгук системи та фактичний результат, отриманий під час прогону тесту на локальному сервері.

Перший етап тестування спрямований на перевірку стійкості системи до несанкціонованого вступу та коректності роботи механізмів захисту від атак повного перебору (Brute Force) табл. 4.1.

Таблиця 4.1

Сценарії тестування модуля авторизації

№	Дія користувача (Test Steps)	Очікуваний результат (Expected)	Фактичний результат (Actual)	Статус
1	Введення правильного логіна та пароля зареєстрованого офіцера.	Успішна авторизація, перенаправлення на сторінку Dashboard.	Користувача перенаправлено на /home, сесія створена.	**Успішно**
2	Триразове введення неправильного пароля для існуючого логіна.	Автоматичне блокування облікового запису на 15 хвилин.	Після 3-ї спроби система відхиляє доступ, прапорець	**Успішно**
3	Створення заявки на відновлення пароля із введенням неіснуючого жетона.	Відмова у формуванні заявки, виведення помилки "Офіцера не знайдено".	Заявку не створено. Виведено відповідне попередження на UI.	**Успішно**
4	Спроба звичайного працівника перейти за прямим URL на сторінку адміністрування.	Блокування доступу механізмом авторизації Blazor.	Доступ заборонено, інтерфейс не завантажується.	**Успішно**

Джерело: розроблено автором.

Другий етап тестування перевіряє ядро системи -створення рапортів та програмну генерацію офіційних документів у форматі PDF за допомогою бібліотеки QuestPDF. Сценарії тестування цього модуля відображено у табл. 4.2.

Таблиця 4.2

Сценарії тестування генерації рапортів та процесу погодження

№	Дія користувача	Очікуваний результат	Фактичний результат	Статус
1	Введення даних у поля форми створення нового рапорту.	Динамічне оновлення панелі попереднього перегляду (Live Preview) згенерованим макетом PDF у режимі реального часу.	PDF відображається в панелі прев'ю та миттєво реагує на зміни без збереження на диск.	Успішно
2	Натискання кнопки "Зберегти" після перевірки макета.	Документ зберігається в БД зі статусом "Чернетка". Для автора залишається доступною функція видалення.	Рапорт з'явився в реєстрі зі статусом "Чернетка", кнопка видалення активна.	Успішно
3	Керівник сектору (служби) відкриває рапорт підлеглого та натискає "Погодити".	Зміна статусу на "Погоджено". Документ блокується для видалення автором і стає доступним керівнику управління.	Статус успішно змінено. Рапорт з'явився в реєстрі начальника управління.	Успішно
4	Начальник управління відкриває погоджений рапорт і натискає "Затвердити".	Присвоєння фінального статусу "Затверджено". Генерація остаточного PDF з усіма візами та фіксація у статистиці.	Статус змінено на фінальний. Реквізити затвердження коректно відображаються у PDF.	Успішно

Джерело: розроблено автором.

Третій етап присвячено перевірці правильності роботи алгоритмів ієрархічної фільтрації даних та патерну м'якого видалення (Soft Delete) табл. 4.3.

Таблиця 4.3

Сценарії тестування ієрархії доручень та аудиту

№	Дія користувача (Test Steps)	Очікуваний результат (Expected)	Фактичний результат (Actual)	Статус
1	Керівник відділу відкриває модальне вікно створення доручення.	У випадаючому списку виконавців відображаються лише його підлеглі.	Список відфільтровано коректно через LINQ-запит.	**Успішно**

2	Працівник натискає кнопку "Виконано" на вхідному дорученні.	Доручення переміщується в архів, прапорець IsCompleted = true.	Доручення зникло з активних, статус "Виконано".	**Успішно**
3	Працівник натискає "Видалити чернетку рапорту".	Документ зникає з інтерфейсу, але в базі прапорець IsDeleted стає true.	Документ приховано від користувача. Фізичне видалення не відбулося.	**Успішно**
4	Адміністратор відкриває модуль "Журнал аудиту".	Відображення видаленого рапорту (з кроку 3) із зазначенням ПІБ виконавця.	Видалений рапорт успішно зафіксовано в системному лозі.	**Успішно**

Джерело: розроблено автором.

Отже, результати функціонального тестування засвідчили, що всі ключові модулі веб- сервісу працюють стабільно, алгоритми фільтрації та безпеки відпрацьовують без збоїв, а генерація документів відповідає державним стандартам діловодства. Критичних дефектів, які б перешкоджали впровадженню системи, не виявлено.

