Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мережних технологій факультету інформатики

РОЗРОБКА САЙТУ ДЛЯ ІНФОРМУВАННЯ ГРОМАДЯН ТА ЗДІЙСНЕННЯ КОМУНІКАЦІЇ З ОРГАНАМИ МІСЦЕВОЇ ВЛАДИ ОТГ

Текстова частина до курсової роботи
за спеціальністю «Інженерія програмного забезпечення» 121

Керівник курсової роботи
к. ф-м. н., ст. в. Гречко А. В.

_____

(підпис)

“_____” _____ 2021 р.

Виконала студентка Чумаченко О. Р.

“_____” _____ 2021 р.

Київ 2021

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мережних технологій факультету інформатики

ЗАТВЕРДЖУЮ
Зав. кафедри мережних технологій,
професор, док. ф-м н.,
_____ Г. І. Малашонок
(підпис)
“____” _____2020 р

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студентки Чумаченко Олександри Романівни факультету інформатики 4 курсу

Тема: Розробка сайту для інформування громадян та здійснення комунікації з органами місцевої влади ОТГ.

Зміст ТЧ до курсової роботи:

Анотація

Вступ

Розділ 1. Аналіз предметної області. Постановка завдання курсової роботи

Розділ 2. Проектування системи

Розділ 3. Опис реалізації програмного продукту

Висновки

Додаток А. Діаграма компонентів клієнтської частини.

Додаток Б. Тестові дані користувачів

Список літератури

Дата видачі “___” _____ 2020 р. Керівник _____
(підпис)

Завдання отримав _____
(підпис)

Тема: Розробка сайту для інформування громадян та здійснення комунікації з органами місцевої влади ОТГ.

Календарний план виконання курсової роботи:

№ п/п	Назва етапу курсової роботи	Термін виконання етапу	Примітка
1.	Отримання завдання на курсову роботу	01.11.2020	
2.	Збір інформації щодо предметної області	15.12.2020	
3.	Збір функціональних вимог до системи	15.01.2021	
4.	Проектування моделі даних	25.01.2021	
5.	Реалізація баз даних	31.01.2021	
6.	Створення серверної частини системи	15.02.2021	
7.	Створення клієнтської частини застосунку	01.03.2021	
8.	Тестування системи	14.03.2021	
9.	Написання текстової частини курсової роботи	28.03.2021	
10.	Створення презентації	11.04.2021	
11.	Захист курсової роботи	19.04.2021	

Студент _____

Керівник _____

“____” _____2020 р

Зміст

Анотація

Сьогодні процес діджіталізації набирає все більшого темпу. Створення веб-ресурсів для зв'язку з місцевою владою, інформування громадян є відповіддю на виклики сучасності.

Мета курсової робити створити ресурс здатний задовольнити потребу зв'язку громадян об'єднаних територіальних громад з місцевою владою щодо питань, пропозицій та отримання інформації про новини та події в режимі онлайн.

Результатом даної роботи є функціонуюча платформа для забезпечення зв'язку з місцевою владою, яка дозволяє створювати та редагувати новини і події, подавати звернення, розміщувати петиції та підтримує різні рівні доступу.

Вступ

У сучасному світі все більшої швидкості набуває процес діджіталізації. Подія останнього року – пандемія коронавірусу, лише збільшила темпи оцифровування сфер життя та актуальність даного процесу. Створення веб-ресурсів для зв'язку з місцевою владою, інформування громадян є відповіддю на виклики сучасності. Такі платформи мають в собі безліч переваг, серед яких швидкий доступ до інформації будь-де та будь-коли, зменшення часу комунікації з чиновниками та можливості повідомляти громадян про важливі події за допомогою електронної пошти.

Мета курсової робити створити ресурс здатний задовольнити потребу зв'язку громадян об'єднаних територіальних громад з місцевою владою щодо питань, пропозицій та отримання інформації про новини та події. Завданням курсової роботи є створення веб-платформи, що дозволяє переглядати новини, події, створювати звернення та петиції.

Об'єктом дослідження є створення та розміщення на хмарних платформах серверної і клієнтської частини веб-застосунка.

Предметом дослідження є веб-системи, які забезпечують автентифікацію та авторизацію користувачів, та здатні задовольнити мінімальні інформаційні потреби, які існують у громадян об'єднаних територіальних громад.

Практичним значенням одержаних результатів є функціонуюча платформа для забезпечення зв'язку з місцевою владою, яка дозволяє створювати та редагувати новини і події, подавати звернення та розміщувати петиції. Платформа забезпечує різнорівневий доступ до функцій, підтримує автентифікацію та авторизацію.

Реалізація серверної частини відбувалась на мові C# з використанням модульної платформи для розробки програмного забезпечення .NetCore 5 в

середовищі розробки Microsoft Visual Studio 2019. Для підключення до баз даних використовувався Entity Framework. Реалізація клієнтської частини відбувалась в інтегрованому середовищі розробки WebStorm 2019, з використанням javascript фреймворку Vue.js. Для розгортання застосунку було обрано хмарну PaaS-платформу Heroku. Для збереження даних була обрана система управління базами даних Postgres.

Робота складається зі вступу, трьох розділів, списку використаних джерел та двох додатків.

Перший розділ присвячений аналізу предметної області. Обґрунтовується актуальність, розглядаються та аналізуються аналоги розробки.

В другому розділі описується проектування системи, наводяться зібрані функціональні вимоги, описується модель даних та вибір програмних засобів.

Третій розділ присвячено опису процесу розробки. Розглядається реалізація та підключення баз даних, створення серверної та клієнтської частин, розгортання в хмарі та тестування.

Розділ 1. Аналіз предметної області.

1.1 Аналіз сучасного стану питання та обґрунтування теми

На сьогоднішній день ресурси для подання петицій та висвітлення новин існують в органах центральної влади, в великих населених пунктах (містах, обласних центрах) присутні платформи, що надають інформацію щодо подій та іноді надають змогу залишити звернення. В менших містах та селищах подібні сервіси частіше за все представлені групами в соціальних мережах, або сайтами, інформація на яких буває застарілою. Крім того існує проблема у відсутності централізованої платформи з новинами та подіями громади. Тобто існують сайти, або сторінки в соціальних мережах від різних організації таких, як школа, лікарня, селищна рада, тощо. Як результат важливі події певної організації можуть не привернути увагу громадськості.

Яскравим прикладом подібної проблеми є новостворена об'єднана територіальна громада в Запорізькій області. В ОТГ основним джерелом новин та подій є група в соціальній мережі Facebook. Варто зауважити, до моменту об'єднання селищ, у кожного з них було по власній групі новин та окремо існували новини від певних організацій. Також функціонує сайт ради ОТГ, який містить деякі новини, але він є менш популярним за групу та не забезпечує необхідною кількістю інформації.

Сьогодні будь-хто може створювати допис в групі ОТГ. Результатом цього є спам та складнощі пошуку важливих повідомлень від органів місцевої влади.

1.2 Огляд існуючих аналогів розробки

За наявності аналогів програмних продуктів, які розв'язують подібні задачі, потрібно коротко охарактеризувати основні обмеження та недоліки й запропонувати шляхи їх усунення.

Останні роки швидкими темпами відбувається цифрова трансформація суспільства, багато послуг вже можливо отримати в онлайн-режимі. Сьогодні веб-сайти з наданням загальної інформації та новин стають майже обов'язковим атрибутом організацій та органів влади.

1.2.1 Офіційний сайт Запорізької міської влади

Сторінки сайту мають лаконічний, приємний дизайн, основними кольором якого є білий. На сайті представлено 10 розділів: новини, влада, економіка, документи, суспільство, covid-19, е-сервіси, корисне, місто та вибори-2020.

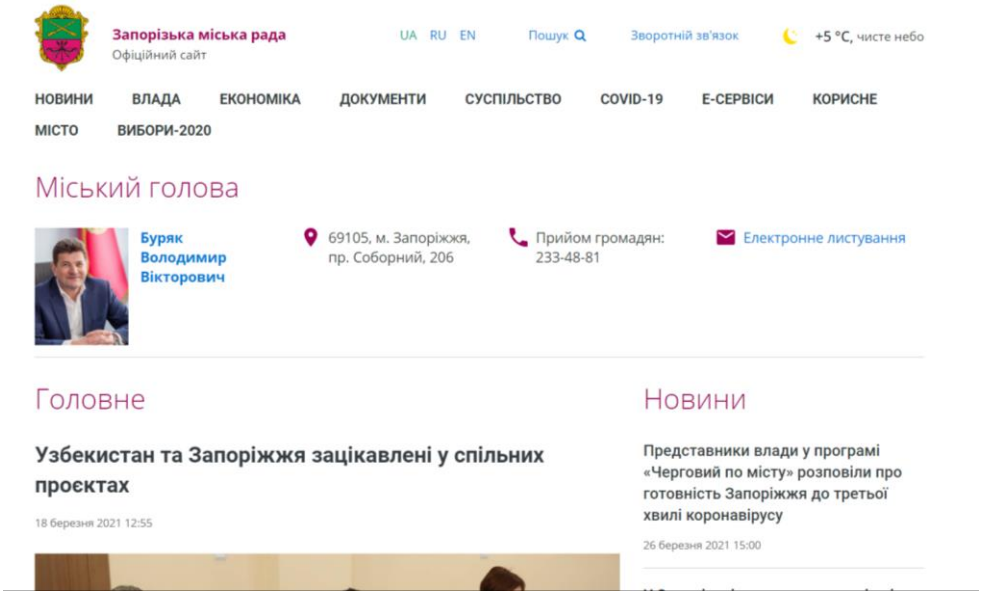


Рис. 1 Офіційний сайт Запорізької міської ради

Існує форма зворотного зв'язку для здійснення мобільного зв'язку з органами місцевої влади. Веб-портал використовується для надання громадянам актуальної інформації про новини, прийняті рішення органів влади, доступні сервіси, серед яких – сервіс електронних петицій. [1]

Перегляд петицій доступний всім користувачам, голосування та створення нових – лише зареєстрованим. Для реєстрації необхідно ввести ім'я,

по-батькові, прізвище, електронну пошту та пароль, є можливість зареєструватись за допомогою електронного підпису. На сторінці кожної петиції виводиться інформація про автора, дата початку збору підписів, заголовок та текст петиції, кількість зібраних голосів та кількість днів, що залишилась. Необхідна кількість голосів відсутня.

1.2.2 Електронні петиції до Президента України

Дизайн сайту лаконічний та інтуїтивно зрозумілий. На сайті є можливість фільтрації петицій, при перегляді петиції відображається заголовок, текст петиції, автор, дата оприлюднення, кількість днів, що залишились, кількість голосів, необхідна кількість голосів, можна переглянути підписантів, та відповідь, якщо вона надана.

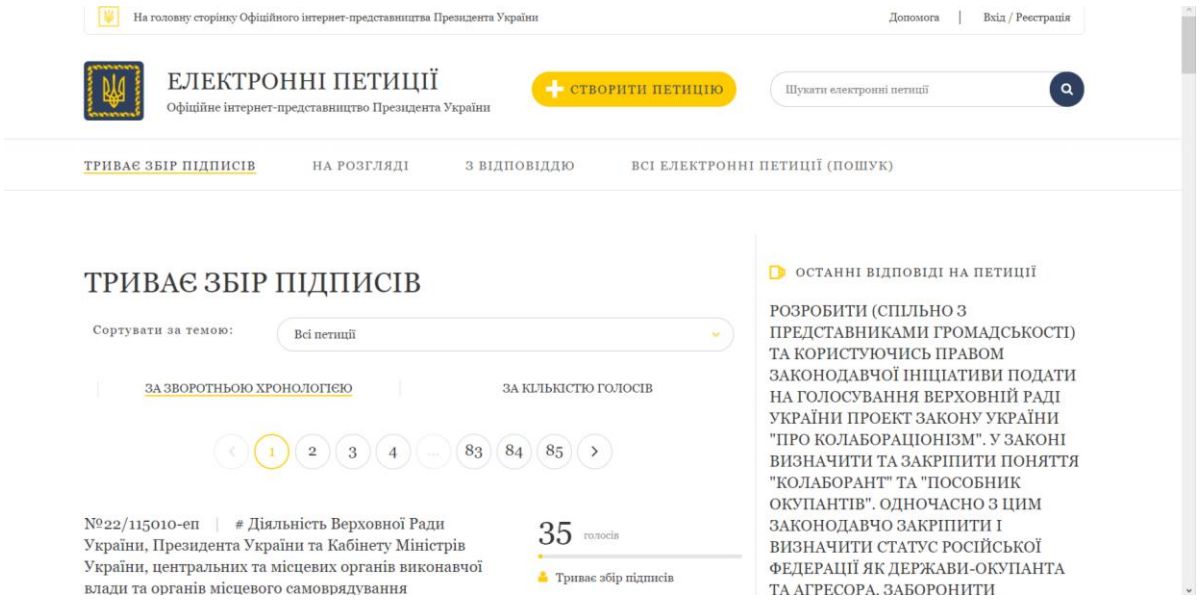


Рис. 2 Сайт електронних петицій до Президента

Для створення та голосування петицій необхідна реєстрація, яка виконується за допомогою BankId. Найстаріші петиції, що розміщені на сайті, датуються серпнем 2015 року. [2]

1.3 Постановка задачі

Створити веб-платформу, що підтримує авторизацію з різними рівнями доступу (адміністратори, зареєстровані користувачі, незареєстровані користувачі) та дозволяє розміщувати новини та події, створювати петиції та забезпечує зв'язок з місцевою владою. Реєстрація повинна відбуватись з використанням електронної пошти та повинен бути передбачений механізм підтвердження проживання особи на території громади. Дизайн веб-сайту повинен бути лаконічним та зручним.

Розділ 2. Проектування системи

2.1 Збір функціональних вимог

Система може бути поділена на п'ять основні складові: користувачі та організації, новини та події, звернення, петиції.

