

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

**«Багатофункціональний міський портал взаємодії
жителів з муніципалітетом»**

Виконав студент групи ПЗ-22-1
КОРЕЦЬКИЙ Юрій Ігорович

Керівник роботи:
САВІЦЬКИЙ Роман Святославович

Рецензент:
БАКАЛЮК Тетяна Миколаївна

Житомир – 2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення
Тетяна Вакалюк

«28» лютого 2026 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **КОРЕЦЬКИЙ Юрій Ігорович**

Керівник роботи: **САВІЦЬКИЙ Роман Святославович**

Тема роботи: **«Багатофункціональний міський портал взаємодії жителів з муніципалітетом»**,

затверджена Наказом закладу вищої освіти від **№ 92/с від 27 лютого 2026р.**

Вихідні дані для роботи: **процес організації взаємодії жителів міста з муніципалітетом, що охоплює етапи взаємодії користувачів з веборієнтованою платформою, включаючи подання та опрацювання заявок на вирішення міських проблем, отримання офіційних сповіщень і новин та участь в опитуваннях і голосуваннях.**

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Савіцький Р.С.	02.03.2026р.	02.03.2026р
2	Савіцький Р.С.	07.04.2026р	07.04.2026р
3	Савіцький Р.С.	04.05.2026р	04.05.2026р

РЕФЕРАТ

Кваліфікаційна робота бакалавра складається з програмного комплексу вебдодатку багатофункціонального міського порталу взаємодії жителів з муніципалітетом та пояснювальної записки. Пояснювальна записка до кваліфікаційної роботи містить: список використаних джерел з 44 найменувань; в роботі наведено 49 рисунків та 34 таблиці; загальний обсяг роботи складає 116 сторінок, з них 113 сторінок основного тексту.

Метою роботи є розробка веборієнтованої платформи для взаємодії жителів міста з муніципалітетом, спрямованої на забезпечення прозорості, оперативності та ефективності вирішення міських проблем із використанням сучасних технологій та засобів розробки.

У роботі було визначено функціональні та нефункціональні вимоги до платформи, а також обґрунтовано вибір архітектури системи. Також було проведено аналіз існуючих рішень у сфері електронної демократії та громадянської участі й обрано відповідні інструменти для реалізації серверної та клієнтської частин. Розроблено структуру бази даних для ефективного зберігання та обробки заявок громадян, сповіщень, новин та результатів голосувань.

У першому розділі проведено аналіз предметної області та існуючих аналогів платформ електронної демократії й громадянської участі («Київ Цифровий», «Дія», FixMyStreet, Decidim), за результатами якого обґрунтовано доцільність розробки власної системи. Визначено функціональні та нефункціональні вимоги до платформи, спроєктовано її архітектуру на основі клієнт-серверної моделі та модульного моноліту з багатоорендною організацією, а також обґрунтовано вибір технологій і засобів реалізації серверної та клієнтської частин.

					ІПЗ.КР.Б – 121 – 26 – ПЗ						
Змн.	Арк.	№ докум.	Підпис	Дата							
Розроб.		Ю.І. Корецький			Багатофункціональний міський портал взаємодії жителів з муніципалітетом			Лім.	Арк.	Аркуші	
Керівник.		Р.С. Савіцький								3	116
								Житомирська політехніка, Група ІПЗ-22-1			
Рецензент		Т.М. Вакалюк									
Зав. каф.		Т.М. Вакалюк									

У другому розділі виконано детальне проєктування системи: визначено акторів і варіанти використання, побудовано діаграми варіантів використання, класів та діяльності. Спроектовано структуру реляційної бази даних PostgreSQL, описано основні алгоритми роботи системи із застосуванням сучасних патернів проєктування, а також реалізовано функціональну частину серверного та клієнтського застосунків.

У третьому розділі описано порядок встановлення й налаштування системи, наведено діаграму розгортання та перелік параметрів конфігурації. Розглянуто структуру інтерфейсу вебдодатку й порядок роботи з ним для різних ролей користувачів, а також проведено тестування платформи з використанням автоматизованих та ручних методів, що підтвердило її працездатність і відповідність поставленим вимогам.

КЛЮЧОВІ СЛОВА: ВЕБОРІЄНТОВАНА ПЛАТФОРМА, КЛІЄНТ-СЕРВЕРНА АРХІТЕКТУРА, БАГАТОРІВНЕВА АРХІТЕКТУРА, БАГАТООПЕНДНІСТЬ, REST API, WEBSOCKET, РОЛЬОВА МОДЕЛЬ ДОСТУПУ (RBAC), NESTJS, REACT, PRISMA ORM, POSTGRESQL, ВЗАЄМОДІЯ ЖИТЕЛІВ З МУНІЦИПАЛІТЕТОМ, ЕЛЕКТРОННА ДЕМОКРАТІЯ.

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				
Змн.	Арк.	№ докум.	Підпис	Дата		4

ABSTRACT

The bachelor's qualification work consists of a software complex – a web application of a multifunctional urban portal for residents' interaction with the municipality – and an explanatory note. The explanatory note contains a list of 44 references; the work includes 49 figures and 34 tables; the total volume of the work is 116 pages, of which 113 pages are the main text.

The aim of the work is to develop a web-oriented platform for interaction between city residents and the municipality, aimed at ensuring transparency, responsiveness, and efficiency in resolving urban issues using modern technologies and development tools.

The work defines the functional and non-functional requirements for the platform and justifies the choice of system architecture. An analysis of existing solutions in the field of e-democracy and civic participation was carried out, and appropriate tools were selected for implementing the server and client parts. A database structure was developed for the efficient storage and processing of citizens' requests, alerts, news, and voting results.

The first chapter analyzes the subject area and existing analogues of e-democracy and civic participation platforms ("Kyiv Digital", "Diia", FixMyStreet, Decidim), based on which the feasibility of developing a custom system is substantiated. The functional and non-functional requirements for the platform are defined, its architecture is designed on the basis of a client-server model and a modular monolith with a multitenant organization, and the choice of technologies and tools for implementing the server and client parts is justified.

The second chapter presents the detailed design of the system: the actors and use cases are identified, and use-case, class, and activity diagrams are constructed. The structure of the relational PostgreSQL database is designed, the main algorithms of the system are described using modern design patterns, and the functional part of the server and client applications is implemented.

The third chapter describes the procedure for installing and configuring the system, provides a deployment diagram and a list of configuration parameters. The

		Ю.І. Корецький			ІІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Р.С. Савіцький				5
Змн.	Арк.	№ докум.	Підпис	Дата		

structure of the web application interface and the order of working with it for different user roles are examined, and the platform is tested using automated and manual methods, which confirmed its operability and compliance with the stated requirements.

KEYWORDS: WEB-ORIENTED PLATFORM, CLIENT-SERVER ARCHITECTURE, LAYERED ARCHITECTURE, MULTITENANCY, REST API, WEBSOCKET, ROLE-BASED ACCESS CONTROL (RBAC), NESTJS, REACT, PRISMA ORM, POSTGRESQL, CITIZEN–MUNICIPALITY INTERACTION, E-DEMOCRACY.

		Ю.І. Корецький			ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Р.С. Савіцький				
Змн.	Арк.	№ докум.	Підпис	Дата		6

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	5
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	9
ВСТУП.....	11
РОЗДІЛ 1. АНАЛІЗ АНАЛОГІВ ТА ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ БАГАТОФУНКЦІОНАЛЬНОГО МІСЬКОГО ПОРТАЛУ ВЗАЄМОДІЇ ЖИТЕЛІВ З МУНІЦИПАЛІТЕТОМ	14
1.1. Постановка задачі.....	14
1.2. Аналіз існуючих рішень для платформ електронної демократії та громадянської участі	16
1.3. Планування та проєктування архітектури міського порталу взаємодії жителів з муніципалітетом.....	24
1.4. Обґрунтування використання технологій та засобів.....	31
Висновки до першого розділу.....	34
РОЗДІЛ 2. ПРОЄКТУВАННЯ СИСТЕМИ ВЗАЄМОДІЇ ЖИТЕЛІВ З МУНІЦИПАЛІТЕТОМ	38
2.1. Визначення варіантів використання та структури платформи.....	38
2.2. Розробка структури бази даних вебсервісу	49
2.3. Проєктування та реалізація алгоритмів роботи системи	63
2.4. Реалізація функціональної частини системи.....	68
Висновки до другого розділу	79
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ ВЗАЄМОДІЇ ЖИТЕЛІВ З МУНІЦИПАЛІТЕТОМ	81
3.1. Порядок встановлення та налаштування параметрів системи.....	81
3.2. Структура інтерфейсу вебдодатку та порядок роботи	85
3.3. Тестування роботи та використання вебсервісу	100
Висновки до третього розділу	104
ВИСНОВКИ	106
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	109

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				7
Змн.	Арк.	№ докум.	Підпис	Дата		

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				
Змн.	Арк.	№ докум.	Підпис	Дата		8

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних

ЗК – зовнішній ключ (у таблицях опису структури бази даних)

ПК – первинний ключ (у таблицях опису структури бази даних)

СУБД – система керування базами даних

ACID (Atomicity, Consistency, Isolation, Durability) – набір властивостей транзакцій: атомарність, узгодженість, ізольованість і довговічність

API (Application Programming Interface) – інтерфейс прикладного програмування

CORS (Cross-Origin Resource Sharing) – механізм спільного використання ресурсів між різними джерелами

CRUD (Create, Read, Update, Delete) – базові операції створення, читання, оновлення та видалення даних

cuid (Collision-resistant Unique Identifier) – стійкий до колізій унікальний ідентифікатор записів

DDD (Domain-Driven Design) – предметно-орієнтоване проектування

DI (Dependency Injection) – впровадження залежностей

DTO (Data Transfer Object) – об'єкт передачі даних

e2e (End-to-End) – наскрізне тестування

ER (Entity-Relationship) – діаграма «сутність–зв'язок»

HTML (HyperText Markup Language) – мова розмітки гіпертексту

HTTP (HyperText Transfer Protocol) – протокол передачі гіпертексту

HTTPS (HyperText Transfer Protocol Secure) – захищений протокол передачі гіпертексту

IoC (Inversion of Control) – інверсія керування

JSON (JavaScript Object Notation) – об'єктна нотація JavaScript

jsonb (JSON binary) – двійковий тип зберігання JSON у PostgreSQL

JWT (JSON Web Token) – веб-токен у форматі JSON для автентифікації

MIME (Multipurpose Internet Mail Extensions) – стандарт визначення типу вмісту файлу

OAuth (Open Authorization) – відкритий протокол авторизації

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				
Змн.	Арк.	№ докум.	Підпис	Дата		9

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

RBAC (Role-Based Access Control) – рольова модель керування доступом

REST (Representational State Transfer) – архітектурний стиль взаємодії з програмним інтерфейсом

S3 (Simple Storage Service) – протокол об'єктного сховища даних

SDK (Software Development Kit) – набір засобів розробки

SEO (Search Engine Optimization) – оптимізація для пошукових систем

SPA (Single Page Application) – односторінковий вебдодаток

SQL (Structured Query Language) – мова структурованих запитів

SSR (Server-Side Rendering) – рендеринг сторінок на боці сервера

TCP (Transmission Control Protocol) – протокол керування передачею даних

TLS (Transport Layer Security) – протокол захисту транспортного рівня

UI (User Interface) – користувацький інтерфейс

UML (Unified Modeling Language) – уніфікована мова моделювання

URL (Uniform Resource Locator) – уніфікований локатор ресурсу

WebSocket – протокол двонаправленого зв'язку між клієнтом і сервером у реальному часі

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				
Змн.	Арк.	№ докум.	Підпис	Дата		10

ВСТУП

Актуальність теми. У сучасному світі цифрові технології кардинально змінюють підходи до управління містом та взаємодії між громадянами і владою [38]. Звертаючись до муніципалітету у звичний спосіб, мешканці й досі стикаються з чергами, паперовою тяганиною та відсутністю реакції на свої звернення, що вже не відповідає запитам сучасного міського суспільства. Громадяни потребують зручних, прозорих і швидких інструментів для участі в житті міста, а органи місцевого самоврядування – ефективних засобів для опрацювання звернень, інформування населення та збору зворотного зв'язку.

Платформи електронної демократії та громадянської участі стають невід'ємним елементом сучасного міського управління [14]. Їх впровадження дозволяє підвищити прозорість роботи муніципальних служб [17], скоротити час реагування на проблеми інфраструктури, залучити жителів до прийняття рішень через опитування та голосування, а також сформувати культуру відповідального громадянства.

Жителі міста отримують можливість подавати заявки на вирішення конкретних проблем із прив'язкою до місця, відстежувати їх статус у режимі реального часу, отримувати термінові сповіщення від муніципалітету та брати участь у голосуваннях з соціально важливих питань. Муніципальні служби, своєю чергою, можуть ефективно опрацьовувати звернення, публікувати офіційні новини та аналізувати результати взаємодії з громадянами.

Метою роботи є розробка багатofункціонального міського порталу взаємодії жителів з муніципалітетом. Платформа має забезпечувати функціонал, спрямований на оптимізацію процесів подання та опрацювання заявок, інформування громадян і проведення опитувань, що надасть жителям і муніципальним службам можливість ефективно взаємодіяти між собою.

Поставлена мета обумовлює необхідність виконання наступних **завдань**:

– провести аналіз функціональних потреб системи та існуючих аналогів у сфері електронної демократії та громадянської участі;

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				11
Змн.	Арк.	№ докум.	Підпис	Дата		

- обрати оптимальну архітектуру функціонування програмного комплексу;
- визначити технології та інструменти для реалізації серверної та клієнтської частин;
- здійснити проєктування структури бази даних й основних алгоритмів платформи;
- розробити програмне забезпечення та перевірити його на відповідність вимогам.

Об'єктом дослідження є процес організації взаємодії жителів міста з муніципалітетом, що охоплює етапи подання заявок на вирішення міських проблем, отримання офіційних сповіщень та участі в опитуваннях і голосуваннях.

Предметом дослідження є методи та засоби проєктування і розробки веборієнтованої платформи для взаємодії жителів з муніципалітетом, з модулем заявок та відстеження їх статусу, системою публікації сповіщень і новин та модулем опитувань і голосувань.

Практичне значення одержаних результатів. Розроблений багатофункціональний міський портал Urban Civic може бути використаний органами місцевого самоврядування для цифровізації комунікації з жителями, підвищення прозорості роботи муніципальних служб та забезпечення зручного доступу громадян до міських сервісів. Практична цінність платформи полягає в тому, що вона поєднує в єдиному рішенні той функціонал, який наявні аналоги пропонують лише частково: подання звернень із геоприв'язкою, медіадоказами та відстеженням статусу, двосторонню комунікацію між жителем і муніципалітетом у реальному часі, проведення опитувань і голосувань, а також рольове розмежування доступу. Завдяки багатоорендній архітектурі одна інсталяція платформи здатна одночасно обслуговувати декілька міст із повною ізоляцією їхніх даних, що робить рішення економічно вигідним та придатним для масштабного впровадження в різних громадах без розгортання окремої системи для кожної з них.

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				
Змн.	Арк.	№ докум.	Підпис	Дата		12

Апробація результатів роботи. Основні положення та результати кваліфікаційної роботи пройшли апробацію на двох наукових конференціях та висвітлені у тезах доповідей (у співавторстві) [2, 3].

Перша робота присвячена обґрунтуванню моделі ізоляції даних, покладеної в основу багатоорендної архітектури платформи, а друга – механізму безпечної верифікації офіційного домену муніципалітету, що застосовується під час створення міста в системі.

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				
Змн.	Арк.	№ докум.	Підпис	Дата		13

РОЗДІЛ 1. АНАЛІЗ АНАЛОГІВ ТА ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ БАГАТОФУНКЦІОНАЛЬНОГО МІСЬКОГО ПОРТАЛУ ВЗАЄМОДІЇ ЖИТЕЛІВ З МУНІЦИПАЛІТЕТОМ

1.1. Постановка задачі

Міські портали взаємодії жителів з муніципалітетом представляють собою цифрові екосистеми, що надають громадянам та органам місцевого самоврядування розширені інструменти для оперативної комунікації, подання звернень та інформування населення. Вони забезпечують прозорість роботи муніципальних служб, скорочують час реагування на проблеми міської інфраструктури та формують культуру відповідального громадянства. Це середовище та інструмент для побудови діалогу між містом і його жителями.

Основною метою створення платформи є забезпечення зручного, прозорого та ефективного механізму взаємодії жителів міста з муніципалітетом через єдину цифрову точку доступу. На платформі громадяни можуть подавати заявки на вирішення конкретних проблем міської інфраструктури, відстежувати їх статус, отримувати термінові сповіщення від муніципалітету та брати участь у голосуваннях і опитуваннях. Муніципальні служби, своєю чергою, можуть опрацьовувати звернення, публікувати офіційні новини та аналізувати зворотний зв'язок від жителів.

Цільову аудиторію платформи становлять три основні групи користувачів. Першою й найчисленнішою є жителі міста віком приблизно 18–65 років, які подають заявки щодо стану інфраструктури, відстежують їх виконання, отримують сповіщення та беруть участь у голосуваннях. Другою є працівники муніципальних служб, які опрацьовують звернення, звітують про виконані роботи й публікують новини та оголошення. Третьою – адміністратори системи, що відповідають за керування містами, користувачами й налаштування прав доступу. Такий поділ аудиторії визначає рольову модель системи та перелік функцій, доступних кожній категорії користувачів.

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				
Змн.	Арк.	№ докум.	Підпис	Дата		14

Для успішного створення платформи необхідно виконати низку завдань. Передусім слід проаналізувати наявні аналоги платформ електронної демократії та громадянської участі, визначивши їхні переваги й недоліки, які варто врахувати при розробці власної системи; це дозволить уникнути притаманних їм обмежень і отримати якісний та функціонально повний продукт. На основі цього аналізу необхідно визначити функціональні та нефункціональні вимоги до платформи, зокрема можливості подання й опрацювання заявок, публікації сповіщень і новин, проведення голосувань та забезпечення рольової моделі доступу.

В основі розробки лежить обрана оптимальна архітектура функціонування системи, а також технології та фреймворки серверної і клієнтської частин. Рішення також базується на чітко визначених акторах, варіантах використання системи та готовій структурі бази даних для зберігання заявок, сповіщень, новин і результатів голосувань.

Ключовим завданням є розробка програмного застосунку, що має забезпечувати створення та відстеження заявок на вирішення проблем міської інфраструктури з підтримкою медіадоказів і статусною моделлю опрацювання, чат-комунікацію між заявником і муніципалітетом у межах конкретного звернення, публікацію термінових сповіщень, новин та офіційних оголошень муніципалітету, звітність про виконану роботу з боку муніципальних служб, а також проведення опитувань і голосувань з відкритою публікацією результатів. Окремими завданнями є створення адаптивного, зручного та інтуїтивно зрозумілого інтерфейсу й тестування роботи додатку на відповідність поставленим вимогам.

Розробку програмного коду платформи можна розділити на три ключові етапи: проєктування бази даних, створення користувацького інтерфейсу та розробка RESTful-сервера, що забезпечує взаємодію з функціями додатку через визначені URL-адреси та HTTP-методи (GET, POST, PUT, PATCH, DELETE та інші).

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				
Змн.	Арк.	№ докум.	Підпис	Дата		15

Результатом успішної реалізації поставленого завдання є багатофункціональний міський портал взаємодії жителів з муніципалітетом, що забезпечує прозоре та оперативне вирішення міських проблем, інформування громадян і залучення їх до участі в житті міста.

1.2. Аналіз існуючих рішень для платформ електронної демократії та громадянської участі

Для обґрунтування доцільності розробки власної системи було проведено порівняльний аналіз чотирьох найбільш відомих рішень у сфері цифрової взаємодії громадян з міською владою: «Київ Цифровий», «Дія», FixMyStreet та Decidim. Кожне з цих рішень охоплює лише окремий аспект взаємодії громадян із міськими інституціями, що підтверджує необхідність створення комплексної екосистеми.

«Київ Цифровий» – це платформа міських електронних сервісів, розроблена Департаментом інформаційно-комунікаційних технологій Київської міської державної адміністрації [1]. Платформа доступна як мобільний застосунок (iOS, Android) та веб-портал і надає жителям столиці єдину точку доступу до міських послуг (рис. 1.1).

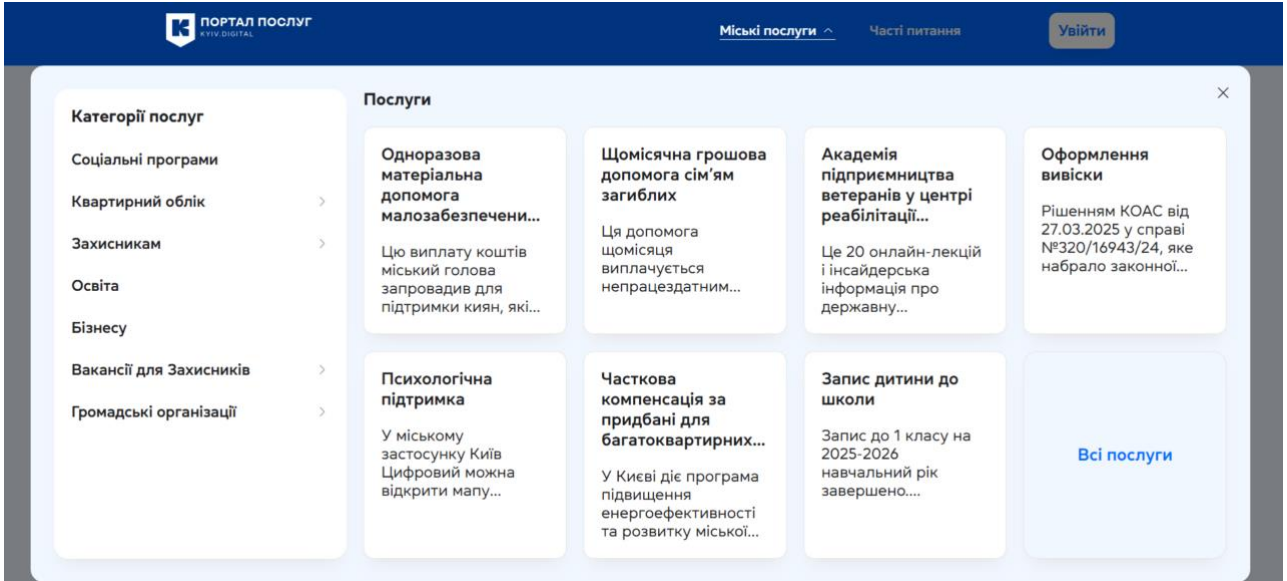


Рис. 1.1. Навігаційне меню на головній сторінці веб-порталу «Київ Цифровий»

Основний функціонал платформи охоплює кілька напрямів. Передусім це подання електронних звернень – громадяни можуть створювати звернення щодо

проблем міської інфраструктури (ямковий ремонт, вивіз сміття, освітлення тощо), відстежувати статус їх обробки в реальному часі та отримувати сповіщення про зміну статусу. Крім того, завдяки інтеграції з комунальними підприємствами платформа дозволяє здійснювати оплату комунальних послуг – за газ, воду, електроенергію та інші – безпосередньо через додаток. Передбачено також підтримку електронного квитка для громадського транспорту Києва з можливістю поповнення та перевірки балансу. Доповнює функціонал інформаційний портал, на якому публікуються новини, оголошення та інформація про планові відключення комунальних послуг.

Проте «Київ Цифровий» має суттєві обмеження з точки зору побудови комплексної екосистеми громадянської активності. Платформа є закритою муніципальною системою, локалізованою виключно для Києва, без можливості адаптації для інших міст. Вона не підтримує механізмів партисипативної демократії – голосувань та ініціатив, які б дозволяли громадянам безпосередньо впливати на рішення муніципалітету. Відсутні інструменти для організації волонтерських проєктів, тренінгів чи інших форм розвитку громадян. Комунікація є переважно односторонньою – від громадянина до муніципалітету через систему звернень, без повноцінного зворотного зв'язку у форматі чату.

«Дія» (аббревіатура від «Держава і я») – це державна платформа електронних послуг та цифрових документів України, розроблена Міністерством цифрової трансформації та запущена у 2020 році [4]. Платформа є флагманським проєктом цифровізації країни й реалізує концепцію «державна у смартфоні», мета якої – зробити взаємодію громадянина з державою максимально простою, швидкою та позбавленою бюрократичних процедур. «Дія» доступна одночасно як мобільний застосунок (iOS, Android) і як вебпортал (рис. 1.2), забезпечуючи громадянам доступ до понад 150 електронних послуг та низки цифрових документів. За кількістю активних користувачів, що обчислюється мільйонами, вона стала однією з наймасштабніших державних цифрових ініціатив у Європі та неодноразово відзначалася на міжнародному рівні.

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				17
Змн.	Арк.	№ докум.	Підпис	Дата		

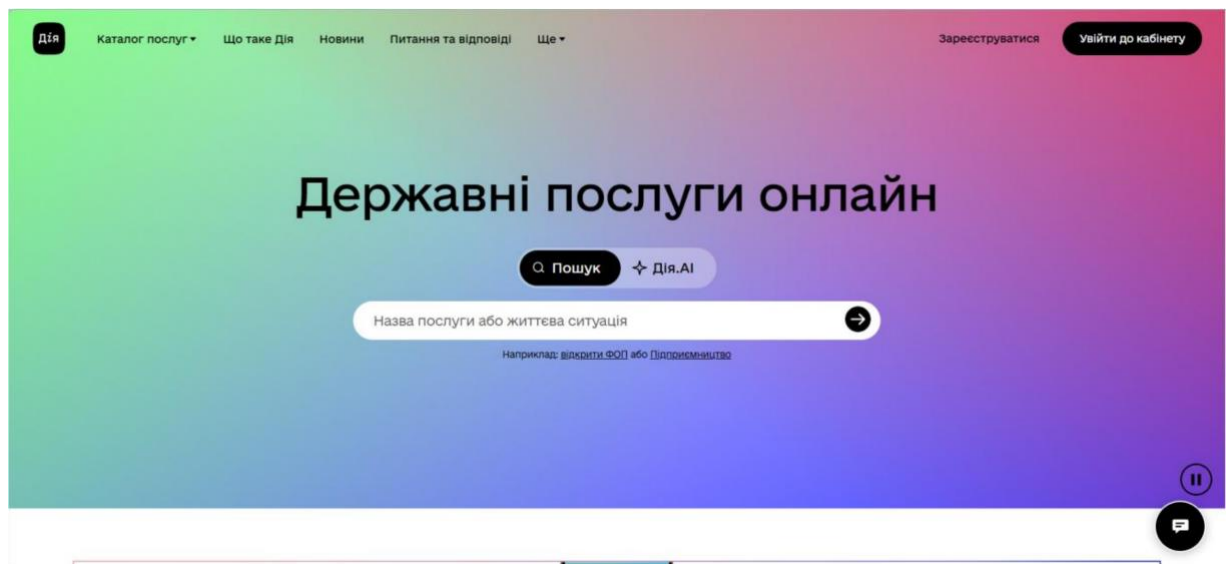


Рис. 1.2. Головна сторінка порталу «Дія»

Ключовий функціонал платформи охоплює кілька основних напрямів. Насамперед це цифрові документи – електронний паспорт громадянина, закордонний паспорт, водійське посвідчення, студентський квиток, свідоцтво про народження дитини, реєстраційні документи на транспортний засіб та інші документи, що мають юридичну силу нарівні з паперовими аналогами та можуть пред'являтися безпосередньо зі смартфона. Окрім того, платформа надає десятки електронних державних послуг (рис. 1.3), серед яких реєстрація бізнесу (ФОП), подання податкової декларації, оформлення допомоги при народженні дитини (єМалятко), реєстрація місця проживання, отримання різноманітних довідок та виплат. Важливою складовою є цифрова ідентифікація (Дія.Підпис) – кваліфікований електронний підпис, що дозволяє юридично значуще підписувати документи та ідентифікуватися на інших державних і приватних платформах без використання фізичних носіїв. Доповнює функціонал освітній напрям Дія.Освіта – платформа з безкоштовними онлайн-курсами з цифрової грамотності, що сприяє підвищенню рівня цифрових навичок населення. Розвитком екосистеми також стали портали Дія.Бізнес для підприємців та інструменти отримання державних послуг у єдиному вікні.

		Ю.І. Корецький			ІІЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				18
Змн.	Арк.	№ докум.	Підпис	Дата		

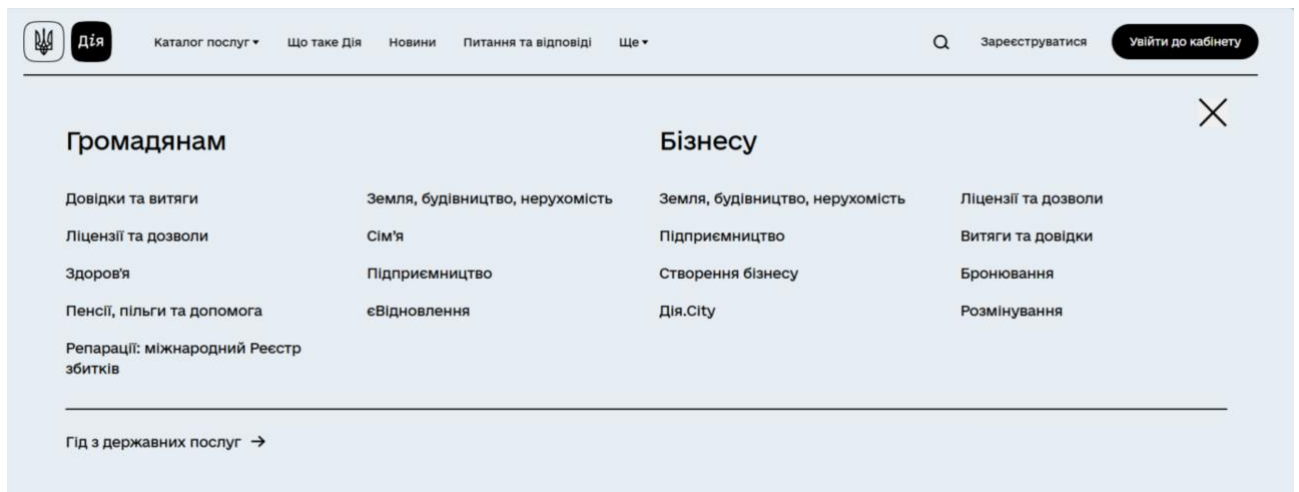


Рис. 1.3. Каталог е-послуг на порталі «Дія»

Попри безсумнівну цінність для цифрової трансформації країни, «Дія» зосереджена на державних послугах загальнонаціонального рівня та не адресує потреби локальної міської взаємодії. Платформа не підтримує механізмів подання заявок на конкретні інфраструктурні проблеми міста з геоприв'язкою, не має модулів голосувань та опитувань з питань міського життя, не забезпечує функціоналу тематичних спільнот чи організації волонтерських проєктів. Фактично «Дія» вирішує проблему взаємодії «громадянин–держава», тоді як розроблювана система фокусується на рівні «громадянин–місто» та «громадянин–громадянин».