4.3. Розробка інструкції користувача веб-сервісу

Успішне впровадження будь-якого програмного забезпечення у повсякденну діяльність підрозділів Національної поліції вимагає наявності чіткої, структурованої та зрозумілої документації. Для забезпечення швидкої адаптації персоналу до нової системи електронного документообігу розроблено базову інструкцію користувача, яка описує типові сценарії взаємодії з веб-сервісом.

Крок 1. Авторизація та початок роботи

Для початку роботи з системою працівник поліції повинен відкрити веб-браузер та перейти за внутрішньою IP-адресою сервера. На стартовій сторінці необхідно ввести свої облікові дані (логін та пароль). *Увага:* Система обладнана захистом від підбору паролів. У разі триразового введення неправильних даних обліковий запис буде автоматично заблоковано на 15

хвилин. Для дострокового розблокування або у разі втрати пароля необхідно скористатися функцією «Відновлення доступу», вказавши свій ПІБ та унікальний номер жетона.

Крок 2. Навігація та робота з реєстром звітів

Після успішної авторизації користувач потрапляє на головну робочу панель. Навігація між модулями системи здійснюється за допомогою бокового меню, яке автоматично адаптується під рівень доступу працівника (звичайні працівники не бачать розділів аналітики та адміністрування).

Основним робочим простором є модуль «Реєстр звітів» (рис. 4.1). Цей екран містить таблицю всіх створених користувачем документів. Для зручності навігації таблиця оснащена вкладками-фільтрами: «Всі звіти», «На підпис» та «Архів».

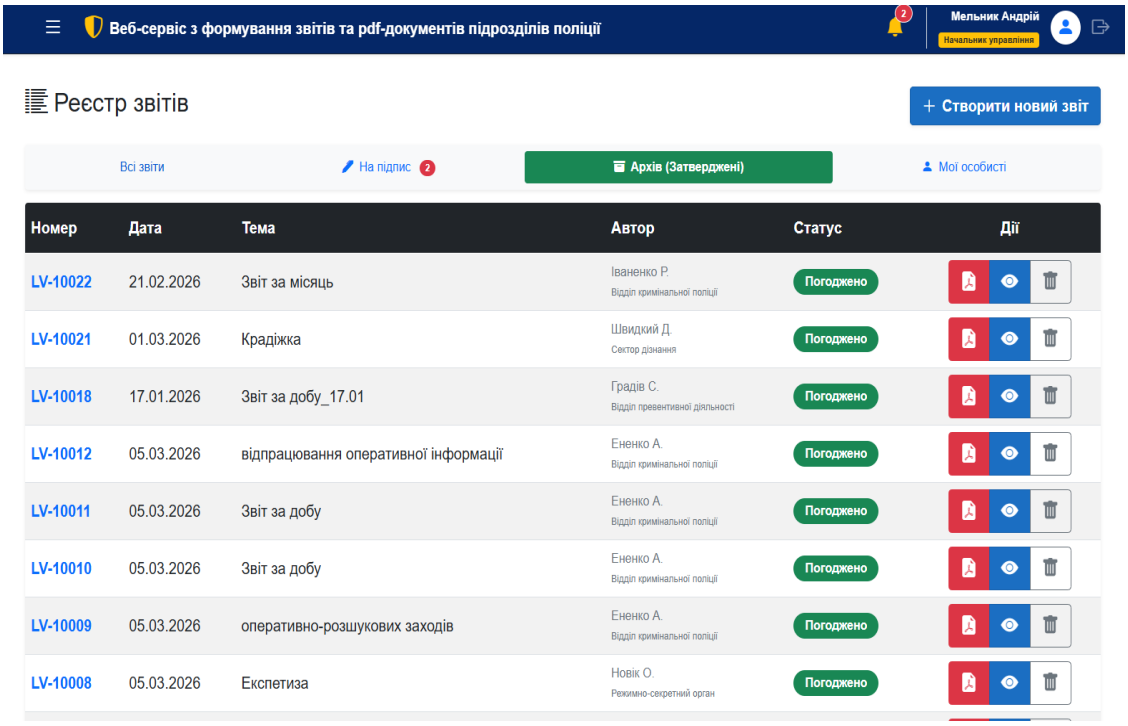


Рис. 4.1. Інтерфейс робочого простору «Реєстр звітів»

Кожен рядок таблиці містить базову інформацію про документ (тема, дата, тип) та візуальний кольоровий індикатор (бейджик) його поточного статусу. Сірий колір означає «Чернетку», жовтий - «Подано на розгляд», зелений -«Затверджено».