2.1.1 Користувачі та організації

2.1.1.1 Авторизація та автентифікація користувачів

Автентифікація повинна відбуватись з використанням токену, що видається після підтвердження існування в системі користувача з введеним паролем та логіном.

Під час реєстрації користувачеві необхідно ввести ім'я, по-батькові, прізвище, адресу електронної пошти, логін та пароль. Після реєстрації на вказану адресу буде надіслано лист з підтвердженням пошти. Без підтвердженої пошти вхід систему неможливий.

В системі повинна бути авторизація користувачів базована на ролях. Повинні бути реалізовані ролі:

- користувач (роль надається одразу після реєстрації, дозволяє створювати звернення);
- підтверджений користувач, або користувач, який проживає на території громади (роль повинна автоматично надаватись після підтвердження факту проживання);
- адміністратор новин та подій (користувач з такою роллю отримує можливість створювати та редагувати створені ним новини та події);
- модератор (користувач має право редагувати новини та події);
- адміністратор звернень та петицій (роль дозволяє додавати відповіді до петиції, опрацьовувати та відповідати на звернення);

– адміністратор користувачів (роль дозволяє створювати нові організації, редагувати права користувачів та приналежність їх до організацій).

Незареєстровані користувачі мають право зареєструватись, переглянути новини, події, петиції та інформацію щодо організацій.

Для користувачів повинна бути можливість скинути пароль.

2.1.1.2 Підтвердження факту проживання користувача на території громади

В рамках даної системи для підтвердження факту проживання користувача та розширення його прав дозволами на створення та голосування петицій повинно існувати два варіанти: підтвердження на основі документів та зміна прав адміністратором. Підтвердження на основі документів відбувається шляхом порівняння даних облікового запису користувача (імені, по-батькові, прізвища), введених користувачем даних (номеру паспорта або реєстраційного номеру облікової картки платника податків) з даними що існують в базі даних, підключеній до системи.

2.1.1.3 Кабінет користувача

Зареєстрований користувач після входу до системи отримує доступ до свого кабінету. В кабінеті повинно бути реалізовано:

– Редагування ім'я, по-батькові, прізвища;
– Зміна номер телефону;
– Редагування електронної пошти;
– Зміна логін та зміна паролю;

– Розширення прав, якщо факт проживання на території громади не підтверджений.

При редагування логіну або електронної пошти повинна відбуватись перевірка, що введені дані не мають дублікатів в системі. Після зміни електронної пошти на введену адресу повинен надсилатись лист з підтвердженням.

Для розширення прав користувачу необхідно ввести номер паспорту, або реєстраційний номер облікової картки платника податків. Якщо введені дані існують в системі, обліковому запису користувача надається право на створення та голосування петицій.

2.1.1.4 Перелік користувачів

Перелік користувачів доступний лише обліковим записам, які мають роль адміністратор звернень та петицій або адміністратор користувачів.

При перегляду списку повинно відображатись ім'я та прізвище, логін, електронна пошта, чи підтверджена пошта та спрощене представлення ролі (підтверджений та непідтверджений користувач та адміністратор). Повинна надаватись можливість фільтрувати користувачів за рівнем доступу.

При перегляди інформацію про конкретного користувача має відображатись: прізвище, ім'я, по-батькові, логін, електронна пошта, номер телефону, організація та перелік всіх ролей. Якщо перегляд виконує користувач з роллю "Адміністратор користувачів" також повинна бути доступна зміна ролі та організації.

2.1.1.5 Організації

При перегляду списку про кожну організацію має відображатись її назва та контактний телефон. При перегляду конкретної організації, окрім назви та номеру телефону, повинна відображатись її адреса. Для користувачів з роллю "Адміністратор користувачів" повинно бути доступне редагування організації, видалення та перегляд користувачів організації та видалення організації, якщо жоден користувач їй не належить.

При створенні нової організації необхідно вказати її назву, адресу та контактний телефон.

2.1.2 Новини та події

Перегляд новин та подій повинен бути доступний всім користувачам, враховуючи незареєстрованих у системі.

При перегляді списку новин повинно відображатись заголовок новини, організація, користувач якої створив новину, частина тексту, фотографії, додані до новини та дата створення. Також може відображатись помітка "Редаговано", якщо після створення новини були внесені якісь зміни та автор новини, якщо при створення було вказано відображення автора.

При перегляді конкретної новини має відображатись заголовок, дата створення, організація, текст повністю, зображення та може відображатись автор, або помітка "Редаговано".

Якщо новину переглядає модератор, або автор новини має бути доступна функція редагування новини. Також модератор може видалити новину.

При створенні новини необхідно заповнити поле заголовку, тексту новини та можна завантажити до 5 фотографій, можна обрати чи показувати автора новини.

При перегляді списку подій необхідно відображати заголовок події, організацію, ім'я автора, частину тексту, дату та час початку події, дату і час завершення та прикріплені до події фото. Якщо подія була редагована, повинна відображатись відповідна помітка. Ім'я автора може бути приховане, якщо вибрана відповідна опція при створенні.

При перегляді окремої події повинна відображатись вся інформація про подію. Якщо подію переглядає автор або модератор повинно бути доступним редагування події, а для модератора також повинно бути доступне видалення.

При створенні новини необхідно заповнити поле заголовку, тексту новини, обрати дату та час початку і закінчення події та можна завантажити до 5 фотографій, можна обрати чи показувати автора події та розіслати оповіщення користувачам.

2.1.3 Петиції

Перегляд петицій повинен бути доступним всім користувачам. Під час перегляду списку петицій необхідно виводити тему петиції, кількість голосів, скільки днів для розгляду залишилось, та статус петиції. Статус петиції може набувати трьох значень:

- триває збір підписів;
- на розгляді;
- з відповіддю;
- не підтримано.

Після створення петиції її статус приймає значення "Триває збір підписів", якщо петиція набирає необхідну кількість голосів її статус

змінюється на статус "На розгляді", якщо відведений час сплинув і петиція не набрала необхідної кількості голосів – статус змінюється на "Не підтримано". Якщо петиція набрала необхідну кількість голосів і до неї додали відповідь, то її статус автоматично змінюється на "З відповіддю". Після додання відповіді, відповідно до статті 23^1 закону «Про звернення громадян» [3], на електронну пошту автору петиції має надсилатися лист, що містить інформацію про відповідь до петиції.

Під час перегляду петиції відображається її тема, повний текст, автор, дата початку збору підписів, кількість голосів, необхідна кількість голосів, кількість днів до завершення збору підписів. Має бути можливість переглянути список людей, що проголосувала.

При створенні петиції необхідно заповнити поле теми та тексту петиції. Можливість редагування та видалення петиції не передбачена.

Після того як петиція набирає необхідну кількість голосів у адміністратора повинна з'являтись можливість додати відповідь на петицію.

2.1.4 Звернення

Перегляд усіх звернень має бути доступним лише для користувачів з роллю "Адміністратор звернень та петицій", для зареєстрованих користувачів можливо переглянути лише створені ними звернення. При виводу списку звернень потрібно відображати ім'я автора, дату створення, статус звернення та, якщо надана відповідь, то автора та дату відповіді.

При перегляді окремого звернення повинне відображатися його статус, автор, час створення, тема, тест звернення, прикріплені фотографії. Якщо відповідь надана, то автор відповіді, час відповіді та сама відповідь.

При створенні звернення необхідно вказати тему та текст звернення та можна прикріпити до 5 зображень. Редагування та видалення звернень не передбачені.

Звернення можуть буть трьох статусів:

– очікується;
– в процесі;
– закрите.

Після створення звернення його статус стає "Очікується". Після того як адміністратор бере в обробку звернення, його статус повинен змінюватись на статус "В процесі". Після додання відповіді адміністратором, звернення змінює статус на "Закрите", а автору надсилається повідомлення на пошту з текстом відповіді.

2.2 Проектування бази даних

Враховуючи функціональні вимоги до системи можна виокремити дві бази даних - основну базу даних застосунку, з інформацією про новини, події, петиції, звернення, облікові записи користувачів, та базу даних, яка існує незалежно від системи та надає системі доступ на читання. Остання база даних буде представлена в системі досить спрощено, і міститиме лише одну таблицю користувачів з необхідної інформацією: ім'ям, прізвищем, по-батькові, номером паспорту та реєстраційним номером облікової книжки платника податків.

Основна база даних системи міститиме 9 таблиць: Користувачі, Організації, таблиця з кодами підтвердження електронної пошти, Події, Новини, Петиції, Подання, Мультимедіа, Голоси. Атрибути кожної з сутностей представлені на рисунку 3.

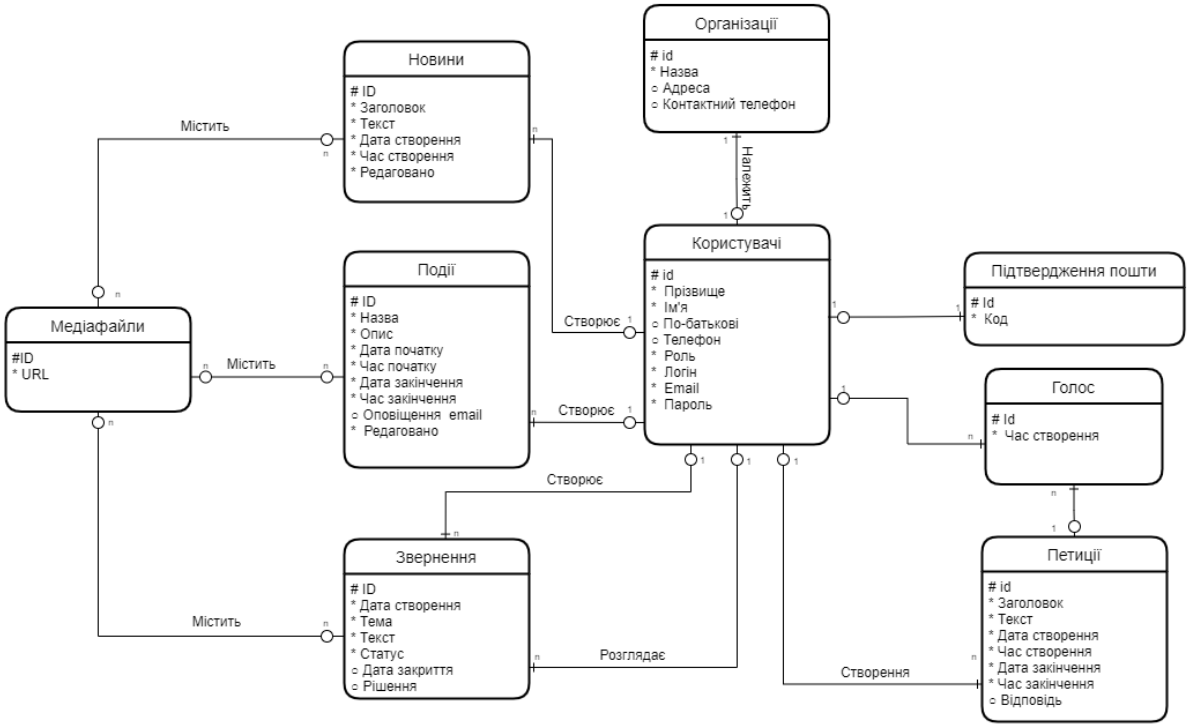


Рис. 3 ЕР-модель бази даних

2.3 Вибір програмних засобів

2.3.1 Система управління базою даних

Вибір системи управління базою даних відбувався між двома провайдерами: MySQL та PostgreSQL.

MySQL – одна з найбільш популярних реляційних систем управління базами даних, яка розробляється і підтримується корпорацією Oracle. Це безкоштовне програмне забезпечення, яке часто покращує безпеку і запроваджує нові функції. Також існують платні випуски призначені для комерційного використанню.

Основними перевагами є:

– безкоштовність,
– багато функцій доступні для безкоштовної версії,

– можливо налаштувати для роботи з іншими базами даних, наприклад Oracle.

Основними недоліками є:

– відсутність вбудованої підтримки XML,
– відсутня автоматизація деяких можливостей, наприклад створення інкрементних резервних копій,
– підтримка, яка є платною.

PostgreSQL – система управління базами даних, яка часто використовується для веб-баз. Це перша з розроблених СУБД, яка дозволяє користувачам управляти як структурованими так і неструктурованими даними.

Основні переваги PostgreSQL:

– механізм управління базою даних який масштабується і може обробляти терабайти даних,
– підтримка JSON,
– багато пре визначених функцій та інтерфейсів. [4]

Основним недоліком є те, що швидкість може знизитись під час великих операцій або запитів читання.

Враховуючи всі переваги та недоліки було прийнято рішення використовувати в роботі PostgreSQL. Також вагомим була безкоштовний хостинг СУБД хмарною платформою Heroku.

2.3.2 Програмні засоби серверної частини

2.3.2.1 Asp.Net Core

Asp.Net Core – це кросплатформна, високопродуктивна платформа з відкритим кодом для створення програм, що підтримують хмарні технології і

є підключеними до інтернету. За допомогою платформи можливо створювати веб-програми та сервіси, програми Інтернету речей та мобільні серверні системи. Asp.Net Core дозволяє використовувати різні засоби розробки в Windows, Linux та macOS та розгортається в хмарі або локально. [5]

Платформа пропонує багато типів застосунків, серед яких для даної роботи найбільше підходить два: Web app та Web API. Для подальшої розробки було обрано саме Web API спираючись на те, що він підтримує RESTful HTTP сервіси та дозволяє в подальшому розширити клієнтські застосунки мобільним додатком та, наприклад, ботом в Telegram.

2.3.2.2 Entity Framework Core

Entity Framework (EF) Core – полегшена, розширена, відкрита та кросплатформна версія популярної технології доступу до даних Entity Framework. [6] EF Core підтримує багато різних баз даних, крім того він полегшує перехід від одного провайдеру баз даних до іншого.

