НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ
Завідувач кафедри
інформаційних систем і технологій
_____ Швиденко М.З.,доц.,к.е.н.
«___»_____2025р

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему

«Використання сучасних інформаційних рішень для бюрократичних процесів»
Спеціальність 122 – «Комп'ютерні науки»

Гарант освітньої програми
Д.е.н., професор _____ Руденський Р.А.

Керівник бакалаврської кваліфікаційної роботи

д. філос. н. (спец. 122 "Комп'ютерні науки"), ст. викл. _____ Золотуха Р. А.
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав _____ Черненко Павло Віталійович
(підпис) (ПІБ студента)

КИЇВ – 2025

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інформаційних систем і технологій
_____ Швиденко М.З.,доц.,к.е.н.
«___»_____2025р

ЗАВДАННЯ

на виконання бакалаврської кваліфікаційної роботи студенту
Черненко Павло Віталійович
(прізвище, ім'я, по батькові)

Спеціальність 122 – «Комп'ютерні науки»

Тема бакалаврської кваліфікаційної роботи: «Використання сучасних інформаційних рішень для бюрократичних процесів»
затверджена наказом ректора НУБіП України від "16"12 2024 р.
№2246 "С"
Термін подання завершеної роботи на кафедру 2025 (рік) . 05 (місяць) . 25 (число)

Вихідні дані до роботи: дані громадян, типи звернень, текстові повідомлення у запитах, дати подачі, статуси обробки, службова інформація про відповідальних користувачів, відділи та посади.

4. Перелік питань, що розглядаються:
1. Аналіз проблемної області
2. Моделювання предметної області
3. Інформаційне забезпечення системи
4. Програмне забезпечення системи
5. Тестування програмного продукту

Дата видачі завдання 16 грудня 2024 р.
Керівник бакалаврської кваліфікаційної роботи
д. філос. н. (спец. 122 "Комп'ютерні науки"), ст. викл. _____ Золотуха Р. А.
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Завдання прийняв до виконання _____ Черненко П. В.
(підпис) (ПІБ студента)

ЗМІСТ

ВСТУП

У сучасному суспільстві, яке стрімко розвивається в напрямі цифровізації, однією з найбільш уразливих ланок залишаються бюрократичні процеси, що формують основу взаємодії між громадянами та державними установами. Незважаючи на активне впровадження електронного урядування у провідних країнах світу, в Україні все ще зберігається значна частка ручного документообігу, надмірної паперової звітності, дублювання функцій між органами влади та тривалих термінів обробки звернень. Це не лише уповільнює управлінські процеси, а й знижує рівень довіри громадян до ефективності державних інституцій.

Значну увагу в наукових та прикладних дослідженнях отримали інформаційні системи підтримки адміністративних процедур. Наприклад, системи типу CRM (Customer Relationship Management), BPM (Business Process Management) або ECM (Enterprise Content Management), що широко використовуються у приватному секторі, демонструють високу ефективність в автоматизації документообігу та внутрішніх процесів [1]. Водночас, їх адаптація до публічної сфери стикається з низкою бар’єрів: нормативною невідповідністю, складністю інтеграції з реєстрами, відсутністю уніфікованих форматів даних [2]. Існуючі державні рішення, зокрема «Єдине вікно для подання звернень» або сервіси «Дія», хоч і є прогресивними, однак не охоплюють повністю специфіку міжвідомчої взаємодії на локальному рівні, особливо в частині аналізу ефективності та прозорості бюрократичних процедур.

У таких умовах постає необхідність розробки адаптивних, спеціалізованих інформаційних рішень, здатних забезпечити централізоване зберігання, обробку та аналітичну оцінку звернень громадян. Особливої актуальності ця задача набуває в умовах воєнного часу та післявоєнного відновлення України, коли державне управління має бути максимально швидким, чітким і прозорим. Такі системи мають не лише забезпечити оперативність обробки звернень, а й сформувати науково обґрунтовану основу для оптимізації управлінських рішень на базі об'єктивних статистичних даних.

Об’єктом дослідження є процес цифрової трансформації бюрократичних процедур у системах публічного управління та документообігу.

Предметом дослідження є інформаційні моделі, програмні рішення та засоби автоматизації, що забезпечують облік, обробку та аналіз звернень громадян у межах адміністративних процесів.

Метою даної роботи є розробка інформаційного рішення для автоматизації та оптимізації бюрократичних процесів в межах системи обробки звернень громадян, що забезпечує централізоване зберігання даних, зручну взаємодію з користувачем та підтримку аналітичних функцій для прийняття управлінських рішень.

Для досягнення поставленої мети необхідно вирішити такі основні задачі:

- провести системний аналіз предметної області, зокрема виявити типові структури бюрократичних процесів, проаналізувати наявні інформаційні рішення та сформувати вимоги до функціональності майбутньої системи;

- побудувати формалізовані моделі предметної області, включаючи об’єктно-орієнтоване та функціональне моделювання, що дозволить чітко визначити логіку взаємодії між елементами системи;

- спроєктувати архітектуру програмної системи, включаючи структуру бази даних, логічні та фізичні компоненти, а також обґрунтувати вибір технологій і інструментів реалізації з урахуванням особливостей задачі;

- реалізувати дослідний зразок програмного забезпечення, провести його тестування, перевірити відповідність функціональним і нефункціональним вимогам.

Структура роботи. Дипломна робота побудована з урахуванням логіки проєктування інформаційної системи та складається із наступних розділів:

У першому розділі виконано аналіз проблемної сфери: сформульовано основні характеристики предметної області, розглянуто сучасні інформаційні рішення, що застосовуються для автоматизації бюрократичних процедур, а

також сформульовано технічну постановку задачі та основні вимоги до функціонування майбутньої системи, зокрема з боку користувачів та інтерфейсу.

Другий розділ присвячено формалізації предметної області шляхом побудови моделей. У ньому подано вихідні положення для моделювання, представлено об'єктно-орієнтований та функціональний підхід до опису логіки роботи системи. Також розглянуто абстрактні сутності, що визначають структуру даних, та побудовано діаграму класів, яка відображає взаємозв’язки між основними елементами системи.

У третьому розділі розкрито етапи проєктування програмного забезпечення. Тут створено логічну структуру бази даних, обґрунтовано вибір СУБД для реалізації, описано програмну архітектуру, функціональні модулі та методи їх взаємодії. Також надано аргументи щодо вибору середовища розробки та інструментальних засобів, які використовувались для побудови системи.

Четвертий розділ охоплює питання впровадження та тестування створеного програмного продукту. Зокрема, визначено технічні умови для інсталяції системи, представлено результати перевірки її працездатності та виконано аналіз функціональної відповідності вимогам. Окремо запропоновано практичні рекомендації щодо експлуатації системи в реальних умовах.

1 АНАЛІЗ ПРОБЛЕМНОЇ ОБЛАСТІ

1.1 Аналіз предметної області, її структурних та функціональних особливостей

Бюрократичні процеси – це сукупність формалізованих процедур, правил і документообігу, необхідних для функціонування державних установ, комерційних організацій та інших інституцій. Їхня структура зазвичай відзначається послідовністю дій, фіксацією відповідальності на кожному етапі, необхідністю дотримання певного регламенту та суворим контролем за дотриманням термінів виконання [3]. Для кращого розуміння логіки функціонування таких процесів у межах обробки звернень громадян на рисунку 1.1 зображено узагальнену схему, що ілюструє основні етапи та інформаційні компоненти предметної області.

Рис. 1.1 Особливості предметної області

На схемі структуровано ключові напрями функціонування бюрократичного середовища, умовно поділені на три блоки, кожен з яких має своє кольорове позначення. Фіолетовим кольором виокремлено сферу взаємодії з громадянами та бізнесом, яка охоплює надання держпослуг, електронні петиції та механізми зворотного зв’язку. Жовті елементи позначають типові

адміністративні дії, зокрема видачу паспортів, оформлення довідок, реєстрацію документів та отримання ліцензій. Зелений блок відображає внутрішні адміністративні процедури, пов’язані з документальним обігом, звітністю та погодженням управлінських рішень.

Кольорове розмежування у схемі використано не випадково: воно дозволяє візуально виокремити різні функціональні зони предметної області та підкреслити структурну взаємодію між зовнішніми та внутрішніми аспектами бюрократичного процесу. Це спрощує сприйняття складної системи та підсилює її логічну організацію, що особливо важливо при моделюванні відповідного програмного забезпечення.

На основі цих даних звернення підлягає реєстрації в інформаційній системі, де фіксуються всі супровідні реквізити. Далі воно направляється до відповідного структурного підрозділу згідно з темою або функціональною компетенцією, після чого закріплюється за конкретним виконавцем. Призначений працівник здійснює обробку звернення відповідно до встановленого регламенту: надає відповідь, ініціює внутрішню перевірку, передає іншим підрозділам або закриває запит зі збереженням результатів. Кожен етап супроводжується оновленням статусу, який реєструється в системному журналі подій. Після завершення опрацювання інформація підлягає аналітичній обробці з формуванням звітності щодо кількісних та якісних показників: динаміки звернень, навантаження на виконавців, ефективності реагування тощо. Уся схема в цілому ілюструє взаємозв’язок між інформаційними сутностями предметної області та їх інтеграцію в єдину цифрову інфраструктуру обліку, аналізу та контролю.

На рис. 1.2 згруповано ключові технології, що сприяють покращенню бюрократичних процесів у контексті цифрової трансформації адміністративних процедур. Візуалізація представлена у вигляді багаторівневої структури, що відображає системний характер трансформації та наголошує на міждисциплінарності підходу. Для полегшення сприйняття інформації та розмежування різноспрямованих технологічних напрямів, діаграму було

реалізовано з використанням кольорового кодування. Кожен колір відповідає окремій функціональній групі технологій: рожевий блок охоплює інструменти на основі штучного інтелекту, машинного навчання та аналізу даних; фіолетовий – пов’язаний із блокчейн-технологіями в рамках нотаріального супроводу та прозорості; жовтий – відображає цифровізацію державних послуг і доступні електронні сервіси; зелений – охоплює системи електронного документообігу; червоний – репрезентує засоби автоматизації.

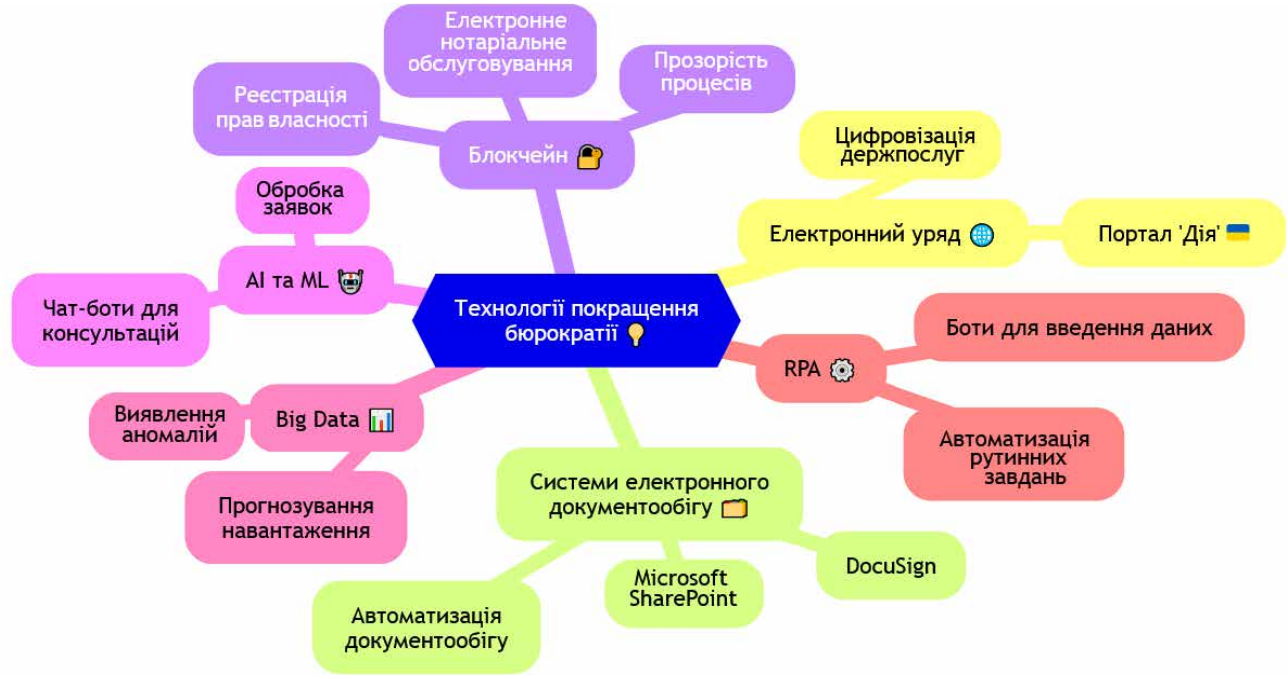


Рис. 1.2 Технології покращення бюрократичних процедур

У центрі уваги знаходиться концепція технологічного удосконалення бюрократії, що розглядається як інтеграційний вузол кількох стратегічних напрямів цифрової трансформації. Ці напрями формують основу нової парадигми взаємодії між громадянами та інституціями, де домінують автоматизовані, прозорі та передбачувані механізми обробки запитів. Один із напрямів охоплює застосування штучного інтелекту та методів машинного навчання для класифікації звернень, виявлення аномальних ситуацій і прогнозування навантаження на підрозділи. Також активно впроваджуються чат-боти, які виконують роль цифрових консультантів для попереднього збирання даних і відповіді на типові запити.

Іншим важливим напрямом є розвиток електронного нотаріального обслуговування на основі блокчейн-технологій, що гарантують незмінність та достовірність записів, особливо актуальних у процесах реєстрації майнових прав, заявок та актів. Значну роль у підвищенні ефективності адміністративної діяльності відіграють системи електронного документообігу. Використання таких платформ, як Microsoft SharePoint або DocuSign, забезпечує впорядкований облік, архівацію, швидке погодження та передачу документів, що суттєво знижує ймовірність людських помилок і прискорює прийняття рішень [4].

Ще одним напрямом, що швидко розвивається, є роботизована автоматизація процесів (RPA), орієнтована на виконання повторюваних рутинних дій без втручання людини – зокрема, введення типових даних, заповнення шаблонів та автоматичне формування відповідей. Завершує загальну картину цифровізації державного управління розвиток електронних сервісів, зокрема таких, як електронний уряд, портал «Дія» та інші ініціативи, що створюють прозорий і доступний канал взаємодії громадян із державою [5]. У цілому структура рисунка відображає багаторівневу, взаємопов’язану екосистему технологій, здатних не лише оптимізувати адміністративні функції, а й суттєво змінити логіку їхньої організації відповідно до принципів відкритого та орієнтованого на результат публічного управління [6].

Рисунок 1.3 узагальнює основні проблеми, які вирішуються завдяки впровадженню сучасних інформаційних технологій у сфері бюрократичного адміністрування. Для чіткішої візуалізації та логічного структурування ключових напрямів застосовано кольорову диференціацію зон: зелений колір позначає наслідки зменшення людського фактору та помилок, жовтий – автоматизацію та прискорення операцій, бузковий – розширення доступу до сервісів і підвищення зручності для громадян, рожевий – прозорість, відкритість і аналітичну підзвітність. Такий підхід дозволяє акцентувати увагу на багатовекторному впливі ІТ-рішень, демонструючи як окремі інструменти сприяють досягненню стратегічних цілей публічного управління.

Рис. 1.3 Проблеми, які вирішують IT-рішення

Використання сучасних цифрових інструментів дозволяє істотно зменшити вплив людського фактору на прийняття рішень, що своєю чергою знижує кількість помилок та зменшує ризики корупційних проявів. Завдяки автоматизації рутинних процесів досягається прискорення обробки звернень та адміністративних процедур, що позитивно впливає як на навантаження працівників, так і на задоволеність користувачів. У впровадженні онлайн-сервісів для громадян особливе значення має забезпечення цілодобового доступу до послуг, що суттєво підвищує їхню доступність і зменшує потребу у фізичній присутності в органах влади.

Значну роль відіграє підвищення прозорості та відкритості процесів, що реалізується шляхом надання публічної аналітики та звітності. Такі інструменти сприяють підзвітності виконавців, формуванню довіри до інституцій та створюють умови для суспільного контролю. Важливою перевагою є також можливість відстеження статусу справи в реальному часі, що забезпечує оперативну інформованість заявника про хід розгляду його звернення. Сукупність цих рішень формує якісно новий рівень взаємодії між державою та громадянином, орієнтований на ефективність, прозорість і цифрову інклюзію.

1.2 Огляд існуючих програмних рішень

У процесі цифрової трансформації органів державної влади та місцевого самоврядування особливу увагу приділяють впровадженню програмних рішень, що дозволяють автоматизувати бюрократичні процедури та покращити якість взаємодії з громадянами. З розвитком інформаційних технологій з'явилася низка програмних продуктів, які частково або повністю покривають функціональні потреби, пов'язані з реєстрацією, обробкою, маршрутизацією та аналізом звернень громадян. Ці рішення активно застосовуються як у великих містах, так і на рівні територіальних громад, однак ступінь їхньої ефективності, відкритості коду, адаптованості до українського законодавства та гнучкості конфігурації залишається неоднозначним.

Серед доступних на ринку продуктів можна відзначити як універсальні платформи управління документообігом, так і вузькоспеціалізовані системи, орієнтовані виключно на сферу адміністративних послуг. Водночас частина існуючих рішень має обмежений функціонал, не передбачає розширеної аналітики або не дозволяє гнучко модифікувати модулі згідно зі зміною регламентів та законодавчих норм. У деяких випадках такі системи орієнтовані переважно на внутрішнє використання і не мають повноцінної інтеграції з публічними онлайн-сервісами. Усе це створює об'єктивну необхідність критично оцінити існуючі розробки та визначити напрями для створення або вдосконалення програмного забезпечення, що відповідатиме сучасним вимогам функціональності, безпеки та відкритості.

DocStar ECM – це корпоративна система електронного керування контентом, що призначена для автоматизації документообігу, зберігання, обробки та організації електронних документів у межах підприємств, державних установ та організацій різного профілю (рис. 1.4). Основна мета застосунку полягає у зменшенні обсягу паперової роботи, підвищенні прозорості внутрішніх процесів, покращенні контролю за виконанням задач та прискоренні циклів ухвалення рішень [7]. DocStar використовується для централізованого керування

договорами, заявками, фінансовими документами, зверненнями, кадровими записами та іншими видами адміністративної документації.

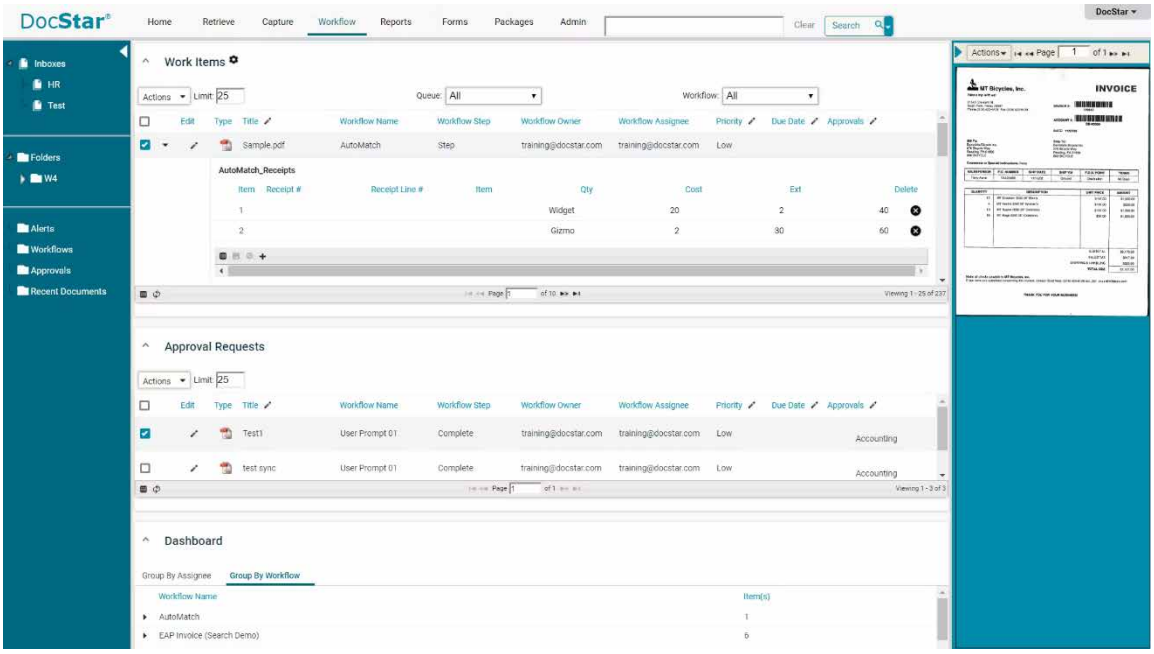


Рис. 1.4 Приклад інтерфейсу системи «DocStar ECM» [8]

Архітектура системи базується на модульному підході з використанням веборієнтованого інтерфейсу. Застосунок підтримує як локальне розгортання у внутрішній інфраструктурі замовника, так і хмарну модель (SaaS), що забезпечує гнучкість у розгортанні та масштабуванні. Серверна частина реалізує логіку обробки запитів, управління базами даних, контролем доступу та маршрутизацією задач. Клієнтська частина представлена через браузерний інтерфейс, сумісний із сучасними вебтехнологіями, що дозволяє працювати з документами з будь-якого пристрою. До складу системи входять засоби індексації, повнотекстового пошуку, генерації звітів, електронного підпису, інтеграції з іншими ERP та CRM-системами. Завдяки гнучким налаштуванням бізнес-процесів DocStar ECM дозволяє моделювати складні адміністративні сценарії без потреби програмування, що робить його придатним для широкого спектра застосувань у сфері управління інформацією.

Переваги DocStar ECM:

- гнучка архітектура з можливістю локального та хмарного розгортання;

- потужний модуль для побудови бізнес-процесів без потреби у програмуванні [9];

- інтеграція з популярними ERP/CRM-системами та підтримка електронного підпису;

- дружній вебінтерфейс з повнотекстовим пошуком та високим рівнем доступності.

Недоліки DocStar ECM:

- висока вартість впровадження та ліцензування для невеликих організацій;

- обмежена локалізація інтерфейсу, зокрема – відсутність української мови;

- потреба у попередньому налаштуванні інфраструктури при локальному встановленні [10];

- деякі користувачі відзначають складність адаптації персоналу на початкових етапах.