Крок 3. Створення та подача рапорту керівництву

Алгоритм створення нового документа складається з наступних дій:

1) У вікні «звітів» натиснути кнопку «Створити новий звіт».

2) У формі, що з'явиться, обрати тип документа (наприклад, «Денний звіт (оперативний)» або «Інцидент») та формат подання.

3) Ввести текстовий опис події (фабулу) у відповідне поле. Система автоматично підтягне посаду, звання та ПІБ автора.

4) Ознайомитися з макетом майбутнього документа за допомогою панелі попереднього перегляду (Live Preview), що розташована праворуч.

5) Натиснути кнопку «Зберегти». Документ з'явиться в реєстрі зі статусом «Чернетка».

На етапі «Чернетки» автор може необмежену кількість разів редагувати текст або видалити документ (у такому разі він буде прихований з інтерфейсу, але збережеться в журналі аудиту). Коли документ готовий, працівник повинен натиснути кнопку «Подати на погодження». Після цього рапорт блокується для редагування та миттєво з'являється в системі його безпосереднього керівника.

Крок 4. Робота зі службовими дорученнями

Для перевірки наявності нових завдань працівник повинен перейти до розділу «Службові доручення». Вхідні завдання, які потребують негайного виконання, позначаються червоним індикатором «ТЕРМІНОВО». Отримавши доручення, поліцейський має ознайомитися з його змістом та дедлайном. Для зручності, безпосередньо з картки доручення можна перейти до форми створення рапорту. Після фактичного виконання вимог керівництва, працівник зобов'язаний натиснути кнопку «Виконано» на картці завдання, після чого воно буде переміщено до архіву.

Запропонована інструкція покриває понад 90% повсякденних операцій рядового складу. Завдяки інтуїтивно зрозумілому інтерфейсу, адаптація персоналу до роботи з веб-сервісом не потребує проведення довготривалих навчальних семінарів.

4.4. Оцінка ефективності впровадження системи

Головним показником успішності розробленого програмного забезпечення є його здатність оптимізувати робочі процеси та економити людські ресурси. Для доведення практичної користі веб-сервісу проведемо розрахунок зекономленого робочого часу на прикладі процесу створення одного стандартного поліцейського рапорту.

Традиційний підхід (використання Microsoft Word): При ручному створенні документа працівник витрачає час на пошук потрібного файлу-шаблону на комп'ютері, ручне введення поточної дати, зміну ПІБ (своїх та керівника), **ручне вирівнювання реквізитів документа та тексту під актуальні стандарти оформлення** та подальший друк або відправку. У середньому цей рутинний процес займає близько 15 хвилин.

Автоматизований підхід (розроблений веб-сервіс): При використанні веб-сервісу працівник просто обирає тип рапорту з випадаючого списку та вводить лише зміст події (фабулу). Всі системні реквізити, звання, дати та форматування генеруються бібліотекою QuestPDF миттєво і автоматично. Цей процес займає близько **3 хвилин**.

Організаційний та економічний ефект: Чиста економія часу на формуванні лише одного документа становить **12 хвилин**. Для розрахунку ефективності візьмемо середньостатистичний відокремлений підрозділ поліції, де в зміні працює 20 офіцерів, кожен з яких за добу складає мінімум 2 рапорти:

1) **20 працівників × 2 рапорти × 12 хвилин = 480 хвилин (рівно 8 робочих годин).**

Окрім прямої економії часу, система успішно вирішує критичну проблему непередбачуваних кадрових змін (відпустки, лікарняні, переведення або звільнення керівництва). У традиційному паперовому документообігу подача рапорту на ім'я керівника, який раптово став недоступним, вимагає від підлеглого повного передруковування документа з новими реквізитами.

Натомість у розробленому веб-сервісі адміністратор просто делегує права доступу особі, яка тимчасово виконує обов'язки (ТВО). Усі раніше подані електронні рапорти залишаються дійсними і можуть бути погоджені новим керівником без жодних переписувань, що повністю виключає ризик зупинки робочих процесів.

