

Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____ *Комп'ютерних наук* _____
(повна назва)

Кафедра _____ *Системотехніки* _____
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА Пояснювальна записка

_____ *перший (бакалаврський)* _____
(освітньо-кваліфікаційний рівень)

*Розробка компонентів інформаційної системи обліку електронних петицій
міської ради* _____
(тема роботи)

Виконав: студент IV курсу, групи ITKHу-19-1
спеціальність 122 – «Комп'ютерні науки» тип
програми освітньо-професійна
освітня програма комп'ютерні науки
(шифр і назва спеціальності)

_____ *Ігошин Д.В.* _____
(прізвище, ініціали)

Керівник _____ *доц. Решетнік В.М.* _____
(прізвище, ініціали)

Допускається до захисту

Зав. кафедри СТ _____ *проф. Гребеннік І.В.* _____
(підпис) (прізвище, ініціали)

2022 р.

Кваліфікаційна робота оформлена у відповідності до вимог діючих стандартів та методичних вказівок.

Матеріали кваліфікаційної роботи не містять відомостей, що заборонені для опублікування у відкритих виданнях.

Попередній захист проведено.

Керівник кваліфікаційної роботи

Решетнік В.М.

Факультет *комп'ютерних наук*

Кафедра *системотехніки*

Рівень вищої освіти *перший (бакалаврський)*

Спеціальність *122 «Комп'ютерні науки»*

Тип програми *освітньо-професійна*

Освітня програма *комп'ютерні науки*

ЗАТВЕРДЖУЮ:

Зав. кафедри *СТ*

проф. Гребеннік І.В.

" " 2022 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Студентові *Ігошину Данилу Валерійовичу*

(прізвище, ім'я, по батькові)

1. Тема роботи *Розробка компонентів інформаційної системи обліку електронних петицій міської ради*

затверджена наказом по університету від " 28 " квітня 2022р. № 582Ст

2. Термін подання студентом роботи *22 червня 2022 р.*

3. Вихідні дані до роботи (проекту) *Функції системи: облік електронних петицій. Перелік використовуваних програмних засобів: ОС Microsoft Windows v.10. Технічне забезпечення: IBM-сумісний ПК з процесором Core I5 10 TH GEN. Інтегроване середовище розробки Microsoft Visual Studio v.2019, Microsoft SQL Server v.18*

4. Зміст пояснювальної записки (перелік питань, що потрібно розробити)

4.1 Вступ. 4.2 Аналіз проблемної області. 4.2.1. Аналіз предметної області, яка визначає діяльність ІС електронних петицій органів місцевого самоврядування. 4.2.2. Процедура додаткової перевірки голосів. 4.2.3. Проблема обговорення петицій. 4.2.4. Статуси електронної петиції в ІС обліку електронних петицій. 4.2.5. Аналіз наявних систем обліку електронних петицій. 4.2.6. Постановка задачі. 4.3. Проектування та уточнення вимог компонентів системи обліку електронних петицій. 4.3.1. Розробка системних вимог. 4.3.2. Розробка функціональних вимог системи обліку електронних петицій. 4.3.3. Розробка моделі потоків даних компонентів ІС обліку електронних петицій. 4.3.4. Діаграма варіантів використання системи обліку електронних петицій. 4.3.5. Діаграма класів системи обліку електронних петицій. 4.3.6. Діаграма послідовності дій функції «Створити петицію». 4.3.7. Діаграма кооперацій функції «Створити петицію». 4.3.8. Діаграма станів об'єкту петиція. 4.3.9. Діаграма діяльності системи обліку електронних петицій. 4.4. Опис прийнятих проектних рішень. 4.4.1. Обґрунтування вибору мови програмування та середовища розробки. 4.4.2. Обґрунтування використаних технологій. 4.4.2.1. Архітектура ASP.NET. 4.4.2.2. Використання фреймворку Bootstrap. 4.4.3. Обґрунтування вибору СУБД. 4.4.4. Опис структури розробленої системи обліку електронних петицій. 4.4.5. Логічне та фізичне моделювання системи обліку електронних петицій. 4.4.6. Карта сайту. 4.4.7. Розробка інтерфейсу клієнтської частини ІС обліку електронних петицій. 4.4.8. Розробка контролеру для обробки петицій. 4.4.9. Розробка функцій аутентифікації та авторизації громадянина. 4.4.10. Захист інформації ІС обліку електронних петицій. 4.4.11. Аналіз дослідницької експлуатації та варіантів застосування ІС обліку електронних петицій. 4.5. Висновки. 4.6. Перелік джерел посилання. 4.7. Додатки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, плакатів)
5.1 Постановка задачі (1 аркуш формату А4). **5.1.** Контекстна діаграма системи обліку електронних петицій. **5.2.** Декомпозиція контекстної діаграми. **5.3.** Декомпозиція функції «Оформлення нової петиції». **5.4.** Декомпозиція функції «Голосування за петицію». **5.5.** Декомпозиція функції «Облік петицій». **5.6.** Контекстна діаграма системи DFD. **5.7.** Декомпозиція функції «Облік електронних петицій». **5.8.** Декомпозиція функції «Оформлення нової петиції». **5.9.** Декомпозиція функції «Авторизація громадянина». **5.10.** Декомпозиція функції «Голосування за петицію». **5.11.** Декомпозиція підфункції «Голосування». **5.12.** Декомпозиція функції «Облік петицій». **5.13.** Декомпозиція «Сповіщення про зміну стану петицій». **5.14.** Діаграма варіантів використання. **5.15.** Діаграма класів. **5.16.** Діаграма класів на основі моделі MVC. **5.17.** Діаграми послідовності. **5.18.** Діаграма кооперації функції «Створення петиції». **5.19.** Діаграма зміну станів об'єкта петиція в системі. **5.20.** Діаграми діяльності.

6. Дата видачі завдання 2 травня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

Пор. №	Назва етапів атестаційної роботи	Термін виконання етапів роботи	Примітка
1.	Отримання завдання атестаційної роботи	2.05.2022	
2.	Аналіз завдання, літератури та аналогів з теми атестаційної роботи	2.05 - 10.05.2022	
3.	Проектування та уточнення вимог до системи	10.05 - 15.05.2022	
4.	Структурне проектування	15.05 - 20.05.2022	
5.	Вибір середовища розробки програми	20.05 - 25.05.2022	
6.	Розробка програми	25.05 – 1.06.2022	
7.	Експлуатація програми	1.06 – 3.06.2022	
8.	Розробка «Посібника користувача»	3.06 – 5.06.2022	
9.	Оформлення пояснювальної записки та програмної документації	6.06 – 15.06.2022	
10.	Оформлення графічної частини та презентаційних матеріалів комп'ютерного захисту	17.06.2022	
11.	Представлення на рецензування	19.06.2022	
12.	Попередній захист	20.06.2022	
13.	Представлення атестаційної роботи в ДЕК	22.06.2022	

Студент _____ Ігошин Д.В.
 (підпис)

Керівник роботи _____ доцент Решетнік В.М.
 (підпис)

РЕФЕРАТ

Кваліфікаційна робота містить: 143 с., 3 табл., 51 рис., 4 додатки, 26 джерел інформації, 7 лістингів.

UML, СУБД, ІНФОРМАЦІЙНА СИСТЕМА, КЛІЄНТСЬКА ЧАСТИНА, СИСТЕМА ОБЛІКУ, ПЕТИЦІЯ, ГОЛОСУВАННЯ, ФУНКЦІОНАЛЬНІ ВИМОГИ, IDEF0, SQL, C#, ASP NET CORE 5, HTML, CSS, RAZER PAGES, MS SQL SERVER

Об'єктом розробки – процес обліку електронних петицій міської ради.

Предметом розробки – інформаційні технології і програмні методи розробки компонентів інформаційної системи, що дозволяє автоматизувати облік електронних петицій міської ради.

Мета роботи: розробка компонентів інформаційної системи обліку електронних петицій міської ради.

Методи розробки – метод аналізу літератури, метод аналізу нормативно-правової документації, огляд існуючих систем, системний підхід, методи структурного аналізу і моделювання реляційних баз даних, методи функціонального моделювання, методи проектування клієнт-серверних додатків для Інтернет-мереж.

В результаті роботи було проаналізовано предметну область, спроектовано та розроблено компоненти інформаційної системи обліку електронних петицій.

Галузь застосування – в органах місцевого самоврядування територіальних громад, де потрібна система автоматизації та обліку електронних петицій.

ABSTRACT

Qualification work contains: 143 pages, 3 tables, 51 figures, 4 appendices, 26 sources of information, 7 listings.

UML, DBMS, INFORMATION SYSTEM, CLIENT PART, ACCOUNTING SYSTEM, PETITION, VOTING, FUNCTIONAL REQUIREMENTS, IDEF0, SQL, C#, ASP NET CORE 5, HTML, CSS, RAZER PAGES, MS SQL SERVER

The object of development is the process of accounting for electronic petitions of the city council.

The subject of development – information technology and software methods for developing components of the information system, which allows you to automate the accounting of electronic petitions of the city council.

Purpose: development of components of the information system of accounting of electronic petitions of the city council.

Development methods – method of literature analysis, method of analysis of legal documentation, review of existing systems, system approach, methods of structural analysis and modeling of relational databases, methods of functional modeling, methods of designing client-server applications for Internet networks.

As a result of the work the subject area was analyzed, the components of the information system of electronic petitions accounting were designed and developed.

Scope – use in local governments of settlements where a system of automation and accounting of electronic petitions is required.

ЗМІСТ

РЕФЕРАТ.....	5
ABSTRACT.....	6
ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАК, ОДИНИЦЬ І ТЕРМІНІВ.....	9
ВСТУП.....	10
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ.....	12
1.1 Аналіз предметної області, яка визначає діяльність ІС електронних петицій органів місцевого самоврядування	12
1.2 Процедура додаткової перевірки голосів	14
1.3 Проблема обговорення петицій.....	15
1.4 Статуси електронної петиції в ІС обліку електронних петицій	15
1.5 Аналіз наявних систем обліку електронних петицій	16
1.6 Постановка задачі	22
2 ПРОЄКТУВАННЯ ТА УТОЧНЕННЯ ВИМОГ ДО КОМПОНЕНТІВ СИСТЕМИ ОБЛІКУ ЕЛЕКТРОННИХ ПЕТИЦІЙ.....	24
2.1 Розробка системних вимог	24
2.2 Розробка функціональних вимог системи обліку електронних петицій.....	24
2.3 Розробка моделі потоків даних компонентів ІС обліку електронних петицій	30
2.4 Діаграма варіантів використання системи обліку електронних петицій.....	35
2.5 Діаграми класів системи обліку електронних петицій	38
2.6 Діаграма послідовності прецеденту «Створити петицію».....	41
2.7 Діаграма кооперацій функції «Створити петицію»	42
2.8 Діаграма станів об'єкту «Петиція»	43
2.9 Діаграма діяльності системи обліку електронних петицій.....	44
3 ОПИС ПРИЙНЯТИХ ПРОЕКТНИХ РІШЕНЬ	47
3.1 Обґрунтування вибору мови програмування та середовища розробки.....	47
3.2 Обґрунтування використаних технологій	48
3.2.1 Архітектура ASP.NET	49
3.2.2 Використання фреймворку Bootstrap.....	50
3.3 Обґрунтування вибору СУБД.....	51
3.4 Опис структури розробленої системи обліку електронних петицій	52

3.5	Логічне та фізичне моделювання даних системи обліку електронних петицій	53
3.6	Карта сайту веб-застосунку обліку електронних петицій	60
3.7	Розробка інтерфейсу клієнтської частини ІС	61
3.8	Розробка контролеру для обробки петицій	63
3.9	Розробка функцій аутентифікації та авторизації громадянина	67
3.10	Захист інформації ІС обліку електронних петицій	68
3.11	Аналіз дослідної експлуатації та варіантів застосування ІС обліку електронних петицій	71
ВИСНОВКИ		78
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ		79
ДОДАТОК А Графічні матеріали.....		81
ДОДАТОК Б Посібник користувача		104
ДОДАТОК В Текст програми.....		121
ДОДАТОК Г Відомість атестаційної роботи.....		142

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАК, ОДИНИЦЬ І ТЕРМІНІВ

Фреймворк	–	програмна платформа, яка визначає структуру програмної системи, програмне забезпечення, що полегшує розробку і об'єднання різних компонентів великого програмного проекту
СУБД	–	система управління базами даних
ІС	–	інформаційна система
БД	–	база даних
URL	–	uniform resource locator
UML	–	уніфікована мова моделювання (Unified Modeling Language)
SQL	–	мова структурованих запитів
SATD	–	(Structured Analysis and Design Technique) – методологія структурного аналізу і проектування)
IDEF0	–	методологія функціонального моделювання і графічна нотація
DFD	–	діаграма потоків даних

ВСТУП

Сучасний світ – це світ інформаційних технологій, розвиток людства дійшов до того моменту, що більшість систем, які працювали раніше у письмовому виді з великою кількістю людських резервів нині переведено у цифровий формат та автоматизовано. Інформаційні технології впливають на наше життя та покращують, роблять зручніше. Сучасний світ швидкоплинно розвивається, населенні пункти активно розвиваються та громадяни хочуть впливати на розвиток. Для цього є петиції в яких громадяни можуть закликати органи місцевого самоврядування до тих чи інших дій.

Актуальність теми дипломної роботи обумовлена соціальною позицією громадян, які хочуть впливати на розвиток свого населеного пункту. Подання петиції до органів місцевого самоврядування – це довгий та складний процес, який займає багато часу, документів, слухань та підписів громадян. Більшість громадян повинні працювати та в них не має часу на соціальну позицію. Тому автоматизована система обліку електронних петицій надала б змогу громадянам роботи свій внесок у розвиток міста, селища чи іншого населеного пункту в якому вони проживають. Тема обумовлена не тільки громадянами та також органами місцевого самоврядування. За допомогою системи, органи місцевого самоврядування зможуть спростити процес подання петицій, позбавитися документування за петиціями та матимуть змогу отримувати звіти по тій чи іншій проблемі, по уподобаннях громадян в подробицях.

Об'єкт дослідження кваліфікаційної роботи – процеси обліку електронних петицій: створення, голосування, обговорення, обліку петицій та документування всіх облікових даних.

Предмет дослідження кваліфікаційної роботи є інформаційні технології й програмні методи створення веб-застосунку інформаційної системи обліку електронних петицій.

Ціль кваліфікаційної роботи є розгляд алгоритму подання електронних петицій, виявлення недоліків у реалізованих системах, та також пошук шляхів для

вирішення існуючих проблем та покращення механізму подання електронних петицій, голосування, обговорення та документування облікових даних.

Для досягнення цілей, поставленні наступні завдання:

- проаналізувати предметну область та виявити проблеми та недоліки, можливі вдосконалення існуючих систем та алгоритмів подання петицій;
- уточнити та систематизувати вимоги до системи, які б не мали існуючих недоліків;
- спроектувати інформаційну систему без існуючих недоліків та з можливими перевагами;
- розробити інформаційну систему та провести експлуатацію на відповідність висунутим вимогам.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз предметної області, яка визначає діяльність ІС електронних петицій органів місцевого самоврядування

Наразі серед громадян України набуло значного поширення використання ІС подання петицій як на рівні президента, так і на рівні обласних та міських рад. Сьогодні вже існують такі системи та їх кількість значно зростає з кожним роком. Наприклад, система подання петицій президенту вже розроблена і працює. Системи подання петицій на рівні обласних та міських рад тільки почали розробляти, а ті, що розроблені, мають істотні недоліки. Отже, обрана тема є досить актуальною.

Електронна петиція – це зручна форма колективного звернення громадян до Президента України, Верховної Ради та Кабінету Міністрів, або органу місцевого самоврядування для розв'язання певної проблеми або пропозицій які громадяни хотіли б ввести. ІС подання електронних петицій регулюються правовими документами [1].

Алгоритм подання петиції [2] передбачає, що громадяни України мають право створювати петиції, голосувати за них. Щоб створити або проголосувати за певну петицію, громадянин повинен зареєструватися (або увійти, якщо у нього є обліковий запис) на сервісі. Для цього необхідно виконати «Реєстрацію / Авторизацію». Ввести дані, якщо користувач вже має обліковий запис.

Авторизація [3] – це процес надання певній особі прав на виконання деяких дій, функціоналу у системі та також процес при якому виконується перевірка даних прав під час спроби виконання цих дій.

Якщо користувач не має облікового запису він повинен зареєструватися. Для цього потрібна така інформація: прізвище ім'я по батькові (ПІБ) громадянина, електронна пошта на яку потім будуть відправлятися повідомлення про змін стану петицій, номер телефону та адреса громадянина. Для перевірки громадянина потрібні копії паспорту з пропискою, або інші документи які вказують на

належність громадянина до міста. Такими документами є електронний підпис, ідентифікаційний код платника податків України, або обліковий запис в одному з державних банків чи систем міста. Як, наприклад «Картка Харків'янина». Після реєстрації громадянин повинен бути перевірений адміністратором сервісу. Це необхідно для того, щоб зрозуміти, чи є користувач городянином міста. Після успішної верифікації, користувач буде переадресований на головну сторінку сайту через кілька секунд очікування, яке буде показане на сторінці сайту.

Щоб підтримати наявну електронну петицію, користувач повинен продивитися петиції, відсортувати за параметрами, які представляють для нього інтерес, і перейти на сторінку петиції. Після ознайомлення з текстом петиції користувач може проголосувати. Голос користувача зараховується відразу. Громадянин, який не авторизувався на сайті, повинен спочатку увійти в систему. І якщо обліковий запис користувача не було підтверджено, слід дочекатися отримання електронного листа з підтвердженням.

Для створення нового електронного запиту, необхідно вибрати створення нової петиції. Для користувача, який вже увійшов на сайт, відкриється спеціальна форма для створення нової електронної петиції. Користувач, який не увійшов на сайт не матиме такої можливості. На сторінці створення нової петиції, у спеціально відведених для цього полях, зазначають суть звернення та текст самої петиції. Після створення петиції, виконується перевірка адміністратором сайту на відповідність вимогам законодавства [4]. Перевірений електронний запит буде оприлюднено на сайті зі статусом «Збір підписів триває» протягом двох робочих днів після подання автором (ініціатором) петиції. Якщо петиція не відповідає вимогам, оприлюднення петиції не здійснюватиметься, про що автор петиції отримає повідомлення на електронну адресу, вказану при реєстрації облікового запису.

Звернення до місцевої влади збирається протягом дев'яноста днів і залежить від збору необхідної кількості голосів громадян від дня розміщення петиції на сайті. Якщо електронна петиція не набере необхідної кількості голосів громадян протягом терміну збору голосів, вона зберігається на веб-сайті після закінчення

терміну та розглядається як звернення громадян відповідно до чинного законодавства [5]. Електронна петиція яка набрала потрібну кількість голосів, буде оприлюднена на сайті зі статусом «На розгляді» не пізніше трьох днів після отримання петицією необхідної кількості голосів. Електронна петиція передається до необхідного органу місцевого самоврядування на розгляд протягом десяти робочих днів, починаючи з дня оприлюднення інформації про зібрання потрібної кількості голосів.

Про підтримку або відхилення електронної петиції повідомляється ініціатору петиції та вона з'являється у розділі звітності. Відповідь на електронну петицію опубліковується зі статусом «З відповіддю». Усі електронні звернення зберігаються не менше трьох років з дати подання петиції. Петиції, що очікують на розгляд отримують статус «Архів».

1.2 Процедура додаткової перевірки голосів

Згідно з правовими документами, на сайті електронних петицій обов'язково має бути забезпечено недопущення автоматичного введення інформації ботами. Ботом називають вузькоспеціалізоване програмне забезпечення яке виконує автоматичні дії за заданим алгоритмом через інтерфейси, які були створені для людей. Наприклад: на сайті електронних петицій президенту України були спроби використати подібну автоматизацію для накручування голосів. Щоб уникнути подібних дій на сервісі було впроваджено додаткову перевірку голосувань [6].

Алгоритм перевірки додаткових голосів такий: на час перевірки голоси, які додатково перевіряються, фіксуються, але не враховуються у загальній кількості тих голосів, які були віддані за електронну петицію тобто вони не публікуються. Під час такої перевірки може виконуватися додаткова поштова розсилка з проханням особисто підтвердити свій голос.

1.3 Проблема обговорення петицій

Наявною проблемою більшої кількості сервісів електронних петицій є відсутність форуму або коментарів під петиціями. Проблема є актуальною тому, що більшість громадян при розгляді петиції можуть трактувати електронну петицію іншим змістом, ніж те що має на увазі автор (ініціатор) петиції, а запитати змоги немає.

Коментар – це роз’яснювальні та критичні зауваження або запитання з приводу яких-небудь подій, явищ або суті написаного тексту. Роз’яснення думок з приводу написаного. Коментар у своїй структурі має такі поля: ПІБ людини яка його залишає, дата та час, суть звернення у коментарі.

Ще однією можливістю для обговорення людей у мережі Інтернет є форум. Веб-форум – це сервіс для спілкування між користувачами мережі Інтернет (понад двоє учасників) на одну або декілька тем (залежить від спеціалізації форуму). Одна із форм соціальних мереж. Форум є загалом великим окремим додатком.

Концепція роботи форуму полягає у створенні користувачами своїх тем з їх подальшим обговоренням, шляхом розміщення повідомлень усередині цих тем. Користувачі можуть коментувати заявлену тему, ставити запитання за нею та отримувати відповіді, а також самі відповідати на запитання інших користувачів форуму та давати їм поради.

Веб-форум – велика програма за обсягом розробки та функціоналу, відповідає вимогам повноцінного сервісу.

Коментарі будуть доповнювати функціонал петицій. Автори зможуть пояснювати зміст та сенс петиції, а користувачі матимуть змогу ставити питання та отримувати відповідь від автора електронної петиції.

1.4 Статуси електронної петиції в ІС обліку електронних петицій

Статус – це стан петиції у процесі обробки у системі. Під час оформлення нової петиції вона може мати статус «Нова петиція». Петиція з таким статусом не

з'являється у пошуку петицій. Після перевірки адміністратором на заборонену інформацію петиції присвоюється статус «Триває збір підписів». Статус «На розгляданні» означає, що петиція збрала необхідну кількість підписів та очікує розглядання органом місцевого самоврядування до якого була направлена. За результатами прийняття рішення щодо петиції, адміністратор отримує рішення, публікує рішення та змінює статус електронної петиції – «З відповіддю». Електронні петиції які не набрали необхідної кількості голосів за виділений проміжок часу, система в автоматичному режимі присвоює статус «Архів».