Отже, DocStar ECM є потужним засобом електронного керування контентом, що поєднує гнучкість, функціональність і масштабованість. Завдяки широким можливостям інтеграції та налаштування ця система придатна для використання в організаціях, які прагнуть підвищити прозорість, контроль і ефективність своїх документальних процесів. Водночас її повноцінне використання потребує значних ресурсів на впровадження та початкове навчання персоналу.

Laserfiche – це інтегрована система керування корпоративним контентом і бізнес-процесами, розроблена для автоматизації документообігу, цифрової трансформації адміністративних процедур та забезпечення централізованого контролю над інформаційними потоками (рис. 1.4). Основне призначення застосунку полягає у створенні цифрового середовища для обробки документів, організації процесів затвердження, маршрутизації та зберігання записів у

відповідності до внутрішніх регламентів та зовнішніх нормативних вимог [11]. Система широко використовується у державних установах, фінансовому секторі, освіті та охороні здоров'я.

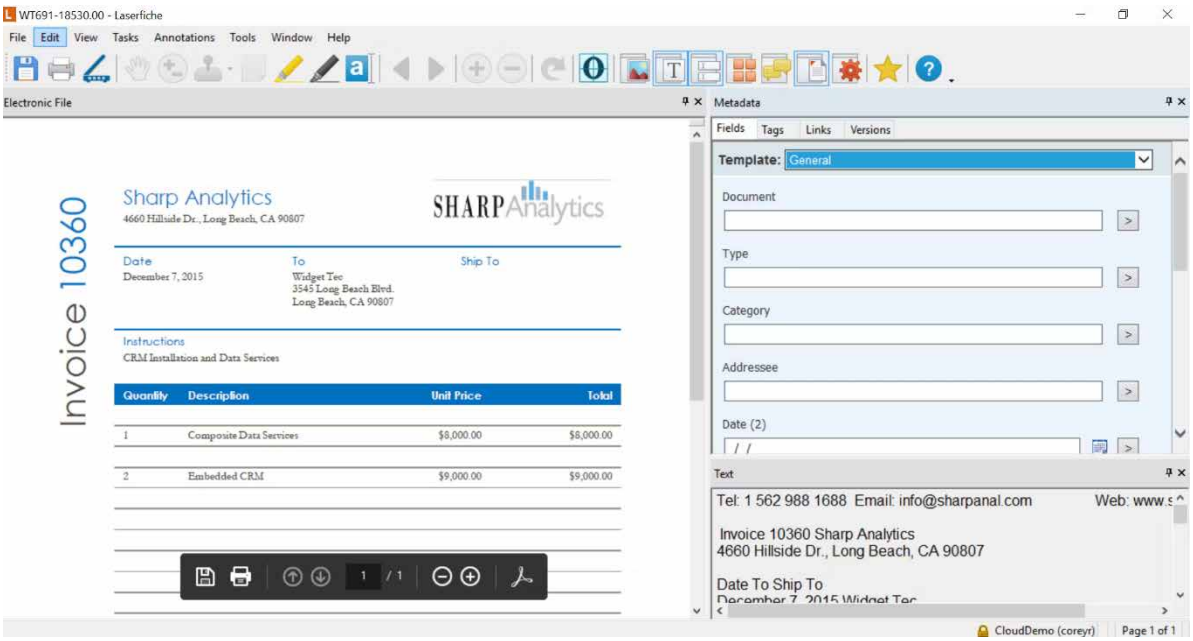


Рис. 1.5 Приклад інтерфейсу системи «Laserfiche» [12]

Архітектура Laserfiche побудована на базі вебтехнологій та підтримує як хмарне, так і локальне розгортання. Основні компоненти включають серверну платформу з обробкою логіки бізнес-процесів, базу даних для зберігання метаданих, файлове сховище для документів, а також клієнтський інтерфейс, доступний через браузер або десктопний застосунок. До системи інтегровано конструктор процесів, за допомогою якого користувачі можуть створювати адаптивні схеми документообігу із розгалуженими умовами та подіями. Laserfiche забезпечує високий рівень безпеки, контроль доступу на основі ролей, ведення журналів дій та інтеграцію з Active Directory. Крім того, у систему вбудовані аналітичні модулі для візуалізації даних і контролю ефективності процесів, що робить її інструментом не лише управління контентом, а й стратегічного аналізу діяльності установи.

Переваги Laserfiche:

- потужний візуальний конструктор бізнес-процесів із підтримкою умов, маршрутів і автоматизації [13];

- висока масштабованість та можливість інтеграції з численними зовнішніми системами (ERP, CRM, AD тощо);

- підтримка як хмарної, так і локальної інфраструктури з гнучкими параметрами безпеки;

- вбудовані інструменти аналітики та моніторингу ефективності адміністративних процесів.

Недоліки Laserfiche:

- складність первинного налаштування та потреба у залученні сертифікованих фахівців для впровадження;

- висока вартість ліцензування та технічної підтримки [14];

- надмірна функціональність для невеликих установ, що потребують базових можливостей документообігу;

- обмежена кількість офіційних локалізацій, зокрема відсутність повної підтримки української мови.

Отже, Laserfiche є високотехнологічною платформою для комплексної цифрової трансформації документообігу та адміністративних процедур. Її функціональність орієнтована на середні й великі організації, які мають складну структуру бізнес-процесів та потребують повної автоматизації з можливістю розширеного контролю, безпеки й аналітики. Проте система вимагає серйозних інвестицій на етапі впровадження й навчання персоналу, що слід враховувати при ухваленні рішення щодо її використання.

M-Files – це інтелектуальна система керування інформацією, розроблена для автоматизації документообігу, покращення пошуку, обліку та контролю ділової інформації незалежно від місця її зберігання (рис. 1.6). Основне призначення цього застосунку полягає у впровадженні контекстно-орієнтованого доступу до документів і записів, що дає змогу організаціям ефективно керувати контрактами, заявами, запитами, внутрішніми процедурами та регламентованими документами [15]. Система орієнтована на гнучку

адаптацію до бізнес-процесів і потреб різних галузей, зокрема юридичного, фінансового, державного та виробничого секторів.

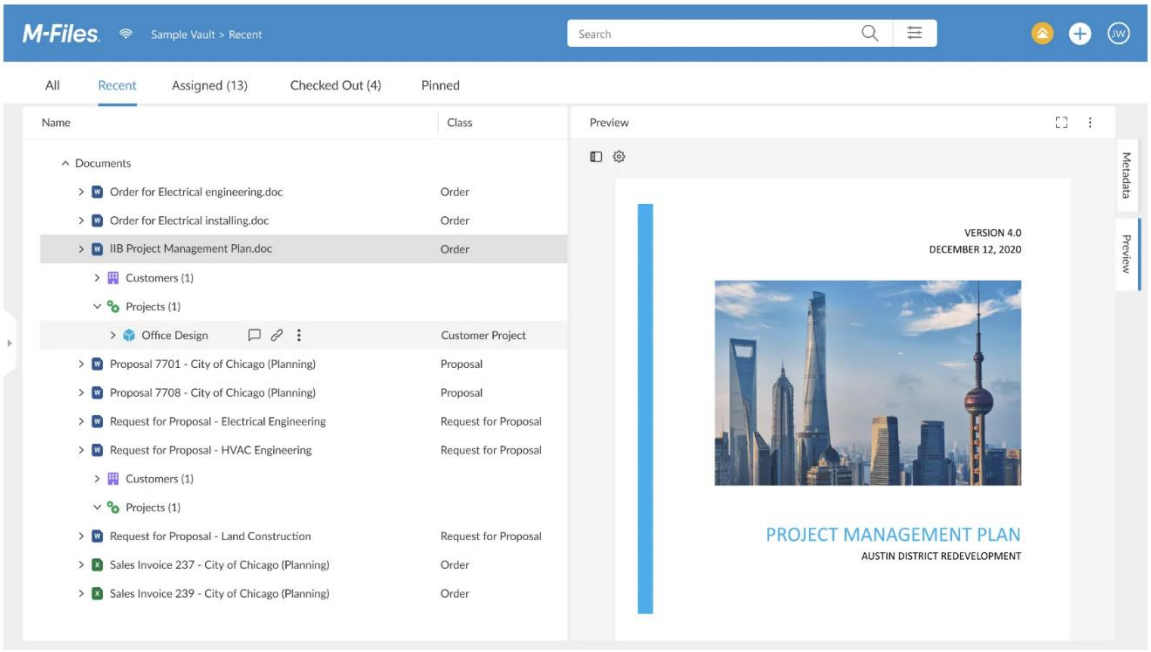


Рис. 1.6 Приклад інтерфейсу системи «M-Files» [16]

Архітектура M-Files базується на метаданих, що дозволяє відмовитися від традиційної ієрархії папок і працювати з інформацією за її сутністю – типом, станом, автором або іншими властивостями. Система має серверний компонент, який відповідає за зберігання документів, базу метаданих, контроль версій та підтримку автоматизованих робочих процесів. Доступ до функціоналу здійснюється через клієнтський інтерфейс, що реалізований у вигляді десктопного застосунку, вебінтерфейсу та мобільних додатків. M-Files підтримує локальне, хмарне або гібридне розгортання, забезпечує інтеграцію з Microsoft 365, Salesforce, SharePoint, мережевими дисками та сторонніми СУБД. До системи вбудовано засоби автоматизації процесів, сповіщення, контроль термінів, електронний підпис та інструменти аналітики, що робить її універсальним рішенням для управління інформаційними потоками та цифрової модернізації організаційної діяльності.

Переваги M-Files:

- унікальна архітектура на основі метаданих, що забезпечує зручний доступ до інформації за сутністю, а не місцем зберігання;

- гнучке розгортання – локальне, хмарне або гібридне – із високою адаптивністю до інфраструктури замовника [17];

- розширена інтеграція з популярними офісними та корпоративними платформами, такими як Microsoft 365, SharePoint, Salesforce;

- інтуїтивно зрозумілий інтерфейс і потужні засоби автоматизації бізнес-процесів без потреби у програмуванні.

Недоліки M-Files:

- початкове налаштування структури метаданих може бути складним і потребує ретельного аналізу процесів;

- вартість ліцензій та технічної підтримки може бути суттєвою для малих організацій;

- високі вимоги до навчання персоналу через відмінність концепції від традиційних файлових систем [18];

- часткові обмеження щодо локалізації і можливості зміни інтерфейсу під нетипові мови (у тому числі українську).

Отже, M-Files є інноваційною системою керування інформацією, яка забезпечує сучасний, контекстно-орієнтований підхід до документообігу та обробки бізнес-даних. Її модель, інтегрованість і гнучкість роблять платформу особливо корисною для середніх і великих організацій, які прагнуть до цифрової трансформації та ефективного управління інформаційними активами. Проте ефективне використання системи потребує методичної підготовки, належного проєктування та навчання персоналу.

Аналіз існуючих програмних рішень засвідчив, що більшість із них орієнтовані на великі організації, мають складну структуру, потребують високих витрат на впровадження та значного навчання персоналу. У багатьох випадках функціональність перевищує реальні потреби невеликих установ, які прагнуть лише базової автоматизації звернень та елементарного контролю процесів. Крім того, більшість рішень не мають повноцінних безкоштовних версій або не

підтримують українську локалізацію. Це створює об'єктивну доцільність розробки нового, простішого у використанні та доступного програмного продукту, орієнтованого на легке розгортання, мінімальні технічні вимоги та відкритість у використанні для малих організацій і місцевих органів влади.

1.3 Постановка завдання

У межах автоматизації бюрократичних процедур, пов’язаних з обробкою звернень громадян до органів державної або муніципальної влади, виникає необхідність у створенні прикладного програмного рішення, яке забезпечить зручний і структурований облік запитів, супровід адміністративних операцій, ведення довідників та аналітичну обробку накопичених даних. Основна мета полягає у побудові компактної, доступної та простої у впровадженні системи з чітко визначеним функціоналом і можливістю використання у невеликих установах без надмірних витрат.

Передбачається реалізація таких ключових функцій:

- реєстрація користувачів і керування обліковими записами;
- створення, перегляд, редагування та видалення звернень громадян;
- ведення довідників типів звернень, посад, відділів і громадян;
- збереження історії обробки звернень;
- формування статистичних звітів за критеріями: середня кількість звернень на тип, громадяни з понад трьома зверненнями, найпоширеніші типи за 6 місяців, звернення у неробочий час, розподіл за статусами;
- перегляд системних подій користувачів і змін у системі.

Для зберігання структурованих даних у проєкті використовуються такі основні таблиці бази даних:

- Користувачі – для обліку облікових записів і ролей;
- Журнал подій – для реєстрації дій користувачів у системі;

- Громадяни – зберігання даних про осіб, які подають звернення;
- Звернення – основна таблиця з реєстрацією запитів;
- Типи звернень – довідник категорій запитів;
- Відділи – інформація про структурні підрозділи;
- Посади – перелік посад працівників установи.

Технічна реалізація програмного забезпечення буде виконана з використанням мови програмування C# у середовищі .NET із побудовою графічного інтерфейсу на основі Windows Forms. Для зберігання даних застосовується система управління базами даних Microsoft SQL Server.

Запропоноване завдання передбачає розроблення десктопного застосунку, що охоплює усі основні потреби в обробці адміністративних звернень. Реалізація такої системи дозволить скоротити витрати часу на облік запитів, підвищити прозорість процесів і сформувати підґрунтя для подальшої цифрової трансформації установи.

1.4 Функціональні та нефункціональні вимоги

У процесі формування вимог до системи автоматизації бюрократичних процедур, зокрема в частині обробки звернень громадян, доцільно виділити ключові групи вимог: функціональні, нефункціональні та вимоги до інтерфейсу користувача. Вони мають на меті забезпечити ефективність, надійність та простоту використання програмного продукту в умовах роботи державних установ, територіальних громад або інших організацій, які здійснюють облік і розгляд звернень. Функціональні вимоги окреслюють основні можливості системи, пов’язані з реєстрацією, опрацюванням, маршрутизацією та аналітичною обробкою звернень у межах встановленого регламенту. Вони описують поведінку системи при взаємодії з користувачем, визначають, як саме відбувається обробка інформації, які операції доступні для різних типів

користувачів і якими мають бути результати виконання тих чи інших дій. Такий підхід дозволяє створити програмне забезпечення, здатне адаптуватися до специфіки конкретної установи, водночас дотримуючись загальних вимог до прозорості, контрольованості та доступності бюрократичних процедур. У подальшому розділі буде детально розглянуто перелік функціональних і нефункціональних вимог до розроблюваної системи.

Таблиця 1.1 – Функціональні вимоги до застосунку

Вимоги	Опис
REQ-1	Реєстрація користувачів та автентифікація з доступом до функцій згідно з роллю
REQ-2	Введення, редагування, перегляд та видалення звернень громадян із фіксацією дати та статусу
REQ-3	Перегляд історії обробки звернень з урахуванням усіх змін у статусі та описах
REQ-4	Управління інформацією про громадян, які подають звернення, включаючи контактні дані
REQ-5	Адміністрування довідників: типи звернень, відділи, посади
REQ-6	Ведення обліку користувачів системи з можливістю перегляду й редагування профілю
REQ-7	Фіксація подій у системі з відображенням дій користувачів у журналі
REQ-8	Формування звіту про середню кількість звернень за кожним типом
REQ-9	Побудова списку громадян, які подали понад три звернення
REQ-10	Автоматичне визначення та виведення звернень, що подані після 18:00 або у вихідні дні

Ці вимоги створюють основу для системи, яка забезпечує цифрову підтримку бюрократичних процедур, дозволяючи користувачам ефективно організовувати облік звернень, контролювати адміністративні процеси та отримувати аналітичну інформацію для прийняття обґрунтованих управлінських рішень.

Нефункціональні вимоги описують загальні характеристики застосунку, які визначають рівень його надійності, продуктивності, зручності використання та безпеки. Для системи, що автоматизує бюрократичні процеси, ці вимоги

мають особливе значення, оскільки вона буде використовуватись у щоденній роботі установ, де важлива стабільність роботи, швидкість доступу до інформації та інтуїтивність інтерфейсу. Крім того, враховуючи обмежені ресурси багатьох установ, важливо забезпечити легкість впровадження і відсутність залежності від складних або дорогих інструментів. Нефункціональні вимоги також сприяють підтримці функціональності застосунку при зростанні обсягу даних і кількості користувачів. У таблиці 1.2 наведено перелік основних нефункціональних вимог, які визначають якість роботи системи.

Таблиця 1.2 – Нефункціональні вимоги до застосунку

Вимоги	Опис
NFR-1	Система повинна забезпечувати стабільну роботу без збоїв під час виконання типових щоденних операцій
NFR-2	Інтерфейс повинен бути простим та зрозумілим для користувачів з базовими навичками роботи з ПК
NFR-3	Застосунок має демонструвати швидкий відгук на дії користувача, включаючи обробку звернень і завантаження списків
NFR-4	Програмне забезпечення повинно бути сумісним із популярними версіями ОС Windows без потреби встановлення сторонніх компонентів
NFR-5	Структура системи має передбачати можливість розширення, зокрема додавання нових типів довідників або звітів
NFR-6	Усі зміни в системі повинні фіксуватись у журналі подій з прив’язкою до користувача та часу
NFR-7	Доступ до системи повинен бути захищеним через механізм автентифікації з унікальними обліковими записами
NFR-8	Інтерфейс користувача має бути локалізованим українською мовою з чіткою термінологією та однозначними підписами

Ці вимоги гарантують, що застосунок буде не лише функціональним, а й зручним, безпечним, адаптованим до контексту використання в державних або комунальних структурах, і здатним до подальшого вдосконалення.

1.5 Вимоги до інтерфейсу користувача

Інтерфейс користувача є важливою складовою будь-якого програмного забезпечення, орієнтованого на щоденне використання в умовах адміністративної роботи. У контексті автоматизації бюрократичних процесів, де переважають рутинні дії із введення, перегляду та аналізу звернень громадян, зручність інтерфейсу набуває критичного значення. Система повинна мати інтуїтивно зрозумілу логіку навігації, мінімалістичний дизайн та єдину стилістику оформлення всіх екранів і форм. Оскільки кінцевими користувачами є працівники органів влади або установ із базовим рівнем володіння комп’ютером, інтерфейс має бути максимально простим, без перевантаження надлишковими елементами.

Інтерфейс має бути реалізований у вигляді десктопного застосунку для ОС Windows із віконною структурою. Основні модулі повинні бути доступні через головне меню або панель навігації. Кожна форма повинна містити чіткі підписи до полів, випадаючі списки для вибору значень з довідників, кнопки з однозначними діями. Важливу роль відіграє уніфікованість усіх вікон: однакова структура редагування записів, однакове розміщення кнопок дій, наявність повідомлень про успішність або помилки.

Передбачено також реалізацію повідомлень про помилки з поясненням причин, повідомлень про підтвердження важливих дій (наприклад, видалення запису), а також базових візуальних елементів контролю введення (обов’язковість полів, перевірка формату email чи номера телефону). Окремо варто зазначити, що інтерфейс має бути повністю локалізованим українською мовою, включаючи системні повідомлення, заголовки та підказки.

Нижче у табл. 1.3 наведено перелік основних вимог до інтерфейсу користувача для даної системи.

Таблиця 1.3 – Вимоги до інтерфейсу користувача

Вимога	Опис
1	2

UI-1	Інтерфейс має бути реалізований у вигляді десктопного застосунку для ОС Windows
UI-2	Всі основні модулі повинні бути доступні через панель навігації або меню головного вікна
UI-3	Вікна введення даних повинні містити чітко підписані поля, логічно згруповані за змістом
UI-4	Кнопки дій (додати, зберегти, редагувати, видалити) мають бути однаково розташовані в усіх формах

Продовження таблиці 1.3

1	2
UI-5	Повідомлення про помилки або успішні дії повинні відображатися у зрозумілій формі з поясненням
UI-6	Поля з некоректними або обов’язковими даними повинні візуально виділятися (наприклад, червоним кольором)
UI-7	Інтерфейс має бути повністю локалізований українською мовою
UI-8	Система повинна запам’ятовувати останні дії користувача (наприклад, фільтри, останній відкритий модуль)
UI-9	Усі елементи інтерфейсу мають бути доступними з клавіатури та відповідати принципам доступності
UI-10	При закритті вікна без збереження змін має відображатися підтвердження дії користувача

Загалом, вимоги до інтерфейсу користувача формуються з урахуванням практичної зручності, однозначності дій та простоти освоєння системи персоналом без спеціальної ІТ-підготовки. Такий підхід дозволяє забезпечити ефективну взаємодію користувача з програмним забезпеченням у щоденній роботі установи.

Висновки до 1 розділу

У межах першого розділу було проведено всебічний аналіз предметної області, зосереджений на цифровій трансформації бюрократичних процедур та впровадженні сучасних ІТ-рішень для оптимізації взаємодії між громадянами й

органами публічної влади. На основі побудованих схем і діаграм визначено ключові особливості функціонування адміністративних процесів, виявлено проблеми, що мають бути усунені за допомогою інформаційних технологій, та окреслено відповідні технічні і функціональні зони.

Проведено огляд сучасних програмних продуктів, зокрема DocStar ECM, Laserfiche та M-Files, із визначенням їх сильних сторін і наявних обмежень. Встановлено, що наявні системи або орієнтовані на корпоративне середовище з вузькою спеціалізацією, або вимагають суттєвих ресурсних витрат на адаптацію до вимог державного управління, що зумовлює доцільність створення нової цільової системи.

На основі аналізу сформульовано постановку задачі, яка охоплює розробку інформаційної системи, здатної автоматизувати облік звернень громадян, забезпечити статистичну аналітику, підвищити прозорість процедур та створити зручний графічний інтерфейс для користувачів. Окрему увагу приділено формалізації функціональних і нефункціональних вимог, а також характеристикам інтерфейсу користувача, орієнтованого на інтуїтивну взаємодію та підтримку основних сценаріїв роботи в адміністративному середовищі.

Результати розділу створили концептуальне, аналітичне й технічне підґрунтя для подальшого моделювання, проєктування й реалізації системи.

2 МОДЕЛЮВАННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

2.1 Об’єктне та функціональне моделювання

У процесі розроблення програмного забезпечення для автоматизації бюрократичних процедур особливу роль відіграє попереднє моделювання предметної області. Саме моделі дозволяють чітко структурувати ключові компоненти системи, виявити логічні зв’язки між ними, а також задати основу для архітектурного проєктування. У цьому контексті доцільним є використання мови візуального моделювання UML (Unified Modeling Language), яка є стандартом у галузі об’єктно-орієнтованого аналізу та проєктування.