Отже, математичний розрахунок та аналіз бізнес-процесів доводять, що впровадження даного веб-сервісу лише в одному відділі щодня вивільняє **один повноцінний 8-годинний робочий день** та зводить до нуля бюрократичні затримки. Цей колосальний часовий ресурс працівники Національної поліції зможуть витратити на виконання своїх прямих правоохоронних обов'язків (патрулювання, слідчі дії), а не на механічну паперову рутину.

Крім часового ресурсу, варто відзначити й прямий матеріальний та екологічний ефект від впровадження системи. У традиційному діловодстві через помилки ручного форматування або описки працівники часто змушені переробляти та передруковувати документи по кілька разів. Автоматична генерація PDF-файлів зводить кількість зіпсованих чернеток до нуля, що суттєво скорочує фінансові витрати підрозділу на канцелярські матеріали (папір формату А4, тонер для принтерів, зношування друкарської техніки). Крім того, надійне зберігання електронних копій у базі даних SQLite зменшує потребу у фізичних приміщеннях для архівування паперових справ, що повністю відповідає державній стратегії цифровізації та переходу до режиму (безпаперового офісу).

Висновки до четвертого розділу

У четвертому розділі кваліфікаційної роботи проведено комплексне тестування розробленого веб-сервісу та оцінено ефективність його впровадження.

Для перевірки працездатності системи було обрано метод «чорного ящика» та розгорнуто локальне тестове середовище. Сценарії функціонального тестування довели безвідмовну роботу ключових алгоритмів: механізмів авторизації із захистом від підбору паролів, ієрархічної фільтрації даних (залежно від посади користувача) та швидкої програмної генерації PDF-звітів. Усі заявлені модулі, включно з «Журналом аудиту» та системою м'якого видалення (Soft Delete), відпрацювали коректно.

Для забезпечення швидкого впровадження програмного продукту на робочих місцях розроблено базову інструкцію користувача. Вона покриває основні сценарії роботи поліцейського: від авторизації до створення рапорту та опрацювання службових доручень.

Проведені розрахунки ефективності переконливо довели високу практичну цінність розробки. За рахунок автоматизації процесів форматування система здатна економити до 8 робочих годин щодня в межах одного підрозділу. Крім того, доведено високу організаційну гнучкість системи при кадрових змінах, що повністю позбавляє поліцейських необхідності передруковувати документи у разі раптової відсутності керівника. Це суттєво оптимізує документообіг та підвищує загальну ефективність роботи територіальних органів Національної поліції України.

ВИСНОВКИ

У кваліфікаційній роботі вирішено актуальне науково-практичне завдання, яке полягає у проєктуванні та розробці спеціалізованого веб-сервісу для автоматизації документообігу, генерації звітності та контролю службових доручень у відокремленому підрозділі Національної поліції України. Підсумовуючи результати проведеного дослідження, можна зробити наступні висновки:

1. **Досліджено** процеси звітності та документообігу в підрозділах Національної поліції України. **Встановлено**, що незважаючи на наявність глобальних баз даних, початкова стадія створення рапортів та зведення локальної аналітики залишається рутинною і виконується здебільшого вручну за допомогою стандартних текстових редакторів. Це призводить до витрат робочого часу та порушення стандартів оформлення. Мета кваліфікаційної роботи досягнута, а всі поставлені у вступі завдання виконані в повному обсязі.

2. **Проаналізовано** співвідношення виконаної розробки з вітчизняними та світовими аналогами. **Виявлено**, що існуючі комерційні генератори звітів (Microsoft SSRS, FastReport) та відомчі системи електронного документообігу (СЕД, наприклад «МІА: Документообіг») орієнтовані на маршрутизацію вже готових документів і не надають гнучкого, легкого у розгортанні інструментарію для зручного покрокового створення специфічних галузевих звітів «з нуля». Натомість розроблений веб-сервіс доповнює ці системи, вирішуючи проблему «останньої милі» та забезпечуючи миттєву генерацію стандартизованих звітів без необхідності закупівлі дорогих ліцензій.

3. **Обґрунтовано** вибір сучасного технологічного стеку та **розроблено** архітектуру системи за принципами багаторівневої «Чистої архітектури» (Clean Architecture). Застосування платформи .NET 8, серверної моделі обчислень Blazor Server та СУБД SQLite з технологією Entity Framework Core дозволило створити ізольовану, масштабовану і захищену систему, де

вихідний код та чутливі оперативні дані ніколи не передаються на клієнтські пристрої.

