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

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

(назва освітнього ступеня)

на тему: Розробка інформаційно-аналітичної системи для реєстрації звернень громадян з використанням технологій .NET

Виконав(ла): студент(ка) VI курсу, групи СПм-61
спеціальності 121
Інженерія програмного забезпечення
(шифр і назва спеціальності)

(підпис) Сарновський В.П. (прізвище та ініціали)

Керівник (підпис) Стоянов Ю. М. (прізвище та ініціали)

Нормоконтроль (підпис) Стоянов Ю. М. (прізвище та ініціали)

Завідувач кафедри (підпис) Петрик М. Р. (прізвище та ініціали)

Рецензент (підпис) Лещишин Ю. З. (прізвище та ініціали)

Тернопіль 2024

АНОТАЦІЯ

Кваліфікаційна робота магістра на тему «Розробка інформаційно-аналітичної системи для реєстрації звернень громадян з використанням технологій.NET ». Сарновський Володимир Павлович, Факультет комп'ютерно-інформаційних систем і програмної інженерії, Кафедра програмної інженеріїСПм-61, Тернопіль, 2024. С. – __, рис. – __, табл. – __, слайдів. – __, додатки. – , джерела. – .

Метою проєкту є створення інформаційно-аналітичної системи для реєстрації звернень громадян, що дозволяє автоматизувати процеси збору та обробки даних. У роботі представлено програмне забезпечення, розроблене з використанням сучасних технологій .NET, зокрема ASP.NET Core для вебдодатків та Entity Framework Core для роботи з базами даних.

Система дає змогу реєструвати звернення громадян, здійснювати їх обробку та аналіз даних з метою покращення якості надання послуг. Інтерфейс програми розроблений за допомогою ASP.NET MVC (Model-View-Controller), що забезпечує зручний та інтуїтивно зрозумілий досвід для користувачів веб-додатка.

Завдяки впровадженню цієї системи, органи влади зможуть ефективніше реагувати на запити громадян, а також здійснювати аналіз звернень для виявлення основних проблем та потреб населення. Система також сприятиме підвищенню прозорості та підзвітності державних установ.

Ключові слова: ІНФОРМАЦІЙНО-АНАЛІТИЧНА СИСТЕМА, РЕЄСТРАЦІЯ ЗВЕРНЕНЬ ГРОМАДЯН, .NET, ASP.NET CORE, MVC, ENTITY FRAMEWORK CORE, АНАЛІЗ ДАНИХ, ДЕРЖАВНІ УСТАНОВИ.

ABSTRACT

Master's thesis titled «Development of an Information and Analytical System for Citizen Request Registration Using .NET Technologies» by Volodymyr Pavlovych Sarnovskyi, Faculty of Computer and Information Systems and Software Engineering, Department of Software Engineering, SPm-61, Ternopil, 2024. Pages – __, figures – __, tables – __, slides – __, appendices – __, references – __.

The goal of the project is to develop an information and analytical system for registering citizen requests, which will automate the processes of data collection and processing. The work presents software based on modern .NET technologies, specifically ASP.NET Core for web applications and Entity Framework Core for working with databases.

The system enables the registration, processing, and data analysis of citizen requests to improve the quality of service delivery. The program's interface is designed using ASP.NET MVC (Model-View-Controller), providing a user-friendly and intuitive experience for web users.

With the implementation of this system, the authorities will be able to respond more effectively to citizens' requests, as well as analyze requests to identify the main problems and needs of the population. The system will also help to increase the transparency and accountability of government agencies. The system will also help to increase the transparency and accountability of government agencies.

Keywords: INFORMATION-ANALYTICAL SYSTEM, REGISTRATION OF CITIZENS' APPEALS, .NET, ASP.NET CORE, MVC, ENTITY FRAMEWORK CORE, DATA ANALYSIS, GOVERNMENT AGENCIES.

ЗМІСТ

ВСТУП

У сучасному світі швидкий доступ до інформації та ефективна комунікація між громадянами та державними установами є важливими частинами демократичного суспільства. Зростаючі вимоги до прозорості та підзвітності державних органів вимагають впровадження нових технологій, які здатні забезпечити оперативну обробку звернень громадян та їх аналіз. У цьому контексті інформаційно-аналітичні системи стають невіддільною частиною управлінських процесів, оскільки вони дозволяють не лише реєструвати звернення, але й аналізувати дані для покращення якості надання послуг.

Технології .NET, завдяки своїй гнучкості, продуктивності та кросплатформеності, є ідеальним вибором для розробки таких систем. Використання .NET Core, ASP.NET Core MVC та Entity Framework Core дозволяє створити масштабовану та надійну архітектуру програми, що відповідає сучасним вимогам до безпеки та продуктивності. Крім того, інтеграція з хмарними сервісами Microsoft Azure відкриває нові можливості для обробки великих обсягів даних і забезпечення доступності системи.

Метою даної роботи є розробка інформаційно-аналітичної системи для реєстрації звернень громадян, яка забезпечить автоматизацію процесів збору, обробки та аналізу інформації. У рамках роботи буде проведено аналіз наявних рішень у цій сфері, визначено основні функціональні вимоги до системи, а також реалізовано прототип на базі технологій .NET.

Таким чином, результати цієї роботи можуть стати основою для подальшого розвитку системи звернень в Україні та сприяти покращенню ефективності взаємодії між громадянами та державними органами.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ

1.1 Огляд конкурентів

У сфері реєстрації звернень громадян в Україні існує кілька основних систем, які виконують подібні функції, проте кожна з них має свої особливості, переваги та недоліки. Однією з найбільш значущих є Єдина система опрацювання звернень, яка була створена для забезпечення належного реагування на запити громадян через різноманітні канали зв'язку, включаючи телефонні дзвінки та інтернет. Ця система об'єднує різні органи виконавчої влади, зокрема Урядовий контактний центр та регіональні контактні центри, які працюють за принципом «єдиного вікна». Це дозволяє громадянам отримувати допомогу у вирішенні актуальних питань зручним для них способом [1].

Наступним важливим конкурентом є Система електронного документообігу Держмистецтв, яка дозволяє реєструвати звернення громадян через електронну платформу. Ця система включає формування реєстраційно-моніторингових карток (РМК), що забезпечує контроль за дотриманням строків розгляду звернень. Перевагою цієї системи є автоматизація процесу реєстрації, що значно спрощує роботу з документами та підвищує ефективність обробки запитів.

Міністерство економіки України також має свою систему для реєстрації усних і письмових звернень. Вона передбачає ведення окремого журналу для обліку звернень та створення сканованого образу кожного запиту, що дозволяє здійснювати моніторинг та аналіз звернень. Це забезпечує прозорість процесу та можливість контролю за виконанням.

Крім того, існують методичні рекомендації по роботі зі зверненнями громадян, які визначають порядок роботи з пропозиціями, заявами та скаргами. Ці рекомендації включають вимоги до реєстрації та контролю за виконанням звернень, що сприяє стандартизації процесів обробки запитів у державних органах.

Аналіз конкурентів показує, що теперішні системи вже мають певний рівень автоматизації та організації процесів реєстрації звернень громадян. Однак вони

часто стикаються з проблемами, такими як недостатня інтеграція між різними державними органами, обмежені можливості для аналізу даних і недостатня гнучкість у налаштуванні під специфічні потреби користувачів [1].

Розробка нової інформаційно-аналітичної системи на базі технологій .NET може заповнити ці прогалини, пропонуючи більш ефективні рішення для обробки звернень. Використання сучасних технологій дозволить не лише покращити управління даними, але й забезпечити вище реагування на запити громадян. Інтеграція аналітичних інструментів у систему дозволить здійснювати глибокий аналіз звернень, виявляти основні проблеми та потреби населення, а також підвищити загальну якість надання послуг. Таким чином, нова система може стати важливим кроком до покращення взаємодії між громадянами та державними установами, сприяючи розвитку прозорості та підзвітності в управлінні.

1.2 Обґрунтування вибору напрямку дослідження

Для розробки інформаційно-аналітичної системи реєстрації звернень громадян важливим є правильний вибір платформи, фреймворку, мови програмування та системи управління базами даних (СУБД). Кожен з цих компонентів впливає на продуктивність, безпеку, масштабованість та зручність підтримки майбутньої системи. Детальне обґрунтування вибору кожного з компонентів для створення даної системи наведено нижче.

Платформа .NET є однією з найбільш потужних та популярних середовищ для розробки різноманітного програмного забезпечення, включаючи веб-додатки, десктопні програми, мобільні додатки, хмарні рішення та мікросервіси. Вона розроблена і підтримується компанією Microsoft, що гарантує її стабільність, постійні оновлення та наявність підтримки впродовж тривалого часу. Для розробників ця платформа надає багатий набір інструментів і бібліотек, що

дозволяють значно прискорити процес розробки, забезпечити масштабованість і високу продуктивність системи [2].

Основні переваги .NET:

– Кросплатформність. Одна з ключових переваг сучасної версії .NET це підтримка розробки додатків для різних операційних систем (Windows, Linux, macOS). Це дозволяє організаціям обирати сервери і операційні системи, що найкраще підходять для їх інфраструктури, а також значно знижує витрати на технічну підтримку. Для інформаційно-аналітичної системи реєстрації звернень громадян така гнучкість є критичною, оскільки різні державні установи можуть мати різні вимоги щодо розгортання додатку.

– Висока продуктивність. Завдяки оптимізованим механізмам виконання коду, зокрема Just-In-Time (JIT) компіляції, .NET забезпечує високу швидкість обробки великих обсягів даних. Це є ключовою перевагою для системи, яка має оперативно реєструвати та обробляти значну кількість запитів.

– Масштабованість. Додатки, розроблені на .NET, легко масштабуються як вертикально, шляхом збільшення потужності серверів, так і горизонтально, додаючи нові сервери або мікросервіси, що дозволяє забезпечити стабільну роботу системи при підвищених навантаженнях.

– Гнучкість у розробці. Платформа .NET підтримує кілька архітектурних підходів, серед яких монолітні додатки, мікросервіси та хмарні рішення. Для інформаційно-аналітичної системи реєстрації звернень громадян особливо важливою є можливість використовувати мікросервісну архітектуру. Цей підхід дозволяє розділити систему на незалежні компоненти, кожен з яких виконує окремі функції. Наприклад, окремі мікросервіси можуть відповідати за реєстрацію звернень, їх обробку, формування звітів тощо. Така архітектура дозволяє не тільки спростити підтримку системи, але й легко масштабувати окремі її частини при зростанні навантаження.

ASP.NET Core — це фреймворк, який є частиною платформи .NET і використовується для створення веб-додатків та API. Він був обраний для

реалізації серверної частини інформаційно-аналітичної системи через свої переваги у продуктивності, масштабованості та підтримці сучасних стандартів безпеки.

Основні переваги ASP.NET Core:

– Кросплатформеність: Як і сама платформа .NET, ASP.NET Core підтримує кросплатформеність, коли системи можна розгортати на серверах, що працюють під управлінням різних операційних систем: Windows, Linux, macOS. Це може знизити витрати на ліцензування та технічну підтримку, а також дасть більшу гнучкість у виборі інфраструктури.

– Висока продуктивність: ASP.NET Core демонструє одну з найвищих продуктивностей серед веб-фреймворків завдяки своїй оптимізації для роботи з великими навантаженнями та використанню сучасних технологій. Це критично важливо для інформаційно-аналітичної системи, яка повинна ефективно обробляти великі масиви даних та відповідати на запити великої кількості користувачів одночасно.

– Контейнерна обробка і мікросервісна архітектура: ASP.NET Core підтримує мікросервісну архітектуру, контейнеризацію за допомогою Docker та інших технологій для полегшення розгортання, масштабування та системної підтримки. Іншими словами, можливість окремих сервісів системи працювати самостійно забезпечує надійність і масштабованість.

– Безпека: Є критично важливим аспектом для будь-якої системи, що працює з конфіденційними даними, такими як звернення громадян. ASP.NET Core забезпечує високий рівень безпеки завдяки вбудованим механізмам для захисту від поширених атак, таких як SQL Injection, Cross-Site Scripting (XSS) та Cross-Site Request Forgery (CSRF). Крім того, фреймворк підтримує сучасні стандарти аутентифікації та авторизації, такі як OAuth 2.0, OpenID Connect, що дозволяє безпечно керувати доступом до системи та захищати особисті дані користувачів. Для підвищення рівня безпеки також використовуються SSL/TLS сертифікати для шифрування даних під час їх передачі між сервером і клієнтами, що забезпечує конфіденційність інформації навіть у випадку перехоплення трафіку.