Центральної концепцією Entity Framework є поняття сутності або entity. Сутність представляє набір даних, асоційованих з певним об'єктом. Тому дана технологія передбачає роботу не з таблицями, а з об'єктами і їх наборами.

Entity Framework передбачає три підходи взаємодії з базою даних:

- Database First: Entity Framework створює набір класів, які відображають модель конкретної бази даних.
- Model First: спочатку розробник створює модель бази даних, по якій потім Entity Framework створює реальну базу даних на сервері.
- Code First: розробник створює клас моделі даних, які будуть зберігатися в базі даних, а потім Entity Framework за цією моделлю генерує базу даних і її таблиці. [7]

В роботі для основної бази даних використано підхід Code First, а для другої бази даних – Database First.

2.3.2.3 Autofac

Autofac - це контейнер інверсії керування для .Net Core, Asp.NET Core, .Net 4.5.1+, універсальних програм для Windows та інших. Ідея інверсії контролю полягає в тому, що замість того аби пов'язувати класи застосунку разом і "поновлювати" їх залежності, просто перемикали залежності і передавати їх під час побудови класу. [8]

2.3.2.4 AutoMapper

AutoMapper - це засіб відображення об'єкта на об'єкт. Співставлення об'єкт-об'єкт працює шляхом перетворення вхідного об'єкту одного типу в вихідний об'єкт іншого типу.

Співставлення об'єкт-об'єкт може відбуватись в багатьох місцях застосунку, але найчастіше на кордонах між рінями, наприклад рівнем служби і домена. Проблеми одного рівня часто конфліктують з проблемами іншого, тому співставлення об'єкт-об'єкт призводить до розділення моделей, де проблеми кожного рівня можуть впливати тільки на типи цього рівня. AutoMapper забезпечує просте налаштування типів. [9]

2.3.3 Програмні засоби клієнтської сторони

2.3.3.1 Webpack

Webpack – це збирач статичних модулів для сучасних застосунків JavaScript. Під час обробки застосунку він будує граф залежностей, який

відображає кожен модуль, що потрібен проекту і генерує один або декілька пакетів. [10]

2.3.3.2 Node package manager

Node package manager (npm) – це, по-перше, найбільший реєстр програмного забезпечення. Розробники з відкритим кодом з усіх континентів використовують npm для обміну та запозичення пакетів, а багато організацій використовують npm для управління приватною розробкою. [11] По-друге, npm – це утиліта командного рядка для взаємодії з вказаним репозиторієм, яка допомагає в встановленні пакетів, управління версіями і залежностями.

2.3.3.3 Vue.js

Vue – це прогресивний фреймворк для створення інтерфейсів користувачів. Основна бібліотека орієнтована лише на рівень представлення, її легко інтегрувати з іншими бібліотеками або існуючими проектами. Vue також підтримує складні односторінкові застосунки при використанні в поєднанні з сучасними інструментами та додатковими бібліотеками. [12]

Причиною вибору фреймворку в першу чергу був невеликий попередній досвід роботи з ним та бажання поглибити знання і отримати практичний досвід.

2.3.3.4 Vuex

Vuex - це шаблон та бібліотека управління станом для застосунків Vue.js. Він слугує централізованим сховищем для всіх компонентів застосунку

з правилам, які гарантують, що стан може змінюватись тільки передбаченим шляхом. [13]

2.3.3.5 Vue Router

Vue Router - офіційний маршрутизатор для Vue.js. Він інтегрований з ядром Vue.js, що спрощує створення односторінкових застосунків. Містить в собі наступні можливості:

- Вкладене співставлення маршруту;
- Модульна конфігурація маршрутизатора на основі компонентів;
- Параметри маршруту, запити;
- Деталізоване управління навігації;
- Посилання з автоматично активними класами CSS;
- Режим історії HTML5;
- Налаштування поведінки прокрутки. [14]

Розділ 3. Опис реалізації програмного продукту

3.1 Реалізація та підключення баз даних

3.1.1 Реалізація основної бази даних

Для реалізації бази даних відповідно до спроектованої концептуальної моделі, використовувався підхід Code First. В рамках цього підходу спочатку для кожної таблиці створюються класи-сутності, їх поля – атрибути сутності. Зв'язки між сутностями прописуються в окремому для кожної сутності класі. Первинний, альтернативні, зовнішні ключі та назви таблиць і полів можна визначити за допомогою анотацій в класі-сутності, або в конфігураційному класі. Для даного застосунку були визначені класи зображені на рисунку.

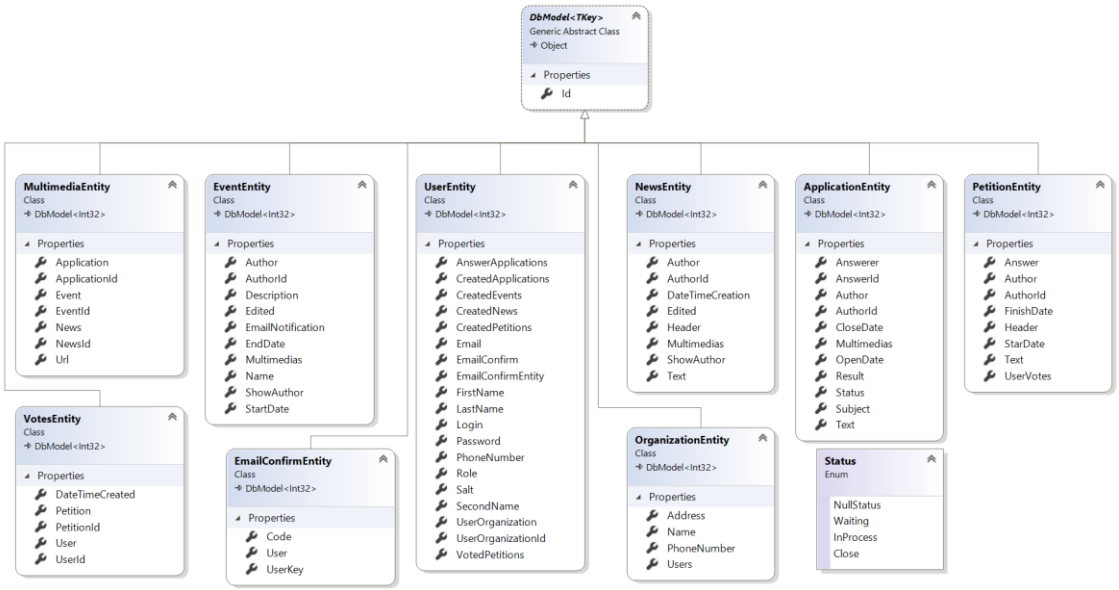


Рис. 4 Діаграма сутностей бази даних

Для їх конфігурації (опис ключів, визначення зв’язків, назви стовпців та таблиць) створені класи, що зображені на рис . Вони наслідують інтерфейс IEntityTypeConfiguration<T> та реалізують єдиний метод Configure().

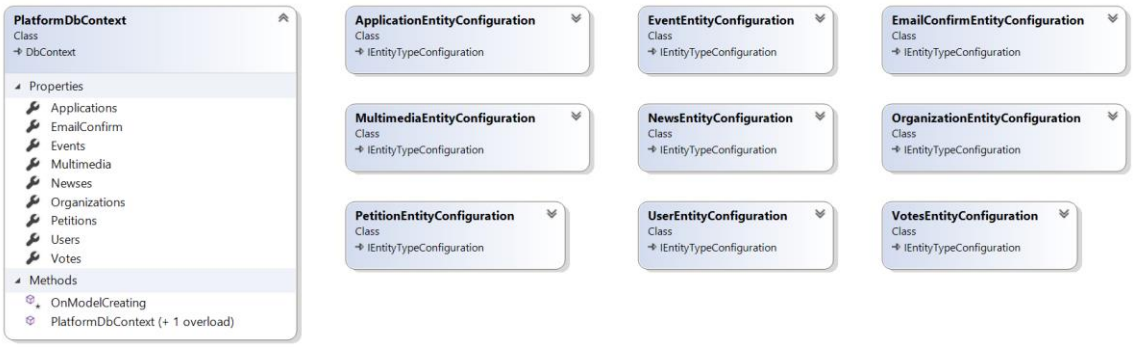


Рис. 5 Діаграма конфігурацій бази даних

Для зв'язку один до одного, в одному з конфігураційних файлів сутностей, що приймають участь у зв'язку, необхідно розмістити наступний код:

```
builder.HasOne(e => e.EmailConfirmEntity)
        .WithOne(u => u.User)
        .HasForeignKey<EmailConfirmEntity>(e => e.UserKey)
        .OnDelete(DeleteBehavior.Cascade);
```

В даному прикладі реалізований зв'язок між таблицею з кодами підтвердження пошти (EmailConfirmEntity) і таблицею користувачів (User). Метод HasOne вказується властивість що відповідає сутності EmailConfirmEntity в середині User. Метод WithOne вказує яка властивість в середині EmailConfirmEntity відповідає User. Наступним кроком за допомогою .HasForeignKey визначається зовнішній ключ, що знаходиться в таблиці з кодами підтвердження пошти. Метод OnDelete визначає поведінку при видаленні головної сутності.

Зв'язок один до багатьох визначається в конфігураційному файлі основної сутності наступним чином:

```
builder.HasMany(u => u.CreatedApplications)
        .WithOne(an => an.Author)
        .HasForeignKey(an => an.AuthorId);
```

Метод HasMany вказує на колекцію залежних сутностей, WithOne вказує, що в кожній сутності колекції існує зв'язок лише з однією основною сутністю. HasForeignKey вказує який атрибут залежної сутності є зовнішнім ключем.

Наступним кроком необхідно створити клас контексту та визначити в ньому DbSet<> для кожної сутності.

В методі ConfigureServices(IServiceCollection services) класу Startup додається контекст бази даних, та визначається стрічка підключення та провайдер. В якості провайдера для бази даних PostgreSQL використовується Npgsql.EntityFrameworkCore.PostgreSQL.

Після цього командою: Add-Migration name --context ContextName створюється міграція. Міграції в EF Core забезпечують поступове оновлення схеми бази даних, синхронізують її з моделлю даних програми, зберігаючи наявні дані в базі даних.[15] Для оновлення бази даних використовується команда Update-database --context ContextName.

3.1.2 Підключення до зовнішньої бази даних

Для зовнішньої бази даних використовувався DatabaseFirst підхід. В рамках цього підходу база даних вже існує і EF Core генерує класи на її основі. Для генерації класів з існуючої бази даних PostreSQL використовувалась команда: dotnet ef dbcontext scaffold ConnectionString Npgsql.EntityFrameworkCore.PostgreSQL -o Models, де ConnectionString – стрічка підключення, що містить назву бази, ім’я користувача, пароль, порт та хост і може містити додаткові опції, такі як ввімкнення режиму SSL, а Models – вихідна папка в якій будуть розміщені згенеровані класи. Після виконання команди був згенерований клас-сутність та клас контексту бази даних.

3.1.3 Патерни репозиторій та специфікація

Взаємодія застосунку з базами даних виконується з допомогою патернів репозиторій та специфікація.

3.1.3.1 Патерн Репозиторій

Репозиторії - це класи або компоненти, які інкапсулюють логіку доступу до джерел даних. Вони централізують загальну функціональність доступу до даних, забезпечуючи кращу підтримку та відокремлюють інфраструктуру або технологію, що використовуються для доступу до баз даних від рівня доменної моделі. [16]

Патерн репозиторій - це добре задокументований спосіб роботи з джерелом даних. Мартін Фаулер описує репозиторій наступним чином: "Репозиторій виступає посередником між рівнями моделі та даними. Клієнтські об'єкти декларативно створюють запити та надсилають їх до репозиторіїв для одержання відповідей. Концептуально, репозиторій інкапсулює набір об’єктів, які зберігаються в базі даних, і операції, які можна виконувати над ними. Також репозиторії чітко і в одному напрямку відокремлюють залежність між доменом та розподілом даних." [17]

Патерн репозиторій рекомендується використовувати для досягнення наступних цілей:

- максимально збільшити обсяг коду, який можна автоматизовано тестувати і ізолювати рівень даних для підтримки модульного тестування;
- отримати доступ до джерела даних з різних місць і використовувати централізоване управління та логіку доступу;

– реалізувати та централізувати стратегію кешування для джерела даних
– покращити підтримку і читання коду, виокремив бізнес-логіку від логіки доступу до даних або служб;
– використання строго типізованих бізнес-об'єктів для виявлення проблем під час компіляції, а не під час виконання;
– застосувати модель предметної області для спрощення складної бізнес-логіки. [18]

В даній роботі патерн використовується для досягнення цілей по покращенню читання та підтримки коді і отриманні доступу до джерел даних з використанням централізованого управління. В результаті для кожної таблиці основної бази даних був створений свій репозиторій, який наслідував загальний клас EfRepository, що реалізовував методи CRUD (рис. 7). Для зовнішньої бази даних створено клас-репозиторій, що містить методи читання та пошуку за різним атрибутом (рис. 8). Кожен з репозиторію наслідує відповідний інтерфейс, що дозволяє використовувати Dependency Injection.

