m-math_2025_020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРНІВЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ЮРІЯ ФЕДЬКОВИЧА

Факультет математики та інформатики
кафедра математичного моделювання

Автоматизація процесу замовлення та видачі документів в великих організаціях

Кваліфікаційна робота

Рівень вищої освіти – другий (магістерський)

Виконав:
студент 2 курсу, 601 групи
Шелеста Михайла Віталійовича
Керівник:
кандидат технічних наук,
Шкільнюк Д.В.

До захисту допущено
на засіданні кафедри
протокол № 5 від 27 листопада 2024 р.
Зав. кафедрою _____ проф. Черевко І.М.

Чернівці – 2024

АНОТАЦІЯ

Розроблено веб-додаток системи видачі та зберігання документів з використанням C#, ASP.NET, Entity Framework, HTML, CSS, Angular та MySQL.

Реалізовано рольовий доступ до застосунку, де користувач має змогу отримати необхідні документи, працівник, відповідно до свого відділу, опрацьовувати їх, а адміністратор керувати правами доступу вищезгаданих.

Ключові слова:

Веб-додаток, документи, адміністратор, користувач, ролі, авторизація, .NET, HTML, CSS, Angular, MySQL, JWT

ABSTRACT

Developed a web application for document issuing and storage system using C#, ASP.NET, Entity Framework, HTML, CSS, Angular, and MySQL.

The application features role-based access control where the user can request the necessary documents, the employee can process them according to their department, and the administrator manages the access rights of the aforementioned roles.

Keywords:

Web application, documents, administrator, user, roles, authorization, .NET, HTML, CSS, Angular, MySQL, JWT

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів наукових досліджень інших авторів мають посилання на відповідне джерело.

Зміст

Вступ

Актуальність роботи. У сучасному діджиталізованому світі стрімко зростає кількість інформації, яку зручно обробляти та зберігати в електронному вигляді. З розвитком інформаційних технологій виникає потреба в автоматизованих системах, які забезпечують зручний і швидкий доступ до документів. Особливо важливим це стає в державних установах, університетах, банківській сфері, і у багатьох інших організаціях, де зберігаються значні обсяги документів. Традиційні методи пошуку та зберігання документів вже не відповідають вимогам сучасності, що зумовлює необхідність створення вбудованих інформаційних систем для запиту документів.

Реалізована система запиту та управління документами дозволяє значно скоротити час на пошук інформації, підвищити ефективність роботи співробітників та дозволити збереження документів. Це особливо важливо в умовах зростання обсягу даних і необхідності дотримання нормативних вимог щодо збереження інформації. Таким чином, розробка системи запиту та керування документами є актуальною задачею, яка має значний потенціал для впровадження у різні галузі бізнесу та державного управління.

Метою роботи є розробка системи запиту документів, яка забезпечить ефективне зберігання, управління та пошук документів відповідно до вимог користувачів, працівників та адміністраторів системи. Для досягнення цієї мети необхідно вирішити низку завдань:

- Аналіз існуючих систем зберігання та запиту документів, їх переваг та недоліків.
- Реалізація функціоналу для різних категорій користувачів: користувача, працівника та адміністратора.
- Створення інтуїтивно зрозумілого інтерфейсу, побудованого за допомогою Angular, який дозволить користувачам легко знаходити та

запитувати документи, а адміністраторам та працівникам керувати ними.

- Розробка серверної частини на основі ASP.NET з використанням Entity Framework для забезпечення надійного збереження даних та виконання бізнес-логіки.
- Забезпечення безпеки даних та доступу до документів відповідно до прав користувачів з використанням Identity Framework.

Об’єктом дослідження є інформаційні системи для зберігання та запиту документів. Предметом дослідження виступають методи і технології розробки програмних систем, що забезпечують ефективне управління документами, їх пошук і запит у відповідності до ролей користувачів (користувач, працівник, адміністратор).

У роботі використовуються такі **методи дослідження**:

- Аналіз – для вивчення існуючих рішень та визначення основних вимог до розроблюваної системи.
- Програмування – для реалізації клієнтської частини на Angular та серверної частини на ASP.NET Core і MySQL як бази даних.

Наукова частина роботи полягає в розробці системи, що використовує сучасні гнучкі технології веб-розробки, для управління правами доступу користувачів. Використання Angular у якості front-end дозволяє створити динамічний і швидкий інтерфейс SPA додатку, тоді як ASP.NET, як back-end, забезпечує високу продуктивність та масштабованість системи. Використання Entity Framework дозволяє зручно працювати з базою даних MySQL, забезпечуючи швидкий і безпечний доступ до інформації, абстрагований від звичайного використання SQL.

Практичне значення роботи полягає у можливості застосування розробленої системи в організаціях, де є потреба в автоматизації процесу

зберігання та пошуку документів. Система може бути легко адаптована під різні потреби замовників, що робить її універсальним рішенням для організацій.

Розділ 1. Огляд проблем, вирішених класифікаційною роботою

1.1 Проблематика

У сучасних організаціях проблема ефективного управління документами стає все більш актуальною. Зі збільшенням обсягів інформації та документів, які зберігаються в електронному вигляді, виникає необхідність у створенні надійних і зручних систем для управління цими даними. Традиційні методи роботи з документами, такі як паперові архіви або прості файлові системи, вже не відповідають вимогам сучасного ринку. Це зумовлює необхідність впровадження систем, що дозволяють автоматизувати процеси збереження, пошуку та обробки документів.

Основні проблеми, з якими стикаються організації при роботі з документами:

- **Зростаючий обсяг даних:** З кожним роком кількість документів збільшується, що ускладнює процес їх зберігання та пошуку.
- **Недостатня систематизація:** Відсутність чіткої структури та організації документів призводить до втрат часу на їх пошук та обробку.
- **Проблеми з доступністю:** Користувачі часто не мають зручного інструменту для швидкого доступу до необхідних документів, що знижує ефективність роботи.
- **Безпека даних:** Збереження конфіденційних документів потребує додаткових заходів безпеки, зокрема контролю доступу та захисту від несанкціонованого використання.

Розробка системи запиту документів дозволяє вирішити ці проблеми шляхом автоматизації процесів управління, що забезпечує швидкий доступ до даних, їх безпеку та зручність використання для кінцевих користувачів.

1.2. Огляд існуючих систем управління документами

Поки що не існує багато прикладів побудованої в ході цього проєкту системи, оскільки створення таких рішень потребує великих затрат і високого рівня

безпеки. Однак, є декілька додатків, розповсюджених на території певних країни, і розроблених за підтримки їхнього уряду, наприклад:

- **Дія** — застосунок, розроблений Міністерством цифрової трансформації України. У додатку на високому рівні запрограмована велика кількість корисного функціоналу, а саме мова йде про можливість перегляду документів користувача, прив'язаних до певної системи. Подібно нашій розробці, якщо користувачу було видано державним органом або іншою офіційною установою документ або посвідчення (прикладом державних можна назвати паспорти, свідоцтва, а іншою установою — студентський квиток, посвідчення водія), він має повний доступ до неї у цифровому форматі.

  Дія стрімко розвивається, і за невеликий проміжок часу, окрім документів, у ній почала з'являтися вся офіційна інформація про зареєстровану людину, у тому числі судимості та штрафи. Подібно до реалізованої нами системи, у Дії можна подати заявку на отримання витягів та довідок, таких як «Витяг про місце проживання» або «Витяг про несудимість», які, після обробки запиту, стануть доступними для завантаження користувачеві.

- **E-Estonia** — це платформа уряду Естонії, яка стала однією з найбільш розвинених систем електронного урядування у світі. Ключовим елементом є електронне посвідчення особи — e-ID. За допомогою e-ID або мобільної версії Smart-ID користувачі мають доступ до своїх особистих документів, таких як паспорт, посвідчення водія, медичні картки та інформація про власність, через різні електронні сервіси.

  Документи та інформація зберігаються у децентралізованих базах даних, об'єднаних технологією X-Road. Це забезпечує безпечний доступ та обмін даними між різними державними установами, при цьому користувач може перевіряти, хто отримував доступ до його документів. Крім того, через систему e-Estonia можна замовити витяги та сертифікати, що схоже на

функціонал української «Дії», але з більшим охопленням послуг та інтеграцією з усіма державними структурами.

Розділ 2. Використанні технології

2.1. Система управління базами даних СУБД

MySQL — це одна з найпопулярніших систем управління реляційними базами даних (**СУБД**) з відкритим вихідним кодом, яка підходить для використанні в різних проєктах: від малих додатків до великих корпоративних систем. Дана СУБД підтримується корпорацією Oracle, та чудовою її особливістю є можливість безкоштовного використання програми, перегляду відкритого вихідного коду, а також наявності комерційних ліцензій для підприємств, які потребують розширеної підтримки.

Вона базується на моделі «клієнт-сервер», де сервер, тобто MySQL, виконує всі операції з базою даних, а клієнтські програми звертаються до нього через SQL. Архітектура «клієнт-сервер» дозволяє ефективно зберігати дані централізовано, а також забезпечує зручний доступ як з локальних, так і з віддалених клієнтів через локальну мережу або Інтернет. Це робить MySQL ефективним вибором для проєктів різного масштабу завдяки високій продуктивності, надійності та доступності. MySQL працює на багатьох операційних системах, включаючи Windows, Linux, macOS, що є великим плюсом у командній розробці, оскільки дозволяє використовувати різні платформи.