Мова програмування C# є основною для платформи .NET і була обрана для цього проєкту завдяки своїй гнучкості, високій продуктивності та тісній інтеграції з платформою. Вона підтримує об’єктно-орієнтоване програмування, що забезпечує створення добре структурованих і зручних для підтримки додатків [2].

Основні переваги C#:

– Об'єктно-орієнтоване програмування (ООП): C# підтримує всі принципи ООП, такі як інкапсуляція, спадкування та поліморфізм. Це дозволяє створювати добре структуровані додатки, де окремі компоненти можуть бути легко підтримуваними і модернізованими. Наприклад, для системи реєстрації звернень громадян окремі модулі можуть відповідати за реєстрацію звернень, їх обробку або аналіз.

– Безпека та автоматичне управління пам'яттю: C# використовує механізм Garbage Collection, що дозволяє уникнути витоків пам'яті та інших помилок, пов'язаних з ручним управлінням ресурсами. Це робить розробку додатків на C# надійною та безпечною, зменшуючи кількість помилок.

Для зберігання даних системи було обрано MySQL — одну з найпопулярніших і надійних СУБД з відкритим вихідним кодом. MySQL забезпечує високу продуктивність і надійність під час обробки великих обсягів даних, що робить її оптимальним вибором для інформаційно-аналітичних систем [3].

Основні переваги MS SQL Server:

– Висока продуктивність: MS SQL Server адаптований для роботи з великими обсягами даних і виконання складних запитів, що забезпечує стабільну та високу продуктивність бази даних навіть за умов значного навантаження. Це критично важливо для системи реєстрації звернень, яка буде обробляти значні обсяги інформації.

– Безпека та надійність: MS SQL Server забезпечує надійний захист даних завдяки вбудованому шифруванню, а також підтримує ефективні механізми резервного копіювання та відновлення, що гарантує збереження даних навіть у випадку непередбачених обставин.

– Масштабованість: MySQL легко масштабується для підтримки зростання обсягів даних і кількості користувачів. Вона підтримує горизонтальне та вертикальне масштабування, що дозволяє додавати нові сервери або збільшувати потужність існуючих без порушення роботи системи.

Архітектура MVC (Model-View-Controller) є однією з найбільш поширених архітектур для розробки веб-додатків. Цей підхід розділяє додаток на три окремі компоненти:

Model (Модель): Відповідає за обробку даних і бізнес-логіку системи. У випадку інформаційно-аналітичної системи модель буде працювати з базою даних, виконуючи операції зберігання та обробки звернень громадян.

View (Представлення): Це інтерфейс користувача, який відображає дані з моделі та дозволяє користувачам взаємодіяти із системою. У нашому випадку це буде веб-інтерфейс, через який громадяни та адміністратори зможуть взаємодіяти із системою.

Controller (Контролер): Відповідає за управління взаємодією між моделлю і представленням. Контролер обробляє запити від користувачів, отримує дані з моделі та передає їх до представлення для відображення.

Основні переваги архітектури MVC:

– Чітке розділення відповідальностей: Кожен компонент (модель, представлення, контролер) відповідає за свою частину системи, що спрощує її підтримку, розвиток та тестування. Для системи реєстрації звернень це означає легше масштабування і модернізацію окремих частин без впливу на інші компоненти.

– Модульність та масштабованість: MVC дозволяє розробникам швидко додавати нові функціональні можливості або змінювати існуючі компоненти без необхідності переписувати весь код. Це робить архітектуру ідеальною для великих систем, таких як інформаційно-аналітична система реєстрації звернень громадян.

РОЗДІЛ 2. РОЗРОБКА МОДЕЛІ ТА ПРОГРАМНОГО КОМПЛЕКСУ

2.1 Проєктування інформаційно-аналітичної системи для реєстрації звернень громадян

Розробка інформаційно-аналітичної системи для реєстрації звернень громадян є ключовим етапом у процесі автоматизації збору, обробки та зберігання звернень громадян. Основною метою системи є створення інтегрованого середовища, яке забезпечить ефективну роботу з поданими зверненнями, дозволить операторам швидко обробляти запити, а адміністраторам — контролювати процеси, генерувати звіти та управляти користувачами.

2.1.1. Розробка моделі предметної області

Модель предметної області охоплює не лише сутності, але й їхні атрибути, зв’язки та кардинальність відносин. Вона надає чітке уявлення про те, як інформація зберігається та обробляється в системі, гарантуючи узгодженість і цілісність даних.Основні завдання моделі предметної області:

– Ідентифікувати ключові сутності, які представляють дані, необхідні для функціонування системи.

– Визначити зв’язки між сутностями, які відображають логічну взаємодію даних.

– Створити оптимальну структуру для подальшого розроблення бази даних та програмної логіки системи.

Основними сутностями предметної області в системі реєстрації звернень громадян є громадяни (Citizen), які подають звернення, звернення (Appeal), що містять запити громадян, оператори (Operator), які обробляють ці звернення, та адміністратори (Admin), що контролюють роботу системи й управляють доступами

та ресурсами. Ці сутності відображають основні функціональні елементи системи, а їх атрибути забезпечують повний опис необхідної інформації.

У проєктуванні інформаційної системи розробнику допомагає уніфікована мова моделювання (UML) [4]. Основною причиною використання мови UML є забезпечення взаємодії розробників між собою на етапах проєктування і реалізації системи. UML дозволяє моделювати процеси та компоненти системи незалежно від деталей програмного коду, що спрощує розуміння і взаємодію між розробниками.

Моделювання в UML виконується для створення абстрактної моделі системи, що використовується як основа для подальшої реалізації. Мова UML дозволяє уникнути плутанини під час обговорень і деталізації, оскільки вона надає точні означення для опису системи. UML корисна в тих випадках, коли необхідна точність і структурованість, але не потрібні зайві подробиці програмної реалізації.

Для розробки інформаційної системи реєстрації звернень громадян були створені такі діаграми:

- Діаграма використання (Use Case Diagram);
- Діаграма класів (Class Diagram).

Діаграма використання є графічним представленням взаємодії користувачів (акторів) із системою. Вона показує основні варіанти використання, що відображають дії користувачів під час взаємодії з системою. Актори взаємодіють із системою через конкретні сценарії, які описують шлях до досягнення певної мети.

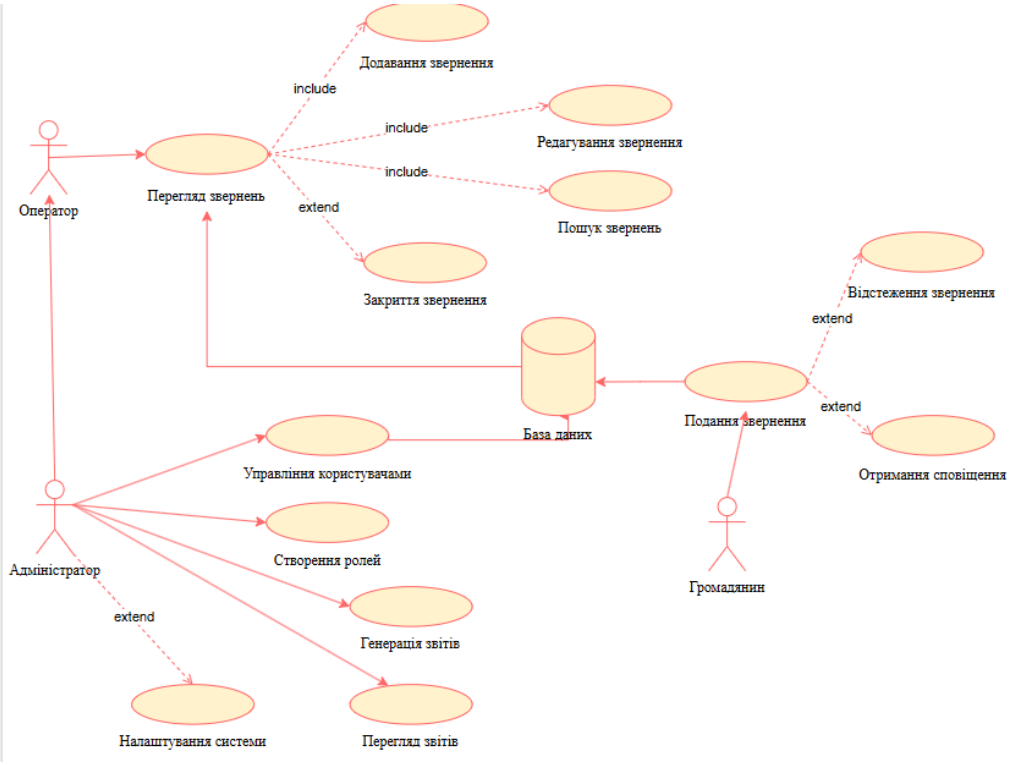

Рисунок 2.1 – Діаграма варіантів використання програмного забезпечення

В даній системі основними користувачами є:

– Оператор – відповідає за обробку звернень громадян: перегляд даних, додавання нових записів, редагування існуючих.

– Адміністратор – має розширені можливості: керування користувачами, перегляд статистики роботи системи, контроль за роботою операторів.

Основні варіанти використання програмного забезпечення включають наступні функціональні можливості:

– Робота з даними про звернення громадян: Оператор має можливість переглядати, додавати, редагувати, видаляти звернення.

– Управління користувачами: Адміністратор має можливість додавати нових користувачів, редагувати їхні дані, видаляти облікові записи, а також здійснювати пошук необхідних користувачів у системі.

– Вхід у систему: Функції авторизації, аутентифікації та ідентифікації користувачів.

2.1.2. Проєктування бази даних

Проєктування бази даних для інформаційно-аналітичної системи реєстрації звернень громадян є ключовим етапом, який забезпечує ефективне зберігання, управління та доступ до даних. Грамотно спроєктована структура бази даних дозволяє забезпечити швидкий доступ до інформації, зберігаючи її у безпечному, цілісному та впорядкованому вигляді, що є критично важливим для стабільної роботи системи.

Основною метою бази даних є зберігання інформації про громадян, їхні звернення, статуси звернень, а також дії операторів і адміністраторів [5]. Проєкт бази даних має включати ключові сутності, їх атрибути та зв’язки між ними, що відповідають логіці роботи системи. Опис основних сутностей бази даних наведено у таблиці 2.1..

Таблиця 2.1 – Опис сутностей бази даних

Ім’я сутності	Опис	Псевдоніми	Особливості використання
1	2	3	4
Citizen (Громадянин)	Зберігає інформацію про громадян, які подають звернення. Містить особисті дані, контактну інформацію.	User, Client	Використовується для зберігання інформації про осіб, які подають звернення. Один громадянин може подати кілька звернень.
Appeal (Звернення)	Представляє звернення, яке подає громадянин. Містить тему, зміст звернення та його статус.	Request, Query	Звернення містить дані про проблему або запит громадянина. Кожне звернення має унікальний ідентифікатор і прив’язане до громадянина та оператора.

Продовження таблиці 2.1

1	2	3	4
Operator (Оператор)	Описує співробітника, який обробляє звернення громадян. Містить дані про ім'я, роль та контактні дані оператора.	Agent, Worker	Використовується для відслідковування та управління зверненнями, обробленими конкретними операторами. Один оператор може обробляти кілька звернень.
Admin (Адміністратор)	Описує адміністратора системи, який має повний доступ до управління користувачами та зверненнями.	Supervisor, Manager	Адміністратор призначає звернення операторам, контролює роботу системи та генерує звіти. Може мати розширені права доступу.
AppealStatus (Статус звернення)	Містить дані про статус звернення, наприклад, "Нове", "В обробці", "Закрито".	Status, State	Використовується для відслідковування поточного етапу обробки звернення. Статус змінюється оператором або адміністратором.

– Citizen (Громадянин): Ця сутність відображає всі необхідні атрибути користувача системи — громадянина, який подає звернення. Вона також використовується для створення зв’язку зі зверненнями, що належать цьому громадянину.

– Appeal (Звернення): Основна сутність, яка зберігає всі дані про звернення, що подаються громадянами. Вона пов’язана з сутністю "Citizen" через унікальний ідентифікатор громадянина.

– Operator (Оператор): Ця сутність містить інформацію про співробітників, які обробляють звернення. Використовується для призначення відповідальних за обробку звернень.

– Admin (Адміністратор): Описує особу з повним доступом до системи, яка управляє процесом обробки звернень і користувачами системи.

