МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ПРИРОДОКОРИСТУВАННЯ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЗАОЧНОЇ ТА ПІСЛЯДИПЛОМНОЇ ОСВІТИ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

другого (магістерського) рівня вищої освіти

на тему:

ПРОЕКТУВАННЯ ВЕБЗАСТОСУНКУ ІНФОРМАЦІЙНОЇ СИСТЕМИ «РОЗУМНЕ МІСТО»

Виконав: здобувач групи ІТ-71з
спеціальності 126 «Інформаційні системи та технології»

_____ Дзидз В. В.
(прізвище та ініціали)

Керівник:_____ Пташник В. В.
(прізвище та ініціали)

Рецензент_____
(прізвище та ініціали)

ДУБЛЯНИ-2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ПРИРОДОКОРИСТУВАННЯ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЗАОЧНОЇ ТА ПІСЛЯДИПЛОМНОЇ ОСВІТИ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Другий (магістерський) рівень вищої освіти
Спеціальність 126 «Інформаційні системи та технології»

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____
(підпис)
д.т.н., професор, Тригуба А. М.
(вч. звання, прізвище, ініціали)
“____” _____2024 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дзидз Вікторія Володимирівна
(прізвище, ім’я, по батькові)

1. Тема роботи *Проектування вебзастосунку інформаційної системи «розумне місто»*
керівник роботи *к. т. н., доцент., Пташник В. В.*
(наук.ступінь, вч. звання, прізвище, ініціали)

затверджені наказом Львівського НУП *від 08.03.2024 року № 171/к-с*
2. Строк подання студентом роботи *06 грудня 2024 року*
3. Вихідні дані до роботи: *характеристика сучасних інформаційних систем розумного міста; технічна документація та стандарти бездротових протоколів зв’язку; специфікація систем та пристроїв на основі технології інтернету речей; науково-технічна і довідкова література.*
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
Вступ
1. Архітектура та технології розумного міста
2. Вибір технологій зв’язку та загальної архітектури вебзастосунку
3. Розробка вебзастосунку «розумне місто»
4. Охорона праці та безпека в надзвичайних ситуаціях
5. Економічна ефективність
Висновки
Список використаних джерел
5. Перелік графічного матеріалу
Графічний матеріал подається у вигляді презентації

6. Консультанти розділів

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата		Відмітка про виконання
		завдання видав	завдання прийняв	
1, 2, 3, 5	*Пташник В. В., к.т.н., доцент*			
4	*Городецький І. М., к.т.н., доцент*			

7. Дата видачі завдання *11 березня 2024 р.*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Відмітка про виконання
1	*Аналіз архітектури та технології розумного міста*	*11.03.2024 – 30.04.2024*	
2	*Вибір технологій зв'язку та загальної архітектури вебзастосунку*	*01.05.2024 – 31.05.2024*	
3	*Розробка елементів вебзастосунку «розумне місто»*	*08.07.2024 – 30.09.2024*	
4	*Розгляд питань з охорони праці та безпеки у надзвичайних ситуаціях*	*01.10.2024 – 15.10.2024*	
5	*Оцінка економічної ефективності прийнятих рішень*	*15.10.2024 – 31.10.2024*	
6	*Завершення оформлення розрахунково-пояснювальної записки та презентаційного матеріалу*	*01.11.2024 – 20.11.2024*	
7	*Завершення роботи в цілому. Підготовка до захисту кваліфікаційної роботи*	*21.11.2024 – 06.12.2024*	

Здобувач _____ *Дзидз В. В.*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Пташник В. В.*

УДК 681.521 / 681.518

Проектування вебзастосунку інформаційної системи «розумне місто». Дзидз В. В. Кафедра інформаційних технологій – Дубляни, Львівський національний університет природокористування, 2024.
Кваліфікаційна робота: 70 сторінок текстової частини, 39 рисунків, 21 джерело літератури.

Метою кваліфікаційної роботи полягає в розробці вебзастосунку інформаційної системи «Розумне місто» для автоматизації та оптимізації управління міськими ресурсами, покращення якості життя мешканців та забезпечення інтерактивної взаємодії між громадянами і міською владою.

Об'єктом дослідження є процеси збору, обробки та інтеграції даних для забезпечення функціонування системи «Розумне місто».

Предмет дослідження вивчає методи, інструменти та технології проектування і розробки вебзастосунку для управління інформаційною системою «Розумне місто».

У кваліфікаційній роботі розглянуто суть поняття «Розумне місто» у розрізі розробки програмного забезпечення для захищеної та безпечної передачі даних у рамках необхідних комунікаційних технологій. Розроблено багаторівневу модель функціонування розумного міста. Проаналізовано базові складові розумного міста відповідно до розробленої багаторівневої моделі. Обгрунтовано вибір технологій бездротового зв'язку, загальну архітектуру вебзастосунку, проведено аналіз популярних форматів передачі даних та систем керування базами даних. У роботі розроблено структуру бази баних вебзастосунку «Розумне місто», описано базові запити та моделі функціонування відповідних рівнів програмного продукту.

Ключові слова: інформаційна система, розумне місто, вебзастосунок.

ЗМІСТ

ВСТУП

Сучасні міста стикаються з численними викликами: стрімке зростання населення, перевантаженість транспортної та комунальної систем, необхідність скорочення споживання енергії та підвищення якості життя містян. Для вирішення цих проблем використовуються інноваційні підходи, зокрема концепцію «розумних міст». Розумні міста – це міські екосистеми, які забезпечують сталий розвиток за допомогою комплексних технологій, таких як Інтернет речей, великі дані, штучний інтелект і хмарні обчислення.

Важливим елементом роботи розумного міста є інформаційні системи, які обробляють і аналізують дані, отримані від різноманітних датчиків і пристроїв. Вебдодатки в таких системах служать інтерфейсами для доступу користувачів до інформації, взаємодії з міськими службами та управління ними. Очікується, що вебдодатки для розумних міст матимуть високу продуктивність, безпеку, зручність і здатність обробляти великі обсяги даних у реальному часі.

Проектування мережевих додатків для інформаційних систем «розумного міста» передбачає розробку структури, яка включає модулі для моніторингу та управління ресурсами, послугами та даними. Це дозволяє інтегрувати різні системи, такі як освітлення, дорожній рух, комунальні послуги та управління безпекою. Розробка таких додатків вимагає врахування сучасних принципів дизайну, вибору відповідних технологій, побудови масштабованої архітектури та забезпечення надійності системи.

Тому метою цього дослідження є аналіз та розробка ефективних структур для вебдодатків, які відповідають вимогам інформаційних систем розумного міста. Це не лише покращить якість життя містян, а й сприятиме сталому розвитку міської інфраструктури, зменшить її екологічний слід та покращить взаємодію між мешканцями та міськими службами.

РОЗДІЛ 1
АРХІТЕКТУРА ТА ТЕХНОЛОГІЇ РОЗУМНОГО МІСТА

1.1 Екосистема розумного міста

Відповідно до Стратегії сталого розвитку України – 2020, актуальні завдання інтелектуалізації українського суспільства зумовлені проблемами, що виникають у великих містах, зокрема: енергетика, відходи та вода, охорона здоров'я, громадський транспорт, безпека, освіта, система зайнятості та економічного розвитку. Проектування інформаційної системи «Розумне місто» доцільно реалізовувати враховуючи вищезгадані проблеми з використанням сучасного технологічного підходу для забезпечення їх ефективного вирішення.

Інформаційна система «розумного міста» — це взаємопов'язана структура інтелектуальних компонентів, інтелектуальних операційних технологій та елементів захисту інформації, комплексно розроблена для забезпечення безпеки, екологічності, комфортної експлуатації та управління проблемними ситуаціями соціальної інфраструктури.

Основними суб'єктами розумних міст є міська влада та національні установи. Інфраструктура та послуги розумного міста спрямовані на освіту, охорону здоров'я, навколишнє середовище, транспорт, водопостачання, переробку відходів, розподіл енергії, захист людини та довкілля. Основними інформаційними технологіями, що задіяні у реалізацію технології розумних міст є аналітика даних, кібербезпека, геоінформаційні системи, фізична інфраструктура та комп'ютерні мережі. Поняття розумне місто є надзвичайно комплексним і включає у себе ряд інших структурних елементів, зокрема розумний розподіл енергії, розумні подорожі, розумна економіка, розумне середовище, розумна культура, розумна медична допомога, розумна безпека, розумне правосуддя та розумне управління (рис. 1.1).

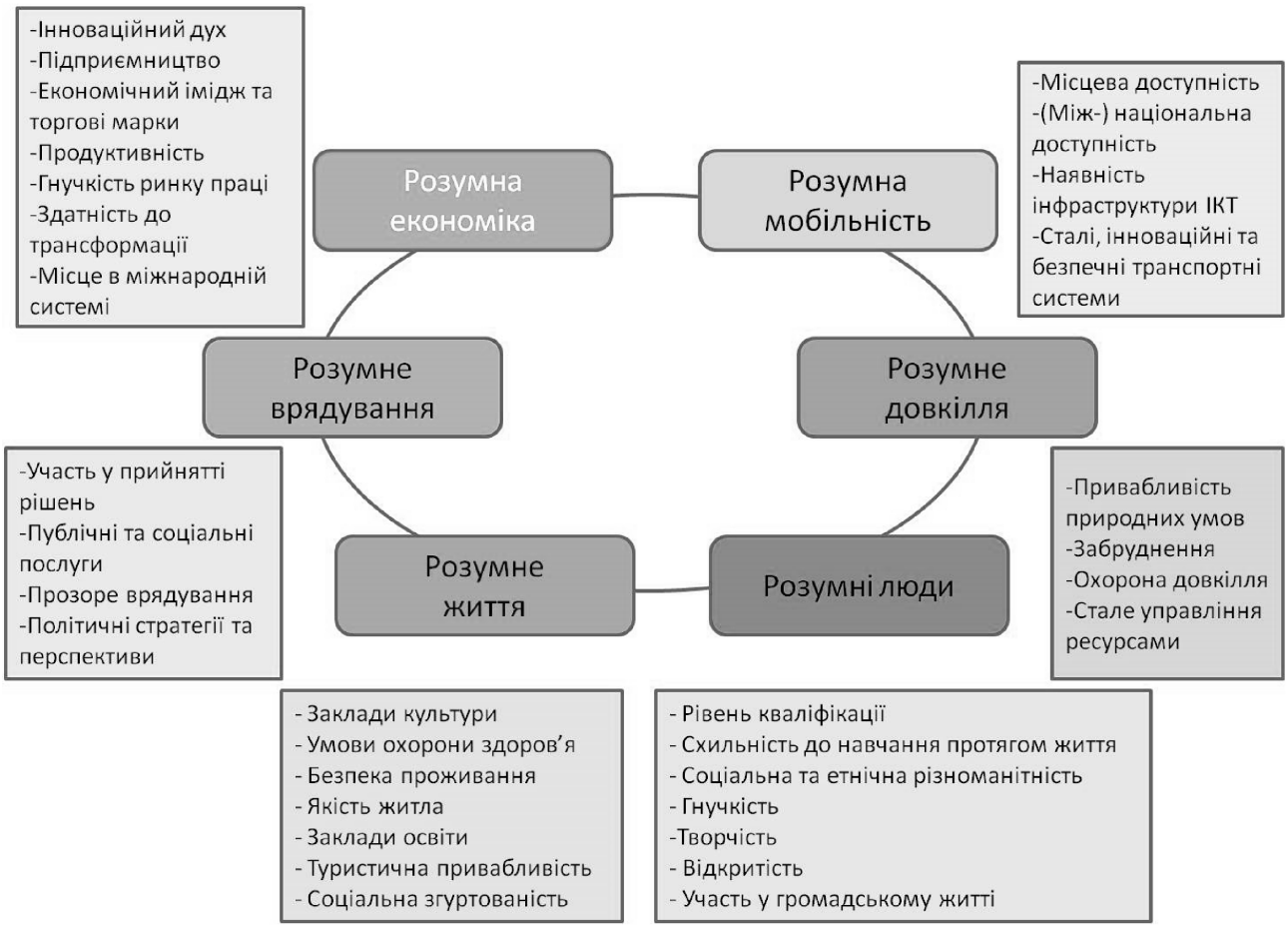

Рисунок 1.1 – Структурні елементи розумних міст

Базовими принципами, які обов'язково необхідно враховувати під час проектування системи розумне місто та її структурних елементів є організованість, солідарність, інтелектуальність, інформатизація та взаємозв'язок усіх елементів. Організованість полягає у наданні організованих адміністративних послуг мешканцям і установам, що здійснюються відповідно до усталених норм та законодавства. Солідарність розумного міста полягає спільному об'єднанні адміністрації міста та його мешканців, ключовими аспектами якого є прозорість та відкритість даних. Інтелектуальність розумного міста передбачає використання аналітичних і когнітивних інструментів для стимулювання зміни поведінки різних груп мешканців. Цей процес також супроводжується інформатизацією розумного міста завдяки збору, інтеграції, доповненню, зберіганню та передачі даних для покращення процесу прийняття рішень.