1.5 Аналіз наявних систем обліку електронних петицій

Проведено аналіз чинних систем електронних петицій різних великих міст, таких як Харків, Київ, Дніпро та Одеса. Ці міста мають мільйонне населення та найкраще показують, як повинна працювати дана система. ІС обліку електронних петицій проаналізовано за такими критеріями:

- правильна реєстрація, яка не дозволить голосувати за рішення в місті, в якому громадянин не проживає та не має прописки або інших документальних підтверджень;
- зручний та багатофункціональний інтерфейс сайту;
- платформа під петиціями, де громадяни могли б виказати свої думки щодо петиції (коментарі);
- функціональне сортування петицій (за категоріями, за статусами петиції тощо);
- підзвітність та прозорість сервісу петицій.

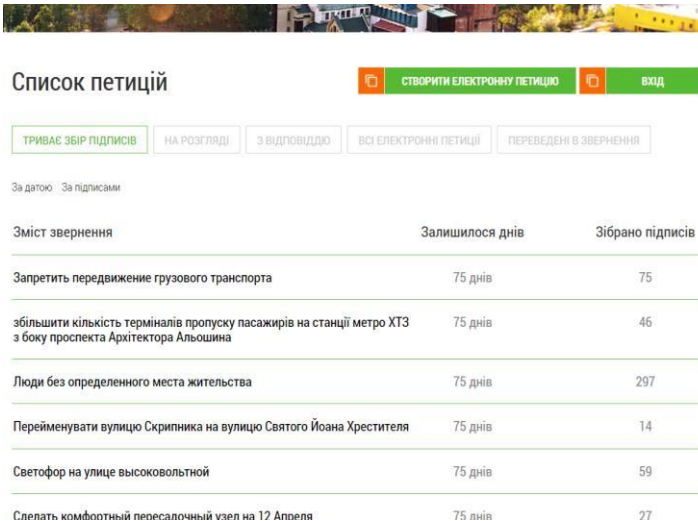
ІС електронних петицій міста Харкова [7] відрізняється від інших систем мінімалістичним інтерфейсом, але малим реалізованим функціоналом. Звітності за петицією та категорій електронних петицій немає. Сортування можливе лише за статусами петицій, за датою опублікування та за кількістю петицій. інтерфейс переліку петицій подано на рисунку 1.1.

Неzareєстрований користувач ІС може продивитися петиції, сортувати петиції. Реєстрація в застосунку проста, та не потребує від громадянина

супроводжувальних документів або сторонніх API-систем. Як наприклад, bankID ПриватБанку, де база даних з переліком документів що підтверджують: фотографії сторінок паспорта, прописки та ідентифікаційний код громадянина України. Форма реєстрація нового користувача подано на рисунку 1.2.

Це недолік, що кожен охочий, навіть не з України може зареєструватися та створювати петиції чи голосувати за чинні петиції на сайті.

В цьому сервісі громадяни, які авторизовані в системі не можуть обговорити петицію у коментарях чи форумі.



Зміст звернення	Залишилося днів	Зібрано підписів
Запретить передвижение грузового транспорта	75 днів	75
збільшити кількість терміналів пропуску пасажирів на станції метро ХТЗ з боку проспекта Архітектора Альошина	75 днів	46
Люди без определенного места жительства	75 днів	297
Переименовать вулицю Скрипника на вулицю Святого Іоана Хрестителя	75 днів	14
Светофор на улице высоковольной	75 днів	59
Слепая комфортный пешеходный взел на 12 Апеля	75 днів	27

Рисунок 1.1 – Головна сторінка веб-ресурсу Харківських електронних петицій

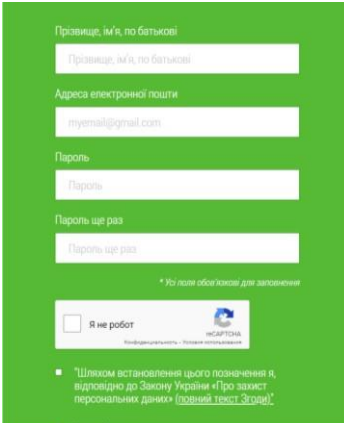


Рисунок 1.2 – Форма для реєстрації користувача та створення особистого кабінету

Столична ІС електронних петицій [8] має якісний та приємний інтерфейс, та великий реалізований функціонал. Сортування можна виконати за сьома

параметрами за: датою, напрямками (категорії), статусом, кількістю підписів, автором петиції (ПІБ), ключовими словами, датою опублікування (рис 1.3). Реалізована функція звітності за останній місяць: за виконаними петиціями, за кількістю створених петицій, петицій які набрали необхідну кількість голосів та переведені у статус «На розгляді». Статистику можна викачати з сайту у PDF-форматі (рис 1.4).

Реєстрація в веб-застосунку реалізована вірним та надійним способом. В алгоритмі реєстрації або авторизації у сервісі використовуються наступні можливості: КиївID (Єдиний обліковий запис Киянина), ЕЦП (Електронний цифровий підпис), PrivatBankID, BankID НБУ, Mobile ID Kyivstar, Mobile ID Vodafone (рис 1.5).

В сервісі користувачі не можуть обговорювати петицію або залишити коментар зі своїми думками.

Петиції

Рисунок 1.3 – Веб-сторінка з переліком пунктів для сортування петицій на сайті



Рисунок 1.4 – Можливі типи реєстрації для того, щоб залишити петицію або підписати петицію

Статистика

від 1 жовтня 2015 р. по 26 листопада 2021 р.

весь час рік місяць квартал

Завантажити звіт

6827 петицій опубліковано	57 петицій що набрали достатньо підписів	797988 активних користувачів	2837073 віддано підписів	46 петицій підтримано владою
-------------------------------------	-------------------------------------------------------	----------------------------------------	------------------------------------	-------------------------------------------

Рисунок 1.5 – Можливість переглянути статистику з петицій та можливість зберегти на ПК PDF-звіт

ІС обліку електронних петицій міста Одеси [9] має дружній до користувача інтерфейс та реалізує достатній функціонал. Петиції розбиті за рубриками та підкатегоріями. Сортування виконується за параметрами: рубрики, статус петицій, кількість голосів (рис 1.6).

Громадяни можуть зареєструватися для того, щоб мати можливість створювати петиції та голосувати. Але голосування, як і створення петиції можливе лише після верифікації облікового запису. В результаті будь-яка людина не зможе проголосувати або створити петицію без верифікації. При створенні петиції до неї можна долучити файли, які будуть прикріплені до петиції (рис 1.7).

На відміну від попередніх сервісів є можливість коментування на сайті Одеських електронних петицій, де користувачі можуть обмінятися думками щодо петицій. Нажаль відсутня підзвітність щодо петицій, вона знаходиться на самому сайті міськради міста Одеси і ніяких переходів до неї немає, треба шукати власноруч.

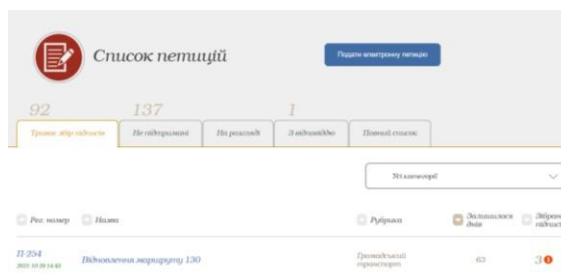


Рисунок 1.6 – Головна сторінка веб-ресурсу Одеських петицій

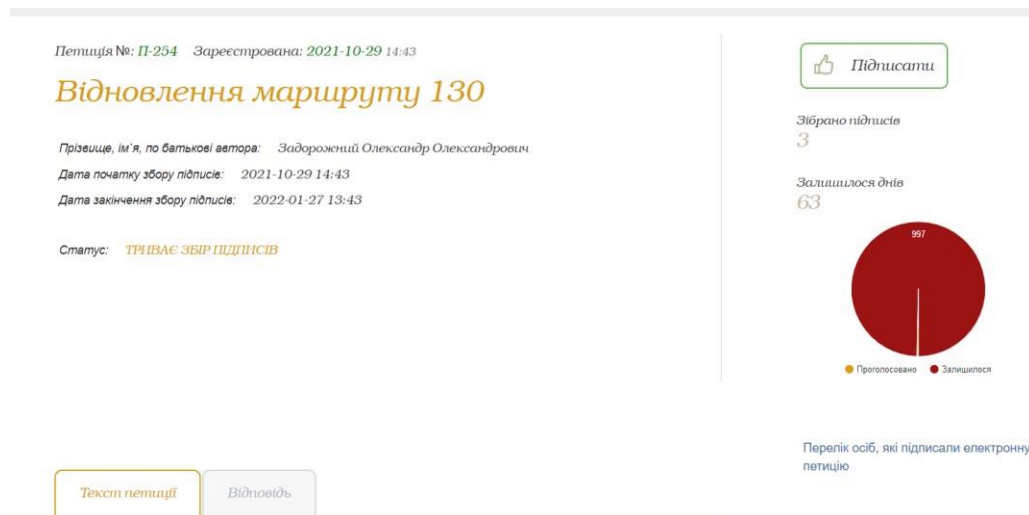


Рисунок 1.7 – Представлення петиції на сайті

Реєстрація у веб-застосунку міста Дніпра [10] відбувається за допомогою банківської системи ідентифікації або за допомогою електронного підпису громадянина. Будь-яка особа яка не є громадянином України та міста Дніпро не зможе створювати та голосувати за чинні петиції (рис 1.8).

Інтерфейс застосунку має недоліки у дизайні та зображенні деяких петицій, нагромадження елементів які відволікають (рис 1.9). Коментарів або іншої платформи для обговорення у даного сервісу немає.

У веб-ресурсі мається сортування за категоріями петицій та статусом голосування.

Банківські системи ідентифікації або ЕЦП

Увійти в єдиний кабінет →

Вхід у систему з обліковим записом

Електронна пошта

Пароль

Увійти [Забули пароль?](#)

[Не маєте облікового запису?](#)

Чому варто використовувати єдиний кабінет?

Щоб підписувати петиції в деяких містах або громадах використовується обов'язкова авторизація через BankID або електронного цифрового підпису через єдиний кабінет

Рисунок 1.8 – Форма входу до особистого кабінету

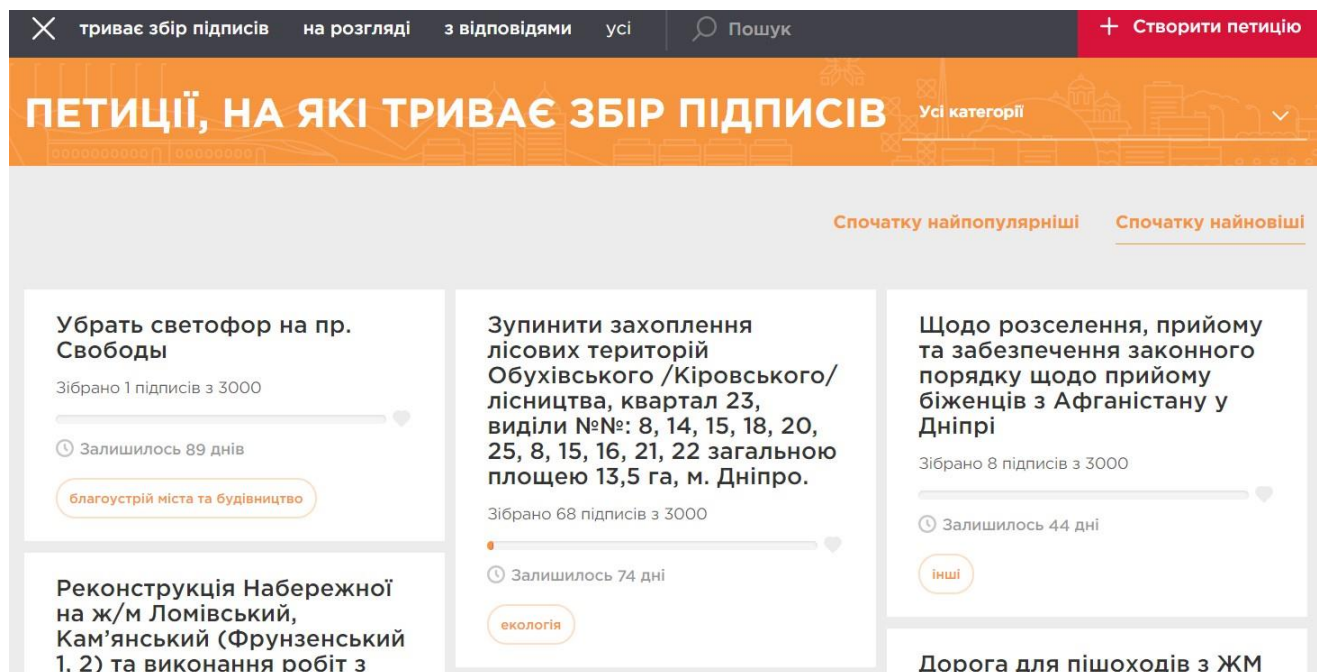


Рисунок 1.9 – Головна сторінка веб-ресурсу петицій Дніпра

Отже, проведений аналіз реалізованих систем доводить, що завдання розробки компонентів інформаційної системи обліку електронних петицій міської ради є актуальним. Веб-застосунки дуже схожі своїм функціоналом, але є такі недоліки як: відсутність коментарів для обговорення петицій, не дохідливий та не послідовний інтерфейс, недостатня реєстрація з ненадійною верифікацією.

Провівши аналіз веб-застосунків електронних петицій найбільших міст України маємо такі результати: зліва в таблиці надано назви міст, а зверху критерії. Слово «так» в таблиці означає, що функціонал присутній, слово «ні», що такого немає.

Таблиця 1.2 – Порівняння веб-ресурсів за можливостями

Місто	Форум або коментарі	Багатофункціональний та простий інтерфейс	Підзвітність та прозорість	Правильна реєстрація	Зручне сортування петицій
Харків	Ні	Так	Так	Ні	Ні
Київ	Ні	Так	Так	Так	Так
Одеса	Так	Ні	Ні	Так	Так
Дніпро	Ні	Ні	Ні	Так	Так

Результати аналізу доводять, що ІС обліку електронних петицій міста Харкова та Дніпра, мають найбільш невдалу реалізацію відносно висунутих критеріїв аналізу порівняно з містами Київ та Одеса. Отже, треба розробити ІС обліку електронних петицій міської ради та позбавитися недоліків вже реалізованих систем.

1.6 Постановка задачі

Виходячи з аналізу предметної області ІС електронних петицій міської ради, як і будь-яка інша ІС потребує продуманого підходу послідовної розробки. Аналіз предметної області та аналіз реалізованих систем обліку петицій дозволяє зробити висновок, що розробка компонентів інформаційної системи залишається актуальною.

Користувачами системи є адміністратор, громадянин (користувач) та гість.

Адміністратор – це людина, яка має права щодо верифікації користувачів, призначення інших адміністраторів, перевірки петицій на предмет цензурування.

Громадянин (користувач) – це людина, яка має обліковий запис у системі та вже верифікована адміністратором системи, як городянин. Може переглядати електронні петиції, сортувати петиції за параметрами, створювати петиції, коментувати петиції, голосувати за петиції та може редагувати власний обліковий запис.

Гість – це людина, яка не зареєстрована в системі, не має облікового запису, не може створювати петиції, брати участь у голосуванні та обговоренні петицій. Але може зареєструватися, сортувати та переглядати петиції.

Мета розвитку ІС обліку електронних петицій міської ради:

а) з боку міської ради:

- 1) підвищення швидкості обробки петицій громадян;
- 2) отримання звітності за різними параметрами. Наприклад, про кількість петицій, кількість свідомих городян та кількість прийнятих петицій голосуванням;
- 3) підвищення надійності голосування громадян та волевиявлення;

4) надання виборчим процесам прозорості та впливу людей на розвиток міста;

б) з боку користувача (громадянина):

- 1) спрощення процесу подачі петицій до органів міського самоврядування;
- 2) забезпечення правильності та надійності голосування;
- 3) забезпечення можливості обговорення електронних петицій у коментарях;
- 4) зменшення часу на обробку електронних петицій.

Для досягнення мети розробки треба на основі проведеного аналізу предметної області та аналізу реалізованих ІС обліку електронних петицій виконати такі завдання:

- уточнити системні та функціональні вимоги до ІС;
- виконати функціональне та об'єктно-орієнтоване проектування компонентів ІС;
- виконати логічне та фізичне моделювання даних БД;
- провести моделювання клієнтської частини ІС;
- розробити вимоги до функцій серверної частини ІС;
- розробити програмне забезпечення для компонентів ІС;
- провести аналіз досвідної експлуатації та варіантів використання ІС.

2 ПРОЄКТУВАННЯ ТА УТОЧНЕННЯ ВИМОГ ДО КОМПОНЕНТІВ СИСТЕМИ ОБЛІКУ ЕЛЕКТРОННИХ ПЕТИЦІЙ

2.1 Розробка системних вимог

Компоненти, що розробляються, призначені для обліку електронних петицій у веб-застосунку міської ради. Для функціонування необхідні такі компоненти:

- база даних та схема даних;
- контролери (Controller) – використовуються для обробки запитів та повернення результатів користувачам;
- репозиторії (Repository) – сховища даних, яке тимчасово зберігає інформацію з БД;
- представлення (View) – приймає та виводить інформацію від користувачів;
- технічні застосунки (сервер).

Виходячи з цілей розробки створення компонентів системи, розглянемо системні вимоги:

- компоненти повинні бути реалізовані у вигляді веб-застосунку;
- база даних повинна взаємодіяти з репозиторіями;
- репозиторії повинні взаємодіяти з контролерами;
- контролери повинні взаємодіяти з сервером та обробляти запити користувачів;
- контролери повинні взаємодіяти з представленнями, приймати та виводити інформацію через них.

2.2 Розробка функціональних вимог системи обліку електронних петицій

Найбільш зручною та наглядною для формалізації та опису бізнес процесів є методологія функціонального моделювання – IDEF0. Опис виглядає як «чорний ящик» із виходами, входами, керуванням та механізмами, який поступово деталізується до необхідного рівня. На ній зображені сукупність взаємодіючих робіт та функцій [11].

Перша діаграма – це контекстна діаграма – головна. Вона вміщає тільки один блок – це головний процес системи. Головний процес системи – це облік електронних петицій. В процес входить інформація про уподобання громадянина (користувача) та дані авторизації. Ця інформація потрібна для створення петицій, голосування та обговорення громадян. Голосування та створення петицій регулюються правовими документами та регламентом подання петицій. В процесі «Облік петицій» приймають участь громадянин, ІС та адміністратор. В результаті виконання процесу – звіт про діяльність процесу «Облік електронних петицій» та повідомлення про зміну стану петиції за яку голосував чи створював громадянин.

Контекстну діаграму подано на рисунку 2.1.

Після опису системи в цілому, проводиться розбиття цієї системи на підсистеми – це процес називається функціональною декомпозицією. Після декомпозиції виходять підсистеми першого рівня, а на далі проводиться декомпозиція великих фрагментів на більш дрібні і так далі поки не досягнуто необхідного рівня деталізації. Досягається відповідність моделі реальним процесам на кожному рівні.

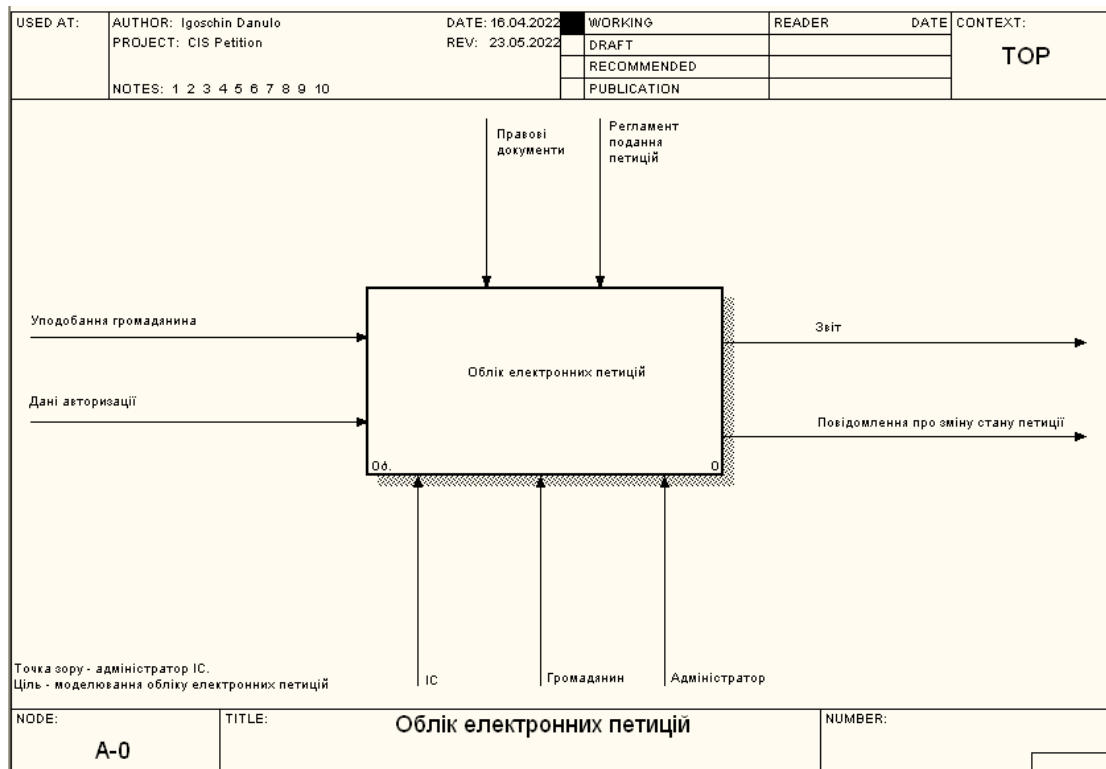


Рисунок 2.1 – Контекстна діаграма

Після декомпозиції основного процесу «Облік електронних петицій» основний процес розбито на чотири підпроцеси:

- оформлення нової петиції – створення електронної петиції зареєстрованим та авторизованим громадянином у веб-застосунку. Введення інформації про тему та суть петиції, прикладання копій необхідних документів та зображень, фіксування дати та часу, а також ПІБ автора петиції;
- голосування за петицію – вибір електронної петиції за уподобанням громадянина та якщо громадянин авторизований, віддача голосу за петицію;
- облік петицій – підрахунок голосів за петицією. При набиранні петицією необхідної кількості голосів, виконується зміна стану та відправлення на розгляд;
- сповіщення про зміну стану петиції – при зміні стану петиції користувач який голосував за неї чи створив, отримає повідомлення на електронну пошту.