Дана СУБД використовує реляційну модель зберігання даних, де інформація організована в таблиці з рядками та стовпцями. Таблиці можуть бути пов’язані між собою за допомогою первинних і зовнішніх ключів, що забезпечує логічну структуру даних і підтримує цілісність. Це особливо зручно при розробці додатків, де дані, такі як інформація про користувачів та документи, повинні бути взаємопов’язані. Крім того, MySQL підтримує складні SQL-запити для отримання потрібних даних, що дозволяє ефективно маніпулювати великими обсягами інформації.

MySQL є легко масштабованою СУБД, що дозволяє обробляти великі обсяги даних і витримувати високі навантаження. Завдяки функціям реплікації, шардінгу та кластеризації, MySQL може створювати розподілені системи, які здатні обробляти тисячі запитів одночасно. Реплікація дозволяє дублювати дані на декілька серверів, що підвищує надійність і доступність системи. Шардінг ж розподіляє дані між різними серверами, дозволяючи зменшити навантаження і підвищити продуктивність. Кластеризація забезпечує високу доступність, автоматично розподіляючи запити між серверами в кластері.

MySQL підтримує транзакції та принципи ACID (***A**tomictity (атомарність), **C**onsistency (постійністю), **I**solation (ізольованість), **D**urability (довговічність)*), що гарантує цілісність даних навіть у випадку збоїв. Це критично важливо для додатків, де потрібно забезпечити точність і надійність даних, таких як фінансові системи. Важливою складовою цих принципів є транзакції, які дозволяють групувати декілька SQL-запитів у одну операцію, що забезпечує виконання всіх дій цілком або не виконання їх зовсім у разі помилки. Наприклад, використання транзакцій не дозволить у банківській структурі зробити окремими операціями відправлення та надходження коштів, оскільки у такій ситуації гроші можуть просто втратитися внаслідок раптової помилки.

Однією з найважливіших функцій для підвищення ефективності MySQL є використання індексів. Індекси створюються на стовпцях таблиць і значно зменшують час пошуку даних, що важливо при виконанні складних запитів, таких як пошук або сортування. Індекси зменшують кількість рядків, які потрібно перевірити, тим самим підвищуючи ефективність запитів. Існує багато видів індексів, і кожен підходить під певну ситуацію. Найпопулярнішими виборами є кластерні та хеш індекси. Однак, ними не варто зловживати і завжди треба брати до уваги, що створення великої кількості індексів може уповільнити операції вставки, оновлення та видалення даних, тому найкраще їх застосування є таблицях, які найрідше зазнають описаних операцій і є масштабними.

MySQL має розвинену систему безпеки, що забезпечує захист даних від несанкціонованого доступу. Вона включає шифрування даних як у стані спокою, так і під час передачі через SSL/TLS, що захищає інформацію від перехоплення. Управління доступом на основі ролей дозволяє адміністраторам задавати права доступу для різних користувачів, що забезпечує додатковий рівень безпеки. Автентифікація користувачів може здійснюватися різними методами, включаючи інтеграцію з системами керування ідентифікацією, такими як LDAP.

Широка екосистема інструментів MySQL включає MySQL Workbench та MySQL Shell, які значно спрощують адміністрування та розробку баз даних. MySQL Workbench надає зручний графічний інтерфейс для проектування баз даних, створення запитів і аналізу продуктивності. Це особливо корисно для розробників і адміністраторів, які хочуть швидко проєктувати структури бази даних і оптимізувати запити.

2.2. Мова програмування C# та платформа .NET

C# (*Сі шарп*) — це сучасна, об'єктно-орієнтована мова програмування, розроблена компанією Microsoft, яка використовується для створення різноманітних додатків на платформі .NET. Мова C# має строгу типізацію, яка дозволяє уникати багатьох типових помилок у програмуванні.

Розробники C# на початку орієнтувалися на популярність мов програмування Java та C++, і, подібно ним, C# також є «C-подібною» мовою. Подібно Java, C# дозволяє розробникам забути про турботи керування пам’яттю, маючи вбудований **Garbage Collector** (дослівно *збирач сміття*), який бере на себе усю згадану мороку, забезпечуючи продуктивність та зручність розробки. C# є популярним вибором для розробників через її зручний синтаксис, підтримку асинхронного програмування, а також широку екосистему бібліотек і фреймворків.

Початково, C# була частиною платформи **.NET Framework**, яка була випущена Microsoft у 2002 році. .NET Framework забезпечувала розробку Windows-

додатків, підтримуючи широкий спектр технологій, таких як Windows Forms, WPF (Windows Presentation Foundation) та ASP.NET для веб-додатків. Однак, вона мала значний недолік — прив'язаність до операційної системи Windows, що обмежувало можливості крос-платформного використання.

Через це у 2016 році був випущений фреймворк **.NET Core**, який приніс суттєві зміни, а саме:

- **Крос-платформність:** виправивши згаданий вище недолік, .NET Core працює на Windows, macOS та Linux, дозволяючи розробникам створювати додатки, які можуть запускатися на будь-якій з цих платформ.
- **Відкритий вихідний код:** .NET Core став відкритим проєктом на GitHub, що дозволило спільноті розробників брати участь у його розвитку.
- **Висока продуктивність:** .NET Core був оптимізований для кращої продуктивності, включаючи покращення роботи з асинхронними запитами та зменшення споживання ресурсів.

У 2020 році Microsoft випустила .NET 5, що об'єднала .NET Framework та .NET Core в єдину платформу, відому просто як .NET. Це дозволило спростити розробку і дало можливість користуватися найновішими особливостями, незалежно від обраної операційної системи.

C# та .NET мають багато корисних функцій, які допомагають спростити розробку та підвищити продуктивність програмістів. Деякі з них включають:

- **LINQ (Language Integrated Query):** LINQ дозволяє виконувати запити до колекцій даних (масивів, списків, та навіть баз даних) за допомогою зручного синтаксису, надзвичайно подібного до SQL. Це робить код більш читабельним і зручним для обробки даних.
- **Асинхронне програмування:** Асинхронне програмування стало однією з революцій, яку приніс C# у світ програмування. Він дозволяє виконувати довготривалі операції (наприклад, звернення до бази даних або веб-запити, зчитування з файлу) без блокування основного потоку. Це забезпечує

швидшу відповідь додатків та кращу користувацьку взаємодію. Цей функціонал показав себе настільки добре, що наразі його використовують майже в усіх відомих мовах програмування.

- **Автоматичні властивості:** Це спрощений спосіб створення геттерів та сеттерів для властивостей класу без необхідності явно оголошувати приватне поле. Це рятує від великої зайвої писанини, яка необхідна в уже згаданих Java та C++.

Однією з ключових причин популярності платформи .NET є її висока продуктивність та можливість легко масштабувати додатки. .NET включає Just-In-Time та Ahead-Of-Time компіляцію, що дозволяє програмам працювати швидше, особливо у сценаріях високих навантажень. Крім того, підтримка вже згаданого Garbage Collector'а знижує ризик витоків пам'яті, автоматично звільняючи непотрібні об'єкти.

C# та .NET облегшують роботу ще й завдяки наявності розвиненої екосистеми інструментів, таких як:

- **Visual Studio та Visual Studio Code**: потужні IDE від Microsoft з підтримкою режима відладки (дебагінгу), аналізу коду, пам'яті, та інтеграцією з Git.
- **.NET CLI (Command Line Interface):** інструмент командного рядка для створення, збору та розгортання проєктів на .NET, що особливо корисно в середовищах без графічного інтерфейсу.
- **NuGet Package Manager:** Система керування пакетами, що дозволяє легко додавати сторонні бібліотеки та фреймворки до проєктів.

2.3. Фреймворк ASP.NET

ASP.NET — це фреймворк для створення веб-додатків на згаданій платформі .NET. Він був так само розроблений компанією Microsoft для спрощення процесу створення динамічних веб-сайтів, API та сервісів. З часів переходу на ASP.NET Core (який зараз і є нам відомим ASP.NET), ASP.NET підтримує крос-

платформну розробку, що робить його універсальним вибором для розробників, які працюють із різними операційними системами.

ASP.NET пропонує кілька підходів для створення веб-додатків, включаючи:

- **ASP.NET Web Forms** — перший і класичний підхід до розробки веб-додатків в ASP.NET, створений ще за часів .NET Framework. Він основується на подіях і дуже схожий за принципом роботи на Windows Forms для комп’ютерних додатків. Web Forms дозволяє створювати веб-додатки, використовуючи концепцію drag-and-drop (тягни і кидай) у Visual Studio, де елементи інтерфейсу додаються в дизайн і автоматично зв'язуються з серверним кодом.

  Незважаючи на простоту використання, Web Forms має обмеження у масштабуванні та складності підтримки, особливо для великих додатків із динамічним інтерфейсом. Крім того, використання ViewState, однієї з особливостей фреймворку, може збільшувати розмір сторінки, що впливає на продуктивність. Через це у сьогоденні Web Forms здебільшого використовуються лишень для підтримки legacy-додатків.
- **ASP.NET MVC (Model-View-Controller)** був представлений як альтернатива Web Forms для розробників, які прагнуть більшого контролю над архітектурою та поведінкою додатків. MVC дозволяє створювати веб-додатки на основі популярного шаблону **«Model-View-Controller»** (модель (сутність, таблиця), представлення (вигляд, інтерфейс), контролер (логіка)). Серед його особливостей:
  - **Розділення логіки та інтерфейсу:** Згаданий принцип MVC, який реалізує популярну в програмуванні концепцію **Separation of Duties** (розділення відповідальностей), де кожна складова програми має власну, незалежну від інших логіку, що дозволяє писати більш гнучкий та масштабований код. Це також є перевагою для розробників, які намагаються притримуватися принципів розробки

Test-Driven Development, роблячи код легшим для написання юніт-тестів завдяки розділенню відповідальностей.