Функціональність систем розумного міста чітко пов'язана з трьома основними технологіями концепції Industry 5.0 (п'ята промислова революція): автоматизацією, Інтернетом речей, машинним навчанням та інформаційно-комунікаційними технологіями. Інтернет речей (IoT – Infrastructure) – це набір датчиків, перетворювачів, пристроїв і вимірювальних пристроїв, які реєструють/вимірюють дані об'єктів у фізичному середовищі, порівнюють їх із встановленими стандартами та передають оброблену інформацію на загальну платформу керування. У екосистемі розумного міста можна виділити чотири типи творців цінностей. Вони створюють і споживають цінності довкола одного з шарів, наведених на рисунку 1.2.

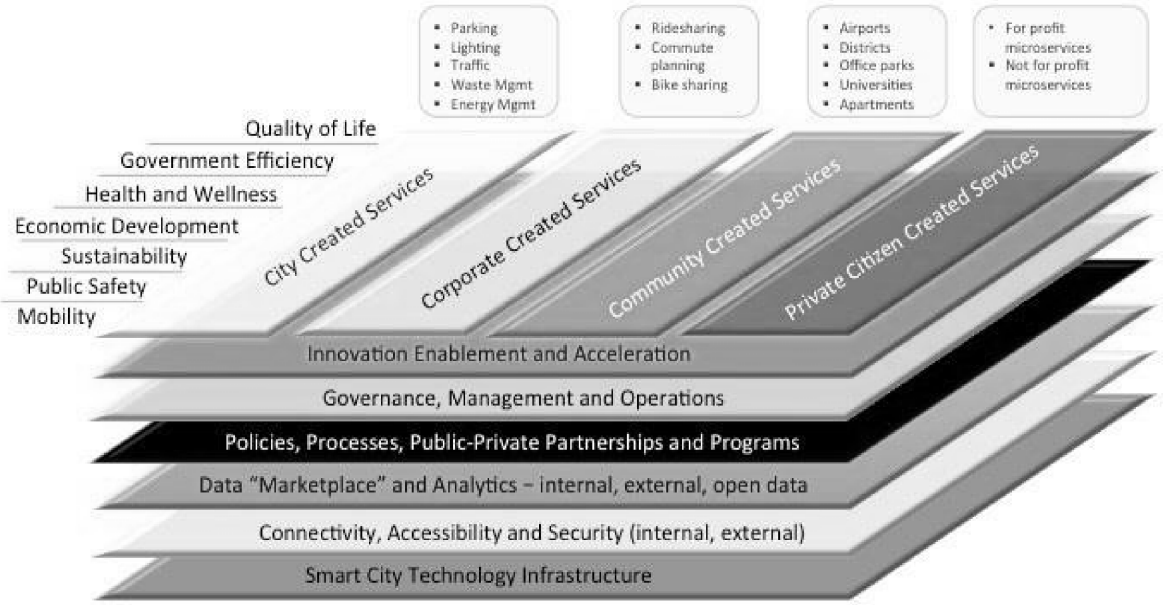

Рисунок 1.2 – Ціннісні шари та творці цінностей розумного міста

Коли люди думають про розумні міста, вони автоматично думають про послуги, які надають муніципальні та квазіурядові установи, такі як розумне паркування, розумне управління водою, розумне освітлення тощо.

Підприємства та організації можуть створювати служби, які використовують і створюють інформацію для надання результатів зацікавленим сторонам. Деякі приклади «розумного» бізнесу включають Uber і

Uklon для особистої мобільності, NextDoor для обміну інформацією та Waze/Google для транспорту та планування маршрутів.

Громади — це мікророзумні міста, але з дуже локальними потребами. Деякі приклади потенційних розумних спільнот включають університетські кампуси, офісні парки, аеропорти, вантажні порти, багатоквартирні будинки або житлові комплекси, комерційні райони та навіть окремі «розумні» будівлі. У них є розумні потреби в обслуговуванні, які можна пристосувати спеціально до їхніх зацікавлених сторін.

Мешканці або окремі громадяни також є постачальниками розумних послуг у розумних містах. Мешканці, які живуть поблизу небезпечних перехресть, можуть скеровувати камери на перехрестя та передавати цю інформацію до планувальників дорожнього руху в режимі реального часу. Мешканці встановлюють датчики якості повітря на своїх ділянках, щоб контролювати рівень забруднення та пилу в певну пору року та передавати цю інформацію іншим членам громади. Мешканці можуть користуватися цими смарт-сервісами тимчасово чи постійно, безкоштовно чи за плату.

1.2 Багаторівнева модель розумного міста

Розумне місто складається з кількох «рівнів можливостей». Хоча технологія є найважливішим фактором, це лише одна з багатьох базових можливостей, які має забезпечити кожне розумне місто. Усі можливості, що надає мешканцям розумне місто є рівноцінними, кожна з них відіграє свою роль у функціонуванні громади розумного міста. Ці можливості повинні бути об’єднані та скоординовані один з одним для виконання своєї місії (рис. 1.3).

Ціннісний рівень, найбільш помітний для жителів міста, підприємств, відвідувачів, працівників, студентів, туристів та інших. Цей рівень є каталогом послуг «розумного міста» або «варіантів використання», які орієнтовані на

результат, надаються творцями цінності та споживаються зацікавленими сторонами міста.

Щоб залишатися актуальними, творці цінностей у розумних містах повинні постійно впроваджувати інновації та вдосконалювати послуги, які вони надають своїм зацікавленим сторонам. Таким чином формується інноваційний рівень. Розумні міста активно сприяють цьому через різноманітні інноваційні проекти, включаючи лабораторії, інноваційні зони, навчання, творчі майстерні тощо.

Розумні міста сприяють цифровій трансформації існуючих процесів і послуг. Моделі управління розумним містом повинні інтегрувати нову екосистему творців цінностей та інноваторів. Вони повинні планувати, підтримувати та отримувати прибуток від нових бізнес-моделей, процесів і послуг. Вони повинні модернізувати існуючу інфраструктуру та процеси управління для підтримки «розумних» послуг. Нарешті, вони повинні використовувати новий набір показників для вимірювання ефективності міста. Саме таким чином формується управлінський або операційний рівень розумного міста.

Розумні міста не з'являються одномоментно. Розбудова, експлуатація та підтримка «розумних міст» вимагає нового набору моделей взаємодії, правил, джерел фінансування та партнерів. Міста повинні розвивати новий набір «розумних» можливостей, щоб брати участь і залишатися в грі розумного міста. Таким чином формується фінансовий рівень розумного міста, який по суті забезпечує його сталий розвиток та прогрес.

Безперечно основою життя розумних міст є інформація (інформаційний рівень). Розумні міста мають сприяти цьому різними способами, включаючи ініціативи відкритих даних, кіски даних, аналітичні служби та політику монетизації. Не менш важливо, щоб функціонували програми, що заохочують обмін даними, і політику конфіденційності, яка захищає дані та їх власників.

Рівні підключення, доступності та безпеки забезпечують зв'язок людей, речей та системи в розумних містах. Можливість легко підключити всі три

рівні, керувати та автентифікувати користувачів. Першочерговим завданням розумного міста є забезпечення безперебійного рівня надійного підключення.

Технологічна інфраструктура розумного міста має виходити за межі традиційних муніципальних користувачів і підтримувати нові типи творців цінності та зацікавлених сторін міста/користувача формуючи технологічний рівень розумного міста.

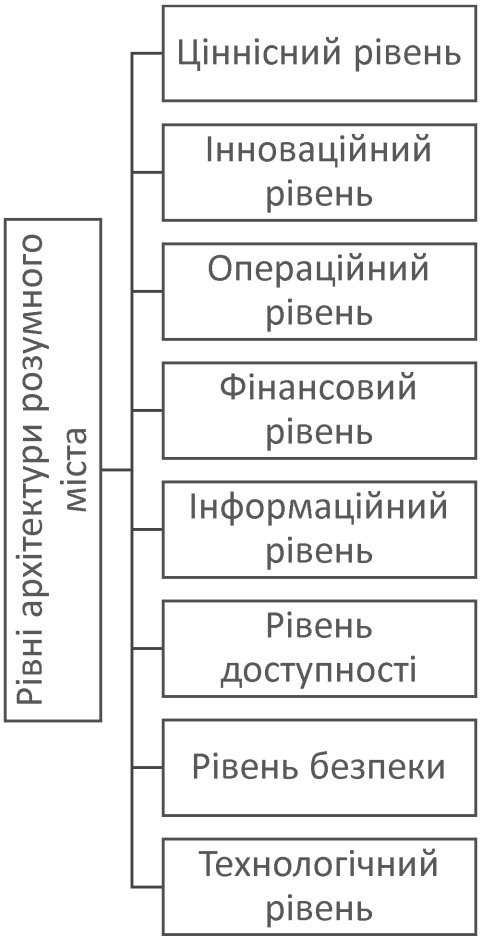

Рисунок 1.3 – Багаторівнева модель розумного міста

Таким чином стає очевидно, що розумне місто – це складна екосистема людей, процесів, політики, технологій та інших факторів, які працюють разом для досягнення ряду результатів. Розумні міста не «виключно належать» містам. Інші творці цінностей також залучені, іноді працюють спільно, а іноді індивідуально. Успішні та стійкі розумні міста використовують програмний підхід для залучення зацікавлених сторін до екосистеми.

Іноді не використовують екосистемний підхід до проектів розумних міст. Частково це пояснюється тим, що проектами «розумного міста» керують організації інформаційних технологій, чиї статути відповідають за розробку та розгортання систем. І навпаки, розумні міста з досвідом керують своїми ініціативами за допомогою внутрішньої міжфункціональної трансформації або інноваційних організацій.

Незалежно від того, на якому етапі розвттку знаходяться міста, на шляху до розумного міста вони повинні залишатися попереду в своїх проектах розумного міста. Вони починають із створення ширшої екосистеми для створення стійких і масштабованих розумних міст. Але тим не менше необхідно послідовно реалізувати декілька обов'язкових кроків.

Спочатку слід адаптувати типову структуру екосистеми розумного міста до реалій конкретного міста. Після цього відбувається оцінка існуючих проектів та нових проектів та ініціативи «розумного міста» в екосистемі. Це дозволяє визначити чого не вистачає в плані проекту, що необхідно для повного успіху проекту та у якій послідовності краще реалізовувати відповідні проекти. Завершальним етапом є визначення пріоритетів та розвиток можливостей на різних рівнях екосистеми. Розумні міста вимагають нових навичок і можливостей. У разі потреби необхідно нарощувати наявні можливості за рахунок стратегічного партнерства та контрактів із постачальниками послуг.

1.3 Функціональні можливості розумних міст

Потенціал розумних міст справді безмежний, і в майбутньому кількість таких міст лише зростатиме. Однак це не єдина сфера, на яку IoT матиме значний вплив у найближчому майбутньому. Ось чому Bi-Intelligence склав звіт про дослідження IoT. Розглянемо основні компоненти розумного міста та їхній вплив у контексті Інтернету речей (рис. 1.4).

Рисунок 1.4 – Функціональні можливості розумного міста

Міста повинні створювати умови для постійного розвитку цифровізації та інтелектуальної інфраструктури. Цифрові технології зараз стають все більш важливими, міська інфраструктура та будівлі повинні проектуватися більш ефективно та стійко. Розумні міста використовують технологічну майстерність для побудови екологічно чистої та енергоефективної інфраструктури, наприклад розумне освітлення повинно світитися лише тоді, коли хтось дійсно проходить повз, наприклад, регулюючи рівні освітлення та контролюючи щоденне споживання електроенергії, щоб зменшити споживання.

Все більшої популярності набувають проекти, пов’язані з дослідженням та прогнозуванням стану атмосферного повітря у мегаполісах. На думку багатьох спецаалістів сучасні технології IoT дозволяють генерувати прогнози стану повітря з точністю до 90 % на період три-п’ять наступних днів. Саме прогнози забруднення атмосферного повітря з вимірюванням ефективності та застосовані технології роблять методологію управління міським повітрям унікальною. Проекція походить від комп’ютерного алгоритму, який використовує штучну нейронну мережу.

Компанія Siemens створила програмний комплекс «The City Air Management Tool», цей інструмент збирає інформацію про забруднення в реальному часі та прогнозує викиди. Інструмент управління міським повітрям – це програмний пакет, який зберігається в хмарі та має інформаційну панель, яка візуалізує інформацію про якість повітря, яку виявляють датчики в місті та прогнозують на наступні три-п'ять днів. Міська влада або громада має можливість вжити різних заходів протягом наступних трьох-п'яти днів з метою недопущення перевищення ГДК забрудників, це сприяє появі нових екологічних регіонів (зони з низьким рівнем викидів)

Відповідальність великих міст із розумними функціями полягає в тому, щоб максимізувати трафік. Через напружений графік у Лос-Анджелесі запроваджено інтелектуальне транспортне рішення, щоб регулювати дорожній рух. Датчики, розташовані на поверхні дороги, миттєво зв'язуються з центральною системою управління дорожнім рухом. Туди передається інформація про транспортний потік і там аналізуються дані. Автоматичне регулювання світлофорів здійснюється за інформацією, надходить від мережі дорожніх датчиків у режимі онлайн. Крім того система використовує історичну інформацію для прогнозування ймовірних місць транспортних потоків — це робиться в автономному режимі.

Інтелектуальні рішення для паркування здатні розпізнавати, коли автомобілі виїхали зі стоянки. Наземні датчики повідомляють на мобільний телефон водія про вільні паркувальні місця. Інші використовують зворотний зв'язок автомобіля, щоб знайти пробоїни та штовхати інші автомобілі по шляху найменшого опору. Таке рішення не лише півищує комфорт мешканців міста, але й сприяє зменшенню заторів та порушень правил паркування. Розумне паркування тепер стало реальністю, і не потребує складної інфраструктури чи значних інвестицій, це ідеальний варіант для міст із середньо-великою чисельністю мешканців.