UML забезпечує універсальні засоби для представлення структури та поведінки інформаційної системи у вигляді діаграм, що полегшують розуміння функціонування окремих елементів та їх взаємодії. Основою побудови моделей є принцип абстрагування: у межах кожної діаграми фіксуються лише ті аспекти системи, що є суттєвими на конкретному етапі розробки, тоді як другорядні елементи можуть бути опущені для забезпечення фокусування на логіці роботи. Важливо також ураховувати принцип моделювання, згідно з яким комплексне програмне забезпечення не може бути адекватно описане лише однією діаграмою. Залежно від потреб проєкту, створюються окремі діаграми класів, варіантів використання, діяльності, послідовностей тощо.

Для даної системи моделювання дозволяє описати як структуру зберігання даних (довідники, звернення, користувачі), так і логіку взаємодії користувачів із системою (реєстрація, обробка запитів, перегляд звітів). Це дає змогу формалізувати вимоги до системи ще до початку кодування та спростити комунікацію між розробником і замовником. Згодом такі моделі можуть бути використані як база для автоматичної генерації шаблонного коду або як основа для реверс-інжинірингу вже реалізованого функціоналу.

Використання UML-моделей дозволяє сформувати уніфіковане бачення архітектури програмного забезпечення, зменшити кількість помилок на етапі проєктування та створити фундамент для подальшої розробки, тестування та підтримки системи автоматизації бюрократичних процесів.

На етапі концептуального проєктування інформаційної системи, що автоматизує облік звернень громадян, особливого значення набуває побудова об’єктних та функціональних моделей. Об’єктне моделювання дозволяє визначити основні сутності системи, їхні властивості та взаємозв’язки, що є базовими для структури бази даних і логіки взаємодії між компонентами. Функціональне моделювання, своєю чергою, фокусується на описі поведінки системи з позиції користувача, тобто на визначенні варіантів використання, які відображають конкретні сценарії взаємодії з програмним забезпеченням.

У табл. 2.1 наведено опис основних варіантів використання системи, які охоплюють усі ключові функції, необхідні для реалізації автоматизованого обліку та обробки звернень громадян.

Таблиця 2.1 – Опис варіантів використання системи

Варіант	Ім’я	Опис
1	2	3
UC1	Реєстрація користувача	Дозволяє користувачам створювати облікові записи в системі
UC2	Авторизація користувача	Надає можливість проходити ідентифікацію в системі
UC3	Перегляд списку відділів	Дає змогу користувачам переглядати існуючі структурні підрозділи
UC4	Створення нового відділу	Дозволяє додавати нові записи про відділи до довідника
UC5	Редагування відділу	Забезпечує можливість оновлення даних про існуючі відділи
UC6	Перегляд посад	Дозволяє користувачам переглядати список посад у довіднику
UC7	Додавання нової посади	Дозволяє вносити нові посади до системного довідника
UC8	Модифікація посади	Забезпечує редагування записів про посади у довіднику
UC9	Перегляд списку типів звернень	Дає змогу ознайомитися з переліком категорій звернень
UC10	Додавання типу звернення	Дозволяє додавати нові типи звернень до відповідного довідника

Продовження таблиці 2.1

1	2	3
UC11	Редагування типу звернення	Забезпечує редагування існуючих типів звернень
UC12	Перегляд списку громадян	Дозволяє користувачам переглядати зареєстрованих громадян
UC13	Додавання нового громадянина	Дозволяє вносити нового громадянина до бази даних
UC14	Модифікація громадянина	Забезпечує оновлення інформації про наявного громадянина
UC15	Перегляд списку звернень	Дозволяє користувачам переглядати всі зареєстровані звернення
UC16	Додавання нового звернення	Дозволяє створити нове звернення із зазначенням усіх параметрів
UC17	Модифікація звернення	Забезпечує зміну даних у вже зареєстрованому зверненні
UC18	Перегляд історії звернень	Дає змогу бачити послідовність змін і обробки звернень
UC19	Формування статистики	Середня кількість запитів на кожен тип звернення, список громадян з понад 3 запитами, топ-3 типи звернень за 6 місяців, розподіл за статусами, звернення у неробочий час
UC20	Перегляд користувачів	Дозволяє адміністраторам бачити список зареєстрованих користувачів
UC21	Додавання нового користувача	Дозволяє адміністратору створювати новий обліковий запис
UC22	Редагування даних користувача	Дозволяє змінювати інформацію про користувача, зокрема роль і опис
UC23	Перегляд подій	Дозволяє користувачам відстежувати події в системі через журнал активності

Для візуалізації основних функцій, доступних користувачеві з роллю адміністратора, побудовано діаграму варіантів використання, подану на рис. 2.1. Дана діаграма дозволяє графічно відобразити взаємозв’язок між актором

(адміністратором) та можливими сценаріями взаємодії з системою, демонструючи як базові дії, так і розширення до них.

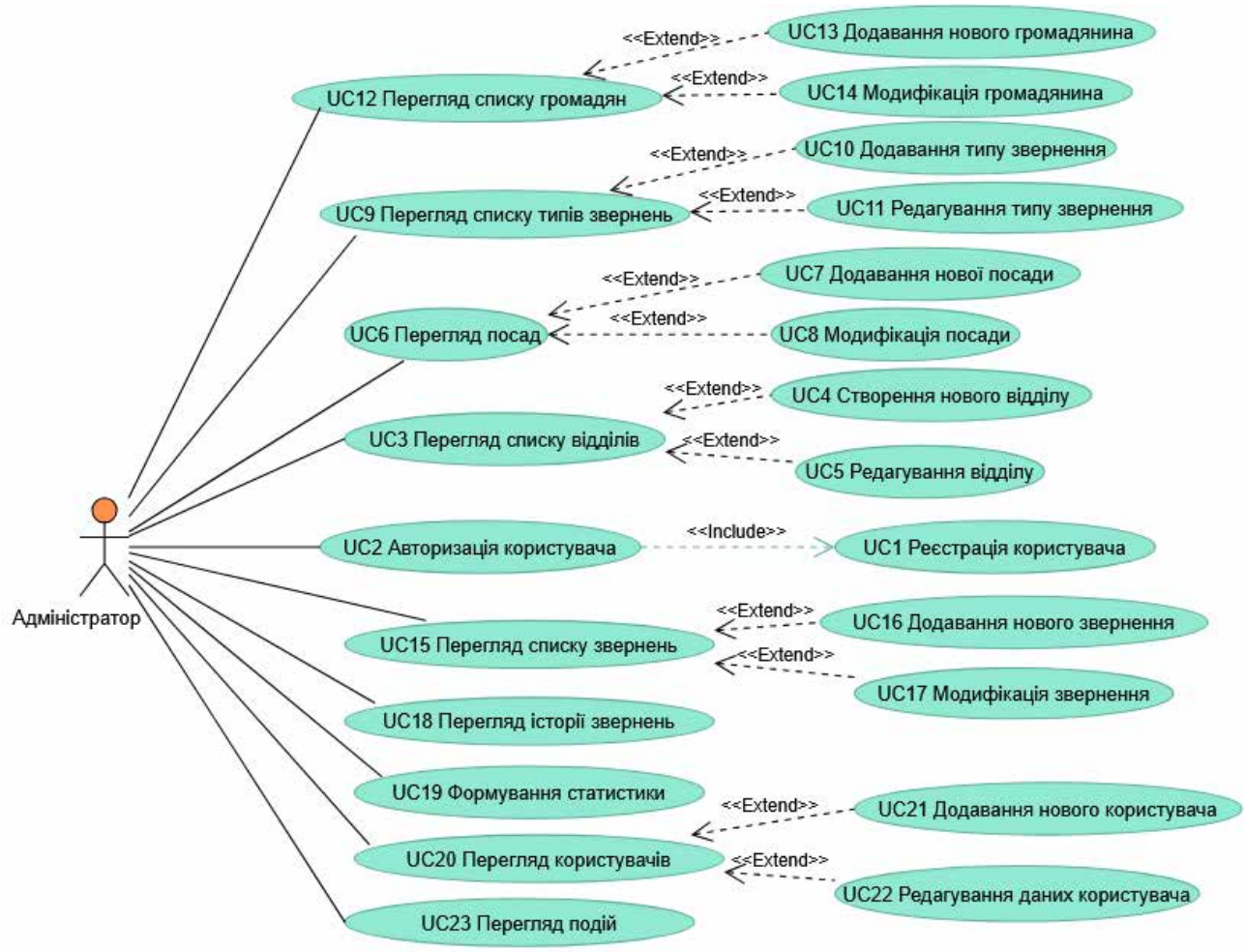


Рис. 2.1 Діаграма варіантів використання для ролі адміністратора

На рисунку 2.1 представлено повний набір варіантів використання, до яких має доступ адміністратор системи. Серед них – управління зверненнями, громадянами, довідниками (типи звернень, посади, відділи), обліковими записами користувачів, а також перегляд історії подій і статистики. Зв’язки типу <<include>> та <<extend>> наочно показують залежності між операціями, наприклад, створення звернення як розширення перегляду списку або обов’язкове включення процесу реєстрації до авторизації. Такий підхід дозволяє не лише структурувати функціонал, а й чітко визначити межі відповідальності користувача з правами адміністратора.

Аналогічно до ролі адміністратора, для користувача системи сформовано окрему діаграму варіантів використання, яка подана на рис. 2.2. Вона відображає

доступний функціонал для звичайного користувача – тобто працівника установи без розширених прав адміністрування.

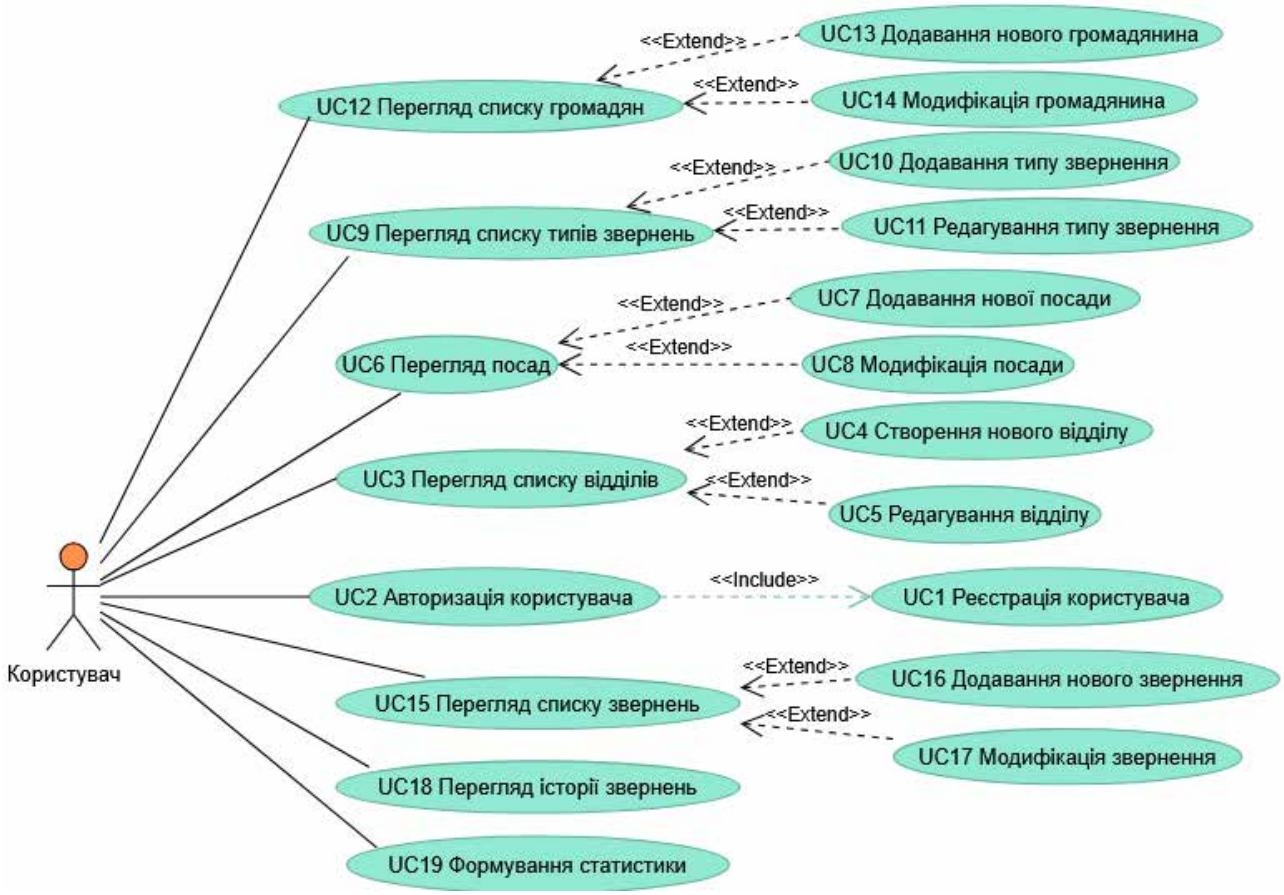


Рис. 2.2 Діаграма варіантів використання для ролі користувача

Діаграми послідовностей у мові UML використовуються для моделювання взаємодії між об’єктами системи в межах конкретного сценарію використання. Вони демонструють порядок обміну повідомленнями між об’єктами та відображають часову послідовність дій. Такі діаграми є особливо корисними для аналізу логіки обробки запитів, уточнення функціональних вимог і перевірки коректності взаємодії між компонентами системи.

На рис. 2.3 наведено діаграму послідовності для сценарію додавання нового громадянина. Вона ілюструє послідовні дії користувача та взаємодію між основними компонентами системи: формою введення CitizensForm, модулем валідації, провайдером збереження інформації про громадян і провайдером подій. Користувач відкриває форму, після чого система ініціалізує порожні поля для введення даних. Після заповнення ПІБ, email, телефону, адреси та дати,

користувач натискає кнопку «Додати». Далі здійснюється перевірка правильності введених даних.

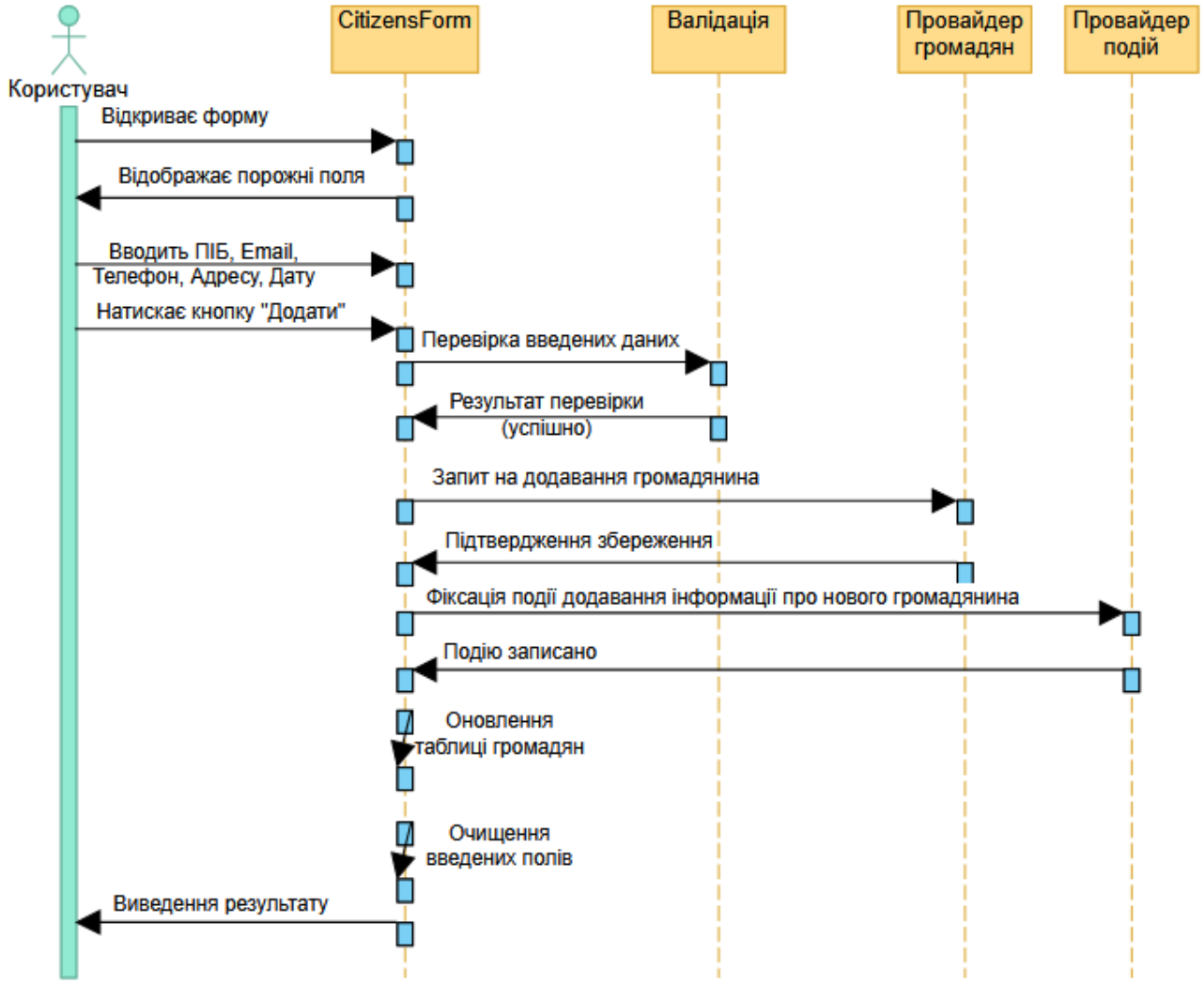


Рис. 2.3 Діаграма послідовності додавання інформації про нового громадянина

У разі успішної валідації формується запит на збереження інформації, який передається до провайдера громадян. Після підтвердження збереження активується провайдер подій, що фіксує подію додавання нового запису. Коли подію успішно зафіксовано, система оновлює таблицю громадян у інтерфейсі, очищає введені поля та відображає результат користувачу. Така схема відображає логіку на рівні користувацького сценарію, забезпечуючи цілісне уявлення про взаємодію елементів системи при створенні нового запису.

На рис. 2.4 зображено діаграму послідовності, яка відображає логіку формування звіту про відсоткове співвідношення звернень за статусами. Цей сценарій є частиною аналітичного функціоналу системи, що дає змогу отримати

уніфіковане уявлення про стан обробки звернень у межах заданого періоду або всієї бази.

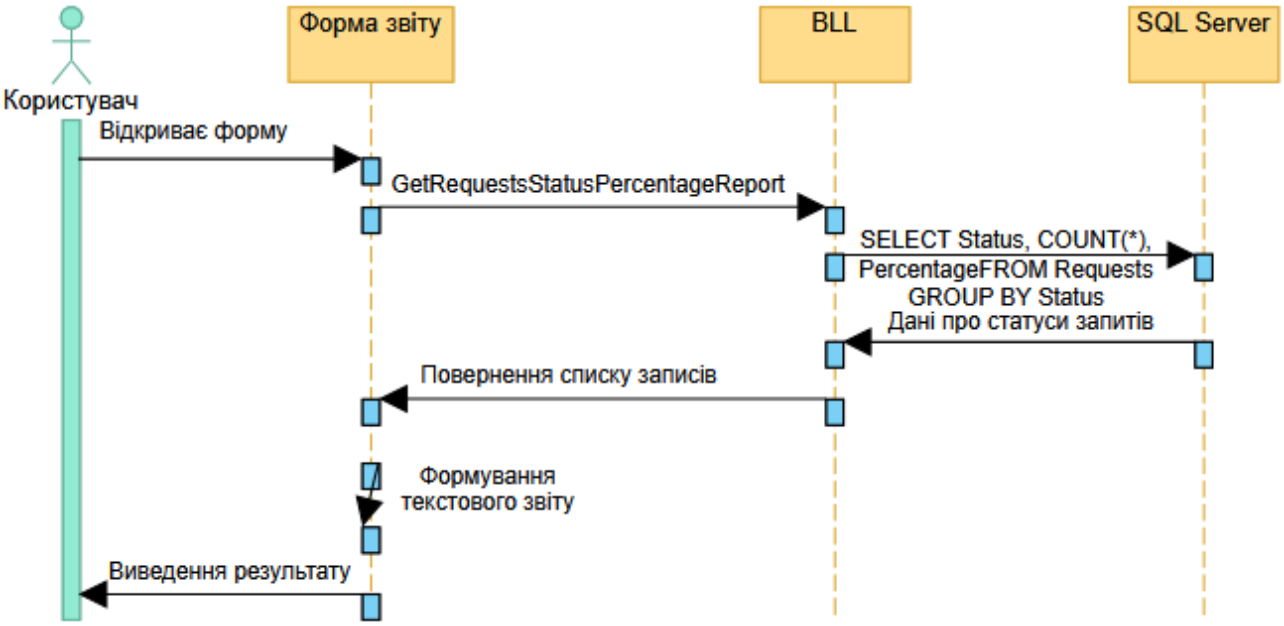


Рис. 2.4 Формування звітності про відсоткове співвідношення запитів за статусами

Користувач ініціює процес, відкриваючи відповідну форму звітності. Після цього відправляється запит GetRequestsStatusPercentageReport до рівня бізнес-логіки (BLL), який у свою чергу формує SQL-запит до бази даних. На рівні СУБД (SQL Server) виконується агрегація записів за полем Status за допомогою запиту SELECT Status, COUNT(*) FROM Requests GROUP BY Status. Результатом є список статусів разом із кількістю звернень для кожного з них.

Після повернення зведених даних до BLL і далі до форми звіту, система виконує формування текстового звіту з обчисленням відсотків для кожної категорії статусу. Завершальним етапом є відображення результату у вікні звіту для користувача. Такий механізм дозволяє оперативно оцінити структуру оброблених і поточних звернень, що є важливою передумовою для управлінського аналізу та прийняття рішень щодо оптимізації бюрократичних процедур.

Діаграми активностей у мові UML слугують для відображення алгоритмів виконання процесів у вигляді послідовностей дій, умов та переходів. Вони дозволяють представити логіку виконання операцій як з боку користувача, так і

всередині системи, включаючи паралельні гілки, цикли, перевірки умов та завершення сценаріїв. Такі діаграми широко застосовуються при проєктуванні прикладного ПЗ для формалізації бізнес-процесів, моделювання користувацької взаємодії та верифікації функціонального потоку.

На рис. 2.5 наведено діаграму активності, яка ілюструє процес додавання нового звернення громадянина.