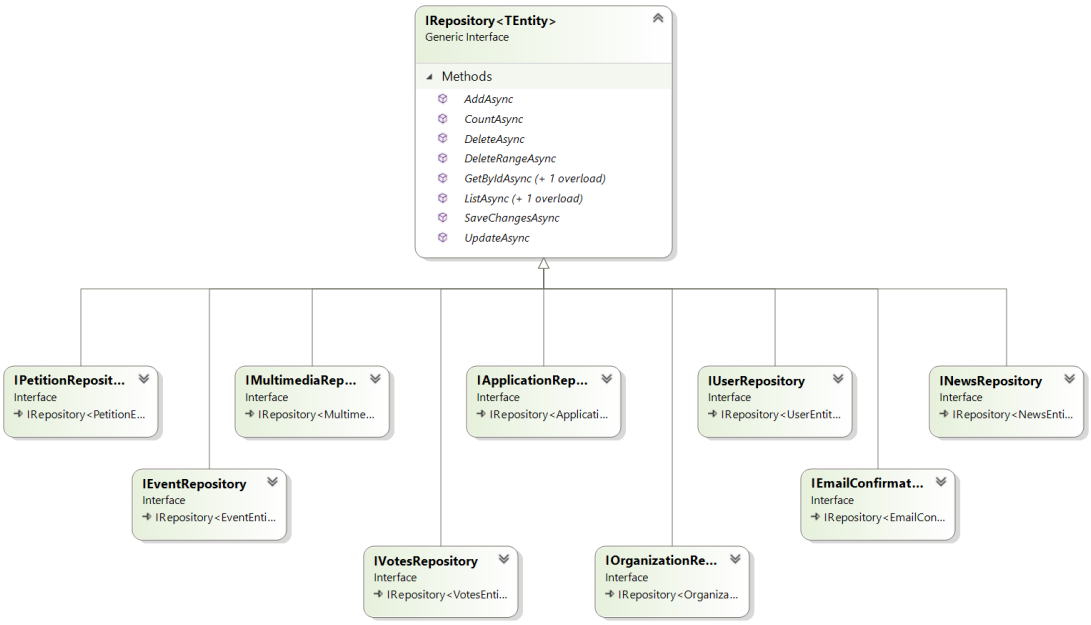


Рис. 6 Діаграма інтерфейсів репозиторіїв основної бази даних

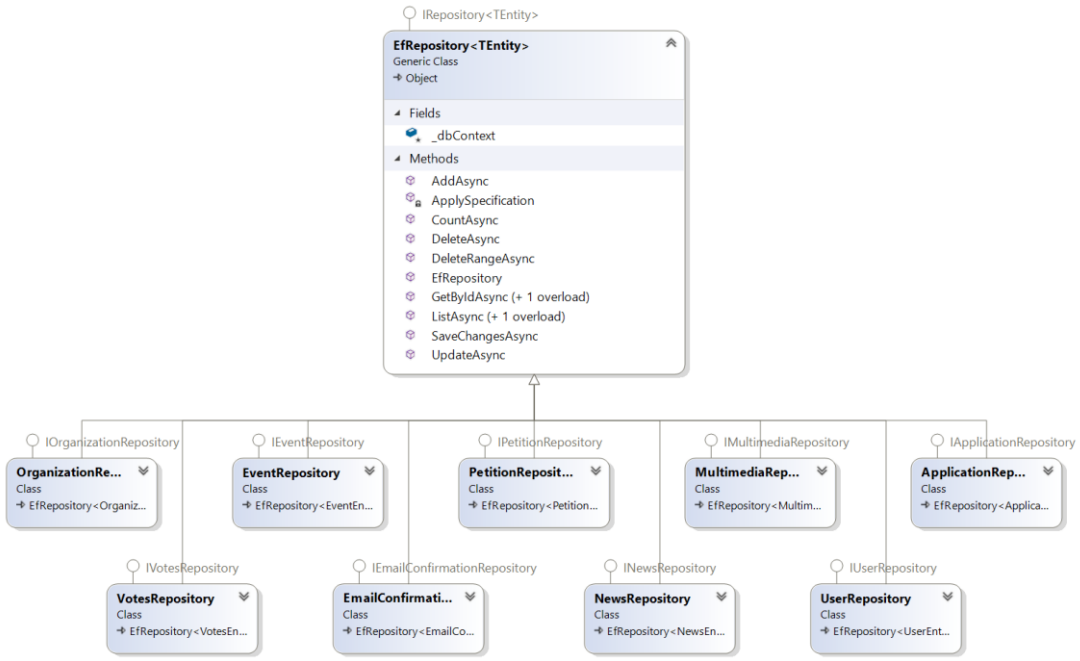


Рис. 7 Діаграма класів репозиторіїв основної бази даних

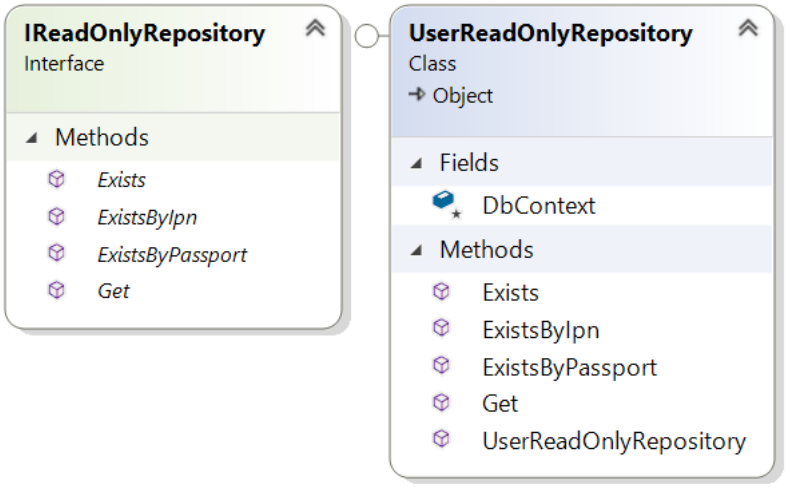


Рис. 8 Клас репозиторію та інтерфейс від якого він наслідується зовнішньої бази даних

3.1.3.2 Патерн Специфікація

Патерн специфікації дозволяє інкапсулювати деяку частину знань предметної області в єдину одиницю – специфікацію, і повторно використовувати її в різних частинах коду.

Це забезпечує рішення проблеми, де розмістити логіку запитів, сортування та підвантаження, оскільки описує запит в об’єкті.

Переваги патерну:

– Специфікація має назву;
– Його можна протестувати ізольовано;
– Його можна легко використати повторно;

Може використовуватися для опису форми даних, що повертаються - запити можуть повертати лише ті дані, які вони вимагали, усуваючи необхідність лінивого завантаження у веб-додатках [19].

В даній роботі патерн реалізовано з використанням бібліотеки Ardalis.Specification. Всі реалізовані специфікації наслідують узагальнений клас Specification<T>, який надається бібліотекою. Результатом реалізації патерну є класи зображені на рис. 9 .

Рис. 9 Специфікації до баз даних

3.2 Реалізація серверної частини

3.2.1 Архітектура сервера

Архітектура сервера побудована з використанням підходу DDD. Domain-Driven Desing (DDD) – проблемно-орієнтоване проектування – засноване на підході, при якому класи моделюються на основі реальних бізнес-процесів. В контексті створення застосунку DDD розглядає проблеми як предметні області. Цей підхід описує незалежні проблемні області, як обмежені контексти (кожен обмежений контекст відповідає мікрослужбі).[20] Відповідно до підходу серверна частина поділяється на три рівні:

- рівень моделі предметної області, що відповідає за представлення концепції бізнес-процесі, бізнес-правил. Цей рівень в застосунку представляють інтерфейси і класи Services;
- рівень застосунку, що визначає задачі, які має виконувати програмне забезпечення. Цей рівень не містить бізнес-правил, а лише делегує виконання роботи і координує задачі. В даній роботі його представляють контролери;
- рівень інфраструктури, що визначає спосіб збереження даних, які містяться в сутності предметної області, базі даних, або інших сховищах даних. У роботі цей рівень представлений репозиторіями, що створювались разом з базами.

3.2.1.1 Рівень моделі предметної області

Результатом реалізації стали 12 класів-сервісів (рис. 10), з яких 9 реалізують роботу з сутностями основної бази даних через репозиторії.

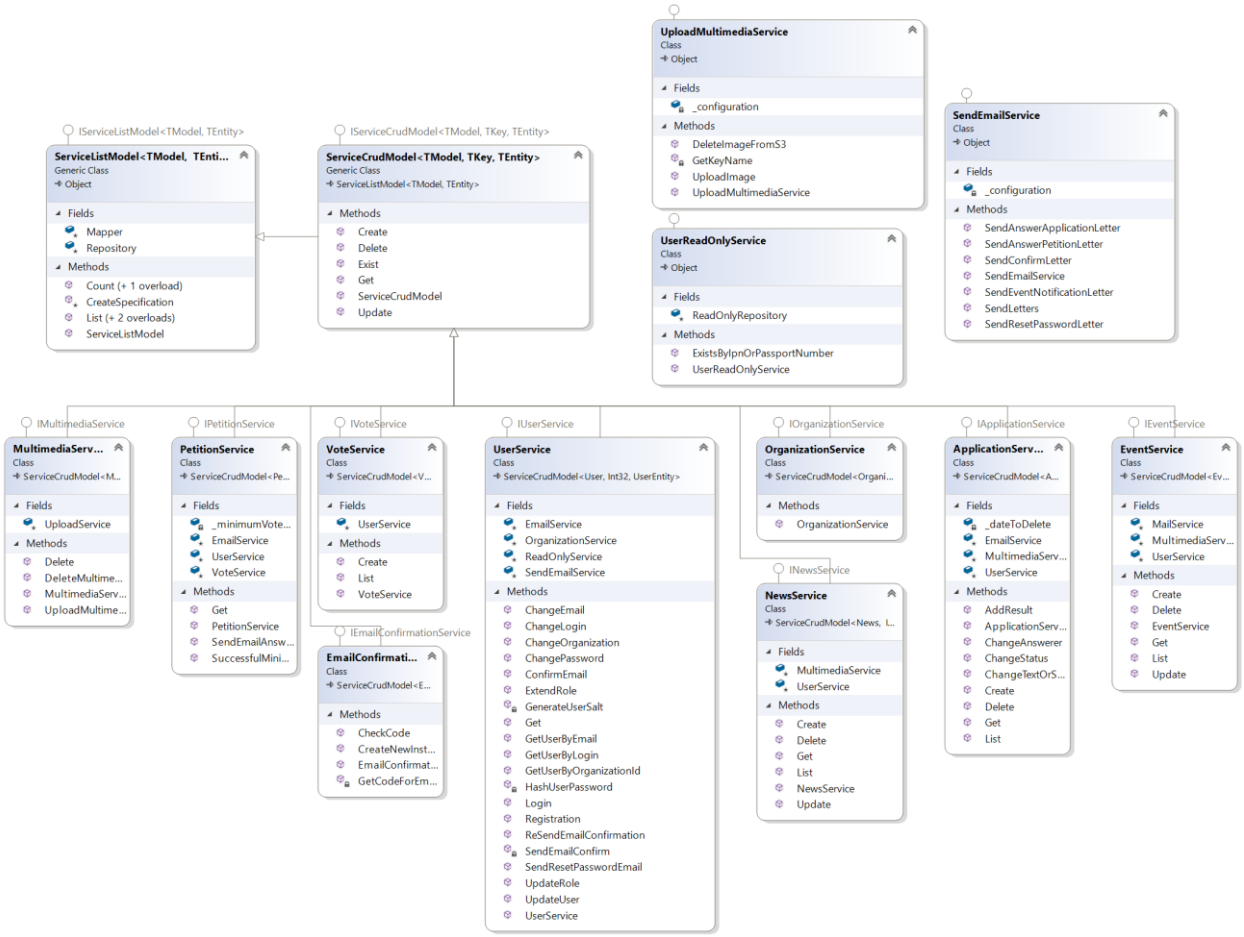


Рис. 10 Діаграма класів сервісів

Ці класи наслідують узагальнені класи ServiceListModel та ServiceCrudModel, що реалізовують методи CRUD. Для роботи з даними виконується перетворення об'єктів entity (які повертають репозиторії) в model з використанням AutoMapper. Для кожного типу сутності був створений відповідний тип моделі. Клас UserReadOnlyService відповідає за роботу з зовнішною базою даних. Клас SendEmailService створений для виконання завдання надсилання користувачам листів електронною поштою. Клас UploadMultimediaService реалізує завантаження зображень на Amazon S3, та видалення вже завантаженого зображення.

На рівні моделі предметної області також реалізовані класи що забезпечують конфігурацію AutoMapper, реєстрацію інтерфейсів та відповідним їм класів в Autofac.

3.2.1.2 Рівень застосунку

На цьому рівні реалізовано 7 класів контролерів (рис. 11), які наслідують узагальнений клас CrudControllerBase та реалізують необхідні методи.

Рис. 11 Діаграма класів контролерів

На цей рівень приходять дані від клієнта у вигляді об'єктів DTO. В середині контролера ці об'єкти за допомогою AutoMapper перетворюються в моделі та надсилаються відповідним сервісам. Сервіси надсилають результат і всередині контролера відбувається зворотне перетворення моделі в об'єкт DTO, який надсилається клієнту.

Крім того на цьому рівні реалізується автентифікація за допомогою токенів та авторизація з використанням ролей. Токени створюються за допомогою JWTSecurityToken. Для авторизації над кожним методом

контролера використовується відповідна анотація, наприклад: [AllowAnonymous] – дозволяє доступ до методу без автентифікації, а [Authorize(Roles = ApplicationAdmin, UserManager")] – дозволяє доступ до методу лише користувачам з роллю ApplicationAdmin, або UserManager.

3.2.2 Програмний інтерфейс застосунку

Застосунки Web Api Asp.Net Core 5 за замовчуванням підтримують Swagger. Swagger - це простий в використанні набір інструментів розробника API, який дозволяє розробляти протягом всього життєвого циклу API, починаючи від проектування і документації до тестування і розгортання.[21] За допомогою нього на етапі розробки було доступне зібране в одному місці API застосунку рис 12.