Рішення щодо управління відходами сприяють підвищенню ефективності збору відходів, знижують витрати, пов'язані з експлуатацією, і покращують

вплив неефективного збору відходів на навколишнє середовище. Відходи з контейнера визначаються рівнеміром; коли він досягне певного порогу, генерується системний запит для вивезення відходів. Очевидно, що сміттєвоз прибуде до заповненого контейнеру, що дозволяє уникнути необхідності очищення половини контейнера.

Майбутнє Інтернету речей невизначене. Воно пропонує рішення в багатьох сферах, включаючи виробництво, моду, ресторани, охорону здоров'я, освіту тощо. Розумні міста можуть використовувати спільну платформу для розумних міст, що особливо вигідно для невеликих міст. Хмарна природа рішень IoT для розумних міст сприяє створенню спільної платформи даних. Малі міста мають потенціал для формування колективної міської екосистеми. У результаті рішення в малих і великих містах підключаються та управляються через центральну платформу, прив'язану до хмари. Зрештою, розмір міста не є перешкодою для того, щоб стати «розумним містом». Кожна група міст має переваги, пов'язані з розумними технологіями.

РОЗДІЛ 2
ВИБІР ТЕХНОЛОГІЙ ЗВ'ЯЗКУ ТА ЗАГАЛЬНОЇ АРХІТЕКТУРИ ВЕБЗАСТОСУНКУ

2.1 Аналіз технологій та протоколів зв'язку елементів системи розумного міста

Система Інтернету речей здатна повноцінно обробляти та транслювати інформацію лише онлайн, тому пристрої в IoT підтримують безперервний зв'язок з комунікаційною мережею. Тому необхідно оцінити чи реально підтимувати постійний зв'язок між усіма пристроями розумного міста і які типи з'єднань доступні для їх спілкування один з одним. Для відповіді на ці питання необхідно проаналізувати сучасні Інтернет-протоколи, оскільки саме вони полегшують цю взаємодію та забезпечують інформаційну безпеку. До цього часу було створено багато протоколів, але розвиток на цьому не припиняється, впроваджуються більш сучасні та безпечніші протоколи.

2.1.1 Bluetooth

Одним із найпоширеніших бездротових протоколів малого радіусу дії є Bluetooth. Протокол забезпечує можливість швидкого підключення до пристроїв, які мають Bluetooth, ця технологія зручна для підключення до смарт-пристроїв. Нещодавно представлений протокол для Bluetooth, який є частиною IoT – протокол Ionic, також відомий як протокол Bluetooth Low Energy. Він забезпечить стандартну комбінацію Bluetooth із меншим енергоспоживанням (рис. 2.1).

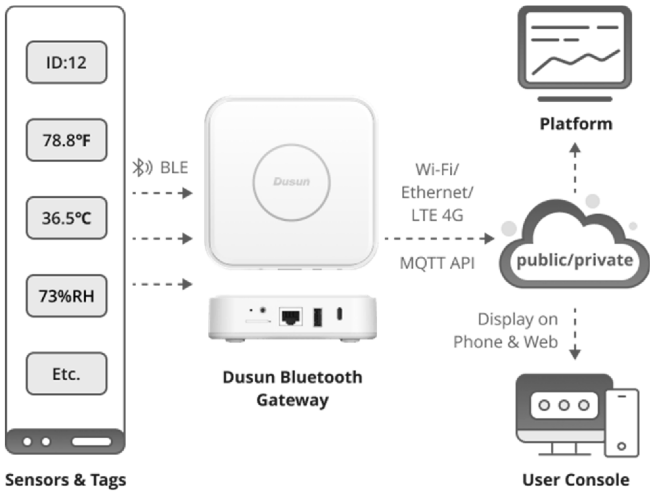

Рисунок 2.1 – Сфера використання Bluetooth у проектах IoT

Важливо пам’ятати, що BLE не призначений для передачі великих обсягів даних і ідеально працює з невеликими порціями інформації. Саме протокол Bluetooth відповідає за більшість протоколів IoT цього століття.

2.1.2 Wi-Fi

Wi-Fi є найпопулярнішою технологією для багатьох розробників комп’ютерів та електроніки. Протокол забезпечує високу швидкість передачі даних, а також здатність передавати великі обсяги інформації. Загальний протокол Wi-Fi 802.11 дозволяє передавати дані зі швидкістю до 800 мегабіт на секунду (рис. 2.2).

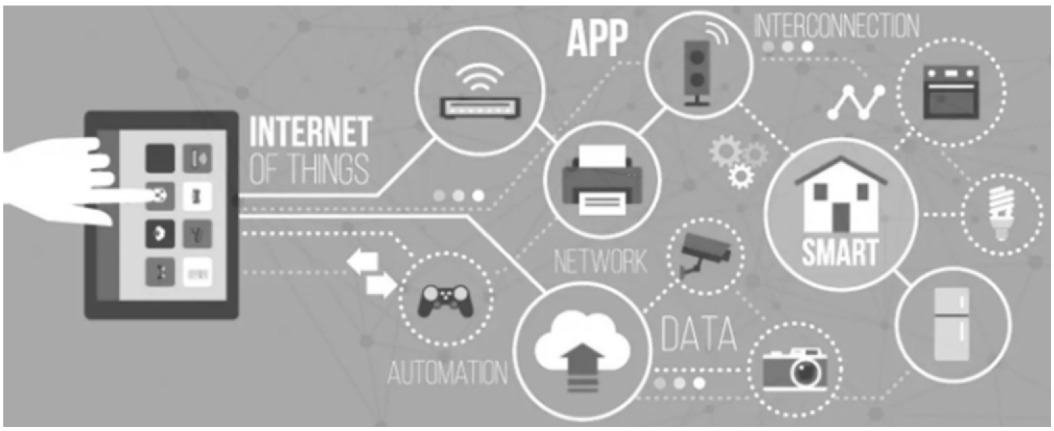

Рисунок 2.2 – Сфера використання Wi-Fi у проектах IoT

Основним відомим недоліком цієї технології у розрізі IoT є те, що вона може споживати велику кількість енергії, що недопустимо для деяких пристроїв IoT. Ефективний радіус Wi-Fi покриття становить близько 50 метрів та забезпечує доступ до хмарної інфраструктури Інтернету речей, а також розробку стандартів для Інтернет-протоколу. Робочі частоти протоколу знаходяться в діапазонах 2,4 ГГц і 5 ГГц.

2.1.3 ZigBee

Подібно до Bluetooth, ZigBee має велику кількість прихильників. Серед різноманітних протоколів Інтернету речей ZigBee призначений насамперед для використання в промисловості, ніж для приватних цілей. Зазвичай він працює на частоті 2,4 ГГц. Ідеально підходить для промислових об'єктів, дані зазвичай передаються невеликими пакетами та легко передаються між будівлями (рис. 2.3).

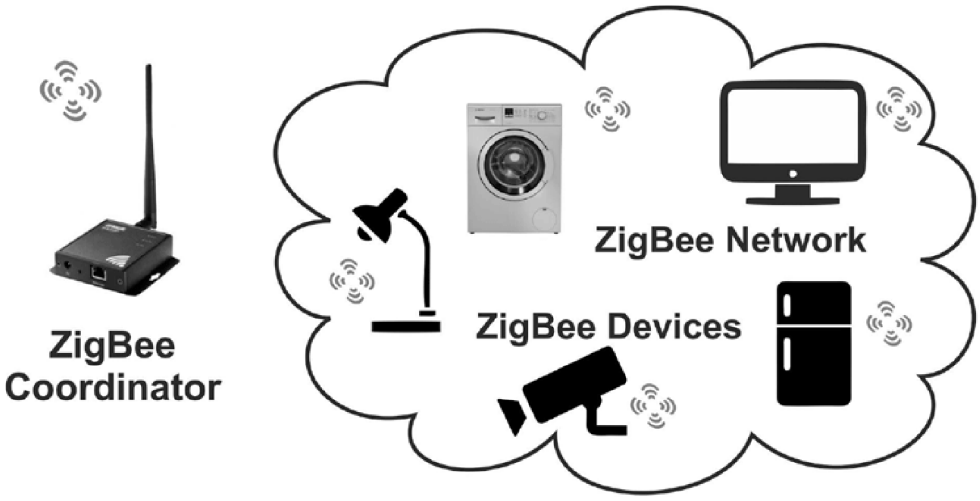

Рисунок 2.3 – Сфера використання ZigBee у проектах IoT

Сьогодні рішення з використанням ZigBee популярні у першу чергу через їх здатність підтримувати безпечні, малопотужні та масштабовані системи з великою кількістю учасників, саме такими є критерії розгортання виробничих інформаційних систем.

2.1.4 MQTT IoT

Протокол MQTT IoT був розроблений у 1999 році Арленом Ніппером (Arcom) та Енді Стенфорд-Кларком (IBM). Він використовується переважно для взаємодії IoT пристроїв з віддаленими територіями. Основною функцією MQTT є незалежне отримання інформації від багатьох електронних пристроїв.

Він часто використовується як заміна протоколу TCP, цей протокол надійний, але простий у передачі даних. Протокол MQTT забезпечує взаємодію трьох основних компонентів або механізмів: підписник, видавець і брокер. Метою видавця є збір інформації та її передача підписнику через брокера. Завданням брокера полягає у забезпеченні безпеки передачі даних та комутації видавця та підписника між собою (рис. 2.4).

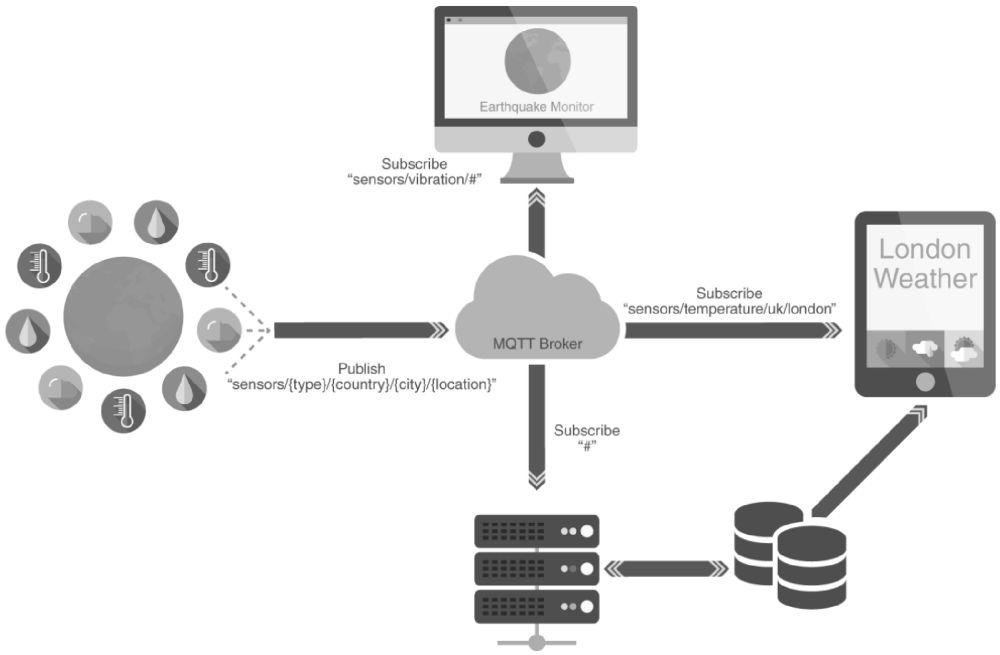

Рисунок 2.4 – Концепція протоколу MQTT

Цей протокол вважається найефективнішим для всіх пристроїв на основі Інтернету речей, він також має можливість надавати достатню інформацію щодо розповсюдження недорогих малопотужних пристроїв через низьку пропускну здатність і вразливу мережу. Ефективне використання технології MQTT передбачає не лише розподіл ролей усіх учасників, але і можливість контролювати процес передачі інформації на усіх етапах.

2.1.5. CoAP

Протокол CoAP, також відомий як Constrained Application Protocol, призначений для роботи розумних пристроїв з обмеженими можливостями. Він охоплює загальні вузли та компоненти мережі, а також різні пристрої, підключені до Інтернету. Системи IoT CoAP, які базуються на HTTP, можна легко поєднати з протоколом CoAP IoT. Він використовує протокол UDP для передачі невеликих пакетів інформації. Подібно до HTTP, він також використовує дизайн RESTful, підтримується мобільними телефонами та інших пристроях, які є фундаментальними для соціальних мереж. CoAP полегшує усунення неоднозначності методів HTTP (рис. 2.5).