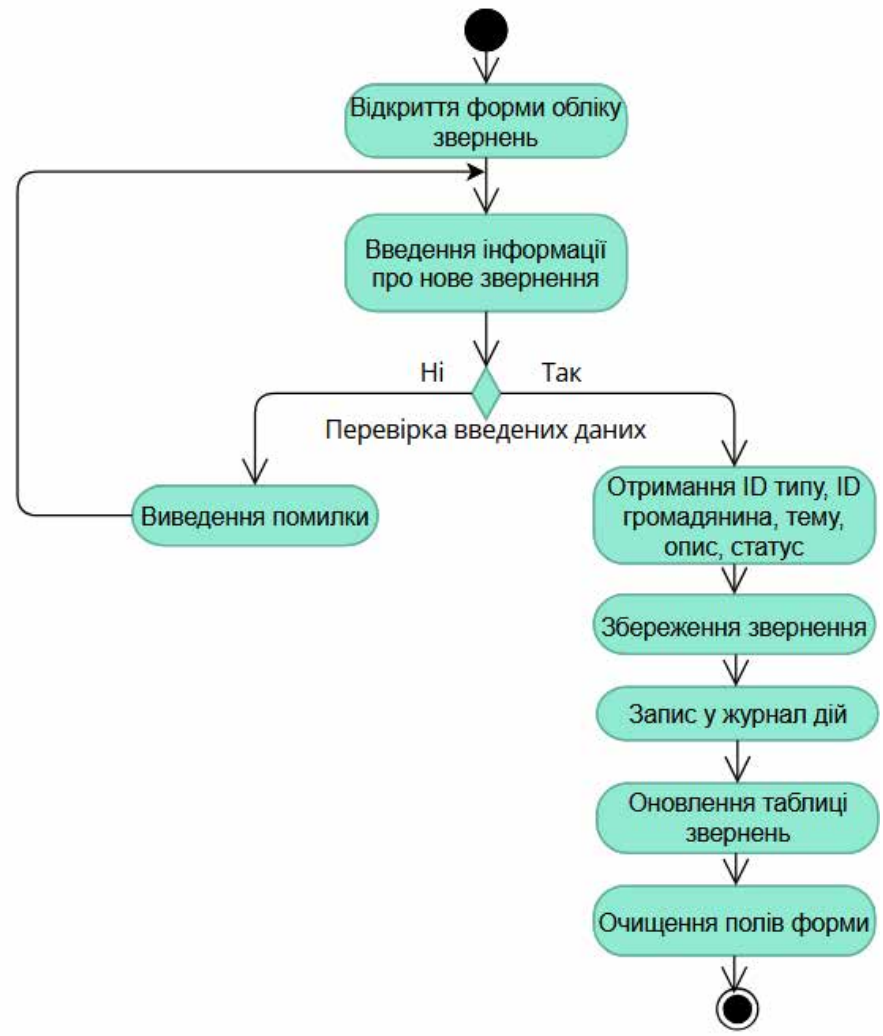


Рис. 2.5 Додавання інформації про нове звернення громадянина

Процес починається з відкриття форми обліку звернень, після чого користувач заповнює необхідні поля, включаючи тип звернення, особу заявника, тему, опис і статус. Система виконує перевірку введених даних, і в разі виявлення помилки – повідомляє користувача, повертаючи його до введення. Якщо всі дані коректні, система отримує відповідні ідентифікатори, проводить збереження звернення у базі даних, а також фіксує подію в журналі дій. Після цього оновлюється таблиця звернень, щоб відобразити новий запис, і

очищуються поля введення для подальшої роботи користувача. Така послідовність дій забезпечує цілісність даних, зручність взаємодії з інтерфейсом та повну трасованість дій у системі.

На рис. 2.6 зображено діаграму активності, яка моделює процес редагування інформації про звернення громадянина.

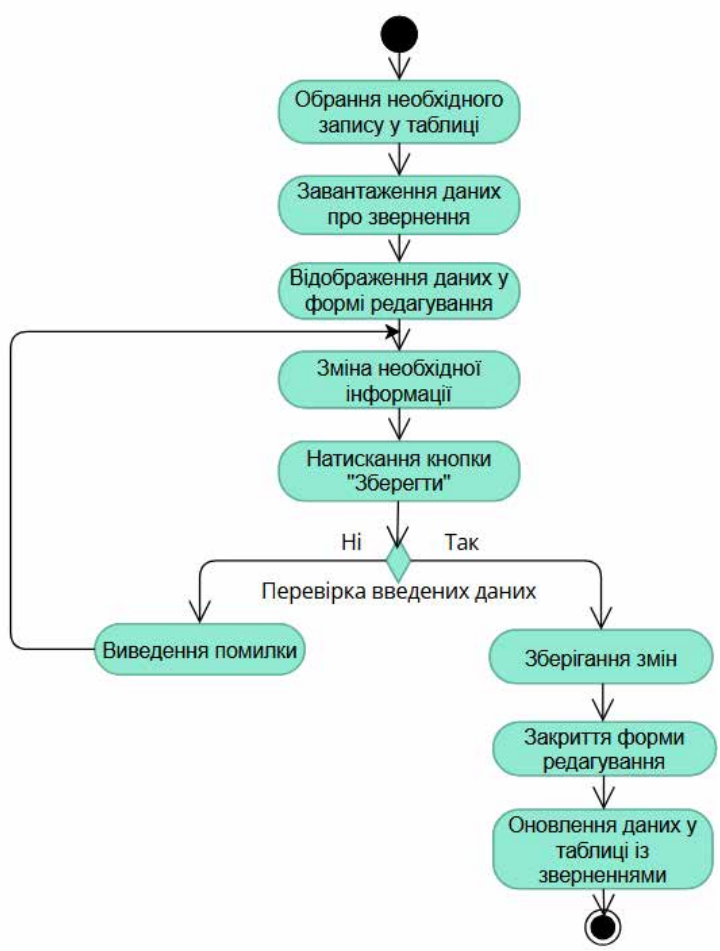


Рис. 2.6 Редагування інформації про звернення громадянина

Редагування інформації про звернення передбачає внесення змін до вже наявного запису в базі даних, що може знадобитися у разі виявлення помилки, уточнення теми звернення, оновлення статусу або зміни опису. Даний процес починається з обрання необхідного запису із таблиці звернень. Після цього система завантажує відповідні дані та відображає їх у формі редагування. Користувач змінює потрібні поля й натискає кнопку «Зберегти». Далі відбувається перевірка введеної інформації: у разі помилок система виводить відповідне повідомлення, повертаючи користувача до редагування. Якщо всі дані коректні, виконується збереження змін, після чого закривається форма

редагування, а таблиця звернень оновлюється для відображення актуалізованої інформації.

2.2 Абстракції предметної області

Абстракція виділяє істотні характеристики деякого об'єкта, що відрізняють його від усіх інших видів об'єктів і, таким чином, чітко окреслює його концептуальні межі з позиції користувача або системного аналітика.

Для розроблюваної системи автоматизації обліку звернень громадян у межах бюрократичних процедур було спроєктовано низку ключових абстракцій, які відображають основні сутності предметної області. Ці абстракції подано на рис. 2.7–2.12.

Абстракція: Громадянин
Важливі властивості: ПІБ (Прізвище, ім’я, по батькові) Email Номер телефону Адреса проживання Дата реєстрації
Обов’язки: Надання звернень до установи Оновлення персональних даних (через працівника установи) Ідентифікація у системі як суб’єкта звернення Подання звітності

Рис. 2.7 Абстракція «Громадянин»

Абстракція: Подія системи
Важливі властивості: Назва типу Опис Дата створення / оновлення
Обов’язки: Категоризація звернень Доступ до вибору у формі реєстрації звернення Участь у побудові статистики за категоріями

Рис. 2.8 Абстракція «Тип звернення»

Абстракція: Звернення
Важливі властивості: ID звернення Громадянин (ідентифікатор) Тип звернення Тема звернення Опис звернення Статус (нове, в обробці, завершене тощо) Дата створення
Обов’язки: Створення нового звернення Редагування або уточнення інформації Зміна статусу у процесі опрацювання Відображення у звітності

Рис. 2.9 Абстракція «Звернення»

Абстракція: Користувач
Важливі властивості: Логін Пароль (захешований) ПІБ Роль (адміністратор / користувач) Дата створення облікового запису
Обов’язки: Авторизація у системі Додавання, редагування та перегляд інформації згідно з роллю Реєстрація нових користувачів (адміністратор) Ведення довідників та обробка звернень

Рис. 2.10 Абстракція «Користувач»

Абстракція: Подія системи
Важливі властивості: Тип події (створення, редагування, видалення, авторизація) Користувач, який ініціював подію Дата й час події Об’єкт події (звернення, громадянин тощо)
Обов’язки: Фіксація дій користувачів у системі Забезпечення прозорості змін Підтримка функції журналювання для аудиту

Рис. 2.11 Абстракція «Доходи»

Абстракція: Відділ
Важливі властивості: Назва відділу Скорочена назва Опис функціонального призначення Дата створення запису
Обов’язки: Участь у маршрутизації та обробці звернень Відображення у довіднику для вибору при заповненні звернення Структуризація даних для внутрішньої аналітики Зв’язок із посадовими особами та користувачами системи

Рис. 2.12 Абстракція «Відділ»

Даний підхід гарантує цілісність даних, мінімізує ризик логічних помилок та забезпечує контрольованість процесу оновлення інформації, що є важливим у межах формалізованих адміністративних процедур.

2.3 Діаграма класів

Діаграма класів є ключовим елементом структурного моделювання інформаційної системи, що відображає логічну модель предметної області у вигляді набору класів, їхніх властивостей (атрибутів), методів та зв’язків між ними. Цей тип діаграми дозволяє формалізовано описати структуру даних, визначити об’єкти, з якими оперує система, а також встановити відношення між об’єктами, включаючи асоціації, агрегації та наслідування. Таке представлення полегшує розробку програмної архітектури, забезпечуючи візуальну уніфікацію логіки взаємодії між складовими системи. Крім того, діаграма слугує ефективним засобом комунікації між розробниками та аналітиками під час проектування.

Для розроблюваної системи автоматизації бюрократичних процедур побудовано діаграму класів, яка представлена на рис. 2.13. Вона охоплює всі основні об'єкти системи, що формують її логічну основу, та відображає їхні взаємозв’язки відповідно до реалізованої структури бази даних.

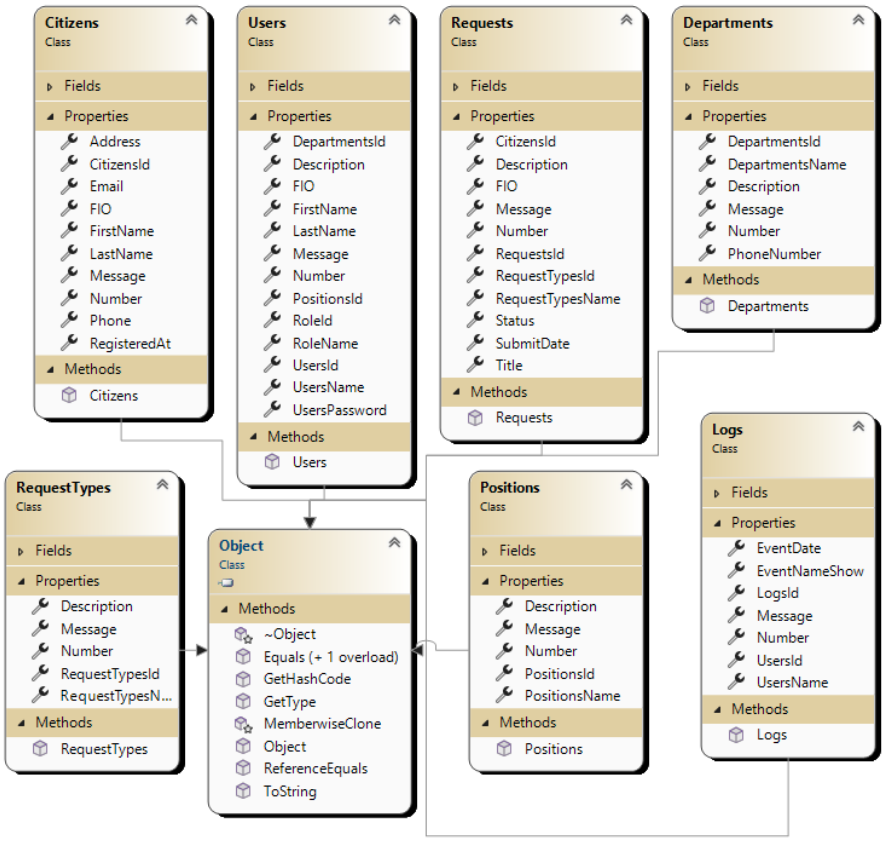


Рис. 2.13 Діаграма класів системи

Діаграма відображає такі ключові класи: Citizens, Users, Requests, RequestTypes, Departments, Positions та Logs. Клас Citizens містить властивості, що описують громадянина (адреса, ПІБ, контактні дані, дата реєстрації), які використовуються при поданні звернень. Клас Requests є центральним і описує інформацію про звернення: прив’язку до громадянина, типу звернення, текст повідомлення, тему, дату подачі тощо. Він має зовнішні ключі, які зв’язують його з таблицями Citizens та RequestTypes.

Клас Users відповідає за облікові записи користувачів системи з розмежуванням ролей, і пов’язаний із таблицями Positions та Departments, де зберігаються службові відомості про користувача. RequestTypes містить перелік типів звернень, а Logs – журнал подій, який фіксує активність користувачів у системі.

Висновки до 2 розділу

У даному розділі виконано формалізацію предметної області шляхом побудови концептуальних моделей, що відображають функціональну та структурну організацію майбутньої інформаційної системи. Обґрунтовано доцільність використання мови візуального моделювання UML як універсального засобу опису складних об’єктно-орієнтованих систем, що дозволяє ефективно представити логіку взаємодії користувачів із системою.

На основі сценаріїв використання побудовано діаграми варіантів використання для основних ролей – адміністратора та звичайного користувача. Для критичних функцій реалізовано діаграми послідовностей, які демонструють динаміку обміну повідомленнями між об’єктами під час додавання громадянина та формування статистики за статусами звернень. Додатково розроблено діаграми активностей, які ілюструють алгоритми обробки звернень на прикладах створення та редагування записів.

Також проведено виділення та опис основних абстракцій предметної області, серед яких – «Громадянин», «Звернення», «Користувач», «Доходи», «Тип звернення» і «Відділ». Кожна з них охарактеризована за ключовими властивостями та функціональними обов’язками, що забезпечило чітке уявлення про об’єкти, які підлягають подальшому програмному відтворенню.

Виконане моделювання дозволило сформувати цілісну структурно-функціональну модель системи, яка забезпечує узгодженість між вимогами користувача та технічною реалізацією, а також є надійним підґрунтям для етапу проектування програмного забезпечення.

3 ПРОЄКТУВАННЯ ТА ЕКСПЛУАТАЦІЯ ПРОГРАМНОЇ СИСТЕМИ

3.1 Логічна модель даних

Логічна модель бази даних відображає структуру даних, які використовуються в межах інформаційної системи, а також описує взаємозв'язки між цими даними без урахування технічних особливостей фізичного зберігання. Вона є фундаментальним етапом у процесі проєктування системи, оскільки визначає, які сутності будуть задіяні, які атрибути вони містять, та яким чином між ними встановлюються логічні зв'язки. Такий рівень абстракції забезпечує узгодженість даних, мінімізує дублювання інформації та слугує базою для побудови реляційної структури бази даних.

На рис. 3.1 наведено логічну модель бази даних для програмного забезпечення, яке автоматизує облік звернень громадян у рамках бюрократичних процедур.

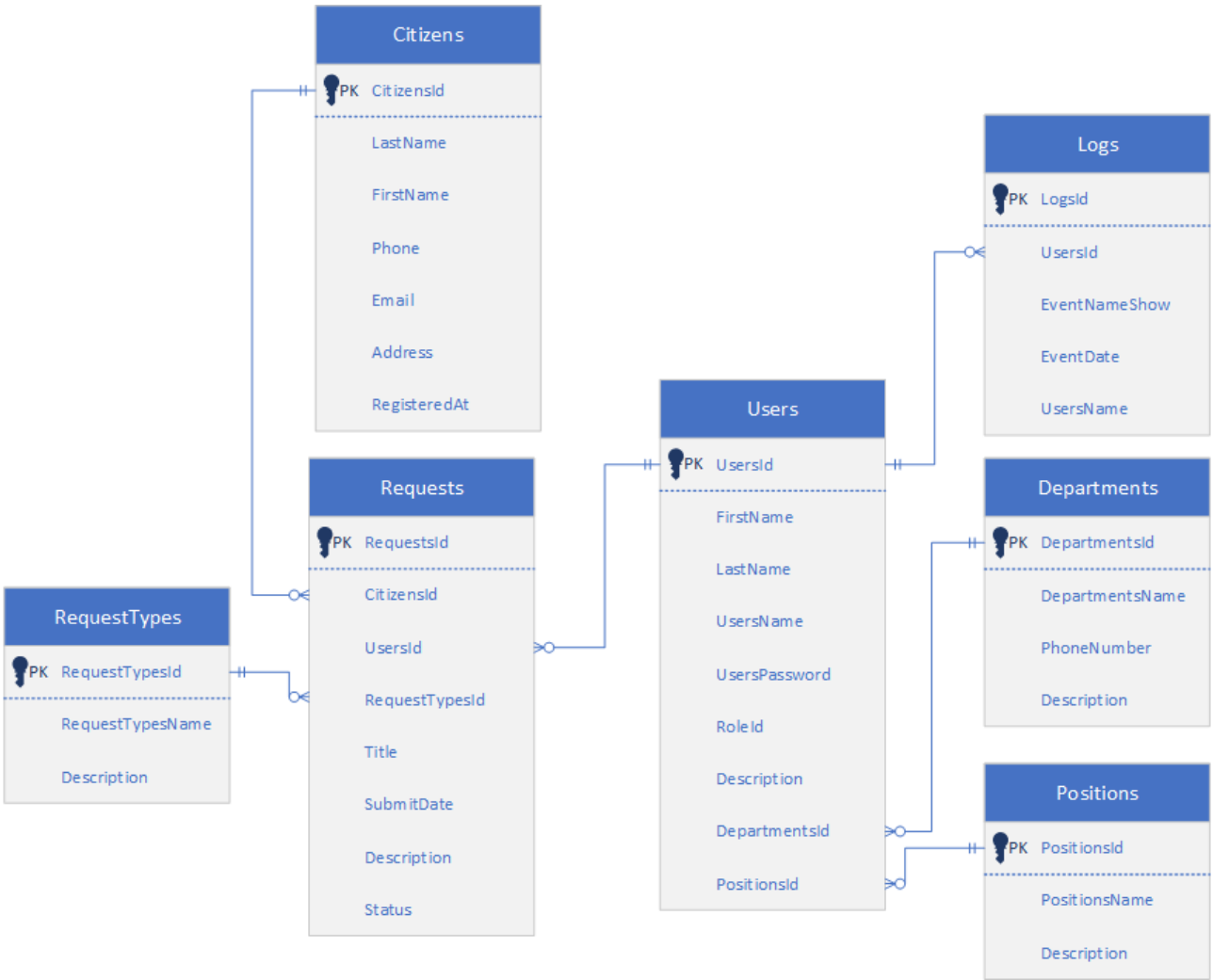


Рис. 3.1 Логічна модель бази даних

Модель включає таблиці, які охоплюють як основні об’єкти предметної області, так і допоміжні довідники. Зокрема, виділено такі сутності: Citizens, Requests, RequestTypes, Users, Departments, Positions та Logs. Усі таблиці взаємопов’язані за допомогою зовнішніх ключів, що забезпечує цілісність даних та дозволяє ефективно реалізувати механізми фільтрації, звітності й контролю змін.

Зв’язок між таблицями реалізовано за наступною логікою: кожне звернення (Requests) прив’язується до конкретного громадянина (Citizens), типу звернення (RequestTypes) та виконавця (Users). Облікові записи користувачів зберігаються в таблиці Users, яка також містить зовнішні ключі на Departments (структурний підрозділ) та Positions (посада). Таблиця Logs слугує для фіксації подій у системі, містячи інформацію про дату, подію, користувача та пов’язану з ним дію.

Побудована логічна модель, формує концептуальний каркас системи управління зверненнями, забезпечуючи логічно цілісну, розширювану та уніфіковану структуру даних, придатну для подальшої реалізації на рівні фізичної СУБД.

3.2 Вибір системи управління базою даних та її реалізація

Під час проєктування інформаційної системи для автоматизації обліку звернень громадян ключовим етапом стало визначення оптимальної системи управління базами даних (СУБД), яка забезпечить надійне зберігання, швидкий доступ і цілісність інформації. До розгляду було відібрано три найпоширеніші СУБД, що мають широку підтримку, стабільні механізми транзакцій і відповідають вимогам проєкту за критеріями масштабованості, простоти інтеграції та ресурсної ефективності. З урахуванням специфіки предметної області, середовища розгортання та рівня технічної підтримки було проведено порівняльний аналіз найбільш релевантних рішень: MS SQL Server, MySQL та Firebird.

MS SQL Server – це потужна реляційна СУБД, розроблена компанією Microsoft, яка підтримує широкий набір інструментів для адміністрування, оптимізації запитів та забезпечення безпеки даних [19]. Вона добре інтегрується з екосистемою .NET, що робить її зручною для розробки систем на мові C#.

MySQL – це одна з найпоширеніших СУБД з відкритим кодом, що відзначається високою продуктивністю, кросплатформеністю та простотою у розгортанні [20]. Вона активно використовується у веброзробці та підтримує базовий набір транзакційних можливостей.

Firebird – це легка, вбудована реляційна СУБД з відкритим кодом, яка не потребує встановлення серверної частини та вирізняється низьким споживанням ресурсів [21]. Вона підтримує повноцінну SQL-функціональність, транзакції, тригери та збережені процедури, що робить її придатною для локальних десктопних застосунків. Firebird часто обирають для систем, що потребують мінімального адміністрування та автономної роботи.