FixMyStreet – це відкрита платформа (Open Source), розроблена британською некомерційною організацією mySociety у 2007 році та спеціалізована для подання заявок на проблеми міської інфраструктури [15]. Це один із найвідоміших і найдовговічніших проєктів у сфері civic tech, що став взірцем для багатьох подібних рішень у всьому світі. Завдяки відкритому вихідному коду платформу успішно впроваджено у понад 40 країнах, включаючи Великобританію, Норвегію, Швецію, Малайзію та інші, де вона функціонує під локальними назвами та адаптована до особливостей місцевого самоврядування. Основна ідея проєкту полягає в максимальному спрощенні процесу повідомлення про проблему для пересічного громадянина. Передбачається, що користувачеві не доведеться самотійно з'ясовувати, яка саме служба відповідає

за її вирішення – достатньо буде лише вказати місце на карті та коротко описати суть справи (рис. 1.4).

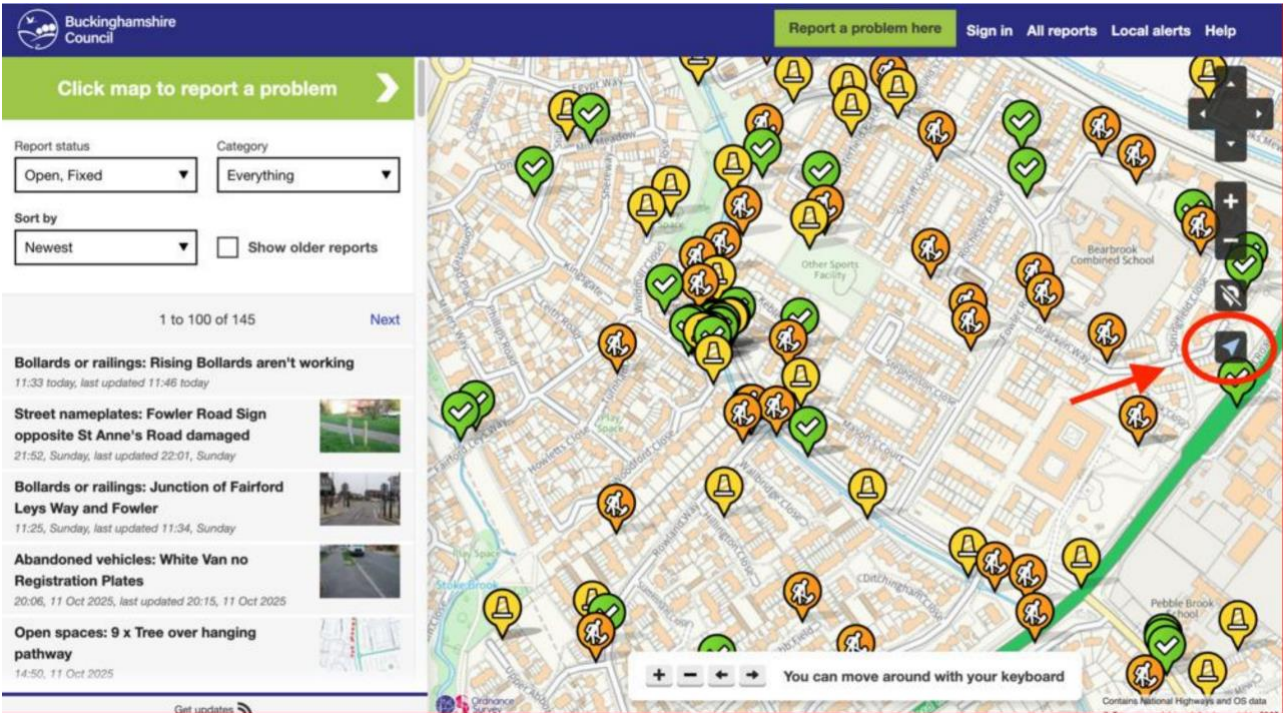


Рис. 1.4. Головна сторінка FixMyStreet з інтерактивною картою

Основний функціонал платформи зосереджено навколо подання заявок з геоприв'язкою: користувач вказує місце проблеми на інтерактивній карті (на основі OpenStreetMap), обирає категорію (вибоїни, освітлення, сміття, графіті, тротуари тощо), додає текстовий опис та фотографії, після чого система автоматично визначає відповідальний муніципальний орган на основі геолокації та надсилає заявку напрямую електронною поштою. Подані заявки мають статусну модель опрацювання, а користувач отримує сповіщення про зміну статусу (рис. 1.5). Усі заявки відображаються в публічному реєстрі на карті, що забезпечує прозорість роботи муніципальних служб та дозволяє іншим громадянам бачити вже зафіксовані проблеми у їхньому районі, уникаючи дублювання звернень. Крім того, платформа надає відкритий RESTful API для інтеграції з іншими системами та візуалізації даних, а її модульна архітектура дозволяє містам розгортати власні екземпляри й налаштовувати перелік категорій та відповідальних органів.

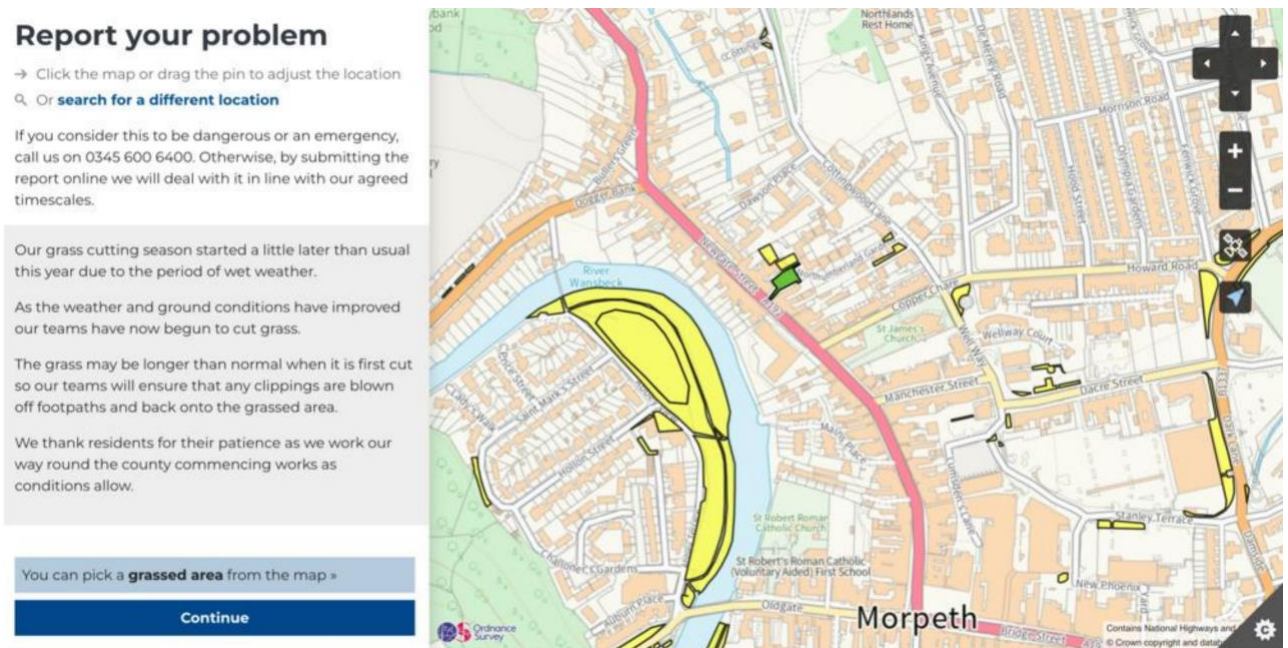


Рис. 1.5. Сторінка деталей звернення на FixMyStreet

Перевагами FixMyStreet є простота використання, мінімальний поріг входу для громадян, відкритий вихідний код (ліцензія AGPL), можливість самостійного розгортання та підтверджена роками ефективність у різних країнах. Водночас платформа має вузьку спеціалізацію – вона реалізує виключно модуль звернень щодо інфраструктурних проблем. FixMyStreet не підтримує голосувань та опитувань, не має інструментів для організації проєктів, волонтерства чи тренінгів, не передбачає створення спільнот та будь-якої соціальної взаємодії між користувачами. Комунікація є лінійною – від громадянина до муніципалітету – без механізмів обговорення, зворотного зв'язку у форматі діалогу чи колективного прийняття рішень. Таким чином, FixMyStreet є ефективним, але вузькоспеціалізованим інструментом, що розв'язує лише одну з багатьох задач комплексної взаємодії жителів із містом.

Decidim – це відкрита платформа для партисипативної демократії, розроблена муніципалітетом Барселони у 2016 році [13]. Назва походить від каталонського дієслова «decidir» (вирішувати) та відображає основну місію платформи – надати громадянам реальні інструменти для участі у прийнятті міських рішень. Decidim використовується більш ніж 400 організаціями у 30+ країнах, включаючи муніципалітети Гельсінкі, Мехіко, Нью-Йорка та Європейську Комісію (рис. 1.6).

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				21
Змн.	Арк.	№ докум.	Підпис	Дата		

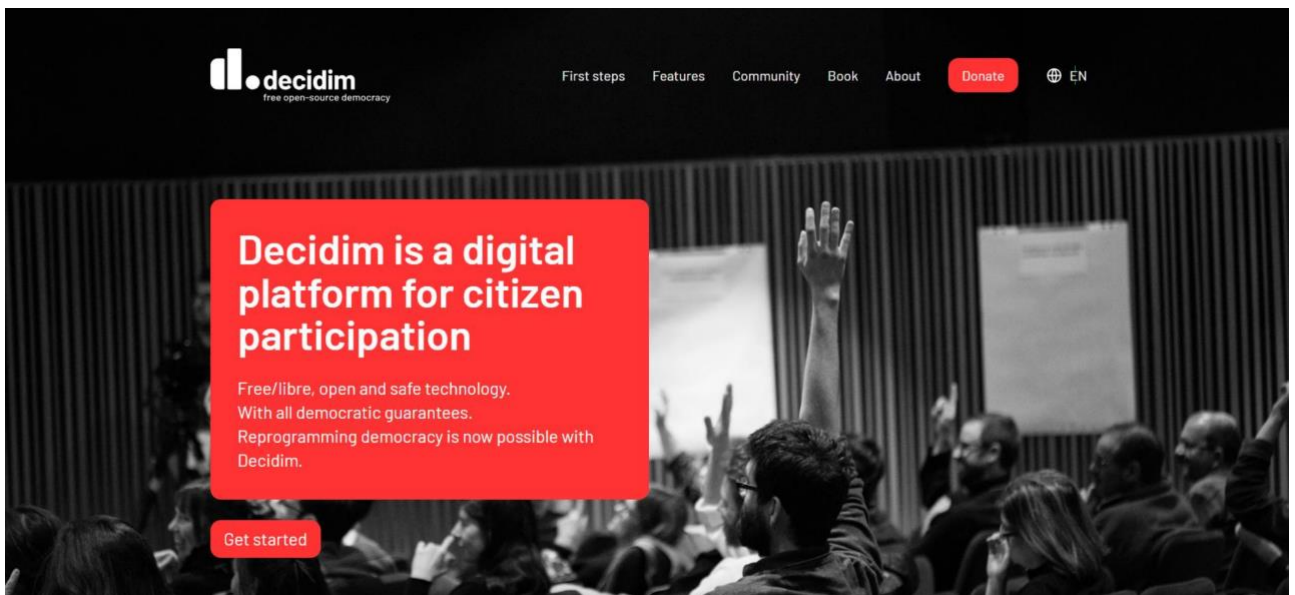


Рис. 1.6. Головна сторінка платформи Decidim

Ключовий функціонал платформи побудовано навколо партисипативних процесів – модуля для організації комплексних процесів прийняття рішень, що охоплюють етапи подання пропозицій, обговорення, голосування та звітування про результати. У межах цих процесів громадяни можуть подавати власні пропозиції та ініціативи щодо покращення міського середовища, а інші користувачі – підтримувати й коментувати їх. Окремий модуль партисипативного бюджету призначений для розподілу частини міського бюджету на основі голосування громадян. Для організації офлайн- та онлайн-заходів передбачено інструменти зборів і конференцій з можливістю реєстрації учасників та публікації протоколів, а для інформування – модулі блогів та інформаційних сторінок, на яких публікуються новини й матеріали, пов'язані з партисипативними процесами.

Decidim є потужною платформою для партисипативної демократії, проте її функціонал обмежений саме процесами колективного прийняття рішень. Платформа не підтримує оперативних звернень щодо конкретних інфраструктурних проблем з геоприв'язкою та відстеженням статусу виконання. Відсутні модулі розвитку громадян – події, тренінги, волонтерські проєкти з менеджментом учасників та завдань. Платформа не забезпечує повноцінної функціональності спільнот із груповими чатами та стрічками постів,

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				
Змн.	Арк.	№ докум.	Підпис	Дата		
						22

обмежуючись лише коментарями до пропозицій. Також відсутній модуль краудфандингу для фінансування громадських ініціатив.

Узагальнені результати порівняння розглянутих платформ та розроблюваної системи за ключовими критеріями наведено в таблиці 1.1. Для оцінювання використано такі позначення: + – критерій підтримується повністю, ± – підтримується частково, – – не підтримується.

Таблиця 1.1

Порівняльний аналіз аналогів

Критерій	Київ Цифровий	Дія	FixMyStreet	Decidim	Urban Civic
Подання звернень на проблеми	+	–	+	–	+
Геоприв'язка звернень	±	–	+	–	+
Медіадокази (фото / відео)	+	–	+	–	+
Статусна модель та відстеження	+	–	+	–	+
Чат-комунікація (зворотний зв'язок)	–	–	–	±	+
Сповіщення та новини	+	±	±	+	+
Голосування та опитування	–	±	–	+	+
Рольова модель (RBAC)	±	+	базова	±	+

Адаптивність для різних міст (багатоорендн ість)	—	+	+	+	+
--	---	---	---	---	---

Як видно з таблиці 1.1, кожен із розглянутих аналогів покриває лише частину необхідної функціональності та має суттєві обмеження. «Київ Цифровий» і FixMyStreet забезпечують повноцінні механізми звернень із геоприв'язкою, медіадоказами та відстеженням статусу, проте не підтримують голосувань, опитувань і зворотного зв'язку у форматі чату, а «Київ Цифровий» до того ж не адаптується для інших міст. «Дія» зосереджена на загальнодержавних електронних послугах і не адресує рівень локальної міської взаємодії. Decidim має потужні інструменти голосувань та партисипативної демократії, але позбавлений модуля оперативних звернень із геоприв'язкою та відстеженням виконання. Жоден із проаналізованих рішень не поєднує всіх необхідних можливостей в єдиній системі. Натомість наш застосунок Urban Civic об'єднує в одному рішенні подання звернень із геоприв'язкою, медіадоказами та статусною моделлю, чат-комунікацію зі зворотним зв'язком, публікацію сповіщень і новин, голосування й опитування, повноцінну рольову модель (RBAC), а також адаптивність для різних міст завдяки багатоорендній архітектурі, що й усуває виявлені в аналогах недоліки.

1.3. Планування та проєктування архітектури міського порталу взаємодії жителів з муніципалітетом

Для реалізації платформи Urban Civic використовується клієнт-серверна архітектура. Модель такої системи полягає в тому, що клієнт, використовуючи браузер, відправляє запит на віддалений сервер. На стороні сервера отриманий запит обробляється, формується відповідь, після чого готовий результат повертається клієнтові.

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				
Змн.	Арк.	№ докум.	Підпис	Дата		24

Перевагами архітектурного стилю є централізоване зберігання даних, оскільки вся інформація розміщується на сервері, а не на пристроях користувачів, що спрощує її резервне копіювання та підтримання в актуальному стані. Окрім того, це забезпечує підвищений рівень безпеки даних, адже доступ до них контролюється на сервері, який має кращий захист, ніж клієнтські пристрої. Також важливою складовою є простота обслуговування та оновлення, оскільки логіку системи достатньо змінити в одному місці – на сервері, без потреби оновлювати кожного клієнта окремо. Основним недоліком такої архітектури є те, що несправність сервера робить систему недоступною для користувачів за відсутності резервних серверів. Крім того, якщо окремий модуль зазнає значно більшого навантаження, ніж інші, за недостатньої потужності сервера можуть виникати затримки в роботі всієї системи [11].

Архітектурний підхід розділяє функціональні обов'язки між клієнтською та серверною частинами. На сервері реалізуються зберігання і захист даних (заявок, сповіщень, новин, результатів голосувань), обробка клієнтських запитів та виконання бізнес-логіки (опрацювання заявок, розмежування прав доступу, формування сповіщень), а також повернення результату клієнту у форматі JSON або у вигляді інших ресурсів – зображень чи файлів. На стороні клієнта здійснюється надання користувацького інтерфейсу, формування запиту до сервера та його надсилання, а також отримання, обробка й подальше відображення результатів запиту.

Серверну частину платформи реалізовано у вигляді модульного моноліту з використанням фреймворку NestJS. Усі функціональні можливості системи розгортаються та виконуються як єдиний застосунок, проте його внутрішня структура поділена на окремі логічно відокремлені модулі. Такий підхід забезпечує цілісність серверного застосунку, спрощує його розгортання та супровід, водночас зберігаючи зрозумілу модульну організацію коду.

Модульний моноліт доцільно застосовувати в проєктах із помірним навантаженням, оскільки він не потребує складної інфраструктури, характерної для мікросервісної архітектури. Основними обмеженнями монолітного підходу

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				25
Змн.	Арк.	№ докум.	Підпис	Дата		

є складність незалежного масштабування окремих частин системи та потенційна єдина точка відмови [21]. Мікросервісна архітектура, навпаки, забезпечує незалежне розгортання сервісів, кращі можливості масштабування та технологічну гнучкість, однак суттєво ускладнює проектування, підтримку та експлуатацію системи [22]. З огляду на обсяг функціональності та очікуване навантаження платформи Urban Civic, модульний моноліт є оптимальним компромісом між простотою реалізації, структурованістю коду та можливістю подальшого розвитку системи.

Узагальнене порівняння монолітної та мікросервісної архітектур наведено на рис. 1.7.

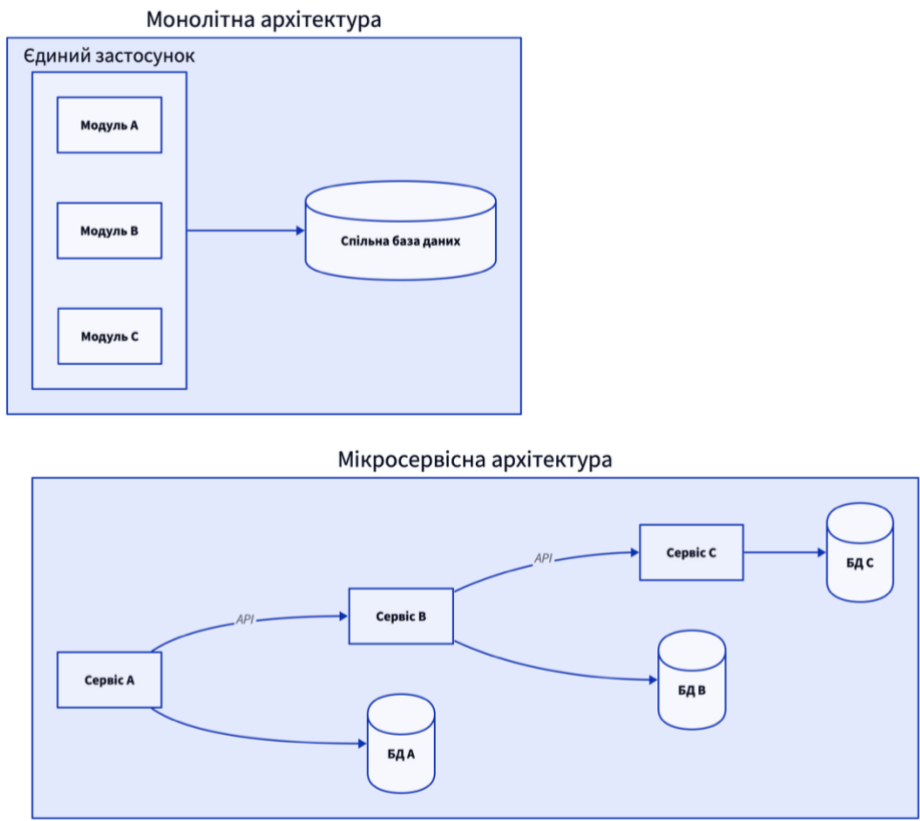


Рис. 1.7. Монолітна та мікросервісна архітектури

Усередині кожного функціонального модуля використано багат шарову архітектуру, що забезпечує розподіл відповідальностей між окремими компонентами системи. Архітектура складається з доменного, прикладного, API, інфраструктурного рівнів, а також рівня доступ до даних (рис. 1.8). Кожен рівень

виконує власні функції та взаємодіє з іншими через визначені інтерфейси, що підвищує підтримуваність і розширюваність програмного забезпечення [12].

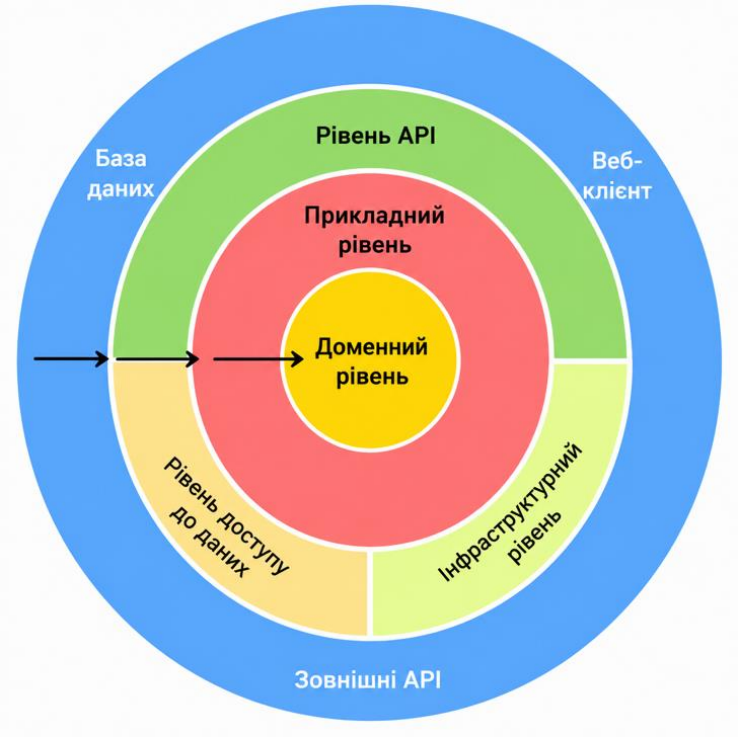


Рис. 1.8. Рівні архітектури

В основі системи знаходиться доменний рівень, який містить основні сутності предметної області, перерахування, типи даних та правила, що визначають структуру й поведінку об'єктів. Цей шар використовується іншими компонентами застосунку як фундамент для реалізації бізнес-функціональності.

За реалізацію прикладної логіки відповідає прикладний рівень. На ньому виконуються сценарії використання системи, перевіряються бізнес-правила, координується взаємодія між окремими компонентами та здійснюється підготовка даних для подальшої передачі між рівнями.

Взаємодія з клієнтськими застосунками забезпечується компонентом API. До його складу входять контролери та WebSocket-шлюзи, які приймають запити від користувачів, виконують перевірку вхідних даних, звертаються до сервісів прикладного рівня та повертають результати виконання операцій.

Доступ до даних реалізовано на рівні доступу до даних, який забезпечує взаємодію з базою даних PostgreSQL за допомогою ORM Prisma. Тут

виконуються операції створення, читання, оновлення та видалення записів, а також формуються запити до сховища даних.

Інтеграцію із зовнішніми сервісами та технічними компонентами системи забезпечує інфраструктурний рівень. До нього належать механізми роботи з файловим сховищем Cloudflare R2, сервісом надсилання електронних листів Resend, сервісом автентифікації Google OAuth, чергами фонових завдань BullMQ, кешуванням на базі Redis та іншими зовнішніми ресурсами.

Зовнішній шар архітектури утворюють клієнтський вебзастосунок, база даних PostgreSQL та сторонні сервіси, з якими система взаємодіє під час роботи. Саме вони забезпечують зберігання даних, обмін інформацією та надання додаткових функціональних можливостей.

Застосування багат шарової архітектури дозволяє відокремити бізнес-логіку від механізмів представлення даних, доступу до сховища та інтеграції із зовнішніми сервісами. Такий підхід підвищує зрозумілість структури проєкту, спрощує тестування окремих компонентів і полегшує подальший розвиток та масштабування системи [16].

Окремим архітектурним рішенням, що безпосередньо усуває недолік адаптивності, виявлений під час аналізу аналогів, є побудова системи навколо сутності «місто» як орендаря. Усі дані – заявки, сповіщення, новини, опитування, департаменти – належать конкретному місту та ізольовані в його межах, завдяки чому одна інсталяція платформи здатна обслуговувати багато міст одночасно [8].

Клієнтську частину, яка відповідає за складний інтерфейс з багатьма інтерактивними елементами (інтерактивна карта заявок, чат із муніципалітетом, форми голосування), реалізовано як сучасний реактивний вебдодаток на основі компонентного підходу. На відміну від класичного односторінкового додатку, який завантажується один раз і динамічно змінює вміст без перезавантаження сторінки, обраний фреймворк підтримує також рендеринг сторінок на боці сервера. Це поєднує переваги SPA – плавну взаємодію та оновлення лише необхідних частин сторінки – з усуненням типових недоліків SPA, а саме труднощів з індексацією контенту (SEO) та тривалого першого

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				28
Змн.	Арк.	№ докум.	Підпис	Дата		

завантаження [23]. Обмін даними між клієнтом і сервером відбувається асинхронно у форматі JSON, що зменшує навантаження на сервер, оскільки немає потреби щоразу генерувати повну HTML-сторінку.

Для забезпечення оперативності, визначеної як одна з ключових вимог до системи, синхронний обмін за моделлю REST доповнено каналами реального часу. Для чату між заявником і муніципалітетом у межах конкретного звернення застосовано двонаправлений протокол WebSocket, що дозволяє доставляти повідомлення миттєво, без повторного опитування сервера. Для доставки користувачам системних сповіщень (про публікацію новини, термінове сповіщення чи зміну статусу заявки) використано односпрямований канал та механізм внутрішньосистемних сповіщень.

Щоб гарантувати, що зміна стану системи неодмінно дійде до адресата, застосовано патерн «вихідних повідомлень» (Transactional Outbox): подія в межах однієї транзакції з основними даними записується до спеціальної таблиці, після чого фоновий обробник за допомогою черги завдань надійно розсилає її каналами доставки [29]. Це унеможливорює втрату сповіщень навіть за тимчасових збоїв.

У системі реалізовано рольову модель доступу, за якою права визначаються набором дозволів, згрупованих у ролі (житель, працівник муніципальної служби, адміністратор міста, суперадміністратор системи). Це забезпечує гнучке та безпечне керування доступом до функцій платформи [30].

Після вибору архітектури було розроблено діаграму компонентів, яка дозволяє перейти від загального розуміння проєкту до його реалізації у вигляді коду та показує взаємозв'язки між складовими системи [25]. Загальну структуру платформи наведено на рис. 1.9. Компонент клієнтського вебдодатку взаємодіє з компонентом серверного API через інтерфейс на основі методів HTTP-запитів та каналів реального часу. Серверні модулі звертаються до рівня доступу до даних, який взаємодіє з базою даних PostgreSQL, та до рівня інтеграцій, відповідального за зв'язок з об'єктним сховищем медіафайлів, чергою завдань і сервісом електронної пошти.

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				
Змн.	Арк.	№ докум.	Підпис	Дата		29



Рис. 1.9. Діаграма компонентів

У підсумку проектування архітектури отримано схему обробки запиту (рис. 1.10), яка візуально відображає послідовність кроків від дії користувача в інтерфейсі до бази даних або зовнішнього сервісу та у зворотному напрямку. Взаємодія розпочинається з дії користувача в клієнтському додатку, який формує та надсилає HTTP-запит до серверного API. На стороні сервера запит спершу потрапляє до контролера, де проходить валідацію вхідних даних та перевірку прав доступу відповідними охоронцями, після чого керування передається сервісу, що містить бізнес-логіку. Сервіс звертається до рівня доступу до даних, який через ORM виконує відповідний запит до бази даних PostgreSQL, а за потреби – і до зовнішніх сервісів (об'єктного сховища, черги завдань чи пошти). Отриманий результат у зворотному порядку повертається сервісу й контролеру, який формує відповідь у форматі JSON, після чого клієнтський додаток оновлює інтерфейс для користувача.