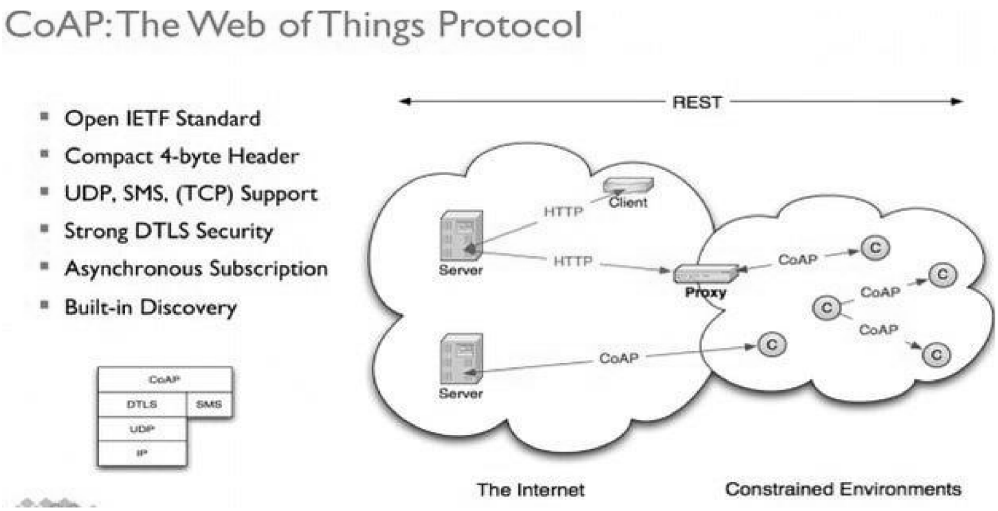

Рисунок 2.5 – Концепція протоколу CoAP

2.1.6. NFC

Одним з популярних протоколів IoT є NFC, який використовує переваги безпечного каналу зв'язку. Протокол NFC або NearField дозволяє клієнтам підключатися до електронних пристроїв, використовувати цифровий контент і проводити безготівкові безконтактні розрахунки. Основне призначення NFC – розширити технологію «безконтактних» карток. Він працює на порівняно невеликих відстанях дозволяючи машинам спілкуватися одна з одною через чіп посередника (рис. 2.6).

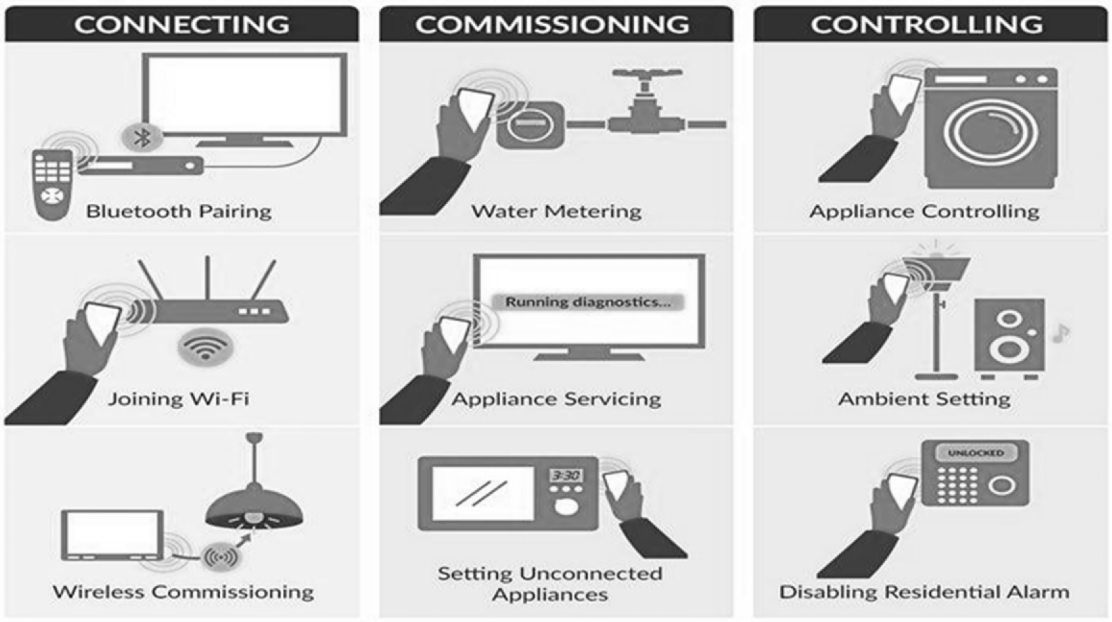

Рисунок 2.6 – Приклади використання NFC у проектах IoT

2.1.7 Sigfox

Sigfox вважається однією з найефективніших альтернативних технологій, яка володіє властивостями як стільникової, так і бездротової технології. Сучасні протоколи для IoT призначені для використання з платформою M2M, тому вони можуть надсилати дані лише на низькому рівні. З технологією Ultra Narrowband (UNB) Sigfox здатний підтримувати швидкість передачі даних до 10 біт на секунду. Для цього потрібно лише 50 мікроват потужності (рис. 2.7).

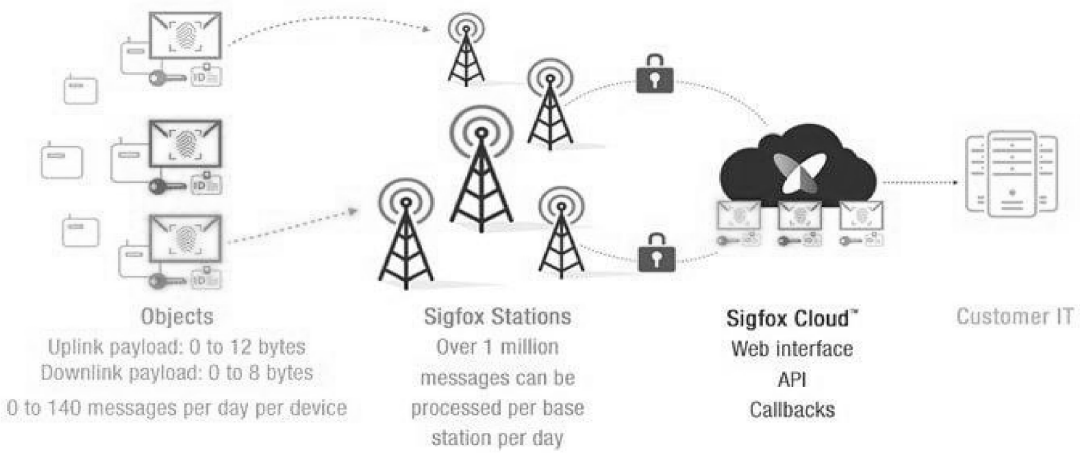

Рисунок 2.7 – Характеристика Sigfox

2.2 Вибір технологій і засобів збереження даних

Дані вважаються одним із найважливіших ресурсів сучасних технологій. Системи керування реляційними базами даних особливо добре підходять для зберігання та отримання структурованих даних із реляційної області, хоча підтримуються й інші типи даних. Реляційні дані — це дані, які зберігаються в різних частинах бази даних (тобто в таблицях). Ці дані, як правило, пов'язані один з одним і зазвичай знаходяться у відношенні «один до одного» або «багато до одного». Кожна таблиця (або асоціація) має набір рядків (записів) і стовпців (полів або атрибутів), які є унікальними (ключі) для кожного рядка.

Часто дані представлені в програмних додатках через об'єкти домену. Ці об'єкти даних (або сутності) не представлені або структуровані таким же чином у програмному забезпеченні, як у реляційній базі даних, хоча загальний підхід полягає в представленні одного об'єкта домену (або сутності) у вигляді рядка в певній таблиці, яка стосується інших об'єктів.

Через реляційну природу даних часто потрібно зберігати та отримувати до них доступ у кількох таблицях, це створює невідповідність між прикладними програмами та уявленнями бази даних про об'єкт, що зазвичай називають невідповідністю об'єктно-реляційного імпедансу.

Ця розбіжність викликана перш за все необхідністю відображення програмних об'єктів у таблиці бази даних на основі реляційних схем. Щоб усунути цю невідповідність, були розроблені різні шаблони архітектури та програмні додатки для перетворення програмних об'єктів у таблиці бази даних і навпаки. Це називається об'єктно-реляційним відображенням.

Системи управління базами даних зазвичай зосереджені на забезпеченні високої узгодженості та продуктивності транзакцій (але не завжди), великої функціональності, стабільності та надійних гарантій надійності. Вони також добре розуміються на складних питаннях і аналітиці не в реальному часі.

Можливості включають відсутність гнучкості моделювання даних, відсутність масштабованості, відсутність доступності, низьку пропускну здатність, низьку продуктивність і стабільність схеми. Стабільність схеми означає складність зміни як схеми бази даних, так і схеми моделі даних (наприклад, таблиць). Ці закономірності рідко точно визначаються заздалегідь і часто потребують розвитку з часом. Вони можуть призвести до тривалих і складних переміщень даних, а також значного ризику помилок і, отже, вищого рівня якості та тестування.

Зрештою, системи управління базами даних також прості в установці, управлінні та використанні (наприклад, складні запити, збережені процедури, тригери тощо). Ці системи традиційно обслуговувались людьми зі спеціалізованими ролями, такими як адміністратор бази даних (DBA) і, меншою мірою, розробники програмного забезпечення.

Системи управління базами даних MySQL вважалася одним з найефективніших рішень для зберігання даних. MySQL досі вважається однією з найпопулярніших баз даних (рис.2.8). Це відкритий код, надійний, сумісний з усіма основними хостинг-провайдерами, економічно ефективний і простий в управлінні. Багато організацій використовують безпеку даних і потужну підтримку транзакцій, які надає MySQL, щоб забезпечити безпеку онлайн-транзакцій і покращити взаємодію з клієнтами.

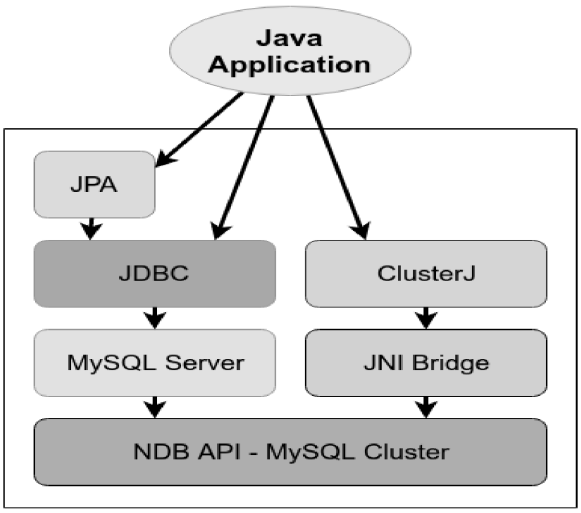

Рисунок 2.8 – Взаємозв’язок сервісів Java та MySQL

Окрім розуміння переваг MySQL як рішення для швидкого зростання, важливо також розуміти і ризики, які це рішення створює для бізнес-операцій (транзакцій). Розглянемо деякі переваги використання технологій MySQL у проектах, пов'язаних із розумним містом.

Транзакції MySQL розглядаються як єдине ціле, що означає, що до тих пір, поки окремі кроки операційного процесу не будуть успішно виконані, транзакція не буде випущена. У результаті, якщо транзакція в будь-який момент буде невдалою, усі наступні транзакції в групі будуть невдалими. На рисунку 2.9 описано етапи транзакції.

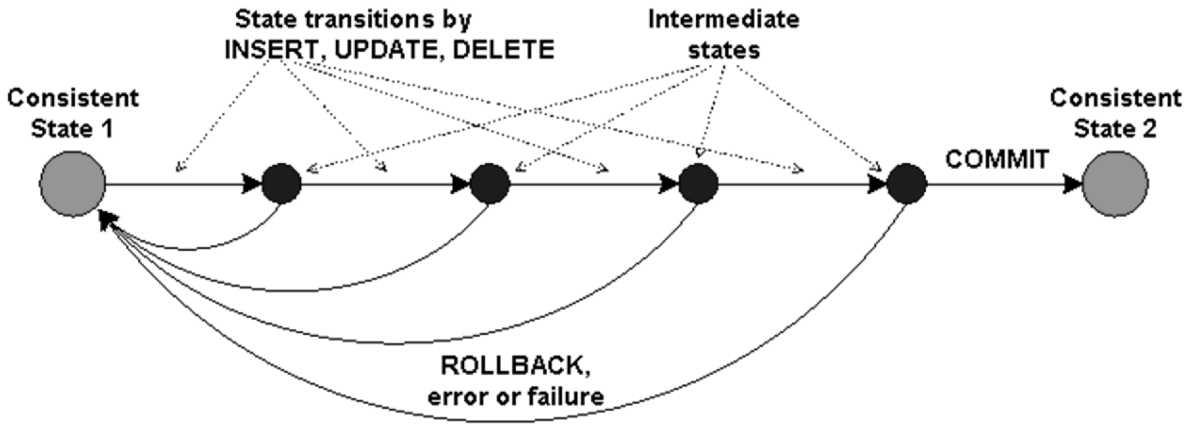

Рисунок 2.9 – Етапи створення MySQL транзакції

Таким чином MySQL гарантує цілісність фінансових операцій, що означає, що клієнти можуть впевнено проводити онлайн-транзакції. Гроші не списуються, доки не буде завершено весь процес, якщо станеться збій на будь-якому етапі система сама повертатиметься до попереднього етапу.

MySQL надає користувачу переваги неперевершеної гнучкості, що дозволяє ефективно керувати глибоко залученими програмами, навіть у великих центрах обробки даних, які містять значні обсяги важливої інформації. Це полегшує повне налаштування системи для задоволення особливих вимог компаній електронної комерції з меншим розміром. MySQL забезпечує найбільшу гнучкість платформи для компаній, які хочуть додати додаткові функції або можливості до своїх серверів баз даних.