Деталізація контекстної діаграми, наведено на рисунку 2.2.

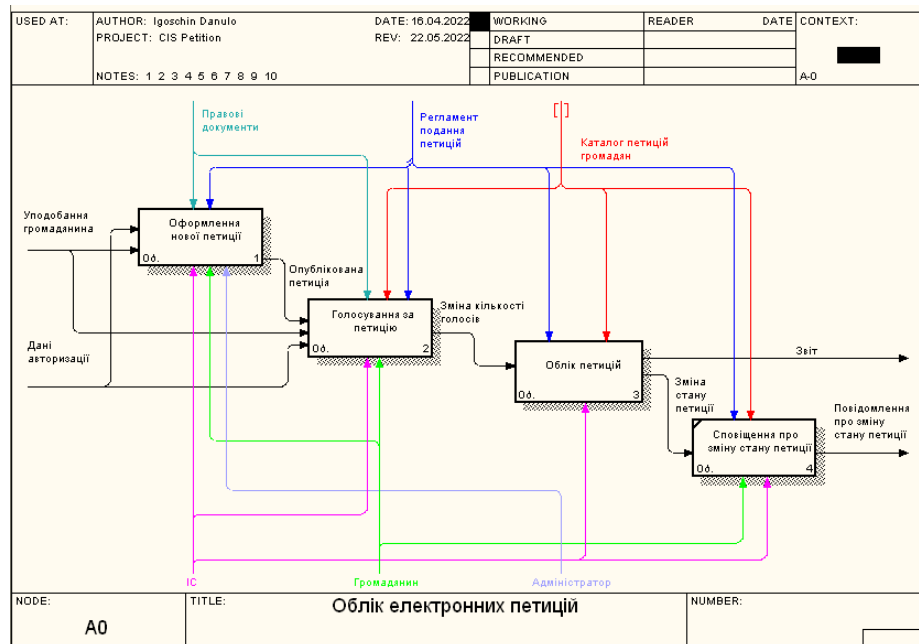


Рисунок 2.2 – Декомпозиція контекстної діаграми

Функцію «Оформлення нової петиції» розбито на п'ять підфункцій: створення петицій, внесення інформації петиції, збереження петиції у БД, перевірка петиції, публікація петиції. Функція є важливою частиною процесу. В функції за допомогою підфункцій створюється нова петиція та додається до БД,

відправляється на перевірку адміністратору. Після перевірки адміністратор може опублікувати петицію або відправити на редагування автору (рис. 2.3).

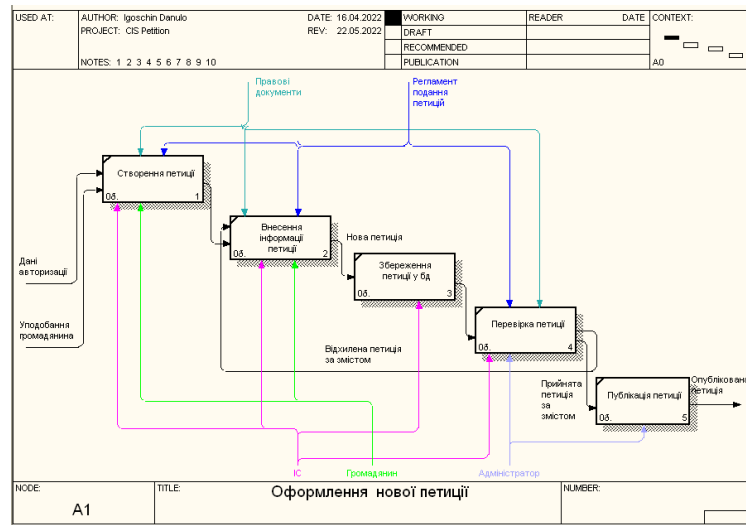


Рисунок 2.3 – Декомпозиція функції «Оформлення нової петиції» Функцію

«Голосування за петицію» розбито на чотири підфункції – авторизація громадянина, обговорення петиції, голосування за вибрану петицію, збереження петиції у БД (рис 2.4). Голосування – це важливий процес без якого облік петицій не має суті. Для голосування кожному громадянину потрібно авторизуватися, надалі громадянин матиме можливість обговорити петицію у коментарях та проголосувати. Результати голосування зберігаються у БД.

Функція «Облік петицій» розбито на чотири підфункції – підрахунок кількості голосів, відправка петиції на розгляд, збереження результату у БД, створення звіту. Процес «Облік петицій» виконує важливу функцію підрахунку голосів петицій та у разі набрання необхідної кількості присвоює петиції статус «На розгляді». Функція передбачає створення звіту по петиції та додає інформацію про зміни у петиціях та громадянах ІС електронних петицій.

Декомпозицію функції «Облік петицій» подано на рисунку 2.5.

Діаграми дерева вузлів показує ієрархію функцій моделі і дозволяє розглянути її повністю. На діаграмах вузлів немає стрілок. Для розглядання системи в цілому була побудована діаграма вузлів, яка подана на рисунку 2.6.

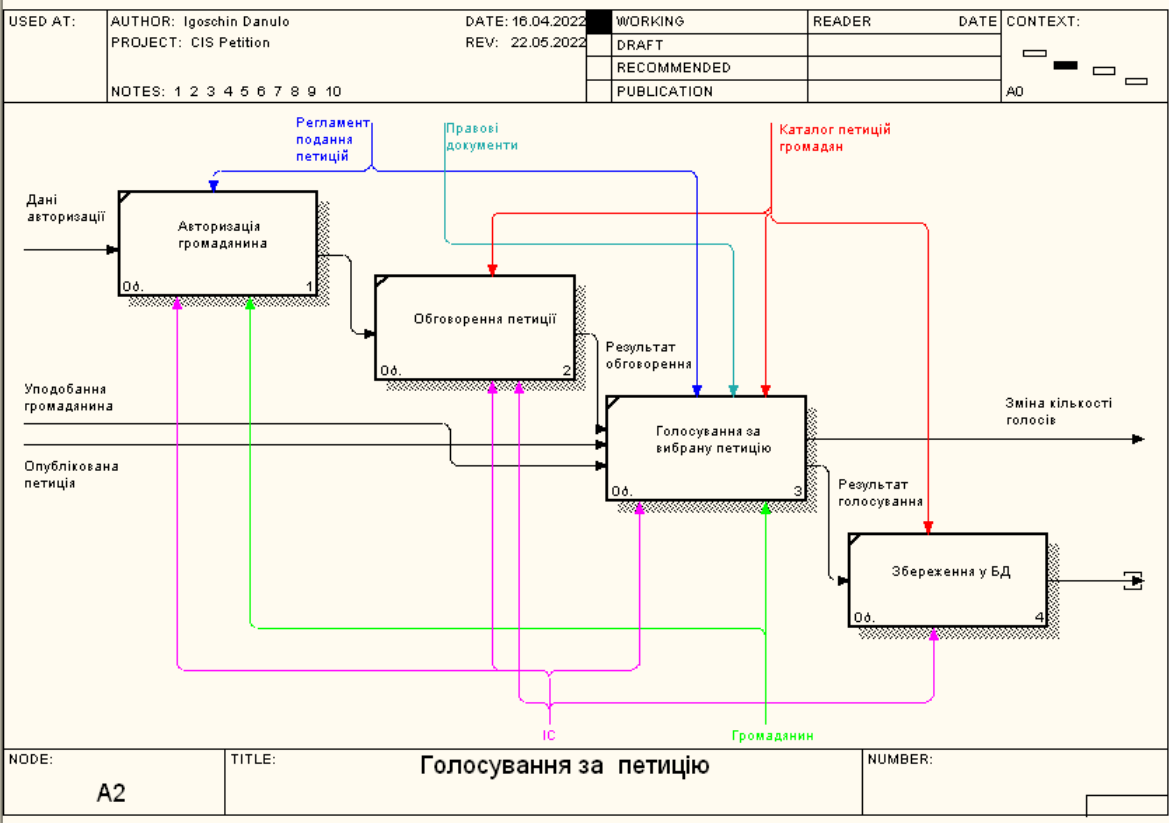


Рисунок 2.4 – Декомпозиція функції «Голосування за петицію»

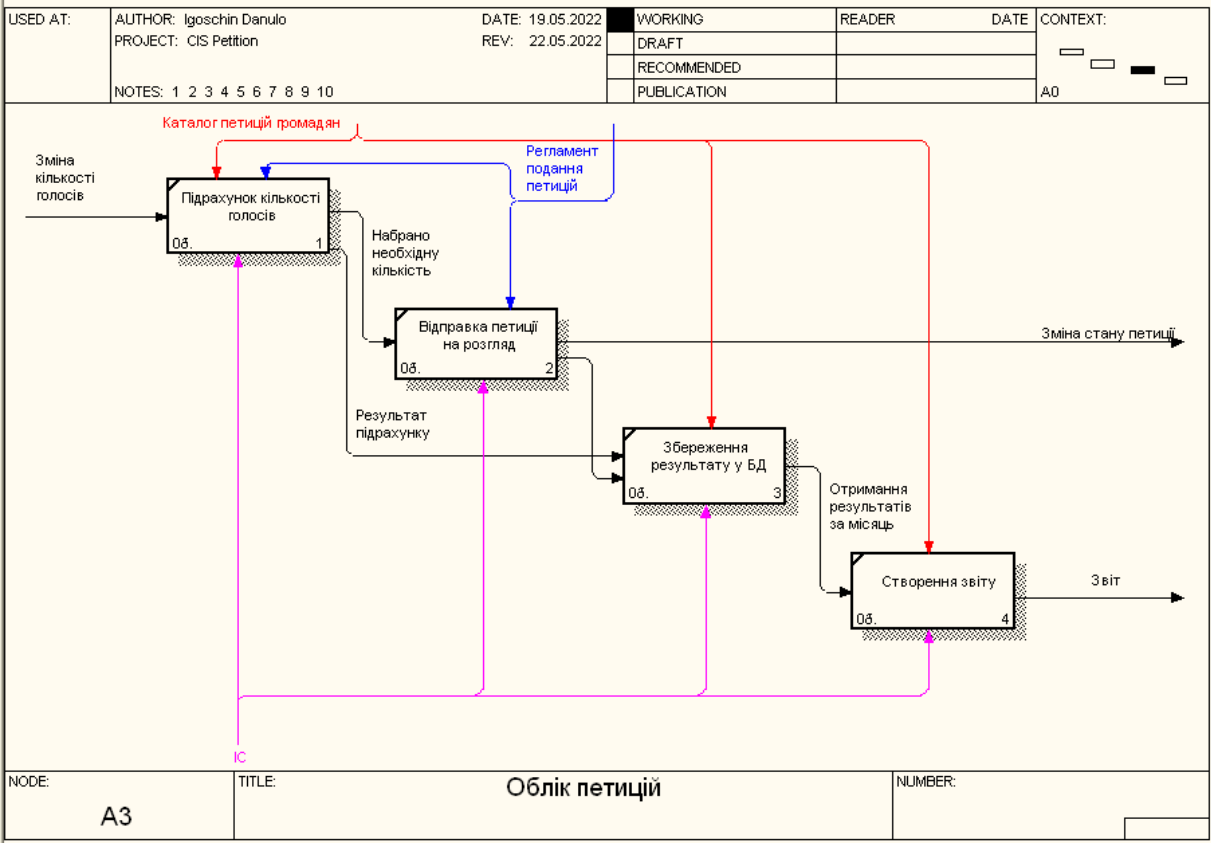


Рисунок 2.5 – Декомпозиція функції «Облік петицій»

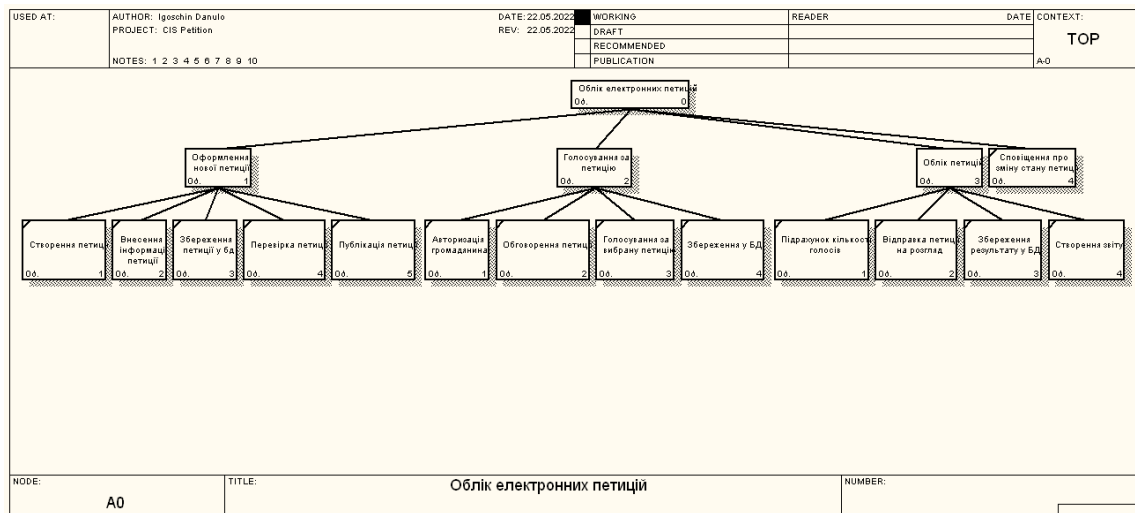


Рисунок 2.6 – Діаграма дерева вузлів

На основі функціонального моделювання вдалося уточнити такі функціональні вимоги до системи обліку електронних петицій, що розробляється:

а) адміністративна частина:

- 1) система повинна забезпечити можливість адміністратору авторизуватися для виконання своїх функцій;
- 2) система повинна надати можливість адміністратору верифікувати громадян;
- 3) система повинна надати можливість адміністратору перевіряти нові петиції;
- 4) система повинна надавати можливість адміністратору змінювати статус петицій;

б) користувацька частина:

- 1) система повинна забезпечити можливість громадянину (користувачу) авторизуватися для використання своїх можливостей;
- 2) система повинна надати можливість створювати та редагувати нові петиції громадянину;
- 3) система повинна надати можливість громадянину переглядати петиції з можливістю сортування за категоріями, статусами та кількістю голосів;
- 4) система повинна забезпечити можливість громадянину переглядати та редагувати власний профіль;
- 5) система повинна надати можливість громадянину переглядати та завантажувати звіт за останній місяць для користувачів за петиціями;

- б) система повинна надавати можливість оповіщення громадянина при зміні статусів петицій: «Потребує редагування», «На розглядані», «З відповіддю»;
- 7) система повинна надавати можливість громадянину обговорювати електронну петицію.

2.3 Розробка моделі потоків даних компонентів ІС обліку електронних петицій

Для більш наочного зображення документообігу та обробки інформації у системі проведено аналіз потоків даних за допомогою DFD. Діаграма потоків даних – це методологія графічного подання структурного аналізу, описує зовнішні за відношенням до системи джерела та адресати даних, логічні функції, потоки даних та сховища даних до яких здійснюється доступ. Діаграма потоків даних доповнює діаграму IDEF0 та показує документообіг та обробки інформації в інформаційних системах. Головна мета DFD – продемонструвати перетворення інформації у системі, як система трансформує вхідні дані у вихідні [12].

Між громадянином та системою обліку електронних петицій є декілька потоків даних, а саме дані які йдуть від громадянина – дані про авторизацію та уподобання. Від системи – повідомлення про помилку, сповіщення про зміну стану петиції та звіт за останній місяць. Описана контекстна діаграма подана на рисунку 2.7.

Процес розбито на чотири функції: оформлення нової петиції, голосування за петицію, облік петицій та сповіщення про зміну стану петиції. При розбитті запропоновано п'ять сховищ даних, а саме: БД ролі в системі, БД громадян, БД петицій, БД обговорень, БД категорій петицій.

Сховище «БД ролі в системі» надає інформацію про ролі користувачів у системі. У Сховищі «БД громадян» знаходиться уся інформація про громадян: ПІБ, номер телефону, роль громадянина в системі, електронна поштова адреса, номер телефону та документи для перевірки.

Сховище «БД петицій» зберігає усю інформацію про петиції, а саме: назву, суть звернення, дату та час, ПІБ ініціатора петиції, кількість підписів петиції.

У сховищі «БД обговорень» зберігається уся інформація про коментарі громадян, обговорення конкретної петиції за її ідентифікатором. Сховище «БД категорій петицій» зберігає усю наявну інформацію про категорії які можуть створювати громадяни. На рисунку 2.8 подано декомпозицію контекстної діаграми.

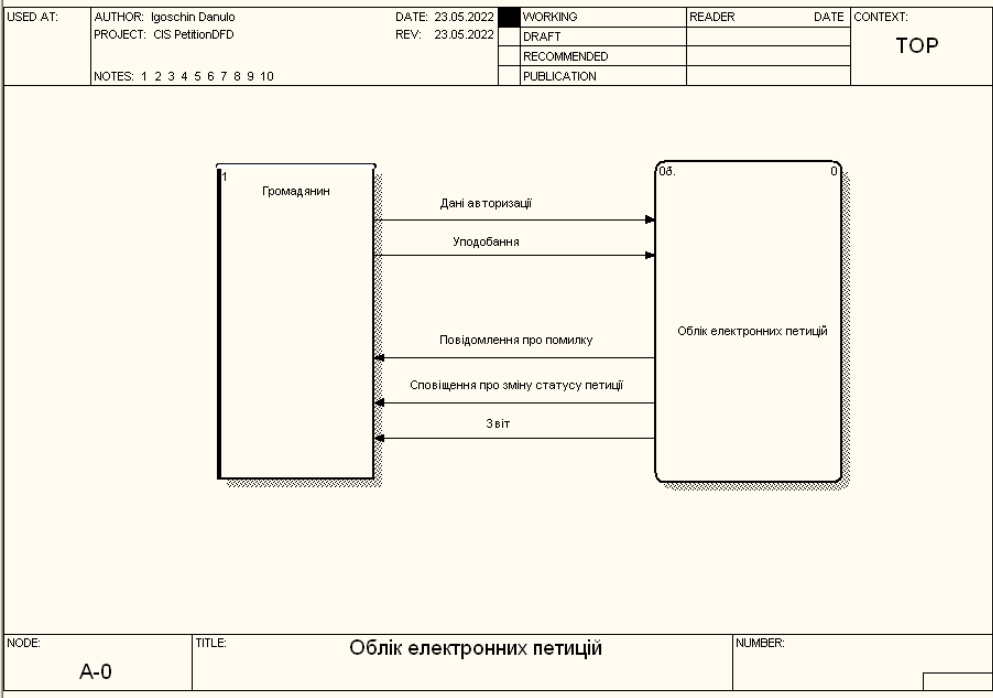


Рисунок 2.7 – Контекстна діаграма DFD

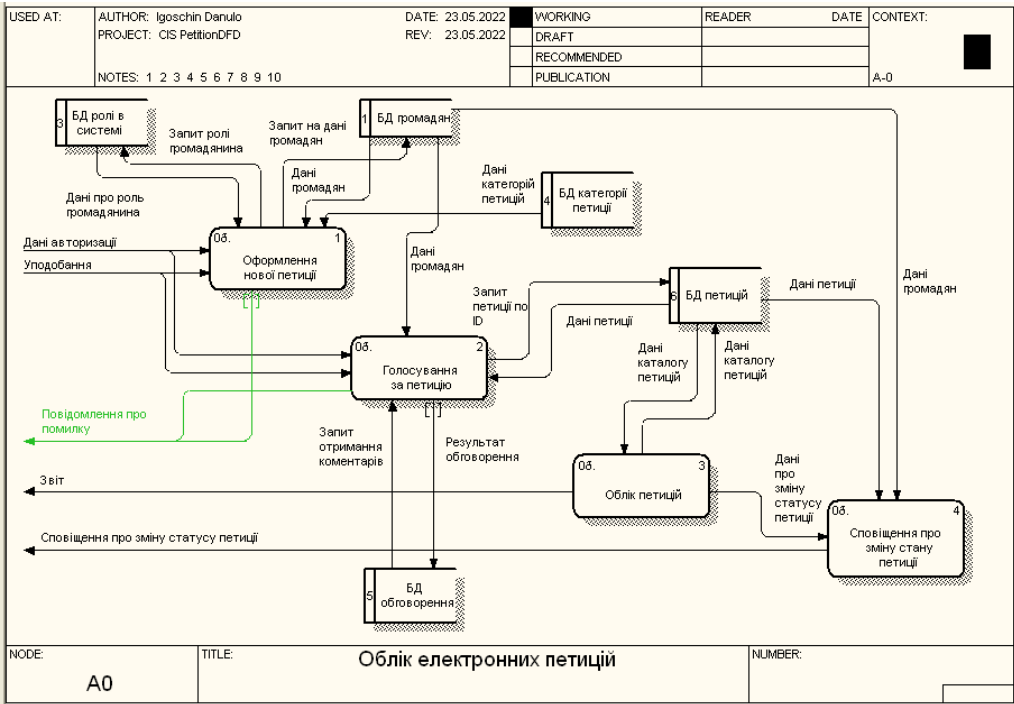


Рисунок 2.8 – Декомпозиція контекстної діаграми

Функція «Оформлення нової петиції» розбита при проектуванні на п'ять підфункцій:

- авторизація громадянина – це підфункція у якій виконується автентифікація, а вже потім авторизація. Тобто перевірка та надання прав на різні дії в ІС. Декомпозиція підфункції подано на рисунку 2.10;
 - створення нової петиції – це підфункція, яка створює макет петиції який надалі громадянин (користувач) заповнює та отримує статус «нова петиція»;
 - внесення інформації петиції – підфункція збирає введену в поля інформацію та записує до об'єкту петиції;
 - перевірка петиції, в підфункції, нова петиція переходить до адміністратора, який перевіряє петицію на предмет цензурування. Якщо в петиції інформація не суперечить правовим нормам, то адміністратор змінює статус на «Триває збір підписів», в разі не виконання норм адміністратор відправляє петицію автору на редагування. В функції статус петиції змінюється на «Потребує редагування»;
 - публікація петиції – відправка автору повідомлення про публікацію.
- На рисунку 2.9 подано декомпозицію функції «Оформлення нової петиції».