- **Гнучкість маршрутизації:** Підтримує налаштування користувацьких маршрутів, що дозволяє створювати більш читабельні URL-адреси.
- **Використання Razor Pages:** Razor Pages — це спрощений підхід до створення сторінок, який є частиною MVC, але не вимагає окремого контролера для кожної сторінки. Razor Pages дозволяє створювати сторінки з логікою безпосередньо в одному файлі, де поєднуються HTML та C# код. Це робить розробку менш громіздкою, особливо для простих сторінок.

Підхід з використанням ASP.NET MVC не має таких явних недоліків, як із застарілим фреймворком ASP.NET Web Forms. Усі проблеми можна вирішити з використанням коду, а спроєктувати додаток можна вже за допомогою зручної розробнику архітектури.

- **ASP.NET Web API** був спеціально розроблений для створення HTTP-сервісів, що відповідають принципам REST архітектури. Це дозволяє легко створювати веб-сервіси, які можуть обробляти HTTP-запити від клієнтів, таких як браузери, мобільні додатки та інші сервери. ASP.NET Web API є чудовим вибором для побудови back-end'у, який надає дані через API у форматі JSON або XML. Серед його особливостей:

  - **Підтримка HTTP-методів:** Дозволяє легко працювати з будь-якими HTTP-методами (GET, POST, PUT, DELETE). Цей факт, знову ж таки, робить Web API ідеальним вибором для побудови RESTful (тих, що притримуються принципів REST) сервісів.
  - **Маршрутизація через атрибути:** Web API використовує атрибути для налаштування маршрутів для запитів, що надходять на сервер, що

дозволяє гнучко та швидко налаштовувати обробку запитів. Це дає змогу визначати маршрути безпосередньо в коді контролера, а не в конфігураційних файлах

- **Автоматична серіалізація даних**: при відповіді Web API автоматично серіалізує готові до відправлення об’єкти у формат JSON або XML, в залежності від заголовків запиту. Це дозволяє уникнути зайвої писанини коду при обробці відповіді.

Основний виклик полягає у безпеці, оскільки відкритий API може бути вразливим до атак, таких як DDoS або несанкціонований доступ, що потребує додаткових заходів безпеки.

2.4. Entity Framework

Entity Framework — це бібліотека для.NET, розроблена Microsoft як ORM (Object-Relational Mapping). Її мета — спростити роботу з базами даних, дозволяючи розробникам абстрагуватися від написання та продумування складних запитів SQL, працюючи з даними у вигляді об’єктів класу. Entity Framework автоматично генерує SQL-запити та виконує їх, що полегшує доступ до даних, управління ними та їх моделювання.

Оскільки це значно прискорює розробку додатків, Entity Framework є надзвичайно популярним для використання для проєктів будь-яких масштабів. Кожного року виходять оновлення, які певним чином поліпшують згенеровані Entity Framework’ом запити. Тим не менш, у проєктах з дуже великою кількістю даних та складною логікою Entity Framework може у певній мірі приторможувати роботу, генеруючи доволі затратні запити, через що розробники вдаються до більш низькорівневих аналогів Entity Framework, таких як Dapper, проте це рідкість і популярність саме Entity Framework серед .NET розробників важко переоцінити.

Entity Framework підтримує як Code-First, так і Database-First підходи для створення бази даних. При Database-First, ми з самого початку проєктуємо нашу

базу даних одразу на якомусь із SQL середовищ, наприклад, в уже згаданому MySQL. Коли нам це буде потрібно, ми можемо вказати, щоб Entity Framework переписав усі таблиці у відповідний код в C#. При Code-First, спочатку будуються класи, а потім Entity Framework на основі цих даних створює таблиці в базі даних.

Класи, які представляють таблиці в C#, називаються сутностям (в англ. *Entity*, від чого і назва самої ORM). Кожен об'єкт такого класу представляє окрему таблицю, а властивості класу представляють рядки таблиці.

Щоб мати можливість повністю абстрагуватися від прямої взаємодії з базами даних, Entity Framework також дозволяє створювати обмеження або спеціальні перевірки для полів бази даних. Наприклад, розглянемо наступний клас:

```
class User
{
    [Key]
    public int Id { get; set; }
    [Required]
    public string Name { get; set; }
    [EmailAddress]
    public string Email { get; set; }
    [MaxLength(100)]
    public string PhoneNumber { get; set; }
}
```

Ось що він означатиме, коли буде переведений на «мову бази даних»:

- Буде створена таблиця User
- **Атрибут *Key*** вказує на те, що поле Id має бути головним ключем у таблиці
- **Атрибут *Required*** вказує на Not Null властивість, тобто поле не може бути порожнім
- **Атрибут *EmailAddress*** робить автоматичну перевірку на валідність електронної адреси, яку намагаються добавити до таблиці

- **Атрибут *MaxLength*** вказує на максимальну кількість допустимих символів у рядку, який буде переданий для цієї властивості. Подібно до нього також існує атрибут ***MinLength***

Крім них, існує і багато інших властивостей. Також існує можливість для створення зовнішніх ключів. Окрім атрибутів, також можна створити окремий клас, який буде окремо описувати властивості полів та їхні зв’язки.

Всі дані, які має знати Entity Framework для взаємодії з базою даних, у тому числі таблиці та конфігурації, вказані у класі, який має успадковуватися від **DbContext** — головного об’єкту фреймворку. Він керує підключенням до бази даних, відстеженням змін сутностей та виконанням запитів.

Entity Framework використовує увесь потенціал уже згаданої особливості C# та .NET — LINQ. Оскільки LINQ запроваджує синтаксис, максимально наближений до SQL, але у той же час спрощений, Entity Framework є дуже зручним для людей, які ніколи раніше з ним не працювали, але мають певний доступ у розробці. Це є перевагою ще й тому, що сам синтаксис SQL дуже легко читається для будь-якої людини, яка хоча б на мінімальному рівні знайома з англійською мовою.

Entity Framework підтримує міграції, які дозволяють легко змінювати структуру бази даних у процесі розробки. Це особливо корисно при використанні підходу Code-First, коли база даних створюється на основі класів C#. У міграціях зберігаються всі зміни, які варто застосувати до бази даних з моменту останнього оновлення. Список змін і команди для них генеруються самим Entity Framework. До бази даних можна застосувати як одну міграцію, так і одразу декілька. Також можна відмінити дії певних міграцій, оскільки у самих міграціях, окрім коду для додавання змін, написаний протилежний код, який анулює внесені зміни.

2.5. Identity Framework

ASP.NET Identity Framework — це бібліотека для керування аутентифікацією та авторизацією у додатках, розроблених .NET. Вона

забезпечує розробникам готову інфраструктуру для роботи з користувачами, ролями та правами доступу, що значно спрощує реалізацію функціональності входу, реєстрації та керування сесіями.

Можна виокремити декілька особливостей та зручностей використання Identity Framework для авторизації та автентифікації замість розробки усього самому:

- **Робота з користувачами та ролями:** Identity Framework має вбудований функціонал для легкого створення, редагування та облікових записів користувачів, а також керування їхніми ролями.
- **Хешування паролів:** Identity використовує безпечне хешування паролів, що захищає дані користувачів у разі витоку бази даних.
- **Підтримка двофакторної аутентифікації:** Додає додатковий рівень захисту шляхом використання SMS або автентифікаційних додатків.
- **Інтеграція з соціальними логінами:** Identity Framework підтримує вхід через Google, Facebook, Microsoft та інші сторонні сервіси за допомогою OAuth.

2.6. JWT Токен

Identity Framewok вирішує проблему прав користувача, однак, при взаємодії між додатками треба мати підтвердження того, що користувач є тим, за кого себе видає, і цю проблему вирішує **JWT** (JSON Web Token), який є стандартом для передачі даних у вигляді JSON-токенів між клієнтом та сервером. Токени містять закодовану інформацію про користувача, яка використовується для аутентифікації. Використання JWT замінює класичні сесії та кукі, що робить його популярним вибором для використання.

JWT кодуються та розкодовуються за допомогою алгоритмів, таких як:

- **HMAC SHA256:** Симетричний алгоритм, який використовує один секретний ключ для підпису та перевірки токена. Його легко налаштувати, але секретний ключ повинен бути захищеним.
- **RSA або ECDSA:** Асиметричні алгоритми, які використовують пару ключів (приватний і публічний). Приватний ключ використовується для підпису, а публічний — для перевірки. Це підходить для систем, де сервери мають різні рівні доступу до ключів.

Токен містить усю необхідну інформацію про користувача, яку можна використовувати для аутентифікації та авторизації без додаткових звернень до бази даних. JWT складається з трьох частин, які разом містять закодовану інформацію, необхідну для визначення користувача, серед них:

- **Header (заголовок):** Містить інформацію про тип токена та алгоритм підпису.
- **Payload (навантаження):** Основна частина токена, де зберігаються клейми (іншими словами, данні, що передаються) користувача. Payload може містити стандартні та користувацькі клейми:
    - **Registered claims**: Стандартні поля, визначені специфікацією JWT, такі як sub (ідентифікатор користувача), exp (час закінчення дії токена).
    - **Public claims:** Користувацькі поля, які можуть бути додані для передачі додаткових даних, наприклад, ролі користувача або його електронна пошта.
    - **Private claims:** Специфічні для додатку поля, які використовуються лише всередині системи.