Ще одним незмінним атрибутом MySQL є постійна доступність – компанії, які її використовують, можуть користуватися цілодобовим доступом. MySQL постачається з різноманітними кластерними серверами та конфігураціями головний-підлеглий, які забезпечують миттєве безперервне перемикання після відмови. Незалежно від того, чи використовується вебсайт електронної комерції чи система, що швидко розвивається, MySQL створена для роботи з мільйонами запитів і тисячами транзакцій, водночас забезпечуючи унікальні кеші в пам'яті, повнотекстові індекси та чудову швидкодію.

MySQL сприяє безпеці даних завдяки унікальній комбінації функцій захисту даних. Шифрування данихзапобігає перегляду даних без авторизації, і підтримуються протоколи SSH і SSL, що забезпечує більш безпечне з'єднання. MySQL також має потужну систему, яка надає доступ до сервера лише авторизованим користувачем, а також може блокувати користувачів на рівні «людина-машина». Зрештою, функція резервного копіювання даних полегшує відновлення даних у певному тимчасовому місці.

Система MySQL відрізняється вимокою швидкістю, незалежно від основної платформи. Вона має такі функції, як автоматичний перезапуск, розширення простору та автоматичні зміни конфігурації, які спрощують керування. Також міститься повний набір інструментів міграції та повністю завантажений пакет графічного адміністрування. Моніторинг продуктивності MySQL у реальному часі дозволяє швидко усунути неполадки в роботі робочої станції.

Таким чином проведений аналіз показав, що для розробки вебзастосунку «Розумне місто» доцільно використати реляційну базу даних MySQL з відповідною системою керування. При цьому ключовими критеріями є її ефективність, надійності та широке розповсюдження MySQL, що дозволяє забезпечити якісне обслуговування замовників, полегшить пошук розробників та сприятиме збільшенню прибутку.

2.3 Аналіз та вибір архітектури вебзастосунку

Архітектура програмного забезпечення — це формальний дизайн високорівневих компонентів програмної системи. Сьогодні найбільш вдалими вважаються дворівневі та трирівневі архітектурні конструкції.

Дворівнева система є похідною від дизайну клієнт-сервер, коли пряме спілкування відбувається між клієнтом і сервером, підтримується їх безпосередній зв'язок. Завдяки стисненню потоку даних 2-рівнева програма працюватиме швидше. На рис. 2.10 показано конструкцію дворівневої системи без жодного посередника.

Класична дворівнева архітектура складається з двох окремих частин:

– база даних (рівень даних);

– клієнтський додаток (рівень клієнта).

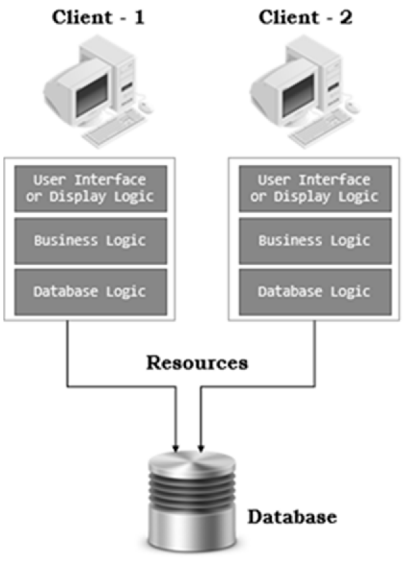

Рисунок 2.10 – Розмежування шарів додатків з дворівневою архітектурою

Код, який зберігає дані на сервері бази даних, знаходиться на стороні клієнта. Клієнт запитує дані від сервера, запит обробляється та повертається клієнту. Основна проблема дворівневої архітектури полягає в тому, що сервер не може адресувати кілька запитів одночасно, що призводить до проблеми

цілісності даних. Через це відображення логіки та інформації про базу даних стає неправильним та/або перезаписується на другому рівні системи клієнт-сервер.

Основними перевагами цього підходу є простота обслуговування та підтримки продукту, а також швидкий зв'язок між сервером і клієнтом. Однак є й значні недоліки, пов'язані з великою кількістю клієнтів: низька продуктивність і економічна невигідність. У зв'язку з цим варто переглянути більш серйозний підхід, який передбачає 3-рівневу структуру.

Трирівнева архітектура програмного забезпечення складається з трьох рівнів логічного мислення. Вони часто використовуються як окрема система клієнта та сервера. 3-рівневі архітектури мають численні переваги у виробничих середовищах і середовищах розробки завдяки модульній структурі інтерфейсу користувача, бізнес-логіки та рівнів зберігання даних. Це дозволяє розробнику оновлювати певну частину програми без залежності від інших частин. Ця додаткова гнучкість може допомогти скоротити загальний час виходу на ринок і цикл розробки системи, дозволяючи командам замінювати або оновлювати окремі рівні без негативного впливу на інші частини системи. На рисунку 2.11 представлено приклад 3-ри рівневої архітектури застосунку.

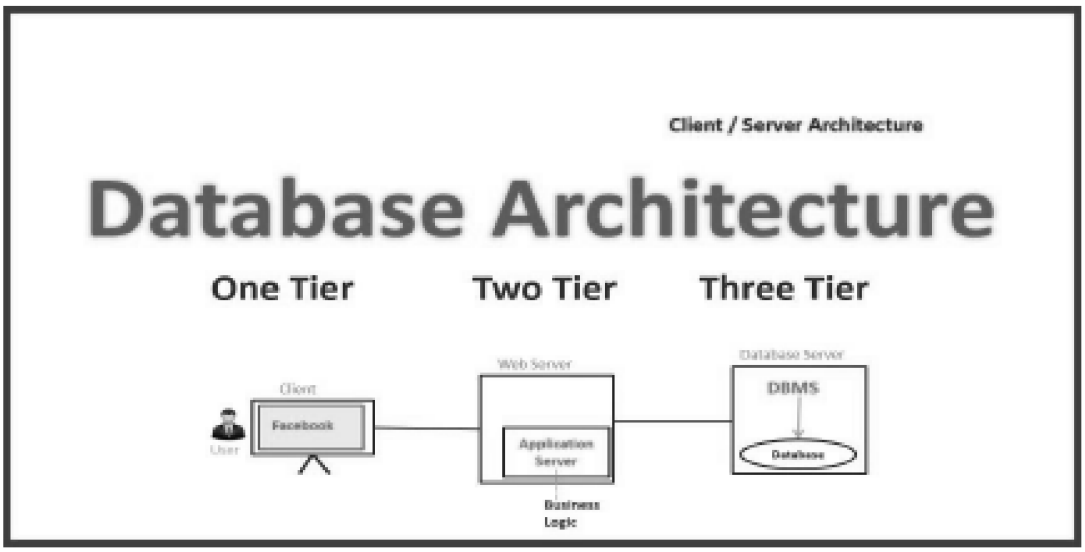

Рисунок 2.11 – Структура трирівневого застосунку з використанням бази даних

Наприклад, інтерфейс користувача вебпрограми можна змінити або модернізувати без негативного впливу на функціональну бізнес-логіку та логіку доступу до даних. Цей архітектурний стиль часто є вигідним для інтеграції стороннього програмного забезпечення в існуючу програму. Ця гнучкість інтеграції також робить його ідеальним для інтеграції аналітики в існуючі програмні програми, і зазвичай використовується для цієї мети постачальниками вбудованої аналітики. 3-рівневі структури часто використовуються в хмарних або локальних програмах, а також програмах, що працюють за принципом (SaaS).

Дизайн складається з трьох окремих рівнів:

– Рівень презентації – це рівень, який є найближчим до користувача в 3-рівневій системі, цей рівень складається з інтерфейсу, який призначений для нього. Інтерфейс користувача зазвичай графічний і доступний через веббраузер або вебдодаток, він відображає вміст і інформацію, які є цінними для кінцевого користувача. Цей рівень зазвичай створюють на основі таких технологій, як HTML, CSS, JavaScript або інших популярних платформ веброзробки. Він взаємодіє з іншими рівнями через запити API.

Серверний рівень програми містить функціональну бізнес-логіку, яка керує основними можливостями програми. Його часто пишуть на Java, Net, C#, Python і C++.

Рівень доступу до даних складається з бази даних і системи зберігання даних, а також рівня доступу до даних. Прикладами такого типу систем є MySQL, Oracle, PostgreSQL, Microsoft SQL Server, MongoDB та інші.

Переваги такого способу створення програми значно збільшуються. Підвищується продуктивність системи. Диференціюючи різні шари, можна досягти індивідуального масштабу для кожного з них залежно від поточної потреби. Наприклад, якщо додаток отримує багато запитів з Інтернету, але небагато з них безпосередньо впливають на рівень додатків то можна збільшити кількість вебсерверів, не впливаючи на сервери додатків. Подібним чином, якщо надходить багато великих запитів на застосування лише від

кількох користувачів то можна збільшити розмір програм і рівнів даних без взаємодії з вебсерверами. Це полегшує балансування навантаження кожного рівня на індивідуальній основі, що підвищує загальну продуктивність, використовуючи найменшу кількість ресурсів. Крім того, модульний підхід до обробки різних рівнів пропонує численні варіанти розгортання. Наприклад, можна вибрати розміщення вебсерверів у загальнодоступній або приватній хмарі, а програми та рівні даних можна розмістити на сайті.

Використовуючи окремі рівні можна підвищити надійність і доступність програми, розмістивши різні частини програми на різних серверах і використовуючи кешовані результати. Архітектура аплікації Розумне місто зозроблена за трирівневою архітектурою представлена на рисунку 2.12.

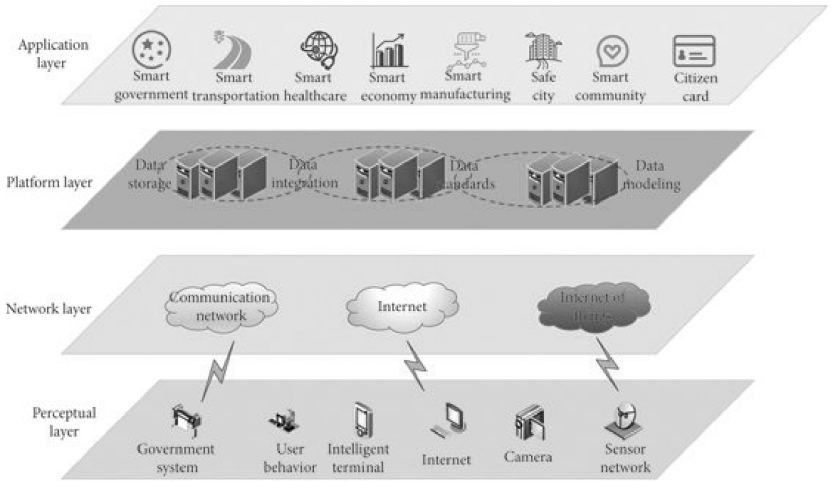

Рисунок 2.12 – Архітектура застосунку Розумне місто

2.4 Порівняння форматів передачі даних

Сьогодні найпопулярнішими форматами передачі даних є XML і JSON. XML — це мова програмування, розроблена Консорціумом Всесвітньої павутини (W3C) для опису синтаксису для кодування документів, який можуть зрозуміти як люди, так і машини. Це робиться за допомогою тегів, які описують

склад документа та спосіб його створення та транспортування. Його можна порівняти з мовою гіпертекстової розмітки HTML, яка використовується для кодування вебсторінок. У HTML використовується заздалегідь розроблений набір символів, які описують передбачуваний формат вмісту вебсайту. На рис. 2.13 продемонстровано приклад XML документу.

```
<?xml version="1.0" encoding="iso-8859-9" standalone="yes" ?>
<CURRENCIES>
  <LAST_UPDATE>2024-09-29</LAST_UPDATE>
  <CURRENCY>
    <NAME>dollar</NAME>
    <UNIT>1</UNIT>
    <CURRENCYCODE>USD</CURRENCYCODE>
    <COUNTRY>USA</COUNTRY>
    <RATE>4.527</RATE>
    <CHANGE>0.044</CHANGE>
  </CURRENCY>
  <CURRENCY>
    <NAME>euro</NAME>
    <UNIT>1</UNIT>
    <CURRENCYCODE>EUR</CURRENCYCODE>
    <COUNTRY>European Monetary Union</COUNTRY>
    <RATE>5.4417</RATE>
    <CHANGE>-0.013</CHANGE>
  </CURRENCY>
</CURRENCIES>
```

Рисунок 2.13 – Приклад представлення даних у XML форматі

До основних переваг цього формату можна віднести те, що XML не залежить від платформи та мови програмування, тобто його можна використовувати в будь-якій системі. Також XML підтримує складні символи використовуючи Unicode — це універсальна система кодування, яка призначена для використання з різними мовами та системами письма, згідно з якою кожній літері, цифрі чи символу присвоюється унікальне числове значення, яке використовується різними комп'ютерами та програмами. Цей атрибут дозволяє XML передавати будь-яку інформацію, написану будь-якою людською мовою.

Дані, що зберігаються та транспортуються через XML, можна будь-коли змінити без негативного впливу на представлення даних. Як правило, для представлення інформації використовується інша мова програмування, наприклад HTML. HTML бере дані з файлу XML і відображає їх у графічному

інтерфейсі користувача, коли інформація оновлюється у файлі XML, вона відображається в HTML без зміни графічного інтерфейсу користувача. Крім того XML сприяє простій передачі даних між різними системами через свою незалежність від платформи. XML-дані не потребують жодного перетворення під час переміщення між різними комп'ютерами.