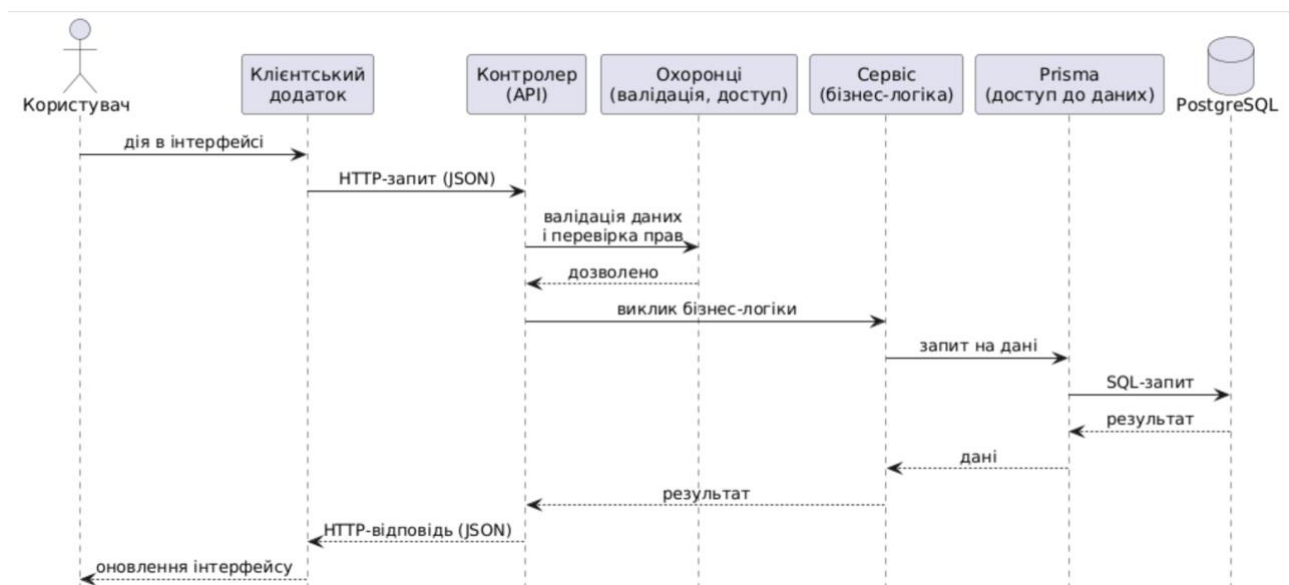


Рис. 1.10. Діаграма послідовності обробки запиту

1.4. Обґрунтування використання технологій та засобів

Ключовим архітектурним рішенням під час вибору технологічного стеку стало використання єдиної мови програмування TypeScript як для серверної, так і для клієнтської частини. TypeScript є надбудовою над JavaScript, що додає сувору статичну типізацію. Це сприяє написанню надійнішого коду, дозволяє виявляти значну частину помилок ще на етапі компіляції та прискорює розробку завдяки кращим підказкам середовища [5]. Спільна мова для обох частин системи дає змогу повторно використовувати описи типів даних і знижує складність розробки.

Для серверної частини обрано Node.js – кросплатформне середовище виконання JavaScript-коду. Завдяки неблокуючій моделі введення-виведення Node.js добре пристосований до застосунків з інтенсивним мережевим обміном та великою кількістю одночасних з'єднань, що є важливим для роботи каналів реального часу, як-от чат заявника з муніципалітетом.

Серверний застосунок реалізовано з використанням фреймворку NestJS. Вибір цього фреймворку зумовлений насамперед його модульною структурою та підтримкою TypeScript, що добре узгоджується з обраною архітектурою серверної частини. NestJS надає зручні механізми для організації коду у вигляді модулів, контролерів і сервісів, а також містить вбудовані засоби для валідації

даних, реалізації WebSocket-взаємодії та документування API. Використання NestJS дало змогу побудувати серверну частину як набір незалежних функціональних модулів, що спрощує подальший супровід і розширення системи.

Для взаємодії з базою даних використано ORM Prisma. У проєкті вона застосовується для опису структури даних, виконання міграцій та формування запитів до PostgreSQL. Prisma забезпечує зручний програмний інтерфейс для роботи з даними та дозволяє працювати з об'єктами мовою TypeScript замість написання великої кількості SQL-запитів вручну. Крім того, механізм міграцій спрощує керування змінами структури бази даних під час розробки та підтримує узгодженість між моделями застосунку і схемою сховища даних.

У якості системи керування базами даних обрано PostgreSQL – об'єктно-реляційну СУБД з відкритим вихідним кодом, що має високу продуктивність і підтримує широкий спектр функцій: індекси, віртуальні таблиці (view), обмеження цілісності (constraints), тип даних JSON та зберігання географічних координат. Дані платформи мають яскраво виражену реляційну природу з численними зв'язками (місто – заявки – звіти – вкладення; опитування – варіанти – голоси), для яких критично важливою є узгодженість. PostgreSQL гарантує цілісність даних завдяки ACID-транзакціям, що робить її надійним вибором для зберігання важливої інформації про звернення громадян та результати голосувань.

Автентифікацію користувачів реалізовано на основі токенів JWT із застосуванням бібліотеки Passport, що підтримує декілька стратегій входу: локальну (за електронною поштою та паролем) і через обліковий запис Google за протоколом OAuth 2.0. Паролі ніколи не зберігаються у відкритому вигляді – лише у вигляді хешів, обчислених стійким алгоритмом bcrypt. Передавання даних здійснюється захищеним каналом за протоколом HTTPS/TLS.

Для асинхронного оброблення фонових завдань, передусім надійної розсилки сповіщень за патерном Transactional Outbox, використано чергу завдань BullMQ, що працює на сховищі Redis. Це забезпечує відкладене та повторюване

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				32
Змн.	Арк.	№ докум.	Підпис	Дата		

виконання завдань доставки з гарантією їх опрацювання навіть у разі тимчасових збоїв.

Фото- та відеодокази заявок, а також вкладення до звітів і новин зберігаються в окремому об'єктному сховищі, сумісному з протоколом S3 [33], доступ до якого здійснюється через офіційний AWS SDK. У базі даних зберігаються лише метадані та посилання на файли. Таке рішення розвантажує основну базу даних і забезпечує ефективну роботу з великими бінарними файлами.

Для автоматичної генерації інтерактивної документації серверного програмного інтерфейсу було використано інструмент Swagger (специфікація OpenAPI), інтегрований у систему за допомогою вбудованих засобів фреймворку NestJS [10]. Таке рішення дозволило автоматично синхронізувати зміни в коді з описом кінцевих точок, мінімізуючи ризик появи застарілої або некоректної документації. Наявність інтерактивного веб-інтерфейсу суттєво спрощує процеси тестування, налагодження та подальшого супроводу API, оскільки розробники та тестувальники мають можливість виконувати HTTP-запити безпосередньо з браузера без використання стороннього програмного забезпечення.

Для реалізації клієнтської частини обрано бібліотеку React у поєднанні з фреймворком Next.js. React дозволяє повторно використовувати раніше написані компоненти та спрощує створення динамічних інтерфейсів з багатьма інтерактивними елементами, як-от карта заявок чи форми голосування. Next.js додає до React маршрутизацію на основі файлової структури, рендеринг на боці сервера та засоби оптимізації продуктивності, що позитивно впливає на швидкість завантаження сторінок.

Для роботи із серверним станом (отримання, кешування та синхронізація даних з API) використано бібліотеку TanStack Query, яка бере на себе кешування, фонове оновлення даних та оброблення станів завантаження й помилок. Локальним станом інтерфейсу керує легковагова бібліотека Zustand.

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				
Змн.	Арк.	№ докум.	Підпис	Дата		33

Для стилізації інтерфейсу застосовано утилітарний CSS-фреймворк Tailwind CSS у поєднанні з доступними компонентами Radix UI, що дозволяє швидко будувати адаптивний та узгоджений за стилем інтерфейс. Роботу з формами та їх валідацію на боці клієнта реалізовано засобами бібліотеки React Hook Form.

Для відображення заявок на інтерактивній карті та вибору місця проблеми використано бібліотеку Leaflet (через обгортку react-leaflet) на основі відкритих картографічних даних. Цей функціонал реалізує одну з ключових переваг, виявлених під час аналізу аналога FixMyStreet, – геоприв'язку звернень. Для двонаправленого зв'язку з сервером у чаті застосовано клієнт Socket.IO, а для підтримки багатомовності інтерфейсу – бібліотеку next-intl.

Для зменшення кількості програмних помилок, забезпечення єдиного стилю коду та спрощення супроводу проєкту в процесі розробки використовувалися інструменти статичного аналізу та автоматичного форматування коду. Зокрема, ESLint виконує перевірку вихідного коду на відповідність визначеним правилам, дозволяючи виявляти потенційні помилки, невикористані змінні, проблеми типізації та порушення прийнятих практик програмування ще до запуску застосунку. Для автоматичного приведення коду до єдиного стилю оформлення використовувався інструмент Prettier, що усуває розбіжності у форматуванні та покращує читабельність програмного коду.

Узагальнено технологічний стек платформи та обґрунтування вибору кожної складової наведено в таблиці 1.2.

Таблиця 1.2

Технологічний стек платформи Urban Civic

Складова / призначення	Технологія	Обґрунтування вибору
Мова програмування	TypeScript	Статична типізація, спільні типи для клієнта й сервера
Середовище виконання сервера	Node.js	Неблокуюче введення-виведення, обробка реального часу

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				
Змн.	Арк.	№ докум.	Підпис	Дата		34

Серверний фреймворк	NestJS	Модульність, впровадження залежностей, відповідність архітектури
ORM	Prisma	Типобезпечний клієнт, декларативна схема, міграції
База даних	PostgreSQL	Реляційна цілісність, ACID, JSON, геокоординати
Автентифікація	JWT, Passport, OAuth 2.0, bcrypt	Захищена автентифікація, вхід через Google, хешування паролів
Черга завдань	BullMQ + Redis	Надійна асинхронна доставка сповіщень
Сховище файлів	Об'єктне сховище (S3-сумісне), AWS SDK	Ефективне зберігання медіа поза БД
Документація API	Swagger (OpenAPI)	Автоматична інтерактивна документація
Клієнтський фреймворк	React + Next.js	Компонентний підхід, SSR, маршрутизація
Стан клієнта	TanStack Query, Zustand	Кешування даних API та керування локальним станом
Інтерфейс	Tailwind CSS, Radix UI	Адаптивний UI та робота з формами
Карти	Leaflet (react-leaflet)	Геоприв'язка звернень
Реальний час / мови	Socket.IO, next-intl	Чат у реальному часі та багатомовність
Якість коду / середовище	ESLint, Prettier, VS Code	Єдині стандарти коду та зручна розробка

Висновки до першого розділу

Було сформульовано задачу розробки багатофункціонального міського порталу взаємодії жителів з муніципалітетом Urban Civic, визначено його мету, цільову аудиторію та перелік завдань, які необхідно виконати для досягнення поставленої мети. Окреслено основні функціональні можливості системи: створення та відстеження заявок на вирішення проблем міської інфраструктури

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				
Змн.	Арк.	№ докум.	Підпис	Дата		35

з підтримкою медіадоказів і статусною моделлю опрацювання, чат-комунікацію між заявником і муніципалітетом, публікацію термінових сповіщень, новин та офіційних оголошень, звітність про виконану роботу, а також проведення опитувань і голосувань з відкритою публікацією результатів.

Проведено порівняльний аналіз чотирьох найвідоміших аналогів у сфері електронної демократії та громадянської участі – «Київ Цифровий», «Дія», FixMyStreet та Decidim. Аналіз показав, що кожне з розглянутих рішень охоплює лише один–два аспекти необхідної функціональності: «Київ Цифровий» і FixMyStreet забезпечують механізми звернень, але не підтримують голосувань; «Дія» зосереджена на загальнодержавних е-послугах і не адресує міський рівень; Decidim має потужні інструменти партисипативної демократії, проте позбавлений модуля оперативних звернень з геоприв'язкою. Жоден з аналогів не поєднує всіх необхідних можливостей, що обґрунтовує доцільність розробки власної комплексної системи та визначає недоліки, які слід усунути.

На основі визначених вимог обрано архітектуру системи. Платформу побудовано за клієнт-серверною моделлю; серверну частину спроектовано як модульний моноліт на фреймворку NestJS із багаторівневою (layered) організацією коду відповідно до принципів чистої архітектури. Ключовим архітектурним рішенням стала багатоорендна модель із сутністю «місто» як орендарем, що безпосередньо усуває виявлений в аналогах недолік адаптивності та дозволяє одній інсталяції обслуговувати багато міст. Для забезпечення оперативності взаємодії в реальному часі реалізовано через протокол WebSocket, а надійну доставку сповіщень – за патерном вихідних повідомлень (Transactional Outbox) із чергою завдань. Розмежування прав доступу реалізовано рольовою моделлю (RBAC).

Обґрунтовано вибір технологій та засобів реалізації. Засадничим рішенням стало використання єдиної мови TypeScript для серверної та клієнтської частин. Для серверної частини обрано середовище Node.js, фреймворк NestJS, ORM Prisma та СУБД PostgreSQL; автентифікацію реалізовано на основі JWT і бібліотеки Passport (зокрема через Google OAuth 2.0), асинхронну обробку – за

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				
Змн.	Арк.	№ докум.	Підпис	Дата		36

допомогою черги BullMQ на сховищі Redis, зберігання медіафайлів – в об'єктному сховищі, сумісному з протоколом S3. Клієнтську частину побудовано на React і Next.js із застосуванням TanStack Query, Zustand, Tailwind CSS, Radix UI, Leaflet, Socket.IO та next-intl.

Таким чином, було визначено задачу та вимоги до системи, обґрунтовано доцільність її розробки на основі аналізу аналогів, спроектовано архітектуру та обрано технології реалізації.

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				
Змн.	Арк.	№ докум.	Підпис	Дата		37

РОЗДІЛ 2. ПРОЄКТУВАННЯ СИСТЕМИ ВЗАЄМОДІЇ ЖИТЕЛІВ З МУНІЦИПАЛІТЕТОМ

2.1. Визначення варіантів використання та структури платформи

Аналіз функціональних та нефункціональних вимог до системи взаємодії жителів з муніципалітетом дав підставу для визначення основних варіантів використання платформи Urban Civic. Визначення акторів і варіантів використання є відправною точкою проєктування, оскільки саме на ньому ґрунтується подальша розробка структури бази даних, алгоритмів роботи та користувацького інтерфейсу платформи.

У системі розрізняють чотирьох ключових акторів: гість, житель, представник муніципалітету та адміністратор системи. Актор – це роль, яку відіграє користувач (або зовнішня система) під час взаємодії з платформою, причому одна людина може виступати в різних ролях залежно від наданих їй прав. Кожен з акторів має власний набір функціональних можливостей і список прав доступу, що визначається рольовою моделлю (RBAC) (табл. 2.1).

Таблиця 2.1

Виявлення акторів

Актор	Короткий опис
Гість	Неавторизований відвідувач платформи. Може переглядати перелік доступних міст, реєструвати обліковий запис або входити в систему за допомогою електронної пошти й пароля чи облікового запису Google.
Житель	Зареєстрований громадянин міста. Подає заявки на вирішення проблем міської інфраструктури з геоприв'язкою та медіадоказами, відстежує їх статус, спілкується з муніципалітетом у чаті заявки, переглядає новини й термінові сповіщення, бере участь у голосуваннях, а також може подати заявку на створення нового міста.
Представник муніципалітету	Працівник муніципальної служби. Опрацьовує заявки громадян (призначає відповідальний департамент, змінює статус), формує звіти про виконану роботу, спілкується із заявником, публікує термінові сповіщення й новини, створює та закриває опитування, керує департаментами міста.

Адміністратор системи	Користувач із найвищим рівнем доступу. Розглядає заявки на створення нових міст, керує містами та обліковими записами користувачів (зміна системної ролі, блокування), а також рольовою моделлю та правами доступу. Має доступ до всіх функцій платформи.
-----------------------	---

Основні функціональні можливості платформи можна згрупувати у вигляді варіантів використання для різних категорій користувачів. Наступна таблиця (табл. 2.2) представляє перелік визначених варіантів використання, їх основних акторів, найменування та формулювання сутності кожного варіанта. Ці варіанти охоплюють ключові аспекти платформи, зокрема реєстрацію та авторизацію, подання й відстеження заявок, комунікацію в чаті, публікацію сповіщень і новин, проведення голосувань, а також адміністрування міст, користувачів та прав доступу.

Таблиця 2.2

Виявлення варіантів використання

Основний актор	Найменування	Формулювання
Гість	Реєстрація	Цей варіант дозволяє створити обліковий запис за допомогою електронної пошти та пароля або через обліковий запис Google.
Гість / Житель	Авторизація	Цей варіант дозволяє ввести свої облікові дані та отримати доступ до системи.
Житель	Приєднання до міста	Цей варіант дозволяє приєднатися до обраного міста, щоб отримати доступ до його заявок, сповіщень, новин та опитувань.
Житель	Подання заявки	Цей варіант дозволяє створити заявку на вирішення проблеми міської інфраструктури із зазначенням опису, категорії, місця на карті (геоприв'язка) та доданням фото- чи відеодоказів.

Житель	Відстеження статусу заявок	Цей варіант дозволяє переглядати власні заявки та стежити за зміною їх статусу (нове → у роботі → виконано / відхилено).
Житель	Чат-комунікація з муніципалітетом	Цей варіант дозволяє в межах конкретної заявки обмінюватися повідомленнями з представником муніципалітету в реальному часі.
Житель	Перегляд новин і сповіщень	Цей варіант дозволяє переглядати новини та термінові сповіщення міста, а також підписуватися на певні типи сповіщень.
Житель	Участь у голосуванні	Цей варіант дозволяє проголосувати в опитуванні, обравши один із запропонованих варіантів, та переглянути відкриті результати.
Житель	Подання заявки на створення міста	Цей варіант дозволяє подати заявку на створення нового міста з підтвердженням права власності на офіційний домен муніципалітету.
Представник муніципалітету	Опрацювання заявок	Цей варіант дозволяє призначати заявці відповідальний департамент та змінювати її статус у процесі вирішення проблеми.
Представник муніципалітету	Формування звітів	Цей варіант дозволяє створювати звіти про хід чи результат виконання заявки (проміжний, про вирішення або про відхилення) з доданням медіадоказів.
Представник муніципалітету	Публікація сповіщень	Цей варіант дозволяє публікувати термінові сповіщення із зазначенням рівня критичності, типу та строку дії.
Представник муніципалітету	Публікація новин	Цей варіант дозволяє створювати та редагувати офіційні новини й оголошення міста.

Представник муніципалітету	Проведення опитувань	Цей варіант дозволяє створювати опитування з переліком варіантів, закривати їх та публікувати результати у відкритому доступі.
Представник муніципалітету	Керування департаментами	Цей варіант дозволяє створювати, редагувати та видаляти департаменти міста, на які призначаються заявки.
Адміністратор системи	Розгляд заявок на створення міста	Цей варіант дозволяє переглядати заявки на створення міст та схвалювати або відхиляти їх із зазначенням причини.
Адміністратор системи	Керування містами	Цей варіант дозволяє переглядати, редагувати та видаляти міста на платформі.
Адміністратор системи	Керування користувачами	Цей варіант дозволяє переглядати список користувачів, змінювати їх системну роль, блокувати та розблоковувати облікові записи.
Адміністратор системи	Керування ролями та правами доступу	Цей варіант дозволяє керувати ролями учасників міста та набором прав доступу для кожної ролі.

На основі таблиці виявлення варіантів використання було створено діаграму варіантів використання платформи (додаток А). Ця діаграма наочно демонструє взаємодію між ключовими акторами системи та основними функціональними можливостями платформи [40].

Після визначення та відображення варіантів використання для різних категорій користувачів було створено функціональну карту вебдодатка.

Функціональність платформи згруповано в кілька логічних розділів. Початковий етап взаємодії користувача із системою охоплює автентифікацію та вибір міста. У межах цих розділів користувач може зареєструватися, увійти до системи, зокрема за допомогою облікового запису Google, знайти потрібне місто, приєднатися до нього або подати заявку на створення нового міста.

Оснoву функціoнальнoсті платфoрми становлять розділи, пов'язані з життєдіяльністю міста. До них належать робота із заявками на вирішення проблем міської інфраструктури, інтерактивна карта, відстеження статусу звернень та обмін повідомленнями між заявником і представниками муніципалітету. Окремими функціoнальними розділами є система оголошень і новин, а також модуль опитувань, який забезпечує проведення голосувань серед учасників міста.

Для користувачів із відповідними повноваженнями передбачено адміністративні розділи. Вони забезпечують керування департаментами, учасниками та ролями в межах окремого міста, а також виконання загальносистемних адміністративних функцій, зокрема розгляд заявок на створення міст, керування містами та обліковими записами користувачів.

Особливістю функціoнальної карти є залежність доступних можливостей від ролі користувача. Відповідно до моделі розмежування доступу окремі сторінки, функції та елементи керування доступні лише користувачам, які мають необхідні дозволи. Загальну структуру та взаємозв'язки функціoнальних компонентів платформи наведено на рис. 2.1.

Усередині кожного розділу діє єдиний принцип переходів: зі списку сутностей (наприклад, переліку заявок чи опитувань) користувач відкриває сторінку конкретного елемента з повним набором деталей і доступних дій, а звідти повертається назад або переходить до іншого модуля міського середовища. Для представників муніципалітету ті самі розділи доступні в розділеному інтерфейсі (split UI): застосунок дає змогу перемикатися між режимом мешканця, у якому контент відображається так, як його бачать звичайні користувачі, та режимом керування з доступом до адміністративних дій. Завдяки цьому одні й ті самі дані подаються у двох ракурсах без дублювання сторінок, а працівник муніципалітету може перевірити, як його зміни виглядатимуть для жителів.

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				
Змн.	Арк.	№ докум.	Підпис	Дата		42

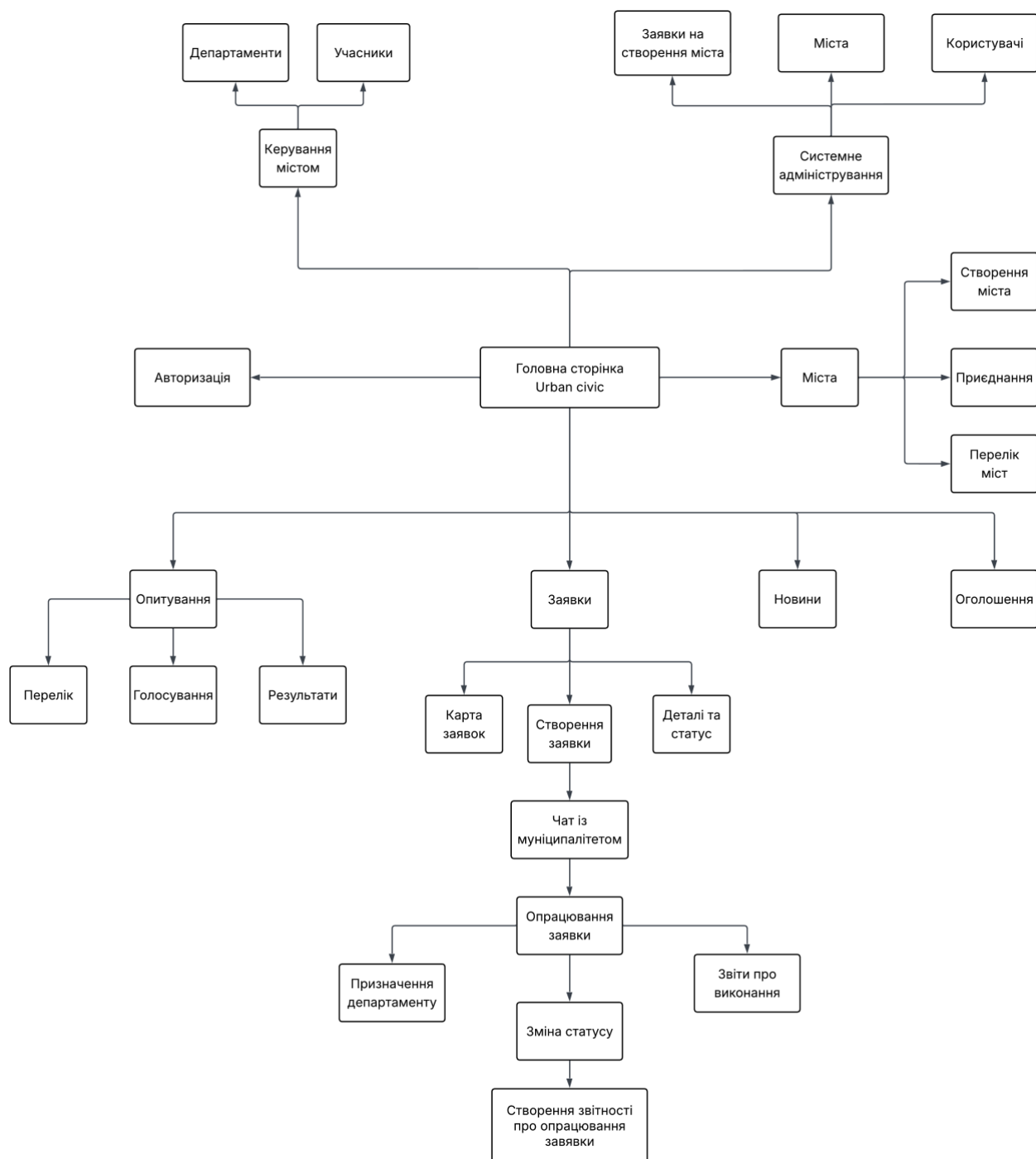


Рис. 2.1. Функціональна карта

Тепер, коли відомі дійові особи й варіанти використання, можна створити попередню об'єктно-орієнтовану модель системи. Для зручності сприйняття класи згруповано за функціональними напрямками платформи та відображено у вигляді окремих діаграм класів [25].

Діаграма класів користувачів та розмежування доступу (рис. 2.2) описує модель автентифікації й рольового доступу. Центральною сутністю є клас User,

що зберігає основні дані облікового запису – ім'я, електронну пошту, хеш пароля, спосіб входу, системну роль та ознаку блокування; системна роль визначається перерахуванням `SystemRole` (звичайний користувач або адміністратор платформи). Оскільки платформа є багатоорендною, належність користувача до конкретного міста фіксує клас `UserCity`. Розмежування прав побудовано за рольовою моделлю: у межах міста користувачеві призначаються ролі (клас `Role` – житель, представник муніципалітету, адміністратор міста) через сполучний клас `UserRole`, а кожна роль пов'язана з набором дозволів, де клас `Permission` описує окремий дозвіл, а клас `RolePermission` зіставляє дозволи з ролями. Завдяки цьому під час кожного запиту система визначає ролі користувача в місті та перевіряє, чи надають відповідні їм дозволи право на виконання потрібної операції.

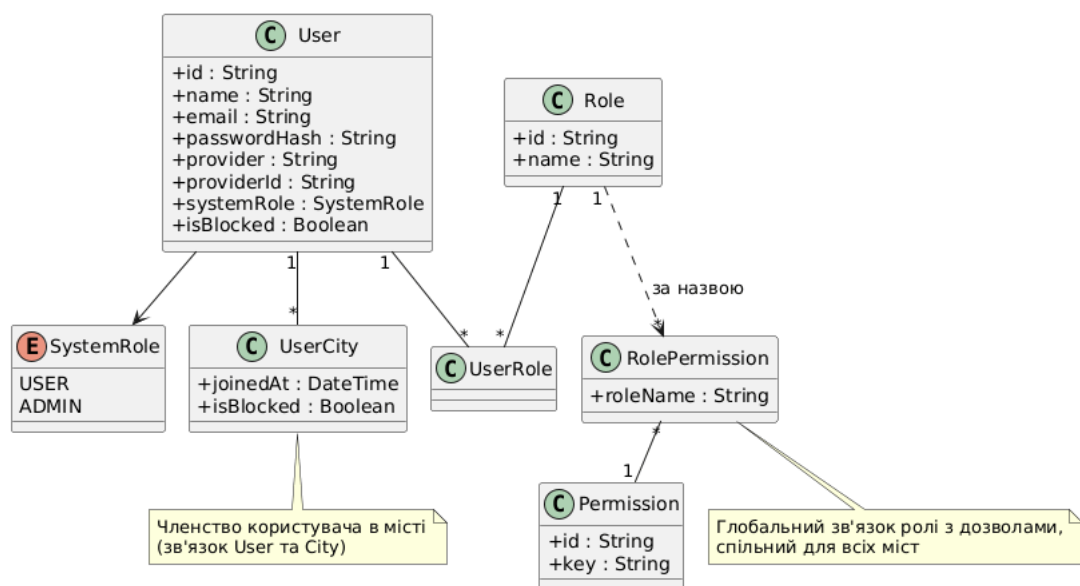


Рис. 2.2. Діаграма класів користувачів та розмежування доступу

Діаграма класів міста (рис. 2.3) відображає багатоорендну модель системи, у якій кожне місто є окремим орендарем (tenant). Клас `City` зберігає назву, регіон та географічні координати центру й виступає кореневою сутністю, до якої прив'язані всі інші дані. Процес створення нового міста передбачає подання користувачем заявки, представленої класом `CityCreationRequest`. Заявка проходить етап розгляду та може перебувати в одному з таких станів: на розгляді, схвалено або відхилено. Для перевірки права власності на офіційний

домен муніципалітету використовується клас DomainVerification. Інформація про успішно підтверджений домен зберігається в класі CityDomain. Після схвалення заявки утворюється місто, у межах якого функціонують департаменти (клас Department) – підрозділи, на які згодом призначаються заявки громадян.