– AppealStatus (Статус звернення): Статуси звернень визначають етап обробки і дозволяють контролювати хід роботи з кожним зверненням.

Таблиця 2.2 – Опис атрибутів сутностей бази даних

Назва сутності	Атрибут	Опис	Домен	Значення за замовчуванням	Псевдонім	Значення Null
1	2	3	4	5	6	7
Citizen	CitizenID	Унікальний ідентифікатор громадянина.	INT	AUTO_INCREMENT	ID, UserID	Ні
	FullName	Повне ім'я громадянина.	NVARCHAR(255)	Немає	Name	Ні
	Email	Контактна електронна пошта громадянина.	NVARCHAR(255)	Немає	UserEmail	Ні
	PhoneNumber	Номер телефону для зв'язку.	NVARCHAR(50)	NULL	Phone	Так
	RegisteredAt	Дата реєстрації громадянина в системі.	DATETIME	CURRENT_TIMESTAMP	DateRegistered	Ні
Appeal	AppealID	Унікальний ідентифікатор звернення.	INT	AUTO_INCREMENT	RequestID	Ні
	CitizenID	Ідентифікатор громадянина, який подав звернення.	INT	Немає	UserID	Ні
	Subject	Тема звернення.	NVARCHAR(255)	Немає	AppealTitle	Ні
	Content	Опис проблеми.	TEXT	Немає	AppealText	Ні

Продовження таблиці 2.2

1	2	3	4	5	6	7
	StatusID	Ідентифікатор статусу звернення.	INT	1 (Нове)	AppealState	Ні
	OperatorID	Ідентифікатор оператора, який обробляє звернення.	INT	NULL	AssignedOperator	Так
	CreatedAt	Дата створення звернення.	DATETIME	CURRENT_TIMESTAMP	DateCreated	Ні
	UpdatedAt	Дата останнього оновлення звернення.	DATETIME	NULL	DateUpdated	Так
Operator	OperatorID	Унікальний ідентифікатор оператора.	INT	AUTO_INCREMENT	AgentID	Ні
	FullName	Ім'я оператора.	NVARCHAR(255)	Немає	OperatorName	Ні
	Role	Роль оператора (оператор або адміністратор).	NVARCHAR(50)	'Operator'	UserRole	Ні
	Email	Контактна електронна пошта оператора.	NVARCHAR(255)	Немає	OperatorEmail	Ні
Admin	AdminID	Унікальний ідентифікатор адміністратора.	INT	AUTO_INCREMENT	AdminID	Ні
	FullName	Ім'я адміністратора.	NVARCHAR(255)	Немає	AdminName	Ні
	Email	Контактна електронна пошта адміністратора.	NVARCHAR(255)	Немає	AdminEmail	Ні

Продовження таблиці 2.2

	Role	Роль адміністратора (за замовчуванням "Admin").	NVARCHAR(50)	'Admin'	UserRole	Hi
AppealStatus	StatusID	Унікальний ідентифікатор статусу звернення.	INT	AUTO_INCREMENT	AppealStateID	Hi
	StatusName	Назва статусу звернення (напр. "Нове", "Закрито").	NVARCHAR(50)	'New'	AppealStateName	H

Під час розробки бази даних необхідно створити її ER-діаграму. Для побудови ER-діаграми спочатку потрібно визначити первинні ключі сутностей та типи зв'язків, а також їх кардинальність [6]. Для кожної сутності було визначено відповідний первинний ключ. У результаті аналізу предметної області були визначені та встановлені зв’язки між сутностями, що забезпечує коректне моделювання даних та їх взаємодію в рамках системи (див. Табл. 2.3).

Таблиця 2.3 – Опис атрибутів сутностей бази даних

Ім’я сутності 1	Назва зв’язку	Ім’я сутності 2	Кардинальність
1	2	3	4
Citizen (Громадянин)	Подати звернення	Appeal (Звернення)	1:N (Один громадянин може подати кілька звернень)
Appeal (Звернення)	Обробляється	Operator (Оператор)	N:1 (Багато звернень можуть бути оброблені одним оператором)

Продовження таблиці 2.3

1	2	3	4
Appeal (Звернення)	Має статус	AppealStatus (Статус звернення)	N:1 (Одне звернення може мати лише один статус одночасно)
Operator (Оператор)	Призначений адміністратором	Admin (Адміністратор)	N:1 (Оператор призначається одним адміністратором)

Далі необхідно обрати модель для проєктування бази даних. Ми використовуємо реляційну модель даних для побудови логічної схеми. У межах реляційної моделі логічне проєктування полягає в створенні реляційної схеми, визначенні структури відношень бази даних та вирішенні низки інших завдань. Правильність логічної моделі перевіряється за допомогою нормалізації, що дозволяє впевнитися в її структурній узгодженості, логічній цілісності та мінімальній надмірності. На цьому етапі буде визначено набір відношень на основі структури логічної моделі даних [6].

На цьому етапі створюються відношення на основі сутностей та їх зв’язків. Зв’язки моделюються через використання первинних та зовнішніх ключів. Для кожної сутності формується окреме відношення, де атрибути сутності стають атрибутами відповідного відношення. У дочірнє відношення додаються атрибути, що відповідають первинному ключу батьківської сутності, які стають зовнішнім ключем у дочірньому відношенні. Після цього виконується процес нормалізації, що дозволяє усунути надлишковість та підвищити структурну коректність логічної моделі даних.

Процедура нормалізації бази даних спрямована на усунення надмірності даних та виявлення функціональних залежностей між атрибутами. Цей процес забезпечує компактність збережених даних, уникаючи дублювання, що допомагає запобігти аномаліям при вставці, видаленні або оновленні даних після реалізації бази. Функціональні залежності встановлюють взаємозв'язок між атрибутами одного відношення і їх значеннями в іншому, що дозволяє підтримувати логічну

цілісність. Правила нормалізації, хоч і прості, але дуже суворі, і кожне наступне правило підвищує рівень відповідності бази теорії реляційних даних [6].

Існують кілька рівнів нормалізації бази даних, серед яких перша нормальна форма (1НФ), друга нормальна форма (2НФ), третя нормальна форма (3НФ), четверта нормальна форма (4НФ) та п’ята нормальна форма (5НФ). Проте жодна з сучасних реляційних СУБД не забезпечує повної підтримки всіх п'яти нормальних форм через високі вимоги до продуктивності. У повністю нормалізованій базі даних для виконання одного запиту може знадобитися об'єднання великої кількості таблиць, що значно знижує продуктивність системи і може не задовольняти користувачів. Тому на практиці зазвичай використовують перші три рівні нормалізації (1НФ, 2НФ та 3НФ), що забезпечує достатню структурну узгодженість даних без істотної втрати продуктивності [6].

Перша нормальна форма (1НФ) означає, що у кожній комірці таблиці бази даних повинно бути лише одне атомарне значення, тобто на перетині рядка та стовпця не може бути множинних значень. Друга нормальна форма (2НФ) вимагає, щоб відношення знаходилось у першій нормальній формі, і кожен атрибут, який не входить до складу первинного ключа, повністю залежав від цього ключа. Третя нормальна форма (3НФ) передбачає, що відношення має знаходитись у 2НФ і не містити атрибутів, які перебувають у транзитивній залежності від первинного ключа.

Аналізуючи структуру відношень розробленої бази даних, можна зробити висновок, що вони будуть нормалізовані до третьої нормальної форми, що забезпечить ефективне зберігання та маніпуляцію даними без зайвої дубльованої інформації. Виходячи з аналізу предметної області, далі буде побудована структура таблиць бази даних за допомогою SQL-запитів. Нижче наведено лістинг створення цих таблиць.допомогою SQL-запитів. Лістинг створення наведено нижче:

Лістинг 2.1 SQL-запит для створення таблиці «Citizen»

```
CREATE TABLE Citizen (
    CitizenID INT PRIMARY KEY AUTO_INCREMENT,
    FullName NVARCHAR(255) NOT NULL,
    Email NVARCHAR(255) NOT NULL,
    PhoneNumber NVARCHAR(50),
    RegisteredAt DATETIME DEFAULT CURRENT_TIMESTAMP
);
```

Лістинг 2.2 SQL-запит для створення таблиці «Appeal»

```
CREATE TABLE Appeal (
    AppealID INT PRIMARY KEY AUTO_INCREMENT,
    CitizenID INT,
    Subject NVARCHAR(255) NOT NULL,
    Content TEXT NOT NULL,
    StatusID INT,
    OperatorID INT,
    CreatedAt DATETIME DEFAULT CURRENT_TIMESTAMP,
    UpdatedAt DATETIME,
    FOREIGN KEY (CitizenID) REFERENCES Citizen(CitizenID),
    FOREIGN KEY (StatusID) REFERENCES AppealStatus(StatusID),
    FOREIGN KEY (OperatorID) REFERENCES Operator(OperatorID)
);
```

Лістинг 2.3 SQL-запит для створення таблиці «Operator»

```
CREATE TABLE Operator (
    OperatorID INT PRIMARY KEY AUTO_INCREMENT,
    FullName NVARCHAR(255) NOT NULL,
    Role NVARCHAR(50) NOT NULL,
    Email NVARCHAR(255) NOT NULL
);
```

Лістинг 2.4 SQL-запит для створення таблиці «AppealStatus»

```
CREATE TABLE AppealStatus (
    StatusID INT PRIMARY KEY AUTO_INCREMENT,
    StatusName NVARCHAR(50) NOT NULL
);
```

Лістинг 2.5 SQL-запит для створення таблиці «Admin»

```
CREATE TABLE Admin (
    AdminID INT PRIMARY KEY AUTO_INCREMENT,
    FullName NVARCHAR(255) NOT NULL,
    Email NVARCHAR(255) NOT NULL,
    Role NVARCHAR(50) NOT NULL DEFAULT 'Admin'
);
```

2.1.3. Проєктування архітектури

Проєктування архітектури інформаційно-аналітичної системи (IAC) реєстрації звернень громадян є ключовим етапом, що забезпечує ефективність, масштабованість та надійність системи. Як основу архітектури було обрано фреймворк ASP.NET Core MVC, який розподіляє функціональність системи на три ключові компоненти: модель (Model), подання (View) та контролер (Controller). Такий підхід забезпечує чітку структуру коду, спрощує процеси тестування та підтримки, а також дозволяє масштабувати систему без значних змін у її основі.

ASP.NET Core MVC забезпечує високу продуктивність та можливість інтеграції з іншими сервісами і технологіями, такими як бази даних SQL, NoSQL або сторонні API, що робить систему гнучкою та готовою до обробки великих обсягів даних. Крім того, система підтримує паралельну роботу багатьох користувачів, що важливо для обробки великої кількості звернень громадян [7].

Для підвищення продуктивності та безперебійної роботи було впроваджено механізми кешування даних, балансування навантаження і асинхронної обробки запитів, що значно скорочує час відповіді на звернення користувачів. Таким чином, обрана архітектура дозволяє забезпечити стабільну роботу IAC навіть при високих навантаженнях та великій кількості звернень.

Архітектура ASP.NET Core MVC базується на трьох ключових компонентах: Модель (Model), Представлення (View) і Контролер (Controller). Вони працюють у взаємодії, забезпечуючи ефективне функціонування інформаційно-аналітичної системи (IAC) для реєстрації звернень громадян. Моделі відповідають за зберігання та управління даними системи, визначаючи структуру таких бізнес-об'єктів, як звернення громадян, статуси та категорії звернень. Це дозволяє чітко організовувати дані та полегшує їх подальше обслуговування і розвиток.

Контролери є ядром бізнес-логіки. Вони обробляють запити від користувачів, працюють з моделями для отримання та обробки інформації, а також підготовлюють дані для подальшого відображення. Контролери діють як проміжна

ланка між користувачем та бізнес-логікою, що дозволяє ефективно керувати запитами і результатами, передаючи їх у відповідній формі для подальшої візуалізації.

Представлення (Views) відповідають за візуальне відображення інформації для користувачів. За допомогою HTML, CSS та JavaScript вони формують інтерфейс, через який користувачі можуть взаємодіяти з системою, реєструвати звернення, відстежувати їх статуси та отримувати необхідні звіти [7].

Основою серверного середовища є ASP.NET Core, що пропонує потужні інструменти для побудови високопродуктивних та кросплатформних веб-додатків. Система підтримує розгортання на різних операційних системах, зокрема Windows, Linux та macOS, що підвищує її гнучкість у використанні, забезпечує ширшу доступність та сприяє зниженню витрат на ліцензування та інфраструктуру. Система вирізняється високою продуктивністю, що дозволяє ефективно обробляти великі обсяги даних і витримувати високі навантаження. Це особливо важливо для опрацювання численних звернень громадян в умовах інтенсивної роботи та значної кількості користувачів.