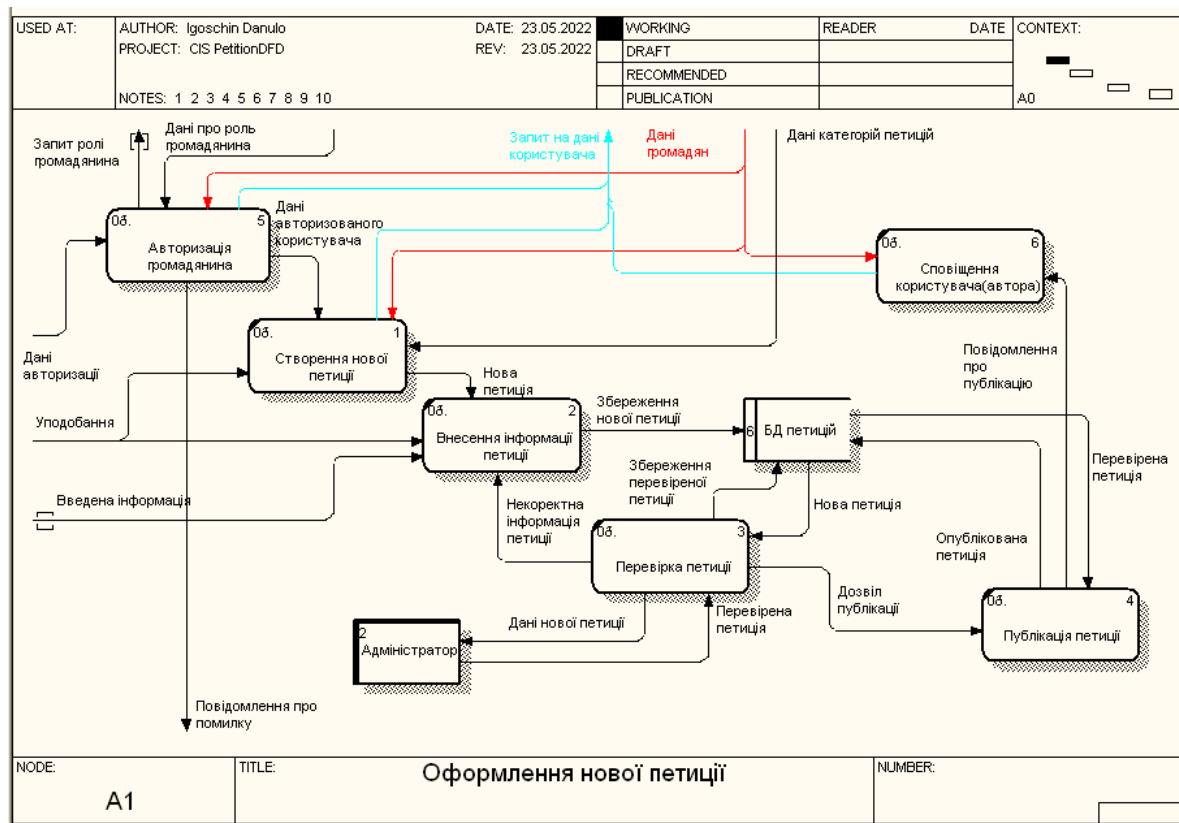


Рисунок 2.9 – Декомпозиція функції «Оформлення нової петиції»

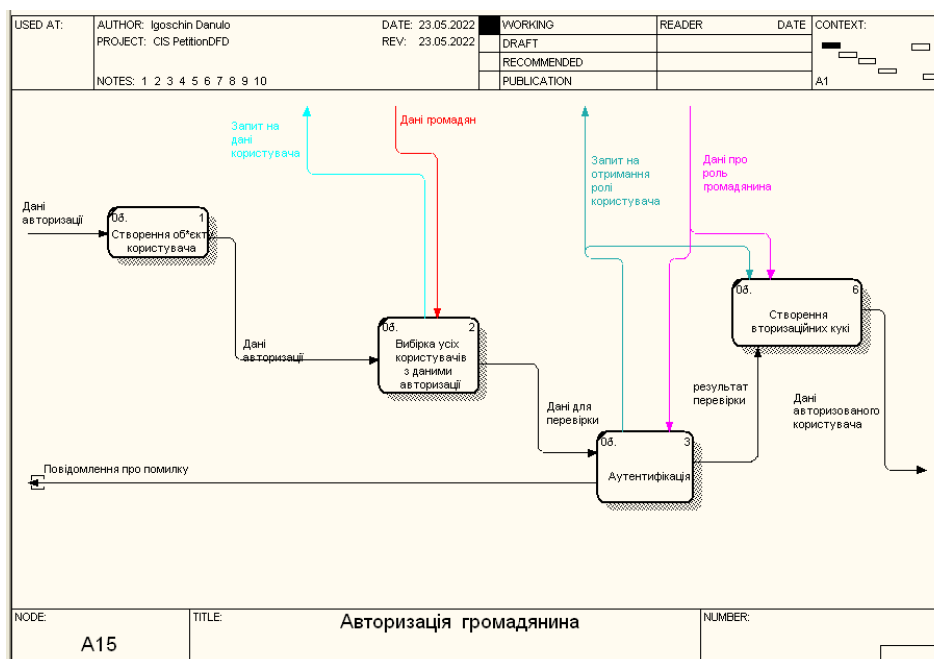


Рисунок 2.10 – Декомпозиція підфункції «Авторизація громадянина»

Декомпозицію функції пропонується розбити на дві підфункції. Підфункція «Обговорення петиції», в якій виконується завантаження коментарів які вже є та збереження нових коментарів. На рисунку 2.11 зображено декомпозицію функції «Голосування за петицію». Підфункція «Голосування» надає можливість голосувати за вибрану петицію з перевіркою на те чи голосував користувач (рис. 2.12).

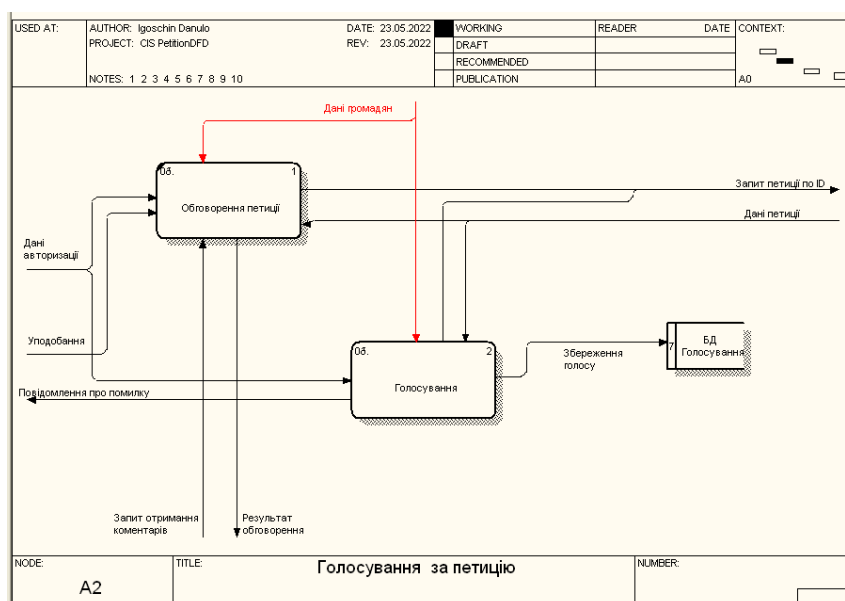


Рисунок 2.11 – Декомпозиція підфункції «Голосування за петицію»

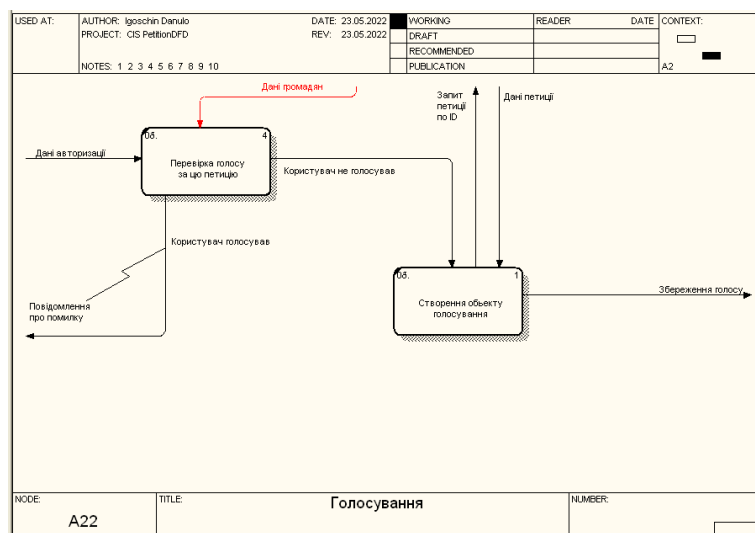


Рисунок 2.12 – Декомпозиція підфункції «Голосування»

Функція виконує підрахунок кількості голосів за конкретною петицією та якщо набрана необхідна кількість, то змінює статус «На розгляді». Також за допомогою функцію створюється звіт за петиціями та громадянами за останній місяць. На рисунку 2.13 подано декомпозицію функції «Облік петицій».

Алгоритм функції полягає у автоматичному створенні тексту повідомлення, відправки у особистий кабінет та на електронну пошту автору. Функція «Сповіщення про зміну стану петиції» подана на рисунку 2.14.

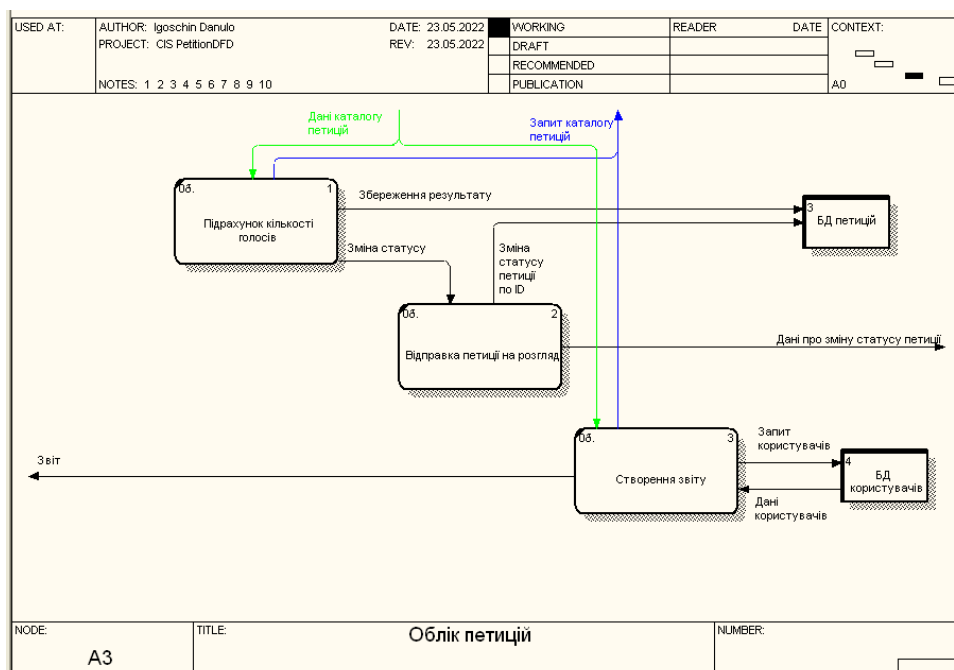


Рисунок 2.13 – Декомпозиція функції «Облік петицій»

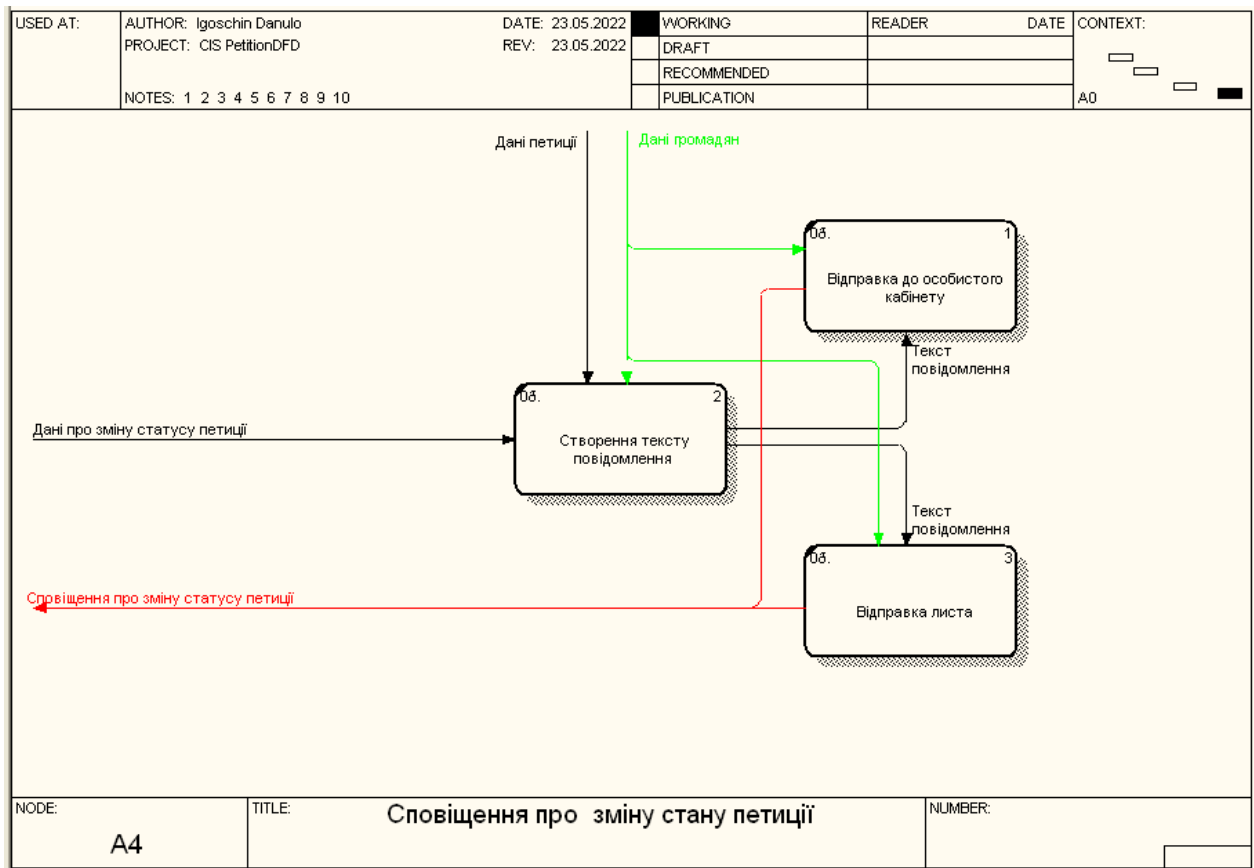


Рисунок 2.14 – Декомпозиція функції «Сповіщення про зміну стану петиції»

2.4 Діаграма варіантів використання системи обліку електронних петицій

Діаграма варіантів використання моделює поведінку системи та допомагає охопити вимоги системи. Діаграма варіантів використання описує функції високого рівня та області застосування системи. Також визначає взаємодію між системою та акторами. Варіанти використання та дійові особи на діаграмах варіантів використання описують, що робить система та як учасники її використовують. Та вона не описує як система працює всередині [13].

Діаграма прецедентів ілюструє і визначає контекст і вимоги всієї системи, або важливих частин системи. За допомогою діаграми можна змоделювати складну систему за допомогою однієї діаграми або створити багато діаграм використання для моделювання компонентів системи.

В ІС обліку електронних петицій є три ролі: гість, громадянин, адміністратор.

Адміністратор – це людина, яка має повний доступ до системи та в обов’язки якої входить верифікація користувачів, перевірка та публікація нових петицій. На рисунку 2.15 подано діаграму варіантів використання ІС обліку електронних петицій міської ради.

Варіант «Реєстрація» дозволяє гостю створити особистий кабінет, щоб надалі використовувати систему в повному обсязі. Тобто стати громадянином (користувачем).

Варіант «Виконати пошук петицій» – прецедент, що передбачає пошук різних петицій з каталогу наявних петицій у БД. Прецедент має три включення:

- пошук за статусом петиції – сортування петицій за статусом: триває збір підписів, на розгляданні, з відповіддю та архів;
- пошук за кількістю голосів – сортування за більшою чи меншою кількістю голосів громадян;
- пошук за категоріями – сортування петицій за категоріями: благоустрій, житлово-комунальне господарство тощо.

Варіант «Продивитися перелік громадян, що підписали» – прецедент, що передбачає проглядання переліку тих хто підписав конкретну електронну петицію.

Варіант «Переглядати статистику» – прецедент, що передбачає проглядання статистики веб-застосунку за петиціями за останній місяць.

Варіант використання «Звантажити PDF-файл статистики за місяць» – прецедент передбачає викачування з веб-застосунку сформований файл зі статистикою про перелік нових петицій, кількості нових громадян, петицій на розгляданні, з відповіддю, виконаних петицій.

Варіант «Обговорення петиції» – прецедент, що надає можливість громадянам задавати автору петиції запитання та обговорювати петицію між собою у коментарях.

Варіант використання «Голосувати» відповідає за голосування, можливість голосування мають лише авторизовані громадяни. Прецедент залежний від авторизації та має розширення «Вже проголосована». Якщо громадянин вже проголосував він не має можливості голосувати знову.

Варіант «Створити петицію» відповідає за створення нової петиції та подання її на розгляд адміністратору для публікації або редагування автором.

Варіант «Верифікація громадян» – прецедент, що надає можливість адміністратору продивитись документ який підтверджує, що людина є громадянином міста та громадянином України.

Варіант використання «Авторизація» – прецедент, що надає можливість громадянам заходити до особистого кабінету та надає право доступу до варіантів використання: голосувати, обговорювати петицію, створювати петицію, продивлятися профіль. Прецедент має два розширення («Дійсний пароль», «Недійсний пароль»). Розширення «Дійсний пароль» – це ситуація коли все введено вірно та такий обліковий запис є в системі. Розширення «Недійсний пароль» відповідає за варіант використання коли облікового запису немає або дані введено невірно. Без авторизації громадянин не зможе голосувати, створювати петиції та обговорювати петиції у коментарях, тому від авторизації залежать прецеденти: «Обговорення петиції», «Створити петицію».

Варіант «Продивитися профіль» надає можливість громадянину зайти до особистого кабінету та продивитися власну інформацію, яка була введена при реєстрації. Варіант використання має включення «Редагування профілю», користувач хоче та має потребу змінити власні дані.

Варіант використання «Перевірити петицію» надає можливість адміністратору продивитися створені петиції громадян зі статусом «Нова петиція» на предмет забороненої інформації. Варіант використання включає варіант «Опублікувати петицію», тобто змінити статус на «Триває збір підписів». Варіант використання має розширення «Відправити на редагування» автору петиції, для виправлення забороненої інформації. Варіант включення «Опублікувати петицію» включає в себе прецедент «Повідомити автора».

2.5 Діаграми класів системи обліку електронних петицій

Діаграма класів – це статичне відображення структури системи, відображає основні сутності та також типи даних, зміст та відношення. Класи характеризуються за допомогою атрибутів і операцій [14]. Діаграму класів наведено на рисунку 2.16. У системі представлені такі класи:

- клас «User» відображає дані про користувачів після витягування даних з БД та використовується для реєстрації нового користувача та авторизації;

- клас «Petition» відображає дані про петиції;
- клас «Vote» – це клас голосування, який не може функціонувати без класів: користувач «User» та петиція «Petition». Клас використовується для створення об'єкту голосування, в якому один об'єкт класу «User» та один об'єкт класу «Petition»;
- клас «Category» відповідає за відображення даних про категорії електронних петицій та можливості надавання інформації про категорії при створенні петиції громадянином;
- клас «Role» – клас відображає інформацію про ролі користувачів в системі;

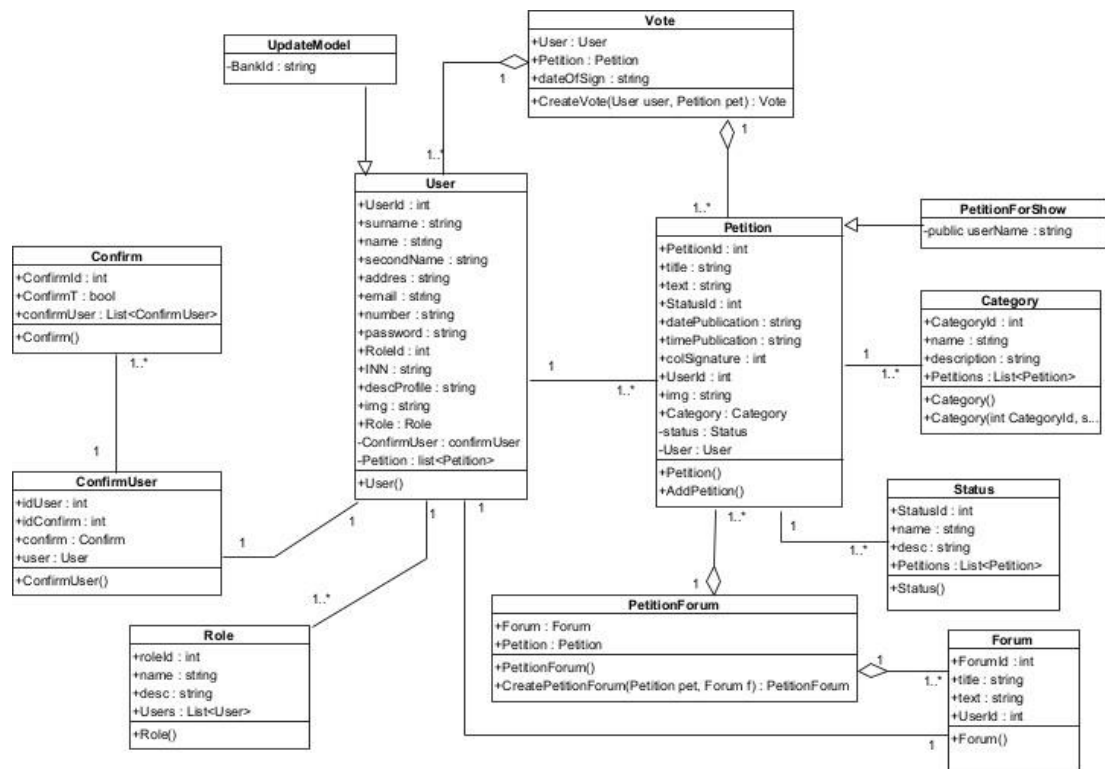


Рисунок 2.16 – Діаграма класів Frontend

- клас «confirm» відображає інформацію про документи які представив користувач для верифікації як громадянин міста;
- клас «ConfirmUser» відображає дані про кількість документів та кількість разів верифікації;
- клас «Status» – це клас що відповідає за відображення інформації про статус петиції у системі;

- клас «UpdateModel» використовується для перевірки та відображення введених даних на сторінці та передачі у контролер;
- клас «Forum» відображає коментар користувача;
- клас «PetitionForum» відображає інформацію про користувача та коментар під конкретною петицією;
- клас «PetitionFoShow» використовується для виведення інформації про петиції на сайті.