4. **Отримано** високі якісні показники роботи системи за рахунок впровадження багаторівневої моделі управління доступом (RBAC). **Розроблено** модулі генерації уніфікованих PDF-документів (з використанням бібліотеки QuestPDF) з урахуванням базових вимог до оформлення документів, контролю службових доручень та інтерактивної аналітики (дашборди для керівництва з фільтрацією даних через LINQ-запити). Реалізовано механізми безпеки, серед яких захист від атак Brute Force, патерн м'якого видалення (Soft Delete) та ведення незмінного «Журналу аудиту».

5. **Доведено** значний очікуваний економічний та організаційний ефект від можливості використання матеріалів кваліфікаційної роботи у практичній діяльності органів Національної поліції. Математичні розрахунки підтвердили, що автоматизація процесу скорочує час на підготовку одного документа на 12 хвилин. Для середньостатистичного підрозділу з 20 офіцерів це вивільняє 8 робочих годин щодня. Крім того, **встановлено**, що система повністю нівелює бюрократичні затримки під час кадрових змін (відпустки, переведення керівництва), оскільки дозволяє гнучко делегувати права доступу без необхідності передруковувати раніше створені паперові рапорти.

6. Виконана робота безпосередньо пов’язана з науково-дослідними розробками **кафедри інформаційний технологій** Львівського державного університету внутрішніх справ, зокрема в напрямі пошуку та впровадження новітніх інформаційних технологій у повсякденну діяльність підрозділів МВС.

7. **Вважається за доцільне** та **рекомендовано** використання результатів кваліфікаційної роботи в навчальному процесі університету. Розроблений програмний продукт, архітектурні діаграми та наведені фрагменти коду (на мові C#) можуть використовуватися як наочний навчально-методичний матеріал під час викладання дисциплін «Інформаційні технології»,

«Проєктування інформаційних систем» та «Основи інформаційної безпеки» для курсантів і студентів.

8. Щодо перспективних напрямів подальших наукових розвідок та розвитку системи **запропоновано**:

1) розширення функціоналу веб-сервісу шляхом інтеграції кваліфікованого електронного підпису (КЕП) безпосередньо на етапі затвердження документа в системі;
2) реалізацію програмних інтерфейсів (API) для автоматичного двостороннього обміну даними з глобальною системою ІПНП (Інформаційний портал Національної поліції);
3) масштабування системи з використанням промислових серверів баз даних (PostgreSQL) для розгортання на рівні Головних управлінь у регіонах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Про Національну поліцію : Закон України від 02.07.2015 № 580-VIII. URL: https://zakon.rada.gov.ua/laws/show/580-19 (дата звернення: 13.11.2025).

2. Про захист інформації в інформаційно-комунікаційних системах : Закон України від 05.07.1994 № 80/94-ВР. URL: https://zakon.rada.gov.ua/laws/show/80/94-вр (дата звернення: 13.11.2025).

3. ДСТУ 4163:2020. Державна уніфікована система документації. Уніфікована система організаційно-розпорядчої документації. Вимоги до оформлення документів. Київ : ДП «УкрНДНЦ», 2020. 34 с.

4. Вишня В. Б., Гавриш О. С., Рижков Е. В. Основи інформаційної безпеки : навчальний посібник. Дніпро : ДДУВС, 2020. 128 с.

5. Інформаційні технології : навчальний посібник / О. І. Зачек, В. В. Сеник, Т. В. Магеровська та ін. Львів : ЛьвДУВС, 2022. 432 с.

6. Іванов І. Є. Проблематика використання автоматизованих інформаційних систем у діяльності МВС України. Юридична наука. 2020. № 5(107). С. 113 - 120.

7. Смагіна О. О., Переяславська С. О. Якість програмного забезпечення та тестування : навч. посіб. Старобільськ : ДЗ «ЛНУ імені Тараса Шевченка», 2021. 286 с.

8. Магеровська Т. В., Сеник В. В., Магеровський Д. В. Проєктування користувацьких інтерфейсів та комп'ютерної графіки : навч. посібник. Львів : ЛьвДУВС, 2025. 406 с.

9. Троелсен Е., Джепікс Ф. Мова програмування C# 9 та платформа .NET 5 / пер. з англ. Київ : Діалектика, 2022. 1328 с.