Таблиця 3.1 – Порівняння СУБД за ключовими характеристиками

Характеристика	MS SQL Server	MySQL	Firebird
Інтеграція з .NET / C#	Повна інтеграція	Часткова (через сторонні конектори)	Обмежена
Інструменти адміністрування	SQL Server Management Studio	MySQL Workbench	isql, FlameRobin (обмежено)
Безпека та контроль доступу	Розширена рольова модель, Active Directory	Базова, без інтеграції з AD	Спрощена
Масштабованість	Висока, підтримка реплікацій	Середня	Низька
Розгортання в корпоративному середовищі	Стандарт де-факто для Windows-інфраструктури	Можливе, але менш інтегроване	Переважно для автономних систем
Підтримка складних запитів і транзакцій	Повна підтримка, оптимізація запитів	Підтримується частково	Підтримується, але менш гнучко

Вибір MS SQL Server для розробки системи є обґрунтованим завдяки його високій сумісності з мовою програмування C# та повноцінною підтримкою середовища .NET, що значно спрощує інтеграцію з застосунком. На відміну від альтернатив, ця СУБД забезпечує розширені можливості адміністрування, включаючи централізований контроль доступу та зручні засоби резервного копіювання. Завдяки потужному рушію обробки запитів та підтримці складних транзакцій, MS SQL Server дозволяє ефективно працювати з великими обсягами даних у багатокористувацькому середовищі. Такий вибір гарантує надійність, масштабованість і відповідність вимогам до корпоративного програмного забезпечення.

Після визначення оптимальної СУБД для реалізації системи доцільно перейти до розгляду структури окремих таблиць, що формують її логічну й фізичну модель.

Таблиця «Citizens» призначена для зберігання персональних даних громадян, які є заявниками у системі обліку звернень. У ній фіксуються ключові атрибути, зокрема прізвище, ім’я, контактна інформація та адреса проживання. Також зберігається дата реєстрації особи в системі, що дозволяє відстежувати хронологію поданих звернень (табл. 3.2).

Таблиця 3.2 – Атрибути сутності «Citizens»

№	Найменування	Тип даних	Призначення
1	CitizensId	INT	Ідентифікатор громадянина
2	LastName	NVARCHAR(50)	Прізвище особи
3	FirstName	NVARCHAR(50)	Ім’я особи
4	Email	NVARCHAR(150)	Адреса електронної пошти для зв’язку
5	Phone	NVARCHAR(20)	Контактний номер телефону
6	Address	NVARCHAR(500)	Адреса місця проживання
7	RegisteredAt	DATETIME	Дата та час реєстрації громадянина в інформаційній системі

Таблиця «Requests» призначена для зберігання інформації про звернення громадян до організації. Вона містить посилання на тип звернення, особу, яка

його подала, а також на користувача, що його обробляє. Крім того, таблиця зберігає тему звернення, детальний опис, дату подачі та поточний статус розгляду (табл. 3.3).

Таблиця 3.3 – Атрибути сутності «Requests»

№	Найменування	Тип даних	Призначення
1	RequestsId	INT	Ідентифікатор звернення
2	RequestTypesId	INT	Зовнішній ключ до типу звернення
3	UsersId	INT	Зовнішній ключ до користувача
4	CitizensId	INT	Зовнішній ключ до громадянина, який подав звернення
5	Title	NVARCHAR(200)	Заголовок або тема звернення
6	SubmitDate	DATETIME	Дата й час подачі звернення
7	Description	NVARCHAR(MAX)	Детальний опис звернення
8	Status	NVARCHAR(50)	Поточний статус звернення (наприклад, нове, у роботі, завершене тощо)

Таблиця «RequestTypes» використовується як довідник типів звернень, які подають громадяни. Вона містить унікальну назву кожного типу та розширений опис, який допомагає класифікувати запити за змістом та напрямком розгляду (табл. 3.4).

Таблиця 3.4 – Атрибути сутності «RequestTypes»

№	Найменування	Тип даних	Призначення
1	RequestTypesId	INT	Унікальний ідентифікатор типу звернення
2	RequestTypesName	NVARCHAR(200)	Назва типу звернення (наприклад, скарга, заява, запит)
3	Description	NVARCHAR(MAX)	Деталізований опис типу звернення

Таблиця «Users» призначена для зберігання інформації про облікові записи працівників установи, які мають доступ до системи. У ній містяться дані про імена, логіни, паролі, ролі, службову інформацію, а також посилання на відповідний відділ і посаду користувача. Ця таблиця слугує основою для ідентифікації, авторизації та управління доступом у межах системи (табл. 3.5).

Таблиця 3.5 – Атрибути сутності «Users»

№	Найменування	Тип даних	Призначення
1	UsersId	INT	Ідентифікатор користувача
2	FirstName	NVARCHAR(50)	Ім’я користувача
3	LastName	NVARCHAR(50)	Прізвище користувача
4	UsersName	NVARCHAR(50)	Логін для входу до системи
5	UsersPassword	NVARCHAR(250)	Пароль (у зашифрованому вигляді)
6	RoleId	INT	Ідентифікатор ролі користувача (права доступу)
7	Description	NVARCHAR(1230)	Коментар або службова характеристика
8	DepartmentsId	INT	Посилання на відділ, до якого належить користувач
9	PositionsId	INT	Посилання на посаду, яку займає користувач

Таблиця «Logs» використовується для фіксації подій, які відбуваються в системі внаслідок дій користувачів. Вона містить інформацію про тип події, дату й час її виконання, а також про користувача, який ініціював відповідну дію. Ця таблиця є важливим елементом для аудиту, контролю змін і забезпечення прозорості системної активності (табл. 3.6).

Таблиця 3.6 – Атрибути сутності «Logs»

№	Найменування	Тип даних	Призначення
1	LogsId	INT	Унікальний ідентифікатор події
2	UsersId	INT	Посилання на користувача, що ініціював подію
3	EventNameShow	NVARCHAR(MAX)	Назва або опис виконаної дії
4	EventDate	DATETIME	Дата та час здійснення події
5	UsersName	NVARCHAR(150)	Логін користувача, який виконав дію

Таблиця «Departments» призначена для зберігання інформації про структурні підрозділи (відділи) установи, працівники яких беруть участь в обробці звернень громадян. У таблиці фіксується назва відділу, контактний номер телефону та текстовий опис його функціонального призначення. Ця

інформація використовується для організації внутрішньої структури системи, фільтрації даних та визначення посадових обов’язків користувачів (табл. 3.7).

Таблиця 3.7 – Атрибути сутності «Departments»

№	Найменування	Тип даних	Призначення
1	DepartmentsId	INT	Ідентифікатор відділу
2	DepartmentsName	NVARCHAR(200)	Назва структурного підрозділу
3	PhoneNumber	NVARCHAR(30)	Контактний номер телефону відділу
4	Description	NVARCHAR(MAX)	Опис функцій або сфери відповідальності відділу

Таблиця «Positions» використовується для ведення довідника посад працівників, які зареєстровані в системі. Вона включає назву посади та додаткову текстову характеристику, що може описувати функціональні обов’язки або рівень відповідальності. Дані з цієї таблиці використовуються при створенні та редагуванні облікових записів користувачів, а також у структурній і аналітичній частині системи (табл. 3.8).

Таблиця 3.8 – Атрибути сутності «Positions»

№	Найменування	Тип даних	Призначення
1	PositionsId	INT	Ідентифікатор посади
2	PositionsName	NVARCHAR(200)	Назва посади
3	Description	NVARCHAR(MAX)	Опис посадових функцій, вимог або зони відповідальності

На основі логічної структури даних, що була сформована в процесі проєктування, побудовано фізичну модель бази даних, яка визначає реалізацію таблиць, полів, зв’язків та обмежень безпосередньо на рівні СУБД. Ця модель відображає конкретну реалізацію структури даних у вигляді реляційної схеми, яка відповідає вимогам до функціональності та цілісності системи. При цьому враховано особливості обраної платформи MS SQL Server, включаючи підтримку зовнішніх ключів, індексацію та типи даних. Фізичне моделювання є

критичним етапом, оскільки забезпечує відповідність між концептуальним дизайном системи та її технічною реалізацією. Фізичну модель представлено на рис. 3.2.

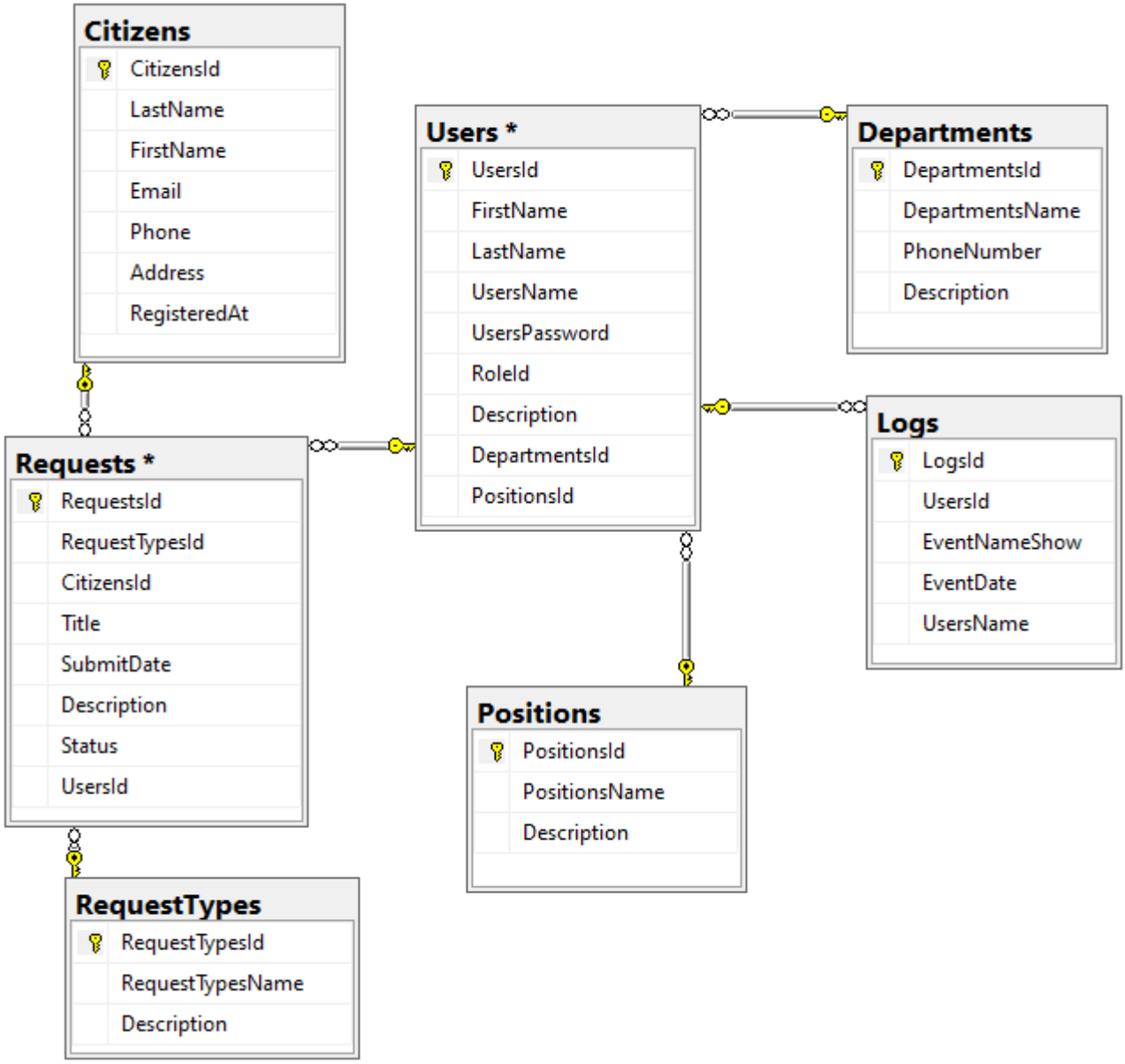


Рис. 3.2 Фізична модель бази даних

Схема демонструє взаємозв’язки між основними таблицями бази даних, включаючи зовнішні ключі, що забезпечують узгодженість при обробці інформації. Така структура дозволяє системі ефективно працювати з даними, підтримуючи стабільність, цілісність і зручність масштабування.

3.3 Архітектура програмного забезпечення

Проєктування архітектури програмного забезпечення є критичним етапом розробки інформаційної системи, оскільки саме архітектурна структура визначає взаємодію між її логічними компонентами, рівень масштабованості, підтримуваність і надійність. У розроблюваній системі автоматизації обліку звернень громадян було прийнято рішення реалізувати трьохрівневу архітектуру, яка передбачає чітке розділення логіки на рівень представлення (Presentation Layer), рівень бізнес-логіки (Business Logic Layer) та рівень доступу до даних (Data Access Layer).

Цей підхід обрано через низку об’єктивних переваг. Він забезпечує високу гнучкість при модифікації окремих частин застосунку – інтерфейс користувача може змінюватися незалежно від логіки обробки або структури бази. Окремі частини системи тестуються ізольовано, що значно зменшує витрати часу на усунення помилок. Така структура сприяє покращенню масштабованості та дозволяє легше адаптувати систему до нових бізнес-вимог. Крім того, архітектурне розділення дозволяє паралельно працювати над різними модулями в команді розробників, що підвищує продуктивність і керованість проєкту. У довгостроковій перспективі це дає змогу реалізувати додаткові сервіси, не порушуючи цілісності вже реалізованих модулів.

Таким чином, трьохрівнева архітектура дозволяє побудувати структуровану, підтримувану та масштабовану систему, яка легко адаптується до змін вимог, відповідає принципам модульності та сприяє підвищенню якості розробки загалом.

Для реалізації доступу до бази даних у розроблюваній системі використано окремий рівень даних, який відповідає за взаємодію між прикладною логікою та фізичним зберіганням інформації. Цей рівень інкапсулює запити до СУБД, забезпечує цілісність і контроль за операціями над даними, а також мінімізує дублювання коду при реалізації повторюваних операцій, таких як вибірка, фільтрація, додавання чи оновлення записів. Такий підхід дозволяє ізолювати деталі реалізації роботи з базою даних, що сприяє легшій підтримці та

модифікації системи в майбутньому. Крім того, централізована обробка запитів підвищує стабільність функціонування застосунку та дозволяє виявляти й локалізувати помилки на ранньому етапі. Його структура подана на рис. 3.3.

Рис. 3.3 Діаграма класів рівня даних

Діаграма класів рівня даних включає наступні компоненти:

- CitizensProvider – клас, який реалізує всі операції доступу до інформації про громадян. Містить методи для отримання, додавання, оновлення та видалення записів, що стосуються осіб, які подають звернення. Здійснює взаємодію з таблицею Citizens у базі даних;

- RequestsProvider – відповідає за доступ до таблиці Requests, яка містить дані про подані звернення. Клас забезпечує механізми пошуку звернень за типом,

статусом, користувачем або датою подачі, а також містить методи для створення, редагування й перегляду історії звернень;

- UsersProvider – клас для роботи з даними про користувачів системи. Дозволяє здійснювати автентифікацію, отримувати списки працівників за фільтрами, а також управляти даними про ролі, посади та належність до відділів. Працює з таблицею Users;

- DepartmentsProvider – реалізує запити до довідника відділів. Дозволяє отримувати дані про структурні підрозділи установи, які відповідають за опрацювання звернень, а також підтримує операції додавання, редагування та видалення;

- PositionsProvider – клас, що забезпечує взаємодію з таблицею Positions. Його методи дозволяють керувати переліком посад у системі, а також прив’язувати ці посади до конкретних користувачів при створенні облікових записів;

- RequestTypesProvider – призначений для роботи з довідником типів звернень. Містить функціонал для створення нових категорій звернень, редагування наявних записів і побудови логіки фільтрації за типом при аналітичній обробці даних;

- LogsProvider – реалізує доступ до журналу подій. Дозволяє отримати повний список змін, що були виконані користувачами, відфільтрувати записи за типом подій, датою чи автором. Забезпечує підтримку аудиту змін та системної прозорості.

Перелічені класи формують незалежний шар доступу до даних, що забезпечує ізоляцію бізнес-логіки від фізичної структури бази даних, полегшує обслуговування системи та підвищує її гнучкість.

З метою реалізації зручної та функціонально насиченої взаємодії з користувачем, було сформовано рівень представлення (Presentation Layer), який відповідає за візуалізацію даних і обробку подій інтерфейсу. Даний рівень є

ключовим у тришаровій архітектурі застосунку, оскільки забезпечує доступ до бізнес-логіки системи через форми, таблиці, звіти та керуючі елементи. Структура відповідних класів візуалізована на рис. 3.4.

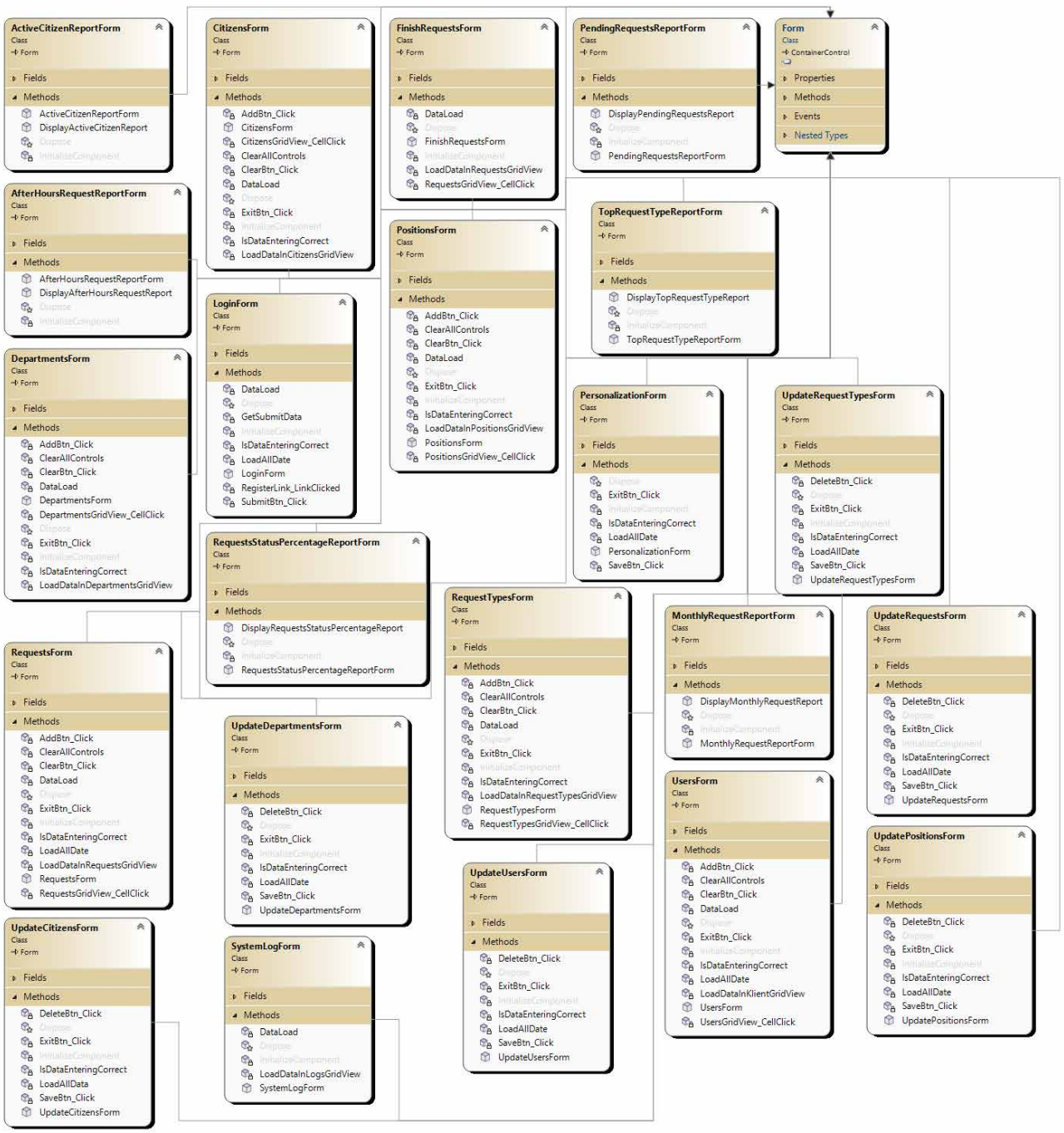


Рис. 3.4 Діаграма класів рівня користувацького інтерфейсу

Діаграма класів рівня користувацького інтерфейсу включає такі основні компоненти:

- CitizensForm – реалізує інтерфейс для перегляду, додавання, редагування та видалення інформації про громадян. Містить обробники подій,

пов’язаних із взаємодією з таблицею, формою введення та перевіркою введених даних.

- UserForm – забезпечує управління обліковими записами користувачів: перегляд, фільтрація, створення та редагування користувачів із валідацією форм. Підключається до довідників посад і відділів.

- DepartmentsForm, PositionsForm – реалізують керування довідниками відділів і посад. Обробники подій дозволяють додавати, редагувати й очищати поля, а також оновлювати пов’язані таблиці.

- RequestForm – основна форма для перегляду та управління зверненнями. Підтримує створення, редагування, перевірку правильності введення, а також відображення звернень у табличному вигляді.

- RequestTypesForm – форма для адміністрування типів звернень. Містить обробку подій для створення, редагування та завантаження відповідного довідника у таблицю.

- LogsForm – на рисунку не виділена окремо, але журнал подій, як правило, переглядається через відповідну форму, яка дозволяє відстежувати всі активності користувачів.

- LoginForm – реалізує процес автентифікації користувача у системі. Містить перевірку логіну й паролю, а також маршрутизацію користувача до основного меню.

- SystemLogForm, PersonalizationForm – допоміжні форми для перегляду подій та зміни особистих налаштувань користувача відповідно.

- UpdateCitizenForm, UpdateUserForm, UpdateDepartmentForm, UpdatePositionForm, UpdateRequestForm, UpdateRequestTypeForm – окремі форми редагування, які реалізують механізми оновлення інформації для відповідних об’єктів. Усі вони містять валідацію введення, підтвердження змін і обробку взаємодії з відповідними таблицями.

- ActiveCitizenReportForm, RequestsStatusPercentageReportForm, AfterHoursRequestReportForm, FinishRequestForm, MonthlyRequestReportForm, TopRequestTypeReportForm, PendingRequestReportForm – форми генерації статистичних і аналітичних звітів, які забезпечують відображення структурованої інформації за статусами, типами, періодами чи часовими рамками.

Загалом, рівень користувацького інтерфейсу в системі побудовано відповідно до принципів розділення відповідальності: кожна форма виконує чітко визначену функцію, забезпечуючи користувачам зручну навігацію, наочність та функціональну завершеність усіх адміністративних і аналітичних процедур. Такий підхід сприяє масштабованості й підтримуваності системи у разі розширення її можливостей.

Для повноцінної реалізації прикладної логіки функціонування системи було сформовано бізнес-рівень (Business Logic Layer), який відповідає за обробку даних, реалізацію звітів, перевірку введення, шифрування, а також інші функціональні аспекти, що виходять за межі відображення чи зберігання інформації. Саме цей рівень виступає посередником між користувацьким інтерфейсом і доступом до бази даних, забезпечуючи реалізацію основних алгоритмів. Структуру відповідних класів наведено на рис. 3.5.