  За замовчуванням JWT не шифрує payload — лише підписує його. Це означає, що дані в payload можна прочитати, якщо токен декодувати. Якщо в токені передаються чутливі дані, слід використовувати додаткове шифрування за допомогою JWE (JSON Web Encryption).

- **Signature (підпис):** Забезпечує захист токена від підробки. Він створюється за допомогою алгоритму підпису. Підпис гарантує, що дані в токені не були змінені, оскільки для перевірки підпису використовується секретний ключ.

Для безпечної передачі даних через JWT токен рекомендується використовувати протокол HTTPS. Для передачі токена серверу, токен слід вказати у заголовку запиту під назвою Authorization, а перед самим токеном добавити слово «Bearer». Тобто в результаті хедер Authorization буде мати такий вигляд: «Bearer *токен*», де токен — переданий токен.

На рис. 1.1 наведений приклад токену та його розшифрованих даних.

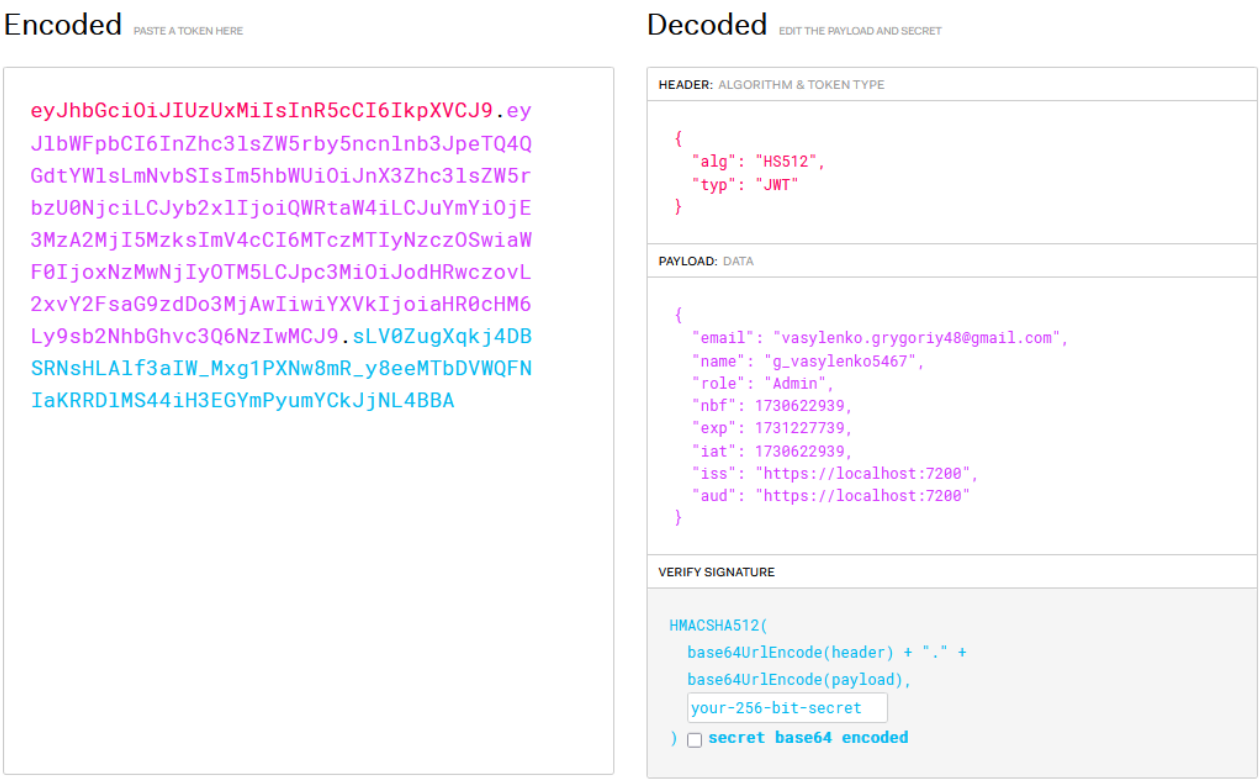

Рис. 1.1. Приклад розшифрування JWT токену

2.7. Angular

Angular — це популярний фреймворк для розробки веб-сайтів, створений командою Google. Angular побудований на використанні мови програмування TypeScript, типізованої версії JavaScript. Angular призначений для створення односторінкових додатків (також відомих як SPA), що дозволяє динамічно

оновлювати контент сторінки без необхідності повного перезавантаження, покращуючи швидкість роботи та зручність для користувачів.

Перевагами Angular є його модульність та чітка структура. Додаток Angular побудований із модулів, компонентів та сервісів, що дозволяє розробникам чітко організувати код, розділити відповідальність між різними частинами проєкту та спрощувати підтримку великих додатків. Компоненти відповідають за відображення інтерфейсу та логіку, використовуючи шаблони HTML та стилі CSS. Вони також взаємодіють з даними через двостороннє зв'язування, що забезпечує автоматичне оновлення інтерфейсу при зміні даних. Модулі ж об’єднують компоненти та сервіси в логічні блоки, що полегшує управління залежностями та завантаження частин додатка.

У ньому використовуюся директиви, що, так би мовити, розширюють звичайні можливості HTML, додаючи спеціальні атрибути та властивості до елементів сторінки. Директиви можуть змінювати поведінку або вигляд елементів, додавати логіку до шаблонів, полегшуючи створення динамічного та інтерактивного змісту сайту, що значно покращує UI/UX частини сайту.

Одним із найважливіших концептів Angular є ін'єкція залежностей, поширена практика серед багатьох ООП мов. Вона дозволяє легко управляти створенням та використанням сервісів — спеціальних класів, які містять спільну логіку або функціональність, наприклад, роботу з API чи обробку даних. Ін'єкція залежностей зменшує зв'язаність коду, що полегшує подальше тестування та рефакторинг.

Angular підтримує маршрутизацію, яка дозволяє створювати багатосторінкові додатки, де різні частини інтерфейсу відображаються залежно від URL-адреси, забезпечуючи плавний перехід між сторінками без перезавантаження.

Фреймворк також має вбудовану підтримку RxJS: бібліотеки для роботи з реактивними програмами, що дозволяє обробляти асинхронні події та потоки

даних. Особливо це може стати в нагоді при обробці HTTP-запитів та користувацького вводу.

Angular включає в себе потужний CLI, який спрощує створення, налаштування та збірку проєктів. CLI дозволяє автоматизувати багато рутинних завдань, таких як генерація компонентів, модулів і сервісів, а також виконує налаштування середовища розробки. Це значно зменшує час на розробку та підвищує продуктивність.

Розділ 3. Функціонал

3.1. Неавторизований користувач та способи авторизації

Коли новий користувач, або користувач, у якого закінчився термін дії попередньої сесії (у нашому випадку JWT токен, який видається будь-якому користувачу при успішній спробі авторизації, діє 7 днів), заходить на сайт, то він бачить сторінку з привітанням (рис.1.1).

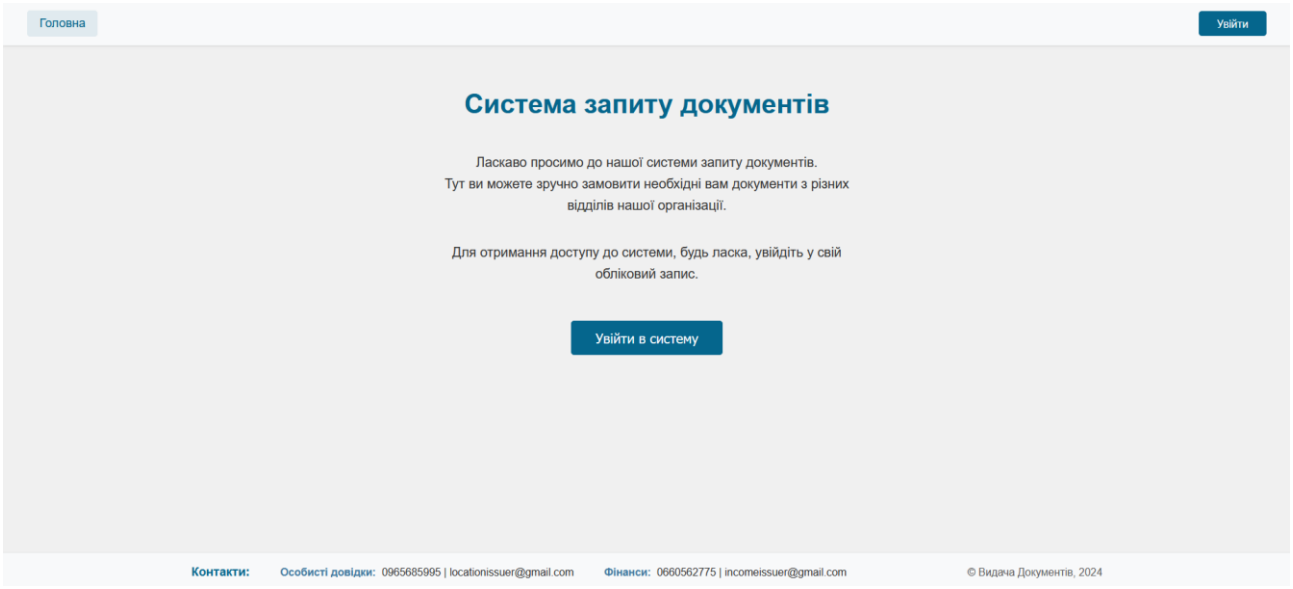

Рис. 1.1. Початкова сторінка

На цій сторінці і надалі завжди будуть присутні хедер (верхня частина, шапка), з можливостю увійти/вийти, та футер, у якому є контактні дані (а саме номера телефону, адреси електронної пошти) для зв’язку з відділами.