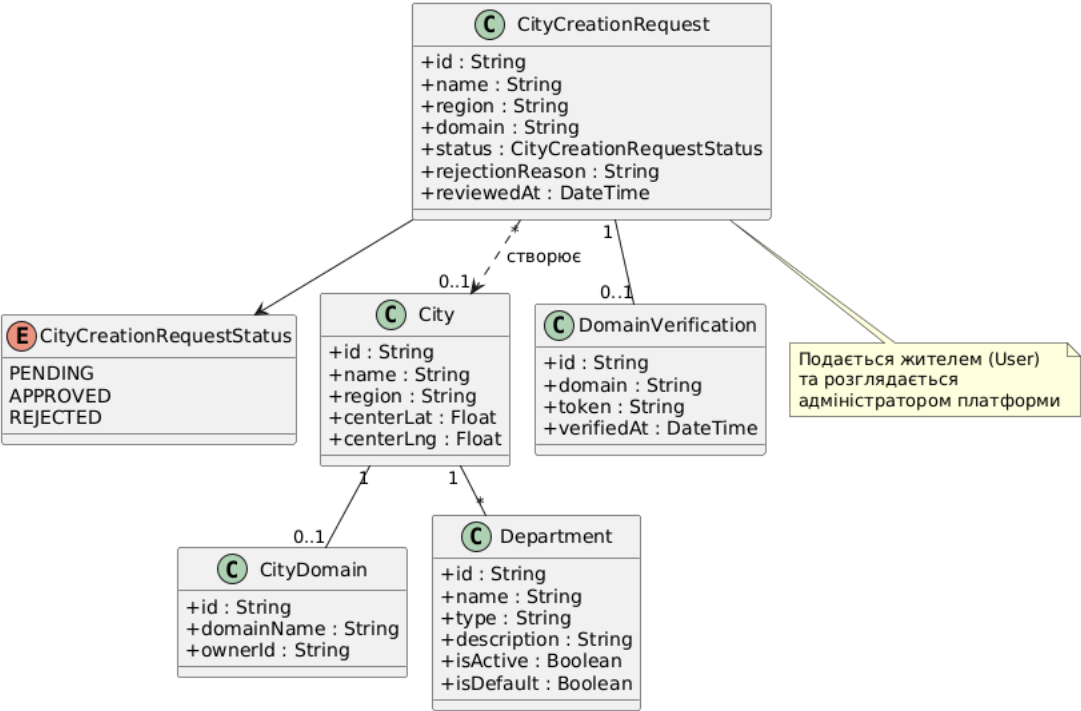


Рис. 2.3. Діаграма класів міста

Діаграма класів заявок (рис. 2.4) є центральною для системи та описує повний життєвий цикл звернення. Основою є клас CityRequest, що містить заголовок, опис, місце проблеми (адресу й географічні координати), пріоритет та статус, який визначається перерахуванням RequestStatus (нове, у роботі, виконано, відхилено). До заявки громадянин може додавати медіадокази – фото чи відео, – представлені класом Attachment, який також пов'язується зі звітами та новинами. Хід і результат виконання заявки фіксуються класом Report, тип якого визначається перерахуванням ReportType (проміжний звіт, про вирішення або про відхилення). Для спілкування заявника з муніципалітетом у межах конкретного звернення створюється клас Chat, у якому накопичуються повідомлення (клас Message). Таким чином, заявка проходить шлях від створення громадянином, через призначення на департамент та обмін

повідомленнями в чаті, до зміни статусу й формування звіту муніципальною службою.

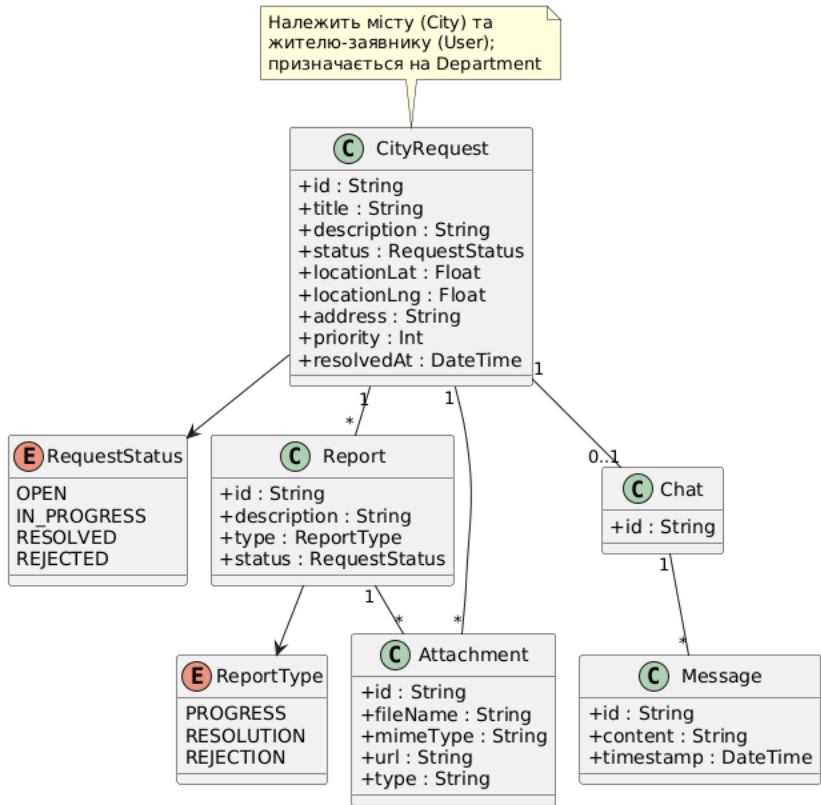


Рис. 2.4. Діаграма класів заявок

Діаграма класів сповіщень (рис. 2.5) відображає механізм надійної доставки внутрішньосистемних сповіщень. Коли в системі відбувається подія, яку потрібно довести до користувачів, вона спершу зберігається у класі DomainEventOutbox – таблиці вихідних подій, що реалізує патерн Transactional Outbox; статус оброблення такої події визначається перерахуванням OutboxStatus (очікує, обробляється, оброблено, помилка). Фоновий обробник зчитує невідправлені події та формує доставки, представлені класом NotificationDelivery, який фіксує факт і статус надсилання кожному отримувачу окремим каналом – внутрішньосистемним або електронною поштою. Власне сповіщення, що відображається користувачеві в інтерфейсі, описує клас InAppNotification, а його різновид визначається перерахуванням InAppNotificationType. Така модель гарантує, що жодне сповіщення не буде втрачене навіть за тимчасових збоїв. Обмеження унікальності записів доставки

та сповіщень водночас унеможливлюють повторне надсилання одного й того самого сповіщення тому самому користувачеві.

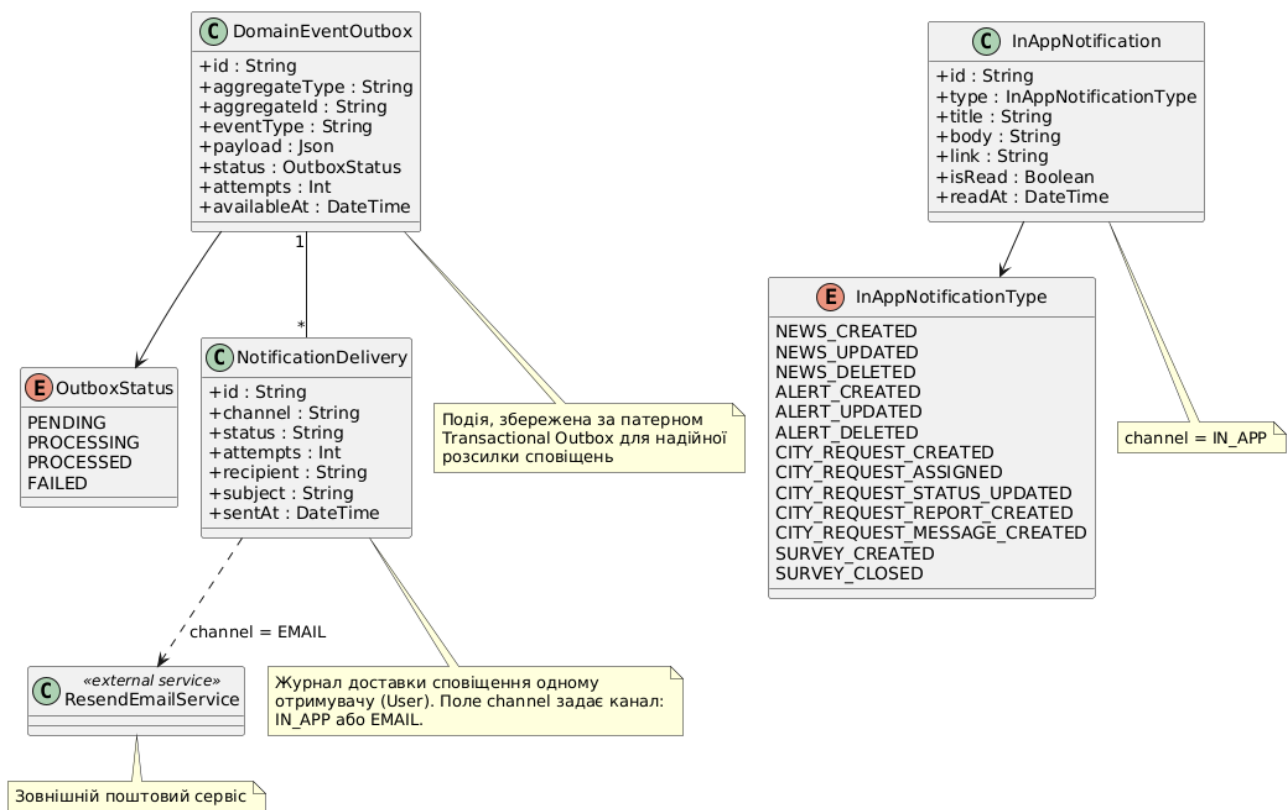


Рис. 2.5. Діаграма класів сповіщень

Діаграма класів оголошень та новин (рис. 2.6) описує засоби інформування громадян муніципалітетом. Термінові оголошення представлено класом **Alert**, що містить заголовок, вміст, строк дії та рівень критичності, який визначається перерахуванням **AlertSeverity** (критичний, високий, середній, низький); кожне оголошення належить до певного типу, описаного класом **AlertType**. Громадяни можуть підписуватися на потрібні типи оголошень – таку підписку фіксує клас **AlertSubscription**. Офіційні новини й оголошення муніципалітету представлено класом **GeneralNews**. Усі ці сутності належать конкретному місту та публікуються його представниками. Публікація нового оголошення чи новини породжує відповідну подію, яка через описаний вище механізм вихідних повідомлень доставляється зацікавленим користувачам у вигляді сповіщень.



Рис. 2.6. Діаграма класів оголошень та новин

Діаграма класів опитувань (рис. 2.7) описує модель проведення голосувань. Клас Survey представляє опитування й містить заголовок, опис, статус (перерахування SurveyStatus – відкрите чи закрите), налаштування видимості результатів (перерахування ResultsVisibility) та строк завершення. Кожному опитуванню належить набір варіантів відповіді, представлених класом SurveyOption. Голос громадянина фіксує клас Vote, який пов'язує користувача, опитування та обраний варіант; обмеження унікальності за парою «опитування – користувач» гарантує, що кожен може віддати лише один голос. Після завершення опитування його результати стають доступними для ознайомлення. Перерахування ResultsVisibility визначає момент оприлюднення результатів – одразу, після власного голосування користувача або лише після закриття опитування, що дозволяє запобігти впливу проміжних результатів на вибір інших учасників. Окреме налаштування опитування визначає, чи дозволено учасникам змінювати свій голос, доки опитування лишається відкритим. Опитування може бути закрите вручну його автором з боку муніципалітету або автоматично після настання запланованого строку завершення, після чого приймання нових голосів припиняється. Кожне опитування належить конкретному місту й доступне лише його учасникам, що забезпечує проведення голосувань у межах відповідної міської спільноти.

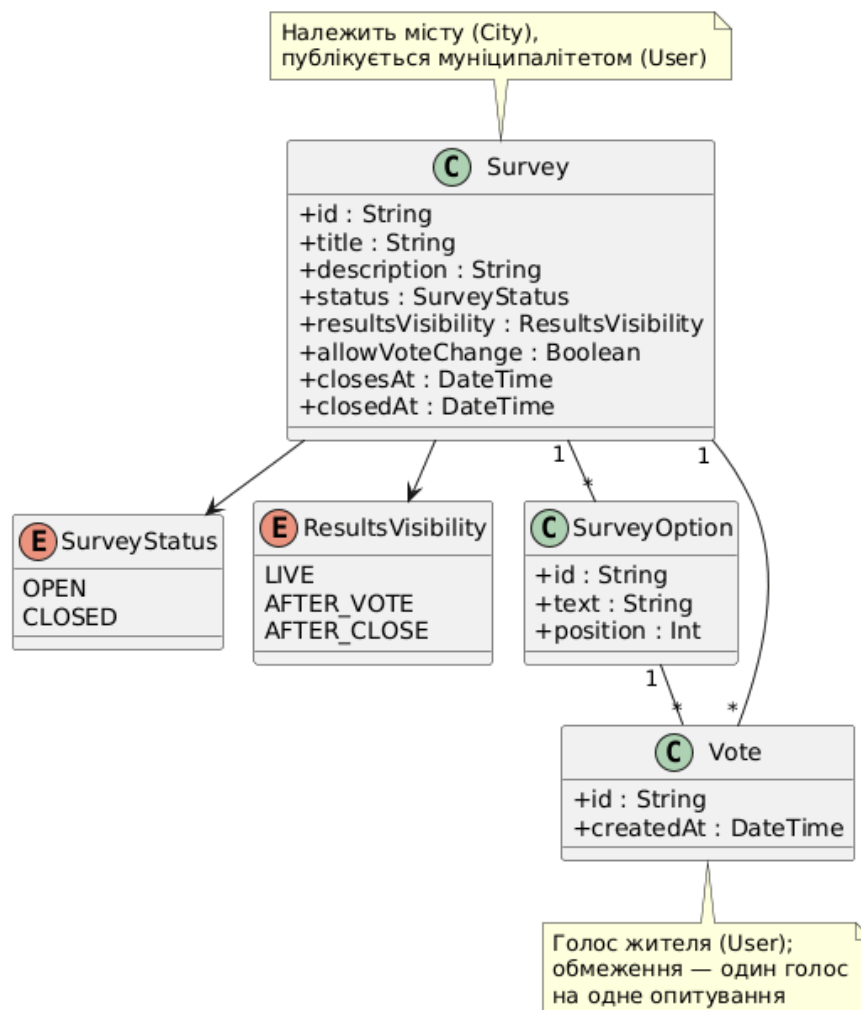


Рис. 2.7. Діаграма класів опитувань

2.2. Розробка структури бази даних вебсервісу

Для зберігання даних платформи Urban Civic використовується об'єктно-реляційна база даних PostgreSQL [26]. Дані системи – користувачі, міста, заявки, сповіщення, новини, опитування – мають чітко визначену та стабільну структуру з численними зв'язками між сутностями. Використання реляційної бази даних забезпечує цілісність даних, ефективні операції читання та запису, а також можливість встановлення складних зв'язків між таблицями за допомогою ключів та обмежень.

Структуру бази даних описано декларативно у схемі Prisma ORM [19], з якої автоматично генеруються таблиці засобами міграцій. Ідентифікатори записів формуються за алгоритмом `uuid` і зберігаються як рядки (тип `text`). Поля-

перерахування реалізовано власними типами PostgreSQL, а поля з гнучкою структурою – типом jsonb. Для сутностей, які не видаляються фізично, передбачено «м'яке» видалення через поле deletedAt. Багатоорендність забезпечується зовнішнім ключем cityId у таблицях, що належать конкретному місту (табл. 2.3) [35].

Таблиця 2.3

Перелік перерахувань

Перерахування	Значення	Опис
RequestStatus	OPEN, IN_PROGRESS, RESOLVED, REJECTED	Статус заявки
SystemRole	USER, ADMIN	Системна роль користувача
CityCreationRequestStatus	PENDING, APPROVED, REJECTED	Статус заявки на створення міста
ReportType	PROGRESS, RESOLUTION, REJECTION	Тип звіту про виконання
SurveyStatus	OPEN, CLOSED	Статус опитування
ResultsVisibility	LIVE, AFTER_VOTE, AFTER_CLOSE	Видимість результатів опитування
AlertSeverity	CRITICAL, HIGH, MEDIUM, LOW	Рівень критичності сповіщення
OutboxStatus	PENDING, PROCESSING, PROCESSED, FAILED	Статус події у таблиці вихідних подій
InAppNotificationType	NEWS_CREATED ... SURVEY_CLOSED	Тип внутрішньосистемного сповіщення

Таблиця «City» є кореневою сутністю багатоорендної моделі та зберігає дані про міста, у межах яких функціонує платформа. До цієї таблиці через зовнішній ключ прив'язані всі інші дані системи (заявки, оголошення, новини, опитування, департаменти та учасники), що забезпечує ізоляцію інформації

кожного міста в межах спільної бази даних. Опис полів таблиці наведено в таблиці 2.4.

Таблиця 2.4

Опис полів таблиці «City»

Назва	Тип даних	ПК	ЗК	Опис
id	text	+	–	Ідентифікатор міста
name	text	–	–	Назва міста
region	text	–	–	Регіон (область)
centerLat	double precision, null	–	–	Широта центру міста
centerLng	double precision, null	–	–	Довгота центру міста
deletedAt	timestamp, null	–	–	Дата м'якого видалення
createdAt	timestamp	–	–	Дата створення
updatedAt	timestamp	–	–	Дата оновлення

Таблиця «CityDomain» зберігає підтверджений офіційний домен міста, що використовується для верифікації під час його створення (табл. 2.5).

Таблиця 2.5

Опис полів таблиці «CityDomain»

Назва	Тип даних	ПК	ЗК	Опис
id	text	+	–	Ідентифікатор домену
domainName	text, unique	–	–	Доменне ім'я міста
ownerId	text	–	–	Ідентифікатор власника домену
cityId	text, unique	–	+	Ідентифікатор міста

Таблиця «User» призначена для зберігання даних про всіх користувачів платформи. Підтримується вхід за паролем та через Google OAuth (табл. 2.6).

Опис полів таблиці «User»

Назва	Тип даних	ПК	ЗК	Опис
id	text	+	–	Ідентифікатор користувача
name	text	–	–	Ім'я користувача
email	text, unique	–	–	Електронна пошта
passwordHash	text, null	–	–	Хеш пароля
provider	text	–	–	Спосіб входу (local / google)
providerId	text, null	–	–	Ідентифікатор у зовнішнього провайдера
systemRole	SystemRole	–	–	Системна роль (USER / ADMIN)
isBlocked	boolean	–	–	Ознака блокування
blockedAt	timestamp, null	–	–	Дата блокування
blockedById	text, null	–	–	Ідентифікатор користувача, який заблокував
createdAt	timestamp	–	–	Дата реєстрації
updatedAt	timestamp	–	–	Дата оновлення

Таблиця «UserCity» встановлює зв'язок «багато до багатьох» між користувачами та містами й фіксує членство користувача в місті (табл. 2.7)

Таблиця 2.7

Опис полів таблиці «UserCity»

Назва	Тип даних	ПК	ЗК	Опис
userId	text	+	+	Ідентифікатор користувача
cityId	text	+	+	Ідентифікатор міста
joinedAt	timestamp	–	–	Дата приєднання до міста
isBlocked	boolean	–	–	Ознака блокування в місті
blockedAt	timestamp, null	–	–	Дата блокування
blockedById	text, null	–	–	Хто заблокував

Таблиця «Role» зберігає ролі, доступні в межах міста. Поєднання міста та назви ролі є унікальним (табл. 2.8).

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				
Змн.	Арк.	№ докум.	Підпис	Дата		52

Таблиця 2.8

Опис полів таблиці «Role»

Назва	Тип даних	ПК	ЗК	Опис
id	text	+	–	Ідентифікатор ролі
name	text	–	–	Назва ролі
cityId	text	–	+	Ідентифікатор міста

Таблиця «Permission» зберігає дозволи, що визначають доступ користувачів до певних функцій платформи (табл. 2.9).

Таблиця 2.9

Опис полів таблиці «Permission»

Назва	Тип даних	ПК	ЗК	Опис
id	text	+	–	Ідентифікатор дозволу
key	text, unique	–	–	Ключ дозволу (наприклад, city_request:create)

Таблиця «UserRole» встановлює зв'язок «багато до багатьох» між користувачами та ролями (табл. 2.10).

Таблиця 2.10

Опис полів таблиці «UserRole»

Назва	Тип даних	ПК	ЗК	Опис
userId	text	+	+	Ідентифікатор користувача
roleId	text	+	+	Ідентифікатор ролі

Таблиця «RolePermission» встановлює зв'язок «багато до багатьох» між ролями та дозволами. Роль ідентифікується назвою без зовнішнього ключа (табл. 2.11).

Опис полів таблиці «RolePermission»

Назва	Тип даних	ПК	ЗК	Опис
roleName	text	+	–	Назва ролі
permissionId	text	+	+	Ідентифікатор дозволу

Таблиця «CityRequest» є центральною сутністю системи та зберігає заявки жителів на вирішення проблем міської інфраструктури (табл. 2.12).

Таблиця 2.12

Опис полів таблиці «CityRequest»

Назва	Тип даних	ПК	ЗК	Опис
id	text	+	–	Ідентифікатор заявки
cityId	text	–	+	Ідентифікатор міста
userId	text	–	+	Ідентифікатор заявника
title	text	–	–	Заголовок заявки
description	text, null	–	–	Опис проблеми
status	RequestStatus	–	–	Статус заявки
locationLat	double precision, null	–	–	Широта місця проблеми
locationLng	double precision, null	–	–	Довгота місця проблеми
address	text, null	–	–	Адреса місця проблеми
priority	integer	–	–	Пріоритет заявки
assignedDepartmentId	text, null	–	+	Призначений департамент
resolvedAt	timestamp, null	–	–	Дата вирішення
createdAt	timestamp	–	–	Дата створення
updatedAt	timestamp	–	–	Дата оновлення

Таблиця «CityCreationRequest» зберігає заявки користувачів на створення нового міста та результат їх розгляду адміністратором (табл. 2.13).

Опис полів таблиці «CityCreationRequest»

Назва	Тип даних	ПК	ЗК	Опис
id	text	+	–	Ідентифікатор заявки
requesterId	text	–	+	Ідентифікатор заявника
reviewedById	text, null	–	+	Ідентифікатор адміністратора-рецензента
cityId	text, null	–	+	Ідентифікатор створеного міста
domainVerificationId	text, unique, null	–	+	Підтвердження домену
name	text	–	–	Назва міста
region	text	–	–	Регіон
centerLat	double precision, null	–	–	Широта центру
centerLng	double precision, null	–	–	Довгота центру
domain	text, null	–	–	Доменне ім'я
status	CityCreationRequestStatus	–	–	Статус розгляду
rejectionReason	text, null	–	–	Причина відхилення
reviewedAt	timestamp, null	–	–	Дата розгляду
createdAt	timestamp	–	–	Дата створення
updatedAt	timestamp	–	–	Дата оновлення

Таблиця «DomainVerification» зберігає дані для підтвердження права власності на домен муніципалітету під час створення міста (табл. 2.14).

Таблиця 2.14

Опис полів таблиці «DomainVerification»

Назва	Тип даних	ПК	ЗК	Опис
id	text	+	–	Ідентифікатор підтвердження
requesterId	text	–	+	Ідентифікатор заявника
domain	text	–	–	Домен, що підтверджується
token	text, unique	–	–	Токен підтвердження
verifiedAt	timestamp, null	–	–	Дата підтвердження

продовж. табл. 2.14

createdAt	timestamp	–	–	Дата створення
updatedAt	timestamp	–	–	Дата оновлення

Таблиця «Department» зберігає департаменти міста, на які призначаються заявки (табл. 2.15).

Таблиця 2.15

Опис полів таблиці «Department»

Назва	Тип даних	ПК	ЗК	Опис
id	text	+	–	Ідентифікатор департаменту
cityId	text	–	+	Ідентифікатор міста
name	text	–	–	Назва департаменту
type	text	–	–	Тип департаменту
description	text, null	–	–	Опис
isActive	boolean	–	–	Ознака активності
isDefault	boolean	–	–	Ознака департаменту за замовчуванням
createdAt	timestamp	–	–	Дата створення
updatedAt	timestamp	–	–	Дата оновлення

Таблиця «Report» зберігає звіти про хід або результат виконання заявки, які формуються представниками муніципальних служб у процесі її опрацювання. Опис полів таблиці наведено в таблиці 2.16.

Таблиця 2.16

Опис полів таблиці «Report»

Назва	Тип даних	ПК	ЗК	Опис
id	text	+	–	Ідентифікатор звіту
description	text, null	–	–	Текст звіту
authorId	text	–	+	Ідентифікатор автора
type	ReportType	–	–	Тип звіту
status	RequestStatus, null	–	–	Статус, встановлений звітом
cityRequestId	text	–	+	Ідентифікатор заявки
createdAt	timestamp	–	–	Дата створення

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				
Змн.	Арк.	№ докум.	Підпис	Дата		56

updatedAt	timestamp	–	–	Дата оновлення
-----------	-----------	---	---	----------------

Таблиця «Attachment» зберігає метадані вкладень до заявок, звітів та новин – ім'я файлу, його тип і посилання на нього, тоді як самі файли зберігаються в окремому об'єктному сховищі, що розвантажує базу даних і забезпечує ефективну роботу з великими медіафайлами. Опис полів таблиці наведено в таблиці 2.17.

Таблиця 2.17

Опис полів таблиці «Attachment»

Назва	Тип даних	ПК	ЗК	Опис
id	text	+	–	Ідентифікатор вкладки
fileName	text	–	–	Ім'я файлу
mimeType	text, null	–	–	MIME-тип файлу
url	text	–	–	Посилання на файл у сховищі
type	text, null	–	–	Тип вкладки
reportId	text, null	–	+	Звіт
cityRequestId	text, null	–	+	Заявка
cityCreationRequestId	text, null	–	+	Заявка на створення міста
newsId	text, null	–	+	Новина
cityId	text, null	–	+	Місто
uploadedAt	timestamp	–	–	Дата завантаження

Таблиця «Chat» зберігає чат комунікації між заявником і муніципалітетом у межах конкретного звернення, причому зв'язок із заявкою є взаємно однозначним – кожній заявці відповідає рівно один чат, у якому накопичуються всі повідомлення сторін. Опис полів таблиці наведено в таблиці 2.18.

Таблиця 2.18

Опис полів таблиці «Chat»

Назва	Тип даних	ПК	ЗК	Опис
id	text	+	–	Ідентифікатор чату
cityRequestId	text, unique	–	+	Ідентифікатор заявки
createdAt	timestamp	–	–	Дата створення

Таблиця «Message» зберігає повідомлення в чаті заявки (табл. 2.19).

Таблиця 2.19

Опис полів таблиці «Message»

Назва	Тип даних	ПК	ЗК	Опис
id	text	+	–	Ідентифікатор повідомлення
chatId	text	–	+	Ідентифікатор чату
authorId	text	–	+	Ідентифікатор автора
content	text	–	–	Текст повідомлення
timestamp	timestamp	–	–	Час надсилання

Таблиця «Alert» зберігає термінові сповіщення міста із зазначенням рівня критичності та строку дії (табл. 2.20).

Таблиця 2.20

Опис полів таблиці «Alert»

Назва	Тип даних	ПК	ЗК	Опис
id	text	+	–	Ідентифікатор сповіщення
cityId	text	–	+	Ідентифікатор міста
publisherId	text, null	–	+	Ідентифікатор автора
alertTypeId	text	–	+	Тип сповіщення
severity	AlertSeverity	–	–	Рівень критичності
expiresAt	timestamp, null	–	–	Строк дії
title	text	–	–	Заголовок
content	text	–	–	Вміст
createdAt	timestamp	–	–	Дата створення
updatedAt	timestamp	–	–	Дата оновлення
deletedAt	timestamp, null	–	–	Дата м'якого видалення

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				
Змн.	Арк.	№ докум.	Підпис	Дата		58

Таблиця «AlertType» зберігає типи термінових сповіщень (табл. 2.21). Назва типу є унікальною.

Таблиця 2.21

Опис полів таблиці «AlertType»

Назва	Тип даних	ПК	ЗК	Опис
id	text	+	–	Ідентифікатор типу
name	text, unique	–	–	Назва типу сповіщення

Таблиця «AlertSubscription» встановлює зв'язок «багато до багатьох» між користувачами та типами сповіщень міста (табл. 2.22).

Таблиця 2.22

Опис полів таблиці «AlertSubscription»

Назва	Тип даних	ПК	ЗК	Опис
userId	text	+	+	Ідентифікатор користувача
cityId	text	+	+	Ідентифікатор міста
alertTypeId	text	+	+	Ідентифікатор типу сповіщення

Таблиця «Survey» зберігає опитування міста з параметрами видимості результатів, можливості зміни голосу та строку завершення, а також зі статусом, що відображає, чи є опитування відкритим для голосування, чи вже закритим. Опис полів таблиці наведено в таблиці 2.23.