10. ISO/IEC 25010:2023. Systems and software engineering -Systems and software Quality Requirements and Evaluation (SQuaRE) -Product quality model. Geneva : ISO, 2023.

11. Martin R. C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Boston : Prentice Hall, 2017. 432 p.

12. Price M. J. C# 12 and .NET 8 -Modern Cross-Platform Development Fundamentals. 8th ed. Birmingham : Packt Publishing, 2023. 822 p.

13. Впроваджуємо електронний документообіг в органах МВС України! Державне підприємство «ІНФОТЕХ». 2024. URL: https://infotech.gov.ua/news/vprovadzhuyemo-elektronnij-dokumentoobig-v-organah-mvs-ukraini! (дата звернення: 09.12.2025).

14. Перелік протестованих СЕД. Система електронної взаємодії органів виконавчої влади (ДП «ДІЯ»). 2024. URL: https://se.diia.gov.ua/sedlist (дата звернення: 11.12.2025).

15. ASP.NET Core Blazor. Microsoft Learn. 2024. URL: https://learn.microsoft.com/en-us/aspnet/core/blazor/ (дата звернення: 03.01.2026).

16. Crystal Reports : Business Intelligence and Reporting. SAP. URL: https://www.sap.com/products/technology-platform/crystal-reports.html (дата звернення: 05.01.2026).

17. FastReport .NET : Reporting tool for .NET 8. Fast Reports Inc. URL: https://www.fast-report.com/en/product/fast-report-net/ (дата звернення: 15.01.2026).

18. jsReport : Open source reporting server. jsReport. URL: https://jsreport.net/ (дата звернення: 15.01.2026).

19. Microsoft SQL Server Reporting Services. Microsoft Learn. URL: https://learn.microsoft.com/en-us/sql/reporting-services/ (дата звернення: 15.01.2026).

20. QuestPDF Documentation: Modern open-source .NET library for PDF generation. QuestPDF. 2024. URL: https://www.questpdf.com (дата звернення: 15.01.2026).

21. SQLite Official Documentation. SQLite. 2024. URL: https://www.sqlite.org/docs.html (дата звернення: 15.01.2026).

22. What is .NET? Introduction and overview. Microsoft Learn. 2024. URL: https://learn.microsoft.com/en-us/dotnet/core/introduction (дата звернення: 17.01.2026).