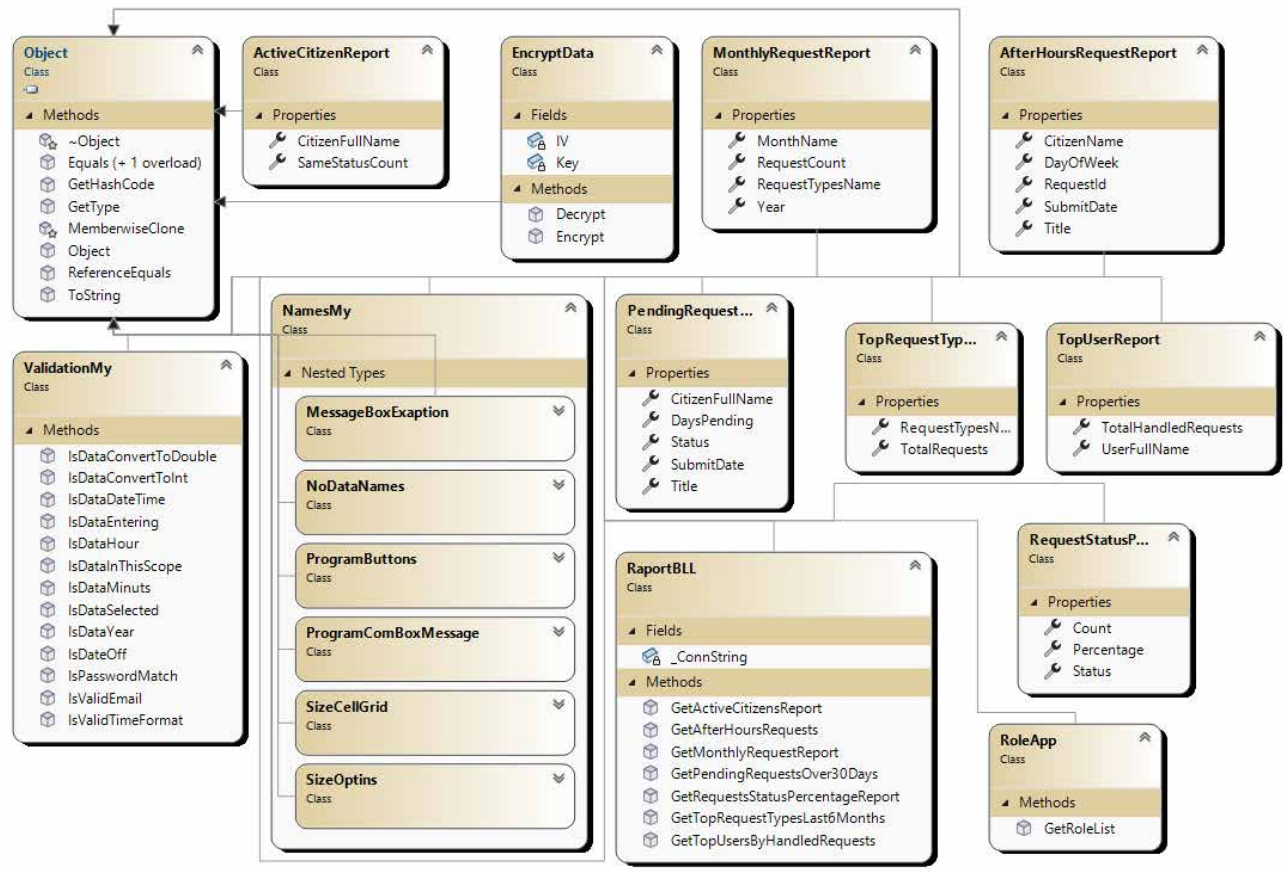


Рис. 3.5 Діаграма класів рівня бізнес-логіки

Діаграма класів рівня бізнес-логіки включає такі ключові компоненти:

- RaportBLL – центральний клас формування звітності, що містить методи для отримання аналітичних даних. Його методи включають генерацію звітів за активними громадянами, зверненнями у позаробочий час, зверненнями, що очікують понад 30 днів, щомісячної звітності, а також рейтингів типів звернень і користувачів. Має приватне поле _ConnString для взаємодії з базою;

- RequestStatusPercentageReport, TopUserReport, TopRequestTypeReport, PendingRequestReport, MonthlyRequestReport, AfterHoursRequestReport, ActiveCitizenReport – класи, які виступають моделями для відповідних звітів. Вони містять властивості, що описують об'єкти звітності, такі як кількість звернень, імена громадян, дати, типи звернень, статуси тощо;

- EncryptData – допоміжний клас, що забезпечує функціональність шифрування та дешифрування. Має поля IV і Key, а також методи Encrypt та Decrypt, що використовуються для захисту конфіденційної інформації;

- ValidationMy – клас, який об'єднує методи перевірки коректності введених даних. Реалізує перевірки типу дати, часу, email, формату часу, пароля тощо. Є універсальним механізмом для забезпечення достовірності інформації на рівні бізнес-логіки перед передачею до бази;

- NamesMy – контейнер допоміжних констант і підкласів, серед яких MessageBoxExaption, NoDataNames, ProgramButtons, ProgramComBoxMessage, SizeCellGrid та SizeOptins. Ці підкласи містять фіксовані текстові повідомлення, ідентифікатори, шаблони кнопок та параметри форматування, що використовуються в системі для узгодженості та зручності;

- RoleApp – клас, що містить метод GetRoleList, який забезпечує отримання переліку доступних ролей користувача. Його призначення – підтримка авторизації та динамічного формування інтерфейсу відповідно до ролей.

Рівень бізнес-логіки відіграє вирішальну роль у забезпеченні функціональності системи, виконуючи обробку даних, перевірку введення, шифрування та генерацію звітів. Він ізольований від інтерфейсу користувача та рівня даних, що забезпечує модульність, спрощує тестування і дозволяє масштабувати логіку без зміни інших частин системи. Така структура сприяє підвищенню надійності та гнучкості розробленого рішення.

3.4 Організаційна структура програмного забезпечення

Організаційна структура програмного забезпечення визначає логічне групування компонентів системи за функціональним призначенням та характером взаємодії між ними. Такий підхід дозволяє досягти високого рівня модульності, спрощує супровід, тестування та масштабування системи. Розподіл класів за окремими функціональними підсистемами (пакетами) є основою для впорядкованої архітектури, що чітко відображає принципи трьохрівневої моделі.

На рис. 3.6 наведено діаграму пакетів, яка ілюструє логічну структуру програмного продукту «Бюрократ.NET» у вигляді взаємозв’язаних підсистем.

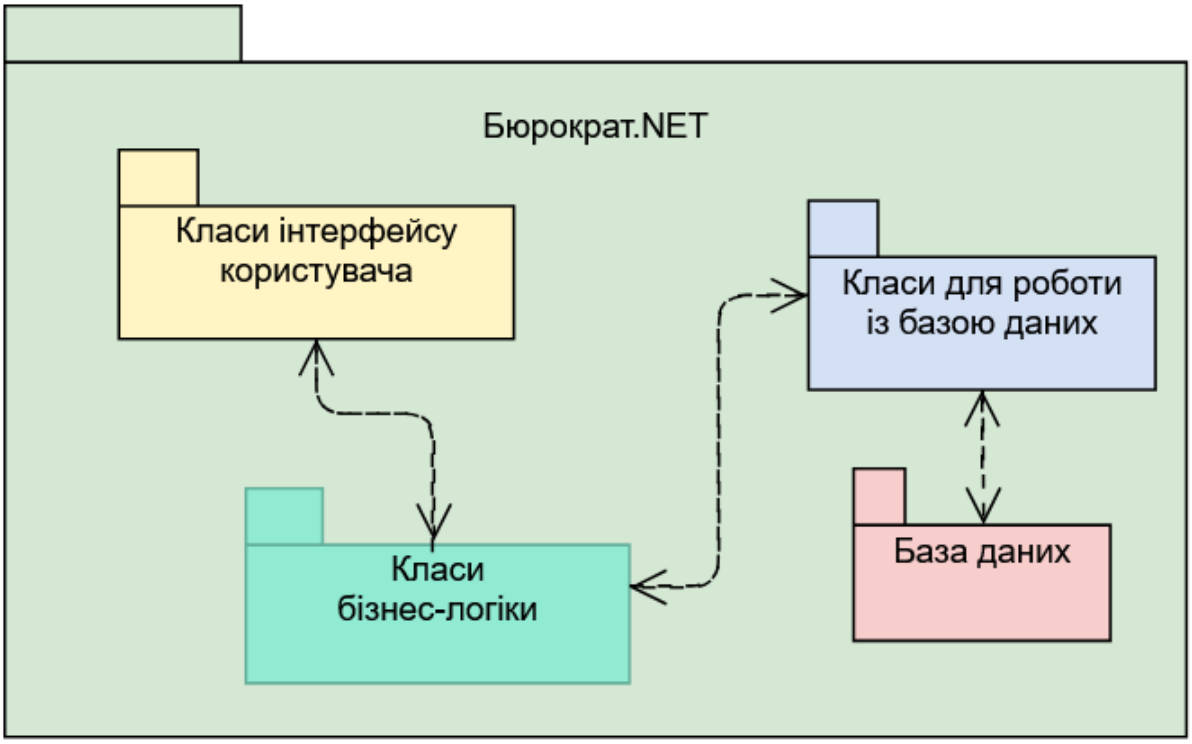


Рис. 3.6 Діаграма пакетів системи

Діаграма пакетів відображає структуру взаємодії між основними складовими програмного забезпечення та їх логічне групування за функціональними ознаками. Вона демонструє, як реалізовано розділення обов’язків між підсистемами, що відповідають за інтерфейс, логіку обробки даних, комунікацію з базою та саму базу даних. У системі виокремлено чотири ключові пакети: модулі інтерфейсу, бізнес-логіки, доступу до даних і сховище даних.

До складу інтерфейсного рівня входять форми та візуальні компоненти, які забезпечують введення даних користувачем, їх перевірку та виведення результатів. Цей рівень є єдиною точкою взаємодії кінцевого користувача із застосунком. Наступний компонент – це пакет бізнес-логіки, який втілює всі прикладні правила, пов’язані з обробкою звернень, фільтрацією, шифруванням, формуванням звітності та інтерпретацією результатів. Саме сюди надходять дані з інтерфейсу, обробляються відповідно до логіки функціонування системи, після чого передаються далі або повертаються для візуалізації.

Проміжною ланкою між логікою та збереженням є набір класів доступу до бази даних. Вони реалізують запити на отримання, додавання, редагування та

видалення інформації у базі, забезпечуючи при цьому ізоляцію логіки від конкретної реалізації зберігання. Завершує взаємодію реляційна база даних, яка містить усі сутності, пов’язані з громадянами, користувачами, зверненнями, подіями та довідниками.

Взаємодія між компонентами чітко регламентована: користувацький інтерфейс звертається до бізнес-логіки, яка, в свою чергу, надсилає запити до рівня даних; відповідь проходить зворотним шляхом. Такий підхід відповідає сучасним вимогам до масштабованості, підтримуваності та зрозумілості архітектури. Діаграма на рисунку 3.6 чітко ілюструє цю багаторівневу організацію та логіку інформаційних потоків між підсистемами.

Висновки до 3 розділу

У даному розділі виконано повний цикл проєктування програмного забезпечення, що включає логічне і фізичне моделювання даних, вибір технологій реалізації, побудову архітектури та перевірку працездатності системи в умовах тестової експлуатації. Побудована логічна модель відобразила структуру предметної області у вигляді таблиць і зв’язків, що охоплюють усі сутності, необхідні для автоматизації обліку звернень громадян. На її основі реалізовано фізичну модель бази даних у середовищі MS SQL Server, що забезпечує збереження, доступність і цілісність інформації відповідно до вимог системи.

Порівняльний аналіз СУБД дозволив обґрунтовано обрати MS SQL Server як основну платформу, зважаючи на її високу інтеграцію з C#, підтримку транзакційності, зручні засоби адміністрування та масштабованість. У рамках архітектурного проєктування розроблено трьохрівневу структуру системи, що включає рівні представлення, бізнес-логіки та доступу до даних. Для кожного з рівнів побудовано діаграми класів, які відображають логіку взаємодії між компонентами та забезпечують розділення відповідальностей. Додатково розроблено діаграму пакетів, яка ілюструє організаційну структуру проєкту з урахуванням логічної класифікації функціональних модулів.

4 РОЗРОБКА ТА ТЕСТУВАННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ

4.1 Вибір інструментарію для створення програмного забезпечення

Одним із важливих етапів розробки інформаційної системи є обґрунтування вибору технологічного інструментарію, зокрема мови програмування та середовища розробки. Вибір мови впливає на продуктивність, зручність підтримки, можливість інтеграції з іншими компонентами та швидкість реалізації необхідного функціоналу. У процесі аналізу було розглянуто кілька поширених мов програмування, зокрема Python, Java та C#, кожна з яких має свої особливості та переваги.

Python – це мова високого рівня, що відзначається простим синтаксисом, швидким освоєнням та великою кількістю бібліотек для наукових обчислень, аналізу даних і автоматизації [22]. Вона ідеально підходить для прототипування, розробки скриптів і побудови систем із гнучкою логікою, однак у десктопних додатках часто потребує додаткових обгорток. Python не є оптимальним вибором для розробки складних графічних застосунків у Windows-середовищі без додаткових фреймворків.

Java – мова, що вирізняється високою портативністю, об’єктно-орієнтованою природою та широкою підтримкою корпоративних застосунків. Вона активно використовується у веброзробці, мобільних застосунках та складних інформаційних системах [23]. Проте створення графічних інтерфейсів у Java може бути менш зручним порівняно з іншими мовами в середовищі Windows.

C# – це сучасна об’єктно-орієнтована мова програмування, розроблена Microsoft, яка забезпечує тісну інтеграцію з платформою .NET [24]. Вона надає широкі можливості для створення десктопних застосунків з графічним інтерфейсом на базі Windows Forms або WPF, підтримує роботу з базами даних, шифрування, обробку подій і звітність. Завдяки високій продуктивності, зручному синтаксису та потужному інструментарію C# було обрано як основну мову реалізації системи автоматизації бюрократичних процесів.

З метою обґрунтування вибору оптимальної мови програмування для реалізації системи автоматизації бюрократичних процедур доцільно провести порівняння за ключовими характеристиками, що визначають зручність розробки, ефективність виконання, підтримку інструментів для побудови графічного інтерфейсу та можливості роботи з базами даних. У табл. 4.1 наведено порівняльний аналіз трьох поширених мов програмування.

Таблиця 4.1 – Порівняльна характеристика мов програмування для створення десктопного ПЗ

№	Характеристика	Python	Java	C#
1	Час компіляції та виконання	Повільне виконання	Середнє	Висока швидкість виконання
2	Підтримка графічного інтерфейсу (WinForms/WPF)	Обмежена	Помірна (Swing, JavaFX)	Повна підтримка (WinForms, WPF)
4	Підтримка ADO.NET / SQL Server	Через сторонні бібліотеки	Можлива через JDBC	Пряма підтримка
5	Порог входу для розробки GUI-додатків	Низький, але нестабільний	Вищий	Помірний і структурований
6	Середовище розробки (IDE)	PyCharm, VS Code	IntelliJ, Eclipse	Visual Studio
7	Доступність бібліотек для шифрування/валідації	Висока	Висока	Висока
8	Активність спільноти (GitHub, StackOverflow)	Висока	Висока	Висока
9	Підтримка типізації	Динамічна	Статична	Статична
10	Обсяг займаної пам’яті	Низький	Помірний	Помірний

Вибір мови програмування C# для розробки системи зумовлений її повною сумісністю з операційною системою Windows та розвиненою підтримкою інструментів створення графічного інтерфейсу, зокрема технологій WinForms та WPF. Завдяки статичній типізації та високій швидкості виконання C# забезпечує стабільну роботу застосунку навіть при обробці значних обсягів даних.

Вбудована підтримка інтеграції з базами даних через ADO.NET, а також зручність використання середовища Visual Studio значно пришвидшують процес розробки. Крім того, чітка модульна структура й багатий набір бібліотек дозволяють ефективно реалізовувати архітектуру з розділенням на рівні інтерфейсу, логіки та доступу до даних. Сукупність цих факторів робить C# доцільним вибором для реалізації надійної інформаційної системи автоматизації бюрократичних процедур.

4.2 Вимоги до апаратного та програмного забезпечення

Для забезпечення стабільної роботи розробленої інформаційної системи автоматизації бюрократичних процедур необхідно чітко визначити вимоги до технічної інфраструктури. Ці вимоги охоплюють апаратну конфігурацію персонального комп'ютера користувача, на якому буде функціонувати застосунок, а також параметри серверного середовища для розміщення бази даних. Окрім цього, важливим є узгодження програмних компонентів, зокрема операційної системи, системи керування базами даних та платформи виконання застосунку.

На рис. 4.1 зображено діаграму розгортання системи, яка демонструє організацію компонентів у типовому середовищі експлуатації.

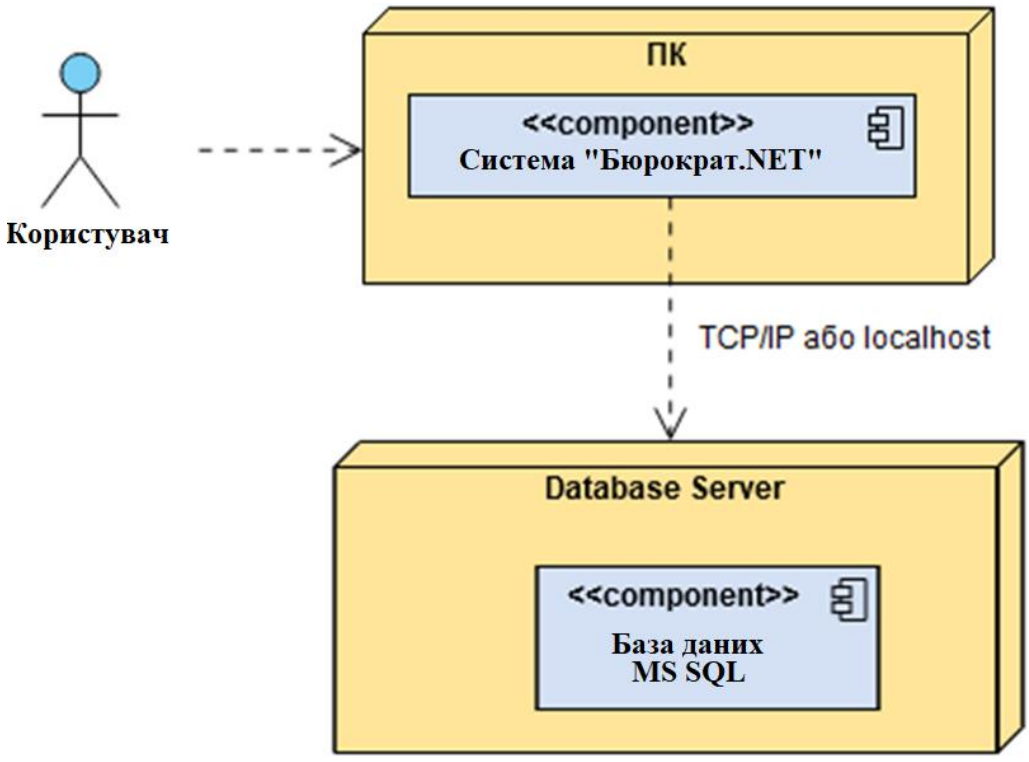


Рис. 4.1 Діаграма розгортання системи

Діаграма ілюструє, що користувач взаємодіє із застосунком через персональний комп'ютер, на якому встановлено клієнтський компонент – систему «Бюрократ.NET». Цей застосунок, у свою чергу, встановлює з'єднання з віддаленим або локальним сервером бази даних через протокол TCP/IP або канал localhost. Серверна частина містить компонент MS SQL Server, який обробляє запити до бази, зберігає, оновлює та надає доступ до даних, необхідних для функціонування системи. Така структура дозволяє реалізувати як однокористувацький варіант із локальною базою, так і мережеву конфігурацію з централізованим зберіганням інформації.

Для коректного функціонування програмного забезпечення, що реалізує автоматизовану систему підтримки бюрократичних процедур, необхідно дотримуватись певних технічних умов. Конфігурація робочої станції має забезпечувати достатню обчислювальну потужність для стабільної роботи графічного інтерфейсу, обробки запитів та взаємодії з сервером бази даних. Нижче наведено мінімальні та рекомендовані вимоги до апаратного і програмного забезпечення, що дозволяють розгорнути та експлуатувати систему в сучасному середовищі.

Мінімальні технічні вимоги:

- операційна система: Windows 10 (64-bit);
- програмна платформа: .NET Framework 4.8 або .NET 6 (для сумісності з новими середовищами розробки);
- оперативна пам'ять: не менше 4 ГБ;
- процесор: Intel Core i3 або аналогічний;
- мережеве підключення: доступ до локального серверу або TCP/IP-з'єднання для віддаленої бази даних;
- вільне місце на диску: від 200 МБ для встановлення клієнтської частини, не менше 1 ГБ для зберігання журналів та супутніх файлів.

Рекомендовані технічні вимоги:

- операційна система: Windows 11 Professional;
- оперативна пам’ять: від 8 ГБ;
- процесор: Intel Core i5 або вище;
- жорсткий диск SSD: з вільним простором від 2 ГБ для швидкодії;
- програмне забезпечення: встановлений MS SQL Server 2019 або новіший (при локальній конфігурації).

Виконання зазначених вимог дозволить забезпечити надійну та безперебійну роботу системи, гарантуючи комфортну взаємодію з інтерфейсом та швидкий доступ до всіх функціональних можливостей застосунку.

4.3 Тестування системи

Після завершення етапу розробки інформаційної системи для автоматизації бюрократичних процедур було проведено тестування функціональних компонентів із метою верифікації їхньої працездатності відповідно до заявлених вимог. Особливу увагу приділено елементам авторизації, обробки звернень, валідації введених даних та генерації звітів. Перевірка охоплює не лише коректність виконання окремих операцій, а й відповідність поведінки системи очікуваному сценарію у типових та граничних випадках.

На рис. 4.2 зображено форму авторизації, яка є першим етапом взаємодії користувача із системою.