Таблиця 2.23

Опис полів таблиці «Survey»

Назва	Тип даних	ПК	ЗК	Опис
id	text	+	–	Ідентифікатор опитування
cityId	text	–	+	Ідентифікатор міста
publisherId	text, null	–	+	Ідентифікатор автора
title	text	–	–	Заголовок
description	text, null	–	–	Опис
status	SurveyStatus	–	–	Статус (відкрите / закрите)
resultsVisibility	ResultsVisibility	–	–	Видимість результатів
allowVoteChange	boolean	–	–	Чи дозволено змінювати голос

продовж. табл. 2.23

closesAt	timestamp, null	–	–	Запланована дата закриття
closedAt	timestamp, null	–	–	Фактична дата закриття
createdAt	timestamp	–	–	Дата створення
updatedAt	timestamp	–	–	Дата оновлення
deletedAt	timestamp, null	–	–	Дата м'якого видалення

Таблиця «SurveyOption» зберігає варіанти відповіді опитування та їх порядок відображення, причому кожен варіант належить конкретному опитуванню й виступає об'єктом, за який голосують учасники. Опис полів таблиці наведено в таблиці 2.24.

Таблиця 2.24

Опис полів таблиці «SurveyOption»

Назва	Тип даних	ПК	ЗК	Опис
id	text	+	–	Ідентифікатор варіанта
surveyId	text	–	+	Ідентифікатор опитування
text	text	–	–	Текст варіанта
position	integer	–	–	Порядковий номер варіанта

Таблиця «Vote» зберігає голоси користувачів. Обмеження унікальності (surveyId + userId) гарантує один голос на опитування (табл. 2.25).

Таблиця 2.25

Опис полів таблиці «Vote»

Назва	Тип даних	ПК	ЗК	Опис
id	text	+	–	Ідентифікатор голосу
surveyId	text	–	+	Ідентифікатор опитування
surveyOptionId	text	–	+	Ідентифікатор обраного варіанта
userId	text	–	+	Ідентифікатор користувача
createdAt	timestamp	–	–	Дата створення
updatedAt	timestamp	–	–	Дата оновлення

Таблиця «GeneralNews» зберігає офіційні новини та оголошення міста (табл. 2.26).

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				
Змн.	Арк.	№ докум.	Підпис	Дата		60

Опис полів таблиці «GeneralNews»

Назва	Тип даних	ПК	ЗК	Опис
id	text	+	–	Ідентифікатор новини
cityId	text	–	+	Ідентифікатор міста
publisherId	text	–	+	Ідентифікатор автора
title	text	–	–	Заголовок
content	text	–	–	Вміст новини
createdAt	timestamp	–	–	Дата створення
updatedAt	timestamp	–	–	Дата оновлення
deletedAt	timestamp, null	–	–	Дата м'якого видалення

Таблиця «DomainEventOutbox» реалізує патерн Transactional Outbox: зберігає доменні події, що виникають під час зміни стану системи, у межах однієї транзакції з основними даними, забезпечуючи їх подальшу надійну розсилку у вигляді сповіщень без ризику втрати. Опис полів таблиці наведено в таблиці 2.27.

Таблиця 2.27

Опис полів таблиці «DomainEventOutbox»

Назва	Тип даних	ПК	ЗК	Опис
id	text	+	–	Ідентифікатор події
aggregateType	text	–	–	Тип агрегату-джерела події
aggregateId	text	–	–	Ідентифікатор агрегату
eventType	text	–	–	Тип події
payload	jsonb	–	–	Корисне навантаження події
status	OutboxStatus	–	–	Статус обробки
attempts	integer	–	–	Кількість спроб
availableAt	timestamp	–	–	Час, з якого подію можна опрацьовувати
lastError	text, null	–	–	Текст останньої помилки
createdAt	timestamp	–	–	Дата створення
updatedAt	timestamp	–	–	Дата оновлення

Таблиця «InAppNotification» зберігає внутрішньосистемні сповіщення користувачів, що відображаються в інтерфейсі (табл. 2.28).

Таблиця 2.28

Опис полів таблиці «InAppNotification»

Назва	Тип даних	ПК	ЗК	Опис
id	text	+	–	Ідентифікатор сповіщення
userId	text	–	+	Ідентифікатор користувача-отримувача
cityId	text	–	+	Ідентифікатор міста
type	InAppNotificationType	–	–	Тип сповіщення
eventId	text, null	–	–	Ідентифікатор події-джерела
title	text	–	–	Заголовок
body	text, null	–	–	Текст
link	text, null	–	–	Посилання
payload	jsonb, null	–	–	Додаткові дані
isRead	boolean	–	–	Ознака прочитання
readAt	timestamp, null	–	–	Дата прочитання
createdAt	timestamp	–	–	Дата створення

Таблиця «NotificationDelivery» є журналом доставки сповіщень окремим отримувачам різними каналами (IN_APP або EMAIL) (табл. 2.29).

Таблиця 2.29

Опис полів таблиці «NotificationDelivery»

Назва	Тип даних	ПК	ЗК	Опис
id	text	+	–	Ідентифікатор доставки
outboxEventId	text	–	+	Ідентифікатор події
userId	text	–	+	Ідентифікатор отримувача
channel	text	–	–	Канал доставки (IN_APP / EMAIL)
status	text	–	–	Статус доставки
attempts	integer	–	–	Кількість спроб
recipient	text	–	–	Адресат (наприклад, email)
subject	text	–	–	Тема повідомлення

lastError	text, null	–	–	Текст останньої помилки
sentAt	timestamp, null	–	–	Дата відправлення
createdAt	timestamp	–	–	Дата створення
updatedAt	timestamp	–	–	Дата оновлення

На основі схеми Prisma за допомогою генератора діаграм було побудовано ER-діаграму бази даних PostgreSQL (додаток Б) [42]. Ця діаграма у вигляді «сутність–зв’язок» детально відображає структуру бази даних: всі таблиці та їхні стовпці з вказаними типами даних, первинні та зовнішні ключі, а також зв’язки між таблицями.

2.3. Проєктування та реалізація алгоритмів роботи системи

Серверну частину платформи Urban Civic побудовано на фреймворку NestJS навколо принципу інверсії керування та механізму впровадження залежностей [6]. Контейнер залежностей створює екземпляри класів-провайдерів (сервісів, репозиторіїв, шлюзів) як одинаки і автоматично передає їх туди, де вони оголошені в конструкторі. Завдяки цьому класи не створюють свої залежності самостійно, а лише декларують потребу в них, що зменшує зв’язаність компонентів, спрощує заміну реалізацій та модульне тестування, оскільки залежності легко підмінюються макетами. Кодова база організована за модульним принципом: кожен функціональний напрям – автентифікація, заявки, сповіщення, новини, опитування, розмежування доступу – інкапсульовано в окремий модуль NestJS, який оголошує власні контролери та провайдери й експортує назовні лише потрібне.

Обробка кожного HTTP-запиту відбувається конвеєром із чітким розподілом відповідальності. Спершу запит потрапляє до контролера, який відповідає лише за маршрутизацію та приймання вхідних даних. Перед виконанням методу контролера дані проходять через канал валідації: глобальний механізм перевірки звіряє тіло запиту з об’єктом передачі даних, правила якого описані деклараторами бібліотеки class-validator, та за потреби перетворює типи.

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				
Змн.	Арк.	№ докум.	Підпис	Дата		63

Це гарантує, що до бізнес-логіки потрапляють лише коректні дані, а правила валідації зосереджені в одному місці й не дублюються в коді контролерів. Далі запит проходить через ланцюг охоронців, після чого керування передається до сервісу, який містить бізнес-логіку, а доступ до бази даних здійснюється через єдиний рівень доступу до даних.

Механізм перевірки прав доступу побудовано на основі патерна «Охоронець» (Guard), який передбачає попередню перевірку умов перед виконанням основної дії. У системі Urban Civic цей патерн реалізовано засобами NestJS через клас PermissionsGuard.

Для опису правил доступу використовується патерн «Декоратор» (Decorator). На рівні методів контролера задається декоратор `@RequirePermissions(...)`, у якому вказуються дозволи, необхідні для виконання певної операції. Сам декоратор не перевіряє користувача, а лише додає до методу службові метадані з переліком потрібних прав.

Під час надходження HTTP-запиту спочатку виконується алгоритм перевірки доступу. PermissionsGuard зчитує метадані методу через Reflector, визначає місто, у межах якого виконується дія, та передає ці дані до RbacService. Далі сервіс перевіряє відповідність ролей і дозволів поточного користувача потрібній операції. Якщо потрібного дозволу немає, виконання методу контролера блокується ще до переходу до бізнес-логіки.

Алгоритм автентифікації спроектовано на основі патерна «Стратегія» (Strategy). Кожен спосіб перевірки облікових даних винесено в окрему стратегію Passport. Локальна стратегія обробляє вхід за електронною поштою та паролем, JWT-стратегія перевіряє токен доступу під час звернення до захищених маршрутів, а окремі стратегії відповідають за оновлення токенів.

Для входу через Google використовується стратегія OAuth 2.0. Після підтвердження облікового запису Google сервер отримує дані профілю користувача, знаходить або створює відповідний запис у системі та формує власні токени доступу. На рівні контролерів потрібний спосіб автентифікації підключається через відповідний охоронець Passport.

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				
Змн.	Арк.	№ докум.	Підпис	Дата		64

Робота з базою даних побудована з використанням патерна «Адаптер». У системі таким адаптером виступає PrismaService, який надає єдину точку доступу до PostgreSQL через Prisma Client. Сервіси не працюють із підключенням до бази напряду, а звертаються до методів Prisma через спільний сервіс.

Для складніших операцій із даними застосовується патерн «Репозиторій» (Repository) [43]. Наприклад, OutboxRepository інкапсулює роботу з подіями, які очікують обробки: отримання невідправлених записів, блокування їх для обробки, зміну статусу після успішного передавання або фіксацію помилки. Такий репозиторій є окремою частиною алгоритму обробки доменних подій.

Для формування подій системи застосовано патерн «Фабрика» (Factory) [37]. Спеціалізовані фабричні функції створюють уніфіковане корисне навантаження доменних подій (для заявок, сповіщень, новин та опитувань), додаючи до нього обов'язкові службові поля – унікальний ідентифікатор події, час виникнення, ідентифікатори міста та автора. Винесення побудови подій у фабрику гарантує єдину структуру навантаження незалежно від місця створення події та усуває дублювання коду. Фабричний підхід застосовано і до конфігурації застосунку: налаштування (база даних, JWT, об'єктне сховище, пошта) описані фабриками конфігурації, що повертають згруповані параметри й роблять їх типобезпечно доступними через службу конфігурації.

Ключовим алгоритмічним рішенням серверної частини є надійна доставка сповіщень за патерном «Вихідні повідомлення» [29]. Коли в системі відбувається подія, яка має сповістити користувачів – публікація термінового сповіщення чи новини, створення заявки, зміна її статусу, додавання звіту, нове повідомлення в чаті, створення або закриття опитування, – сервіс у межах однієї транзакції з основною зміною даних додатково записує подію до таблиці вихідних подій. Оскільки обидві операції виконуються атомарно, подія не може бути втрачена, якщо основна зміна збереглася, і не може «осиротіти», якщо транзакція відкотилася. Корисне навантаження події формується описаною вище фабрикою, що забезпечує його узгоджену структуру.

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				65
Змн.	Арк.	№ докум.	Підпис	Дата		

Фактичне розсилання подій винесено з основного потоку обробки запиту й виконується асинхронно фоновим диспетчером подій. Цей обробник реалізовано з використанням хуків життєвого циклу модуля: під час ініціалізації він запускає періодичне опитування таблиці вихідних подій із заданим інтервалом, а під час завершення роботи коректно зупиняє таймер. На кожному циклі релей вибирає партію невідправлених подій і передає кожну з них до черги завдань через клас-постачальник. Реалізовано захист від повторного входження: якщо попередній цикл ще не завершився, наступний пропускається, що унеможливорює паралельну обробку тієї самої партії.

Доставка реалізована за патерном «Постачальник–Споживач» на основі черги завдань BullMQ, що працює на сховищі Redis [27]. Постачальник додає до черги завдання з ідентифікатором події, а споживач (обробник черги) забирає його та виконує власне розсилання за кількома каналами: створює внутрішньосистемні сповіщення для учасників міста та, для подій, що цього потребують, формує й надсилає електронні листи через зовнішній поштовий сервіс. Винесення доставки в чергу робить обробку запиту користувача швидкою (воно не очікує на розсилання) та забезпечує повторні спроби в разі тимчасових збоїв. Повторне доставлення не призводить до дублювання завдяки обмеженням унікальності: внутрішньосистемне сповіщення унікальне за парою «користувач–подія», а запис журналу доставки – за трійкою «подія–користувач–канал», що реалізує властивість ідемпотентності.

Взаємодію в реальному часі реалізовано за патерном «Спостерігач» (Observer) через вебсокет-шлюз на основі Socket.IO [41]. Під час встановлення з’єднання шлюз автентифікує клієнта за токеном і приєднує його до «кімнати» відповідної заявки. Коли в межах заявки відбувається зміна – нове повідомлення в чаті, зміна статусу, призначення департаменту чи додавання звіту, – сервіс через шлюз надсилає подію всім учасникам відповідної кімнати, і їхній інтерфейс оновлюється миттєво, без повторних запитів до сервера. Такий підхід забезпечує оперативність комунікації між заявником і муніципалітетом.

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				66
Змн.	Арк.	№ докум.	Підпис	Дата		

Однією з ключових операцій системи є подання та опрацювання заявки. Цей процес можна проілюструвати за допомогою діаграми діяльності, яка містить дві доріжки (рис. 2.8). Житель створює заявку із зазначенням опису, місця на карті та медіадоказів, після чого заявка отримує статус «нове» й стає доступною муніципалітету. Представник муніципалітету розглядає заявку, призначає відповідальний департамент та переводить її в статус «у роботі». У межах заявки сторони обмінюються повідомленнями в чаті для уточнення деталей. Після виконання робіт муніципалітет формує звіт та переводить заявку у статус «виконано» або, якщо заявку неможливо опрацювати, у статус «відхилено» із зазначенням причини. Про кожну зміну заявник отримує сповіщення, після чого процес переходить до кінцевого стану.

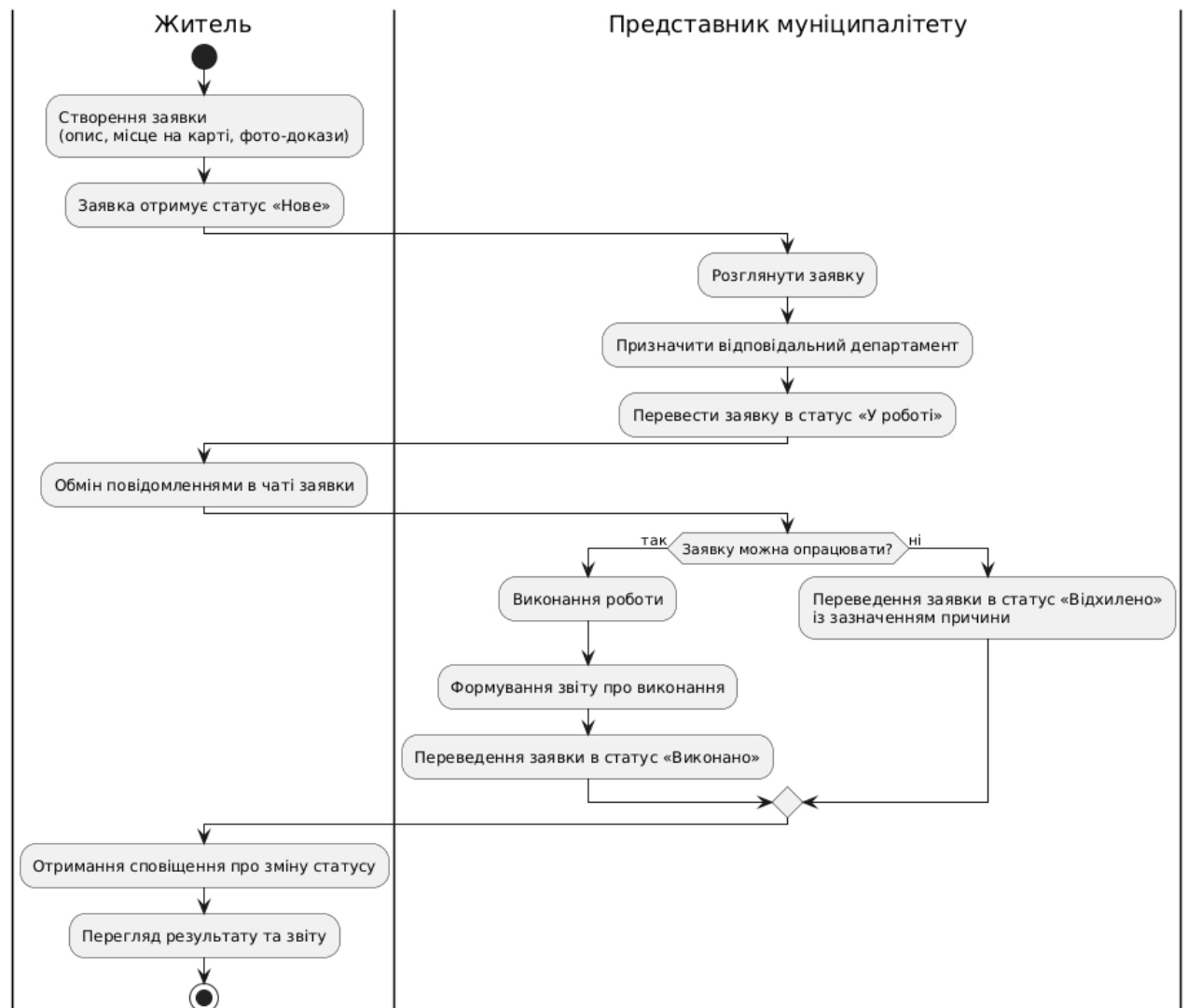


Рис. 2.8. Діаграма діяльності опрацювання звернення

Наступним важливим алгоритмом є надійна доставка сповіщень за патерном вихідних повідомлень. Цей процес зображено на діаграмі діяльності (рис. 2.9), доріжки якої відповідають серверу обробки запиту, фоновому релею, черзі завдань та обробникові доставки. Коли муніципалітет публікує сповіщення, сервер у межах однієї транзакції зберігає сповіщення та запис події у таблицю вихідних подій і одразу повертає відповідь користувачеві. Фоновий релей періодично вибирає невідправлені події та передає їх до черги завдань. Обробник черги забирає завдання й розсилає сповіщення каналами доставки – створює внутрішньосистемні сповіщення та, за потреби, надсилає електронні листи. У разі успіху подію позначають як оброблену, а в разі помилки вона залишається для повторної спроби, що гарантує доставку без втрат і без дублювання.

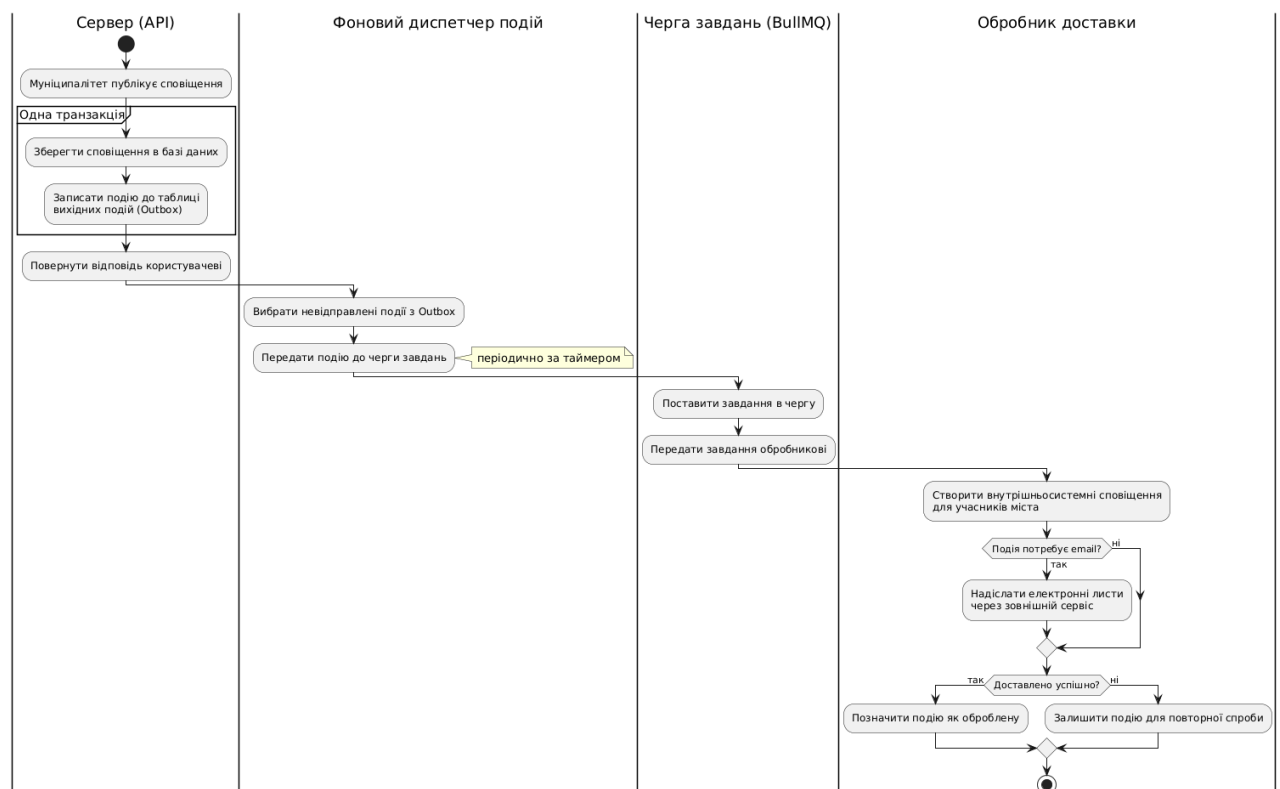


Рис. 2.9. Діаграма діяльності доставки сповіщень

2.4. Реалізація функціональної частини системи

Розробка серверного RESTful API на NestJS виконувалася так: було описано модулі предметних областей, у кожному з них створено контролери та

сервіси, налаштовано єдиний рівень доступу до даних на основі Prisma, реалізовано автентифікацію за токенами та авторизацію на основі прав доступу.

Доступ до бази даних інкапсульовано у спільному провайдері PrismaService, який є єдиною точкою взаємодії з PostgreSQL. Він успадковує згенерований клієнт Prisma та підключається до бази через драйверний адаптер PrismaPg з налаштуваннями пулу з'єднань, а хуки життєвого циклу onModuleInit і onModuleDestroy відкривають та коректно закривають з'єднання разом із застосунком. Завдяки впровадженню залежностей цей сервіс надається будь-якому іншому сервісу системи без створення додаткових з'єднань.

У конструкторі сервіс отримує рядок підключення з конфігураційного сервісу та перевіряє його наявність – відсутність значення призводить до негайного викидання помилки на етапі старту. Адаптер PrismaPg налаштовується з параметрами пулу: максимальна кількість одночасних з'єднань (poolMax, за замовчуванням 5), таймаут очікування з'єднання (connectionTimeoutMs, 10 секунд), таймаут простою (idleTimeoutMs, 30 секунд) та таймаут виконання запиту (queryTimeoutMs, 15 секунд). Усі ці значення зчитуються з конфігурації застосунку, що дозволяє змінювати їх без перекомпіляції коду.

Лістинг провайдера PrismaService.

@Injectable()

```
export class PrismaService
  extends PrismaClient
  implements OnModuleInit, OnModuleDestroy
{
  constructor(configService: ConfigService) {
    const connectionString = configService.get<string>('db.url');
    const adapter = new PrismaPg({
      connectionString,
      max: configService.get<number>('db.poolMax') ?? 5,
      // connectionTimeout, idleTimeout, queryTimeout
    });
  }
}
```

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				
Змн.	Арк.	№ докум.	Підпис	Дата		69

```
});
super({ adapter }); }
async onModuleInit() { await this.$connect(); }
async onModuleDestroy() { await this.$disconnect(); }}
```

Механізм генерації токена доступу JWT використовується для автентифікації користувачів та авторизації їхнього доступу до ресурсів системи. Для створення токена доступу реалізовано метод `generateAccessToken`, який приймає сутність користувача [18]. У корисне навантаження токена додаються ідентифікатор користувача (`sub`), електронна пошта, системна роль, ознака блокування та перелік дозволів користувача, попередньо обчислений сервісом розмежування доступу. Це дозволяє клієнтській частині приймати рішення про відображення елементів інтерфейсу без додаткових запитів. Термін дії токена визначений у конфігурації платформи.

```
private async generateAccessToken(user: User): Promise<string> {
  const permissions = await this.rbacService.getUserPermissionsGlobal(user.id);
  return this.jwtService.sign(
    {
      sub: user.id,
      email: user.email,
      permissions,
      systemRole: user.systemRole,
      isBlocked: user.isBlocked,
    },
    { expiresIn: Number(this.configService.get('jwt.expiresIn')) },
  );
}
```

Після налаштування токена доступу реалізовано генерацію токена оновлення, який дозволяє користувачам отримувати новий токен доступу без повторного введення пароля, коли термін дії поточного токена доступу завершується. Токен оновлення підписується окремим секретним ключем і має

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				
Змн.	Арк.	№ докум.	Підпис	Дата		70

триваліший строк дії; обидва токени встановлюються клієнту у вигляді захищених cookie.

Лістинг методу generateRefreshToken:

```
private generateRefreshToken(user: User): string {  
    return this.jwtService.sign(  
        { sub: user.id, email: user.email },  
        {  
            secret: this.configService.get('jwt.refreshSecret'),  
            expiresIn: Number(this.configService.get('jwt.refreshExpiresIn')),  
        },  
    )  
}
```

Для забезпечення авторизації на основі прав доступу використовується власний декоратор `@RequirePermissions(...)`, який оголошує над методом контролера перелік необхідних дозволів. Декоратор лише прикріплює метадані до методу за допомогою `SetMetadata`, не виконуючи самої перевірки.

Лістинг декоратора RequirePermissions:

```
export const PERMISSIONS_KEY = 'permissions';  
export const RequirePermissions = (...permissions: string[]) =>  
    SetMetadata(PERMISSIONS_KEY, permissions);
```

Безпосередньо перевірку доступу виконує охоронець `PermissionsGuard`. На початку методу `canActivate` він за допомогою службового об'єкта `Reflector` зчитує оголошені метадані одразу з двох рівнів – методу-обробника та класу контролера, що дозволяє визначати дозволи як для окремого ендпоінту, так і для всього контролера. Якщо метадані відсутні, охоронець повертає `true`, пропускаючи запит без перевірки – це забезпечує роботу публічних маршрутів без додаткових декораторів.

За наявності дозволів охоронець отримує ідентифікатор користувача з об'єкта запиту, збагаченого попереднім `JwtGuard`, та визначає ідентифікатор міста – спочатку з параметрів маршруту, а за їх відсутності з тіла запиту. Такий підхід робить охоронець універсальним: він однаково працює як для маршрутів

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				71
Змн.	Арк.	№ докум.	Підпис	Дата		

виду /city/:cityId/..., так і для запитів, де cityId передається у тілі. Перевірка кожного з необхідних дозволів виконується паралельно через Promise.all зверненням до RbacService, після чого результати агрегуються методом every(Boolean) – доступ надається лише за умови, що користувач має всі необхідні дозволи в межах цього міста. Будь-яка помилка у процесі перевірки перехоплюється блоком catch і повертає false, гарантуючи безпечну поведінку за замовчуванням.

Лістинг методу canActivate охоронця PermissionsGuard:

```
async canActivate(context: ExecutionContext): Promise<boolean> {
  const requiredPermissions = this.reflector.getAllAndOverride<string[]>(
    PERMISSIONS_KEY,
    [context.getHandler(), context.getClass()],
  );
  if (!requiredPermissions || requiredPermissions.length === 0) return true;
  const request = context.switchToHttp().getRequest<RequestWithUser>();
  const { id: userId } = request.user;
  const cityId = request.params.cityId ?? request.body.cityId;
  const results = await Promise.all(
    requiredPermissions.map((p) =>
      this.rbacService.hasPermission(userId, cityId, p),
    ),
  );
  return results.every(Boolean);
}
```

Логіку самої перевірки інкапсульовано в сервісі RbacService. Метод hasPermission отримує назви ролей користувача в конкретному місті та перевіряє наявність зв'язку між цими ролями та потрібним дозволом у глобальній таблиці відповідності ролей і дозволів.

Лістинг методу hasPermission:

```
async hasPermission(
```

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				72
Змн.	Арк.	№ докум.	Підпис	Дата		

```

userId: string,
cityId: string,
permissionKey: string,
): Promise<boolean> {
  const roleNames = await this.getUserRoleNames(userId, cityId);
  if (roleNames.length === 0) return false;
  const match = await this.prisma.rolePermission.findFirst({
    where: {
      roleName: { in: roleNames }, permission: { key: permissionKey },
    }
  }); return match != null; }

```

Узагальнений процес авторизації запиту зображено на рисунку 2.10. Після надходження запит послідовно проходить перевірку автентифікації, під час якої перевіряється коректність токена доступу. Далі виконується перевірка прав доступу шляхом зіставлення дозволів, необхідних для виконання операції, із фактичними правами користувача в межах міста. У разі успішного проходження всіх перевірок керування передається відповідному методу контролера, інакше система повертає повідомлення про відсутність доступу.