ДОДАТКИ

Додаток А

Фрагмент вихідного коду генерації PDF-документа рапорту

```
using System.Linq;
using QuestPDF.Fluent;
using QuestPDF.Helpers;
using QuestPDF.Infrastructure;
using PoliceReportSystem.Models;
using PoliceReportSystem.Core.Enums;

namespace PoliceReportSystem.PDF
{
   public class ReportDocument : IDocument
   {
      public Report ReportModel { get; }

      public ReportDocument(Report report)
      {
         ReportModel = report;
      }

      public byte[] GetPdfBytes() => this.GeneratePdf();

      public DocumentMetadata GetMetadata() => DocumentMetadata.Default;

      public void Compose(IDocumentContainer container)
      {
         QuestPDF.Settings.License = LicenseType.Community;

         container.Page(page =>
         {
            page.Margin(40);
```

```
            page.DefaultTextStyle(x=>
x.FontSize(12).FontFamily(Fonts.TimesNewRoman));

            page.Header().Element(ComposeHeader);
            page.Content().Element(ComposeContent);
            page.Footer().AlignCenter().Text(x =>
            {
                x.Span("Стор. ");
                x.CurrentPageNumber();
            });
        });
    }

    void ComposeHeader(IContainer container)
    {
        container.Row(row =>
        {
            // Формування лівої частини: номер рапорту
            row.RelativeItem().Column(column =>
            {
                column.Item().Text($"№
{ReportModel.ReportNumber}").FontSize(12).Bold();
            });

            // Формування правої частини: гриф затвердження керівником
            row.RelativeItem().AlignRight().Column(column =>
            {
                if (ReportModel.ChiefApprover != null)
                {
                    var chief = ReportModel.ChiefApprover;
                    column.Item().Text("ЗАТВЕРДЖУЮ").Bold();
```

```
                column.Item().Text(chief.Position ?? "Начальник управління").FontSize(11);
                column.Item().Text("ГУНП у Львівській області").FontSize(11);
                column.Item().Text($"{GetUaRank(chief.Rank)} поліції").FontSize(11);

                string lastName = (chief.LastName ?? "").ToUpper();
                string initial = (!string.IsNullOrEmpty(chief.FirstName)) ? $"{chief.FirstName}" : "";

                column.Item().PaddingTop(10).Text(text =>
                {
                    text.Span("_____ ").FontColor(Colors.Black);
                    text.Span($"{initial} {lastName}").Bold();
                });
            }
            else
            {

column.Item().PaddingTop(5).Text("ПРОЄКТ").FontColor(Colors.Grey.Lighten2).FontSize(20).Bold();
            }
        });
    });
}

void ComposeContent(IContainer container)
{
    container.PaddingVertical(20).Column(column =>
    {
        // Заголовок та тип документа
```

```
column.Item().PaddingTop(20).AlignCenter().Text("РАПОРТ").Bold().FontSize(16);
            column.Item().AlignCenter().Text($"про {GetUaType(ReportModel.Type)}").FontSize(14).Italic();

            // Відображення звітного періоду за необхідності
            bool showPeriod = ReportModel.Type == ReportType.Monthly ||
                              ReportModel.Type == ReportType.Quarterly ||
                              ReportModel.Type == ReportType.Annual ||
                              ReportModel.Type == ReportType.Weekly ||
                              ReportModel.Type == ReportType.Statistical;

            column.Item().PaddingBottom(5).Text(text =>
            {
                if (showPeriod && ReportModel.PeriodStart != default && ReportModel.PeriodEnd != default)
                {
                    string periodStr = $" за період з {ReportModel.PeriodStart:dd.MM.yyyy} по {ReportModel.PeriodEnd:dd.MM.yyyy}";
                    text.Span(periodStr).SemiBold();
                }
            });

            // Основний текст (фабула) з імітацією абзацного відступу
            if (!string.IsNullOrEmpty(ReportModel.Description))
            {
                column.Item().PaddingBottom(10).Text(text =>
                {
                    text.Justify();
                    text.Span("      " + ReportModel.Description);
```

```
            });
        }

        // Генерація таблиці статистичних показників
        if (ReportModel.Statistics != null && ReportModel.Statistics.Any())
        {
            column.Item().PaddingTop(10).Table(table =>
            {
                table.ColumnsDefinition(columns => { columns.RelativeColumn(3); columns.RelativeColumn(1); });
                table.Header(header => {

header.Cell().Border(1).Padding(2).AlignCenter().Text("Показник").Bold();

header.Cell().Border(1).Padding(2).AlignCenter().Text("Кількість").Bold();
                });
                foreach (var stat in ReportModel.Statistics)
                {
                    table.Cell().Border(1).Padding(2).Text(stat.Label ?? "-");

table.Cell().Border(1).Padding(2).AlignCenter().Text(stat.Value.ToString());
                }
            });
        }

        column.Item().PaddingTop(40);

        // Блок візування документа
        bool isSelfApproved = ReportModel.ApprovedBy != null &&
                              ReportModel.CreatedBy != null &&
                              ReportModel.ApprovedBy.Id == ReportModel.CreatedBy.Id;
```

```
            // Відображення резолюції погодження безпосереднім керівником
            if (ReportModel.ApprovedBy != null && !isSelfApproved)
            {
                column.Item().Text("ПОГОДЖЕНО").Bold();
                ComposeSignatureRow(column, ReportModel.ApprovedBy, ReportModel.ApprovedAt, isAuthor: false);
                column.Item().PaddingTop(20);
            }

            // Блок підпису автора рапорту
            var author = ReportModel.CreatedBy ?? new Officer { LastName = "Невідомий", FirstName = "", Position = "Працівник" };
            ComposeSignatureRow(column, author, ReportModel.ReportDate, isAuthor: true);
        });
    }

    // Метод динамічного формування структури підпису згідно з вимогами діловодства
    void ComposeSignatureRow(ColumnDescriptor column, Officer officer, DateTime? date, bool isAuthor)
    {
        column.Item().PaddingBottom(20).Column(c =>
        {
            c.Item().Text(officer.Position ?? "Посада не вказана");

            string dept = ReportModel.Department?.Name ?? "Львівське районне управління поліції №2";
            c.Item().Text(dept);

            c.Item().Row(r =>
```

```
            {
                r.RelativeItem().Text($"{GetUaRank(officer.Rank)} поліції");

                string fullName = $"{officer.FirstName} {officer.LastName.ToUpper()}";
                r.RelativeItem().AlignRight().Text(fullName).Bold();
            });

            string dateStr = date.HasValue ? date.Value.ToString("dd.MM.yyyy") : "__.__.20__";
            c.Item().Text(dateStr);
        });
    }
  }
}
```