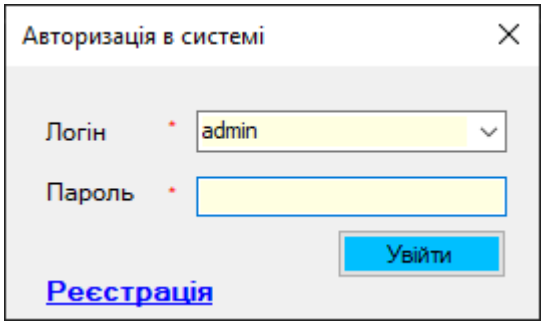


Рис. 4.2 Форма авторизації користувача

Ця форма забезпечує вхід зареєстрованого користувача до системи шляхом введення логіна та пароля. У ході тестування перевірено, що при введенні некоректних облікових даних система надає відповідне повідомлення про помилку, не допускаючи несанкціонований доступ. У випадку правильного введення облікових реквізитів виконується успішна автентифікація та перенаправлення користувача до головного інтерфейсу застосунку. Також проведено перевірку коректності роботи гіперпосилання на форму реєстрації нового користувача, яка відкривається у разі натискання на відповідний текстовий елемент. Під час тестування встановлено, що форма стабільно працює, поля з обов'язковим заповненням позначені спеціальним маркером, а кнопка підтвердження дії (Увійти) є неактивною при порожніх полях, що відповідає заявленим функціональним вимогам до валідації введення.

На рис. 4.3 наведено інтерфейс форми, що дозволяє вводити назву, номер телефону та опис відділу, а також переглядати вже наявні записи в таблиці. У процесі тестування було перевірено правильність додавання нових записів та їх автоматичне відображення у списку без потреби перезапуску форми. Встановлено, що при відсутності обов'язкових даних система не дозволяє виконати дію додавання, сигналізуючи про помилку. Кнопки очищення та виходу також спрацювали згідно із заданою логікою, забезпечуючи зручність користування. Робота форми відповідає функціональним вимогам та забезпечує цілісність введених даних.

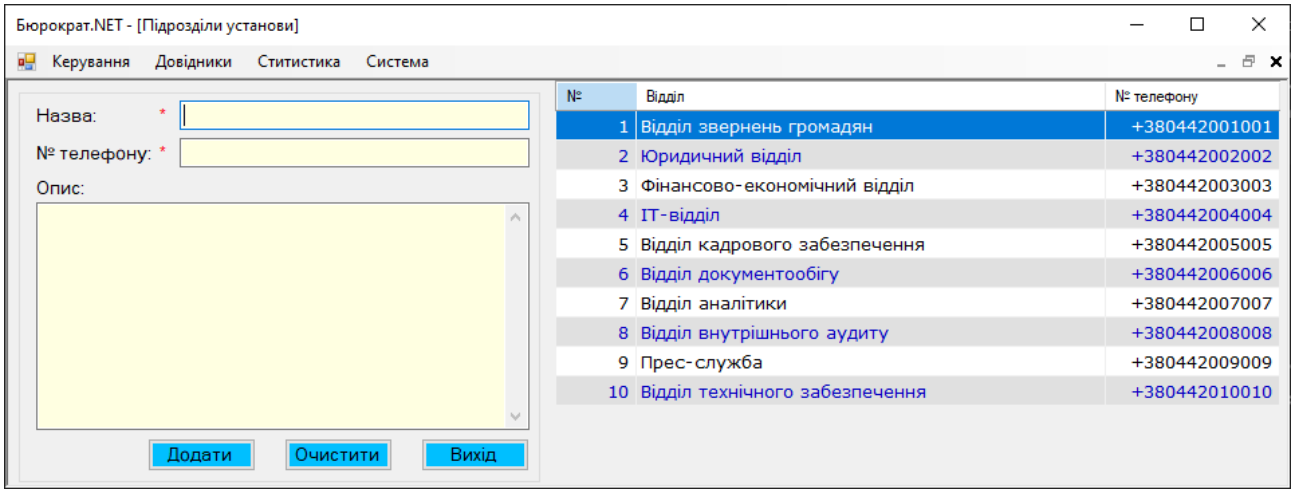


Рис. 4.3 Форма для керування підрозділами установи

Рис. 4.4 відображає інтерфейс, який дає змогу створювати нові записи посад із вказанням назви та опису, а також переглядати повний перелік уже збережених позицій. У процесі перевірки було встановлено, що при додаванні нової посади з порожнім полем назви система коректно виконує валідацію, не допускаючи створення неповного запису. Усі додані дані моментально відображаються у таблиці, що підтверджує правильну реалізацію механізму оновлення вмісту. Зовнішній вигляд та функціональність форми забезпечують зрозумілий та ефективний процес ведення кадрової структури установи.

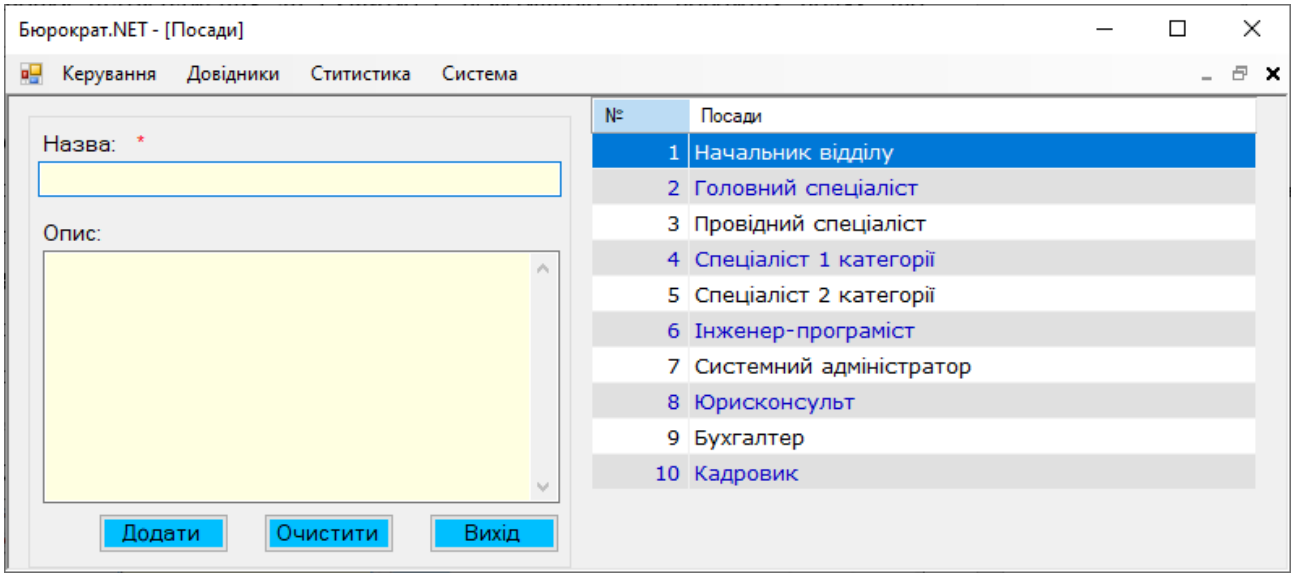


Рис. 4.4 Форма для керування посадами

На рис. 4.5 зображено вікно, в якому користувач має змогу вводити назву та опис категорії звернення, після чого додавати її до загального переліку. Під час тестування було зафіксовано, що система коректно блокує можливість збереження порожніх назв, що забезпечує чистоту довідника. Інтерфейс підтримує інтуїтивне керування: кнопки "Додати", "Очистити" та "Вихід" реагують миттєво, а зміни відображаються у таблиці без затримки. Завдяки цьому форма є зручною для швидкого оновлення типології звернень, що сприяє адаптивності системи до нових адміністративних потреб.

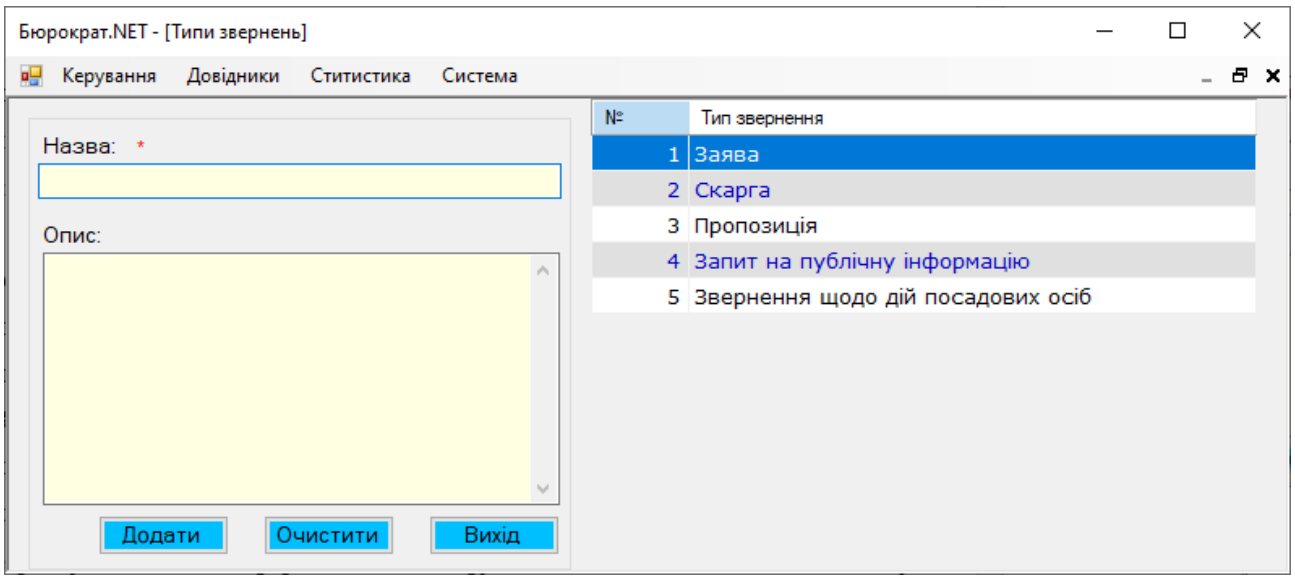


Рис. 4.5 Форма для керування типами звернень

Рис. 4.6 відображає форму, що дозволяє вносити персональні дані осіб: прізвище, ім’я, номер телефону, електронну пошту, дату реєстрації та адресу доставки. У межах тестування перевірено коректність валідації обов’язкових полів, зокрема email і номера телефону. Система успішно обробляє як додавання нових записів, так і відображення актуального списку громадян у таблиці, де результати сортуються у реальному часі. Завдяки зрозумілому інтерфейсу та оперативній реакції на дії користувача ця форма демонструє стабільну поведінку й відповідає заявленим функціональним вимогам.

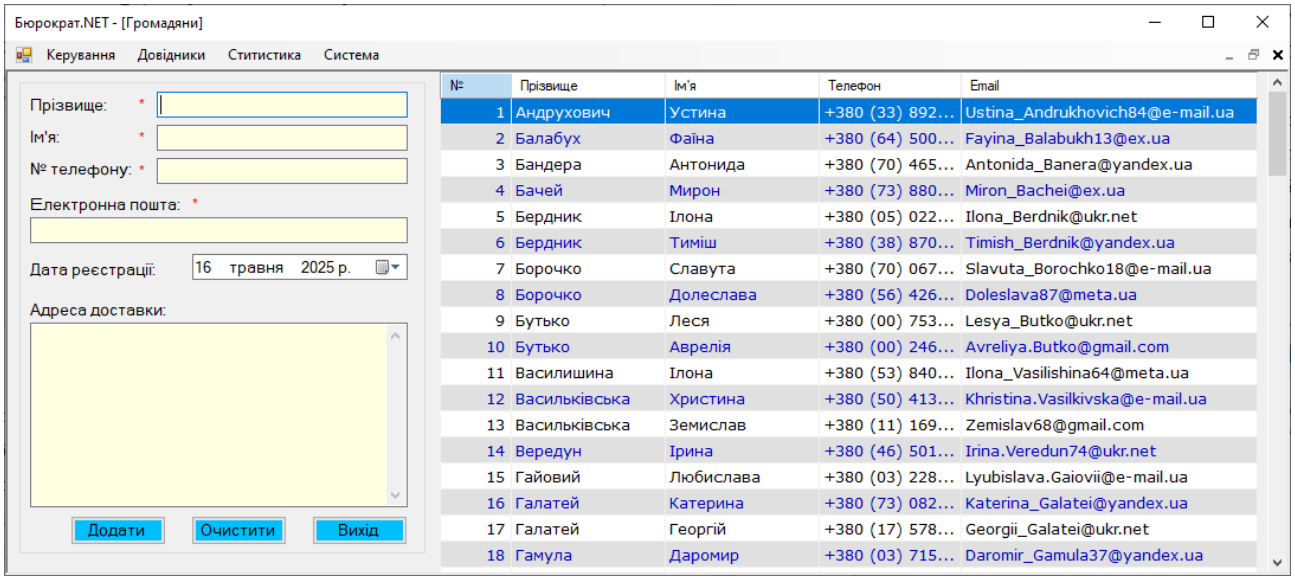


Рис. 4.6 Форма для керування даними громадян

На рис. 4.7 висвітлено інтерфейс, який дозволяє адміністратору обрати тип звернення, вибрати громадянина зі списку, ввести тему, дату подання, статус і

детальний опис ситуації. Під час тестування перевірено взаємозв'язок між полями: при зміні типу звернення або особи автоматично підлаштовується контекст взаємодії.

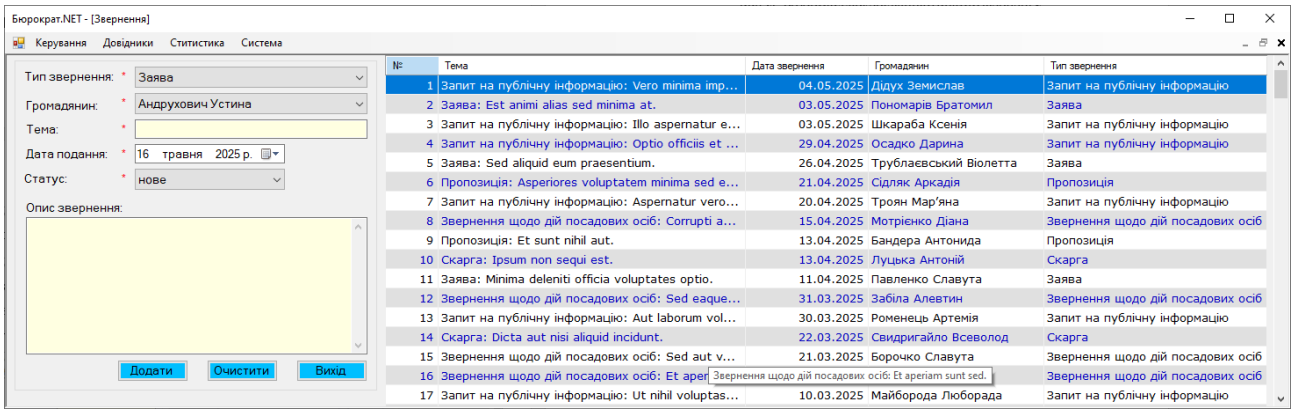


Рис. 4.7 Форма для фіксації звернень громадян

Також перевірено збереження запису з повним набором обов'язкових даних – система без затримок виводила новий запис до таблиці справа. Додатково було протестовано механізм валідації: при спробі зберегти порожню тему або залишити незаповненими критичні поля система реагувала відповідним повідомленням, запобігаючи внесенню некоректної інформації.

На рис. 4.8 наведено діалогове вікно, у якому користувач має змогу змінити тип звернення, громадянина, тему, дату подання, поточний статус розгляду та зміст повідомлення.

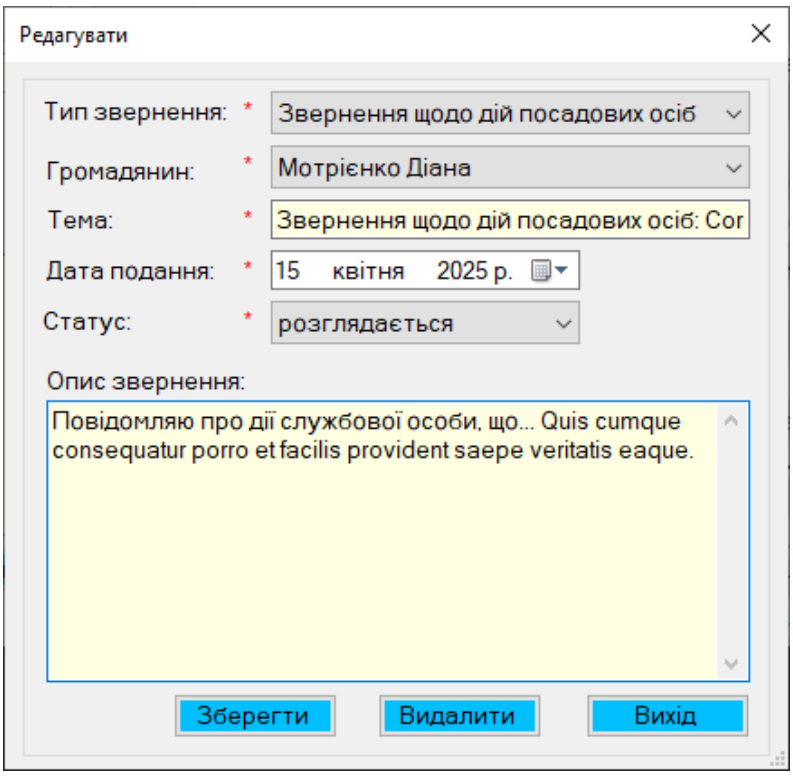
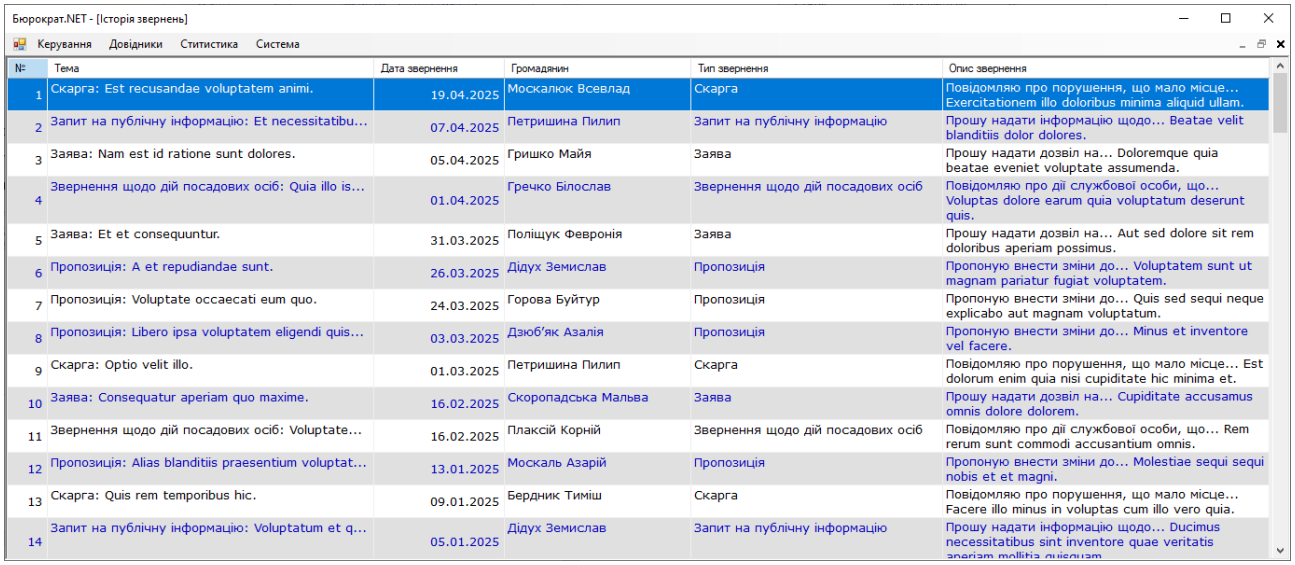


Рис. 4.8 Форма для редагування звернень

У межах функціонального тестування була перевірена здатність форми коректно завантажувати вибрані дані, а також зберігати оновлену інформацію після редагування. Особливу увагу приділено зміні статусу звернення: при переході від початкового стану «нове» до «розглядається» оновлення відбувалося без збоїв, з подальшим відображенням у таблиці загального переліку. Також перевірено дію кнопки «Видалити» – запис видалявся з таблиці після підтвердження, а система забезпечувала оновлення інтерфейсу без необхідності перезапуску форми. Загалом, механізм редагування виявився зручним у використанні та надійним з точки зору контролю даних.

Рис. 4.9 відображає інтерфейс, у якому реалізовано зручну табличну структуру з можливістю перегляду ключових атрибутів кожного звернення: теми, дати подання, прізвища громадянина, типу звернення та змісту.



№	Тема	Дата звернення	Громадянин	Тип звернення	Опис звернення
1	Скарга: Est recusandae voluptatem animi.	19.04.2025	Москалюк Всевлад	Скарга	Повідомляю про порушення, що мало місце... Exercitationem illo doloribus minima aliquid ullam.
2	Запит на публічну інформацію: Et necessitatibu...	07.04.2025	Петришина Пилип	Запит на публічну інформацію	Прошу надати інформацію щодо... Beatae velit blanditiis dolor dolores.
3	Заява: Nam est id ratione sunt dolores.	05.04.2025	Гришко Майя	Заява	Прошу надати дозвіл на... Doloremque quia beatae eveniet voluptate assumenda.
4	Звернення щодо дій посадових осіб: Quia illo is...	01.04.2025	Гречко Білослав	Звернення щодо дій посадових осіб	Повідомляю про дії службової особи, що... Voluptas dolore earum quia voluptatum deserunt quis.
5	Заява: Et et consequuntur.	31.03.2025	Поліщук Февронія	Заява	Прошу надати дозвіл на... Aut sed dolore sit rem doloribus aperiam possimus.
6	Пропозиція: A et repudiandae sunt.	26.03.2025	Дідух Земислав	Пропозиція	Пропоную внести зміни до... Voluptatem sunt ut magnam pariatur fugiat voluptatem.
7	Пропозиція: Voluptate occaecati eum quo.	24.03.2025	Горова Буйтур	Пропозиція	Пропоную внести зміни до... Quis sed sequi neque explicabo aut magnam voluptatum.
8	Пропозиція: Libero ipsa voluptatem eligendi quis...	03.03.2025	Дзюб'як Азалія	Пропозиція	Пропоную внести зміни до... Minus et inventore vel facere.
9	Скарга: Optio velit illo.	01.03.2025	Петришина Пилип	Скарга	Повідомляю про порушення, що мало місце... Est dolorum enim quia nisi cupiditate hic minima et.
10	Заява: Consequatur aperiam quo maxime.	16.02.2025	Скоропадська Мальва	Заява	Прошу надати дозвіл на... Cupiditate accusamus omnis dolore dolorem.
11	Звернення щодо дій посадових осіб: Voluptate...	16.02.2025	Плаксій Корній	Звернення щодо дій посадових осіб	Повідомляю про дії службової особи, що... Rem rerum sunt commodi accusantium omnis.
12	Пропозиція: Alias blanditiis praesentium voluptat...	13.01.2025	Москаль Азарій	Пропозиція	Пропоную внести зміни до... Molestiae sequi sequi nobis et et magni.
13	Скарга: Quis rem temporibus hic.	09.01.2025	Бердник Тиміш	Скарга	Повідомляю про порушення, що мало місце... Facere illo minus in voluptas cum illo vero quia.
14	Запит на публічну інформацію: Voluptatum et q...	05.01.2025	Дідух Земислав	Запит на публічну інформацію	Прошу надати інформацію щодо... Ducimus necessitatibus sint inventore quae veritatis aperiam mollitia quisquam

Рис. 4.9 Форма для перегляду історії звернень

Тестування показало, що дані відображаються у хронологічному порядку, а подвійне натискання на запис забезпечує швидкий доступ до розширеної інформації або переходу до редагування (за наявності відповідних прав доступу). Особливо корисною функцією виявилася здатність візуально ідентифікувати звернення завдяки колірному маркуванню за типом або статусом – це спростило навігацію у великих масивах записів. Система також стабільно обробляла великі обсяги даних без зниження продуктивності, що підтверджує її готовність до реального навантаження. У цілому, форма продемонструвала ефективну реалізацію функції журналу дій, що є критично важливою для контролю процесів у сфері публічного адміністрування.