Для зберігання даних в інформаційно-аналітичній системі (IAC) реєстрації звернень громадян було обрано Microsoft SQL Server як систему управління базами даних (СУБД). Цей вибір обґрунтовано її високою продуктивністю, надійністю та здатністю ефективно працювати з великими обсягами даних. Microsoft SQL Server пропонує розширені можливості управління транзакціями, підтримує масштабування та забезпечує високу доступність даних навіть при значних навантаженнях. Для взаємодії з базою даних у системі використовується Entity Framework Core — об'єктно-реляційний мапер (ORM). Він дозволяє розробникам працювати з базою даних через об'єкти мови C#, значно зменшуючи необхідність написання складних SQL-запитів. Це спрощує процес розробки, підтримки та розширення системи, а також забезпечує зручну реалізацію CRUD-операцій та інших функцій [7].

Безпека системи є важливим аспектом проєктування архітектури, оскільки система працює з конфіденційною інформацією громадян. ASP.NET Core пропонує

вбудовані механізми безпеки, такі як захист від SQL-ін'єкцій, міжсайтових скриптів (XSS) і підробки запитів (CSRF). Це допомагає запобігти основним загрозам і захищає дані від несанкціонованого доступу. Крім того, система використовує сучасні протоколи аутентифікації та авторизації, зокрема OAuth 2.0 та OpenID Connect, що забезпечує надійне управління доступом до чутливих даних.

Структура ASP.NET Core MVC, заснована на принципі розділення відповідальності (MVC), дозволяє чітко розділити логіку додатку на окремі компоненти, що спрощує розробку та підтримку.Модульна структура забезпечує можливість легко додавати нові функції або оновлювати існуючі компоненти без значних змін у всій системі [9]. Це дозволяє системі бути гнучкою і адаптивною до змінних вимог та нових технологій. Також можливість інтеграції з зовнішніми сервісами та інструментами, такими як Power BI для аналітики чи звітування, підвищує функціональність системи і робить її універсальною для подальшого розвитку та масштабування.

Узагальнюючи, архітектура, побудована на основі ASP.NET Core MVC, є оптимальним рішенням для розробки інформаційно-аналітичної системи реєстрації звернень громадян. Система відзначається високою продуктивністю, надійною системою безпеки та гнучкою архітектурою, що дозволяє легко масштабувати її під будь-які потреби. Чіткий поділ на компоненти (модель, представлення, контролер) дозволяє ефективно управляти всіма аспектами функціонування системи, від зберігання і обробки даних до їх візуалізації і надання результатів користувачам. Такий підхід не лише спрощує підтримку та оновлення системи, але й забезпечує гнучкість для подальшої інтеграції з іншими інструментами та сервісами, що відповідають потребам розширення функціональності системи.

2.2. Конструювання інформаційно-аналітичної системи реєстрації звернень громадян

2.2.1. Реалізація ключових класів

У процесі конструювання інформаційно-аналітичної системи реєстрації звернень громадян важливим етапом є реалізація ключових класів, які забезпечують основну бізнес-логіку системи. Ключові класи відповідають за управління даними, взаємодію з користувачами та інтеграцію з іншими частинами системи. Основними класами в цій системі є класи моделей, контролерів та сервісів.

Діаграма класів є важливою частиною проєктування системи, оскільки вона показує основні сутності, їхні атрибути, методи та зв’язки між ними [7]. Нижче наведені основні класи системи та їхні взаємозв'язки.

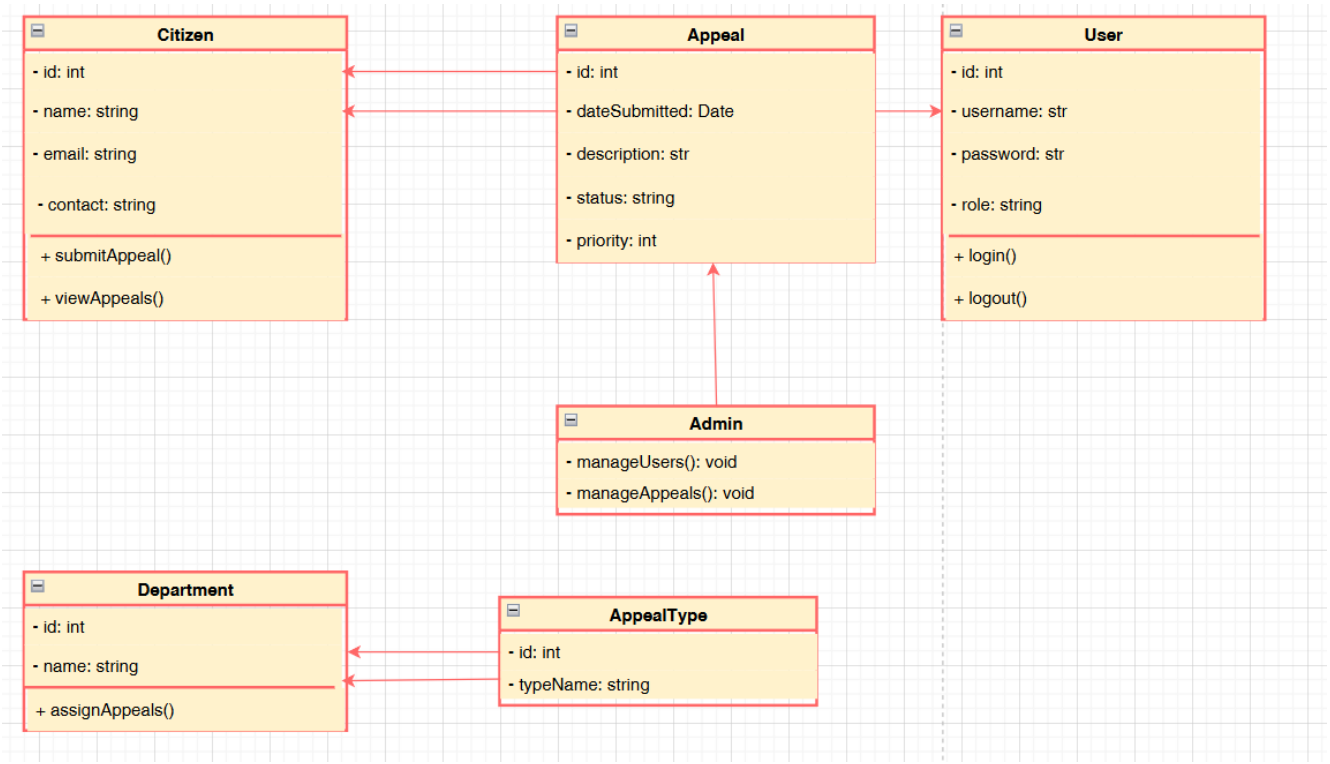

Рисунок 2.2 – Діаграма класів

Основні класи:

1. Citizen (Клас, що представляє звернення, подане громадянином).

Атрибути:

– CitizenID: Унікальний ідентифікатор громадянина.

– FullName: Повне ім'я громадянина.

– Email: Контактна електронна пошта.

– PhoneNumber: Контактний телефон.

Методи:

– SubmitAppeal(subject, content): Метод для подачі нового звернення. Повертає новий об'єкт класу Appeal.

Лістинг 2.6 – Класу Citizen та методу SubmitAppeal

```
public class Citizen
{
    public int CitizenID { get; set; }
    public string FullName { get; set; }
    public string Email { get; set; }
    public string PhoneNumber { get; set; }
    public Appeal SubmitAppeal(string subject, string content)
    {
        return new Appeal
        {
            Subject = subject,
            Content = content,
            CitizenID = this.CitizenID,
            CreatedAt = DateTime.Now,
            StatusID = AppealStatus.New};}}
```

2. Appeal (Клас, що представляє звернення, подане громадянином. Він містить усі необхідні атрибути для обробки звернення та методи для взаємодії із системою.).

Атрибути:

– AppealID: Унікальний ідентифікатор звернення.

– CitizenID: Ідентифікатор громадянина, який подав звернення.

– Subject: Тема звернення.

– Content: Опис проблеми.

– StatusID: Поточний статус звернення

– CreatedAt: Дата створення звернення.

– UpdatedAt: Дата останнього оновлення звернення.

Методи:

– UpdateStatus(newStatus): Оновлює статус звернення на новий.

– AssignOperator(operatorID): Призначає оператора для обробки звернення.

–

Лістинг 2.7 – Класу Appeal та його методів

```
public class Appeal
{
    public int AppealID { get; set; }
    public int CitizenID { get; set; }
    public string Subject { get; set; }
    public string Content { get; set; }
    public int StatusID { get; set; }
    public int? OperatorID { get; set; }  // Може бути null, якщо ще
не призначено оператора
    public DateTime CreatedAt { get; set; }
    public DateTime? UpdatedAt { get; set; }
    public void UpdateStatus(int newStatus)
    {
        this.StatusID = newStatus;
        this.UpdatedAt = DateTime.Now; }
    public void AssignOperator(int operatorID)
    {this.OperatorID = operatorID;
    this.UpdatedAt = DateTime.Now;}}
```

3. Operator (Клас оператора, який обробляє звернення).

Атрибути:

– OperatorID: Унікальний ідентифікатор оператора.

– FullName: Ім'я оператора.

– Role: Роль оператора (наприклад, "Оператор" або "Адміністратор").

– Email: Електронна пошта для зв'язку.

Методи:

– ProcessAppeal(appeal): Обробляє звернення, змінюючи його статус на «в обробці» або «закрито».

Лістинг 2.8 – Класу Operator та його методів

```
public class Operator
{
    public int OperatorID { get; set; }
    public string FullName { get; set; }
    public string Role { get; set; }
    public string Email { get; set; }
    public void ProcessAppeal(Appeal appeal, int newStatus)
    {
        appeal.UpdateStatus(newStatus);
        Console.WriteLine($"Operator {this.FullName} processed
appeal {appeal.AppealID}");}}
```

4. Admin (Клас адміністратора, який керує системою та розподіляє звернення між операторами).

Атрибути:

– AdminID: Унікальний ідентифікатор адміністратора.

– FullName: Ім'я адміністратора.

– Role: Роль адміністратора ("Admin").

– Email: Контактний email.

Методи:

– AssignAppeal(appeal, operator): Призначає звернення оператору для обробки.

Лістинг 2.9 – Класу Admin та його методів

```
public class Admin : Operator
{
    public void AssignAppeal(Appeal appeal, Operator operator)
    {
        appeal.AssignOperator(operator.OperatorID);
        Console.WriteLine($"Admin {this.FullName} assigned appeal
{appeal.AppealID} to operator {operator.FullName}");
    }
}
```

5. AppealStatus (Клас, що зберігає статуси звернень).

Атрибути:

– StatusID: Унікальний ідентифікатор статусу.

– StatusName: Назва статусу.

Лістинг 2.10 – Класу AppealStatus та його методів

```
public class AppealStatus
{
    public int StatusID { get; set; }
    public string StatusName { get; set; }
}
```

2.2.2. Розробка GUI

Графічний інтерфейс користувача (GUI) є важливою частиною інформаційно-аналітичної системи для реєстрації звернень громадян. Він забезпечує зручний і інтуїтивний доступ користувачів до функціональних можливостей системи. Розробка GUI має на меті створення інтерфейсу, який дозволить громадянам подавати звернення, операторам — ефективно їх обробляти, а адміністраторам — контролювати роботу системи[8].

Основні вимоги до GUI:

– Зручність та інтуїтивність: Інтерфейс повинен бути зрозумілим для користувачів без спеціальних технічних знань. Простий і логічно структурований дизайн дозволить користувачам швидко освоїти систему.

– Функціональність: GUI повинен забезпечувати повний доступ до основних функцій системи, таких як подання звернень, перегляд статусу звернень, обробка та моніторинг звернень.

– Адаптивність: Інтерфейс має бути адаптивним для коректного відображення на різних пристроях (настільні комп’ютери, планшети, мобільні телефони).

– Безпека: Всі дії користувачів повинні бути захищені, включаючи процеси авторизації, доступу до конфіденційних даних та обробки звернень.