Проте негативні моменти є суттєвими. Наприклад, синтаксис XML є надто багатослівним і має вищий ступінь надмірності, ніж інші типи текстових даних. Додаткові витрати на резервування синтаксису призводять до більшого навантаження для зберігання та транспортування великого обсягу даних. Вважається, що документ XML менш зрозумілий порівняно з іншими типами текстових даних, наприклад JSON, він не не підтримує масив, а обсяг файлу XML зазвичай досить великий через його багатослівний характер, розмір повністю залежить від його автора.

Іншим популярним та сучасним форматом передачі даних є JSON. Нотація об'єктів JavaScript (JSON) – це поширений формат для запису структурованих даних, який базується на синтаксисі об'єктів JavaScript. Зазвичай він використовується для передачі інформації в вебдодатках (наприклад, для передачі деяких даних із сервера до клієнта, щоб відобразити їх на вебсторінці, або навпаки). З появою вебсайтів із підтримкою AJAX стає все більш важливим, щоб вебсайти могли завантажувати дані швидко й асинхронно або у фоновому режимі, не затримуючи процес візуалізації сторінки.

Перевагою JSON порівняно з XML є простий для розуміння синтаксис. Можна використовувати лише простий для розуміння синтаксис, який виконується швидше. Синтаксичний аналіз даних на стороні сервера також є суттєвою перевагою. Якщо аналіз на стороні сервера є швидким, тоді лише користувач отримає відповідь на своє запитання швидко. У цьому випадку аналіз на стороні сервера JSON є значною перевагою, яка обгрунтовує його використання на стороні сервера.

JSON має широкий діапазон сумісних браузерів та операційних систем, тому додатки, створені за допомогою протоколу JSON, не потребують підтримки браузера. Створюючи новий проект, розробник розглядає різні браузери, але JSON надає таку можливість. Крім того спосіб обміну даними формату JSON дозволяє працювати з будь-якою інформацією, включаючи аудіо, відео та інші носії. Це пояснюється тим, що JSON зберігає інформацію в масивах, тому передача даних є простішою. Як наслідок, JSON є вигідним як формат файлу для вебінтерфейсів API та веброзробки. На рисунку 2.14 продемонстровано приклад JSON документу.

```
{
  "orderID": 12345,
  "shopperName": "vv",
  "shopperEmail": "vv@example.com",
  "contents": [
    {
      "productID": 34,
      "productName": "Super product",
      "quantity": 1
    },
    {
      "productID": 56,
      "productName": "Wonderful product",
      "quantity": 3
    }
  ],
  "orderCompleted": true
}
```

Рисунок 2.14 – Приклад представлення даних у JSON форматі

У результаті було прийнято рішення використовувати цей формат для передачі даних.

РОЗДІЛ 3
РОЗРОБКА ВЕБЗАСТОСУНКУ «РОЗУМНЕ МІСТО»

3.1 Методологія UML аналізу програмного забезпечення

Щоб візуалізувати майбутню взаємодію між системою та користувачем, ще на етапі розробки програмного забезпечення часто використовують концепцію UML. UML (уніфікована мова моделювання) – це поширена мова моделювання, яка об’єднує діаграми, призначені для допомоги розробникам програмного забезпечення у створенні, візуалізації, конструюванні та документуванні артефактів програмного забезпечення, а також у бізнес-моделюванні та інших сферах, не пов’язаних із програмним забезпеченням. UML – це набір ефективних інженерних принципів, які продемонстрували свою ефективність у моделюванні великих і складних систем. UML має вирішальне значення для об’єктно-орієнтованого підходу до розробки програмного забезпечення та процесу розробки програмного забезпечення. В UML графічні методи в основному використовуються для опису прогресу проектів програмного забезпечення. Складні програми вимагають співпраці та планування з боку кількох команд, як наслідок, їм потрібен чіткий і стислий метод спілкування між собою. Бізнесмени не розуміють коду. Як наслідок, UML необхідний для зв’язку з некритичними вимогами, функціями та процесами в системі.

Коли команди можуть візуалізувати процеси, взаємодію користувачів і статичну структуру системи, вони значно скорочують необхідний час. UML асоціюється з об'єктно-орієнтованим проектуванням і аналізом. UML використовує компоненти та зв’язки між ними для створення діаграм. Використання UML полегшує спілкування між проектами та дослідження потенційних дизайнів, а також перевірку дизайну програмного забезпечення.

Найважливішим компонентом UML є візуальне представлення (рис. 3.1), яке полегшує документування, проектування, візуалізацію та специфікацію артефактів програмного забезпечення об'єктно-орієнтованого типу.

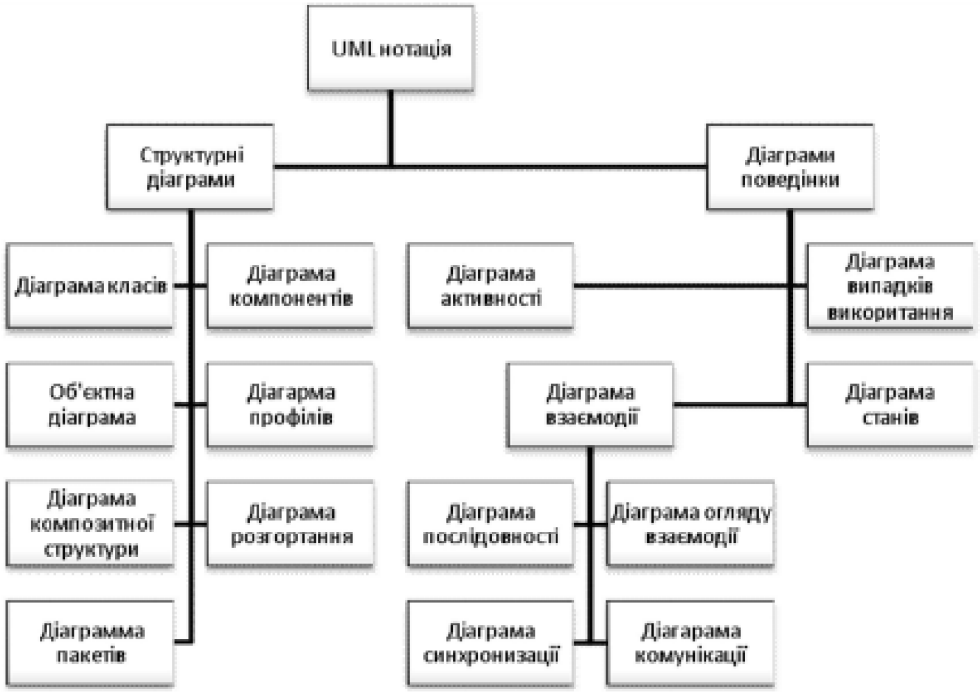

Рисунок 3.1 – Типова структура графічної нотації UML

3.2 UML аналіз вебзастосунку Розумне місто

3.2.1 Опис моделей користувачів та їх ролей

Основним компонентом бази даних є таблиця. У базі даних, яка базується на відношеннях, термін «таблиця» не визначений і зазвичай відноситься до візуального зображення відношення на папері чи екрані.

Відповідно до опису завдання, розроблена програмна система має функціональність для трьох різних ролей: звичайного користувача, системного адміністратора та супервізора. Кожна з них присвячена певній групі привілеїв та обмежень. Фігура UML на рисунку 3.2 ілюструє, які дії зможе виконувати кожен користувач і які привілеї йому потрібні.

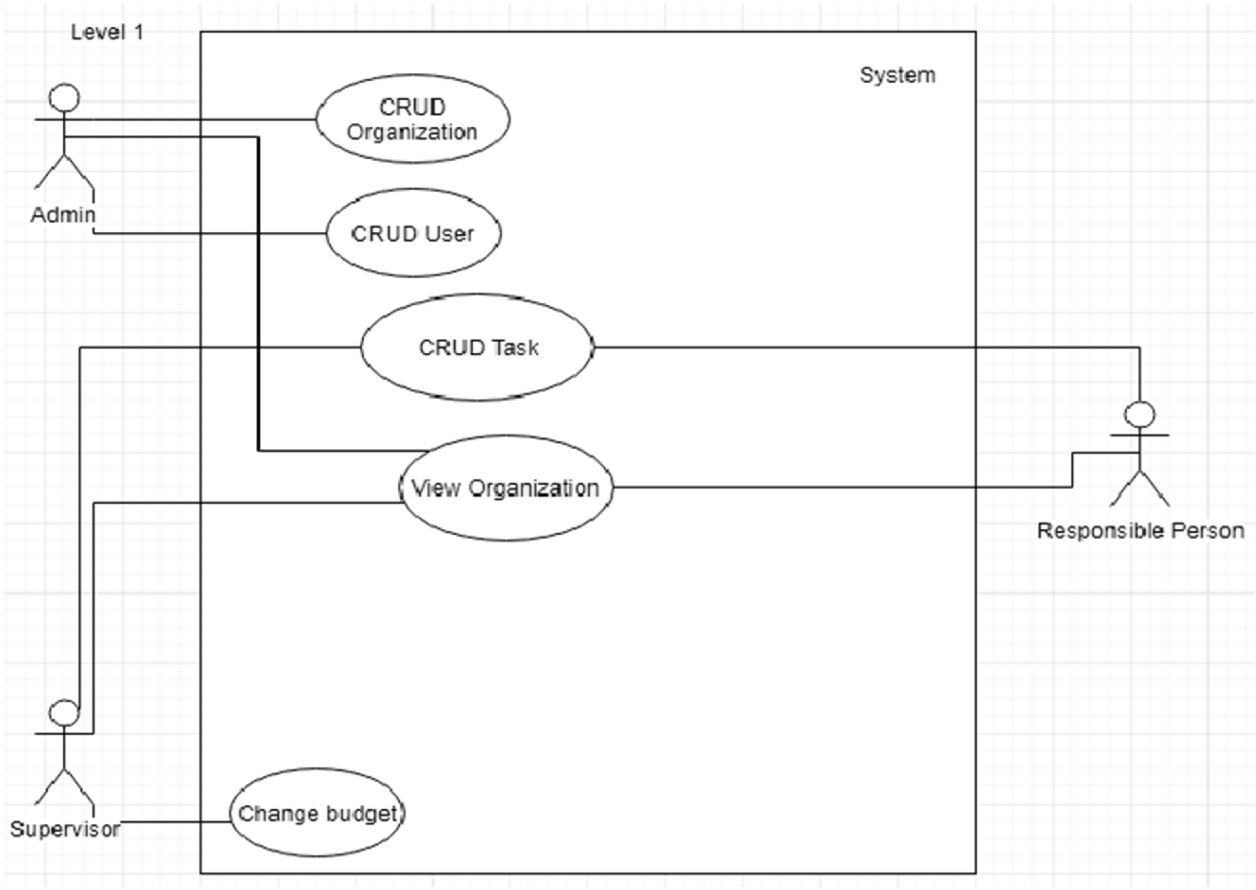

Рисунок 3.2 – UML діаграма користувачів вебзастосунку розумне місто

З наведеної діаграми можна зробити висновок, що ролі користувачів системи мають такі функції:

1) ADMIN:

– може змінювати назву організації;

– може додавати організації;

– може створювати облікові записи іншим користувачам;

– може редагувати облікові записи користувачів;

– може видаляти облікові записи;

– може надавати ролі Supervisor або Admin іншим користувачам;

– може призначати користувача як відповідального (Responsible person) за деяку службу;

– може видаляти організації.

2) SUPERVISOR:

– бачить задачі всіх служб та стан їх виконання;

– може створювати задачі для різних служб;

– може редагувати вже існуючі задачі, зокрема міняти бюджет і термін виконання.

3) RESPONSIBLE PERSON:

– може переглядати задачі служби;

– змінює стани задач (очікує виконання, виконується, виконано) (в межах служби, за яку він відповідальний);

– може додавати задачі (в межах служби, за яку він відповідальний).

Таблиця «КОРИСТУВАЧІ», містить усі поля таблиці. Ця кількість полів дозволяє зберігати інформацію, необхідну для точного опису інформації про користувача.

– IDENTIFICATOR – початковий номер запису в таблиці, який однозначно ідентифікує конкретного користувача. Усі таблиці в базі даних мають цей стовпець, тому в усіх випадках властивість первинного ключа призначена для цього поля.

Створення у кожній базі даних стовпцю «ідентифікатор», сприяє зменшуємо кількість накладних операційних витрат, що даємо змогу ефективно вносити необхідні зміни.

– ACTIVE – зберігає інформацію про те чи користувач досі активний(працює);

– PASSWORD – зашифрований пароль користувача;

– NAME, SURNAME та PHONE_NUMBER – прізвище, ім’я та мобільний номер телефону для ідентифікації;

– EMAIL – електронна пошта;

– CREATED_DATE – визначає дату реєстрації в системі;

– UPDATED_DATE – зберігає дату останньої зміни особистої інформації в системі.

Крім того, деякі поля в цій таблиці служать вторинними ключами для інших пов’язаних таблиць, наприклад «USERS_ORGANIZATIONS», «SEENCOMMENTS» і «USERS_ROLES».

Таблиця USER_ORGANIZATIONS (організації користувачів) (рис. 3.3) є представленням пари користувачів і організації, тобто конкретний екземпляр, який вказує, що користувач є частиною певної організації.