У неавторизрованого користувача є лишень можливість логіну, і, при натисканні відповідної кнопки, він може побачити сторінку з можливістю введення даних для входу в систему (рис 1.2).

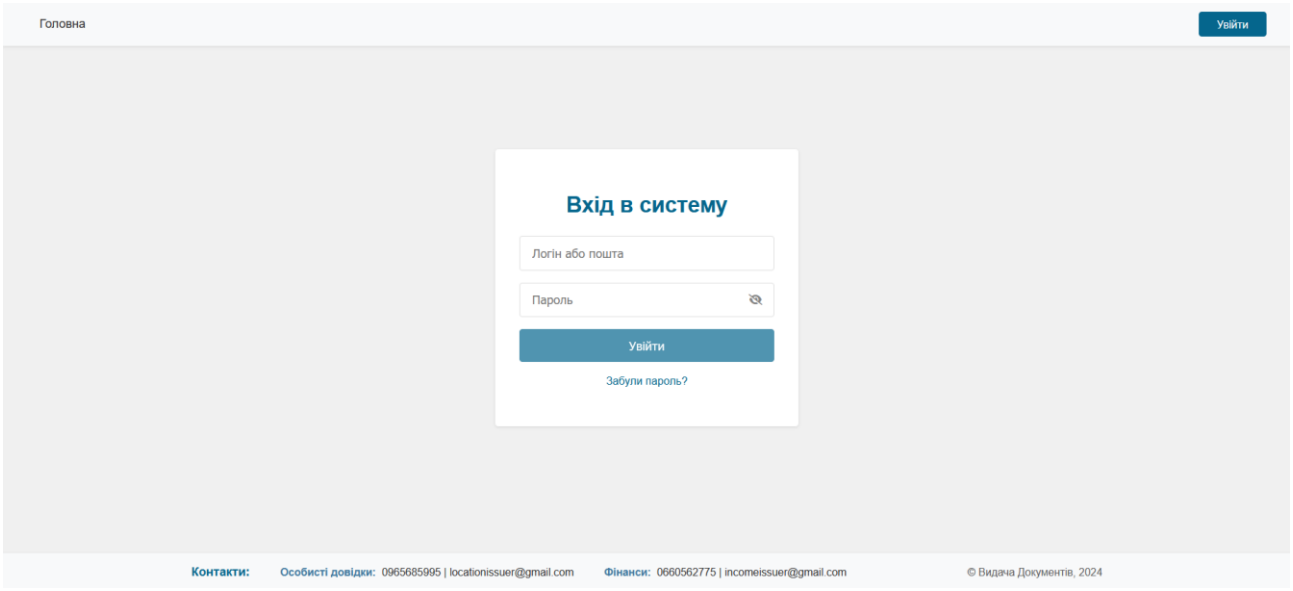

Рис. 2.2. Вікно логіну

Користувач має можливість увійти, ввівши у верхнє поле свою зареєстровану адресу електронної пошти, або виданий логін, та пароль у нижнє поле.

Реєстрація на сайті відсутня, нового користувача добавити у систему може лишень адміністратор, проте ці можливості будуть описані пізніше.

У випадку, якщо користувач забув пароль або хоче його змінити, він може натиснути кнопку «Забули пароль».

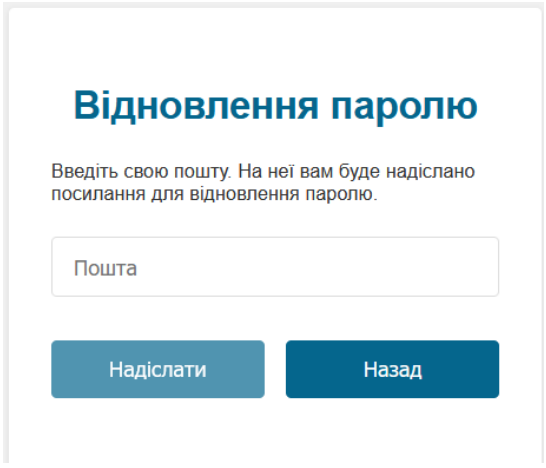

Рис. 2.3. Вікно відновлення паролю

Після введення пошти, якщо вона вже існує в системі, на неї прийде лист із запрошенням (рис. 2.4).

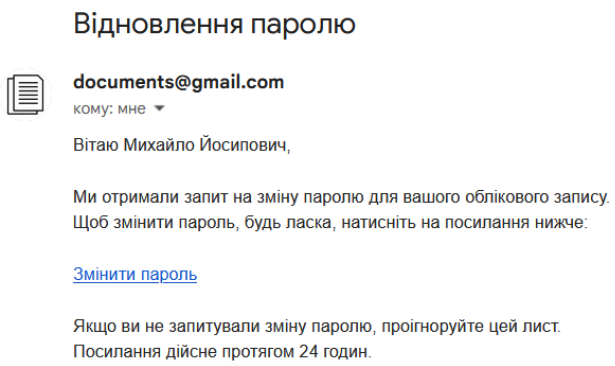

Рисунок 2.4. Лист з посиланням для відновлення паролю

Якщо користувача в системі не існує, то теж прийде повідомлення про те, що дані надіслані, проте насправді нічого не буде надіслано, що зроблено в цілях безпеки. При переході за посиланням, можна бачити вікна, зображенні на рис. 2.5 та 2.6.

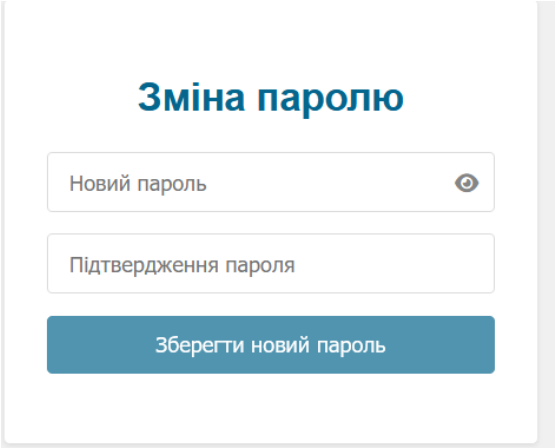

Рисунок 2.5. Зміна паролю

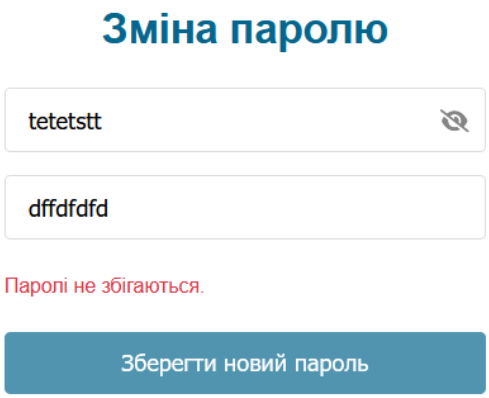

Рисунок 2.6. Валідація полей

У вікні зі зміною паролю буде два поля для введення паролю, проте, якщо вони не збігаються, то можливості зберегти пароль не буде. Коли користувач введе валідний пароль, то його буде перенаправлено назад до вікна входу в систему.

3.2. Функції адміністратора

Якщо авторизуватися із даними адміністратора, то при вході буде сторінка з рис. 2.7.

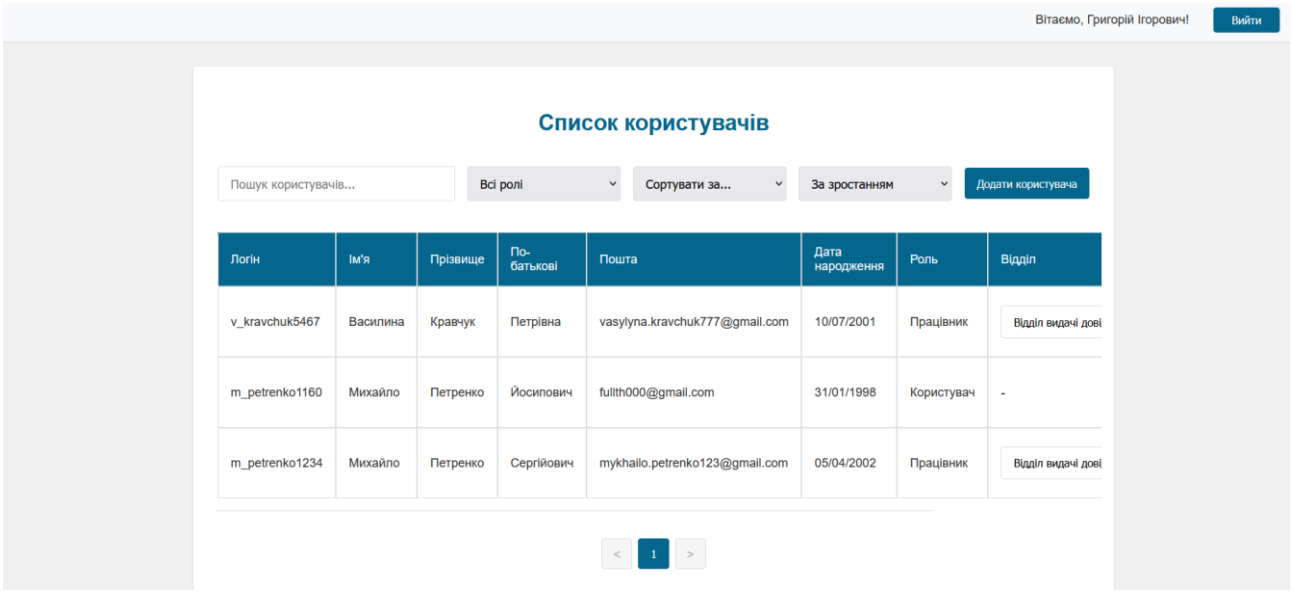

Рисунок 2.7. Сторінка адміністратора

На сторінці присутні:

- В хедері є привітання користувача за іменем та по-батькові.
- В тому ж хедері є можливість вийти зі системи. Якщо натиснути на цю кнопку, то з’явиться модальне вікно з проханням про підтвердження дії.
- В основній частині сайту можна спостерігати список усіх користувачів (за виключенням самого адміністратора), що є зареєстрованими у системі;
- Є поле пошуку, в якому можна шукати користувачів по: імені, прізвищу, по-батькові, логіну, електронній пошті. Якщо робити пошук по імені, прізвищу, та по-батькові, то ці дані можна вписувати разом.