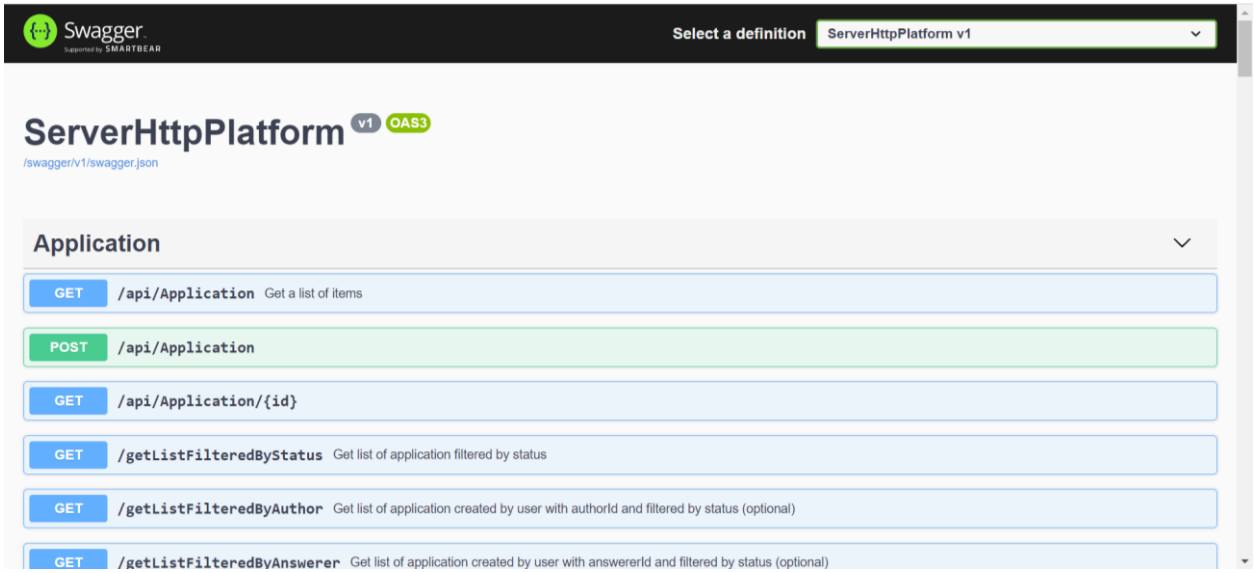


Рис. 12 API

3.2.2.1 Application

В контролері реалізовані методи Get, Put, Post. Метод Delete не передбачений. Всі методи контролеру вимагають наявності токену авторизації в заголовках запиту.

3.2.2.1.1 Get методи, що повертають список

Кожен з методів, що повертає список приймає наступні необов'язкові параметри в стрічці запиту: skip – скільки елементів пропустити, take – кількість елементів, які треба повернути, sortProp – властивість відносно якої відбувається сортування, sortOrder – порядок сортування, можливі значення: asc (за зростанням) або desc (за спаданням).

Кожен з наступних методів може повернути у випадку успіху код 200 та об'єкт, що містить два поля: результат – масив звернень та загальну кількість звернень. Якщо буде відсутній заголовок авторизації, або користувач не матиме необхідного рівню доступу, метод повертає код 401 (Неавторизовано).

GET /api/Application – повертає всі звернення.

GET /getListFilteredByStatus – повертає список звернення з вибраним статусом. В стрічці запиту повинен містити параметр status, який може приймати наступні значення 1 (Очікується), 2 (В процесі), 3 (Закрита).

GET /getListFilteredByAuthor – повертає список звернень вибраних за статусом і автором звернення. Стрічці запиту повинна містити параметри status, який може приймати наступні значення 0 (null), 1 (Очікується), 2 (В процесі), 3 (Закрита) та authorId – ідентифікатор автора звернення. Може повернути код 404 (Не знайдено), якщо автор з таким ключем не буде знайдено.

GET /getListFilteredByAnswerer – повертає список звернень за вибраним статусом і користувачем, який відповідає на звернення. Стрічці запиту повинна містити параметри status, який може приймати наступні значення 0 (null), 1 (Очікується), 2 (В процесі), 3 (Закрита) та answererId – ідентифікатор користувача, що відповідає на звернення. Може повернути код 404 (Не знайдено), якщо користувача з таким ключем не буде знайдено

3.2.2.1.2 Get метод, що повертає елемент

GET /api/Application/{id} – використовується для отримання елементу за ідентифікатором. В стрічці запиту обов'язково повинен містити ідентифікатор звернення. У випадку успіху код 200 та об'єкт звернення. Якщо буде відсутній заголовок авторизації, або користувач не матиме необхідного рівню доступу, метод повертає код 401 (Неавторизовано). Якщо звернення з таким ключем не буде знайдено поверне код 404 (Не знайдено).

3.2.2.1.3 Put методи

Кожен з наступних методів може повернути у випадку успіху код 200 та об'єкт звернення. Якщо буде відсутній заголовок авторизації, або користувач не матиме необхідного рівню доступу, метод повертає код 401 (Неавторизовано). Якщо об'єктів з заданими ключами не буде знайдено повертає код 404 (Не знайдено).

PUT /addResult/{id} – метод для додавання відповіді до звернення. Обов'язковий параметр в стрічці запиту - ідентифікатор звернення. В тілі запиту в форматі form-data необхідно передати параметр result з текстом відповіді.

PUT /changeAnswerer – метод для зміни користувача, який відповідає на звернення. В стрічці запиту повинні передаватись наступні параметри:

applicationId – ідентифікатор звернення, answererId – ідентифікатор користувача, який буде відповідати на звернення.

PUT /changeStatus/{id} – метод для зміни статусу звернення. В стрічці запиту повинні передаватись наступні параметри: id – ідентифікатор звернення, та statusDto, який може приймати наступні значення 1 (Очікується), 2 (В процесі), 3 (Закрита).

3.2.2.1.4 Post метод

POST /api/Application - метод створення нового звернення. В тілі запиту в форматі json повинен передаватись об'єкт звернення. Повертає у випадку успіху код 201 та об'єкт звернення. Якщо буде відсутній заголовок авторизації, або користувач не матиме необхідного рівню доступу, метод повертає код 401 (Неавторизовано).

3.2.2.2 Event

В контролері реалізовані методи Get, Put, Post, Delete.

3.2.2.2.1 Методи Get, що повертають список

Кожен з методів, що повертає список приймає наступні необов'язкові параметри в стрічці запиту: skip – скільки елементів пропустити, take – кількість елементів, які треба повернути, sortProp – властивість відносно якої відбувається сортування, sortOrder – порядок сортування, можливі значення: asc (за зростанням) або desc (за спаданням).

Кожен з наступних методів може повернути у випадку успіху код 200 та об'єкт, що містить два поля: результат – масив подій та загальну кількість подій.

GET /api/Event – повертає список усіх подій.

GET /filter_by_time – повертає список усіх подій фільтрованих за часом (ті що актуальні, ті що минули). В стрічці запиту повинен передавитись параметр filter, який може набувати значення active, або pass. В випадку вводу некоректного значення фільтру повертається код 400 (Поганий запит).

GET /filter_by_organization – повертає список усіх подій фільтрованих за організацією. В стрічці запиту повинен передавитись параметр organizationId – ідентифікатор організації.

GET /filterOrganizationTime – повертає список усіх подій фільтрованих за часом (ті що актуальні, ті що минули) та організацією. В стрічці запиту повинні передавитись параметри: filter, який може набувати значення active, або pass, organizationId – ідентифікатор організації. В випадку вводу некоректного значення параметрів повертається код 400 (Поганий запит).

3.2.2.2.2 Метод Get, що повертає елемент

GET /api/Event/{id} – повертає елемент за ідентифікатором. В стрічці запиту обов'язковий параметр – ідентифікатор події. У випадку успіху повертається код 200 та об'єкт події. Якщо подія з таким ключем не знайдена, повертається статус 404.

3.2.2.2.3 Метод Put

PUT /api/Event/{id} – використовується для редагування події. Вимагає наявності токену авторизації в заголовку запиту. В стрічці запиту обов'язковим параметром передається ідентифікатор події. В тілі запиту повинен передаватись об'єкт події. У випадку успіху повертається код 200 та об'єкт події. Якщо подія з таким ключем не знайдена, повертається статус 404. Якщо буде відсутній заголовок авторизації, або користувач не матиме необхідного рівню доступу, метод повертає код 401.

3.2.2.2.4 Метод Post

POST /api/Event – використовується для створення події. Вимагає наявності токену авторизації в заголовку запиту. В тілі запиту повинен передаватись об'єкт події. У випадку успіху повертається код 201 та об'єкт події. Якщо буде відсутній заголовок авторизації, або користувач не матиме необхідного рівню доступу, метод повертає код 401.

3.2.2.2.5 Метод Delete

PUT /api/Event/{id} – використовується для видалення події. Вимагає наявності токену авторизації в заголовку запиту. В стрічці запиту обов'язковим параметром передається ідентифікатор події. У випадку успіху повертається код 204. Якщо буде відсутній заголовок авторизації, або користувач не матиме необхідного рівню доступу, метод повертає код 401.

3.2.2.3 News

В контролері реалізовані методи Get, Put, Post, Delete.

3.2.2.3.1 Методи Get, що повертають список

Кожен з методів, що повертає список приймає наступні необов'язкові параметри в стрічці запиту: skip – скільки елементів пропустити, take – кількість елементів, які треба повернути, sortProp – властивість відносно якої відбувається сортування, sortOrder – порядок сортування, можливі значення: asc (за зростанням) або desc (за спаданням).

Кожен з наступних методів може повернути у випадку успіху код 200 та об'єкт, що містить два поля: результат – масив новин та загальну кількість новин.

GET /api/News – повертає список всіх новин.

GET /news_by_organization – повертає список всіх новин фільтрованих за обраною організацією. У стрічці запиту повинен містити параметр organizationId – ідентифікатор організації.

3.2.2.3.2 Метод Get, що повертає елемент

GET /api/News/{id} – повертає елемент за ідентифікатором. У стрічці запиту повинен містити ідентифікатор новини. У випадку успіху повертається код 200 та об'єкт новини. Якщо новина з таким ключем не знайдена, повертається статус 404.

3.2.2.3.3 Метод Put

PUT /api/News/{id} – використовується для редагування новини. Вимагає наявності токену авторизації в заголовку запиту. В стрічці запиту обов'язковим параметром передається ідентифікатор новини. В тілі запиту повинен передаватись об'єкт новини. У випадку успіху повертається код 200 та об'єкт події. Якщо новина з таким ключем не знайдена, повертається статус 404. Якщо буде відсутній заголовок авторизації, або користувач не матиме необхідного рівню доступу, метод повертає код 401.

3.2.2.3.4 Метод Post

POST /api/News – використовується для створення новини. Вимагає наявності токену авторизації в заголовку запиту. В тілі запиту повинен передаватись об'єкт новини. У випадку успіху повертається код 201 та об'єкт новини. Якщо буде відсутній заголовок авторизації, або користувач не матиме необхідного рівню доступу, метод повертає код 401.

3.2.2.3.5 Метод Delete

DELETE /api/News/{id} – використовується для видалення новини. Вимагає наявності токену авторизації в заголовку запиту. В стрічці запиту обов'язковим параметром передається ідентифікатор новини. У випадку

успіху повертається код 204. Якщо буде відсутній заголовок авторизації, або користувач не матиме необхідного рівню доступу, метод повертає код 401.

3.2.2.4 Organization

В контролері реалізовані методи Get, Put, Post, Delete

3.2.2.4.1 Методи Get

GET /api/Organization/{id} – повертає елемент за ідентифікатором. У стрічці запиту повинен містити ідентифікатор організації. У випадку успіху повертається код 200 та об'єкт організації. Якщо організація з таким ключем не знайдена, повертається статус 404.

GET /api/Organization – повертає всіх список організацій. Приймає наступні необов'язкові параметри в стрічці запиту: skip – скільки елементів пропустити, take – кількість елементів, які треба повернути, sortProp – властивість відносно якої відбувається сортування, sortOrder – порядок сортування, можливі значення: asc (за зростанням) або desc (за спаданням). Кожен з наступних методів може повернути у випадку успіху код 200 та об'єкт, що містить два поля: результат – масив організацій та загальну кількість організацій.

3.2.2.4.2 Метод Put

PUT /api/Organization/{id} – використовується для редагування організації. Вимагає наявності токену авторизації в заголовку запиту. В стрічці запиту обов'язковим параметром передається ідентифікатор організації. В тілі запиту повинен передаватись об'єкт організації. У випадку успіху повертається код 200 та об'єкт організації. Якщо організація з таким ключем не знайдена, повертається статус 404. Якщо буде відсутній заголовок

авторизації, або користувач не матиме необхідного рівню доступу, метод повертає код 401.

3.2.2.4.3 Метод Post

POST /api/Organization – використовується для створення організації. Вимагає наявності токену авторизації в заголовку запиту. В тілі запиту повинен передаватись об'єкт організації. У випадку успіху повертається код 201 та об'єкт організації. Якщо буде відсутній заголовок авторизації, або користувач не матиме необхідного рівню доступу, метод повертає код 401.

3.2.2.4.4 Метод Delete

DELETE /api/Organization/{id} – використовується для видалення організації. Вимагає наявності токену авторизації в заголовку запиту. В стрічці запиту обов'язковим параметром передається ідентифікатор організації. У випадку успіху повертається код 204. Якщо буде відсутній заголовок авторизації, або користувач не матиме необхідного рівню доступу, метод повертає код 401.

3.2.2.5 Petition

В контролері реалізовані методи Get, Put, Post. Метод Delete не передбачений.

3.2.2.5.1 Методи Get, що повертають список

Кожен з методів, що повертає список приймає наступні необов'язкові параметри в стрічці запиту: skip – скільки елементів пропустити, take – кількість елементів, які треба повернути, sortProp – властивість відносно якої відбувається сортування, sortOrder – порядок сортування, можливі значення: asc (за зростанням) або desc (за спаданням). Кожен з наступних методів може

повернути у випадку успіху код 200 та об'єкт, що містить два поля: результат – масив петицій та загальну кількість петицій.

GET /api/Petition – повертає список усіх петицій.