Рисунок 3.3 – Таблиця USER_ORGANIZATIONS (організації та користувачі)

Таблиця USERS_ROLES на рисунку 3.4 схожа на попередню таблицю, єдина відмінність полягає в тому, що тепер у ній є пара ролей для кожного користувача.

Рисунок 3.4 – Таблиця USERS_ROLES (ролей користувачів)

Таблиця SEENCOMMENTS (переглянуті коментарі) описує користувача та пов'язані повідомлення. Це важливо для збереження інформації про те, чи прочитав конкретний користувач певне повідомлення. Це показано на рис. 3.5.

Рисунок 3.5 – Таблиця SEENCOMMENTS (переглянутих коментарів)

3.2.2 Організаційна структура

Таблиця «Організації» є однією з первинних таблиць, яка містить інформацію щодо первинної одиниці системної інформації — організацій, які відповідають за її управління. Містить такі поля:

– ID – унікальний ідентифікатор організації;

– CREATED_DATE – дата створення;

– ADDRESS – адреса організації;

– NAME – назва організації;

– UPDATED_DATE – дата останнього редагування.

3.2.3 Моделі завдань та коментарів системи

Кожна організація матиме певний перелік обов'язків, які виконуватимуться, кожен обов'язок матиме певний статус, особу, яка за це відповідає та коментує, конкретне потенційне обговорення, роз'яснення між керівником та виконавчою особою. Для збереження інформації про завдання та коментарі учасників були створені таблиці «Завдання» та «Коментарі». Таблиця «Завдання» має такі реквізити:

– ID – унікальний ідентифікатор завдання;

– UPDATED_DATE – дата останнього редагування;

– APPROVED_BUDGET – остаточно затверджений кошторис на завдання;

– BUDGET – кошти, які виділили під завдання;

– DESCRIPTION – опис завдання;

– TASK_STATUS – стан завдання;

– TITLE – заголовок до завдання;

– DEADLINE_DATE – дата завершення;

– CREATED_DATE – дата створення;

– TASK_STATUS – вторинний ключ таблиці юзер-організація, що означає причетність завдання до конкретної організації.

Таблиця «Коментарі» містить такі реквізити:

– ID – унікальний ідентифікатор коментаря;

– USER_ID – вторинний ключ, визначає причетність конкретного користувача до коментаря;

– UPDATED_DATE – дата останнього редагування;

– CREATED_DATE – дата створення;

– DESCRIPTION – текст коментаря;

– TASK_ID – також вторинний ключ, визначає причетність коментаря до конкретного завдання.

3.2.4 Структура виявлених взаємодії

Оскільки у міста є специфічний бюджет, ми будемо розглядати його як окрему, самостійну структуру. Тому в таблиці «Бюджет» буде лише одне поле, присвячене фінансуванню міст, це поле «фонд».

З міського бюджету будуть виділені кошти на підтримку конкретних організацій. Угода в цілому в цьому плані є договором між містом і організацією. Для ведення інформації про ці події створено таблицю «Операції», у цій інформації описуватиметься переказ коштів певній організації. Поля таблиці:

– ID – унікальний ідентифікатор транзакції;

– TRANSACTION_BUDGET – конкретна сума грошей на завдання;

– CURRENT_BUDGET – поточний стан бюджету (грошова вартість) під час створення транзакції;

– TASK_ID – вторинний ключ відповідає за конкретне завдання, пов’язане з транзакцією;

– CREATED_DATE – дата проведення транзакції.

3.3 Розробка серверної частини

Для стабільної та швидкої роботи сервера було вирішено скористатися фреймворком Spring. Spring Framework надає комплексну модель програмування і конфігурації для сучасних корпоративних додатків на базі Java – на будь-якій платформі розгортання.

Ключовим елементом Spring є інфраструктурна підтримка на прикладному рівні: Spring фокусується на техніці корпоративних додатків, щоб команди розробників могли зосередитися на бізнес-логіці прикладного рівня без непотрібних зв'язків з конкретними середовищами розгортання.

Основна перевага Spring – можливість розробки програми як набору слабозв’язаних компонентів. Чим менше компоненти програми знають один про одного, тим простіше розробляти новий і підтримувати існуючий функціонал програми. Класичним прикладом є управління транзакціями. Spring дозволяє повністю відокремити управління транзакціями від основної логіки взаємодії з базою даних. Будь-які зміни в цій логіці не повинні порушувати транзакційність так само, як зміна логіки керування транзакціями не порушує логіку програми. Компоненти можна додавати і видаляти (майже) незалежно один від одного. В принципі, додаток можна розробити таким чином, що він навіть не буде знати, що управляється Spring'ом. Також Spring помітно спрощує

модульне тестування: в компонент, розроблений для роботи в контейнері дуже легко вводити «замкнені» залежності і перевірити роботу тільки цього компонента. І тому можна сформулювати такі переваги:

– забезпечення слабкої зв'язаності компонентів;
– спрощення ініціалізації і налаштування компонентів;
– спрощення модульного тестування;
– спрощення розробки та підтримки програми в цілому.

Тож, так як Spring був розроблений як альтернатива EJB прямо з самого початку, і пропонує багато можливостей, що полегшують розробку, використання цього фреймворку значно покращить якість розробки.

3.3.1 Проектування процесу автентифікації

Вхід до аплікації починається з процесів автентифікації та авторизації. Для забезпечення авторизації та входу в систему «Розумне місто» використано технології генерації токенів для користувачів JWT (рис. 3.6) та фреймворк Spring Security.

Дуже важливо розуміти, що використання JWT не приховує і не маскує дані автоматично. Причина, чому JWT використовуються – це перевірка, що відправлені дані були дійсно відправлені авторизованим джерелом. Дані всередині JWT закодовані і підписані, це не одне і теж, що зашифровані. Мета кодування даних – перетворення структури. Підписані дані дозволяють одержувачу даних перевірити автентифікацію джерела даних. Таким чином закодування і підпис даних не захищає їх. З іншого боку, головна мета шифрування – це захист даних від неавторизованого доступу.

Замість запуску процесу автентифікації шляхом перенаправлення на сторінку входу, коли клієнт запитує захищений ресурс, сервер REST автентифікує всі запити, використовуючи дані, доступні в самому запиті, в даному випадку токен JWT. Якщо така аутентифікація не вдається, перенаправлення не має сенсу. REST API просто надсилає HTTP - код 401

(несанкціонований запит), і клієнти повинні знати, що робити; наприклад, браузер покаже динамічний div, щоб дозволити користувачеві ввести ім'я користувача та пароль. З іншого боку, після успішної аутентифікації на класичних багатосторінкових вебсайтах користувач перенаправляється за допомогою HTTP команд, як правило, на домашню сторінку або на сторінку, яку користувач спочатку запросив.

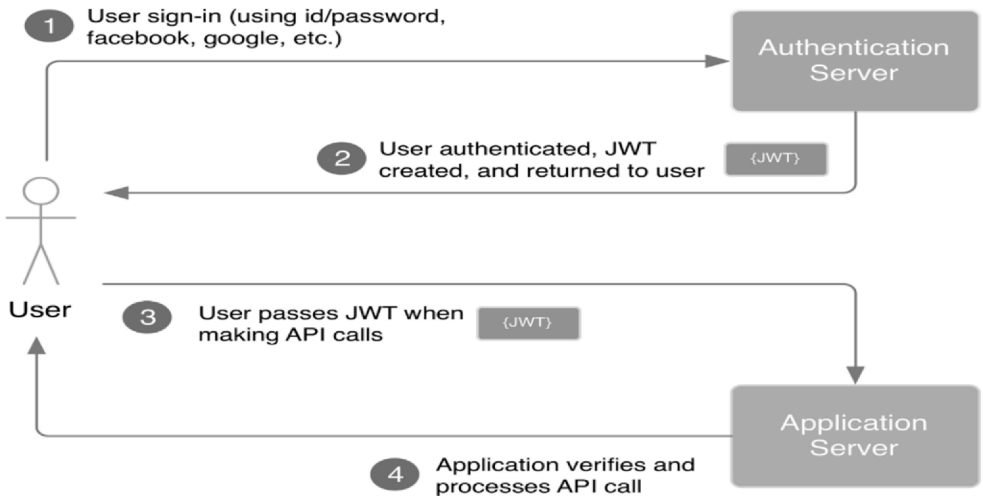

Рисунок 3.6 – Приклад використання JSON Web Tokens

Замість цього ми просто продовжимо запит так, ніби ресурс взагалі не захищений, повернемо http-код 200 (OK) і очікуване тіло відповіді. Можна використати симетричний алгоритм SHA-256 для забезпечення хешування токенів JWT. Сторінка автентифікації системи з отриманим маркером JWT із сервера зображена на рисунку 3.7.

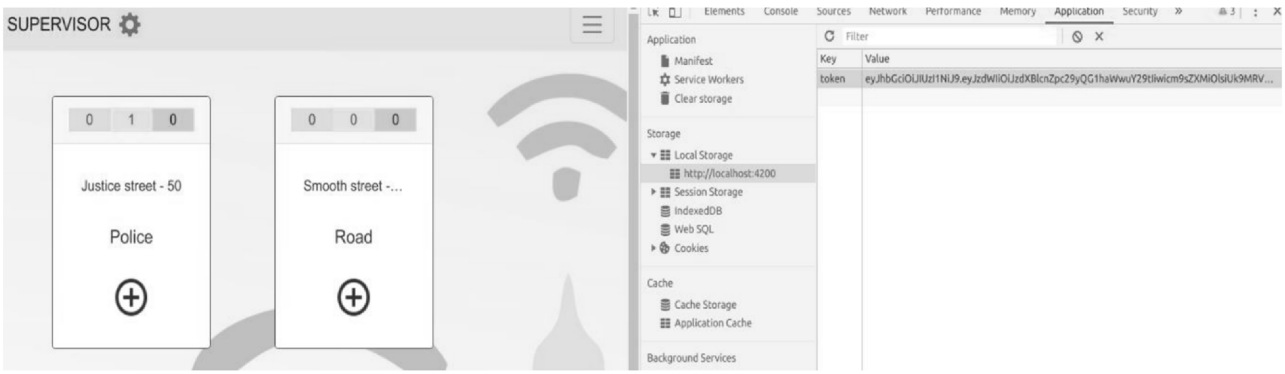

Рисунок 3.7 – Збережений JSON Web Tokens

Аутентифікація JWT реалізована за допомогою мови Java. Створено три класи: JWT Token Filter, JWT Token Provider, JWT TokenConfigurer. Відповідний код наведено на рисунках 3.8-3.11.

```
// Клас JwtConfigurer розширює SecurityConfigurerAdapter для налаштування безпеки в Spring Security.
public class JwtConfigurer extends SecurityConfigurerAdapter<DefaultSecurityFilterChain, HttpSecurity> {
    // Поле для зберігання провайдера токенів JWT.
    private JwtTokenProvider jwtTokenProvider;
    // Конструктор для ініціалізації JwtTokenProvider.
    public JwtConfigurer(JwtTokenProvider jwtTokenProvider) {
        this.jwtTokenProvider = jwtTokenProvider;
    }
    // Перевизначення методу configure для налаштування HttpSecurity.
    @Override
    public void configure(HttpSecurity http) {
        // Створюємо екземпляр фільтра JWT із переданим провайдером токенів.
        JwtTokenFilter customFilter = new JwtTokenFilter(jwtTokenProvider);
        // Додаємо кастомний фільтр перед стандартним UsernamePasswordAuthenticationFilter.
        http.addFilterBefore(customFilter, UsernamePasswordAuthenticationFilter.class);
    }
}
```

Рисунок 3.8 – Клас конфігурації JSON Web Tokens

```
// Метод doFilter обробляє запити, перевіряючи валідність JWT токена.
public void doFilter(ServletRequest req, ServletResponse res, FilterChain filterChain)
        throws IOException, ServletException {
    try { // Приводимо запит до HttpServletRequest для роботи з HTTP-запитами.
        HttpServletRequest httpServletRequest = (HttpServletRequest) req;
        // Отримуємо JWT токен із запиту за допомогою jwtTokenProvider.
        String token = jwtTokenProvider.resolveToken(httpServletRequest);
        // Перевіряємо, чи токен існує та чи є він валідним.
        if (token != null && jwtTokenProvider.validateToken(token)) {
            // Отримуємо поточну аутентифікацію з SecurityContextHolder.
            Authentication authentication =
SecurityContextHolder.getContext().getAuthentication();
            // Якщо аутентифікація відсутня, отримуємо її з токена.
            if (authentication == null) {
                authentication = jwtTokenProvider.getAuthentication(token); } }
        filterChain.doFilter(req, res);
    } catch (InvalidJwtAuthenticationException exception) {
        // У разі помилки, встановлюємо статус відповіді 401 (UNAUTHORIZED).
        ((HttpServletResponse)
res).setStatus(HttpServletResponse.SC_UNAUTHORIZED);
        // Формуємо JSON-повідомлення про помилку.
        ObjectMapper objectMapper = new ObjectMapper();
        String message = objectMapper.writeValueAsString (ExceptionResponse.builder()
                .url(((HttpServletRequest) req).getRequestURI()) // Додаємо URL.
                .message("Token is invalid or expired!") // Додаємо текст помилки.
                .build());
        res.getWriter().write(message);}} // Відправляємо повідомлення у відповідь.
```