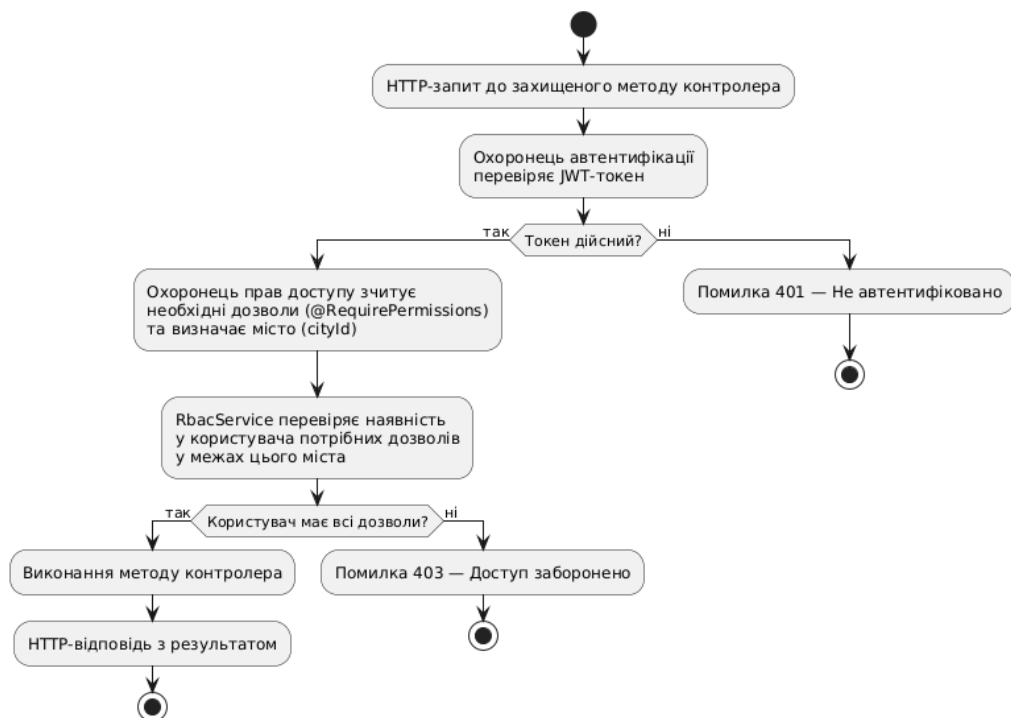


Рис. 2.10. Процес авторизації запиту

Нижче наведено приклад застосування декоратора над методом створення опитування. Виконати цей метод можуть лише користувачі, які мають у відповідному місті дозвіл на створення опитувань. За відсутності такого дозволу охоронець забороняє виконання методу та повертає помилку доступу.

```
@Post()
@UseGuards(PermissionsGuard)
@RequirePermissions(PERMISSIONS_KEYS.SURVEY_CREATE)
createSurvey(
  @Param('cityId') cityId: string,
  @Req() req: RequestWithUser,
  @Body() dto: CreateSurveyDto,
) {
  return this.surveysService.createSurvey(cityId, req.user.id, dto);
}
```

Розглянемо процес публікації термінового сповіщення як приклад операції, що змінює стан системи та потребує подальшого інформування користувачів. Перед створенням сповіщення система перевіряє, чи належить користувач до відповідного міста, а також чи існує вказаний тип сповіщення. Після успішного проходження перевірок у межах однієї транзакції виконується збереження самого сповіщення та створення запису у таблиці вихідних подій (Outbox) [28]. Текстові поля нормалізуються через `.trim()`, а дата завершення перетворюється на об'єкт `Date` або зберігається як `null`. Відповідь транзакції повертає збережений запис із вибіркою полів через `alertSelect()`, що уникає зайвих даних у відповіді.

Для формування даних події використовується функція `buildAlertEventPayload`, яка забезпечує єдиний формат корисного навантаження незалежно від місця його створення. Вона приймає ідентифікатор міста та поля щойно збереженого сповіщення, генерує унікальний `eventId` та фіксує час події. Використання транзакції гарантує узгодженість даних: якщо сповіщення було успішно збережене, відповідна подія також буде записана, а в разі помилки всі

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				
Змн.	Арк.	№ докум.	Підпис	Дата		74

зміни будуть скасовано. Це дозволяє уникнути ситуацій, коли подія існує без пов'язаного сповіщення або навпаки.

Після завершення транзакції запис у таблиці вихідних подій обробляється асинхронним механізмом доставки. На його основі для кожного учасника міста створюється внутрішньосистемна нотифікація, завдяки чому користувачі своєчасно отримують інформацію про опубліковане термінове повідомлення.

```
return this.prisma.$transaction(async (tx) => {
  const alert = await tx.alert.create({
    data: {
      cityId,
      publisherId: userId,
      ...alertData
    },
  });
  await tx.domainEventOutbox.create({
    data: {
      aggregateType: 'alert',
      aggregateId: alert.id,
      eventType: DOMAIN_EVENT_TYPES.ALERT_CREATED,
      payload: buildAlertEventPayload({ cityId, ...alert }),
    },
  });
  return alert;
});
```

Клієнтський додаток розроблено на React [36] із використанням фреймворку Next.js, бібліотеки TanStack Query для роботи з API та бібліотеки Zustand для керування станом. Стан автентифікації зберігається у сховищі useAuthStore, побудованому на Zustand із проміжним шаром persist, що зберігає дані поточного користувача та токен доступу між сеансами [44].

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				
Змн.	Арк.	№ докум.	Підпис	Дата		75

Сховище містить чотири поля стану: `user` — об'єкт поточного користувача або `null`, `token` — рядок токена доступу, `isAuthenticated` — похідний булевий прапор сесії, та `isLoading` — прапор очікування під час асинхронних операцій. Для зміни стану визначено чотири дії: `setUser` встановлює користувача і токен одночасно з підтвердженням автентифікації, `setToken` оновлює лише токен, `setLoading` керує станом завантаження, а `logout` скидає всі поля до початкових значень та додатково очищає cookie з токеном доступу через допоміжну функцію `clearAccessTokenCookie`. Параметр `partialize` визначає, які саме поля зберігаються у локальному сховищі браузера: лише `user` та `token`, тоді як `isAuthenticated` та `isLoading` є похідними або тимчасовими і відновлюються програмно при старті. Сховище надає дії для входу, оновлення токена та виходу, а параметр `name` зі значенням `AUTH_STORAGE_KEY` визначає ключ запису у `localStorage`.

Лістинг сховища `useAuthStore`:

```
export const useAuthStore = create<AuthState>()(persist((set) => ({
  user: null,
  token: null,
  isAuthenticated: false,
  isLoading: false,
  setUser: (user, token) => set({ user, token, isAuthenticated: true }),
  setToken: (token) => set({ token, isAuthenticated: true }),
  logout: () => { clearAccessTokenCookie();
    set({ user: null, token: null, isAuthenticated: false });
  },
  setLoading: (isLoading) => set({ isLoading }),
}),
{
  name: AUTH_STORAGE_KEY,
  partialize: (state) => ({ user: state.user, token: state.token }),
})));
```

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				76
Змн.	Арк.	№ докум.	Підпис	Дата		

Взаємодію із серверним API інкапсульовано у клієнті ApiClient, який автоматично додає токен доступу до заголовків запиту та прозора оновлює його у разі завершення строку дії. Якщо сервер повертає помилку автентифікації, клієнт одноразово виконує запит на оновлення токена через метод refreshAccessToken.

Метод реалізує захист від паралельних оновлень: якщо оновлення вже виконується, повторні виклики отримують той самий проміжний результат refreshPromise замість того, щоб ініціювати окремі запити. Це критично в ситуації, коли кілька паралельних запитів одночасно отримують відповідь 401 — без такого механізму кожен із них намагався б оновити токен незалежно. Запит до /auth/refresh надсилається з параметром credentials: 'include', що дозволяє браузеру автоматично передати httpOnly cookie з refresh-токеном, недоступним для JavaScript. У разі успіху новий токен доступу зберігається у сховищі useAuthStore та повертається для повторення початкового запиту; у разі невдачі викликається forceLogout, який завершує сеанс і очищає стан. У реальній реалізації логіку обгортає блок try/catch для обробки мережових помилок, а finally скидає прапори isRefreshing та refreshPromise незалежно від результату, відновлюючи готовність клієнта до наступного циклу оновлення.

Лістинг методу оновлення токена в ApiClient:

```
private async refreshAccessToken(): Promise<string | null> {  
  if (this.isRefreshing && this.refreshPromise) return this.refreshPromise;  
  this.isRefreshing = true;  
  this.refreshPromise = (async () => {  
    const res = await fetch(`${this.baseUrl}/auth/refresh`, {  
      method: 'POST', credentials: 'include', });  
    if (!res.ok) { await this.forceLogout(); return null; }  
    const { accessToken } = await res.json();  
    useAuthStore.getState().setToken(accessToken);  
    return accessToken;  
  })();
```

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				77
Змн.	Арк.	№ докум.	Підпис	Дата		

```
return this.refreshPromise;
}
```

Отримання, кешування та оновлення даних із сервера реалізовано засобами TanStack Query [34], налаштованими у провайдері QueryProvider. Запити інкапсульовано у спеціалізованих хуках, які приховують деталі звернення до API та керують станами завантаження й помилок. Наприклад, для посторінкового завантаження опитувань міста застосовано хук useInfiniteCitySurveys на основі нескінченного запиту з курсорною пагінацією.

Лістинг хука useInfiniteCitySurveys:

```
return useInfiniteQuery({
  queryKey: queryKeys.surveys.list(cityId, baseQuery),
  queryFn: ({ pageParam }) =>
    citySurveysApi.getCitySurveys(cityId, { ...baseQuery, cursor: pageParam }),
  initialPageParam: undefined as string | undefined,
  getNextPageParam: (lastPage) => lastPage.nextCursor ?? undefined,
  enabled: enabled && !!cityId,
  placeholderData: (previousData) => previousData,
});
```

Розмежування доступу продубльовано й на клієнтській частині для керування відображенням елементів інтерфейсу. Хук usePermission перевіряє наявність у поточного користувача конкретного дозволу в межах міста й повертає ознаку can, на основі якої елементи інтерфейсу (наприклад, кнопки створення сповіщення чи опитування) відображаються лише тим користувачам, які мають відповідні права.

Лістинг хука usePermission:

```
export const usePermission = (
  permission: PermissionKey,
  { cityId, enabled = true }: UsePermissionOptions,
) => {
  const { hasPermission, isLoading, error } = useRBAC({ cityId, enabled });
```

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				
Змн.	Арк.	№ докум.	Підпис	Дата		78

```

return {
  can: hasPermission(permission),
  isLoading,
  error,
};

```

Висновки до другого розділу

Було виконано проєктування та реалізацію платформи Urban Civic. Сформовано модель взаємодії користувачів із системою, визначено ключових акторів та їх функціональні можливості. Для відображення структури майбутнього рішення побудовано діаграму варіантів використання, функціональну карту вебдодатка та набір діаграм класів, що охоплюють основні предметні області платформи.

У межах проєктування сховища даних обрано систему керування базами даних PostgreSQL та розроблено її структуру засобами Prisma ORM. Модель даних охоплює 26 таблиць і набір допоміжних типів-перерахунків. Під час проєктування враховано підтримку багатоорєдності, механізм «м'якого» видалення записів, використання формату jsonb для зберігання гнучких структур даних та застосування таблиці вихідних подій для забезпечення надійної доставки сповіщень. Логічну структуру сховища представлено у вигляді ER-діаграми.

Окрему увагу приділено архітектурним рішенням та алгоритмам роботи системи. Визначено підхід до обробки запитів, розмежування прав доступу, автентифікації користувачів, взаємодії з базою даних і формування подій. Для реалізації окремих компонентів використано низку архітектурних патернів і патернів проєктування, зокрема Transactional Outbox для доставки сповіщень, Producer–Consumer для роботи з чергами завдань та Observer для організації взаємодії в реальному часі. Ключові бізнес-процеси відображено за допомогою діаграм діяльності.

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				
Змн.	Арк.	№ докум.	Підпис	Дата		79

Результатом виконаної роботи стала реалізація серверної та клієнтської частин платформи відповідно до спроектованої архітектури. На серверній стороні реалізовано модулі предметних областей, механізми автентифікації та авторизації, єдиний рівень доступу до даних і систему доставки сповіщень. Клієнтська частина забезпечує керування станом користувача, отримання та кешування даних, а також динамічне розмежування доступу до функцій інтерфейсу відповідно до ролей користувачів.

Отримані результати сформували завершену архітектурну та програмну основу платформи Urban Civic і створили підґрунтя для подальшого опису інтерфейсу, процесу розгортання та результатів тестування.

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				
Змн.	Арк.	№ докум.	Підпис	Дата		80

Центральним вузлом системи є сервер застосунку, на якому розміщено серверний код системи, реалізований на фреймворку NestJS (версія 11) у середовищі Node.js (версія 20 або новіша) [9]. Він реалізований на фреймворку NestJS (версія 11) у середовищі Node.js (версія 20 або новіша) [9]. Сервер відповідає за обробку REST API та взаємодію з основними модулями системи: автентифікацією, розмежуванням доступу, заявками, сповіщеннями, новинами, опитуваннями, департаментами, учасниками, користувачами та вкладеннями. Для обміну даними в реальному часі використовується вебсокет-шлюз на основі Socket.IO. Крім основних API-запитів, сервер виконує фонові задачі. До них належать диспетчер вихідних подій, обробник черги сповіщень та автоматичне закриття опитувань. Взаємодія з базою даних здійснюється через ORM Prisma (версія 7).

Усі структуровані дані системи зберігаються на сервері бази даних PostgreSQL, розгорнутому у хмарному сервісі Supabase. Доступ до бази даних здійснюється через ORM Prisma по TCP-з'єднанню за протоколом PostgreSQL. Для асинхронного оброблення та надійної доставки сповіщень використовується сервер Redis, на основі якого працює черга завдань BullMQ [27]. Медіафайли, зокрема фото- та відеодокази заявок і вкладення, зберігаються в об'єктному сховищі Cloudflare R2 [33], сумісному з протоколом S3. Це дає змогу не перевантажувати основну базу даних файлами великого розміру.

Сервер застосунку взаємодіє з низкою зовнішніх сервісів. Сервіс Google (OAuth 2.0) забезпечує автентифікацію користувачів через обліковий запис Google [39], сервіс Resend відповідає за надсилання сповіщень на електронну поштову скриньку, GeoNames використовується для пошуку населених пунктів та геоданих, а сервер картографічних тайлів на основі OpenStreetMap – для відображення інтерактивної карти заявок.

Взаємодія між вузлами здійснюється захищеними каналами зв'язку: обмін даними між клієнтом і сервером відбувається за протоколом HTTPS, а двосторонній зв'язок у реальному часі – через захищене вебсокет-з'єднання. Такий розподіл системи на окремі вузли забезпечує чіткий поділ

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				
Змн.	Арк.	№ докум.	Підпис	Дата		82

відповідальності, спрощує масштабування та підвищує надійність, оскільки тимчасовий збій окремого сховища чи зовнішнього сервісу не призводить до повної відмови платформи.

Перед запуском серверного додатку необхідно підготувати конфігураційний файл .env, опис основних змінних середовища якого наведено в таблиці 3.1.

Таблиця 3.1

Опис змінних середовища серверного додатку (.env)

Змінна середовища	Опис
NODE_ENV	Середовище виконання (development / production)
PORT	Порт, на якому працює серверний API
CORS_ORIGIN	Дозволене джерело запитів клієнтського додатку
CLIENT_URL	Базова URL-адреса клієнтського додатку
DATABASE_URL	Рядок з'єднання з базою даних PostgreSQL (Supabase)
DATABASE_POOL_MAX	Максимальна кількість з'єднань у пулі
DATABASE_CONNECTION_TIMEOUT_MS	Таймаут встановлення з'єднання з БД, мс
DATABASE_QUERY_TIMEOUT_MS	Таймаут виконання запиту, мс
DATABASE_IDLE_TIMEOUT_MS	Час простою з'єднання до закриття, мс
JWT_SECRET	Секретний ключ для підпису токена доступу
JWT_EXPIRES_IN	Термін дії токена доступу
JWT_REFRESH_SECRET	Секретний ключ для підпису токена оновлення
JWT_REFRESH_EXPIRES_IN	Термін дії токена оновлення
GOOGLE_CLIENT_ID	Ідентифікатор клієнта Google OAuth

GOOGLE_CLIENT_SECRET	Секретний ключ Google OAuth
GOOGLE_CALLBACK_URL	URL зворотного виклику після входу через Google
REDIS_URL	Рядок з'єднання з Redis (або REDIS_HOST / REDIS_PORT)
CLOUDFLARE_R2_ACCOUNT_ID	Ідентифікатор акаунта Cloudflare R2
CLOUDFLARE_R2_ACCESS_KEY_ID	Ключ доступу до сховища R2
CLOUDFLARE_R2_SECRET_ACCESS_KEY	Секретний ключ доступу до сховища R2
CLOUDFLARE_R2_BUCKET_NAME	Назва бакета об'єктного сховища
CLOUDFLARE_R2_ENDPOINT	Кінцева точка (endpoint) сховища R2
CLOUDFLARE_R2_PUBLIC_URL	Публічна URL-адреса для доступу до файлів
RESEND_API_KEY	Ключ API сервісу електронної пошти Resend
EMAIL_FROM	Адреса відправника електронних листів
TLS_ENABLED	Увімкнення HTTPS на сервері
TLS_CERT_PATH	Шлях до файлу сертифіката TLS
TLS_KEY_PATH	Шлях до файлу приватного ключа TLS
NOTIFICATIONS_WORKER_ENABLED	Увімкнення фонового обробника доставки сповіщень
NOTIFICATIONS_RELAY_INTERVAL_MS	Інтервал опитування таблиці вихідних подій, мс
NOTIFICATIONS_RELAY_BATCH_SIZE	Розмір партії подій за один цикл оброблення
SURVEY_SWEEP_ENABLED	Увімкнення фонового автоматичного закриття опитувань
SURVEY_SWEEP_INTERVAL_MS	Інтервал перевірки опитувань для закриття, мс
SURVEY_SWEEP_BATCH_SIZE	Розмір партії опитувань за один цикл

Для налаштування клієнтського додатку у файлі .env достатньо вказати змінні середовища, опис яких наведено в таблиці 3.2.

Таблиця 3.2

Опис змінних середовища клієнтського додатку (.env)

Змінна середовища	Опис
NEXT_PUBLIC_API_URL	Базова URL-адреса серверного API
NEXT_PUBLIC_MAP_TILE_URL	URL-адреса сервера картографічних тайлів
NEXT_PUBLIC_GEONAMES_USERNAME	Ім'я користувача сервісу GeoNames для пошуку населених пунктів

3.2. Структура інтерфейсу вебдодатку та порядок роботи

Користувач, відкривши вебдодаток, потрапляє на головну сторінку, де йому пропонується обрати місто зі списку доступних або скористатися пошуком міст за назвою (рис. 3.2). Платформа є багатоорендною, тож уся подальша взаємодія відбувається в межах обраного міста. У вебзастосунку передбачено підтримку української та англійської мов інтерфейсу, за замовчуванням обирається українська мова. Мінімалістичний інтерфейс побудовано на основі утилітарного CSS-фреймворку Tailwind CSS, що забезпечує єдиний стиль і швидку адаптацію під різні розміри екранів [20].

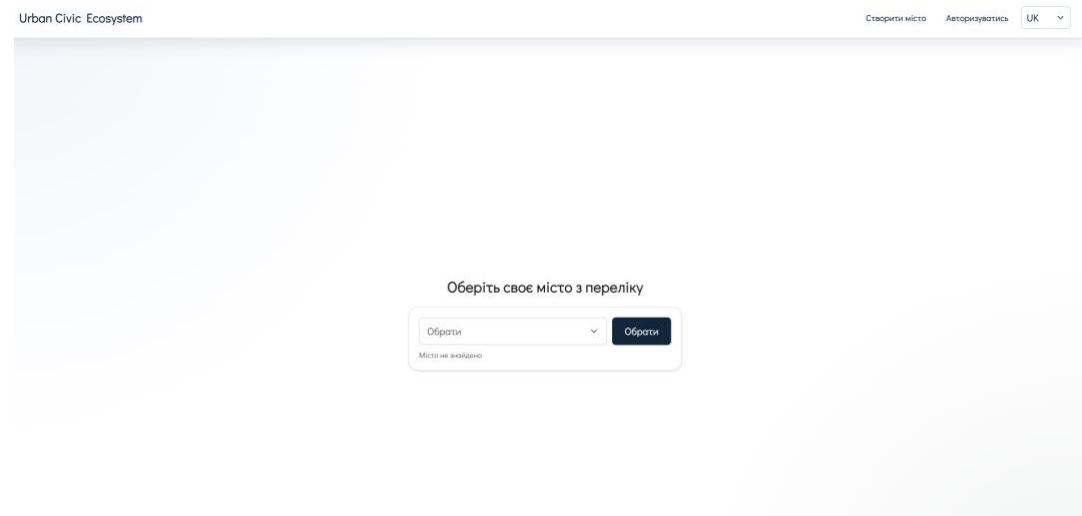


Рис. 3.2. Головна сторінка з вибором міста

Для доступу до більшості функцій користувач має увійти в систему. Натиснувши кнопку входу у верхньому правому куті панелі навігації, він переходить на сторінку авторизації (рис. 3.3), де може ввести електронну пошту та пароль або скористатися входом через обліковий запис Google. Передбачено також реєстрацію нового облікового запису. Після успішної автентифікації токени доступу й оновлення зберігаються, а сесія автоматично подовжується без повторного введення пароля.

Рис. 3.3. Сторінка авторизації

Після входу користувач отримує доступ до сторінки профілю (рис. 3.4), де може переглянути та змінити власні дані, а також змінити пароль. На сторінці відображаються основні відомості про обліковий запис користувача, зокрема адреса електронної пошти та дата реєстрації.

У разі спроби відкрити сторінку без необхідних прав доступу користувач перенаправляється на сторінку з повідомленням про заборону доступу. Для заблокованих учасників передбачено окреме повідомлення, яке інформує про обмеження доступу до функціональних можливостей платформи.

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				
Змн.	Арк.	№ докум.	Підпис	Дата		86

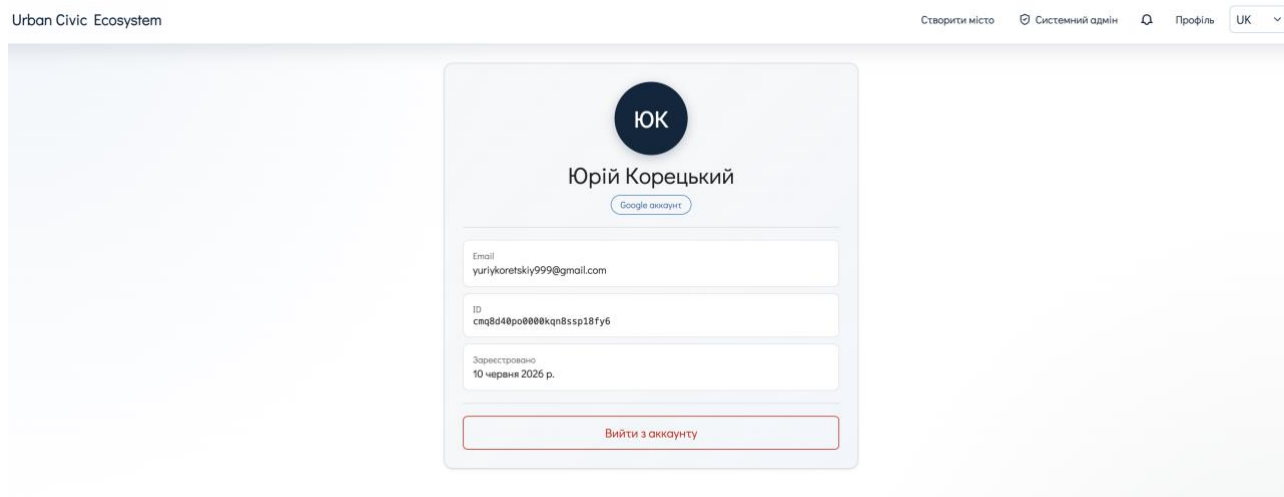


Рис. 3.4. Профіль користувача

Щоб приєднатися до спільноти міста, користувач відкриває його сторінку та натискає кнопку приєднання, після чого отримує роль жителя та доступ до функцій міста. Якщо потрібного міста ще немає на платформі, користувач може подати заявку на його створення на сторінці створення міста (рис. 3.5): він зазначає назву, регіон, розташування на карті та офіційний домен муніципалітету, після чого проходить процедуру підтвердження права власності на домен. Стан поданої заявки (на розгляді, схвалено, відхилено) відображається у блоці відстеження.

Рис. 3.5. Сторінка створення міста та відстеження заявки

Після приєднання до міста користувач потрапляє на його головну сторінку (рис. 3.6), яка є інформаційною панеллю та містить стислі добірки останніх термінових оголошень, новин і звернень, оглядову карту звернень

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				
Змн.	Арк.	№ докум.	Підпис	Дата		87

міста, а також панель, що відповідає ролі користувача. Ліворуч розміщено навігаційну панель для переходу до розділів звернень, оголошень, новин та опитувань.

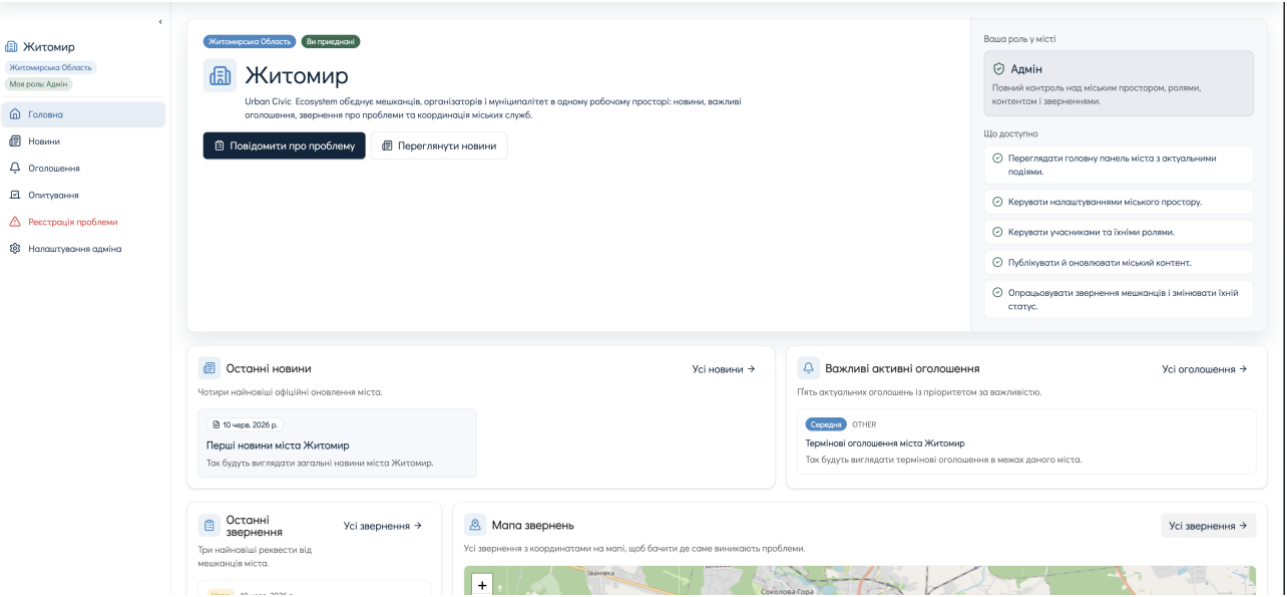


Рис. 3.6. Головна сторінка міста

Розділ звернень (рис. 3.7) відображає перелік звернень міста та їх розташування на інтерактивній карті [7]. Користувач може фільтрувати звернення за статусом, категорією та департаментом. Для подання нового звернення житель заповнює форму із зазначенням заголовка, опису, місця на карті (геоприв’язка) та додає фото- чи відеодокази (рис. 3.8).

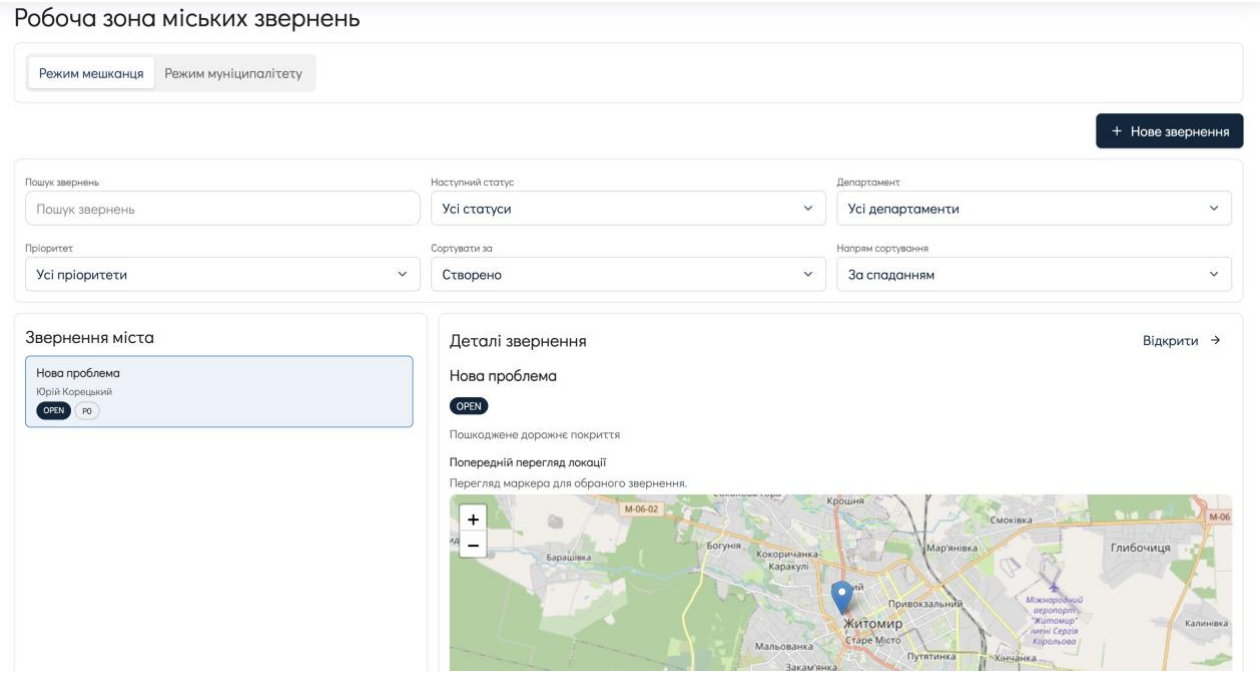


Рис. 3.7. Сторінка міських звернень

Робоча зона міських звернень

Режим мешканця

Режим муніципалітету

X Сховати форму

Створити звернення про проблему

Заголовок

Опис

50.262350

28.679130

Локація проблеми

Клікніть по карті, щоб поставити маркер і автоматично заповнити координати.