На рисунку 2.17 подано діаграма класів взаємодії з PetitionController.

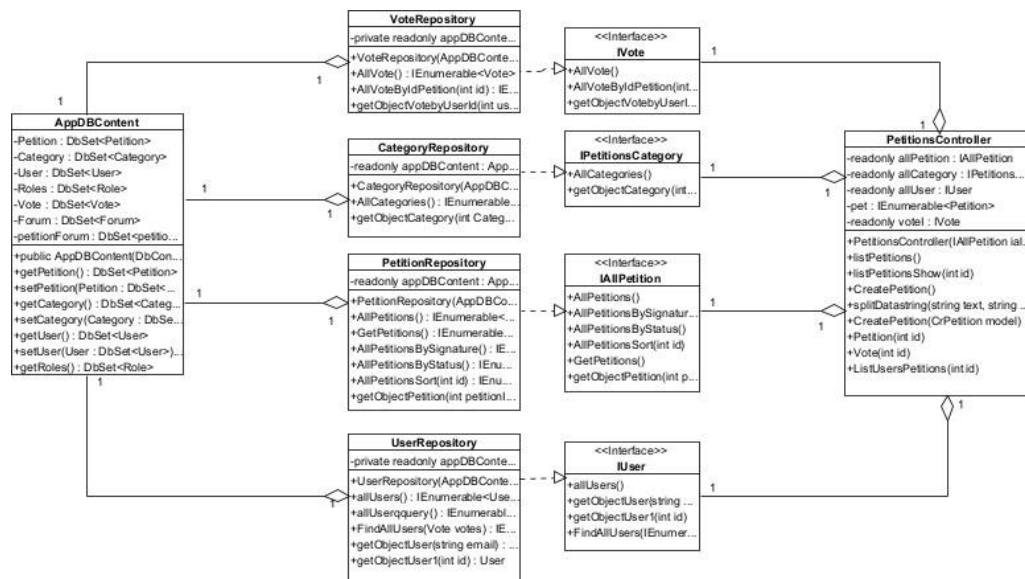


Рисунок 2.17 – Діаграма класів Backend

Клас «AppDBcontent» взаємодіє з БД, використовує для цього dataset. Dataset – це контейнер який є сховищем для будь-якої кількості об’єктів DataTable (таблиць БД). Об’єкт адаптера для конкретного постачальника даних автоматично обслуговує підключення до бази даних. Клас «PetitionController» – це контролер який відповідає за обробку запитів на веб-сторінці петицій. Інтерфейси «IVote (Голосування)», «IPetitionCategory (Категорії петицій)», «IAllPetition (Петиції)» та «IUser (Користувач)» мають методи для роботи з БД. Класи «VoteRepository (Репозиторій голосування)», «CategoryRepository (Репозиторій категорій петицій)», «PetitionRepository (Репозиторій петицій)», «UserRepository (Репозиторій користувачів)» реалізують вище перераховані інтерфейси.

2.6 Діаграма послідовності прецеденту «Створити петицію»

Діаграма послідовностей в контексті UML – це співпраця об'єктів, використовується для визначення послідовностей подій між об'єктами для певного результату. Діаграма послідовностей є важливою складовою, яка використовується в процесах пов'язаних з аналізом, проектування та документацією. Діаграма послідовностей також відома як діаграма часу, діаграма подій та сценарій подій [15].

На ній зображена послідовність дій, після того, як громадянин на сторінці у браузері натисне кнопку меню – «Створити петицію». Спочатку громадянин перевіряється контролером облікових записів (AccountController) на авторизацію громадянина в системі. Після управління передається контролеру петицій (PetitionController), який ініціює функцію створення петиції (CreatePetition). Громадянин заповнює форму з полями для подання петиції. На рисунку 2.18 представлена діаграма послідовності для уточнення прецеденту «Створити петицію».

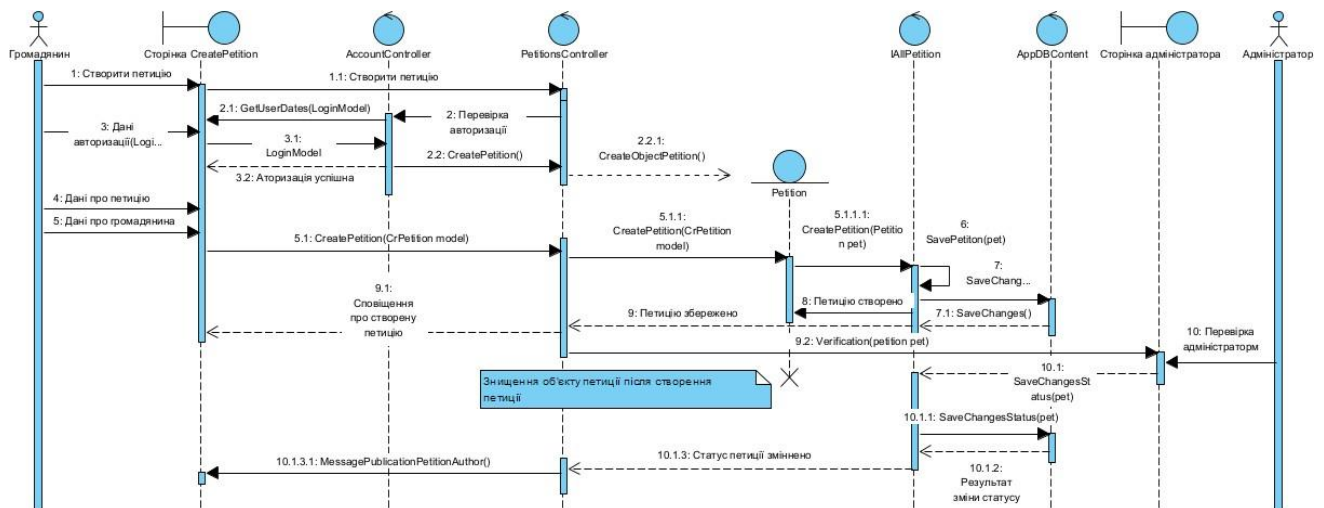


Рисунок 2.18 – Діаграма послідовності прецеденту «Створити петицію»

Форма відправляється до контролера петицій. Далі створюється об'єкт сутності петиція (Petition) в яку записуються дані для збереження у БД. За збереження у БД відповідає функція «SavePetition» з параметром об'єкта петиції.

Викликається функція репозиторію `PetitionRepository`, який відповідає за зберігання та збереження петицій. `AppDBContent` – це клас для роботи з БД. Після зберігання контролер петицій отримує результат збереження та відправляє петицію на перевірку адміністратору. Результат перевірки відправляється у контролер та надалі сповіщається громадянин, спочатку про вдало створену петицію, а потім про публікацію петиції. На рисунку 2.19 подана діаграма послідовності для функції «Авторизація».

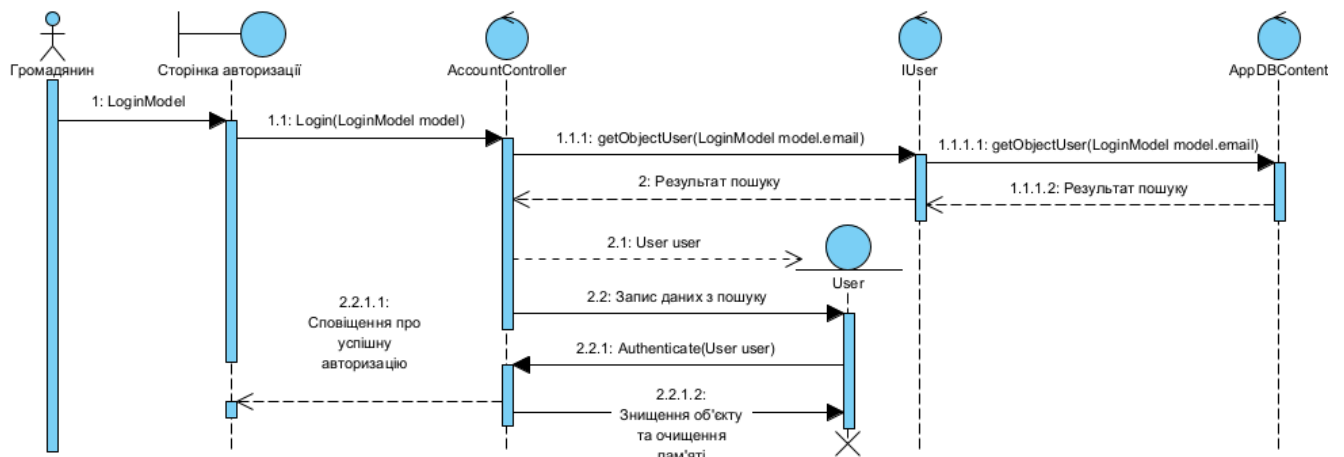


Рисунок 2.19 – Діаграма послідовності прецеденту «Авторизація» Функція

після введення інформація громадянином у відповідні поля моделі, яка має назву «`LoginModel`» передає інформацію до контролеру. Далі контролер облікових записів (`AccountController`) отримує дані з БД за логіном громадянина, створює об'єкт користувача та зберігає інформацію в цей об'єкт. Викликається функція `Authenticate` з об'єктом який буде перевірятися. Після перевірки громадянин отримає повідомлення про вдалу авторизацію.

2.7 Діаграма кооперації функції «Створити петицію»

Діаграма комунікації моделює взаємодії між об'єктами або частинами термінів упорядкованих повідомлень. Комунікаційні діаграми представляють комбінацію інформації, взятої з діаграм класів, послідовності та варіантів використання, описуючи одночасно статичну структуру і динамічну поведінку

системи [16]. На рисунку 2.20 подано діаграму кооперацій функції «Оформлення нової петиції».

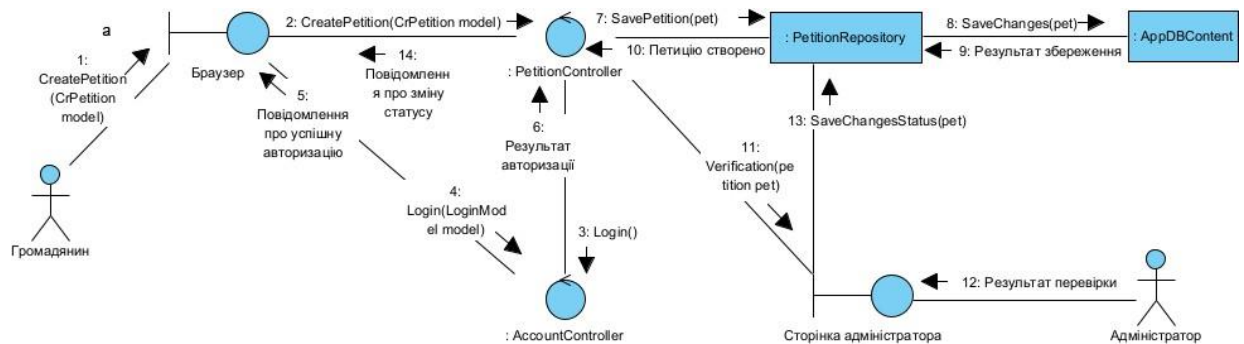


Рисунок 2.20 – Діаграма кооперації функції «Оформлення нової петиції»

2.8 Діаграма станів об'єкту «Петиція»

Діаграма станів показує поведінку, всі можливі стани одного примірника певного класу та можливі переходи об'єкта з одного стану в інший при певній послідовності подій. Діаграма станів найчастіше використовуються для опису поведінки окремих об'єктів, але можуть використовуватись для опису зміни стану функцій, підфункцій, варіантів використання та акторів [17]. На рисунку 2.21 подано діаграму станів об'єкта петиції.

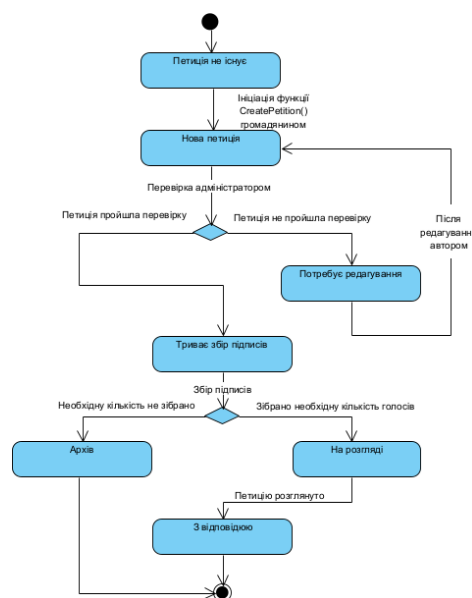


Рисунок 2.21 – Діаграма зміну станів об'єкта петиція в системі

На діаграмі представлена зміна стану петиції з того моменту коли вона ще не існує в системі. Коли користувач створює петицію викликається функція create Petition та після заповнення петиції громадянином система змінює її статус на «Нова петиція». Адміністратор перевіряє петицію та змінює статус на «Триває збір підписів», якщо петиція не відповідає вимогам на «Потребує редагування». Після редагування громадянином петиція повертається на перевірку адміністратору зі статусом «Нова петиція».

Якщо петиція набрала необхідну кількість голосів, стан петиції змінюється ІС на «На розгляді» та відправляється на розгляд до відповідного органу місцевого самоврядування. Петиції які не набрали необхідну кількість голосів відправляються до архіву та отримують статус «Архів». Після розглядання петиція отримує статус «З відповіддю», знайти її можна у відповідному розділі.

2.9 Діаграма діяльності системи обліку електронних петицій

Діаграма діяльності в UML – візуальне представлення графу діяльності. При моделюванні системи виникає необхідність не тільки представити процес зміни станів, а також промодельовати та деталізувати особливості алгоритмічної та логічної реалізації.

Кожна діаграма послідовності звертає увагу на послідовному виконанні дій для досягнення певного результату. Діаграму будують для цілої системи або для окремих її частин, методів та варіантів використання. Графічна нотація за допомогою якої виконується побудування даної діаграми багато в чому схожа на діаграму станів. Діаграма діяльності є різновидом діаграми станів, але якщо на діаграмі станів увага приділяється статичності станів, то на другій саме діям [18].

Діаграма діяльності функції «Створити петицію» зображена на рисунку 2.22. На даній діаграмі зображено процес створення нової петиції. Вона проходить через декілька етапів, від авторизації громадянина до публікації петиції адміністратором. В кінці процесу виконується сповіщення автора петиції.

Діаграма представляє як проводиться голосування за петицію в логічному та алгоритмічному плані. Громадянин після перегляду петиції може перейти до обговорення або одразу до голосування. Громадянин який голосував вже за цю петицію не зможе проголосувати знову. В першому та у другому випадку коли громадянин вдало проголосував він отримає оповіщення, але з різним текстом повідомлення. На рисунку 2.23 подано діаграму діяльності функції «Голосування за петицію».

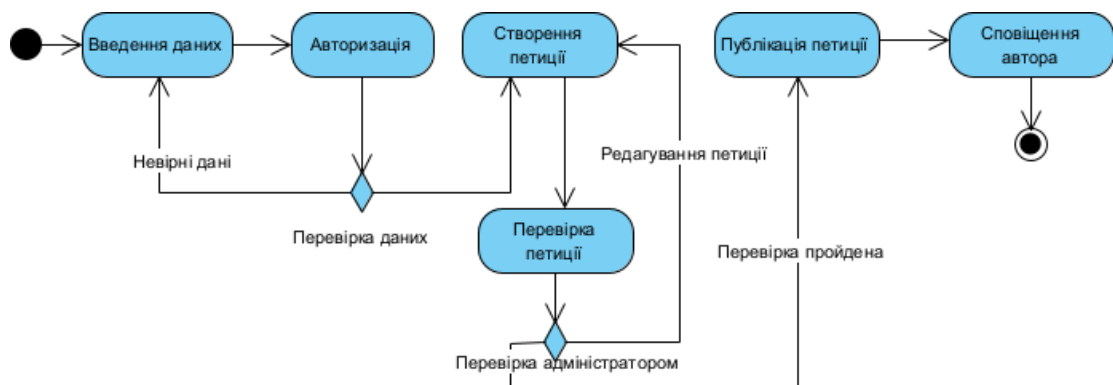


Рисунок 2.22 – Діаграма діяльності функції «Створити петицію»

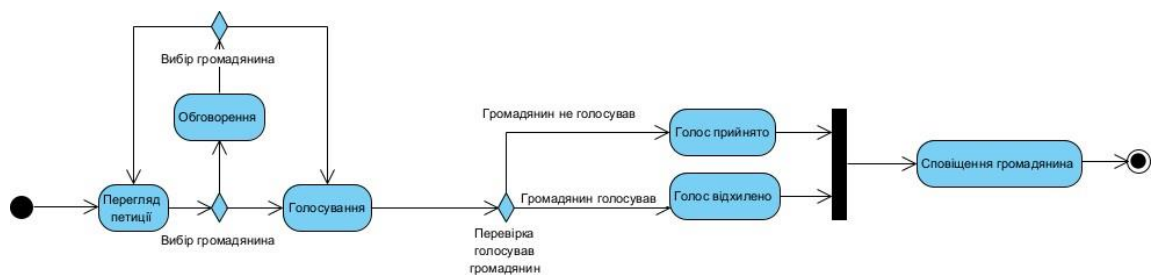


Рисунок 2.23 – Діаграма діяльності функції «Голосування за петицію»

Отже, в результаті проведення об'єктно-орієнтованого та функціонального моделювання, розгляду основних особливостей предметної області та головного процесу дозволило виявити та сформулювати такі вимоги до компонентів системи, які підлягають розробці:

- база даних, що надасть можливість обробляти та зберігати електронні петиції, облікові дані про громадян. Надає можливість виконувати реєстрацію та авторизацію громадян за даними, які зберігаються у БД;

- веб-елементи інтерфейсу, які забезпечують відображення списку петицій, коментарів та перелік форм для створення петицій та відображення інформації;
- серверна частина, що надасть можливість громадянам створювати, обговорювати та голосувати за петиції. Надасть можливість реєстрації та авторизації користувачів;
- блок захисту інформації, що надасть можливість захистити конфіденційну інформацію користувачів.

3 ОПИС ПРИЙНЯТИХ ПРОЕКТНИХ РІШЕНЬ

3.1 Обґрунтування вибору мови програмування та середовища розробки Для розробки веб застосунку ІС обліку електронних петицій була обрана мова програмування C#. C# – це мова в основі якої лежить поняття об'єкту, об'єктно-орієнтована з можливостями для розробки Desktop програм, створення складних веб-сервісів та мобільних додатків [19].

Мова належить до сімейства С-подібних мов, до яких відносять C++ та Java. C# має статичну типізацію, підтримує поліморфізм, делегати, події, перевантаження операторів, ітератори, анонімні функції, LINQ (мова запитів до джерела даних). При розробці мови було усунуто невдалі моделі попередників. Наприклад: C# на відміну від C++ не підтримує множинне успадкування класів, між тим допускається множинна реалізація інтерфейсів. Мова C# розроблялась для прикладного рівня CLR та залежить від можливостей самої CLR.

CLR (Common Language Runtime) – це виконуюче середовище для байт-коду (CIL), в якому компілюються програми, написані на .NET-сумісних мовах програмування (C#, .NET, F# тощо). CLR є одним з основних компонентів пакета Microsoft .NET Framework, який використовується для взаємодії з БД.

Інструментарій мови C# дозволяє вирішувати велике коло завдань: програми для веб-розробки, різні ігрові додатки, програмна платформа під Android та iOS, програми для операційної системи Windows та Linux. Платформа активно розвивається та щомісяця розробники покращують мову новими оновленнями.

Платформа у своєму складі має багато переваг:

- автоматичне складання сміття. Дозволяє не звертати увагу на витік пам'яті в більшості завдань та надає змогу зосередитись на інших аспектах програмування;
- багатоплатформність – можливість працювати більш ніж чим на одній програмній або апаратній платформі;
- асинхронна модель коду – async / await. Надає змогу писати продуктивний код, розбитий на різні потоки. Покращує пропускну здатність серверу;

- asp.net MVC – фреймворк для розробки веб-застосунків. Надає велику кількість готового функціоналу (авторизація, фільтри) та є досить модульним для того, щоб будь-яку частину фреймворку замінити на власну реалізацію;
- відкритий вихідний код, пакетний менеджер NuGet, надає змогу додавати необхідні модулі для розробки програми;
- різноманітність мов, що компілюються під платформу між собою – C#, F#, IronPython.

Але платформа має і недоліки:

- багатослівність мови. Кожна нова версія вносить не тільки зміни та й недоліки. Якщо порівнювати такі ж конструкції з F#, мова C# буде дуже багатослівною та нагромаджена об'ємними конструкціями;
- мала кількість інтеграцій з продуктами для обробки даних;
- перехід зі старих типів програм на нові може бути непростим.

В якості середовища розробки обрано Visual Studio 2019 Community. Microsoft Visual Studio – інтегроване середовище розробки програмного забезпечення, яке має ряд інших додаткових засобів. Дане середовище надає можливість розробляти консольні додатки, програми з графічним інтерфейсом за допомогою Windows Forms та нової технології WPF (Windows Presentation Forms), веб-застосунки, веб служби.

3.2 Обґрунтування використаних технологій

Для створення компонентів ІС обліку електронних петицій використовувались: технологія ASP.NET CORE, мова програмування C#, EntityFramework для роботи з БД та архітектурний шаблон MVC (Model-View-Controller).

Для створення користувацького інтерфейсу використовувались: мова гіпертексту HTML, каскадні таблиці стилів – CSS, Razor pages (сторінки, якіможуть обробляти запити), фреймворк Bootstrap та мова програмування JavaScript для створення анімації та динамічного контенту на сторінках.