GET /filter_by_status – повертає список петицій фільтрованих за статусом. В стрічці запиту повинен містити параметри: timeStatus (допустимі значення: active, act, close, cls) та votesStatus (допустимі значення: successful та unsuccessful).

GET /filter_by_author – повертає список петицій фільтрованих за автором. В стрічці запиту повинен містити параметр userId з ідентифікатором автора.

GET /filter_author_status – повертає список петицій фільтрованих за автором і статусом. В стрічці запиту повинен містити наступні параметри: timeStatus (допустимі значення: active, act, close, cls), votesStatus (допустимі значення: successful та unsuccessful) та userId з ідентифікатором автора.

3.2.2.5.2 Методи Get, що повертають значення

GET /api/Petition/{id} – повертає елемент за ідентифікатором. В стрічці запиту повинен містити ідентифікатор петиції. В випадку успіху повертає код 200 і об'єкт петиції. Якщо петицію з таким ключем не знайдено – повертає 404.

GET /api/Petition/minimum – повертає значення найменшої кількості необхідних для петиції голосів

3.2.2.5.3 Метод Put

PUT /api/Petition/addAnswer - додає відповідь до петиції. Вимагає наявності токену авторизації в заголовку запиту. В стрічці запиту має бути параметр id - ідентифікатор петиції. В випадку успіху повертає код 200 та об'єкт петиції. Якщо буде відсутній заголовок авторизації, або користувач не матиме необхідного рівню доступу, метод повертає код 401. Якщо петиція не набрала необхідної кількості голосів, метод повертає код 403 (Заборонено).

Якщо петиція з даним ключем не знайдена, метод повертає код 404. Якщо відповідь вже була надана, метод повертає код 409 (Конфлікт).

3.2.2.5.4 Метод Post

POST /api/Petition – використовується для створення петиції. Вимагає наявності токену авторизації в заголовку запиту. В тілі запиту повинен передаватись об'єкт петиції. У випадку успіху повертається код 201 та об'єкт петиції. Якщо буде відсутній заголовок авторизації, або користувач не матиме необхідного рівню доступу, метод повертає код 401.

3.2.2.6 User

В контролері реалізовані методи Get, Put Post та Delete.

3.2.2.6.1 Методи Get

GET /api/User – використовується для отримання списку всіх користувачів. Вимагає наявність токену авторизації в заголовках запиту. В стрічці запиту можна передати необов'язкові параметри: skip, take, sortProp, sortOrder. В випадку успіху повертає код 200 та об'єкт, що містить два поля: результат - масив користувачів та загальну кількість користувачів. Якщо токен авторизації відсутній, або користувач не має необхідних прав, метод повертає код 401.

GET /api/get_user_by_organization – використовується для отримання списку всіх користувачів заданої організації. Вимагає наявність токену авторизації в заголовках запиту. В стрічці запиту необхідно передати параметр id – ідентифікатор організації. В випадку успіху повертає код 200 та масив користувачів організації. Якщо токен авторизації відсутній, або користувач не має необхідних прав, метод повертає код 401. Якщо організація з заданим id не знайдена – повертається код 404.

GET /filtered_by_role – використовується для отримання списку всіх користувачів, які належать певній ролі. Вимагає наявності токену авторизації. В стрічці запиту необхідно передати параметр role – назву ролі. Є можливість передати необов'язкові параметри: skip, take, sortProp, sortOrder. В випадку успіху повертає код 200 та об'єкт, що містить два поля: результат – масив користувачів та загальну кількість користувачів, що належать обраній ролі. Якщо токен авторизації відсутній, або користувач не має необхідних прав, метод повертає код 401.

GET /api/User/{id} – використовується для отримання об'єкта користувача за ідентифікатором. Вимагає наявності токену авторизації в заголовках запиту. В стрічці запиту обов'язковим параметром є ідентифікатор користувача. У випадку успіху повертає об'єкт користувача та код 200. Якщо заголовок авторизації відсутній, або токен невалідний метод повертає код 401. Якщо користувач не знайдений: 404.

GET /confirm_email – використовується для підтвердження пошти користувача. В стрічці запиту необхідно передати параметри id – ідентифікатор користувача і code – код підтвердження пошти. У випадку успіху повертає код 200. Якщо код або ідентифікатор неправильні – 404.

GET /emailConfirmResend – використовується для повторного надсилання коду підтвердження на пошту. В стрічку запиту необхідно передати параметр email – електронну пошту, на яку мав прийти код підтвердження. У випадку успіху повертає код 200. Якщо запису на підтвердження такої пошти не знайдено повертається код 404.

3.2.2.6.2 Методи Put

PUT /user/check – використовується для перевірки валідності токена. Токен авторизації передається в заголовку запиту. Якщо токен валідниий повертається код 200, якщо ні – 401.

PUT /change_role – використовується для зміни ролі користувача. Вимагає наявності токена авторизації в заголовку запиту. В стрічці запиту необхідно передати userId – ідентифікатор користувача. В тілі запиту – об'єкт користувача з відредагованими ролями. У випадку успіху повертається код 200 та об'єкт користувача. Якщо токен авторизації відсутній, або користувач не має необхідних прав, метод повертає код 401. Якщо користувач не знайдений – код 404.

PUT /update – використовується для оновлення імені, по-батькові, прізвища та номера телефону користувача. Вимагає наявності токена авторизації в заголовку запиту. В стрічці запиту необхідно передати userId - ідентифікатор користувача. В тілі запиту – об'єкт користувача з відредагованими полями. У випадку успіху повертається код 200 та об'єкт користувача. Якщо токен авторизації відсутній, або користувач не має необхідних прав, метод повертає код 401. Якщо користувач не знайдений – код 404.

PUT /updateOrganization – використовується для зміни організації користувача. Вимагає наявності токена авторизації в заголовку запиту. В стрічці запиту необхідно передати userId – ідентифікатор користувача та organization – ідентифікатор організації. У випадку успіху повертається код 200 та об'єкт користувача. Якщо токен авторизації відсутній, або користувач не має необхідних прав, метод повертає код 401. Якщо користувач або організація не знайдені – код 404.

PUT /update_email – використовується для зміни електронної пошти користувача. Вимагає наявності токена авторизації в заголовку запиту. В стрічці запиту необхідно передати userId – ідентифікатор користувача. В тілі запиту - об'єкт користувача з відредагованими полем. У випадку успіху повертається код 200 та об'єкт користувача. Якщо токен авторизації відсутній, або користувач не має необхідних прав, метод повертає код 401. Якщо користувач не знайдений – код 404.

PUT /change_login – використовується для зміни логіну користувача. Вимагає наявності токена авторизації в заголовку запиту. В стрічці запиту необхідно передати userId – ідентифікатор користувача. В тілі запиту – об'єкт користувача з відредагованими полем. У випадку успіху повертається код 200 та об'єкт користувача. Якщо токен авторизації відсутній, або користувач не має необхідних прав, метод повертає код 401. Якщо користувач не знайдений – код 404. Якщо логін вже зайнято – 409 (Конфлікт).

PUT /change_password – використовується для зміни пароля користувача. Вимагає наявності токена авторизації в заголовку запиту. В стрічці запиту необхідно передати userId – ідентифікатор користувача. В тілі запиту – об'єкт користувача з відредагованими полем. У випадку успіху повертається код 200 та об'єкт користувача. Якщо токен авторизації відсутній, або користувач не має необхідних прав, метод повертає код 401. Якщо користувач не знайдений – код 404.

3.2.2.6.3 Методи Post

POST /login – використовується для отримання токену авторизації. В тілі запиту повинен передаватись об'єкт з двома полями: login – логін користувача та password – пароль користувача. В випадку успіху повертається код 200 з об'єктом що містить токен авторизації та об'єкт користувача. Якщо користувач з таким логіном та паролем не знайдений – код 404.

POST /forgot_password - використовується для відновлення паролю. В стрічці запиту передається параметр email. В випадку успіху повертається код 200, а на вказаний емейл надсилається посилання за яким можна скинути пароль. Якщо такий емейл не зареєстрований – код 404.

POST /registration – використовується для реєстрації нового користувача. В тілі запиту передається об'єкт користувача. У випадку успіху повертається код 200 та об'єкт користувача. Якщо користувач з такою поштою або логіном вже існує – код 409.

POST /loginExists – використовується для перевірки чи такий логін зареєстрований. Код 200 – якщо логін зареєстрований, код 404 – якщо ні.

POST /emailExists – використовується для перевірки чи така пошта вже зареєстрований. Код 200 – якщо пошта зареєстрований, код 404 – якщо ні.

POST /extendRole – використовується для розширення прав користувача. Вимагає наявність токену авторизації в заголовку запита. В стрічці запиту передається параметр userId – ідентифікатор користувача. В тілі запиту в форматі form-data необхідно передати два параметри: inpOrPass – номер паспорту або номеру облікової книжки платника податків та isIpn – булеве значення, що вказує чи передається реєстраційний номер облікової книжки платника податків.

3.2.2.6.4 Метод Delete

DELETE /api/User/{id} – використовується для видалення облікового запису користувача. Вимагає наявність токену авторизації в заголовку запита. В стрічці запиту необхідно передати ідентифікатор користувача. У випадку успіху повертається код 204. Якщо токен авторизації відсутній, або користувач не має необхідних прав, метод повертає код 401.

3.2.2.7 Votes

В контролері реалізовані методи GET, POST та DELETE. Метод PUT не передбачено.

3.2.2.7.1 Методи Get

GET /votes_number – повертає кількість голосів за певну петицію. В стрічці запиту необхідно вказати параметр petitionId – ідентифікатор петиції. У випадку успіху повертає код 200 і об'єкт з полем count, що містить кількість голосів

3.2.2.7.2 Метод Post

POST /api/Votes – використовується для створення голосу за петицію. Вимагає токену авторизації в заголовку запиту. В стрічці запиту необхідно передати userId – ідентифікатор користувача та petitionId – ідентифікатор петиції. У випадку успіху повертає код 201 та об'єкт голосу. Якщо токен авторизації відсутній, або користувач не має необхідних прав, метод повертає код 401.

3.2.2.7.3 Метод Delete

DELETE /api/Votes/{id} – використовується для видалення голосу за петицію. Вимагає токену авторизації в заголовку запиту. В стрічці запиту необхідно передати id – ідентифікатор голосу. У випадку успіху повертає код 204. Якщо токен авторизації відсутній, або користувач не має необхідних прав, метод повертає код 401.

3.3 Реалізація клієнтської частини

3.3.1 Структура клієнтського застосунку

Фреймворк Vue.js дозволяє реалізовувати складні односторінкові структури. В даній роботі кореневим компонентом є компонет App. Всі інші компоненти знаходяться в середині нього. Спрощена ієрархія компонентів зображена на рис. 13, а більш детальна ієрархія розміщена в Додатку А.

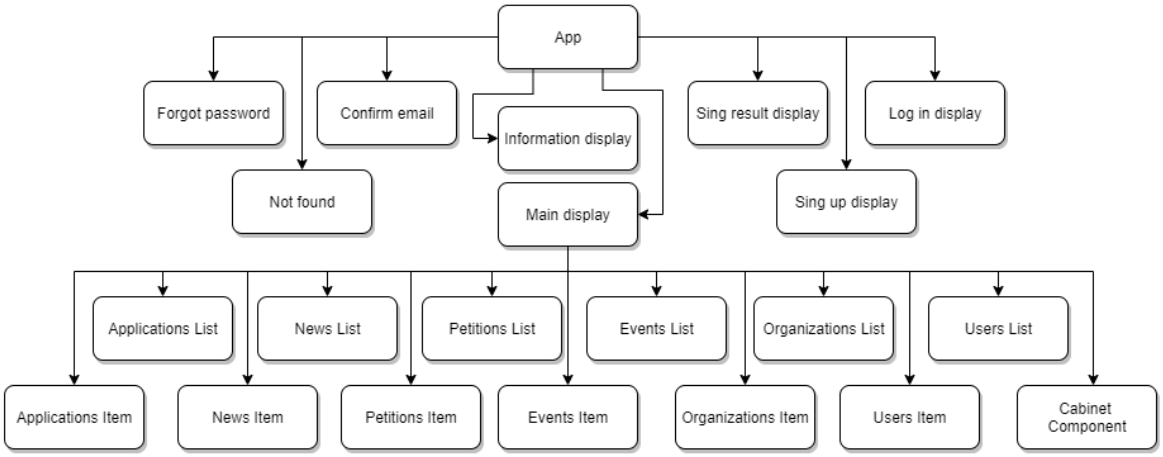


Рис. 13 Ієрархія компонентів клієнтської частини

В компоненті App підключений компонент, що відповідає за меню відображене зверху сторінки (Navigation header). На наступному рівні дочірніми компонентами кореневого є головний контекст (Main display), інформаційна сторінка (Information display), сторінка входу (Login display), сторінка реєстрації (Sing up display), сторінка підтвердження пошти (Confirm email) і сторінка відновленню пароля (Forgot Password). В компоненті головного контексту знаходять дочірні компоненти, що відповідають за відображення списків та інформації про окремий елемент користувачів, новин, подій, петицій, звернень та організацій.

Для задання заголовків сторінок з програмного коду використовувалась бібліотека vue-meta. [22] А для наочного відображення кількості набраних голосів в петиціях використовувався компонент vue-circle. [23]

3.3.2 Опис інтерфейсу

3.3.2.1 Вхід та реєстрація

На сторінці входу відображається форма для вводу емейлу та паролю. Є можливість перейти на сторінку реєстрації. Якщо користувач не пам'ятає

пароль, є можливість перейти на іншу сторінку для його відновлення. Також можна перейти на головну та інформаційну сторінки.