При запуску програми користувачу пропонується заповнити форму авторизації, вказавши свій логін (як правило, це адреса електронної пошти) та пароль. Після натискання кнопки «Увійти» система перевіряє введені дані на

відповідність збереженим у базі даних. У разі успішної перевірки користувач отримує доступ до функціоналу програми. Дизайн форми мінімалістичний і не перевантажений зайвими елементами, що дозволяє легко зорієнтуватися користувачеві. Для забезпечення коректності введених даних, система реалізує функцію перевірки даних та інформування користувача про виявлені помилки..

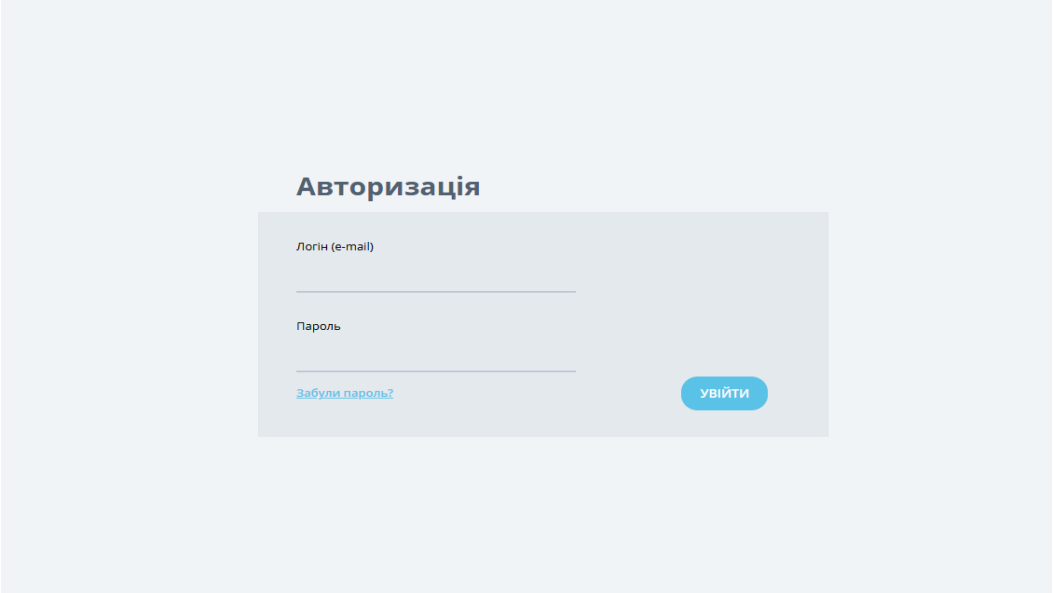

Рисунок 2.3 –Авторизації

Для нових користувачів передбачена форма реєстрації, де необхідно вказати ім’я користувача, електронну пошту та пароль для створення особистого кабінету. Інтерфейс організований так, щоб користувач міг легко заповнити всі необхідні поля без зайвих зусиль.

Після реєстрації користувач отримує повідомлення про успішне завершення операції, що дає йому зворотний зв'язок про виконану дію.

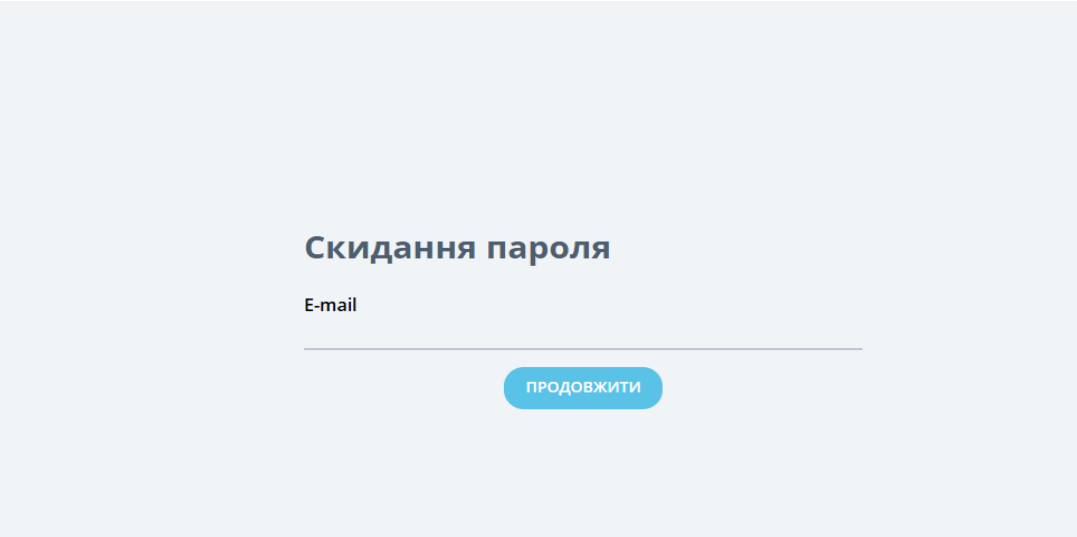

Рисунок 2.4 – Скидання паролю

Головне вікно Після авторизації користувач переходить до головного вікна програми. У цьому вікні відображається список звернень або іншої інформації залежно від ролі користувача. Головне меню дозволяє користувачу вибирати різні опції для взаємодії із системою. Основні елементи інтерфейсу включають:

– Список даних: Наприклад, якщо користувач працює з певною інформацією, наприклад зверненнями громадян або іншими даними, він може переглядати їх у вигляді списку.

– Меню дій: Дозволяє вибирати основні операції, такі як додавання або редагування записів.

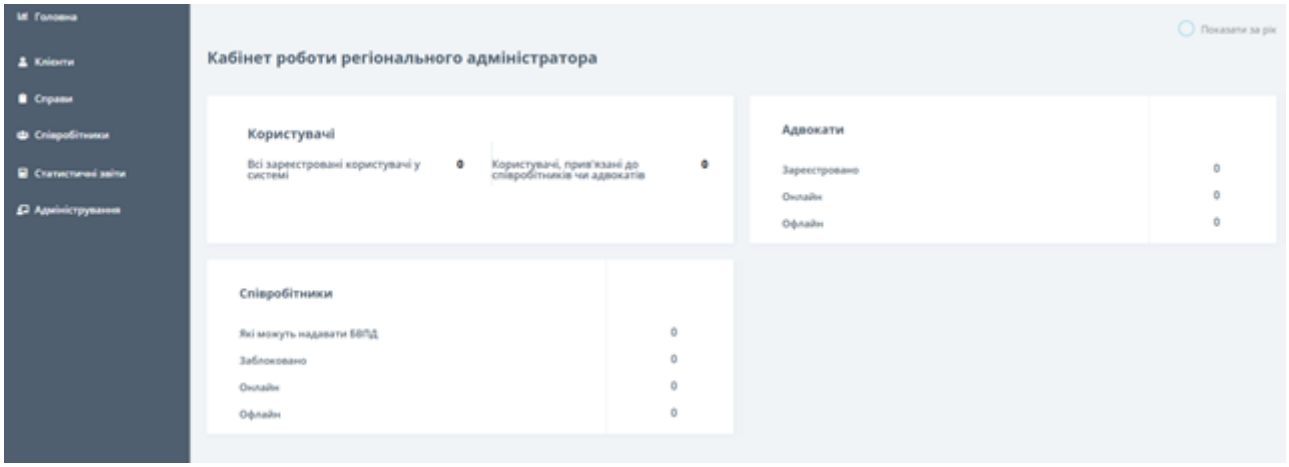

Рисунок 2.5 – Головне вікно програми

Вікно перегляду та оновлення даних При роботі з конкретними записами, такими як звернення чи інші дані, користувач може відкривати вікно для перегляду або оновлення інформації. Це вікно дозволяє переглядати детальну інформацію про вибраний елемент, а також додавати нові дані чи оновлювати існуючі. Інтерфейс підтримує просту навігацію та чітку структуру полів для введення нових даних.

– Клієнти: У цьому розділі користувач може переглядати інформацію про клієнтів, додавати нові записи та редагувати існуючі дані. Цей розділ призначений для комплексного обслуговування інформації про клієнтську базу.

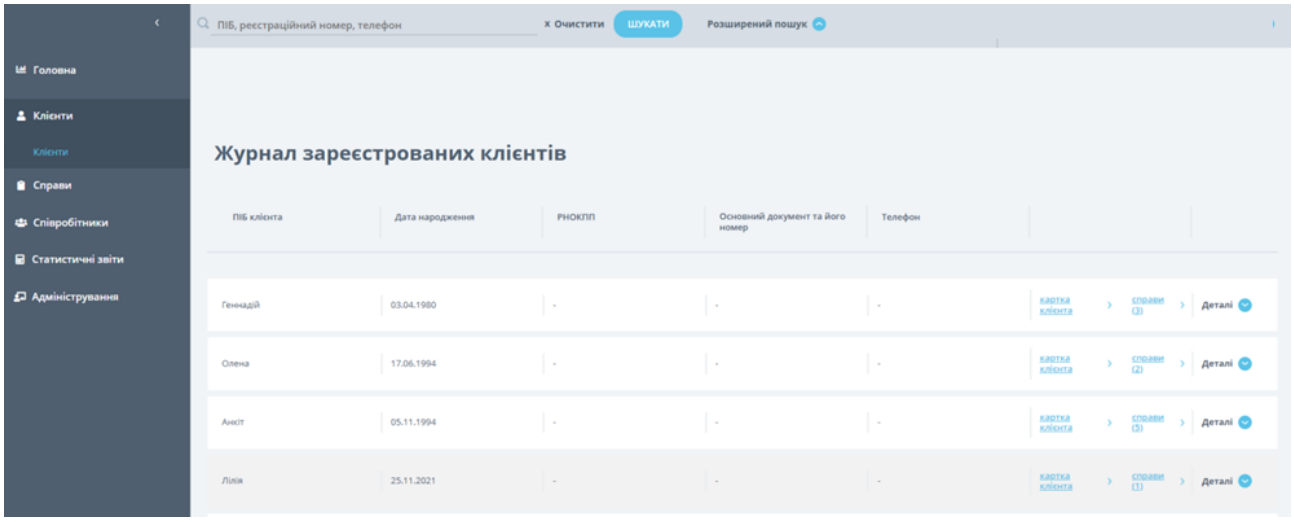

Рисунок 2.6 –Перегляд клієнтів

– Справи: Вкладка для управління справами, що можуть включати звернення клієнтів та інші документи, пов'язані з їх розглядом. Тут користувач може створювати нові справи або працювати з наявними, вносячи необхідні зміни.

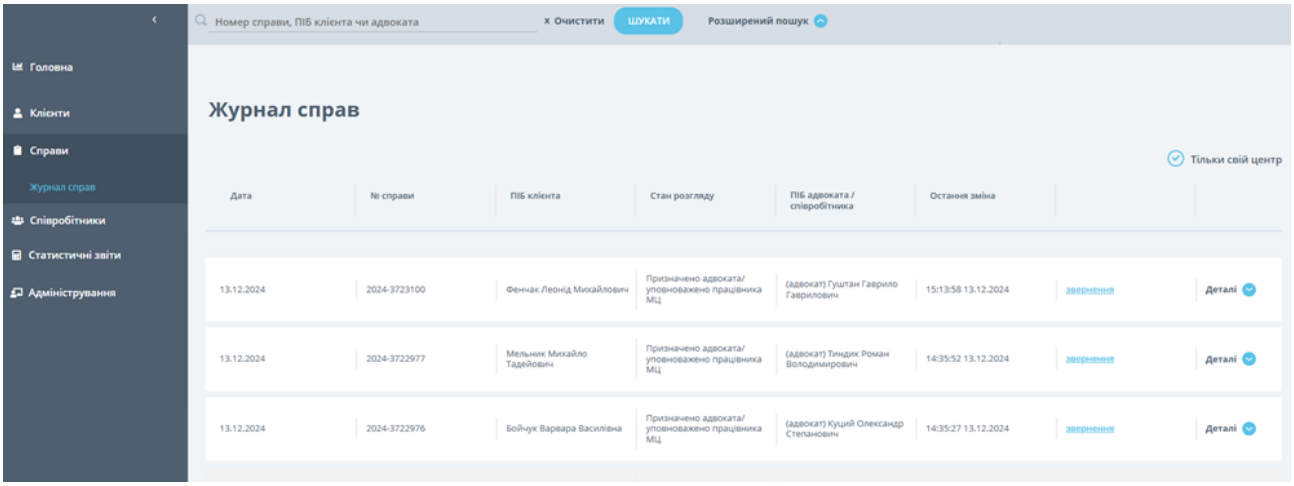

Рисунок 2.7 –Перегляд справ

– Акти: У цьому розділі розміщуються акти, пов'язані з роботою системи або клієнтами. Користувач може переглядати і редагувати акти.