+

-

Натисніть для завантаження файлів

Максимум 1 файл

Створити звернення

Рис. 3.8. Форма створення звернення

На детальній сторінці звернення відображається повна інформація про нього: опис, медіадокази, історія зміни статусів та чат для спілкування заявника з муніципалітетом у реальному часі. Для жителя сторінка має вигляд, наведений на рис. 3.9. Представник муніципалітету додатково має панель керування (рис. 3.10), що дозволяє призначити звернення на відповідальний департамент, змінити його статус (нове → у роботі → виконано / відхилено) та додати звіт про виконані роботи. Про кожну зміну заявник отримує сповіщення.

В роботі

РР

Повідіть та прокоментуйте

12 червня 2026 р. о 12:27

Демонстраційне звернення

В центрі міста помітив наступну проблему, яку прошу якомога швидше усунути: ...

Хід розгляду звернення

PROGRESS

Вже в роботі

Юрій Корецький

12 червня 2026 р. о 12:28

Чат мешканця і муніципалітету

Юрій Корецький

Вітаю! Коли буде взято в роботу дане питання?

Написати повідомлення...

Надіслати

Попередній перегляд локації

Перегляд маркера для обраного звернення:

Вкладення звернення

Рис. 3.9. Сторінка звернення (відображення мешканця)

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				89
Змн.	Арк.	№ докум.	Підпис	Дата		

В роботі

РД

Поліція та громадська безпека

12 червня 2026 р. о 12:27

Демонстраційне звернення

В центрі міста помітив наступну проблему, яку прошу якнайскоріше усунути...

Попередній перегляд локації

Перегляд маркера для обраного звернення

Вкладення звернення

Інструменти муніципалітету

Поліція та громадська безпека

В роботі

Прогрес

Текст звіту

Натисніть для завантаження файлів

Максимум 1 файл

Створити звіт

Чат мешканця і муніципалітету

Юрій Корещак

Вітаю! Коли буде взятو в роботу дане питання?

Напишіть повідомлення...

Надіслати

Рис. 3.10. сторінка звернення (відображення муніципалітету)

У розділі оголошень відображається перелік термінових оголошень міста із зазначенням рівня критичності та строку дії. Для жителя розділ має вигляд, наведений на рис. 3.11, а представник муніципалітету користується сторінкою керування оголошеннями (рис. 3.12), де може знайти, опублікувати, відредагувати чи видалити оголошення.

Оголошення

Дощка оголошень

Режим мешканця

Режим керування

Пошук за заголовком або текстом

Усі рівні

ДТП на Бердичівській

ДТП / дорожній інцидент

Висока - Актуально до: 12.06.2026, 18:32:00

Радимо рухатись альтернативними дорожніми шляхами

12 червня 2026 р. - 12:33

Проблеми з газопостачанням на Київській продовж дня

Проблеми з газопостачанням

Висока - Актуально до: 13.06.2026, 12:31:00

Буде вирішення протягом доби з моменту оголошення

12 червня 2026 р. - 12:31

Термінові оголошення міста Житомир

Інше

Середня - Актуально до: Безстроково

Так будуть виглядати термінові оголошення в межах даного міста.

10 червня 2026 р. - 20:50

Рис. 3.11. Сторінка оголошень (відображення мешканця)

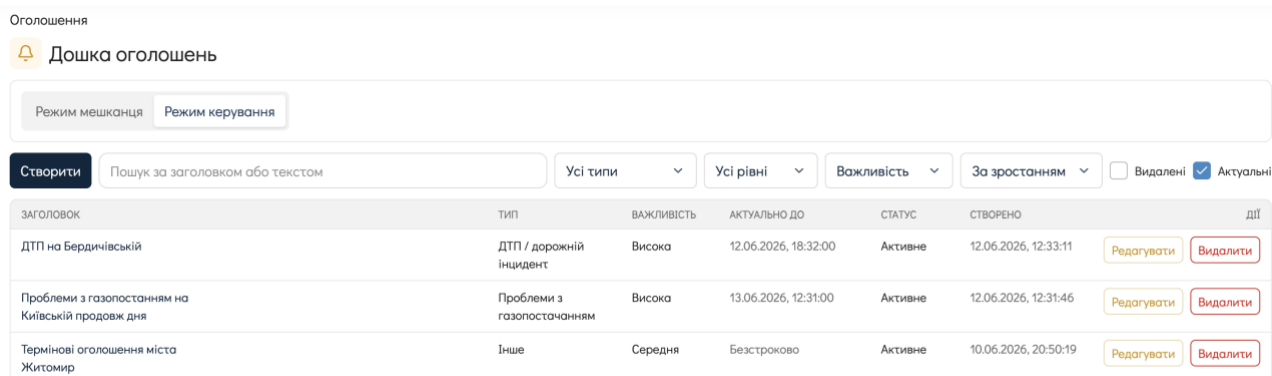


Рис. 3.12. Керування оголошеннями (відображення муніципалітету)

Перехід до окремого оголошення відкриває його детальну сторінку, наведену на рис. 3.13. На ній відображається повний вміст оголошення – заголовок, рівень важливості, текст, строк дії та дата публікації, – поданий у зручному для читання вигляді без обмежень за обсягом, характерних для картки в загальному переліку. Звідси користувач може повернутися до дошки оголошень, а представник муніципалітету – перейти до редагування чи видалення запису залежно від наданих йому дозволів.

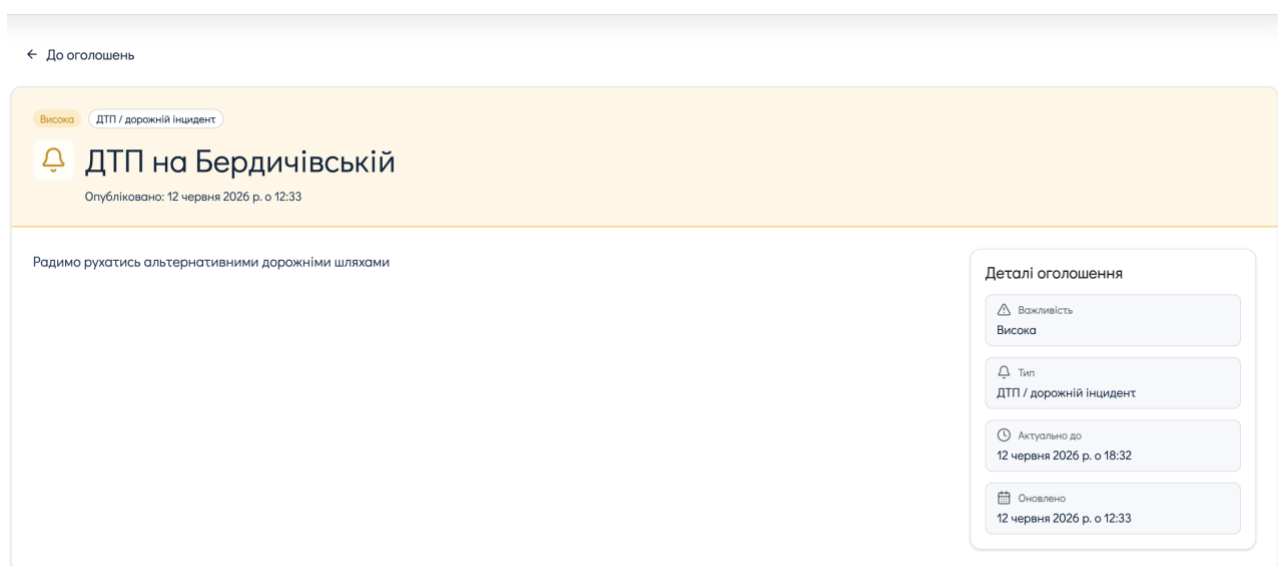


Рис. 3.13. Сторінка деталей оголошення

Аналогічно побудовано розділ новин, у якому відображаються офіційні новини й оголошення муніципалітету з можливістю додавання вкладень. Для жителя розділ має вигляд, наведений на рис. 3.14, муніципалітет користується сторінкою керування новинами (рис. 3.15).

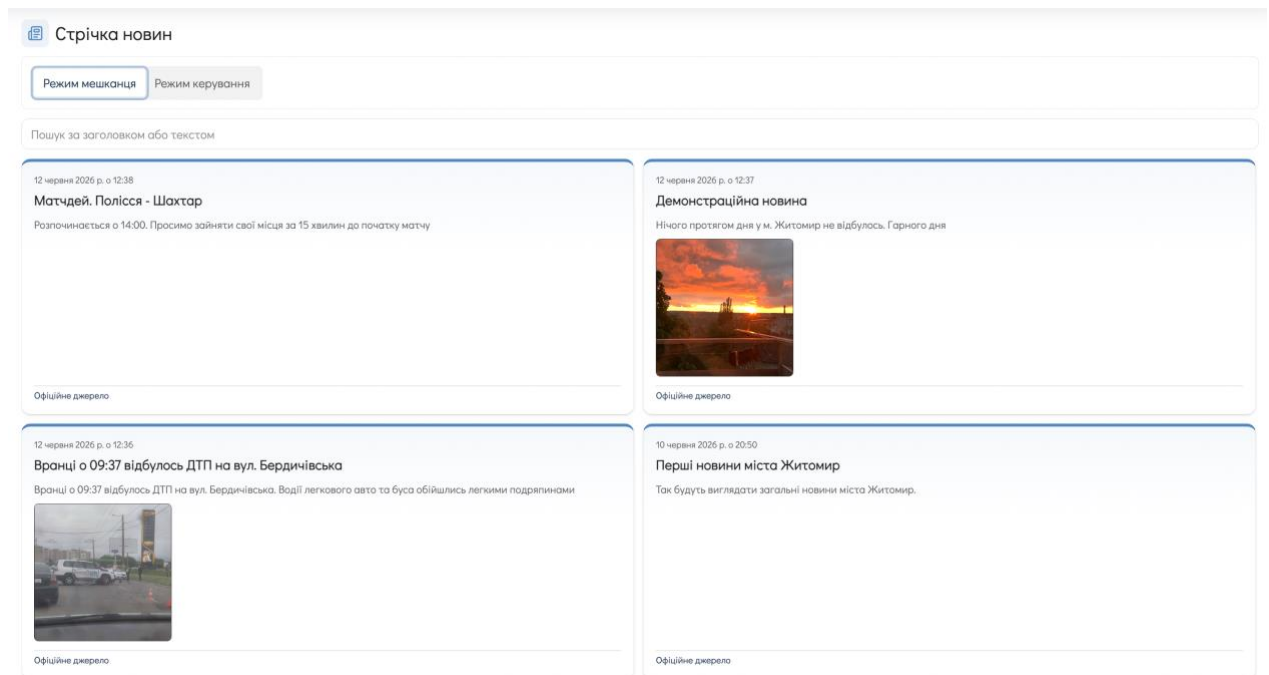


Рис. 3.14. Сторінка новин (відображення мешканця)

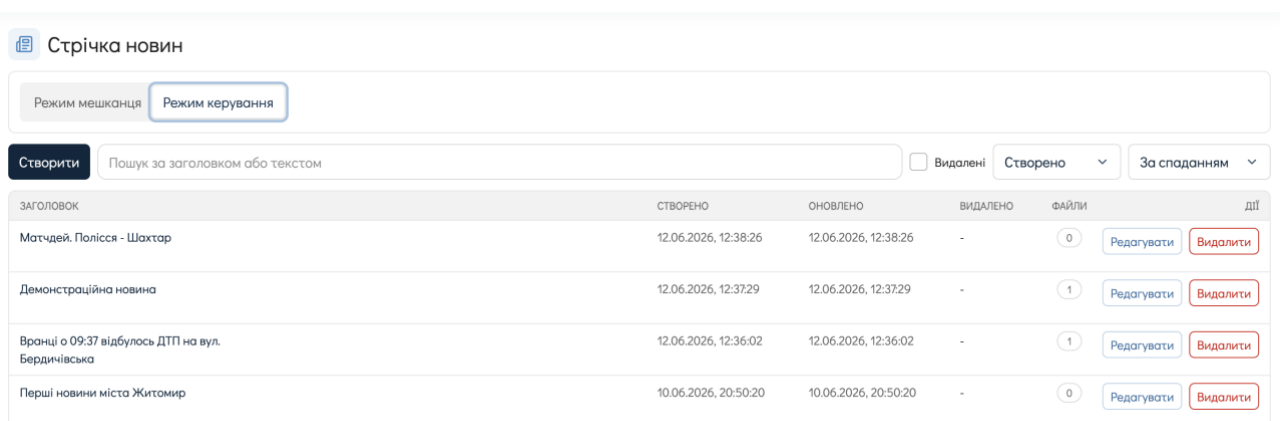


Рис. 3.15. Сторінка керування новинами (відображення муніципалітету)

Після вибору окремої новини користувач переходить на сторінку її детального перегляду. На цій сторінці відображаються повний текст публікації, дата створення та додані вкладення, що забезпечує зручний доступ до всієї пов'язаної інформації (рис. 3.16).

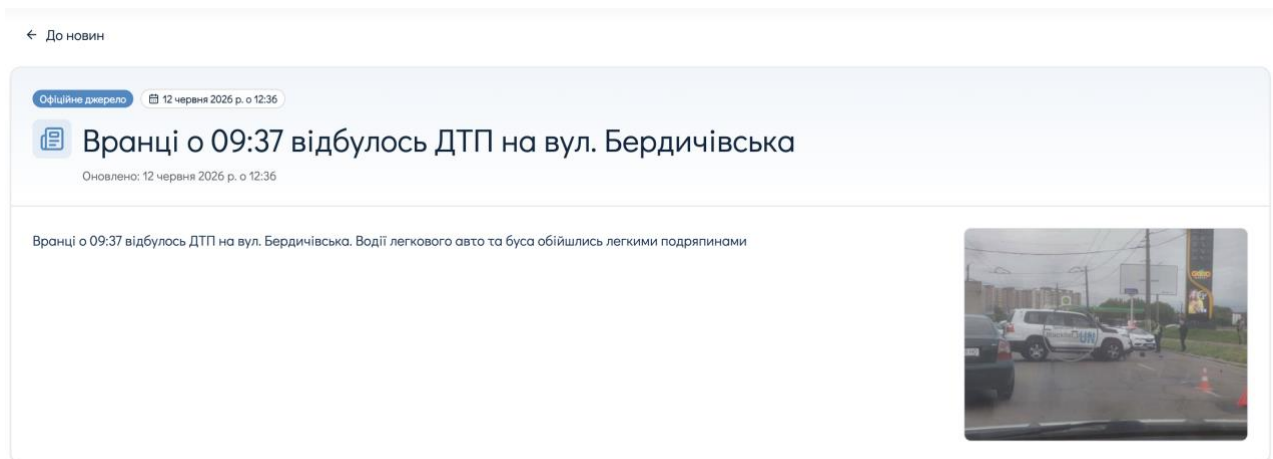


Рис. 3.16. Сторінка деталей новини

Розділ опитувань містить перелік опитувань міста. Для жителя він має вигляд, наведений на рис. 3.17, тоді як представник муніципалітету користується сторінкою керування опитуваннями (рис. 3.18), де має доступ до адміністративних дій, зокрема завершення поточного опитування раніше планового строку та редагування для випадків, якщо деталі були внесені до форми створення неправильно.

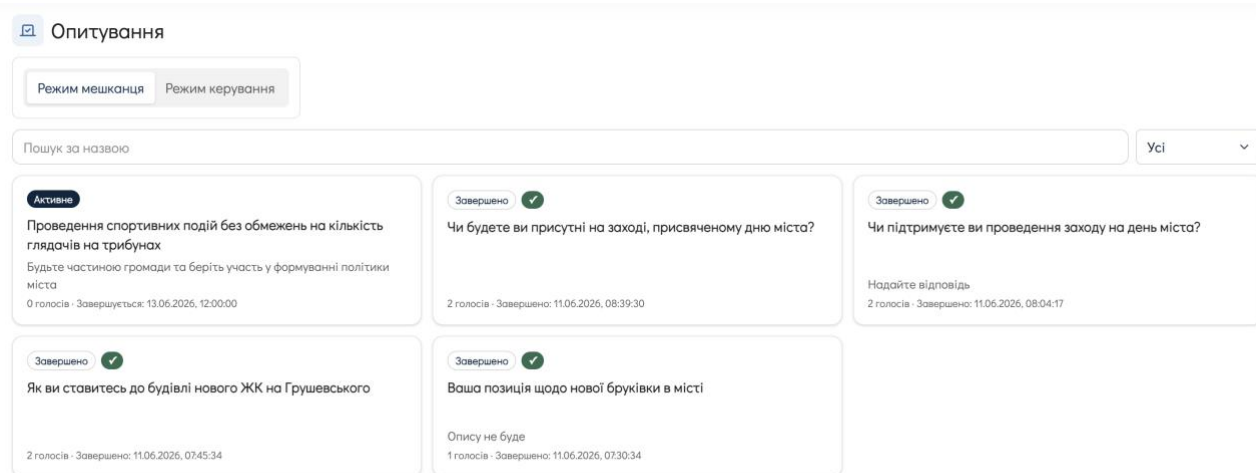


Рис. 3.17. Сторінка опитувань (відображення мешканця)

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				93
Змн.	Арк.	№ докум.	Підпис	Дата		

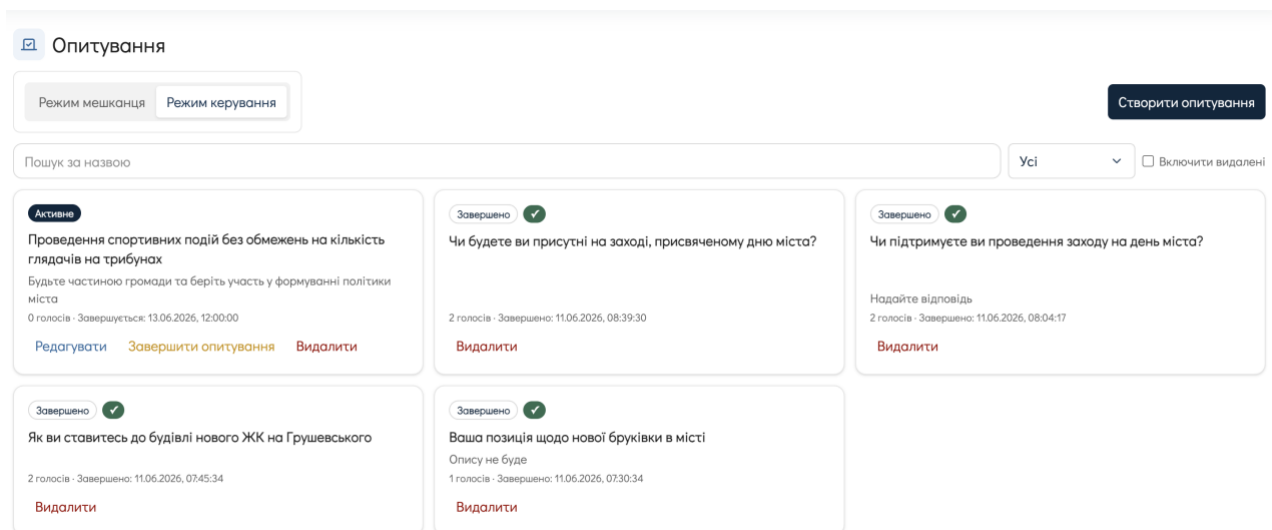


Рис. 3.18. Керування опитуваннями (відображення муніципалітету)

Створення нового опитування виконується через окрему форму, наведену на рис. 3.19. У ній представник муніципалітету заповнює поле запитання та перелік варіантів відповіді, а також налаштовує строк завершення опитування.

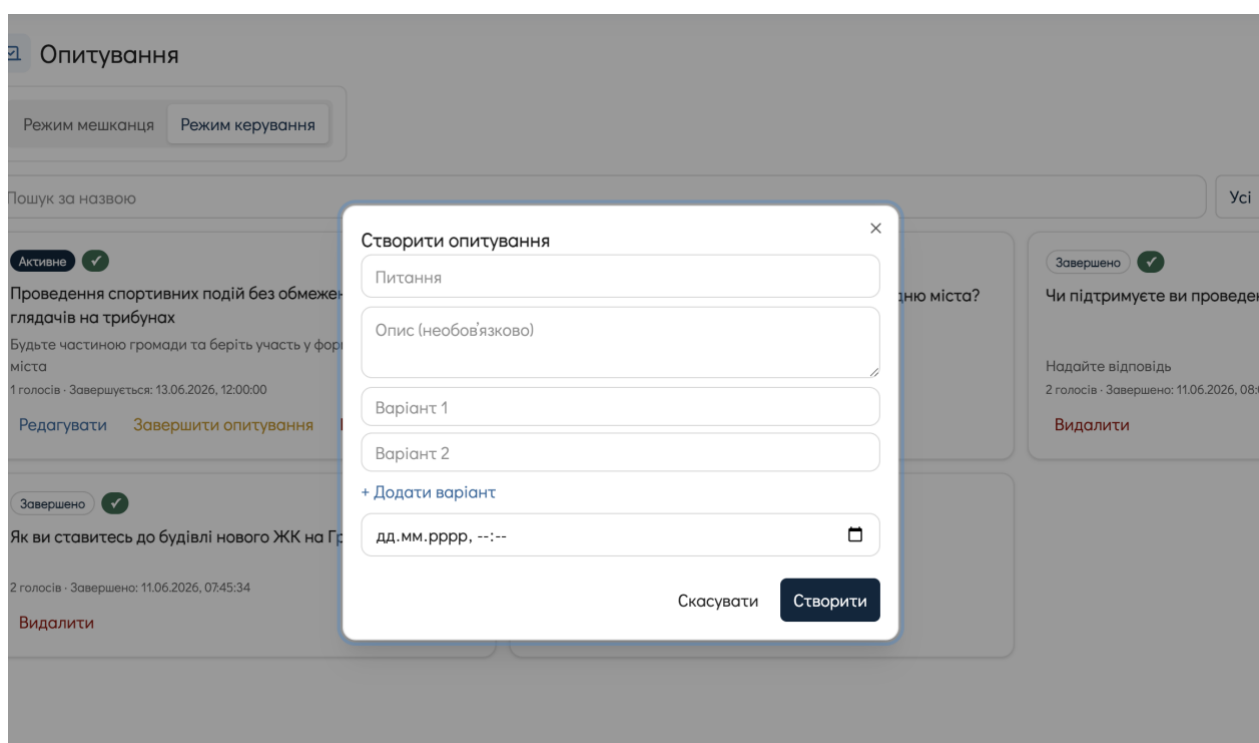


Рис. 3.19. Форма створення опитування (відображення муніципалітету)

Опубліковане опитування стає доступним жителям для голосування. Відкривши його, користувач обирає один із варіантів відповіді (рис. 3.20), після чого його голос враховується в загальному підрахунку. Підсумкові результати відображаються одразу після завершення голосування (рис. 3.21).

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				94
Змн.	Арк.	№ докум.	Підпис	Дата		

Назад до опитувань

Активно

Проголосувати

Проведення спортивних подій без обмежень на кількість глядачів на трибунах

Будьте частиною громади та беріть участь у формуванні політики міста

Позитивно

Негативно

Результати ще недоступні.

Результати

Усього голосів

0

Видимість результатів

Після голосування

Завершується

13.06.2026, 12:00:00

Рис. 3.20. Сторінка участі в опитуванні

Назад до опитувань

Завершено

Проголосувати

Проведення спортивних подій без обмежень на кількість глядачів на трибунах

Будьте частиною громади та беріть участь у формуванні політики міста

Видалити

Позитивно

Негативно

50% (1)

50% (1)

Результати

Усього голосів

2

Видимість результатів

Після голосування

Завершено

12.06.2026, 13:01:16

Рис. 3.21. Результати завершеного опитування

Про важливі події в місті, такі як нове оголошення , зміна статусу звернення, створення або закриття опитування, користувач дізнається завдяки централізованим системним сповіщенням, які накопичуються у відповідній панелі та супроводжуються листами на електронну пошту (рис. 3.22). Сповіщення доставляються персоналізовано: користувач отримує лише ті повідомлення, що стосуються його ролі, обраного міста та подій, на які він підписаний, тож інформаційний потік залишається релевантним і не перевантажує його зайвими даними. У панелі сповіщень кожен запис позначається як прочитаний або непрочитаний, а лічильник непрочитаних повідомлень дає змогу швидко оцінити, чи з'явилися нові події з моменту останнього перегляду. Кожне сповіщення містить посилання на пов'язаний об'єкт, тому з панелі користувач одразу переходить до конкретної заявки, оголошення чи опитування. Надсилення листів на електронну пошту дублює найважливіші повідомлення поза межами застосунку, що дає змогу не пропустити критичні події навіть тоді, коли користувач не перебуває в системі.

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				95
Змн.	Арк.	№ докум.	Підпис	Дата		

Сповіщення

Позначити все прочитаним

Додано новину

Позначити

Вранці о 09:37 відбулось ДТП на вул.
Бердичівська

прочитаним

Житомир, Житомирська Область

12.06.2026, 12:36:04

Додано оголошення

Позначити

ДТП на Бердичівській

прочитаним

Житомир, Житомирська Область

Severity: HIGH

12.06.2026, 12:33:14

Додано оголошення

Позначити

Проблеми з газопостанням на
Київській продовж дня

прочитаним

Житомир, Житомирська Область

Severity: HIGH

Рис. 3.22. Централізовані системні сповіщення про події

Користувачам із відповідними правами доступна сторінка налаштувань міста, на якій адміністратор керує міським середовищем. Перша вкладка налаштувань містить довідкову інформацію про місто (рис. 3.23). Вкладка департаментів дозволяє створювати, редагувати та видаляти зареєстровані департаменти (рис. 3.24).

Налаштування адміна

Керування міським середовищем, ролями, департаментами та публічними даними міста.

Профіль міста | Департаменти | Учасники

Профіль міста

Загальна інформація про місто та документ-доказ, використаний для реєстрації муніципального простору.

Ім'я	Київ
Регіон	Київська область
Домен муніципалітету	Не вказано
Створено	12.06.2026, 12:54:01
Оновлено	12.06.2026, 12:54:01

Документ-доказ муніципалітету

Офіційний документ, завантажений під час створення міста, щоб підтвердити, що цей простір представляє відповідний муніципалітет.



Рис. 3.23. Керування міським середовищем через налаштування адміністратора — довідкова інформація

Налаштування адміна

Керування міським середовищем, ролями, департаментами та публічними даними міста.

Профіль міста | Департаменти | Учасники

Департаменти

Дефолтні департаменти заблоковані. Кастомні департаменти можна додавати для цього міста і використовувати в призначенні звернень.

Назва департаменту	Опис	Додати департамент		
ІМ'Я	ТИП	ОПИС	ПОХОДЖЕННЯ	ДІЯ
Дорожня служба	ROAD_SERVICE	Ями, розмітка, освітлення доріг, зупинки та інфраструктура.	Дефолтний	Заблоковано
Екстрені та спецслужби	EMERGENCY_SPECIAL	Надзвичайні ситуації та координація зі спецслужбами.	Дефолтний	Заблоковано
ЖКГ та комунальні служби	UTILITIES	Вода, тепло, електрика, вивіз сміття, прибудинкові мережі.	Дефолтний	Заблоковано
Інше	OTHER	Інші звернення, що потребують ручної маршрутизації.	Дефолтний	Заблоковано
Пожарна служба	FIRE_SERVICE	Реагування на пожежі, задимлення та інші пожежні загрози.	Дефолтний	Заблоковано
Поліція та громадська безпека	PUBLIC_SAFETY	Порушення громадського порядку та питання безпеки.	Дефолтний	Заблоковано
Соц служба	CUSTOM	Виконання соціально-чутливих завдань	Кастомний	Редагувати Видалити

Рис. 3.24. Керування зареєстрованими департаментами через налаштування міста

Вкладка учасників надає інструменти для керування приєднаними жителями, зокрема дозволяє змінювати їхні ролі та керувати блокуванням у разі необхідності (рис. 3.25).

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				
Змн.	Арк.	№ докум.	Підпис	Дата		97

Налаштування адміна

Керування міським середовищем, ролями, департаментами та публічними даними міста.