3.2.1 Архітектура ASP.NET

Технологія ASP.NET створена компанією Майкрософт та використовується для розробки різного роду веб-застосунків: від невеликих веб-сайтів до великих веб-порталів [20].

ASP.NET – це платформа для створення веб-застосунків та веб-сервісів, що працюють під керуванням Internet Information Services (IIS). На сьогодні існують інші різноманітні технології (наприклад, до них відносяться мови PHP та PERL), що дозволяють створювати веб-програми. Однак ASP.NET відрізняється від інших, високим ступенем інтеграції з сервісними продуктами, та інструментами Майкрософт для розробки доступу до даних і забезпечення безпеки. ASP.NET дозволяє розробляти як веб- та Windows програми, використовуючи дуже схожі технологічні ланцюги, однакові мови програмування та технології доступу до даних. Мова за допомогою якої проводиться розробка веб-застосунка, є повністю об'єктно-орієнтованою, що робить розробку набагато простіше.

Однією з переваг можна відзначити модульність платформи завдяки менеджеру пакетів NuGet. ASP.NET містить фреймворк MVC (Model-View-Controller) який поєднує в собі функціональність MVC, web-API й web-pages, та надає можливості для розробки веб-застосунків.

ASP.NET характеризується розширюваністю. Фреймворк побудований із набору незалежних компонентів. Є можливість використати вбудовану реалізацію цих компонентів, або розширити їх за допомогою механізму спадкування, або створити та застосовувати свої компоненти зі своїм функціоналом.

ASP.NET має такі переваги:

- сучасний за стандартами, модульний конвеєр HTTP-запитів;
- можливість розгорнути програму як на сервері IIS, так і в рамках свого власного процесу;
- можливість додати потрібний модуль у програму за допомогою пакетного менеджера NuGet;
- єдиний стек веб-розробки, що поєднує в собі Web-UI та Web-API;
- конфігурація для використання у хмарному сховищі;

- вбудована підтримка для запровадження залежностей при розширенні веб-застосунку;
- багатоплатформність – можливість розробки та розгортання додатків Asp.Net на Windows, Mac та Linux операційних системах.

3.2.2 Використання фреймворку Bootstrap

Для розроблення зручного та якісного інтерфейсу використано фреймворк Bootstrap – це відкритий та безкоштовний HTML, CSS та JS фреймворк, який використовується веб-розробниками для швидкої адаптивної верстки дизайну сайтів та веб-застосунків [21]. Фреймворк являє собою набір CSS та JavaScript файлів. Щоб використовувати файли фреймворку, необхідно під'єднати файли до сторінки. Фреймворк надає інструменти – система колонок (сітка Bootstrap), класи та компоненти.

Bootstrap складається з таких компонентів:

- інструментів для створення макету (обгорткових контейнерів, системи сіток, гнучких медіа-об'єктів, адаптивних класів);
- класів для стилізації базового контенту: тексту, зображень, коду, таблиць;
- готових компонентів: кнопок, форм, горизонтальних і вертикальних навігаційних панелей, слайдерів, списків, акордеонів, модальних вікон, спливаючих підказок тощо;
- класів для вирішення традиційних завдань, що найбільш часто виникають перед веб-розробниками: вирівнювання тексту та зображень, приховування елементів, завдання кольору фону, зовнішніх та внутрішніх відступів тощо.

До переваг фреймворку серед інших схожих за призначенням відносяться:

- висока швидкість створення якісної адаптивної верстки сайту;
- багатобраузерність та багатоплатформність – коректний показ та робота сайту у всіх підтримуваних цим фреймворком браузерах (Chrome, OperaGX, Mozilla Firefox, Safari тощо) та операційних системах (Windows, IOS, Linux, Android);
- наявність великої кількості готових компонентів, протестованих величезним товариством веб-розробників на різних пристроях;

- можливість налаштування під свій проект, досягається за допомогою CSS змінних.

Фреймворк Bootstrap як і інші технології має не лише переваги та й недоліки, а саме:

- великий розмір кінцевих CSS та JS-файлів проекту, на відміну від файлів написаних власноруч. Це пов'язано з тим, що стилі фреймворку та його JS-код містять універсальний код, та за фактом конкретного проекту з цього може знадобитися лише частина;

- складність використання Bootstrap для верстки сайтів з унікальним дизайном, технологія в цьому випадку супроводжується значною зміною коду.

3.3 Обґрунтування вибору СУБД

Для створення веб-застосунку ІС обліку електронних петицій міської ради була обрана СУБД Microsoft SQL Server 2019. Microsoft SQL Server – це сервер, головна задача якого збереження та за допомогою запитів SQL надання даних, які можуть виконуватись на тому ж сервері або у мережі Інтернет [22].

СУБД Microsoft SQL Server для запитів використовує мову – Transact-SQL (T-SQL). Мова запитів дозволяє виконувати операції над даними: додавання, оновлення та видалення даних. T-SQL дозволяє використовувати розширений опис процедур, що зберігаються та забезпечують підтримку транзакцій.

Microsoft SQL Server має підтримку Open Database Connectivity (ODBC) – інтерфейс взаємодії застосунків з БД. Це дозволяє клієнтським програмам з'єднуватись з SQL Server. SQL Server підтримує віддзеркалювання та створення кластерів даних. Кластер серверу SQL – це термін, який використовується для опису сукупності двох або більше фізичних серверів (вузлів), підключених через локальну мережу. Кожен з серверів містить екземпляр серверу SQL та має однаковий доступ до спільного сховища.

SQL Server можна характеризувати такими особливостями:

- продуктивність, SQL Server працює дуже швидко на відміну від MySQL;
- надійність та безпека, SQL Server надає можливість шифрувати дані;

– з СУБД легко працювати та вести адміністрування.

SQL Server має вбудовану підтримку .NET Framework. Microsoft SQL Server 2000 міцно сидить у серці .NET Framework як основний постачальник даних для цієї платформи. Сто відсотків потужності SQL Server можна використовувати в будь-якому розробленому додатку .NET, а також у будь-якій зовнішній системі.

Завдяки такій підтримці та тому що, технології працюють разом, процедури БД, що зберігаються можуть бути написані під платформу .NET Framework. На відміну від інших процесів, .NET Framework виділяє додаткову пам'ять і будує засоби керування SQL Server, не використовуючи вбудовані засоби Windows.

3.4 Опис структури розробленої системи обліку електронних петицій

Веб-застосунок побудований на триланковій клієнт-серверній архітектурі. Особливість програм на цій архітектурі – веб-додаток перебуває і обробляється на самому сервері, клієнт в свою чергу отримує вже тільки готові результати. Робота ґрунтується на отриманні запиту від користувача, обробки запитів та видачі результату за ними. Передача запитів та результатів здійснюється за допомогою мережі «Інтернет» [23]. На рисунку 3.1 подана триланкова клієнт-серверна архітектура на основі Internet.

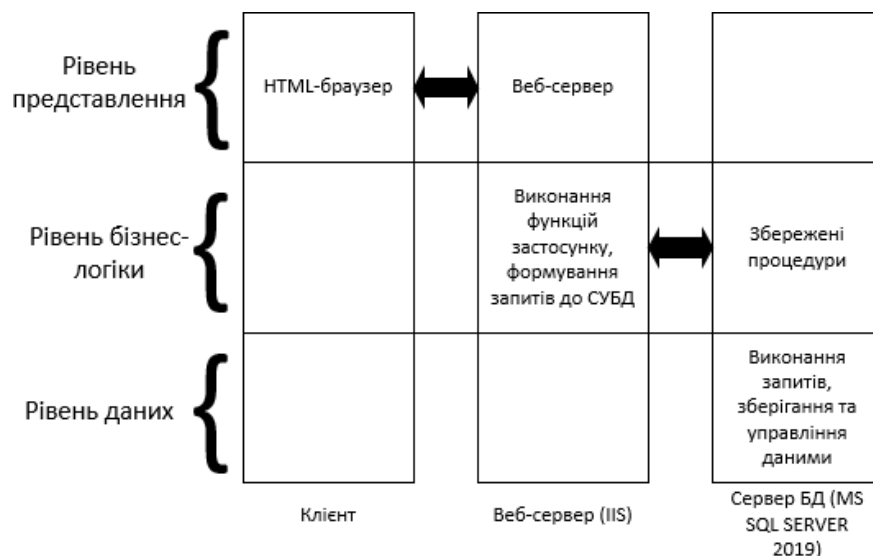


Рисунок 3.1 – Триланкова клієнт-серверна архітектура на основі Internet

Відправкою результатів запитів, а також прийняттям даних від клієнта та їх передачею на сервер займається браузер (Opera, Chrome, OperaGX, Mozilla Firefox тощо). Результат обробки запиту клієнта подається у вигляді веб-сторінки, яка описана мовою гіпертекстової розмітки HTML. На стороні серверу веб-застосунок виконується спеціальним програмним забезпеченням, який приймає запити користувачів, обробляє їх, формує відповідь у вигляді сторінки та передає результат клієнту.

Один із таких серверів – це Internet Information Services (IIS) компанії Microsoft. Даний сервер, єдиний який здатний обробляти запити веб-застосунку створеного з використанням технології ASP. NET.

Внаслідок наявності виконавчої частини, веб-застосунок здатен виконувати всі ті ж операції як і звичайні програми Windows. З єдиним обмеженням, що код виконується на сервері. Інтерфейсом виступає браузер та в якості середовища, за допомогою якого відбувається обмін даними з середовищем Інтернет. До найбільш типових операцій, які виконує веб-застосунок, відносяться:

- прийняття даних від користувача та збереження їх на сервері;
- виконання різних дій з БД на запит користувача (читання, видалення, додавання даних);
- відображення постійно мінливої інформації.

3.5 Логічне та фізичне моделювання даних системи обліку електронних петицій

Проведено логічне та фізичне моделювання даних ІС обліку електронних петицій за допомогою CASE-засобу «Allfusion ErWin Data Modeler». Даний засіб дозволяє створити логічну модель даних, яка не залежить від конкретної технології БД. Дана схематична модель надалі може бути використана для створення фізичної моделі даних та переведена у скрипт для створення БД у вибраній СУБД.

На основі уточнення функціональних вимог проведених в попередній частині за допомогою функціональної діаграми IDEF0 та діаграми потоків даних DFD

створено логічну та фізичну модель даних. Діаграму логічної моделі даних створено у вигляді ER-діаграми згідно з нотацією IDEF1X.

Методика створення логічної моделі даних системи з використанням CASE-засобу «Allfusion Erwin Data Modeler» складається з наступних етапів:

- визначення сутностей та атрибутів. Сутність – це набір (об'єднання) об'єктів, званих екземплярами. Атрибут – це перелік характеристик сутності. Наприклад: адреса, номер телефону, кількість тощо;
- визначення логічних взаємозв'язків. Логічні взаємозв'язки представляють зв'язки між сутностями, вони показують як одна сутність відноситься до іншої. Наприклад, продавець продає багато продуктів. В цей пункт входить визначення первинних та зовнішніх ключів;
- перевірка адекватності логічної моделі. Якщо взаємозв'язки описані правильно, можна описати відношення однієї сутності до іншої.

На рисунку 3.2 подано логічну модель даних обліку електронних петицій.

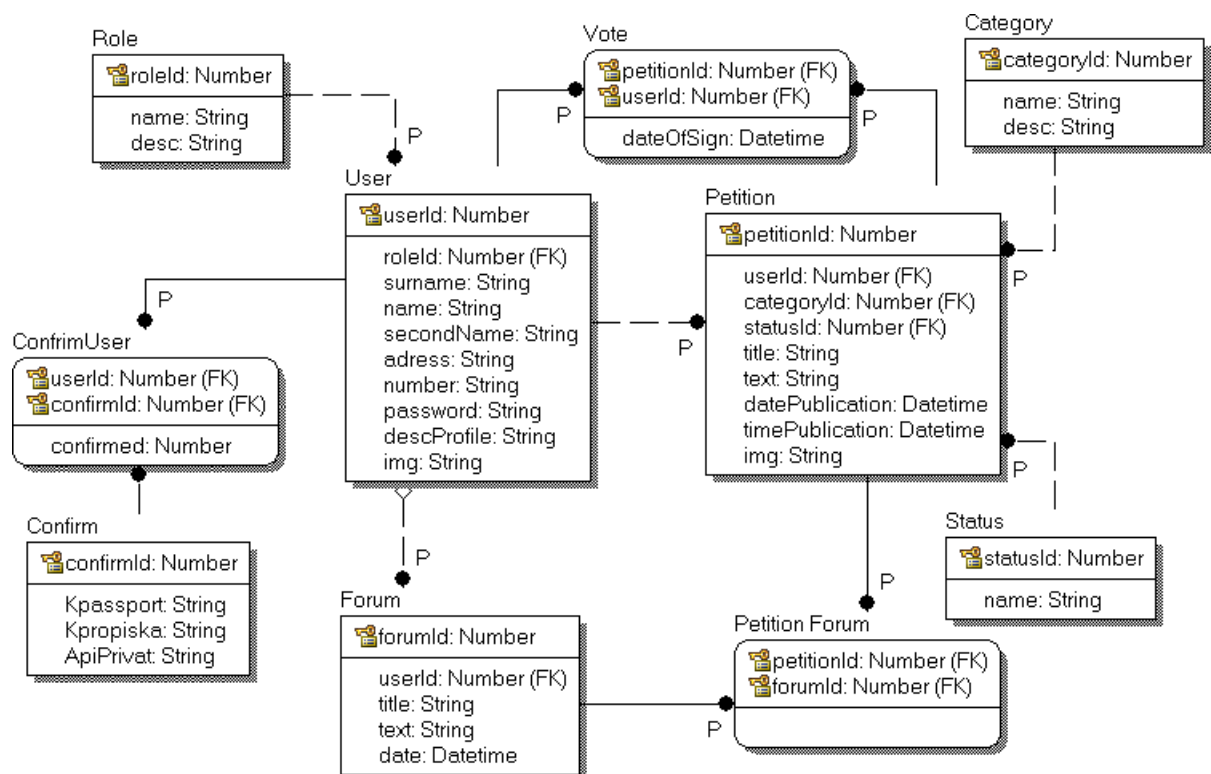


Рисунок 3.2 – ER-діаграма логічної моделі даних

Предметної областю системи є облік електронних петицій міської ради. Сутності які увійшли в логічну модель представлені у таблиці 3.1.

Таблиця 3.1 – Сутності логічної моделі даних БД

№	Назва сутності	Назва атрибуту	Тип даних (домен)	Призначення
1.	User (користувач)	UserId	Лічильник, довге ціле	Первинний ключ
		surname	Текст, довжиною 45 символів	Прізвище
		name	Текст, довжиною 45 символів	Ім'я
		secondName	Текст, довжиною 45 символів	По-батькові
		address	Текст, довжиною 45 символів	Адреса
		email	Текст, довжиною 40 символів	Електронна пошта
		number	Текст, довжиною 20 символів	Номер телефону
		password	Текст, довжиною 50 символів	Пароль
		roleId	Ціле число	Зовнішній ключ
		descProfile	Текст, довжиною 50 символів	Опис профілю
		img	Текст, довжиною 45 символів	Зображення у профілі
2.	Petition (петиція)	PetitionId	Лічильник, довге ціле	Первинний ключ
		title	Текст, довжиною 20 символів	Назва петиції
		text	Текст, довжиною 50 символів	Суть петиції
		datePublication	Дата	Дата публікації
		timePublication	Дата	Час публікації
		colSignature	Ціле число	Кількість підписів
		img	Текст, довжиною 20 символів	Зображення петиції
		categoryId	Ціле число	Зовнішній ключ
		userId	Ціле число	Зовнішній ключ
		statusId	Ціле число	Зовнішній ключ
3.	Role (ролі)	RoleId	Лічильник, довге ціле	Первинний ключ
		name	Текст, довжиною 20 символів	Назва ролі
4.	Category (категорії)	CategoryId	Лічильник, довге ціле	Первинний ключ
		name	Текст, довжиною 20 символів	Назва категорії
		description	ntext	Опис категорії
5.	Status (статус петиції)	StatusId	Лічильник, довге ціле	Первинний ключ
		name	Текст, довжиною 20 символів	Назва статусу
6.	ConfirmUser	ConfirmId	Ціле число	Зовнішній ключ
		UserId	Ціле число	Зовнішній ключ
		Confirmed	Ціле число	Статус верифікування

Продовження таблиці 3.1 – Сутності логічної моделі даних БД

7.	Confirm	ConfirmId	Лічильник, довге ціле	Первинний ключ
		Kpassport	Текст, довжиною 45 символів	Копія паспорту
		Kpropiska	Текст, довжиною 45 символів	Копія прописки
		ApiIdPrivatBank	Текст, довжиною 45 символів	Ідентифікатор ПриватБанку
8.	Vote (Голосування)	UserId	Ціле число	Зовнішній ключ
		PetitionId	Ціле число	Зовнішній ключ
		dataOfSign	Дата	Дата голосування
9.	Forum (Коментарі)	ForumId	Лічильник, довге ціле	Первинний ключ
		title	Текст, довжиною 20 символів	Назва коментаря
		text	Текст, довжиною 50 символів	Текст коментаря
		date	Дата	Дата коментування
		time	Дата	Час коментування
10.	PetitionForum	petitionId	Ціле число	Зовнішній ключ
		forumId	Ціле число	Зовнішній ключ

Для створення фізичної моделі даних вибрано СУБД Microsoft SQL Server. Ціль(або мета) фізичної моделі даних – це створення необхідної системної інформації для перенесення моделі даних у СУБД. А конкретно типи даних які належати до вибраної СУБД, взаємозв'язки, індекси, первинні та зовнішні ключі. На рисунку 3.3 подано фізичну модель даних.

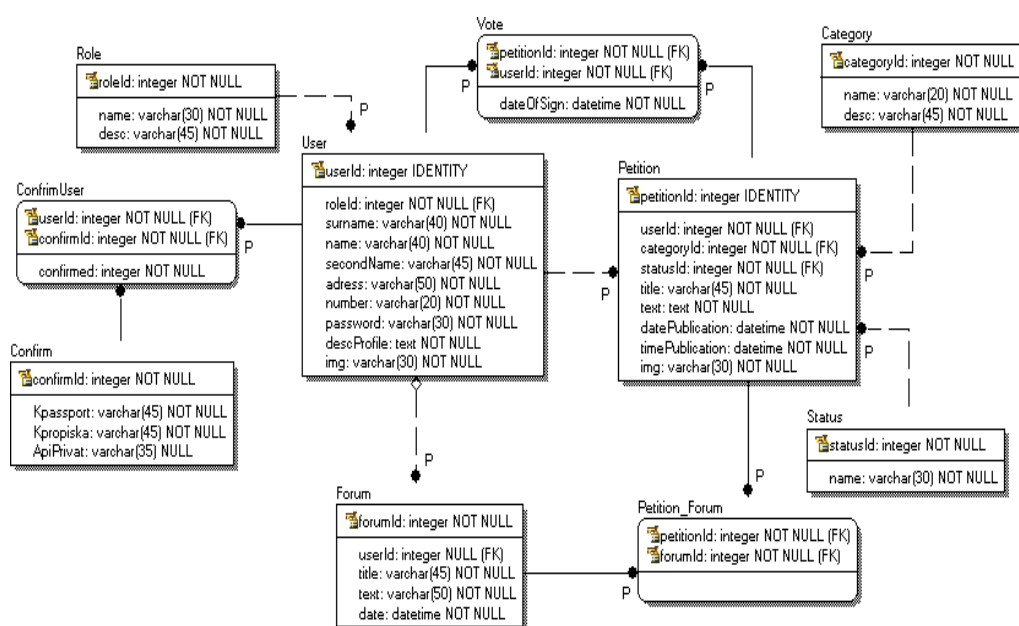


Рисунок 3.3 – ER-діаграма фізичної моделі даних

Підтримка посилальної цілісності у БД виконується за допомогою вбудованих можливостей Microsoft SQL Server. Обмеження цілісності – сукупність правил які, дозволяють забезпечити будь-якої миті часу правильність даних (несуперечність, адекватність наявним знанням у реальному світі) [24]. У стандарті SQL забезпечуються такі дії з підтримкою цілісності:

- cascade, при видаленні чи оновленні даних с батьківської таблиці видаляються або оновлюються дані з дочірньої таблиці;
- no action, з'являється помилка якщо дані зв'язані з іншими даними в дочірніх таблицях;
- set null, дані у зв'язаних таблицях при оновленні чи видаленні отримують значення null, якщо це допустимо;
- set default, встановлення значення за замовчуванням при оновленні чи видаленні.

У таблиці 3.2 представлено типи посилальної цілісності БД.

Таблиця 3.2 – Типи посилальної цілісності БД

№	Ім'я таблиці 1, зовнішній ключ	Ім'я таблиці 2, зовнішній ключ	SQL- інструкція для таблиці 1	Типи посильної цілісності
1	User, roleId	Role, roleId	UPDATE	NO ACTION
2	User, roleId	Role, roleId	DELETE	NO ACTION
3	ConfirmUser, userId	User,userId	UPDATE	NO ACTION
4	ConfirmUser, userId	User,userId	DELETE	CASCADE
5	Confirm, confirmId	ConfirmUser, confirmId	UPDATE	NO ACTION
6	Confirm, confirmId	ConfirmUser, confirmId	DELETE	CASCADE
7	Vote, userId	User, userId	UPDATE	NO ACTION
8	Vote, userId	User, userId	DELETE	NO ACTION
9	Vote, petitionId	Petition, petitionId	UPDATE	NO ACTION
10	Vote, petitionId	Petition, petitionId	DELETE	NO ACTION
11	Petition, categoryId	Category, categoryId	UPDATE	NO ACTION
12	Petition, categoryId	Category, categoryId	DELETE	SET DEFAULT
13	Petition, statusId	Status, statusId	UPDATE	NO ACTION
14	Petition, statusId	Status, statusId	DELETE	SET DEFAULT

Продовження таблиці 3.2 – Типи посильної цілісності БД

15	Petition, userId	User, userId	UPDATE	NO ACTION
16	Petition, userId	User, userId	DELETE	NO ACTION
17	PetitionForum, petitionId	Petition, petitionId	UPDATE	NO ACTION
18	PetitionForum, petitionId	Petition, petitionId	DELETE	NO ACTION
19	Forum, forumId	PetitionForum, forumId	UPDATE	NO ACTION
20	Forum, forumId	PetitionForum, forumId	DELETE	NO ACTION

Створення БД ІС обліку електронних петицій реалізовано на платформі Microsoft SQL Server 2019. На схемі БД представлені основні сутності: петиція «Petition», громадянин (користувач) «User», статус петиції «Status», категорії петицій «Category», ролі в системі «Role», підтвердження громадянина «Confirm», коментарі «Forum». Таблиці які реалізують зв'язок «багато до багатьох»: голосування «Vote», «ConfirmUser», «PetitionForum». Реалізована посильна цілісність, типи атрибутів відповідають типам на фізичній діаграмі які представлені у розділі 3.5.