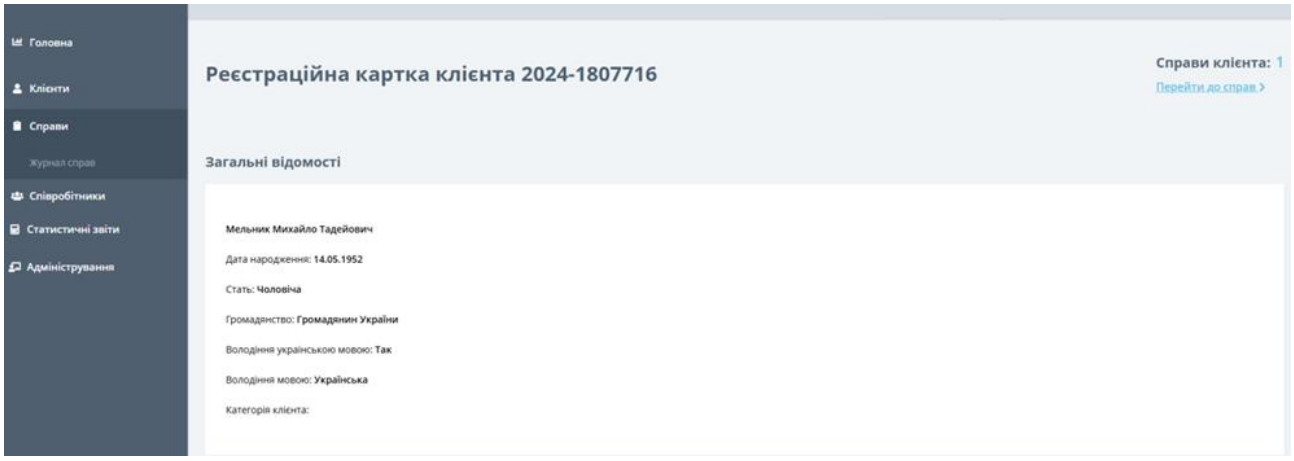

Рисунок 2.8 – Вікно перегляду реєстраційної карти клієнта

– Звіти: У цьому розділі можна генерувати різні види звітів за результатами обробки справ або клієнтських запитів. Звіти допомагають аналізувати ефективність роботи системи та операцій з клієнтами.

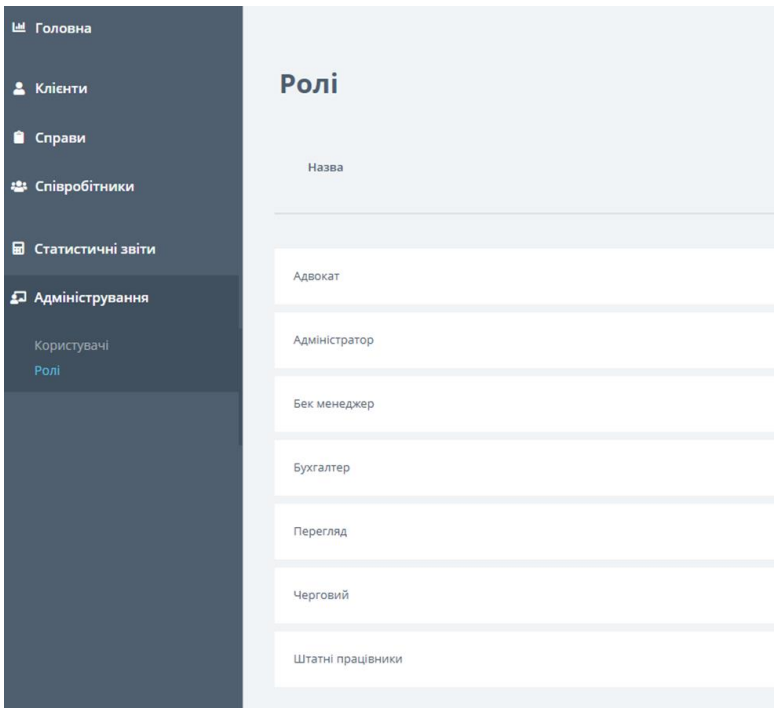

Рисунок 2.9 – Ролі користувачів

– Довідники: Розділ містить каталог довідкових даних, необхідних для роботи системи. Це можуть бути списки організацій, типів документів або інша допоміжна інформація, яку можна редагувати або оновлювати.

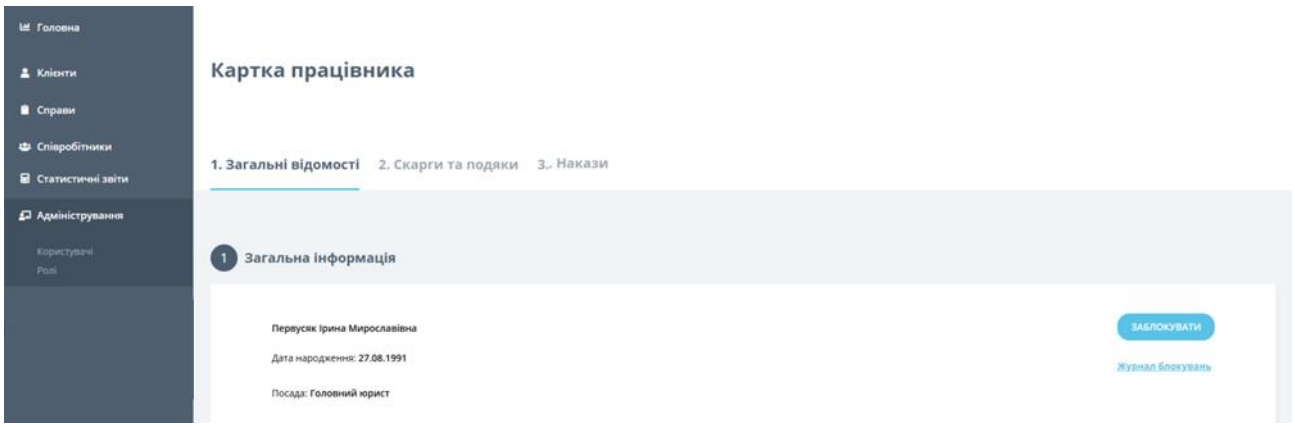

Рисунок 2.10 – Вікно редагування інформації про співробітника

– Статистичні звіти: У цій вкладці користувач може переглядати статистичні звіти, що базуються на даних системи. Вони дозволяють аналізувати різні аспекти роботи системи на основі оброблених звернень, часу їх розгляду, кількості клієнтів та інших показників.

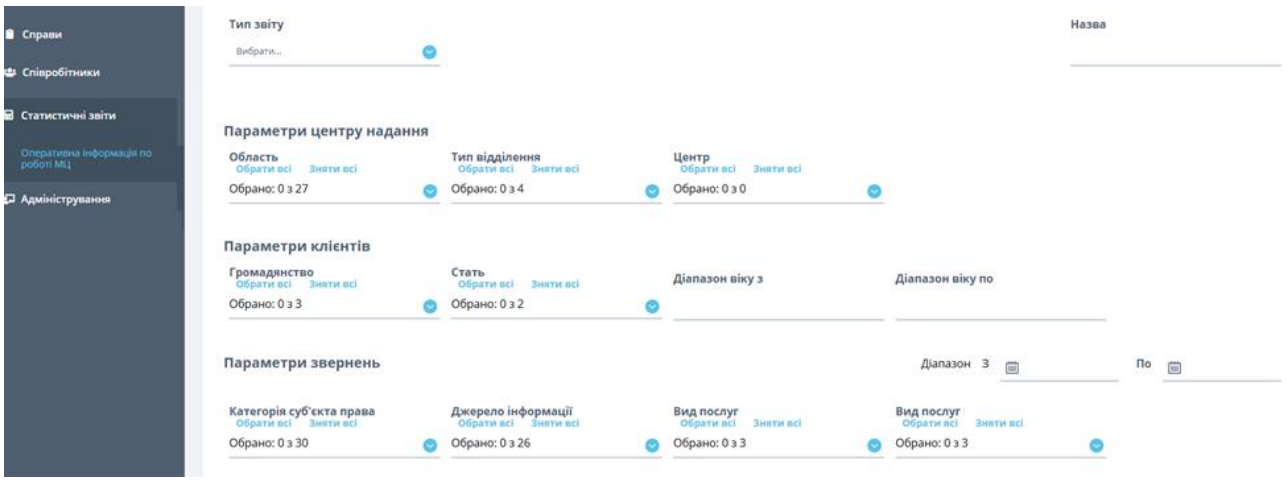

Рисунок 2.11 – Вікно створення статистичних звітів

Для забезпечення зручної роботи з кожним розділом, програма передбачає відкриття окремого вікна перегляду та редагування. Це дозволяє користувачам ефективно керувати записами в базі даних, будь то клієнти, справи або звіти. Вікна включають відповідні поля для введення даних і кнопки для підтвердження змін.

Для швидкого пошуку необхідної інформації передбачена фільтрація . Це дозволяє користувачу вводити різні критерії для фільтрації даних та отримувати результати у вигляді списку. За допомогою цього інструменту можна значно прискорити роботу з великою кількістю записів.

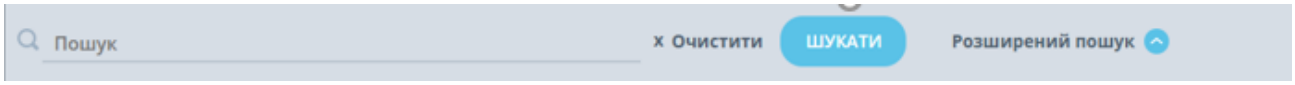

Рисунок 2.12 –Фільтрація

Функція експорту Користувач може експортувати результати пошуку або звіти у формати, зручні для подальшої роботи, наприклад, у Excel або Word. Це підвищує ефективність роботи з даними та дозволяє створювати документи для подальшого використання.

Профіль користувача У системі також є можливість перегляду та редагування профілю користувача. Тут користувач може змінювати особисті дані, такі як ім’я, електронна пошта або пароль.

Чітка структура та простота використання: Інтерфейс розроблений із використанням сучасних принципів дизайну, що забезпечує простоту і зрозумілість. Користувачі легко орієнтуються в системі, швидко знаходять потрібні функції та взаємодіють з даними.

Адаптивність: Хоча інтерфейс орієнтований на використання на настільних комп'ютерах, він залишається функціональним і на інших платформах, завдяки простоті дизайну.

Безпека: Система передбачає захист даних користувачів та контроль доступу, що гарантує безпечне збереження та обробку конфіденційної інформації Графічний інтерфейс користувача системи реєстрації звернень громадян є функціональним та зручним для використання. Основні вкладки меню дозволяють користувачам швидко перемикатися між різними розділами системи та працювати з клієнтами, справами, звітами та іншими даними. Простий дизайн та інтуїтивна навігація сприяють ефективній роботі користувачів і забезпечують зручне управління інформацією

2.2.4. Тестування програмного забезпечення та оцінка якості інформаційно-аналітичної системи реєстрації звернень громадян

Тестування програмного забезпечення є одним із найважливіших етапів розробки інформаційно-аналітичної системи реєстрації звернень громадян (ІАС), оскільки дозволяє забезпечити її надійність, стабільність та високий рівень якості. Оскільки система працюватиме з конфіденційною інформацією громадян та повинна бути доступною для численних користувачів, важливо, щоб вона функціонувала без помилок, була швидкою, безпечною і зручною у використанні. Тестування, проведене на різних етапах розробки, дозволяє виявити потенційні проблеми та знизити ймовірність збоїв у роботі системи в реальному середовищі[9].

– Функціональне тестування

Функціональне тестування в межах ІАС зосереджене на перевірці основних операцій системи, таких як реєстрація звернень, відслідковування статусу звернень, формування звітів, а також коректне функціонування різних сервісів у взаємодії з користувачами та адміністраторами. Тестування включає перевірку правильності роботи форм, валідації даних, логіки реєстрації звернень, можливості збереження та оновлення записів у базі даних.

Завдяки юніт-тестам, реалізованим через фреймворк xUnit, виявляються недоліки в логіці обробки даних, що дозволяє на ранніх етапах усунути можливі помилки. Таке тестування гарантує, що кожен компонент системи працює окремо відповідно до вимог.

– Інтеграційне тестування

Інтеграційне тестування для ІАС включає перевірку взаємодії різних модулів системи, таких як взаємодія між інтерфейсами користувачів, серверною частиною та базою даних. Важливо перевірити, чи система правильно обробляє запити від користувачів та чи всі дані коректно зберігаються та відображаються.