Окреме тестування було присвячено модулю статистичної звітності, який забезпечує аналітичний огляд активності громадян у системі за кількістю поданих звернень.

На рис. 4.10 відображено приклад текстового звіту, який демонструє список громадян, що подали понад три звернення. У процесі тестування перевірено коректність алгоритму вибірки та сортування даних – система точно виявляє користувачів із високою активністю та групує їх відповідно до кількості звернень. Вивід інформації організовано у табличному форматі з чіткою розміткою, що дозволяє швидко орієнтуватися в аналітичних показниках.

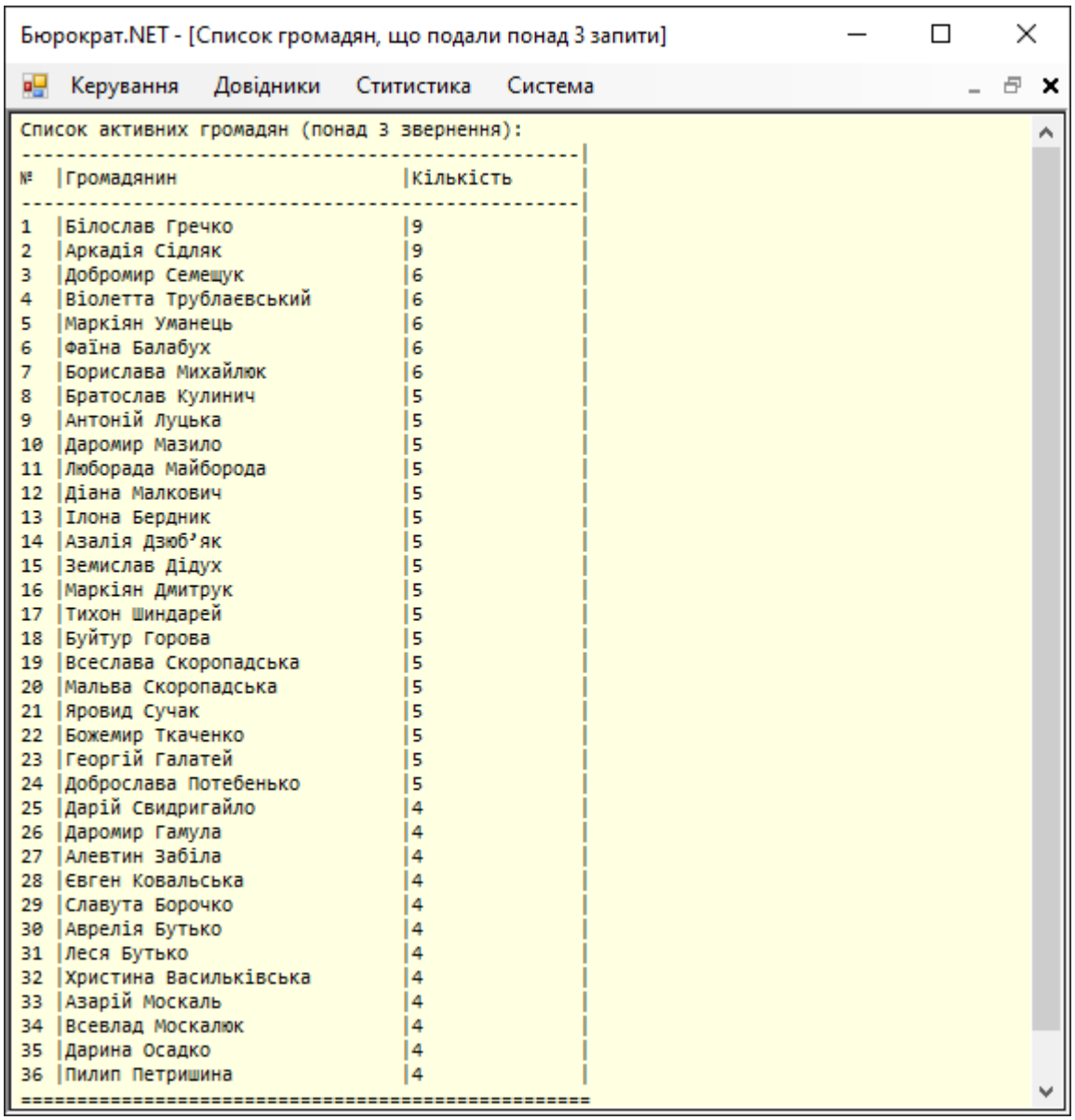


Рис. 4.10 Формування списку громадян, що подали понад 3 запити

При цьому навіть за значного обсягу даних генерація звіту виконується без затримок, що підтверджує належну оптимізацію запитів. Такий функціонал є важливим інструментом для управлінських рішень, зокрема – ідентифікації найбільш залучених громадян або виявлення потенційних системних проблем, які викликають повторні звернення.

Рис. 4.11 відображає результати такого звіту, який формується на основі агрегованих даних та відображає три найчастіші категорії запитів разом із відповідною кількістю.

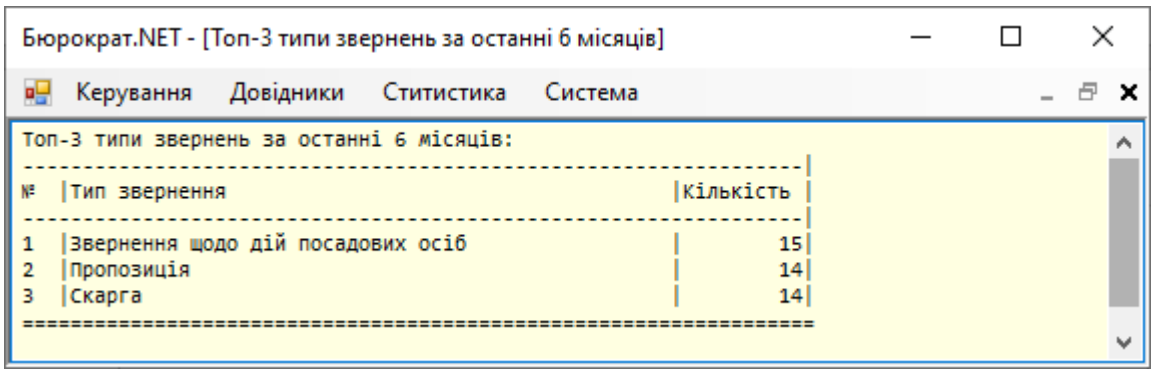


Рис. 4.11 Формування списку топ-3 типи звернень за останні 6 місяців

У ході тестування було підтверджено, що алгоритм коректно враховує часові обмеження та виконує групування по типах із наступним сортуванням за зменшенням кількості. Система формує звіт у компактному табличному вигляді, що дозволяє одразу побачити переважну тематику звернень. Такий функціонал може слугувати інструментом для стратегічного аналізу роботи установи, зокрема – для виявлення повторюваних проблемних аспектів у взаємодії з громадськістю або перегляду пріоритетів обробки запитів.

На рис. 4.12 наведено результат такого звіту, де зафіксовано співвідношення звернень у статусах "завершено", "нове" та "розглядається".

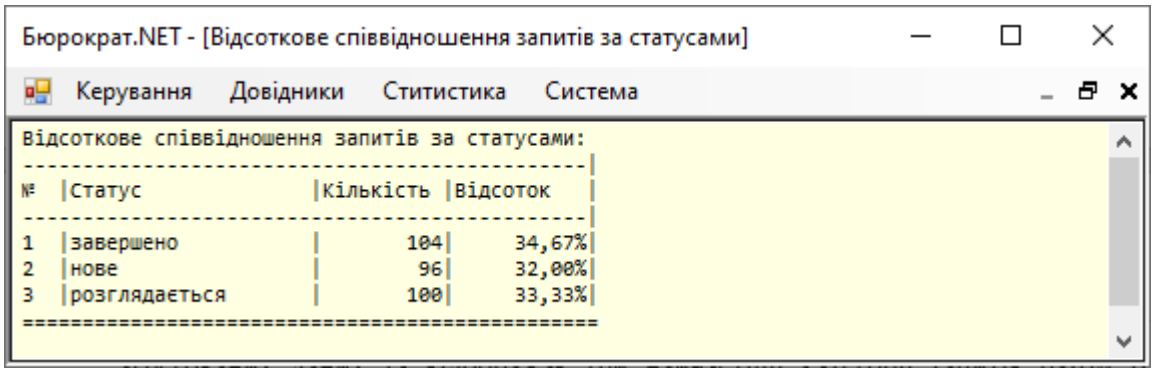


Рис. 4.12 Формування відсоткового співвідношення запитів за статусами

Під час перевірки було підтверджено коректність логіки підрахунку як кількісного значення, так і обчислення відсоткового представлення від загального масиву звернень. Звіт є інформативним інструментом для моніторингу динаміки опрацювання запитів у системі, зокрема виявлення накопичення необроблених заяв або затримок у розгляді. Такий формат подання аналітики особливо зручний для періодичних оглядів діяльності установи та подальшого коригування внутрішніх процедур.

Рис. 4.13 відображає звернення, подані у неробочий час – після 18:00 або у вихідні дні. У ході тестування система успішно ідентифікувала записи з відповідними часовими мітками, підтверджуючи правильну реалізацію фільтрації за критеріями доби й календарного дня.

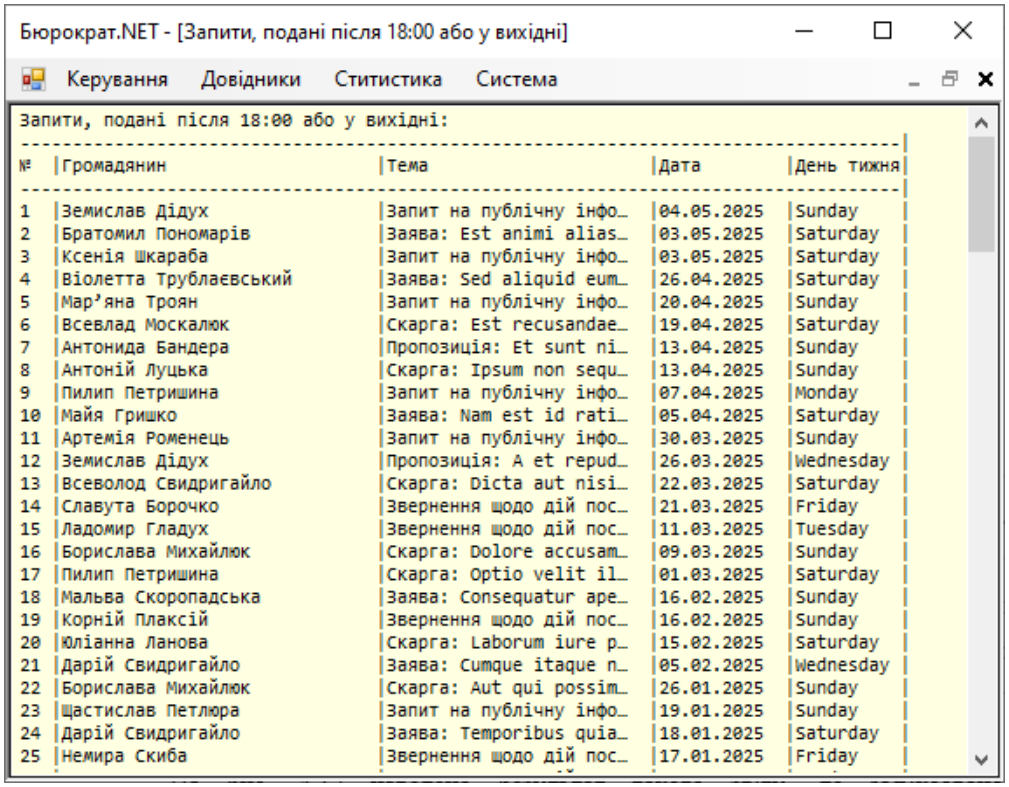


Рис. 4.13 Формування списку запитів, що подані після 18:00 або у вихідні

Загалом, результати тестування підтвердили коректну роботу основних функціональних модулів системи у штатних і нестандартних умовах. Усі протестовані форми, звіти та механізми взаємодії з даними виконували поставлені задачі без критичних помилок.

Висновки до 4 розділу

Проаналізовано особливості використання мов програмування у контексті розробки настільного застосунку. В результаті вибір було зроблено на користь мови C#, яка забезпечує оптимальне поєднання продуктивності, простоти інтеграції з .NET-платформою та підтримки графічного інтерфейсу. У процесі розгортання системи визначено вимоги до апаратного й програмного забезпечення, включно з побудовою діаграми розгортання, яка демонструє взаємодію між користувачем, застосунком і сервером бази даних.

Завершальним етапом стала перевірка працездатності всіх функціональних модулів у межах тестування системи. Описано роботу ключових форм, здійснено перевірку логіки обробки звернень, генерації звітів і реєстрації користувачів. Результати тестування підтвердили відповідність системи заявленим функціональним і нефункціональним вимогам, стабільність її роботи та готовність до практичного застосування в умовах реального адміністративного навантаження.

ВИСНОВКИ

У процесі виконання бакалаврської кваліфікаційної роботи було здійснено повний цикл розробки інформаційної системи підтримки бюрократичних процедур на основі сучасних інформаційних рішень, орієнтованої на автоматизацію опрацювання звернень громадян. В результаті аналізу предметної області було з'ясовано, що бюрократичні процеси є критично важливими для забезпечення прозорої та відповідальної діяльності державних і комунікаційних структур, але водночас залишаються перевантаженими рутинними операціями, затримками та ризиками помилок. Проведено критичний огляд сучасних інформаційних рішень, таких як DocStar ECM, Laserfiche та M-Files, під час якого виявлено низку недоліків, зокрема високу складність впровадження, надмірну функціональність для базових потреб, а також залежність від комерційних ліцензій. Це стало підґрунтям для формулювання завдання розробки власної системи з адаптованим функціоналом, орієнтованої на простоту, доступність і гнучкість у використанні.

В межах моделювання предметної області сформовано UML-діаграми варіантів використання для адміністратора і користувача, побудовано діаграми активності та послідовності, що демонструють ключові процеси системи, такі як додавання та редагування звернень, формування статистичних звітів тощо. Виділено абстракції предметної області, що дозволило чітко структурувати логіку системи. Спроєктовано діаграму класів, яка забезпечує відповідність архітектурного дизайну об'єктно-орієнтованій парадигмі.

Розроблено логічну модель бази даних, яка охоплює всі ключові сутності та зв'язки, після чого реалізовано фізичну модель у середовищі MS SQL Server. Вибір саме цієї СУБД було обґрунтовано її інтеграційною сумісністю з C# та наявністю широкого спектра засобів адміністрування, продуктивної роботи з транзакціями та резервування. Реалізовано трирівневу архітектуру системи з чітким розділенням функцій між рівнем інтерфейсу, бізнес-логіки та доступу до даних. Кожен рівень було представлено діаграмами класів з відповідним описом

функціональних модулів. Для організації структури програмного забезпечення розроблено діаграму пакетів, яка відображає зв’язки між функціональними компонентами системи.

У процесі вибору інструментів реалізації було розглянуто мови програмування Python, Java та C#, з урахуванням продуктивності, гнучкості та відповідності сучасним інженерним вимогам. Вибір C# був підтверджений високою інтеграцією з платформою .NET, підтримкою роботи з формами WinForms та ефективною взаємодією з MS SQL Server. Проведено аналіз вимог до апаратного та програмного забезпечення, включаючи побудову діаграми розгортання, яка ілюструє взаємодію між користувачем, прикладним програмним забезпеченням та сервером бази даних. Розроблено перелік мінімальних і рекомендованих технічних характеристик для запуску системи.

У результаті тестування було перевірено коректність роботи всіх модулів системи, включаючи форми додавання, редагування, перегляду звернень, а також генерацію статистичних звітів. Система успішно пройшла тестування на стабільність, обробку помилок і відповідність функціональним вимогам. Вона забезпечує зручну взаємодію з користувачем, підтримку ряду аналітичних функцій і демонструє високу продуктивність у межах заданого функціоналу.

Запропоноване рішення має низку переваг у порівнянні з відомими аналогами: безкоштовність, простота розгортання, мінімальні апаратні вимоги, а також можливість адаптації під потреби конкретної установи. Технічною новацією є інтеграція механізмів статистичного аналізу звернень без використання складних зовнішніх модулів, що реалізовано засобами C# у середовищі WinForms. Система має потенціал для використання у державних органах, громадських приймальнях, комунальних підприємствах та закладах освіти. Подальше вдосконалення може включати розширення функціональності, реалізацію веб-інтерфейсу, додавання модулів аналітики на основі ML та розширення ролей користувачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Farmer D. Beyond Public Administration: Contemplating and Nudging Government-in-Context. – Tuscaloosa: University of Alabama Press, 2020. 240 p.

2. Gronwald K. Integrated Business Information Systems: A Holistic View of the Linked Business Process Chain ERP-SCM-CRM-BI-Big Data. – Wiesbaden: Springer Vieweg, 2020. 297 p.

3. Goldfinch, S. The challenge of public administration reform: introduction to the Handbook of Public Administration Reform. In Handbook of Public Administration Reform. Edward Elgar Publishing. 2023. pp. 1-26.

4. Bharucha A., Rhodes C., Boos C., Keller D., Dispenzieri A., Oldenburg R. Increased utilization of virtual visits and electronic approaches in clinical research during the COVID-19 pandemic and thereafter. In Mayo Clinic Proceedings. Elsevier. 2021. Vol. 96, No 9, pp. 332-341.

5. Ковальчук О. В., Іванченко С. М. Електронне урядування: основи та стратегії. – Вінниця: ВНАУ, 2021. 120 с.

6. Гайдукевич О. О., Гайдукевич І. О. Основи електронного урядування. – К.: НТУУ «КПІ імені Ігоря Сікорського», 2022. 150 с.

7. DocStar ECM Reviews & Product Details. URL: https://www.g2.com/products/docstar-ecm/reviews (дата звернення 07.05.2025).

8. DocStar ECM | Reviews, Pricing & Demos - SoftwareAdvice NZ. URL: https://www.softwareadvice.co.nz/software/32868/docstar-dms (дата звернення 07.05.2025).

9. DocStar ECM Reviews 2025. Verified Reviews, Pros & Cons. URL: https://www.capterra.com/p/82408/DocStar-ECM-AP-Automation/reviews/ (дата звернення 07.05.2025).

10. DocStar ECM Pros and Cons | User Likes & Dislikes. URL: https://www.g2.com/products/docstar-ecm/reviews?qs=pros-and-cons (дата звернення 07.05.2025).

11. Laserfiche Reviews - Gartner Peer Insights. URL: https://www.gartner.com/reviews/market/document-management/vendor/laserfiche/product/laserfiche (дата звернення 07.05.2025).

12. Laserfiche | Reviews, Pricing & Demos - SoftwareAdvice GB. URL: https://www.softwareadvice.co.uk/software/33063/laserfiche (дата звернення 07.05.2025).

13. Laserfiche Reviews 2025. Verified Reviews, Pros & Cons. URL: https://www.capterra.com/p/62196/Laserfiche/reviews/ (дата звернення 07.05.2025).

14. Laserfiche Reviews 2025: Details, Pricing, & Features. URL: https://www.g2.com/products/laserfiche/reviews (дата звернення 07.05.2025).

15. M-Files Reviews, Ratings & Features 2025. URL: https://www.gartner.com/reviews/market/document-management/vendor/m-files/product/m-files (дата звернення 07.05.2025).

16. M-Files Heightens User Experience with Enhanced Desktop User Interface | M- Files. URL: https://www.m-files.com/press-releases/m-files-heightens-user-experience-with-enhanced-desktop-user-interface/ (дата звернення 07.05.2025).

17. M-Files Reviews & Ratings 2025. URL: https://www.trustradius.com/products/m-files/reviews (дата звернення 07.05.2025).

18. M-Files Pricing, Alternatives & More 2025. URL: https://www.capterra.com/p/122215/M-Files/ (дата звернення 07.05.2025).

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

20. Raharjo, M., Napiah, M., & Anwar, R. S. Perancangan Sistem Informasi Dengan PHP Dan MYSQL Untuk Pendaftaran Sekolah Di Masa Pandemi. Computer Science (Co-Science). 2022. Vol. 2, No1, pp. 50-58.

21. Петров І. Основи Firebird SQL. – Харків, видавництво “Фоліо” 2020. – 350 с. Транслітерація: Ivan Petrov. Fundamentals of Firebird SQL. / Packt Publishing – Birmingham, 2018. 350 p.

22. Бондаренко В.В., Коваленко В.В. Розробка програмного забезпечення з використанням Python, Java та C#: Збірник наукових праць. - Одеса: Видавництво "Сонячна Україна", 2018. - 250 с.

23. Еккель Б. Мислити на Java: Навчальний посібник. - Харків: Видавництво Харківського національного університету імені В.Н. Каразіна, 2016. - 850 с.

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