Створена БД має такі типи зв'язків між таблицями:

- зв'язок один до багатьох між таблицями «Role» та «User». У однієї ролі може бути багато користувачів;
- зв'язок один до багатьох між таблицями «Category» та «Petition». Багато петицій можуть належати до однієї категорії;
- зв'язок один до багатьох між таблицями «Status» та «Petition». Багато петицій можуть мати один статус;
- зв'язок один до багатьох між таблицями «User» та «Petition». Один користувач може мати багато створених петицій.

В БД реалізовані зв'язки «багато до багатьох» за допомогою проміжної таблиці. Таблиця голосування «Vote» є проміжною таблицею між петицією «Petition» та користувачем «User». Таблиця «PetitionForum» – це проміжна таблиця, яка реалізує зв'язок багато до багатьох для коментарів користувачів. Зв'язок реалізується між таблицями петиція «Petition» та форум «Forum».

Схема БД у MSSM (Microsoft SQL Server Management Studio) подано на рисунку 3.4. У БД створена підстановочна таблиця – виведення повної інформації

петиції «PetitionIDShow», поєднує в собі п'ять таблиці: «User», «Petition», «Vote», «Category» та «Status». Створено тригер «Vote_triger_unikal» який не дозволяє голосувати більше ніж один раз одному громадянину за петицію. Даний тригер у таблиці голосування «Vote» порівнює вхідне значення строк «petitionId» та «userId». Виводить кількість таких записів та при значенні більше нуля, виконується операція скасування змін. Код тригера представлено у лістингу 3.1.

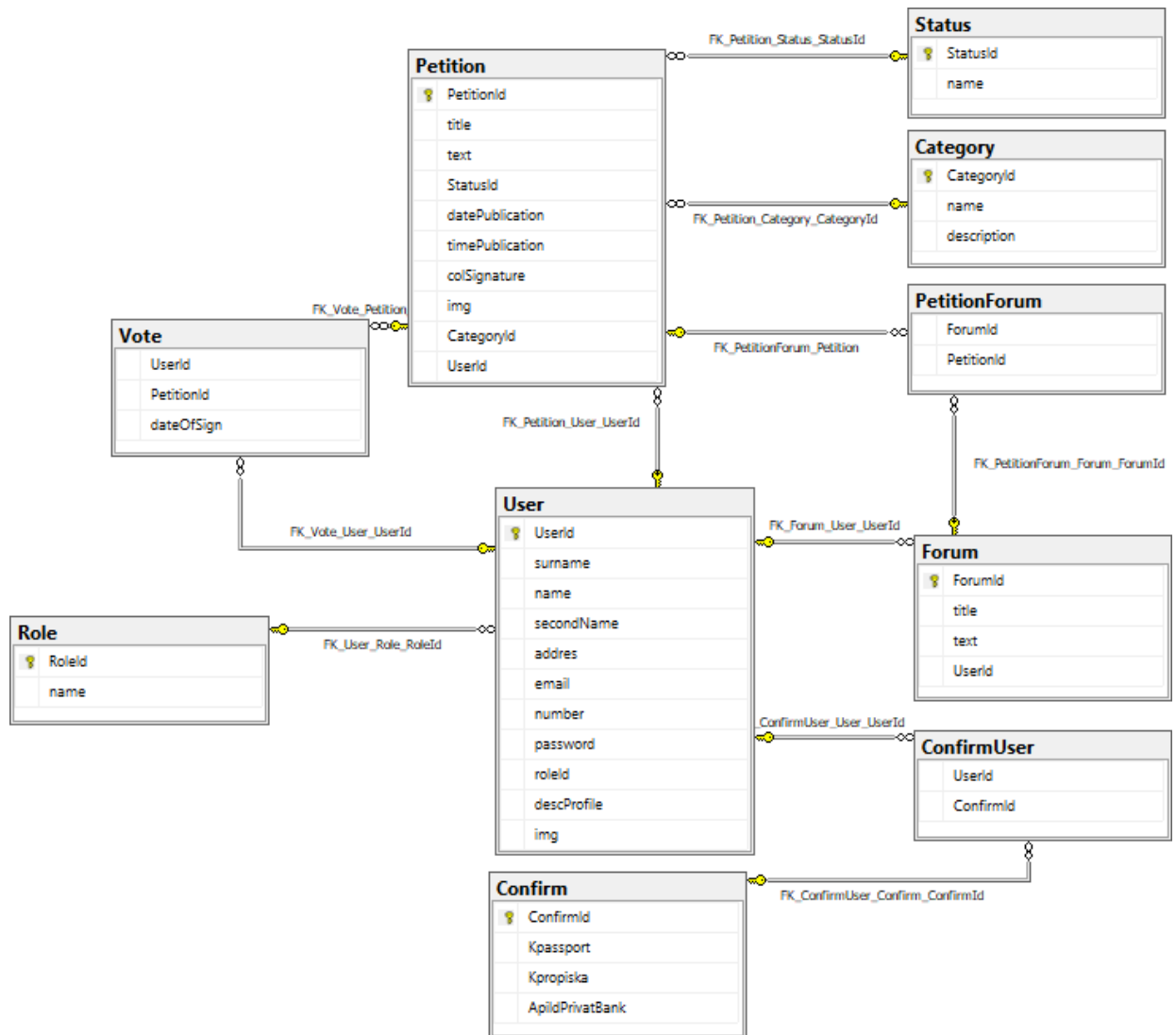


Рисунок 3.4 – Реалізована БД у MS SQL Server

Лістинг 3.1 – Код розробленого тригера «Vote_triger_unikal»

```

Use PetitionsIdentity// використання необхідної БД
GO // початок коду тригера
    
```

```

CREATE TRIGGER Vote_Triger_unikal // створити тригер з такою назвою
ON Vote // в таблиці голосування «VOTE»
AFTER INSERT //після вставки
AS //якщо
    //В таблиці вже є такий запис, тобто громадянин голосував
    if 0 < (Select Count(V.PetitionId)
            From Vote V, inserted I
            Where V.PetitionId=I.PetitionId and
            V.UserId=V.UserId)
    //Почати
    Begin
    Rollback TRAN//виконати відкат
    //Вивести системне повідомлення
    RAISERROR ('Даний голос уже засчитан', 16,10)
END// кінець коду триггеру

```

3.6 Карта сайту веб-застосунку обліку електронних петицій

Карта сайту – це перелік сторінок, що використовується для пошукових систем або користувачів та аналогічна змісту книги. Карта сайту дає змогу: визначити місцеперебування сторінок сайту, вказати час останнього оновлення сторінок, визначити пріоритет сторінок в структурі, вказати рівні доступу користувачів сайту [25].

З веб-застосунком можуть взаємодіяти три актори: гість (неzareєстрований або неавторизований користувач), громадянин (авторизований користувач) та адміністратор ІС. На рисунку 3.5 подана карта сайту ІС обліку електронних петицій за рівнями доступу користувачів.

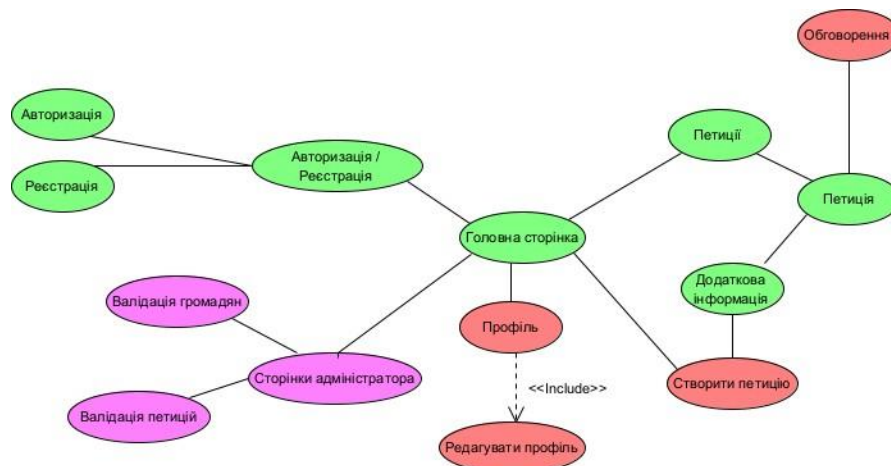


Рисунок 3.5 – Карта сайту ІС обліку електронних петицій

На карті сайту різними кольорами представлено різний рівень доступу до сторінок сайту:

- зеленим кольором виділені веб-сторінки на які може заходити та переглядати незареєстрований користувач – гість;
- червоним кольором виділені веб-сторінки до яких мають доступ авторизовані та верифікованні адміністратором користувачі та сам адміністратор;
- фіолетовим кольором виділені веб-сторінки до яких має доступ лише адміністратор ІС.

3.7 Розробка інтерфейсу клієнтської частини ІС

Доступ до основних функцій системи та доступ до бази даних здійснюється через користувацький інтерфейс. Виконано розробку сторінки виведення усіх петицій. На рисунку 3.6 зображено сторінку електронних петицій.

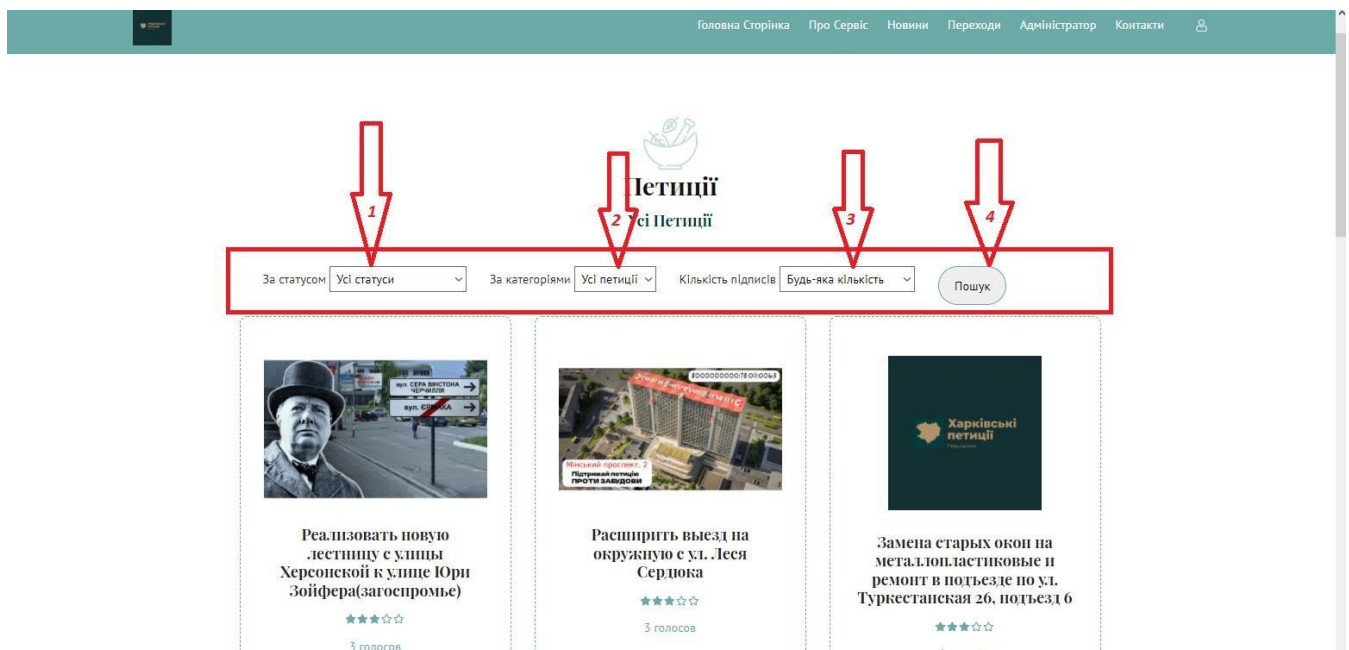


Рисунок 3.6 – Сторінка петицій

Сторінка складається з адаптивного меню та інформації про петиції, які можна сортувати. Одночасно на сторінці знаходиться сімдесят п'ять петицій. Петиції розташовані по три блоки в кожному рядку. Блок петиції складається з таких структурних елементів як зображення, яке надсилає автор, назви петиції,

зірок які виставляються адміністратором за якість викладеної інформації у суті петиції та кількості голосів на даний час.

Сторінка петицій пов'язана з класом «PetitionController», який виконує обробку запитів Get() та Post() зі сторінки та сховища даних «IAIPetition». Сховище пов'язане з таблицею петицій «Petition» у БД. Сторінка реалізована динамічно, кількість та порядок петицій може змінюватись за сортування та через додавання нових петицій у веб-застосунок. Меню та нижній елемент сайту – футер, знаходяться в іншому файлі який має назву Layout. Такий спосіб організації меню та футеру використовується для можливості розширення функціоналу без впливу на інші сторінки та зменшення ваги файлів сторінок.

Сторінка створена за допомогою мови гіпертекстової розмітки (HTML), каскадних таблиць стилів (CSS), фреймворку адаптивної верстки Bootstrap та сторінок Razor pages. Сторінки Razor pages дозволяють вбудовувати код мови C# для передачі інформації на сторінку з контентом. Код сторінки усіх петицій подано у лістингу 3.2.

Лістинг 3.2 – Код сторінки петицій

```
@using VotePetition.Models//використання моделей
@{
    Layout = "_Layout1";//підключення шаблону хедеру та футера
}
@model List<Petition>// використання моделі даних петицій
//підключення каскадних стилів та модулів фреймворку Bootstrap
<link rel="stylesheet" href="~/assets/css/bootstrap.min.css">
<link rel="stylesheet" href="~/assets/css/bootstrap.min.css">
<link rel="stylesheet" href="~/assets/css/all.min.css">
<link rel="stylesheet" href="~/assets/css/font.css">
<link rel="stylesheet" href="~/assets/css/swiper.min.css">
<link rel="stylesheet" href="~/assets/css/style.css">
<link rel="shortcut icon" href="~/assets/images/fav.png" type="image/x-icon">
@{//підключення c# на сторінку
    ViewData["Title"] = "Петиції";//назва сторінки
}
<!-- product start -->
<div class="pa-product pa-product-two spacer-top">//блок виведення петицій
    <div class="container">
        <div class="pa-heading">//виведення заголовків з інформацією
            
            <h1>Петиції</h1>
            <h5>Усі петиції</h5>
        </div>
        <div>//ссылки на сортування петицій
            <a asp-controller="Petitions" asp-action="listPetitions">Всі петиції</a>
            <a asp-controller="Petitions" asp-action="listPetitionsShow" asp-route-
id="1">Сортувати за голосами</a>
```

```

        <a asp-controller="Petitions" asp-action="listPetitionsShow" asp-route-
id="2">Сортувати за статусом</a>
    </div>

    <!--Petitions category END-->
    <div class="row">
        @{//підключення с#
        foreach (Petition pet in Model)//обробка об'єктів моделі у циклі foreach
        {
            <div class="col-lg-4 col-sm-6">
                <div class="pa-product-box">
                    <div class="pa-product-img">
                        //виведення інформації про картинку петиції
                        
                    </div>
                    <div class="pa-product-content">
//Перехід до більш детальної інформації про петицію
                        <h4><a asp-controller="Petitions" asp-action="Petition" asp-route-
id="@pet.PetitionId">@pet.title</a></h4>
                        <p class="pa-product-rating">
<i class="fas fa-star"></i><i class="fas fa-star"></i><i class="fas fa-star"></i><i
class="far fa-star"></i><i class="far fa-star"></i>
                        </p>
//виведення кількості голосів по петиції
                        <p> @pet.colSignature голосов</p>
                    </div>
                    <div class="pa-product-cart">
                        <ul>
                            <li>
                                <a href="#">
                                    
                                </a>
                            </li>
                        </ul>
                    </div>
                </div>
            </div>
        }
    </div>
</div>
<!-- product end -->
//підключення js-файлів для створення анімацій та динаміки сайту
<script src="assets/js/jquery.min.js"></script>
<script src="assets/js/bootstrap.min.js"></script>
<script src="assets/js/SmoothScroll.min.js"></script>
<script src="assets/js/swiper.min.js"></script>
<script src="assets/js/custom.js"></script>
<script src="~/lib/jquery/dist/jquery.min.js"></script>
<script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
<script src="~/js/site.js" asp-append-version="true"></script>

```

Докладний опис інтерфейсу та порядок роботи з веб-застосунком подано у Керівництві користувача (додаток Б).

3.8 Розробка контролеру для обробки петицій

Для забезпечення роботи веб-застосунку розроблено клас контролеру петицій «PetitionController». Клас використовуються для обробки запитів зі

сторінок які належать до петицій: веб-сторінка перегляду усіх петицій, сортування петицій, додання петицій та голосування за петицію. Даний клас успадковується від системного класу «Controller» бібліотеки «Microsoft.AspNetCore.Mvc.Controller».

Клас містить перелік таких функцій:

- функція `listPetitions()` – обробляє get-запит громадянина та отримує перелік усіх петицій з БД окрім петицій зі статусом «Нова петиція» та «Потребує редагування». Список петицій передається у представлення «listPetitions» та виводиться на сторінку браузера користувача системи;
- функція `listPetitionsShow(int id)` – обробляє get-запит користувача на сортування петицій. У функцію передається ідентифікатор за яким параметром сортувати петиції. Після сортування у представлення передається список відсортованих петицій та виводиться на сторінці браузера;
- функція `CreatePetition()` – відповідає за виведення сторінки створення нової петиції верифікованим громадянином. У функції перевіряється право користувача на створення петиції. Якщо користувач не має таких прав, буде виведено повідомлення про помилку;
- функція `CreatePetition(CrPetition model)` – виконує створення нової петиції, у функцію передається post-запит з формою в яку громадянин ввів інформацію про петицію. Функція створює петицію та зберігає у БД зі статусом «Нова петиція». Користувач отримує сповіщення про успішне створення петиції;
- функція `Petition (int id)` – виводить петицію за ідентифікатором петиції, який передається зі сторінки де користувач вибрав яку петицію хоче більш детально розглянути. З БД за ідентифікатором, який передається зі сторінки як параметр та виконується вибірка необхідної петиції. Надалі петиція записується в об'єкт класу «Petition» та передається у представлення та виводиться на сторінці «PetitionController/Petition/petition» у браузері;
- функція `Vote (int id)` – після натискання кнопки «Голосувати» функція перевіряє чи голосував громадянин за цю петицію. Якщо користувач вже голосував, сповіщає про помилку. В іншому разі, створює запис голосування. В таблиці БД зберігається ідентифікатор громадянина, ідентифікатор петиції, дата та час підписання петиції;

– функція `ListUsersPetitions (int id)` – створює перелік користувачів, які підписали петицію. Ідентифікатор петиції передається до функції після натискання громадянином кнопки на сторінці.

У класі реалізовано два конструктори для різних ситуацій, конструктор без параметрів та конструктор з параметрами. Код функції для створення петиції, класу «PetitionController» подано у лістингу 3.3. Код функції детальної інформації за петицією подано на рисунку 3.4.

Для забезпечення виконання функцій клас містить серіалізацію та LINQ.

Лістинг 3.3 – Код функції створення петиції «CreatePetition»

```
[HttpPost] //атрибут за допомогою якого зрозуміло тип
оброблюваного запиту, в даному разі це метод post()
[ValidateAntiForgeryToken]// використання фільтру,
призначений для протидії підробці міжсайтових запитів, роблячи
верифікацію токенів.
public async Task<IActionResult> CreatePetition(CrPetition
model)//функція створення нової електронної петиції з моделі яка
повернення як параметр функції
{
    User user = allUser.getObjectUser(User.Identity.Name);//
отримуємо інформацію про авторизованого користувача та записуємо в
об'єкт користувача
    Petition pt = new Petition();//створюємо об'єкт нової
петиції
    pt.UserId = (int)user.UserId;//записуємо дані в об'єкт
    pt.Category = allCategory.getObjectCategory(1); //вказуєм
категорію
    if (model.img == null)//якщо користувач не додав
зображення, використовуємо стандартне
        pt.img = "LogoKharkiv.jpg";//стандартне зображення
    else//в іншому разі
        pt.img = model.img;//використовуємо зображення, яке
загрузив користувач
    pt.colSignature = 0;//кількість голосів
    pt.StatusId = 0;//встановка статусу
    pt.title = model.title;//назва петиції
    pt.text = model.text;//суть петиції
    string date = DateTime.Now.ToString();//конвертація дати
    string[] dates = splitDatastring(date, " ");//розділення
тексту
    pt.datePublication = dates[0];//запис дати
    pt.timePublication = dates[1];//запис часу

    _context.Petition.Add(pt);//додаємо петицію до БД
```

```

        await _context.SaveChangesAsync();//Зберігаємо нову
петицію
Alert(allUser.getObjectUser(User.Identity.Name));//оповіщення
користувача
        return RedirectToAction("listPetitions",
"Petitions");//повернення на сторінку списку петицій
    }

```

Лістинг 3.4 – Код функції детальної інформації за петицією «Petition»

```

//повернення більш детальної інформації про петиції за
ідентифікатором
    public ActionResult Petition(int id)
    {
        Petition pet = allPetition.AllPetitions
            .FirstOrDefault(g => g.PetitionId == id);//отримання
петиції за ідентифікатором петиції
        User user = allUser.getObjectUser(pet.UserId);//отримання
об'єкту користувача який створив петицію по ідентифікатору
        Category cat =
allCategory.getObjectCategory(pet.Category.CategoryId);//отримання
категорії за ідентифікатором
        PetitionForShow petS = new PetitionForShow();//створення
об'єкту моделі даних для виведення детальної інформації про петицію
        if (pet.StatusId == 0)//якщо ідентифікатор статусу 0
            petS.Status = "Триває збір підписів";//записуємо в
змінну
        else if(pet.StatusId == 1)// якщо ідентифікатор статусу 1
            petS.Status = "На розгляданні";// записуємо в змінну
        else якщо ідентифікатор статусу 2
            petS.Status = "З відповіддю";// записуємо в змінну
        petS.PetitionId = pet.PetitionId;
        petS.img = pet.img;//записуємо дані в модель
        petS.datePublication = pet.datePublication; //записуємо
дані в модель
        petS.timePublication = pet.timePublication; //записуємо
дані в модель
        petS.colSignature = pet.colSignature; //записуємо дані в
модель
        petS.userName = user.surname+" "+user.name+"
"+user.secondName; //записуємо дані в модель
        petS.category = cat.name; //записуємо дані в модель
        petS.text = pet.text; //записуємо дані в модель
        return View(petS);//повертаємо модель у представлення для
виведення
    }

```

Повний код програми наведений у додатку В.