Профіль міста

Департаменти

Учасники

Учасники міста та ролі

4 учасників

Пошук учасників

Усі ролі

Приєднався

ASC

ІМ'Я	EMAIL	Роль	СТАТУС	ПРИЄДНАВСЯ	ДІЇ
Юрій Корецький	yuriykoretsky999@gmail.com	Адмін	Активний	12.06.2026	Зберегти Заблокувати
Юрій Корецький	ipz221_kyui@student.zhu.edu.ua	Мешканець	Активний	12.06.2026	Зберегти Заблокувати
Mykola Andriyenko	andriyenko@gmail.com	Мешканець	Активний	12.06.2026	Зберегти Заблокувати
Олена Коваленко	kovalenko@gmail.com	Мешканець	Активний	12.06.2026	Зберегти Заблокувати

Адмін

Муніципалітет

Мешканець

Рис. 3.25. Керування приєднаними жителями через налаштування міста

Адміністратору платформи доступна окрема сторінка адміністрування, що надає централізований доступ до керування ключовими аспектами системи. В панелі адміністрування він переглядає заявки на створення нових міських середовищ у вигляді впорядкованого списку із зазначенням назви міста, дати подання та поточного статусу кожної заявки (рис. 3.26). Обравши конкретну заявку, адміністратор отримує доступ до її повного змісту: опису міського середовища, контактних даних заявника та додаткової інформації, наданої під час подання (рис. 3.27). На підставі цих відомостей адміністратор приймає рішення – схвалює заявку, ініціюючи створення нового міського середовища, або відхиляє її із зазначенням причини, яка автоматично надсилається заявнику. Такий механізм модерації забезпечує контрольоване розширення платформи та виключає появу міських середовищ без попередньої перевірки.

Системне адміністрування

Перевірка заявок на міста, керування містами та системними ролями користувачів.

Заявки

Міста

Користувачі

Заявки на створення міст

1 записів

Пошук

Очікує

МІСТО	РЕГІОН	ЗАЯВНИК	СТАТУС	СТВОРЕНО	ДІЇ
Київ	Київська область	Юрій Корецький yuriykoretsky999@gmail.com	Очікує	12.06.2026	Документ Детальніше

1-1 із 1

Назад Стор. 1/1 Далі

Рис. 3.26. Сторінка адміністрування платформи – заявки на створення міського середовища

Київ

Київська область

МІСТО

Київ

РЕГІОН

Київська область

ДОМЕН

-

DNS TXT ПІДТВЕРДЖЕНО

-

КООРДИНАТИ

50.45466, 30.5238

ЗАЯВНИК

Юрій Корецький <yuriykoretskiy999@gmail.com>

СТВОРЕНО

12.06.2026, 12:53

ПЕРЕВІРИВ

-

ПЕРЕВІРЕНО

-

Атачмент заявки

spravka_pro_registratsiyu.jpeg

Відкрити файл

ПРИЧИНА ВІДХИЛЕННЯ

Причина відхилення

Відхилити

Схвалити

Рис. 3.27. Сторінка адміністрування платформи – обробка заявки на створення міського середовища

Окремі вкладки дозволяють керувати переліком наявних міських середовищ (рис. 3.28) та обліковими записами користувачів – змінювати їхню системну роль і блокувати їх (рис. 3.29). Таблиці на цих сторінках підтримують фільтрацію, сортування та посторінкове відображення записів.

Системне адміністрування

Перевірка заявок на міста, керування містами та системними ролями користувачів.

Заявки

Міста

Користувачі

Міста платформи

4 записів

Пошук

☐ Показати видалені

МІСТО	РЕГІОН	ДОМЕН	УЧАСНИКИ	СТАТУС	ДІЇ	
Житомир	Житомирська Область		2	Активні	Зберегти	Видалити
Київ	Київська область		1	Активні	Зберегти	Видалити
Львів	Львівська Область		1	Активні	Зберегти	Видалити
Тернопіль	Тернопільська Область		1	Активні	Зберегти	Видалити

1-4 із 4

Назад

Стр. 1/1

Далі

Рис. 3.28. Сторінка адміністрування платформи – керування міськими середовищами

Системне адміністрування

Перевірка заявок на міста, керування містами та системними ролями користувачів.

Заявки

Міста

Користувачі

Користувачі платформи

8 записів

Пошук

Усі ролі

ІМ'Я	EMAIL	РОЛЬ	СТАТУС	МІСТА	ДІЇ
Олег Ольжич	olzhyya@gmail.com	Користувач	Активно	0	Зробити адмінім Заблокувати
Елеонора Рузвельт	ruzevhuo@gmail.com	Користувач	Активно	0	Зробити адмінім Заблокувати
Оксрука Надія	nadiyaye@gmail.com	Користувач	Активно	0	Зробити адмінім Заблокувати
Микола Спайдер	spiderman@gmail.com	Користувач	Активно	0	Зробити адмінім Заблокувати
Іван Федорович	ivanfedor@gmail.com	Користувач	Активно	0	Зробити адмінім Заблокувати
Oleh Petrouych	olehpetrouych@gmail.com	Користувач	Активно	0	Зробити адмінім Заблокувати
Юрій Корещий	ipx221_kyiv@student.zhu.edu.ua	Користувач	Активно	1	Зробити адмінім Заблокувати
Юрій Корещий	yurikoretsky999@gmail.com	Системний адмін	Активно	4	Зняти адміна Заблокувати

1-8 із 8

Назад

Стр. 1/1

Далі

Рис. 3.29. Сторінка адміністрування платформи – керування користувачами

3.3. Тестування роботи та використання вебсервісу

Перевірка коректності роботи програмного забезпечення є важливим етапом у процесі створення будь-якого проєкту. Ця процедура дозволяє виявити наявні недоліки, помилки та дефекти, що сприяє підвищенню загальної якості кінцевого продукту. Лише після ретельного тестування й усунення всіх виявлених проблем розробники можуть бути впевнені, що продукт готовий до впровадження. Основними цілями тестування є забезпечення стабільної, безвідмовної роботи додатку та перевірка всього функціоналу на відповідність вимогам.

Для перевірки серверної частини застосовано автоматизоване модульне тестування з використанням фреймворку Jest [24]. Тестами покрито ключову

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				100
Змн.	Арк.	№ докум.	Підпис	Дата		

бізнес-логіку сервісів та контролерів: розмежування прав доступу, оброблення заявок та чату, публікацію сповіщень і новин, опитування, керування учасниками й департаментами, створення міста та підтвердження домену. Окремо реалізовано наскрізний (e2e) тест для модуля заявок. Автоматизовані тести дозволяють швидко перекоонатися у коректності логіки після внесення змін до коду та запобігають появі регресій.

Окрім автоматизованого, було проведено ручне тестування з використанням димового (smoke) тестування, яке охоплює основну функціональність системи [31]. Це тестування здійснюється з метою перекоонатися, що базові функції програми працюють правильно, без заглиблення в деталі, та спрямоване на те, щоб у найкоротші терміни охопити якомога більше функціоналу.

Під час тестування було застосовано техніку «причина–наслідок». Ця методика передбачає визначення вхідних умов (причин) та очікуваних результатів (наслідків) для перевірки коректної роботи програми відповідно до введених даних. Дану техніку було використано для тестування автентифікації користувачів, оброблення заявок (створення, призначення департаменту, зміна статусу), а також для перевірки роботи з медіафайлами, зокрема їх завантаження та зберігання.

Крім того, під час тестування використовувалася техніка «аналіз граничних значень» для перевірки даних, які вводяться користувачами [32]. Цей метод застосовувався для тестування функцій валідації вхідних даних, таких як перевірка довжини пароля, формату електронної пошти, обов'язковості та довжини полів заявок, сповіщень і опитувань. Наприклад, опитування неможливо створити з кількістю варіантів відповіді меншою за два, а проголосувати – більш ніж один раз.

Для систематизації перевірок та документування результатів тестування було створено чекліст основних функцій платформи Urban Civic, у якому для кожного пункту зафіксовано перевірюваний функціонал та підсумковий результат. Зведені результати такого тестування наведено в таблиці 3.3.

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				101
Змн.	Арк.	№ докум.	Підпис	Дата		

Чекліст для тестування платформи Urban Civic

№	Функціонал	Результат
Автентифікація та доступ		
1	Реєстрація нового користувача	✓ Пройдено
2	Вхід за електронною поштою та паролем	✓ Пройдено
3	Вхід через обліковий запис Google	✓ Пройдено
4	Автоматичне оновлення токена доступу	✓ Пройдено
5	Розмежування доступу залежно від наявних у користувача дозволів	✓ Пройдено
Міста		
6	Пошук та перегляд переліку міст	✓ Пройдено
7	Приєднання користувача до міста	✓ Пройдено
8	Подання заявки на створення міста з підтвердженням домену	✓ Пройдено
9	Розгляд заявки на створення міста адміністратором платформи	✓ Пройдено
Заявки		
10	Створення заявки з геоприв'язкою та медіадоказами	✓ Пройдено
11	Відстеження статусу заявки	✓ Пройдено
12	Чат-комунікація заявника з муніципалітетом у реальному часі	✓ Пройдено

13	Призначення заявки на департамент	✓ Пройдено
14	Зміна статусу заявки муніципалітетом	✓ Пройдено
15	Формування звіту про виконання заявки	✓ Пройдено
Сповіщення та новини		
16	Публікація термінового сповіщення муніципалітетом	✓ Пройдено
17	Перегляд сповіщень та новин жителями	✓ Пройдено
18	Отримання внутрішньосистемних сповіщень та листів	✓ Пройдено
Опитування		
19	Створення опитування муніципалітетом	✓ Пройдено
20	Голосування жителя в опитуванні	✓ Пройдено
21	Перегляд результатів відповідно до налаштованої видимості	✓ Пройдено
22	Закриття опитування та публікація результатів	✓ Пройдено
Адміністрування міста		
23	Керування учасниками та їхніми ролями	✓ Пройдено
24	Блокування та розблокування учасника	✓ Пройдено
25	Створення, редагування та видалення департаментів	✓ Пройдено
Адміністрування платформи		

26	Керування містами	✓ Пройдено
27	Керування користувачами (змінa системної ролі, блокування)	✓ Пройдено

За результатами тестування всі перевірені функції відпрацювали коректно й відповідають визначеним вимогам, що підтверджує стабільність та готовність платформи до використання.

Висновки до третього розділу

Було виконано проєктування та документування процесу розгортання платформи Urban Civic. Побудовано діаграму розгортання, яка відображає основні компоненти системи, їх взаємодію та використані технології. Також визначено набір параметрів конфігурації, необхідних для коректної роботи серверної та клієнтської частин застосунку.

Продемонстровано структуру інтерфейсу вебдодатка та основні сценарії взаємодії користувачів із системою. Описано процес вибору міста, реєстрації та авторизації, роботу із заявками, оголошеннями, новинами, опитуваннями, сповіщеннями та адміністративними функціями. Окрему увагу приділено розмежуванню доступу відповідно до ролей користувачів, що забезпечує контроль доступу до функціональних можливостей платформи.

Для перевірки працездатності програмного забезпечення проведено комплексне тестування. Серверну частину перевірено за допомогою модульних і наскрізних тестів, реалізованих із використанням фреймворку Jest. Додатково виконано ручне тестування основних функціональних сценаріїв із застосуванням технік аналізу причинно-наслідкових зв'язків та граничних значень. Отримані результати підтвердили коректність роботи реалізованого функціоналу та його відповідність визначеним вимогам.

Результатом виконаної роботи стала працездатна платформа цифрової взаємодії жителів із муніципалітетом, яка об'єднує механізми подання та опрацювання звернень, інформування громадян, проведення опитувань і адміністрування міського середовища. Реалізоване рішення може використовуватися як основа для подальшого розвитку цифрових сервісів територіальних громад та впровадження інструментів електронної участі громадян.

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				105
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

Було розроблено багатофункціональний міський портал взаємодії жителів з муніципалітетом Urban Civic, спрямований на забезпечення прозорості, оперативності та ефективності вирішення міських проблем із використанням сучасних технологій і засобів розробки. Поставлену мету досягнуто, а всі визначені у вступі завдання виконано в повному обсязі.

Проведено аналіз функціональних потреб системи та порівняльний аналіз існуючих аналогів у сфері електронної демократії та громадянської участі – «Київ Цифровий», «Дія», FixMyStreet і Decidim. Аналіз показав, що кожне з розглянутих рішень охоплює лише окремі аспекти необхідної функціональності та має суттєві обмеження, зокрема відсутність адаптивності для різних міст або повноцінного зворотного зв'язку. Це обґрунтувало доцільність розробки власної комплексної системи й дозволило врахувати недоліки аналогів при формуванні вимог.

Обрано та обґрунтовано оптимальну архітектуру програмного комплексу. Систему побудовано за клієнт-серверною моделлю; серверну частину спроектовано як модульний моноліт на фреймворку NestJS із багаторівневою організацією коду за принципами чистої архітектури. Ключовим архітектурним рішенням стала багатоорендна модель із сутністю «місто» як орендарем, що безпосередньо усуває виявлений в аналогах недолік адаптивності та дозволяє одній інсталяції обслуговувати багато міст. Для забезпечення оперативності реалізовано взаємодію в реальному часі через протокол WebSocket, а для гарантованого інформування користувачів – надійну доставку сповіщень за патерном вихідних повідомлень (Transactional Outbox) із чергою завдань. Розмежування прав доступу реалізовано рольовою моделлю (RBAC).

Визначено та обґрунтовано технології й інструменти для реалізації серверної та клієнтської частин. Засадничим рішенням стало використання єдиної мови TypeScript для обох частин системи. Серверну частину реалізовано на Node.js, NestJS, ORM Prisma та СУБД PostgreSQL із застосуванням JWT і бібліотеки Passport (зокрема Google OAuth 2.0), черги BullMQ на сховищі Redis

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				
Змн.	Арк.	№ докум.	Підпис	Дата		106

та об'єктного сховища, сумісного з протоколом S3. Клієнтську частину побудовано на React і Next.js із використанням TanStack Query, Zustand, Tailwind CSS, Radix UI, Leaflet, Socket.IO та next-intl.

Здійснено проєктування структури бази даних та основних алгоритмів платформи. Визначено акторів і варіанти використання системи, побудовано діаграми варіантів використання, класів та діяльності. Спроектовано структуру реляційної бази даних PostgreSQL, що складається з 26 таблиць, із застосуванням багатоорендності, «м'якого» видалення та таблиці вихідних подій. Спроектовано й реалізовано ключові алгоритми роботи системи із застосуванням патернів проєктування – «Охоронець», «Декоратор», «Стратегія», «Адаптер», «Репозиторій», «Фабрика», «Постачальник–Споживач» та «Спостерігач».

Розроблено програмний комплекс і перевірено його на відповідність вимогам. Реалізовано повний функціонал платформи: створення та відстеження заявок на вирішення проблем міської інфраструктури з геоприв'язкою, медіадоказами, статусною моделлю опрацювання, призначенням на департаменти та звітністю; чат-комунікацію між заявником і муніципалітетом у реальному часі; публікацію термінових оголошень і офіційних новин; проведення опитувань і голосувань з налаштовуваною видимістю результатів; самообслуговуване створення міста з підтвердженням домену, а також адміністрування міст, користувачів і прав доступу. Працездатність системи підтверджено автоматизованим (Jest) та ручним тестуванням, за результатами якого всі перевірені функції відпрацювали коректно й відповідають визначеним вимогам.

Практична цінність роботи полягає в тому, що розроблений багатофункціональний міський портал може бути використаний органами місцевого самоврядування для цифровізації комунікації з жителями, підвищення прозорості роботи муніципальних служб та забезпечення зручного доступу громадян до міських сервісів. Завдяки багатоорендній архітектурі платформа є масштабованою та придатною для впровадження в різних містах. Подальший розвиток системи можливий у напрямі розширення цифрової екосистеми –

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				107
Змн.	Арк.	№ докум.	Підпис	Дата		

додавання модулів тематичних спільнот, волонтерських проєктів, заходів та краудфандингу громадських ініціатив, що дозволить сформувати безперервний цикл співпраці, розвитку й залучення громадян до життя міста.

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				
Змн.	Арк.	№ докум.	Підпис	Дата		108

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Київ Цифровий: вебпортал міських сервісів.
URL: <https://kyivdigital.com.ua/> (дата звернення: 03.03.2026).

2. Корецький Ю. І., Савіцький Р. С. Верифікація домену офіційного вебсайту муніципалітету за допомогою DNS TXT-записів. Комп'ютерні інтелектуальні системи та мережі (KICM-2026): матеріали XIX Всеукраїнської науково-практичної WEB-конференції аспірантів, студентів та молодих вчених, м. Кривий Ріг, 25–27 березня 2026 р. Кривий Ріг : Криворізький національний університет, 2026. С.101–102.

3. Корецький Ю.І., Савіцький Р.С. Порівняльний аналіз архітектурних патернів ізоляції даних у мультитенантних хмарних застосунках. Тези XVI Міжнародної науково-технічної конференції "Інформаційно-комп'ютерні технології", 02-03 квітня 2026 року. Житомир: «Житомирська політехніка», 2026. С.406.

4. Портал державних послуг «Дія». URL: <https://diia.gov.ua/> (дата звернення: 04.03.2026).

5. 10 Reasons to Switch to TypeScript in 2024. Medium, 2024. URL: <https://medium.com/@asierr/10-reasons-to-switch-to-typescript-in-2024-a856a99e6895> (дата звернення: 06.03.2026).

6. A Guide on Dependency Injection in NestJS. DigitalOcean, 2024. URL: <https://www.digitalocean.com/community/tutorials/a-guide-on-dependency-injection-in-nestjs> (дата звернення: 09.03.2026).

7. Add Interactive Maps to your App with Leaflet. OpenReplay Blog. URL: <https://blog.openreplay.com/add-interactive-maps-with-leaflet/> (дата звернення: 11.03.2026).

8. Architect multitenant solutions on Azure. Microsoft Learn, 2024. URL: <https://learn.microsoft.com/azure/architecture/guide/multitenant/overview> (дата звернення: 13.03.2026).

9. Asynchronous Programming in Node.js: Event-driven Architecture and Non-blocking I/O. Medium, 2024. URL:

<https://webcluesinfo.medium.com/asynchronous-programming-in-node-js-event-driven-architecture-and-non-blocking-i-o-41ac8ce52fc4> (дата звернення: 15.03.2026).

10. Benefits of using the OpenAPI (Swagger) specification for your API. Moesif Blog. URL: <https://www.moesif.com/blog/technical/api-design/Benefits-of-using-the-OpenAPI-Swagger-specification-for-your-API/> (дата звернення: 17.03.2026).
11. Client-Server overview. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Client-Server_overview (дата звернення: 19.03.2026).
12. Cunha P. Structuring a NestJS Project with DDD and Onion Architecture. Medium, 2024. URL: <https://medium.com/@patrick.cunha336/structuring-a-nestjs-project-with-ddd-and-onion-architecture-65b04b7f2754> (дата звернення: 21.03.2026).
13. Decidim: open-source participatory democracy. URL: <https://decidim.org/> (дата звернення: 24.03.2026).
14. Empowering Digital Democracy. Perspectives on Politics. Cambridge Core, 2024. URL: <https://www.cambridge.org/core/journals/perspectives-on-politics/article/empowering-digital-democracy/D39CBD8C8061EA6EE1B0FE0D64FA9E5D> (дата звернення: 26.03.2026).
15. FixMyStreet / mySociety. URL: <https://www.fixmystreet.com/> (дата звернення: 28.03.2026).
16. Gaidi L. Hexagonal, Onion, and Clean Architecture in NestJS. Medium, 2024. URL: <https://medium.com/@lamjed.gaidi070/hexagonal-onion-and-clean-architecture-in-nestjs-c58b526d9f3f> (дата звернення: 31.03.2026).
17. Impact of Information and Communication Technologies on Democratic Processes and Citizen Participation. Societies (MDPI), 2025. URL: <https://www.mdpi.com/2075-4698/15/2/40> (дата звернення: 02.04.2026).

		Ю.І. Корецький			ІІЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				110
Змн.	Арк.	№ докум.	Підпис	Дата		

18. JWT authentication: Best practices and when to use it. LogRocket Blog, 2024.
URL: <https://blog.logrocket.com/jwt-authentication-best-practices/> (дата звернення: 08.04.2026).
19. Kalam N. An Introduction to Prisma ORM: Simplifying Database Operations with Type-Safe Queries. Medium, 2023.
URL: <https://nazhimkalam.medium.com/an-introduction-to-prisma-orm-simplifying-database-operations-with-type-safe-queries-cb522447fe46> (дата звернення: 11.04.2026).
20. Khandelwal V. A Complete Guide to Tailwind CSS: The Utility-First CSS Framework. Medium, 2024.
URL: <https://medium.com/@vansh.khandelwal06/a-complete-guide-to-tailwind-css-the-utility-first-css-framework-thats-changing-web-design-0189d58940a1> (дата звернення: 14.04.2026).
21. Monolithic vs. Microservices Architecture. IBM, 2023.
URL: <https://www.ibm.com/think/topics/monolithic-vs-microservices> (дата звернення: 17.04.2026).
22. Monolith vs Microservice Architecture: A Comparison. Camunda, 2023.
URL: <https://camunda.com/blog/2023/08/monolith-vs-microservice-architecture-comparison/> (дата звернення: 19.04.2026).
23. Next.js for SEO and Server-Side Rendering strategies. Cheesecake Labs, 2024.
URL: <https://cheesecakelabs.com/blog/next-js-for-seo/> (дата звернення: 21.04.2026).
24. NodeJS Unit testing using Jest. BrowserStack, 2024.
URL: <https://www.browserstack.com/guide/unit-testing-for-nodejs-using-jest> (дата звернення: 23.04.2026).
25. OMG Unified Modeling Language (UML) v2.5.1. Object Management Group.
URL: <https://www.omg.org/spec/UML/> (дата звернення: 25.04.2026).
26. PostgreSQL Advantages: Benefits of Using PostgreSQL. Prisma Data Guide.
URL: <https://www.prisma.io/dataguide/postgresql/benefits-of-postgresql> (дата звернення: 27.04.2026).

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				111
Змн.	Арк.	№ докум.	Підпис	Дата		

27. Rana P. Implementing Background Job Queues in Node.js using BullMQ and Redis. Medium, 2024. URL: <https://medium.com/@rpratik.rana/implementing-background-job-queues-in-node-js-using-bullmq-and-redis-f179b699b1ba> (дата звернення: 29.04.2026).
28. Reliable communication using the Transactional Outbox Pattern. Cloudflight Engineering, 2023. URL: <https://engineering.cloudflight.io/reliable-communication-using-the-transactional-outbox-pattern> (дата звернення: 30.04.2026).
29. Richardson C. Pattern: Transactional outbox. microservices.io. URL: <https://microservices.io/patterns/data/transactional-outbox.html> (дата звернення: 02.05.2026).
30. Role Based Access Control (RBAC). NIST. URL: <https://csrc.nist.gov/projects/role-based-access-control> (дата звернення: 04.05.2026).
31. Smoke Testing: How to, Examples and Best Practices. Testlio, 2024. URL: <https://www.testlio.com/blog/smoke-testing-qa> (дата звернення: 06.05.2026).
32. Software Testing: Boundary Value Analysis. Baeldung on Computer Science, 2024. URL: <https://www.baeldung.com/cs/bva> (дата звернення: 08.05.2026).
33. Storage Wars: Cloudflare R2 vs Amazon S3. Vantage, 2024. URL: <https://www.vantage.sh/blog/cloudflare-r2-aws-s3-comparison> (дата звернення: 10.05.2026).
34. TanStack Query v5: The Complete Guide to Mastering Server State in React. Medium, 2024. URL: <https://medium.com/@learning.anand01/tanstack-query-v5-the-complete-guide-to-mastering-server-state-in-react-cbc1905a3095> (дата звернення: 13.05.2026).
35. Thakkar A. Row-Level Security (RLS) in PostgreSQL for Multi-Tenant SaaS Apps. Medium, 2025. URL: https://medium.com/@anand_thakkar/row-level-security-rls-in-postgresql-for-multi-tenant-saas-apps-ef8c324031d0 (дата звернення: 15.05.2026).

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				112
Змн.	Арк.	№ докум.	Підпис	Дата		

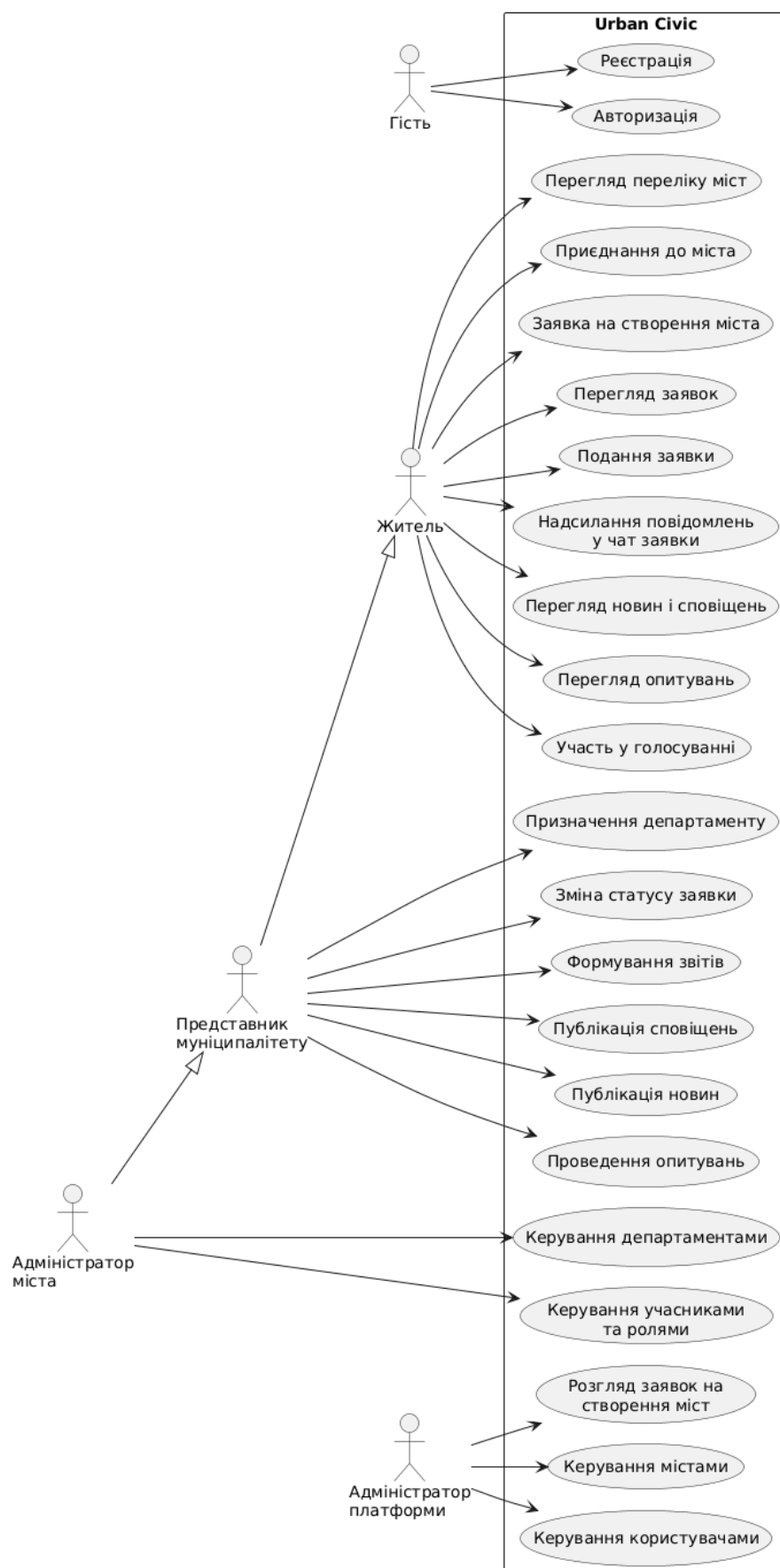
36. The ultimate guide to choosing a React component library. Contentful, 2024.
URL: <https://www.contentful.com/blog/ultimate-guide-react-component-library/> (дата звернення: 17.05.2026).
37. Top Software Design Patterns to Master in 2024. Capaciteam, 2024.
URL: <https://capaciteam.com/top-software-design-patterns/> (дата звернення: 19.05.2026).
38. Transforming Public Governance for Digital Democracy. OECD, 2024.
URL: https://www.oecd.org/en/publications/2024/10/the-oecd-reinforcing-democracy-initiative_458501ab/full-report/component-8.html (дата звернення: 21.05.2026).
39. Understanding OAuth: What Happens When You "Login with Google". ShiftAsia, 2024. URL: <https://shiftasia.com/community/understanding-oauth-what-happens-when-you-login-with-google/> (дата звернення: 23.05.2026).
40. Use Case Diagram (Unified Modeling Language). GeeksforGeeks, 2024.
URL: <https://www.geeksforgeeks.org/system-design/use-case-diagram/> (дата звернення: 25.05.2026).
41. WebSockets and Socket.IO: Real-Time Communication with Node.js. DEV Community, 2024. URL: <https://dev.to/imsushant12/websockets-and-socketio-real-time-communication-with-nodejs-2j5f> (дата звернення: 27.05.2026).
42. What Is an Entity-Relationship Diagram? A Complete Guide. G2, 2024.
URL: <https://www.g2.com/articles/entity-relationship-diagram> (дата звернення: 28.05.2026).
43. What Is the Repository Pattern? ITU Online, 2024.
URL: <https://www.ituonline.com/tech-definitions/what-is-the-repository-pattern/> (дата звернення: 29.05.2026).
44. Zustand vs Redux: Making Sense of React State Management. Wisp, 2024.
URL: <https://www.wisp.blog/blog/zustand-vs-redux-making-sense-of-react-state-management> (дата звернення: 30.05.2026).

		Ю.І. Корецький			ІІЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				113
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

		Ю.І. Корецький			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Р.С. Савіцький				114
Змн.	Арк.	№ докум.	Підпис	Дата		

Діаграма варіантів використання платформи



ER-діаграма