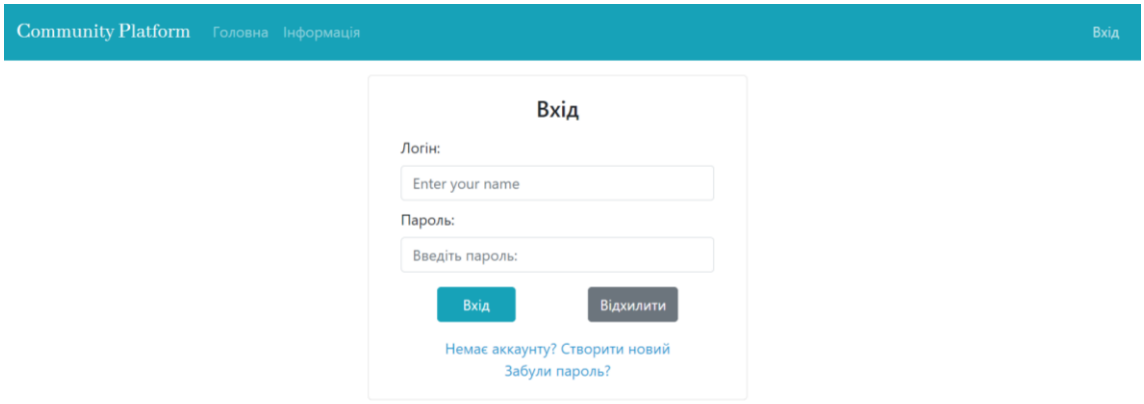


Рис. 14 Інтерфейс сторінки входу

На сторінці реєстрації користувачу пропонується ввести ім'я, по-батькові, прізвище, номер телефону, електронну пошту, логін та пароль. Обов'язкові поля позначені зірочкою. Є можливість перейти на сторінку входу, головну та інформаційну сторінки.

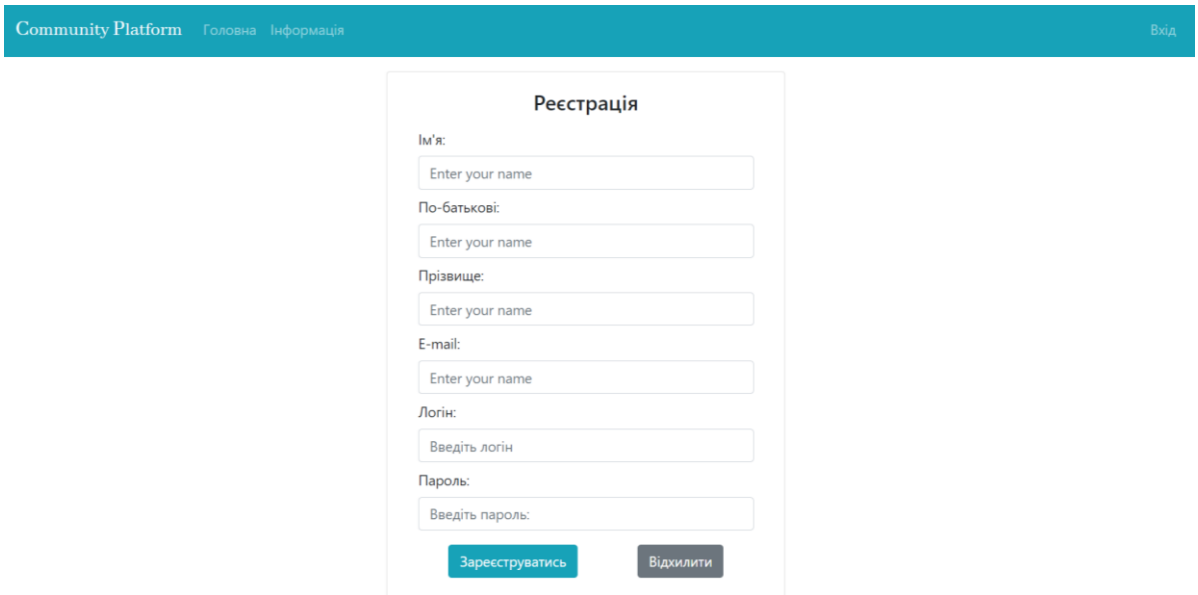


Рис. 15 Інтерфейс сторінки реєстрації

3.3.2.2 Новини

Дана сторінка є головною сторінкою веб-платформи. На ній з лівого боку відображаються доступні користувачеві сторінки сайту. По центру відображається список новин.

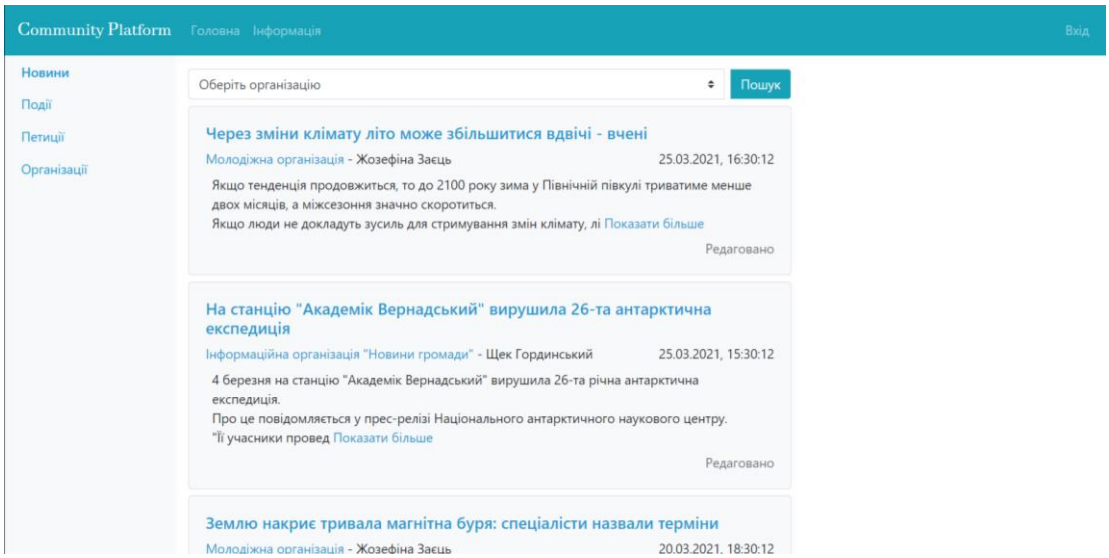


Рис. 16 Інтерфейс сторінки новин

В кожній новині списку виводиться заголовок, дата створення, зображення, організація, може виводитись автор. Текст новини відображається в скороченому вигляді. Зверху сторінки знаходиться поле для фільтрації новин. Якщо користувач має відповідні права зверху сторінки також відображається кнопка для створення нової новини.

3.3.2.3 Події

На сторінці подій по центру зверху знаходиться форма для фільтрації та форма створення нової події (якщо користувач має необхідні права). Нижче відображається список подій.

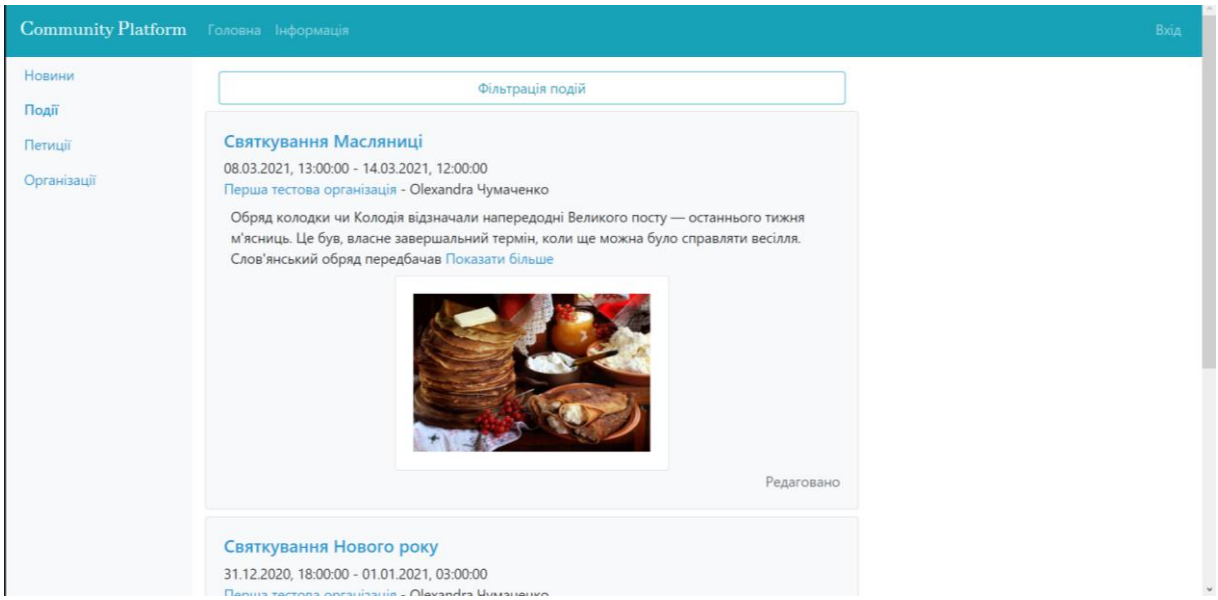


Рис. 17 Інтерфейс сторінки подій

Для кожної події зображено її назву, дату коли вона буде проводитись, зображення, скорочений текст, організацію, та може виводитись автор події.

3.3.2.4 Петиції

Зверху сторінки знаходиться форма для фільтрації петицій і форма створення петицій, якщо користувач має на це права. Нижче знаходиться список петицій. Для кожної петиції за списку відображається її тема, статус, кількість голосів, дата створення.

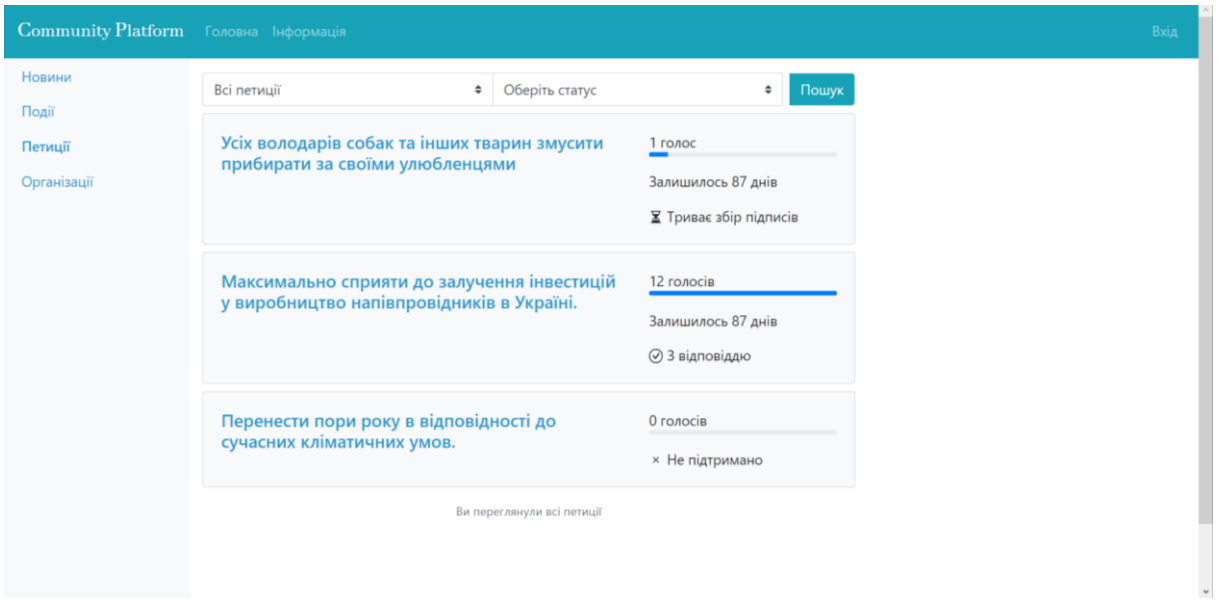


Рис. 18 Інтерфейс сторінки петицій

3.3.2.5 Звернення

Для користувачів з необхідними правами зверху сторінки знаходиться форма для створення нових звернень. Також в верхній частині сторінки знаходиться форма для фільтрації звернень. Під формою відображається список всіх звернень.

Для кожного звернення відображається його автор, час коли звернення було подане, статус звернення, тема звернення та автор і час відповіді, якщо відповідь була надана.

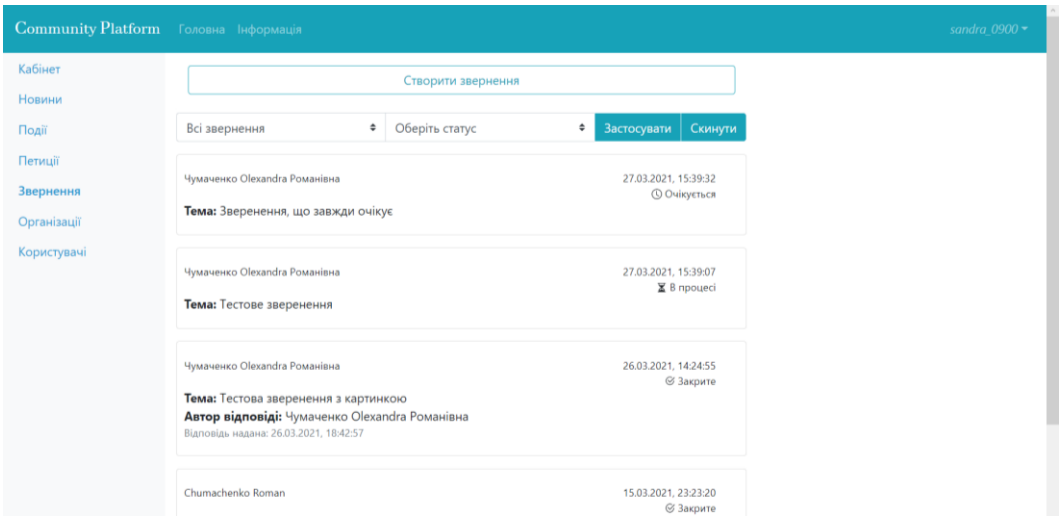


Рис. 19 Інтерфейс сторінки звернень

3.3.2.6 Користувачі

Зверху сторінки відображається форма для фільтрування користувачів за рівнем доступу. Під формою знаходиться список користувачів. Для кожного користувача відображається ім'я та прізвище, логін, електронна пошта, чи є пошта підтвердженою та спрощене представлення ролі користувача.