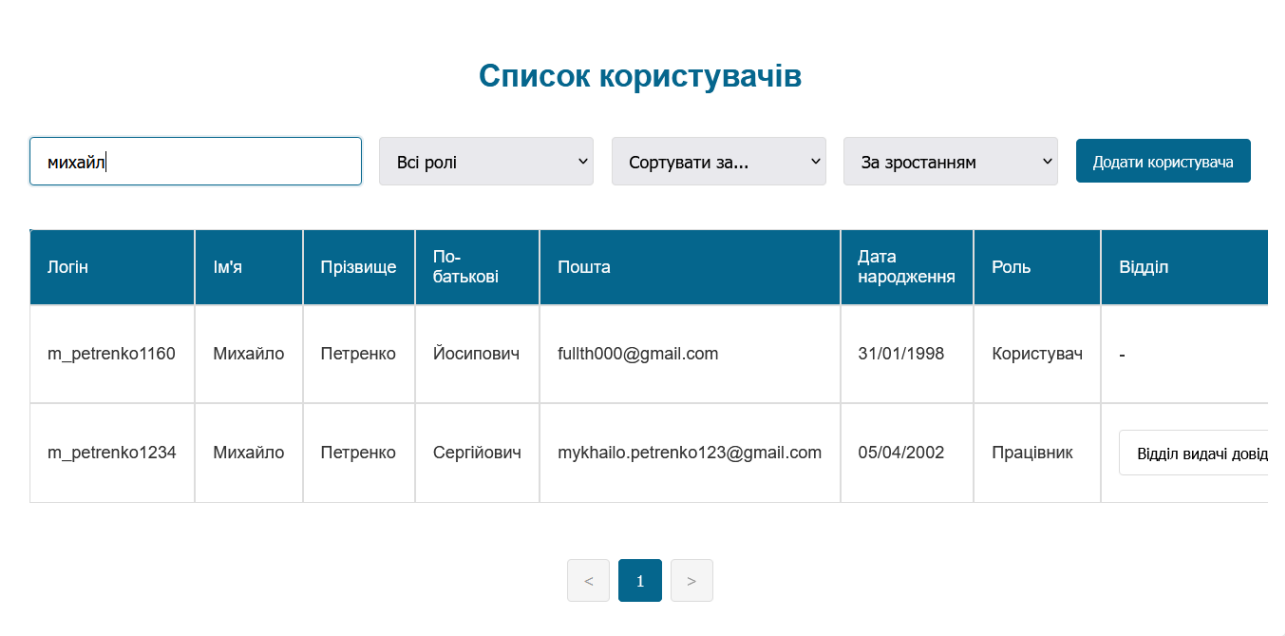

Рисунок 2.8. Приклад пошуку користувачів

- Є декілька випадаючих списків, які дозволяють фільтрувати користувачів:
  - У «Всі ролі» є вибір між ролями «Працівник» та «Користувач», також присутня стандартна опція «Всі ролі».
  - В «Сортувати за…» є наступні варіанти: ім’я, прізвище, дата народження, пошта, відділ.
  - Випадаючий список, де написано «За зростанням» є додатком до «Сортувати за…», і дає вибір між сортуванням за зростанням (стандартна опція) та за спаданням.
- При натисканні на кнопку «Додати користувача», з’являється модальне вікно, де можна ввести данні користувача, якого необхідно добавити в систему (рис. 2.9).

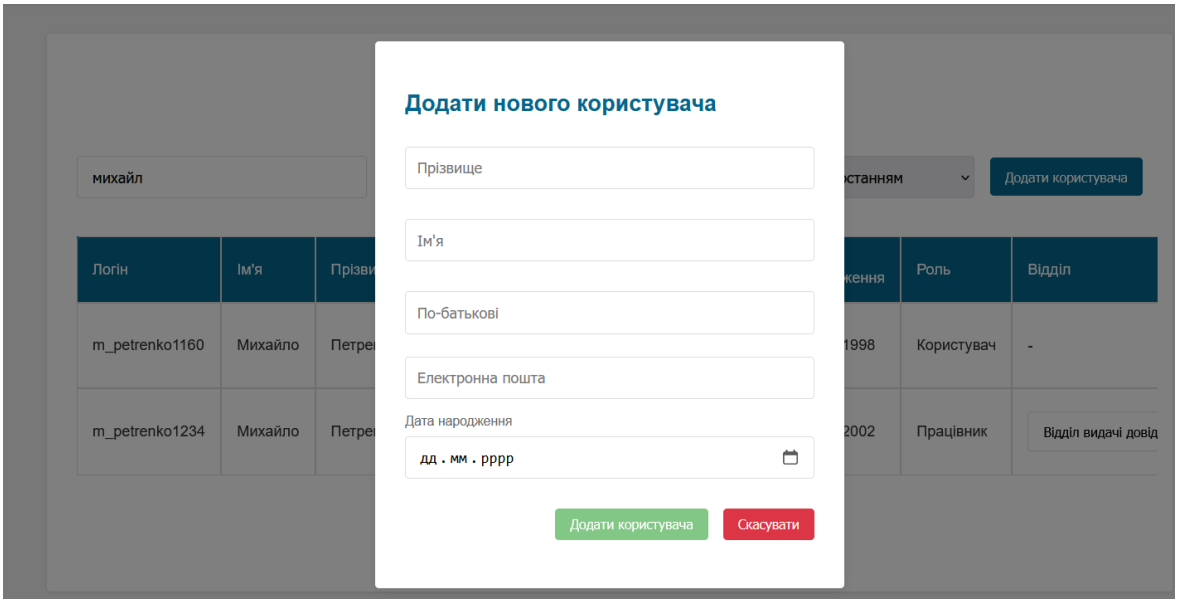

Рисунок 2.9. Вікно додавання нового користувача

Є 5 полів: Прізвище, Ім’я, По-батькові, Електронна пошта та Дата народження, і всі вони є обов’язковими для заповнення. Після заповнення даних про користувача і натискання на кнопку «Додати користувача», на вказану пошту прийде запрошення для входу в систему.

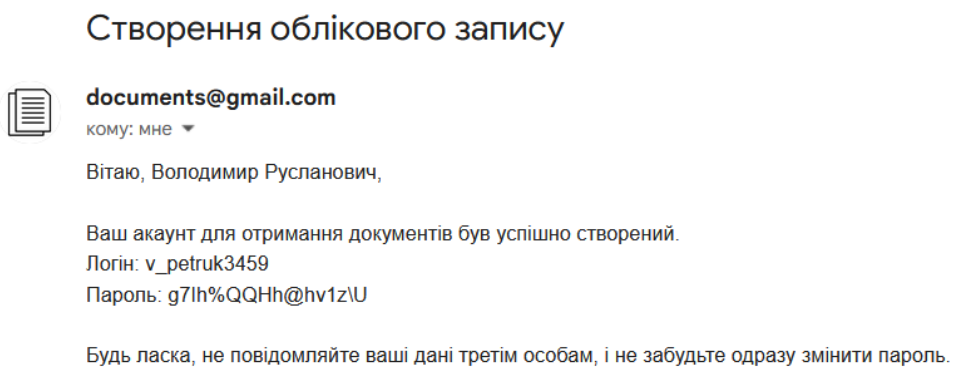

Рисунок 2.10. Лист з запрошенням для створення нового акаунту.

Логін користувача генерується на основі імені та прізвища користувача, з 4 випадковими цифрами, а пароль є повністю випадково згенерованим і надійним.

- У таблиці присутні основні дані про всіх користувачів: логін, ім’я, прізвище, по-батькові, пошта, дата народження, роль та відділ. Роль та відділ вказують на роль та права користувача у системі. Поруч з цим є можливість редагувати права користувача. Працівників можна зробити користувачами, можна змінити відділ, у якому вони працюють, а

користувачів можна зробити навпаки, працівниками, обов’язково приписавши їх до певного відділу.