3.9 Розробка функцій аутентифікації та авторизації громадянина

Для забезпечення роботи системи на необхідному рівні, зокрема для створення та обробки петицій у програмі необхідна авторизація, яка ділила б користувачів на гостя, громадянина та адміністратора.

Аутентифікація – це процедура перевірки та встановлення належності користувачеві введених даних до інформації, яка знаходиться у БД.

Авторизація – це надання користувачеві певних прав у системі.

Для цього в програмі було створено дві функції які надають таку можливість:

- функція `Authenticate (User user)` – виконує перевірку введених даних користувача на відповідність інформації із БД. Функція приймає об'єкт класу користувач як параметр. Код функції подано у лістингу 3.5;

- функція `Login (LoginModel model)` – дає право користувачеві на виконання певних дій у системі. Функція приймає як параметр «LoginModel», модель даних в яку записуються дані введені користувачем у поля вводу на сайті. Функція взаємодіє з підфункцією «Authenticate», яка виконує перевірку. Код функції представлено у лістингу 3.6.

Лістинг 3.5 – Код функції `Authenticate (User user)`

```
private async Task Authenticate(User user)//функція аутентифікації
{
    // створюємо один claim
    var claims = new List<Claim>
    {
        new Claim(ClaimsIdentity.DefaultNameClaimType,
user.email),//записуємо емейл
        new Claim(ClaimsIdentity.DefaultRoleClaimType,
user.roleId.ToString())//записуємо роль
    };
    // створюємо об'єкт ClaimsIdentity
    ClaimsIdentity id = new ClaimsIdentity(claims,
"ApplicationCookie", ClaimsIdentity.DefaultNameClaimType,
ClaimsIdentity.DefaultRoleClaimType);
    // встановлення аутентифікаційних кукі
    await
HttpContext.SignInAsync(CookieAuthenticationDefaults.AuthenticationSc
heme, new ClaimsPrincipal(id));
}
```

Лістинг 3.6 – Код функції Login (LoginModel model)

```

[HttpPost] //атрибут за допомогою якого зрозуміло тип оброблюваного
запиту, в даному разі це метод post()

[ValidateAntiForgeryToken] [ValidateAntiForgeryToken]//
використання фільтру, призначений для протидії підробці міжсайтових
запитів, роблячи верифікацію токенів.

public async Task<IActionResult> Login(LoginModel model)//функція
авторизації
{
    if (ModelState.IsValid)//якщо все введено вірно
    {
        User user = await _context.User
            .FirstOrDefaultAsync(u => u.email == model.Email &&
u.password == HESHPass(model.Password));          if (user != null)//
якщо об'єкт не пустий
        {
            await Authenticate(user); // використання функції
аунтефікації
            return RedirectToAction("Index", "Home");//переадресація на
головну сторінку
        }
        ModelState.AddModelError("", "Некоректний логін або
пароль");//якщо щось введено не вірно
    }
    return View(model);//повертаємо представлення
}

```

3.10 Захист інформації ІС обліку електронних петицій

Для шифрування інформації, а саме паролів користувачів використовується алгоритм хешування MD5.

Алгоритм хешування MD5 – це 128-бітний алгоритм, розроблений професором Рональдом Л. Ривестом з Массачусетського технологічного інституту.

Даний метод шифрування, який приймає повідомлення будь-якої довжини як вхідні дані та повертає як вихідне значення повідомлення фіксованої довжини, яке буде використано для аутентифікації вихідного повідомлення. Алгоритм

хешування широко використовується для перевірки цілісності інформації та хешування паролів.

Алгоритм хешування MD5 складається з підготовки та п'яти етапів:

- на вхід алгоритму заходить потік інформації, яку потрібно перетворити у хеш-код. Довжина самого повідомлення вимірюється у байтах, тому повідомлення може бути різної довжини (навіть нульовим). Довжина повідомлення записується у змінну L, число є цілим та негативним;
- першим етапом виконується вирівнювання потоку. Першим кроком до кінця потоку додають один байт, а потім деяке число нульових біт, щоб число стало порівняне 448 по модулю 512. Вирівнювання виконується завжди, навіть якщо число вже дорівнює 448;
- другим етапом наприкінці повідомлення додають 64-бітове уявлення довжини даних (кількість біт у повідомленні) до вирівнювання. Спочатку записують молодші 4 байти, потім старші. Обчислення ґрунтується на представленні потоку у вигляді масиву слів по 512 байтів;
- третій етап – це ініціалізація буфера. Для обчислень ініціалізують чотири змінні, розмір яких по тридцять два біти. Початкові значення яких задаються шістнадцятковими числами;
- четвертий етап – це виконання обробки значень у циклі. Для кожного раунду, яких чотири штуки, використовується власна функція, яку зображено на рисунку 3.7 Кожен 512-бітний блок проходить 4 етапи обчислень по 16 раундів. Після обробки значення записуються у буфери створені на третьому етапі;
- на п'ятому етапі обчислення які знаходяться у буферах ABCD – це і є хеш-код. Якщо виводити побайтово, починаючи з молодшого байта A і закінчуючи старшим байтом D, ми отримаємо MD5-хеш[26]. На рисунку 3.8 зображено алгоритм хешування MD5.

$$\begin{aligned}
 & \text{1-й етап: } \text{FunF}(X, Y, Z) = (X \wedge Y) \vee (\neg X \wedge Z), \\
 & \text{2-й етап: } \text{FunG}(X, Y, Z) = (X \wedge Z) \vee (\neg Z \wedge Y), \\
 & \text{3-й етап: } \text{FunH}(X, Y, Z) = X \oplus Y \oplus Z, \\
 & \text{4-й етап: } \text{FunI}(X, Y, Z) = Y \oplus (\neg Z \vee X),
 \end{aligned}$$

Рисунок 3.7 – Перелік функцій для хешування

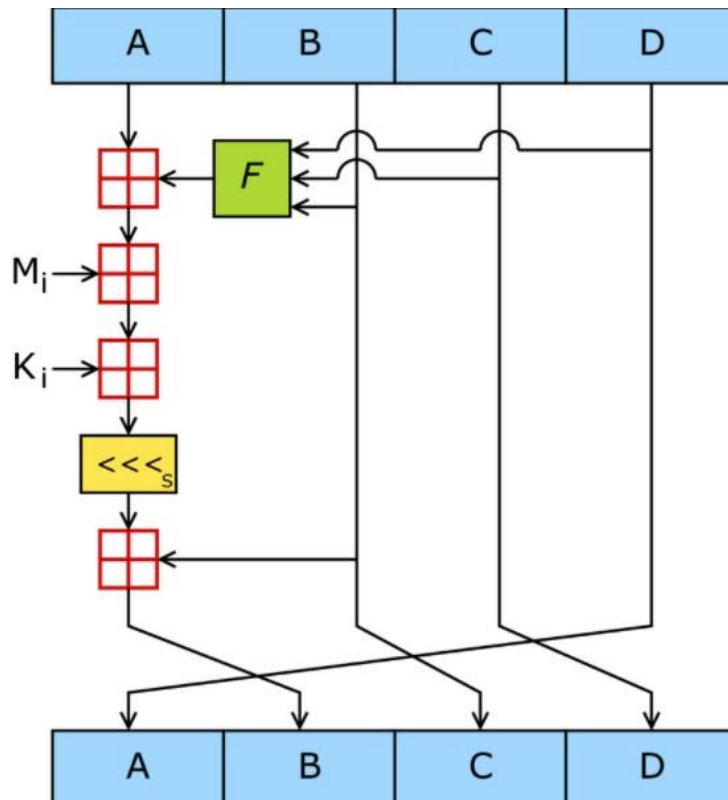


Рисунок 3.8 – Алгоритм хешування MD5

Алгоритм виконується для хешування паролів користувачів ІС обліку електронних петицій міської ради. Для розробки функції хешування використовується вбудована бібліотека, яку надає компанія Майкрософт – це бібліотека «System.Security.Cryptography». Бібліотека спеціально створена для забезпечення захисту інформації. В бібліотеці знаходиться метод хешування MD5, який буде використано. У лістингу 3.7 подано код функції хешування. Спочатку перетворюємо вхідне значення пароля у масив байтів, генеруємо хеш значення та виконуємо перетворення масиву хеш-байтів у рядок.

Після отримання форми користувача з паролем програма перетворює пароль у строку хеш-коду та порівнює з хеш-кодом у БД. Якщо паролі сходяться, громадянин входить на сайт.

Лістинг 3.7 – Код функції хешування MD5

```
//функція хешування
public string HESHPass(string pass)
{
```

```

//Перевірка можливості хешування
using (var md5Hash = MD5.Create())
{
    // Представлення масиву байтів вихідного рядка
    var sourceBytes = Encoding.UTF8.GetBytes (pass);

    // Генеруємо хеш-значення (байтовий масив) для
вхідних даних
    var hashBytes = md5Hash.ComputeHash(sourceBytes);

    // Перетворення масиву хеш-байтів у рядок
    var hash = BitConverter.ToString(hashBytes)
.Replace("-", string.Empty);

    // Вивести хеш MD5
    return hash;
}
}

```

3.11 Аналіз дослідної експлуатації та варіантів застосування ІС обліку електронних петицій

Проведено дослідницьку експлуатацію програмного застосунку, перевірено працездатність основних функцій програми:

а) клієнтська частина:

- 1) авторизація;
- 2) реєстрація;
- 3) створення електронної петиції;
- 4) перегляд та сортування петицій;
- 5) обговорення петиції за допомогою коментарів;
- 6) голосування;

б) адміністративна частина:

- 1) можливість авторизуватися, як адміністратор та мати можливість виконувати свої обов'язки;
- 2) верифікація громадян;
- 3) перевірка петицій громадян.

Перевірка функції «Авторизація» передбачає, що громадянин має змогу ввести облікові дані (електронну пошту та пароль) та після виконання функції буде надано доступ до системи. При виконанні сценарію коли громадянин ввів невірні дані авторизації, функція повинна повернути значення якого поля було введено

невірно. Якщо облікового запису немає, функція повинна повернути помилку з описом проблеми на сторінку авторизації (рис. 3.9).

The screenshot displays a login interface with the following elements:

- Header:** The word "Вхід" (Login) is centered at the top in a large, bold, black serif font.
- Error Message:** Below the header, a red text message reads: "Такого облікового запису немає. Перевірте пароль та логін!" (No such account exists. Check your password and login!).
- Input Fields:** There are two rounded rectangular input fields. The first field contains the email address "denzzar@nure.ua". The second field is empty and has the placeholder text "Пароль" (Password).
- Form Elements:** Below the password field, there is a checkbox labeled "Запомнить меня" (Remember me) and a link "Забули пароль ?" (Forgot password?).
- Buttons:** At the bottom, there are two buttons. The first is a dark green button with the text "Вхід" (Login). The second is a light blue button with the text "Зареєструватися" (Register).

Рисунок 3.9 – Невірно введені дані при авторизації

Перевірка функції «Реєстрація» передбачає введення громадянином даних (ПІБ, електронна пошта, номер телефону, адреса, копія головної сторінки паспорта, копія сторінки прописки та пароль). При вірному введенні даних функція автентифікує громадянина в системі, встановлює роль «Новий користувач» та виводить головну сторінку. При сценарії невірно введених даних громадянин отримає підказки біля поля введення даних (рис 3.10). В результаті реєстрації чи авторизації громадянин має змогу продивитись власний профіль (рис 3.11). В залежності від того верифікований громадянин адміністратором чи ні, залежить поле в профілі та можливість громадянина створювати чи голосувати за петиції. Якщо громадянин не авторизований замість сторінки профіля, користувача буде переадресовано на сторінку авторизації.

Реєстрація

Стахановський

Длина строки должна быть от 3 до 50 символов

1

Не указано по-батькові

По-батькові

Не вірний формат електронної пошти

admin

+380626220987

пл. Паторжинського, 78

Додайте копію чи зображення головної сторінки паспорта

Выберите файл | Файл не выбран

Додайте копію чи зображення сторінки прописки

Выберите файл | Файл не выбран

Длина строки должна быть от 3 до 50 символов

Пароль

Пароли не співпадають

Повторіть пароль

Зареєструватися [Вже є акаунт? Увійти](#)

Рисунок 3.10 – Невірно введені дані реєстрації



ПІБ: Стахановський Ігор Витальович

Додайте опис

Номер телефону:	+380635260987
Email :	StahVital@gmail.com
Адреса:	пл. Паторжинського, 78
Верифікація:	пройдена

Редактировать Профиль

Рисунок 3.11 – Профіль користувача

Перевірка функції «Створення петиції» передбачає що авторизований громадянин вводить дані петиції (заголовок, текст петиції) та відправляє на перевірку. Якщо громадянин не авторизований, буде виконана переадресація на сторінку авторизації. На рисунку 3.12 представлено введення інформації нової петиції та на рисунку 3.13 подана нова петиція у БД.

Деталі петиції

Повна назва петиції

Методика приёма работ по асфальтированию

Повний текст петиції

Проводится много работ по асфальтированию тротуаров, однако после сильного дождя выясняется, что на тротуарах образуются лужи. Зимой в на месте луж образуются толстые ледяные слои, которые не тают при потеплении. Необходимо разработать и сделать обязательной для применения методику приёма работ по асфальтированию тротуаров и дорожек. Методика должна включать испытание на отвод воды. На принимаемый участок покрытия выливать объем воды, соответствующий месячному объему осадков и через час контролировать отсутствие луж. В случае их наличия работа должна быть переделана.

Изображение для петиції

Выбрать файлы Файл не выбран

Відправити

Рисунок 3.12 – Введення інформації петиції

PetitionId	title	text	StatusId	datePublication	timePublication	colSignature	img	Catego
5	Реализовать новую лестницу с улицы Херсон...	Реализовать новую лестницу с улицы Херсонской к ...	0	05.05.2022	14:21:13	1	1.jpg	1
5	Расширить выезд на окружную с ул. Леся Се...	Прошу рассмотреть вопрос с расширением выезда...	0	04.05.2022	10:15:53	5	2.jpg	2
6	Замена старых окон на металлопластиковые...	Наш подъезд срочно нуждается в ремонте: ступень...	0	06.05.2022	13:10:01	8	LogoKharkiv.jpg	1
7	Городская канализация на Минутке Новобав...	Прошу рассмотреть возможность проведения горо...	0	07.04.2022	9:55:02	165	LogoKharkiv.jpg	1
8	"Принот" для бомжей и торговля цветами ст. ...	Очень прошу навести порядок в переходе метропо...	1	10.04.2022	15:13:13	1015	LogoKharkiv.jpg	1
9	Безготівкове поповнення e-ticket	Пропоную створити можливість поповнення карто...	0	9.05.2022	7:45:59	0	LogoKharkiv.jpg	1
10	Вернуть Трамвай Номер 8 на Льва Толстого	Уважаемый Игорь Александрович. Спасибо вам бо...	0	10.05.2022	13:54:23	0	LogoKharkiv.jpg	1
11	Інтеграція зупинки "Одеська" у систему Харкі...	Побудова метро до станції Одеська - це питання пов...	0	10.05.2022	14:03:12	0	LogoKharkiv.jpg	1
12	Обновить транспорт на маршруте 102з	Уважаемый, Игорь Александрович и Департамент т...	0	10.05.2022	14:05:53	0	train.jpg	1
13	Методика приёма работ по асфальтировани...	Проводится много работ по асфальтированию трот...	0	02.06.2022	17:58:55	0	LogoKharkiv.jpg	1
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Рисунок 3.13 – Додана нова петиція у БД

Перевірка функції «Обговорення петиції» передбачає, що громадянин продивившись петицію має змогу продивитись та залишити коментарі. Перейшовши за гіперпосиланням нижче петиції, користувач переходить на сторінку обговорення. Якщо на сторінці не залишено не одного коментаря, буде поле для вводу та запис «Будьте першим, залиште коментар!» (рис 3.14). Якщо коментарі вже були користувач може переглянути та ввести інформацію свого коментаря (рис 3.15).

Будьте першим, залиште коментар!

Введіть коментар

Отправить

Рисунок 3.14 – Сторінка без коментарів

Городская канализация на Минутке Новобаварский район

Текст петиції: Прошу рассмотреть возможность проведения городской канализации на районе "Минутка" (улица Красная аллея и улица Михайловская) Новобаварского района. Мы живём в частных секторах, в 5 минутах от города, и не имеем возможности пользоваться канализацией, как все люди в 21 веке. Недалеко от района "Минутка" проведена канализация, например, по проспекту Ново-баварский. Это недалеко от нашего района. Проведите, пожалуйста, нам городскую канализацию!

Коментарі

100 x 100

Павлюк Данило Васильович

05.06.2022 | 14:53:55

Гарна петиція.

Введіть коментар

Отправить

Рисунок 3.15 – Сторінка з коментарями

Перевірка функції «Голосування» передбачає що користувач натискаючи кнопку голосує за петицію. Користувач має бути авторизований для можливості голосувати. Якщо користувач вже голосував, проголосувати знову він не зможе та отримає повідомлення про помилку (рис 3.16).

Перевірка функції «Перегляд петицій та сортування» передбачає, що користувач може відсортувати та продивитися петиції. Функція виконує сортування за усіма параметрами (статуси, кількість голосів, категорії петицій).

Ви вже голосували за цю петицію

Голосувати за одну й ту ж петицію ви не можете.

Повернутися до переліку петицій

Рисунок 3.16 – Помилка при повторному голосуванні

Перевірка функції «Верифікації користувачів». Функція надає можливість адміністратору переглянути інформацію про користувача, а саме копії паспорта та вирішити питання верифікації громадянина змінивши його статус (рис 3.17). На формі верифікації громадянина мається перелік громадян, яким потрібна верифікація. Натиснувши кнопку детальніше, адміністратор заходить на сторінку детальної інформації (рис 3.18).

Верифікація

ПІБ	Адреса	Email	Номер телефону	Роль
Стахановський Ігор Витальович	пл. Паторжинського, 78	StahVital@gmail.com	+3809966557952	Новий користувач

Документи користувача для верифікації:



☐ Верифікувати

☐ Відхилити

Вибрати

Рисунок 3.17 – Детальна інформація про громадянина для верифікації

ПІБ	Адреса	Email	Номер телефону	Роль	Детальніше
Стахановський Ігор Витальович	пл. Паторжинського, 78	StahVital@gmail.com	+3809966557952	Новий користувач	Верифікувати

Рисунок 3.18 – Кількість нових громадян для верифікації

Перевірка функції «Перевірка петицій громадян» надає можливість адміністратору системи отримати форму з переліком петицій. Переглянути та вибрати відправити на редагування або опублікувати петицію (рис 3.19).

Назва Петиції	Суть Петиції	Статус	Змінити Статус
Создать сайт для "Центра реинтеграции бездомных особ"	Добрый день! Прошу создать сайт для "Центра реинтеграции бездомных особ" где будет размещаться информация о потребностях центра. О нужной одежде, обуви, в каких размерах. Возможно о судьбах людей которые не против огласки. Это очень облегчит и увеличит помощь от людей желающих помочь человеку в сложной ситуации. За ранее спасибо!	Нова петиція	<input type="radio"/> Триває збір підписів <input type="radio"/> Потребує редагування <input type="button" value="Вибрати"/>

Рисунок 3.19 – Форма валідації петицій адміністратором

В результаті дослідницької експлуатації були перевірені основні функції на працездатність та описані сторінками помилок та виключень від можливих дій громадянина. Функції доводять, що веб-застосунок працює коректно.

Розроблений веб-застосунок можна використовувати в населених пунктах органами місцевого самоуправління. Окрім основного призначення можна використовувати для активістських організацій, де потрібно голосування та волевиявлення людей.

Отже, в результаті було розроблений веб-застосунок обліку електронних петицій. Для функціонування веб-застосунку були розроблені:

- контролери – обробляють запити користувача та виводять результати на сторінку;
- репозиторії (сховища) – тимчасово зберігають інформацію про петиції, користувачів тощо;
- представлення – виводять інформацію на сторінку, яка було отримана з БД;
- користувацький інтерфейс – для використання користувачами та адміністратором для відображення елементів меню та сторінок;
- база даних в якій зберігаються дані відносно моделі даних описаній вище;
- тригер для бази даних для виконання функції голосування;
- підстановочна таблиця;
- функція хешування паролів для забезпечення захисту інформації.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи було проаналізовано предметну область обліку та подачі електронних петицій до органів місцевого самоуправління. Були проаналізовані та розглянуті сучасні аналоги веб-застосунків обліку електронних петицій великих населених пунктів України. В результаті аналізу предметної області виявлено алгоритм обліку електронних петицій, який зараз використовується, та також можливі проблеми та недоліки. У результаті аналізу існуючих систем обліку електронних петицій виявлено, що багато систем реалізовано з недоліками або недостатнім функціоналом.

Аналіз сучасних систем обліку електронних петицій дозволив дійти висновку, що завдання розробки компонентів ІС обліку електронних петицій залишається актуальною. Виходячи з результату аналізу предметної області та існуючих систем було прийнято рішення провести функціональне моделювання системи за допомогою діаграми IDEF0 та діаграми потоків даних – DFD, яке надало змогу уточнити функціональні вимоги. Була створена діаграма варіантів використання, яка надала змогу продивитися концептуальну модель системи та сценарії взаємодії акторів з системою. Для проектування системи були побудовані діаграми класів, послідовності, кооперацій та діаграма станів.

Розроблена інформаційна система надасть змогу органам місцевого самоврядування спростити процедуру подачі та обліку електронних петицій та зменшити документообіг та також дозволить отримувати звітність по уподобанням громадян та кількості петицій. Громадяни отримають можливість подавати, голосувати та обговорювати електронні петиції, та це за допомогою автоматизації системи буде займати не багато часу.

При порівнянні розробленої ІС з існуючими системами – не має недоліків існуючих систем та має перевагу у функціоналі обговорення петицій.

Роботу можна використовувати для організації обліку електронних петицій у територіальних громадах та містах органами місцевого самоврядуван