Для реалізації цього типу тестування застосовуються методи тестування API та перевірка інтеграції серверної частини ASP.NET Core з базою даних SQL Server, що забезпечує коректну передачу даних між різними елементами системи. Також здійснюється перевірка на обробку помилок, які можуть виникати під час взаємодії різних модулів.

– Тестування продуктивності

ІАС повинна ефективно обробляти велику кількість одночасних запитів від громадян та адекватно реагувати на пік навантажень, наприклад, у періоди високої активності. Тому тестування продуктивності є важливим етапом, на якому оцінюється здатність системи справлятися з великими обсягами даних і великою кількістю користувачів.

Зокрема, проводяться навантажувальні тести з використанням інструменту Apache JMeter, що дозволяє симулювати великий потік одночасних користувачів і перевіряти, як система реагує на пикові навантаження. Метою є виявлення

можливих вузьких місць у архітектурі та забезпечення стабільної роботи системи при максимальних навантаженнях.

– Безпекове тестування

Зважаючи на чутливість інформації, яку система обробляє, безпека є ключовим аспектом тестування. Для оцінки безпеки IAC проводиться перевірка на наявність вразливостей, таких як SQL-ін'єкції, міжсайтові скрипти (XSS), міжсайтові фальшиві запити (CSRF) та інших типових загроз, які можуть поставити під загрозу конфіденційність даних [10].

Використання інструментів для тестування безпеки, таких як OWASP ZAP, дозволяє виявляти вразливості на етапі розробки, що дає змогу оперативно виправити помилки до випуску програми. Також тестуються механізми аутентифікації та авторизації, щоб гарантувати правильний доступ до даних системи.

– Користувацьке тестування (UX/UI тестування)

Однією з важливих складових успіху IAC є забезпечення зручності для користувачів. Оскільки система повинна бути доступною для людей без технічних знань, важливо провести тестування інтерфейсу користувача для перевірки його інтуїтивності та зручності.

UX/UI тестування включає перевірку навігації по інтерфейсу, функціональність форм, а також можливість швидкого доступу до основних функцій. Для цього тестування використовуються методи, що дозволяють проводити зворотний зв'язок з реальними користувачами, щоб переконатися, що інтерфейс задовольняє потреби кінцевих користувачів, таких як адміністратори системи, держслужбовці та громадяни [10].

– Регресійне тестування

Після кожної зміни або доопрацювання в програмному коді проводиться регресійне тестування, щоб переконатися, що нові функціональні можливості не порушили роботу вже існуючих функцій. Регресійне тестування забезпечує стабільність і функціональну цілісність системи протягом усієї розробки та після випуску оновлень.

РОЗДІЛ 3. ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

3.1. Охорона праці

Процес розробки програмного забезпечення для інформаційно-аналітичних систем реєстрації звернень громадян включає не тільки технічні аспекти, але й дотримання вимог охорони праці. Це забезпечує безпечні та комфортні умови праці для співробітників, які працюють за комп'ютерами, та дозволяє уникати негативних наслідків для здоров'я.

Розробники, які тривалий час перебувають за екранними пристроями, піддаються високому фізичному та психологічному навантаженню. Тому важливо дотримуватися нормативних вимог охорони праці, таких як наказ № 207 від 14.02.2018 НПАОП 0.00-7.15-18, який регламентує безпечну організацію робочих місць [11].

Основні аспекти, на які звертається увага під час організації робочого місця:

– Ергономічність: Робоче місце повинно бути облаштоване ергономічними меблями. Стільці мають бути регульованими по висоті та забезпечувати підтримку спини, щоб уникнути проблем зі здоров'ям, таких як біль у спині чи шиї.

– Розташування екрану: Монітори повинні бути на рівні очей, щоб уникати нахилу голови, що може призвести до перенапруги м'язів шиї. Крім того, важливо використовувати монітори з антибліковим покриттям для зменшення навантаження на очі.

– Освітлення: Згідно з ДСанПІН 3.3.2.007-98, освітлення робочого місця повинно бути достатнім, щоб зменшити втому очей, але водночас не надто яскравим, щоб уникнути відблисків на екрані.

– Мікроклімат: Температурний режим на робочому місці має бути комфортним (20-24°C), а рівень вологості повітря – в межах 40-60%. Це дозволяє уникати перегріву або охолодження, що впливає на працездатність.

Не менш важливим є контроль за рівнем електромагнітного випромінювання. Використання сертифікованого обладнання, яке відповідає міжнародним стандартам, дозволяє мінімізувати вплив випромінювання на здоров'я працівників.

Під час тривалої роботи за комп'ютером рекомендується робити регулярні перерви, під час яких працівники можуть виконувати фізичні вправи для зняття напруги з м'язів та покращення кровообігу. Це допоможе зменшити ризик розвитку захворювань опорно-рухового апарату.

Окрім фізичних аспектів, важливо враховувати і психологічні. Нервово-емоційне навантаження, яке виникає під час розробки складних систем, може негативно впливати на ефективність роботи та самопочуття розробників. Створення сприятливого робочого середовища, зокрема оптимізація робочого процесу, командна підтримка та зменшення надмірного навантаження, є ключовими для забезпечення здорового психологічного клімату на роботі.

Дотримання вимог охорони праці та належна організація робочого місця сприяють підвищенню продуктивності працівників та якості виконуваних робіт.

3.2. Безпека в надзвичайних ситуаціях

Пожежна безпека є однією з основних складових забезпечення безпеки працівників і технічного обладнання в офісах та приміщеннях з комп'ютерною технікою. Враховуючи сучасні умови роботи, де комп'ютери, сервери, периферійні пристрої та інша електроніка є важливими елементами робочого процесу, створення належних умов для забезпечення пожежної безпеки є критичним аспектом.

Забезпечення пожежної безпеки в офісах та приміщеннях з комп'ютерною технікою включає комплекс заходів, спрямованих на запобігання виникненню пожеж, а також на швидке та безпечне реагування у випадку надзвичайної ситуації. Основні принципи пожежної безпеки охоплюють кілька аспектів [12].

– По-перше, на етапі проєктування та планування приміщень необхідно враховувати вимоги до протипожежної безпеки, забезпечити правильне розташування виходів та шляхів евакуації, а також забезпечити належну вентиляцію для уникнення перегріву техніки та коротких замикань.

– По-друге, важливо дотримуватись стандартів електричної безпеки: використовувати тільки сертифіковані електричні кабелі, розетки та обладнання, регулярно перевіряти системи на можливі ушкодження, а також забезпечити автоматичні вимикачі для швидкого відключення електропостачання при перевантаженні.

– По-третє, використання протипожежного обладнання: вогнегасників, систем автоматичного пожежогасіння та датчиків диму і температури для оперативного реагування на загрозу пожежі. Нарешті, важливою складовою є підготовка персоналу, що включає навчання співробітників діям під час пожежі, використанню вогнегасників та евакуації, а також проведення регулярних тренувань для підтримки готовності до надзвичайних ситуацій.

Пожежна безпека є однією з основних складових забезпечення безпеки працівників і технічного обладнання в офісах та приміщеннях з комп'ютерною технікою. Враховуючи сучасні умови роботи, де комп'ютери, сервери, периферійні пристрої та інша електроніка є важливими елементами робочого процесу, створення належних умов для забезпечення пожежної безпеки є критичним аспектом.

Пожежна безпека повинна враховувати етапи проєктування та планування приміщень, зокрема правильне розташування виходів та шляхів евакуації, а також забезпечення достатньої вентиляції для електронних пристроїв і кабелів, щоб уникнути перегріву та коротких замикань. Важливим є також забезпечення належної електричної безпеки через використання сертифікованих кабелів і обладнання, регулярний огляд систем та наявність автоматичних вимикачів для запобігання перевантаженням. Крім того, у приміщеннях мають бути встановлені протипожежні вогнегасники та системи автоматичного пожежогасіння, а також системи виявлення диму та температури для своєчасного реагування. Не менш

важливою є підготовка персоналу, що включає навчання правилам пожежної безпеки, евакуації та використання вогнегасників, а також регулярні тренування для підтримки готовності до надзвичайних ситуацій.

Захист інформації у випадку надзвичайних ситуацій є важливою складовою стратегії безпеки, що дозволяє зберегти цілісність, конфіденційність і доступність даних навіть під час кризових подій, таких як стихійні лиха, технічні аварії, кібератаки чи людські помилки. Для забезпечення захисту інформації у таких ситуаціях необхідно впровадити низку заходів, які включають фізичний, кібернетичний захист, а також планування та резервне копіювання даних.

Одним із основних аспектів є резервне копіювання даних, що дозволяє відновити важливу інформацію після її втрати або пошкодження. Регулярне створення резервних копій на фізичних або хмарних носіях гарантує, що в разі аварії дані не будуть безповоротно втрачені. Для цього необхідно забезпечити зберігання копій у безпечних місцях, відокремлених від основних систем, щоб уникнути одночасного знищення даних при техногенних катастрофах чи інших критичних ситуаціях.

Важливою частиною є також регулярне тестування працездатності резервних копій, щоб бути впевненими в їх надійності при необхідності відновлення.

Фізичний захист інформаційних систем має на меті забезпечення безпеки обладнання та даних на рівні приміщення, в якому зберігаються сервери та інші критичні компоненти. Використання захищених приміщень із спеціальними умовами зберігання, таких як клімат-контроль, пожежні системи та системи безперебійного живлення, гарантує стабільність роботи систем навіть за умов технічних збоїв. Наявність систем відеоспостереження та охорони допомагає запобігти несанкціонованому доступу до обладнання, а також контролювати потенційні загрози. Обмеження доступу до критичних зон для неавторизованого персоналу є важливим кроком для запобігання крадіжкам чи навмисним пошкодженням.

Кібербезпека відіграє вирішальну роль у захисті інформації від зовнішніх і внутрішніх загроз. Для цього важливо впроваджувати системи аутентифікації та

авторизації, щоб забезпечити доступ тільки до тих даних, до яких мають право доступу визначені користувачі. Використання шифрування даних забезпечує їх конфіденційність, навіть якщо вони потраплять до рук сторонніх осіб. Системи контролю доступу дозволяють обмежити доступ до критичних даних лише для авторизованого персоналу, що знижує ймовірність витоку чи несанкціонованого використання інформації.

Нарешті, управління ризиками є важливим елементом стратегії захисту інформації. Моніторинг ситуацій і постійний аналіз можливих загроз дозволяють вчасно виявляти потенційні проблеми і оперативно вживати необхідних заходів для їх нейтралізації. Розробка детальних планів реагування на надзвичайні ситуації гарантує, що персонал зможе швидко відновити доступ до даних і систем навіть у разі серйозних аварій або атак.

Комплексний підхід до захисту інформації дозволяє гарантувати її безпеку, навіть коли ситуація виходить з-під контролю, і сприяє ефективному відновленню роботи організації після виникнення надзвичайних подій.

ВИСНОВОКИ

У рамках дипломного проєкту було розроблено інформаційно-аналітичну систему для реєстрації звернень громадян, яка відповідає сучасним вимогам до подібних систем. Основною метою цієї розробки було створення ефективного інструменту для автоматизації процесів обробки звернень, що забезпечує прозорість і зручність взаємодії громадян із організацією. Завдяки застосуванню технологій платформи .NET та впровадженню сучасних методів проєктування, було реалізовано гнучку і надійну систему, що дозволяє швидко обробляти великі обсяги даних та забезпечувати високий рівень безпеки.

У процесі розробки були досягнуті наступні результати:

– Спроєктована та реалізована база даних, яка забезпечує ефективне зберігання та обробку даних про громадян, звернення, операторів та адміністраторів.

– Розроблено інтуїтивно зрозумілий графічний інтерфейс користувача (GUI), який дозволяє швидко орієнтуватися в системі та виконувати необхідні операції з мінімальними зусиллями. Зручна структура інтерфейсу і чітка навігація полегшують роботу користувачів з різним рівнем підготовки.

– Впроваджено основні функції системи, включаючи авторизацію, реєстрацію нових користувачів, пошук даних, обробку звернень, а також генерацію звітів.

– Система забезпечує високий рівень безпеки даних, що дозволяє захищати персональну інформацію користувачів та гарантувати безпечну роботу з конфіденційними даними.