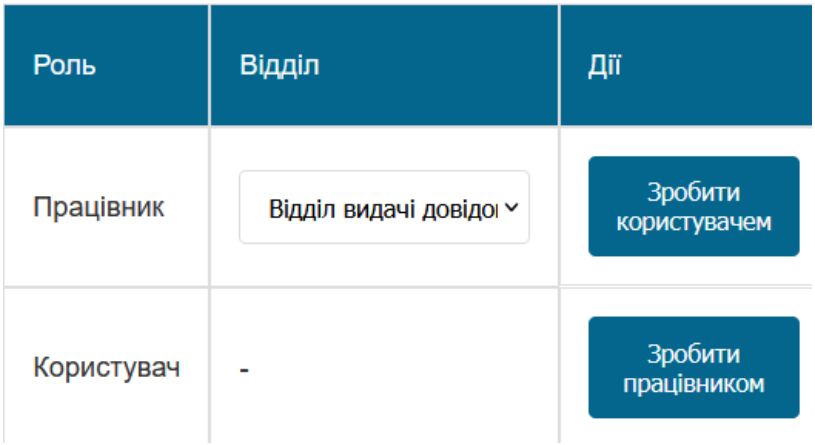

Рисунок 2.11. Додатковий функціонал адміністратора

- Під таблицею присутня пагінація, яка, при наявності великої кількості користувачів, дозволяє переглядати лишень певну кількість користувачів на одній сторінці. Це вирішує проблему з великими нагромадженням даних на сторінку та зменшує навантаження на сервер.

3.3. Функції звичайного користувача

Якщо перейти за даними користувача, якого ми щойно створили, то ми побачимо сторінку з відділами, о яких можна звернутися (рис. 2.12) та відповідними документами (рис. 2.13).

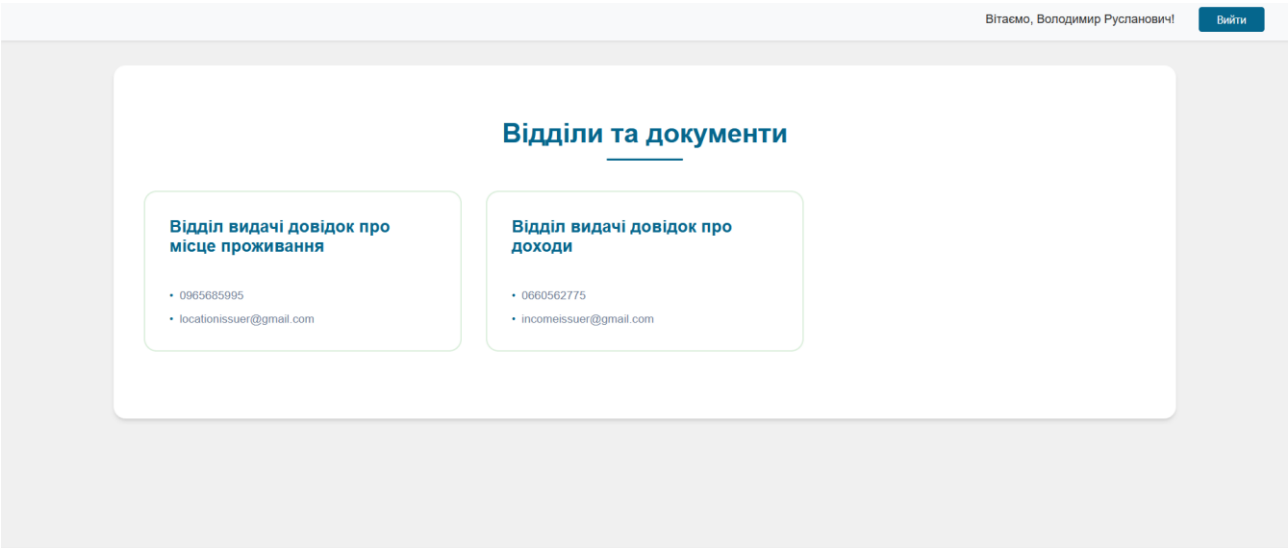

Рисунок 2.12. Відділи, до яких може звернутися користувач

У користувача є інформація про всі доступні у компанії відділи та їхні контактні дані. Він має можливість обрати будь-який з них та надати запит на отримання документу.

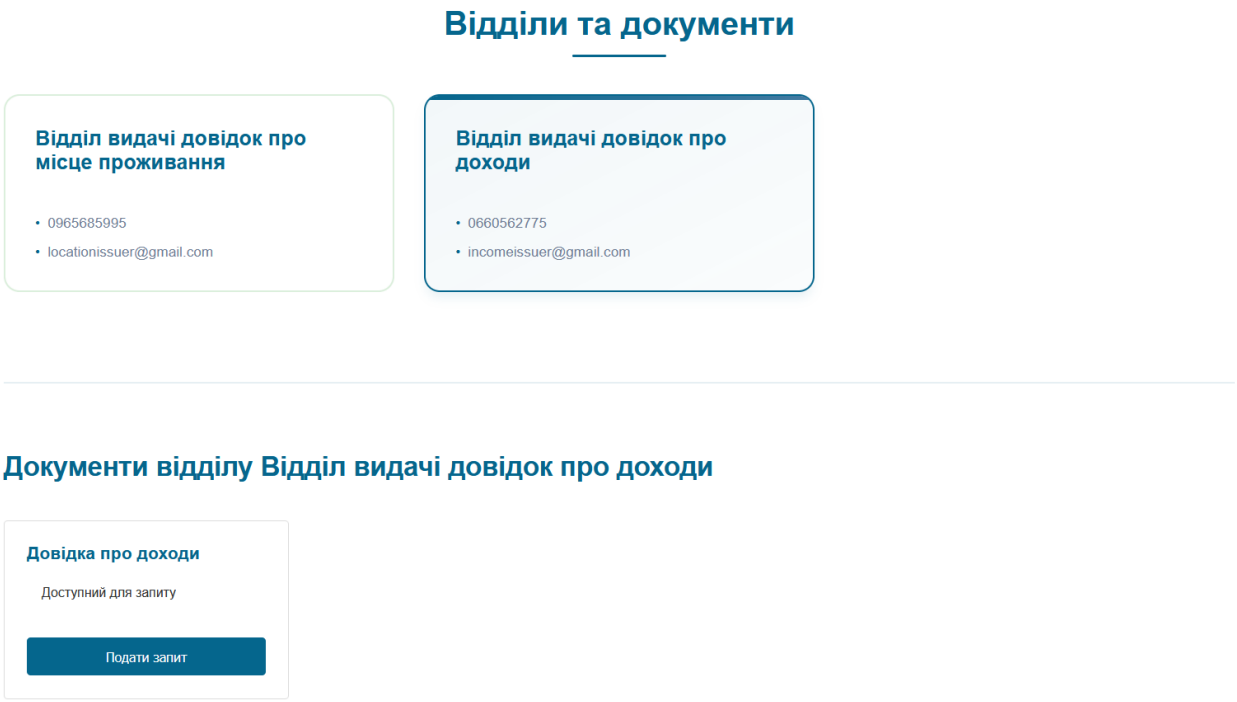

Рисунок 2.13. Документи, які можна отримати у певному відділі.

Якщо користувач запросить цей документ, то у нього оновиться статус.

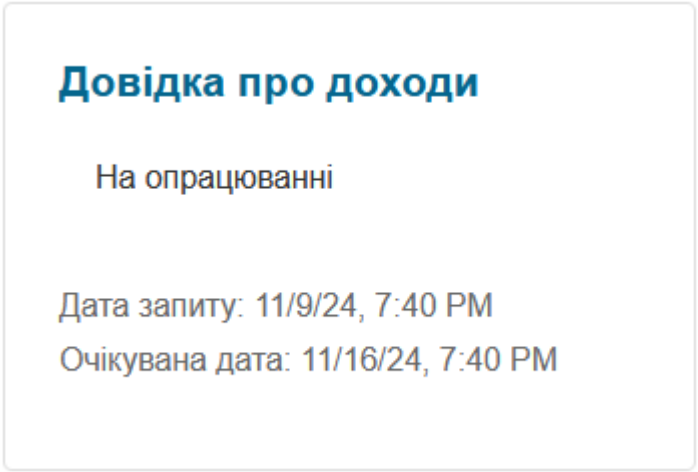

Рисунок 2.14. Оновлений статус документа (на опрацюванні)

У карточці з документом з’явиться інформація про статус документа (наразі «На опрацюванні», дата запиту, та очікувана дата отримання, яка зазвичай рівна 7 дням після надсилання самого запиту. Оскільки користувач є лишень щойно створеним у системі, то, окрім цього документу, у нього більше нема ніяких у наявності.

3.3. Функції працівника

Працівники у системі діляться на відділи. Кожен працівник може бачити лишень документи, які були запрошені у його відділі. Наприклад, щойно був надісланий запит на отримання документу у «Відділ видачі документів про доходи». Якщо зайти на сторінку працівника, який відноситься до іншого відділу, то, відповідно, він цього запиту побачити не зможе.

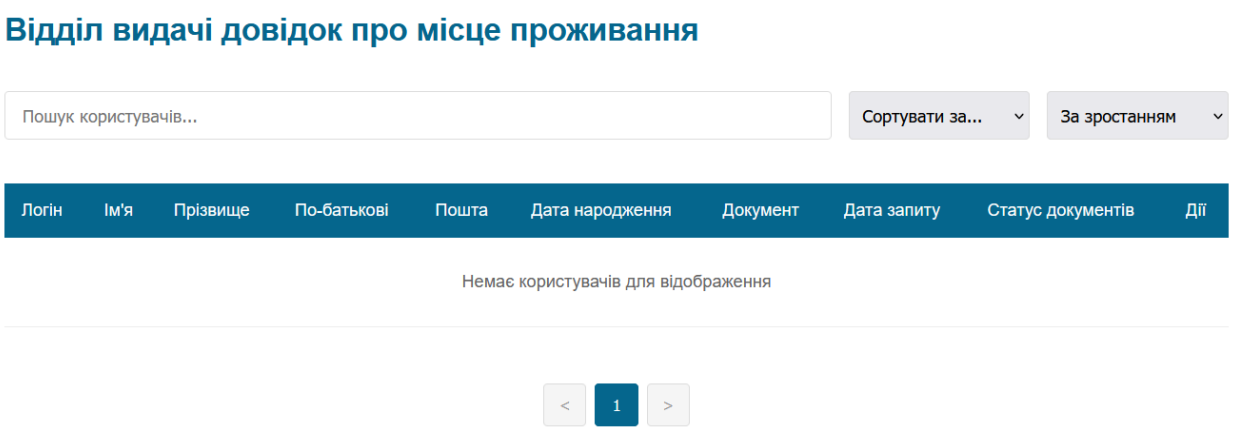

Рисунок 2.15. Відсутні користувачі, які подавали запит у цей відділ

Проте, якщо зайти як працівник іншого відділу, то щойно запитаний документ буде у списку.