Додаток Б

Фрагмент вихідного коду Blazor-компонента обробки та відображення оперативної статистики

```
@page "/statistics"

@using PoliceReportSystem.Models

@using PoliceReportSystem.Core.Enums

@using PoliceReportsService.Services

@inject ReportService ReportService

@inject UserService UserService


<div class="container mt-4">

    <div class="d-flex justify-content-between align-items-center mb-4">

        <h2><i class="bi bi-graph-up-arrow me-2"></i>Оперативна статистика</h2>

        @if (!IsBigBoss && UserService.CurrentUser?.CanApprove == true)

        {

            <button class="btn btn-success" @onclick="GenerateUnitReport">Сформувати зведений рапорт</button>

        }

    </div>


    <div class="card shadow border-top border-4 border-primary">

        <div class="card-header bg-white py-3">

            <h5 class="mb-0 fw-bold text-primary">Деталізація по статтях</h5>

        </div>

        <table class="table table-hover align-middle mb-0">
```

```
<thead class="table-light">
    <tr>
        <th>Кодекс</th>
        <th>Стаття</th>
        <th>Фабула / Опис</th>
        <th>Кількість</th>
    </tr>
</thead>
<tbody>
    @if (AggregatedStats.Any())
    {
        @foreach (var item in AggregatedStats)
        {
            <tr>
                <td><span class="badge bg-primary">@item.Category</span></td>
                <td class="fw-bold fs-5">@item.Article</td>
                <td>@item.Label</td>
                <td class="text-center fw-bold">@item.Count</td>
            </tr>
        }
    }
</tbody>
</table>
```

```
    </div>
</div>

@code {
    private DateTime StartDate = DateTime.Now.AddDays(-7);
    private DateTime EndDate = DateTime.Now;
    private string SelectedUnit = "";
    private bool IsBigBoss = false;

    class AggregatedRow
    {
        public StatCategory Category { get; set; }
        public string Article { get; set; }
        public string Label { get; set; }
        public int Count { get; set; }
    }

    protected override void OnInitialized()
    {
        var user = UserService.CurrentUser;
        if (user != null) IsBigBoss = user.IsChief;
    }

    // Основний алгоритм агрегації та фільтрації даних з бази
```

```
private IEnumerable<AggregatedRow> AggregatedStats
{
    get
    {
        // 1. Отримання та базова фільтрація за статусом і датою
        var reports = ReportService.GetAllReports()
            .Where(r => r.Status != ReportStatus.Draft && r.Status != ReportStatus.Rejected)
            .Where(r => r.ReportDate.Date >= StartDate.Date && r.ReportDate.Date <= EndDate.Date)
            .Where(r => r.IsSummaryReport == false) // Ігнорування зведених звітів
            .ToList();

        // 2. Фільтрація доступу на основі ролей (RBAC)
        if (IsBigBoss && !string.IsNullOrEmpty(SelectedUnit))
        {
            reports = reports.Where(r =>
                UserService.Users.FirstOrDefault(u => u.Id == r.CreatedByOfficerId)?.Unit == SelectedUnit).ToList();
        }
        else if (!IsBigBoss)
        {
            var myUnit = UserService.CurrentUser?.Unit;
            reports = reports.Where(r =>
```

```
            UserService.Users.FirstOrDefault(u => u.Id == r.CreatedByOfficerId)?.Unit == myUnit).ToList();
        }

        // 3. Групування (Map-Reduce) та підрахунок статистики
        var flatStats = reports.SelectMany(r => r.Statistics.Select(s => new { Stat = s }));

        return flatStats.GroupBy(x => new { x.Stat.Category, x.Stat.Article })
            .Select(g => new AggregatedRow
            {
                Category = g.Key.Category,
                Article = g.Key.Article,
                Label = g.First().Stat.Label,
                Count = g.Sum(x => x.Stat.Value)
            })
            .OrderBy(x => x.Category)
            .ThenByDescending(x => x.Count);
    }
}
```