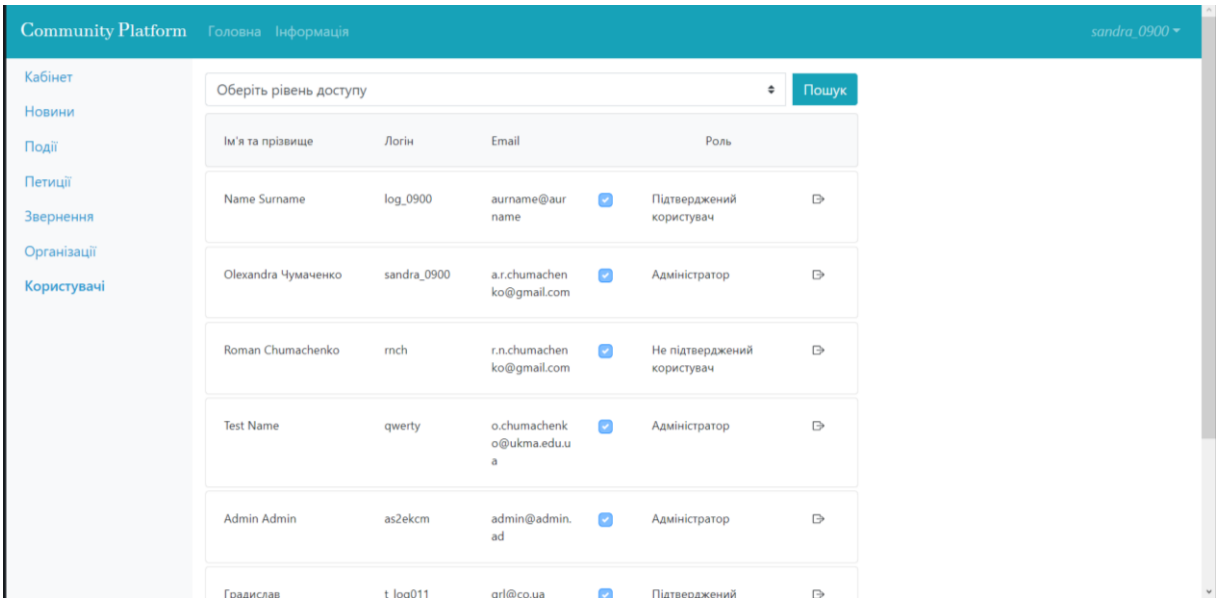


Рис. 20 Інтерфейс сторінки користувачів

3.3.2.7 Кабінет користувача

В кабінеті користувача відображається інформація про поточного користувача, є кнопка для зміни логіну, паролю, розширенню прав (якщо це ще не зроблено) та видаленню облікового запису.

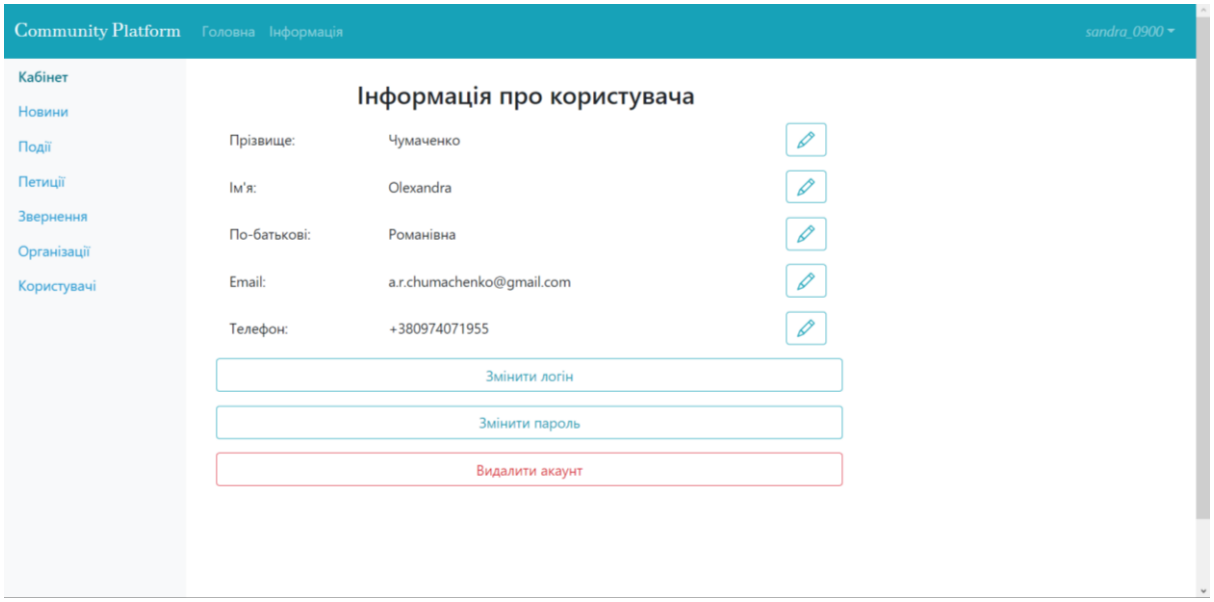


Рис. 21 Інтерфейс кабінету користувача

3.4 Розгортання застосунку

Для зберігання зображень було використано Amazon S3, а бази даних, серверна та клієнтська частина були розміщені на Heroku. Для розміщення серверної частини використовувався Docker. Для підключення до баз даних було використані змінні середовища, що надає Heroku при підключення бази даних до застосунку. Це було використано тому, що платформа зазначає, що періодично змінює ім'я користувача, бази даних та пароль і автоматично міняє ці дані в змінній середовища.

В результаті розгортання застосунок доступний за адресою: https://communication-platform.herokuapp.com

3.5 Тестування програмного продукту

Тестування програмного продукту відбувалося вручну в два етапи. На першому етапі перевірялась відповідність застосунку до визначених функціональних вимог застосунку. На другому етапі проводилось тестування з різними варіантами вводу, для виявлення неопрацьованих помилок. Бази даних наповнювались тестовими даними програмно. Всі тестові дані були визначені в класі Program та додані до баз даних. Після тестування дані не видалялись для забезпечення подальшої демонстрації. Тестові дані користувачів наведені в додатку Б.

Висновок

Впродовж реалізації практичної частини курсової роботи було виконано ознайомлення з проблемно-орієнтованим проектування, патернами репозиторій та специфікація та отримано практичний досвід в написанні програм використовуючи технології .Net на сервері, та фреймворк Vue.js на клієнтській стороні.

Результатом курсової роботи є розгорнутий веб-застосунок - платформа для комунікації з місцевою владою.

В застосунку реалізована можливість створення, перегляду та редагування новин та подій, голосування петицій, обробка звернень, додавання та видалення організацій, створення оповіщень щодо події, відповідей на звернення та петицію, передбачена реєстрації, автентифікація та авторизація, що базується на ролях.

Подальше покращення веб-платформи можливе наступними шляхами:

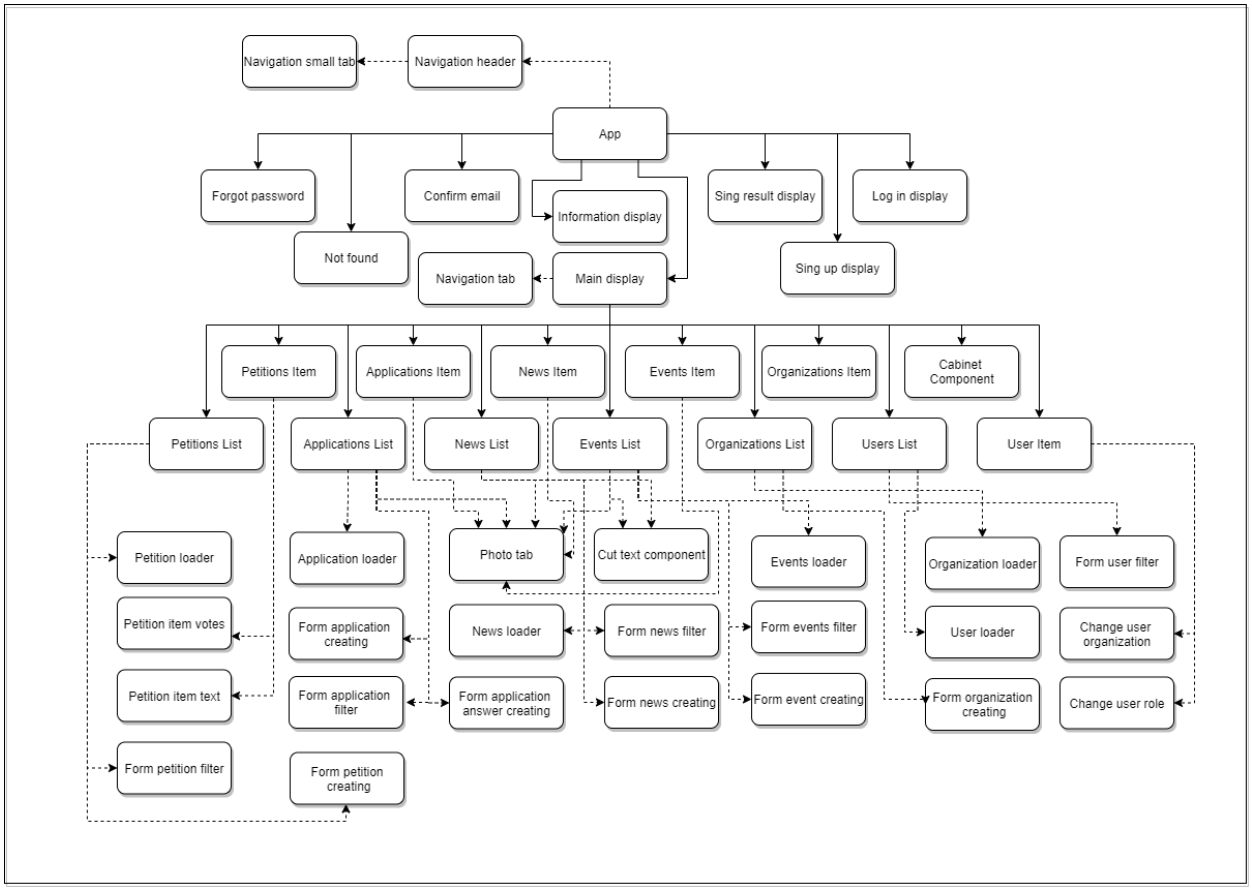
- впровадження СМС оповіщення користувачів;
- впровадження створення довільних оповіщень користувачів;
- розширення варіантів підтвердження факту проживання на території громади.

Список використаних джерел

1. https://zp.gov.ua/uk
2. https://petition.president.gov.ua
3. https://zakon.rada.gov.ua/laws/show/393/96-вр#n62
4. https://www.keycdn.com/blog/popular-databases
5. https://docs.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-5.0
6. https://docs.microsoft.com/en-us/ef/core/
7. https://metanit.com/sharp/entityframework/1.1.php
8. https://autofaccn.readthedocs.io/en/latest/getting-started/index.html
9. https://automapper.org
10. https://webpack.js.org
11. https://docs.npmjs.com/about-npm
12. https://vuejs.org/v2/guide/
13. https://vuex.vuejs.org
14. https://router.vuejs.org
15. https://docs.microsoft.com/en-us/ef/core/managing-schemas/migrations/?tabs=vs
16. https://docs.microsoft.com/en-us/dotnet/architecture/microservices/microservice-ddd-cqrs-patterns/infrastructure-persistence-layer-design#:~:text=of%20Work%20patterns.-,The%20Repository%20pattern,from%20the%20domain%20model%20layer
17. Martin Fowler Patterns of Enterprise Application Architecture
18. https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ff649690(v=pandp.10)
19. https://medium.com/@rudyzio92/net-core-using-the-specification-pattern-alongside-a-generic-repository-318cd4eea4aa

20. https://docs.microsoft.com/en-us/dotnet/architecture/microservices/microservice-ddd-cqrs-patterns/ddd-oriented-microservice
21. https://swagger.io/about/
22. https://vue-meta.nuxtjs.org
23. https://github.com/vrajroham/vue-circle-progress

Додаток А. Діаграма компонентів клієнтської частини.



Додаток Б. Тестові дані користувачів

Інформація про користувачів з основної бази даних:

Id	Login	Password	Role
33	as2ekcm	qwRnvod23_ric	All roles
34	t_log011	1W3e5p78	User, SuperUser
35	t_log012	1Z3e5p79	User, SuperUser
36	t_log013	1Q3e5p80	User, SuperUser
37	t_log014	1S3e5p81	User, SuperUser
38	t_log015	1A3e5p82	User, SuperUser, UserManager
39	t_log016	1Y3e5p83	User
40	t_log0162	1Y3e5p83	User
41	t_log017	1R3e5p84	User, SuperUser
42	t_log018	1SZ3e5p85	User, SuperUser, NewsAndEvents
43	t_log019	1pP3e5p86	User, SuperUser
44	t_log020	1Y3e5p87	User, SuperUser, NewsAndEvents
46	t_log021	1i3e5p88	User, SuperUser
47	ajp_29	nsqEnnc_90	User, SuperUser, Moderator
58	t_log022	rtPTmmwl2	User, SuperUser, ApplicationAdmin

Інформація про користувачів з зовнішньої бази даних:

Логін користувача, якому належать дані	Номер паспорта	Реєстраційний номер облікової книжки платника податків
t_log0162	СЮ124586	3457824840
t_log016	СЮ124584	3457811110