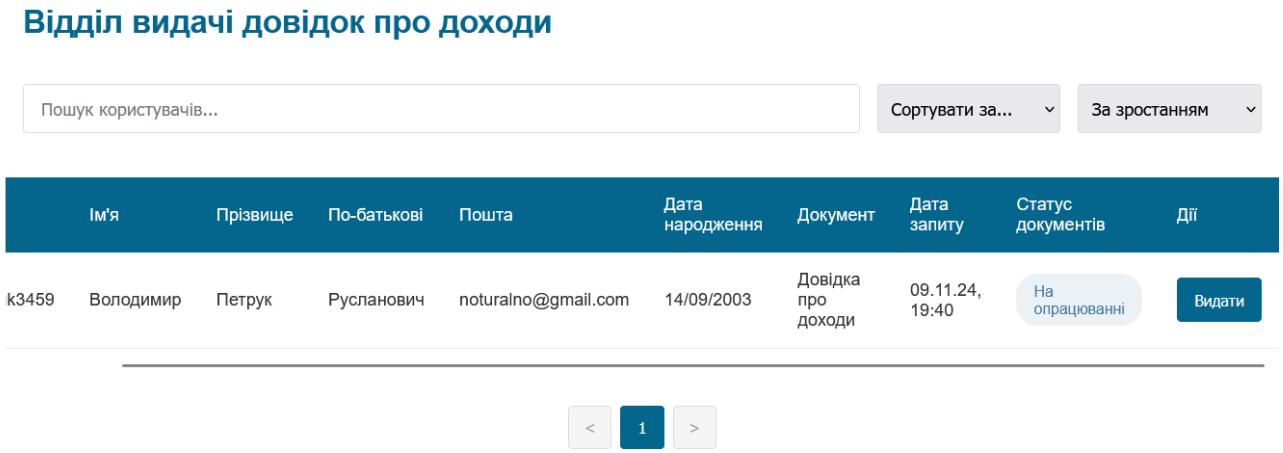

Рисунок 2.16. Інформація про користувачів та запрошені ними документи.

У цілому, працівник має таку ж саму інформацію про користувачів, як і адміністратор, за виключенням їх ролі та відділу, в якому вони працюють.

Проте, працівник може бачити, який документ запитує користувач, дату запиту, статус документу та має дії, які можна виконати над документом користувача, наразі він може лишень видавати запитаний документ. Коли працівник натискає на кнопку «Видати», у нього з’являється модальне вікно з проханням про підтвердження дії, і після цього статус документа оновлюється.

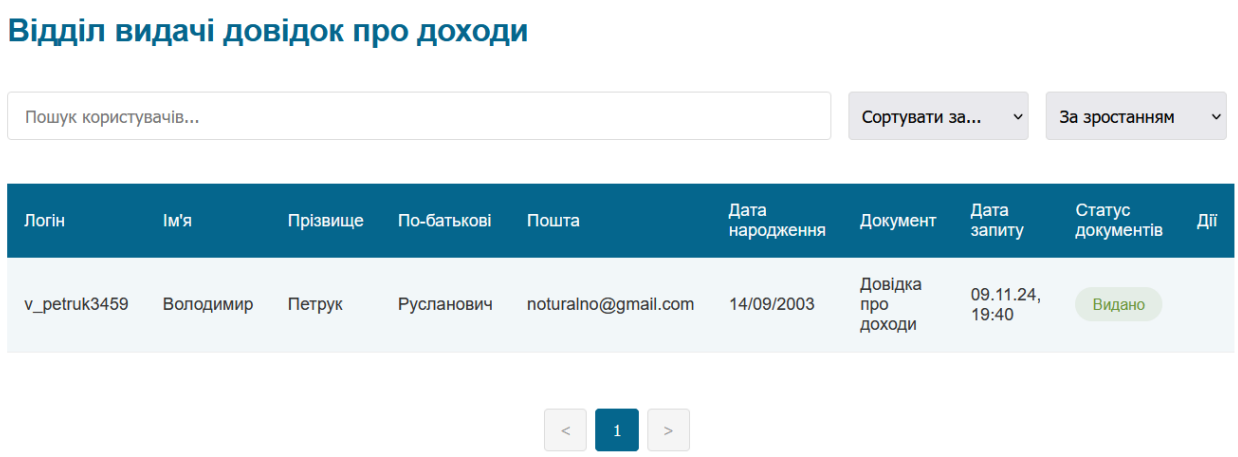

Рисунок 2.17. Вигляд виданого документу на сторінці працівника

Якщо зайти як користувач, який щойно запросив документи, то можна побачити статус «Отримано» та додаткове поле «Дата отримання».

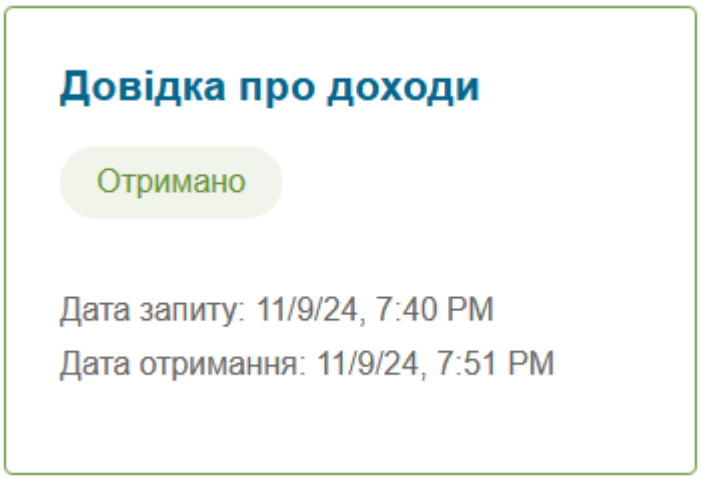

Рисунок 2.18. Приклад отриманого документа

Висновки

У результаті виконання кваліфікаційної роботи було створено веб-додаток з використанням ASP.NET для серверної частини, яка приймає та відправляє запити за архітектурою REST, Angular для front-end частини, MySQL як бази даних, а також декілька бібліотек згаданих технологій.

Технології були вибрані з врахуванням їхньої сучасності, швидкості, надійності та можливості для розробника використовувати їх для швидкої розробки. Це дозволило створити програмний продукт, який відповідає вимогам сучасних інформаційних систем та забезпечує високу продуктивність у роботі з великими обсягами даних.

У розробленому додатку була створена рольова система по видачі документів, яка передбачає як можливість видачі документів користувачам в рамках певної організації. Система дозволяє адмініструвати права доступу для різних категорій користувачів, забезпечуючи високий рівень безпеки даних. Користувачі ж мають змогу переглядати свої документи, і, якщо документ не отриманий, запросити його або слідкувати за його статусом.

Особлива увага була приділена створенню інтуїтивно зрозумілого інтерфейсу, який побудовано з використанням Angular. Це забезпечує динамічність та зручність роботи з додатком, що є важливим фактором для покращення взаємодії користувачів із системою. Використання Entity Framework спростило роботу з базою даних MySQL, дозволяючи реалізувати складні операції з мінімальними зусиллями.

Проведений аналіз існуючих систем управління документами дозволив врахувати їх переваги та недоліки, що вплинуло на вибір архітектури та функціоналу. Поточна реалізація системи забезпечує її стійкість, масштабованість та можливість для подальшого вдосконалення.

Серед перспектив розширення функціоналу можна виділити:

- інтеграцію з іншими інформаційними системами та платформами для обміну даними;
- реалізацію багатомовної підтримки, що дозволить використовувати додаток у різних країнах;
- додавання аналітичних інструментів для відстеження та аналізу документів;
- автоматизацію процесів, таких як надсилання повідомлень про зміни статусу документів.

Поточна реалізація не є кінцевою, тому система може бути адаптована під специфічні потреби замовників. Цей проєкт демонструє значний потенціал для

впровадження у різних організаціях, де є необхідність у автоматизації процесів зберігання, управління та видачі документів. У цілому, розробка цієї системи є важливим внеском у сферу управління документами та сучасних інформаційних технологій.

Список використаної літератури

1. Офіційна документація по MySQL: https://dev.mysql.com/doc/
2. Переваги використання MySQL як СУБД: https://www.oracle.com/mysql/what-is-mysql/
3. Основи HTML: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics
4. Основи CSS: https://developer.mozilla.org/en-US/docs/Learn/CSS/First_steps/What_is_CSS
5. Документація по Angular: https://angular.dev/overview
6. Документація Microsoft по .NET: https://learn.microsoft.com/en-us/dotnet/
7. Документація Microsoft по Entity Framework: https://learn.microsoft.com/en-us/ef/
8. Документація Microsoft по Identity Framework та ASP.NET: https://learn.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-8.0&tabs=visual-studio