Проєктування та реалізація системи були виконані з урахуванням вимог до продуктивності, масштабованості та адаптивності. Завдяки використанню архітектури MVC та сучасних підходів до розробки програмного забезпечення, система може легко адаптуватися до нових викликів і розширювати свій функціонал у майбутньому. Впровадження цієї системи сприятиме покращенню

якості надання послуг громадянам та підвищенню ефективності роботи операторів і адміністраторів.

ПЕРЕЛІК ПОСИЛАНЬ

1. Джонсон С., Хайслоп П. «ASP.NET Core: Learning the Basics». – Бостон: O'Reilly Media, 2019. – 320 с.

2. Microsoft Docs – ASP.NET Core Documentation [Електронний ресурс]. – Режим доступу: https://docs.microsoft.com/aspnet/core/

3. Фрай С. «Entity Framework Core in Action». – Нью-Йорк: Manning Publications, 2018. – 624 с.

4. Маєвський І. В. «Розробка інформаційних систем: методологія та інструментарій». – Київ: Видавничий дім «Київ», 2021. – 512 с.

5. Інформаційно-аналітична довідка про організацію роботи зі зверненнями громадян у ІІІ кварталі 2023 року [Електронний ресурс]. – Режим доступу: https://diam.gov.ua/zvernennya-gromadyan/informatsiino-analitychna-dovidka-pro-orhanizatsiiu-roboty-zi-zvernenniamy-hromadian-u-iii-kvartali-2023-roku

6. Закон України «Про охорону праці» від 14.10.1992 № 2694-XII. – Відомості Верховної Ради України.

7. M. R. Petryk, I. V. Boyko, O. M. Khimich, and O. Yu. Petryk, “High-performance methods of modeling the adsorption with feedback in heterogeneous multicomponent nanoporous media,” Cybern. Syst. Analysis, Vol. 58, No. 5, 787–805 (2022). https://doi.org/https://doi.org/10.1007/s10559-022-00512-8.

8. Тестування програмного забезпечення, яке його значення. URL: https://www.quality- assurance -group. com/shho - take- testuvannya-programnogozabezpechennya-ta-yake-jogo-znachennya (дата звернення 15.10.2024).

9. ДСТУ 7239:2011 «Системи управління охороною праці. Загальні вимоги». – Київ: Держстандарт України, 2011.

10. Берко А. Ю., Верес О. М., Пасічник В. В. Системи баз даних та знань. Книга 2. Системи управління базами даних та знань : навч. посіб. Львів : Магнолія-2006, 2012. 584 с.

11. . UML – діаграма прецедентів. URL: https://uk.ilovevaquero.com/novosti-iobschestvo/78131-uml-diagramma-precedentov.html (дата звернення 26.08.2024).

12. Бутрин В. С., Діденко О. С. «Охорона праці при роботі з комп'ютерними технологіями». – Київ: Кондор, 2019. – 256 с.

13. Костюк В. М. «Охорона праці в інформаційних системах». – Львів: Видавничий дім «Панорама», 2020. – 240 с.

14. Стручок В.С. Методичний посібник для здобувачів освітнього ступеня «магістр» всіх спеціальностей денної та заочної (дистанційної) форм навчання «БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ» / Тернопіль: ФОП Паляниця В. А., –156 с.

15. Стручок В.С. Навчальний посібник «ТЕХНОЕКОЛОГІЯ ТА ЦИВІЛЬНА БЕЗПЕКА. ЧАСТИНА «ЦИВІЛЬНА БЕЗПЕКА»» / Тернопіль: ФОП Паляниця В. А., – 156 с.

.

ДОДАТОКИ

ДОДАТОК А

Лістиг головної сторінки

```
using Microsoft.AspNetCore.Mvc;
using System.Collections.Generic;
using System.Linq;

public class AppealController : Controller
{
    private static List<Appeal> appeals = new List<Appeal>();

    // GET: /Appeal/
    public IActionResult Index()
    {
        return View(appeals);
    }

    // GET: /Appeal/Create
    public IActionResult Create()
    {
        return View();
    }

    // POST: /Appeal/Create
    [HttpPost]
    public IActionResult Create(Appeal appeal)
    {
        appeal.Id = appeals.Count + 1;
        appeal.DateCreated = DateTime.Now;
        appeal.Status = "Open";
        appeals.Add(appeal);
        return RedirectToAction("Index");
    }

    // GET: /Appeal/Edit/5
    public IActionResult Edit(int id)
    {
        var appeal = appeals.FirstOrDefault(a => a.Id == id);
        if (appeal == null)
        {
            return NotFound();
        }
        return View(appeal);
    }

    // POST: /Appeal/Edit/5
    [HttpPost]
    public IActionResult Edit(Appeal updatedAppeal)
    {
```

```
        var appeal = appeals.FirstOrDefault(a => a.Id ==
updatedAppeal.Id);
        if (appeal == null)
        {
            return NotFound();
        }

        appeal.CitizenName = updatedAppeal.CitizenName;
        appeal.Email = updatedAppeal.Email;
        appeal.Subject = updatedAppeal.Subject;
        appeal.Description = updatedAppeal.Description;
        appeal.Status = updatedAppeal.Status;
        return RedirectToAction("Index");
    }

    // GET: /Appeal/Details/5
    public IActionResult Details(int id)
    {
        var appeal = appeals.FirstOrDefault(a => a.Id == id);
        if (appeal == null)
        {
            return NotFound();
        }
        return View(appeal);
    }
}
@model Appeal

<h2>Create New Appeal</h2>

<form asp-action="Create" method="post">
    <div class="form-group">
        <label for="CitizenName">Citizen Name</label>
        <input type="text" name="CitizenName" class="form-control"
/>
    </div>
    <div class="form-group">
        <label for="Email">Email</label>
        <input type="email" name="Email" class="form-control" />
    </div>
    <div class="form-group">
        <label for="Subject">Subject</label>
        <input type="text" name="Subject" class="form-control" />
    </div>
    <div class="form-group">
        <label for="Description">Description</label>
        <textarea name="Description" class="form-
control"></textarea>
    </div>
    <button type="submit" class="btn btn-primary">Submit</button>
</form>
@model IEnumerable<Appeal>
```

```
<h2>List of Appeals</h2>

<table class="table">
    <thead>
        <tr>
            <th>ID</th>
            <th>Citizen Name</th>
            <th>Subject</th>
            <th>Status</th>
            <th>Actions</th>
        </tr>
    </thead>
    <tbody>
        @foreach (var appeal in Model)
        {
            <tr>
                <td>@appeal.Id</td>
                <td>@appeal.CitizenName</td>
                <td>@appeal.Subject</td>
                <td>@appeal.Status</td>
                <td>
                    <a asp-action="Edit" asp-route-id="@appeal.Id"
class="btn btn-secondary">Edit</a>
                    <a asp-action="Details" asp-route-
id="@appeal.Id" class="btn btn-info">View</a>
                </td>
            </tr>
        }
    </tbody>
</table>
public class Appeal
{
    public int Id { get; set; }
    public string CitizenName { get; set; }
    public string Email { get; set; }
    public string Subject { get; set; }
    public string Description { get; set; }
    public DateTime DateCreated { get; set; }
    public string Status { get; set; }  // Open, In Progress, Closed
    public int UserId { get; set; }  // Foreign Key linking to User
}
using Microsoft.AspNetCore.Mvc;
using System.Collections.Generic;
using System.Linq;

public class UserController : Controller
{
    private static List<User> users = new List<User>();

    // GET: /User/Login
    public IActionResult Login()
    {
        return View();
```

```
    }

    // POST: /User/Login
    [HttpPost]
    public IActionResult Login(string username, string password)
    {
        var user = users.FirstOrDefault(u => u.Username == username
&& u.Password == password);
        if (user != null)
        {
            // Set session or cookie here
            return RedirectToAction("Index", "Appeal");
        }
        ModelState.AddModelError("", "Invalid login attempt.");
        return View();
    }

    // GET: /User/Register
    public IActionResult Register()
    {
        return View();
    }

    // POST: /User/Register
    [HttpPost]
    public IActionResult Register(User user)
    {
        if (users.Any(u => u.Username == user.Username))
        {
            ModelState.AddModelError("", "Username already
exists.");
            return View();
        }
        user.Id = users.Count + 1;
        users.Add(user);
        return RedirectToAction("Login");
    }
}
using Microsoft.AspNetCore.Mvc;
using System.Collections.Generic;
using System.Linq;

public class AppealController : Controller
{
    private static List<Appeal> appeals = new List<Appeal>();

    // GET: /Appeal/
    public IActionResult Index()
    {
        return View(appeals);
    }

    // GET: /Appeal/Create
```

```
    public IActionResult Create()
    {
        return View();
    }

    // POST: /Appeal/Create
    [HttpPost]
    public IActionResult Create(Appeal appeal)
    {
        appeal.Id = appeals.Count + 1;
        appeal.DateCreated = DateTime.Now;
        appeal.Status = "Open";
        appeals.Add(appeal);
        return RedirectToAction("Index");
    }

    // GET: /Appeal/Edit/5
    public IActionResult Edit(int id)
    {
        var appeal = appeals.FirstOrDefault(a => a.Id == id);
        if (appeal == null)
        {
            return NotFound();
        }
        return View(appeal);
    }

    // POST: /Appeal/Edit/5
    [HttpPost]
    public IActionResult Edit(Appeal updatedAppeal)
    {
        var appeal = appeals.FirstOrDefault(a => a.Id ==
updatedAppeal.Id);
        if (appeal == null)
        {
            return NotFound();
        }

        appeal.CitizenName = updatedAppeal.CitizenName;
        appeal.Email = updatedAppeal.Email;
        appeal.Subject = updatedAppeal.Subject;
        appeal.Description = updatedAppeal.Description;
        appeal.Status = updatedAppeal.Status;
        return RedirectToAction("Index");
    }
}
<h2>Login</h2>

<form asp-action="Login" method="post">
    <div class="form-group">
        <label for="Username">Username</label>
        <input type="text" name="username" class="form-control" />
    </div>
```

```
    <div class="form-group">
        <label for="Password">Password</label>
        <input type="password" name="password" class="form-control"
/>
    </div>
    <button type="submit" class="btn btn-primary">Login</button>
</form>
```

ДОДАТОК Б

Тези конференції

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
імені ІВАНА ПУЛЮЯ
НАУКОВЕ ТОВАРИСТВО ім. ШЕВЧЕНКА

М А Т Е Р І А Л И

XII НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

«ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ»

18–19 грудня 2024 року

ТЕРНОПІЛЬ
2024

УДК 004.4
В. Сарновський; Ю. Стоянов, к.т.н., доц.
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

ПРОЄКТУВАННЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ УПРАВЛІННЯ ЗАЯВКАМИ ГРОМАДЯН З ВИКОРИСТАННЯМ ТЕХНОЛОГІЙ .NET

UDC 004.4
V. Sarnovskyi; Y. Stoyanov, Ph.D., Assoc. Prof.

DESIGNING AN AUTOMATED SYSTEM FOR MANAGING CITIZENS' APPLICATIONS USING .NET TECHNOLOGIES

Державні установи щодня отримують значний обсяг звернень громадян, які потребують своєчасної та ефективної обробки. Традиційні методи роботи зі зверненнями мають низку недоліків, зокрема ручну обробку даних і відсутність сучасних інструментів для їх аналізу. Дослідження присвячене розробці автоматизованої системи для управління заявками громадян, яка базується на технологіях .NET і забезпечує підвищення продуктивності роботи відповідних служб.

Створення системи, що автоматизує процеси реєстрації, обробки та зберігання звернень громадян, з можливістю аналізу інформації та інтеграції з іншими державними платформами. Така система повинна бути зручною для користувачів та забезпечувати ефективну взаємодію.

Для реалізації проєкту використовується платформа .NET, яка забезпечує розробку масштабованих і надійних рішень. База даних на основі Microsoft SQL Server надає можливість зберігати великі обсяги даних та швидко їх обробляти. Інтерфейс користувача створений за допомогою ASP.NET MVC, що забезпечує зручність роботи з системою. Для взаємодії з базою даних використовується Entity Framework, що полегшує управління дани.

Система надає можливість громадянам подавати заявки онлайн і контролювати їх статус. Автоматизація процесів дозволяє знизити навантаження на персонал державних установ, прискорити реагування на запити та покращити якість надання послугів.

Література

1. Troelsen, A., Japikse, P. Pro C# 8.0 and the .NET Core 3.0 Framework. Apress, 2020.
2. Soni, M., Thakkar, P., Patel, H. ASP.NET MVC with Entity Framework and CSS. BPB Publications, 2021.

ДОДАТОК В

Диск з роботою