Рисунок 3.9 – Метод фільтрації запиту (JWT Token Filter)

```
// Метод для створення JWT токена з вказаним іменем користувача та ролями.
public String createToken(String username, List<String> roles) {
 // Створюємо об'єкт Claims і встановлюємо ім'я користувача як суб'єкт токена.
    Claims claims = Jwts.claims().setSubject(username);
    // Додаємо список ролей до токена як додаткову інформацію.
    claims.put("roles", roles);
    // Поточна дата і час — час створення токена.
    Date now = new Date();
    // Визначаємо дату і час закінчення терміну дії токена.
    Date validity = new Date(now.getTime() + Long.parseLong(validityInMilliseconds));
    // Формуємо JWT токен, використовуючи вказані claims, час створення, час закінчення і секретний ключ.
    return Jwts.builder()
            .setClaims(claims)
            // Додаємо claims до токена.
            .setIssuedAt(now)
            // Встановлюємо час створення токена.
            .setExpiration(validity)
            // Встановлюємо час закінчення дії токена.
            .signWith(SignatureAlgorithm.HS256, secretKey)
            //Підписуємо токен з використанням алгоритму HS256 і секретного ключа.
            .compact();
            // Генеруємо закодований токен у вигляді строки.}
```

Рисунок 3.10 – Метод генерації токену

```
// Метод для перевірки валідності JWT токена.
public boolean validateToken(String token) {
   try {
      // Парсимо токен з використанням секретного ключа для перевірки підпису.
      Jws<Claims> claims = Jwts.parser()
            .setSigningKey(secretKey) // Встановлюємо секретний ключ для верифікації токена.
            .parseClaimsJws(token); // Розбираємо токен і отримуємо claims.
      // Перевіряємо, чи не закінчився термін дії токена.
      if (claims.getBody().getExpiration().before(new Date())) {
         return false; // Якщо термін дії минув, токен невалідний.
      }
      return true; // Якщо токен валідний, повертаємо true.
   } catch (JwtException | IllegalArgumentException e) {
      // Якщо токен невалідний або виникла помилка, кидаємо виняток із повідомленням.
      throw new InvalidJwtAuthenticationException("Expired or invalid JWT token"); } }
```

Рисунок 3.11 – Метод валідації JSON Web Tokens

3.3.2 Шаблон об’єкта доступу до даних та сервісів

Шаблон об’єкта доступу до даних (DAO) — це структурний шаблон, який ізолює прикладний/бізнес-рівень від рівня збереження. Зазвичай це реляційна база даних, однак також можна використовувати будь-який інший механізм збереження, шляхом абстрагування та інкапсуляції всього доступу до постійного сховища на рівні доступу до даних.

Функція цього API полягає в абстрагуванні всіх складнощів виконання операцій CRUD у базовому механізмі зберігання даних від програми. Це дозволяє обом шарам розвиватися окремо, не знаючи жодної специфіки іншого. На рис. 3.12 зображено архітектуру програми з використанням рівня DAO.

Тож в застосунку «Розумне місто» кожна сутність має свій клас, що описує доступ до бази даних. Це такі сутності як:

– Task – таблиця з інформацією про завдання до конкретної організації;

– Organizations – містить дані про ключову одиницю інформації системи, організації, які будуть керуватись відповідальними користувачами;

– UsersOrganizations – відображає зв'язок багато до багатьох між таблицями користувачів та організацій;

– Budget – бюджет системи;

– Comments та SeenComments – інформація про коментарі до певних завдань;

– Users – містить дані про користувачів системи;

– User roles та Roles – ролі в системі;

– Transactions – таблиця з інформацією про переведення коштів на рахунок організації.

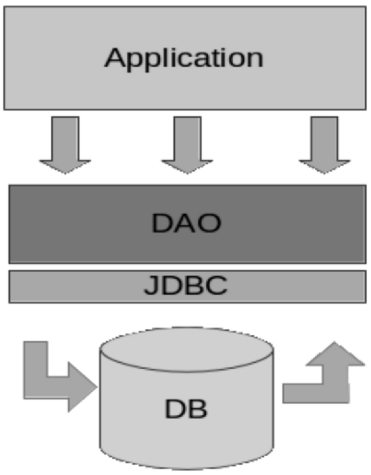

Рисунок 3.12 – Архітектура аплікації з шаблону об'єкта доступу до даних (DAO)

JdbcTemplate обслуговує базу даних, тому не потрібно виконувати багато повторюваних робіт/запитів JdbcTemplate. Клас JdbcTemplate основного пакету JDBC є центральним класом. Він виконує роботу зі створення та випуску ресурсів, при цьому усуває незначні синтаксичні помилки, наприклад пропущене відключення від бази даних. Таким чином, він обробляє всі основні речі, пов'язані з JDBC, у той час як код програми, який потрібно закріпити, є переважно прикладним SQL і очікує повернення результатів. Він виконує запити SQL, використовує оператори оновлення та виклики збережених процедур для отримання результатів, повторює набори результатів і повертає значення параметрів, які були визнані зручними. Клас виявляє будь-які неочікувані помилки, які виявляються під час роботи з операціями бази даних, і переводить їх у загальну стислу інформаційну ієрархію винятків пакета DAO, визначену у Spring.

Для використання JdbcTemplate до роботи з базою даних потрібно лише розширити інтерфейси зворотного виклику PreparedStatementCreator. Він у свою чергу генерує підготовлений оператор з'єднання, який повертає цей клас, із заданим SQL командами і будь-якими необхідними параметрами. Так само інтерфейс CallableStatementCreator створює оператори, які можна викликати. Інтерфейс RowCallbackHandler отримує значення для кожного рядка з набору результатів.

JdbcTemplate використовувався в реалізації DAO шляхом безпосереднього створення налаштованого екземпляра в контейнері Spring IoC. Але представлення даних в інтерфейсі користувача безпосередньо через рівень DAO (або рівень репозиторія) не використовується і вважається дурним тоном. Для вирішення цієї задачі розроблені відповідні сервіси. Сервіс – це клас Java, який містить основу (бізнес-логіку) для роботи з визначеною сутністю. По суті цей сервіс використовує готові DAO/репозиторії або інші сервіси, щоб надати кінцеві дані для представлення користувачу. У системі «розумне місто» використані наступні сервіси:

Адміністратор зможе редагувати дані користувача, передавати список зареєстрованих користувачів системи, деактивувати/реактивувати користувача та додавати/змінити набір ролей, якщо це необхідно.

Системне керування завданнями дозволяє редагувати опис доступних завдань у системі, який потім показується користувачам із відповідною роллю в певній організації.

«Коментарі» дозволяють коментувати певні завдання відповідно до вимог зазначених у таблицях “Бюджет” та “Транзакції” дозволяють проводити операції з бюджетом організацій та переглядати історію платежів, при необхідності відміняти деякі з них.

На рівні сервісу використовується спеціальний маппер (конвертер), який конвертує об’єкти з бази даних у формат JSON. Приклад такої сутності продемонстровано на рисунку 3.13.


```
{
   "id":1,
   "name":"Police",
   "address":"Justice street - 50",
   "responsiblePersons":[
      {
         "id":3,
         "name":"RESPONSIBLE",
         "surname":"PERSON",
         "email":"responsible_person@mail.com",
         "phoneNumber":"0666666666",
         "active":true,
         "createdDate":"2024-09-19T12:15:40.000",
         "updatedDate":"2024-09-19T12:30:17.000"
      }
   ],
   "createdDate":"2024-09-19T12:21:43.000",
   "updatedDate":"2024-09-19T12:21:43.000"
}
```


Рисунок 3.13 – Приклад запиту організації

3.3.3 Розробка контролерів вебзастосунку Розумне місто

Контролери створені та протестовані відповідно до системних вимог Smart City. Це означає, що для кожного варіанту доступу до інформації розроблені рольові обмеження. Ці обмеження описані в класі MethodSecurityConfig (рис. 3.14).

```
@Configuration("securityConfiguration")
@PropertySource(value = "classpath:methodSecurity.properties")
public class MethodSecurityConfig {

    @Value("#{'${userController.deleteUser}'.split(',')}")
    private List<String> userControllerDeleteUserAllowedRoles;

    @Value("#{'ROLE_ADMIN'.split(',')}")
    private List<String> userControllerActivateUserAllowedRoles;

    @Value("#{'ROLE_ADMIN'.split(',')}")
    private List<String> userControllerSetRolesByUserIdAllowedRoles;
```

Рисунок 3.14 – MethodSecurityConfig (Клас безпеки)

У коді використано анотацію PropertySource, тобто деякі властивості будуть взяті з конфігураційних файлів програми. Зокрема, файл methodSecurity.properties вказує, яку роль повинен мати користувач, щоб отримати доступ до шляху, пов'язаного з контролерами. Нижче наведено код контролера організації (рис. 3.15).

```
#OrganizationController
organizationController.create=ROLE_ADMIN
organizationController.update=ROLE_ADMIN
organizationController.delete=ROLE_ADMIN
organizationController.addUserToOrganization=ROLE_ADMIN
organizationController.removeUserFromOrganization=ROLE_ADMIN
```

Рисунок 3.15 – OrganizationController (список ролей та ендпоінтів контролеру організації)

Таким чином, лише адміністратор може видаляти або деактивувати інших користувачів і змінювати їхні ролі. Процес розробки контролера спрощує використання Swagger Documentation. Swagger — це правило (або, можна сказати, специфікація) для формату, який пояснює REST API досить простою, але інформативною мовою. Swagger Documentation використовує спеціальну мову, яку розуміють усі, незалежно від того, розробник чи ні. Тому, розробники програмного забезпечення, менеджери продуктів і проектів, бізнес-аналітики та, можливо, потенційні клієнти всі вони мають доступ до розробки API.

Крім того, такий регульований характер Swagger робить його придатним для тестування та налагодження API. Інший важливий момент, на який слід звернути увагу, полягає в тому, що ідентична документація може бути використана для прискорення різних процесів, пов'язаних з API. Зокрема, у цій програмі застосовано Swagger UI та Swagger Inspector.

Swagger UI – це інтерфейсний інструмент із відкритим вихідним кодом, який можна налаштувати, готовий до розгортання в будь-якому середовищі. Процес створення документації у Swagger UI відбувається автоматизовано.

Swagger Inspector – інструмент для тестування та автоматичного створення документації OpenAPI для будь-якого API. За допомогою Swagger Inspector можна перевірити та протестувати будь-який API без будь-яких обмежень щодо того, що тестується. Тести автоматично зберігаються в хмарі для легкого доступу. На рис. 3.16 перераховано усі використані контролери.

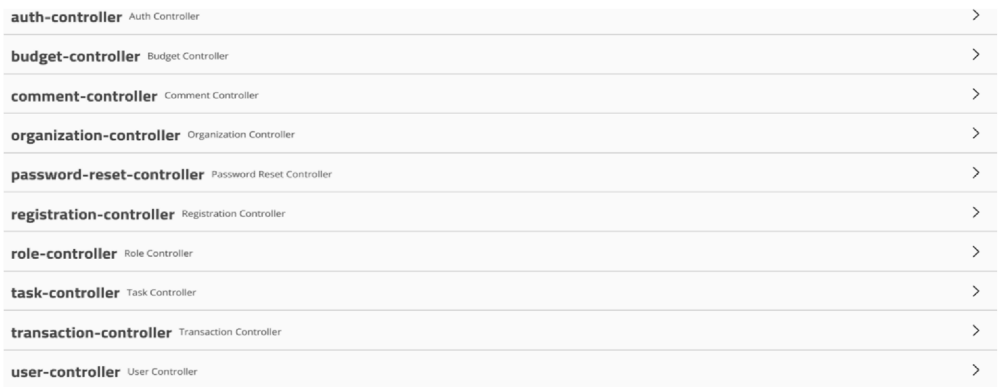

Рисунок 3.16 – Повний список контролерів аплікації

На рисунку 3.17 перераховано маршрути контролера організації. Щоб відтворити процес автентифікації за допомогою Swagger, було вирішено використовувати Bearer Token. Це простий рядок, який представляє запит API автентифікації, який надсилається в заголовку авторизації HTTP. Формат рядка не має значення для клієнтів, які його використовують, і може мати будь-яку довжину.

Рисунок 3.17 – Перелік ендпоінтів контролеру організацій

Для доступу до кожного з них потрібно авторизуватися (токен JWT). Токени-носії роблять запити API набагато легшими, оскільки вони не містять криптографічного шифрування. Компроміс тут полягає в тому, що оскільки запит містить маркер відкритого тексту, який може бути перехоплений і потім використаний будь-ким, усі запити API мають виконуватися через HTTPS. Перевага такого методу полягає в тому, що не потрібно працювати зі складними бібліотеками, щоб надсилати запити, і його досить легко реалізувати як клієнтами, так і серверами. Реалізація контролеру автентифікації подана на рис. 3.18.

Для доступу до ресурсів необхідно ввести логін і пароль. Після успішної автентифікації створюється маркер JWT, і можна спробувати використати інші контролери. Екземпляр відповіді сервера подано на рисунку 3.19.

Рисунок 3.18 – Ендпоінт процесу авторизації в Swagger

Рисунок 3.19 – Відповідь системи на запит логіну

У разі успішної відповіді на авторизацію маркер повертається назад. Тепер слід просто додати цей термін із префіксом «Носій» до заголовків запиту та спробувати отримати список усіх організацій (тільки з правами адміністратора/супервізора) (рис. 3.20).

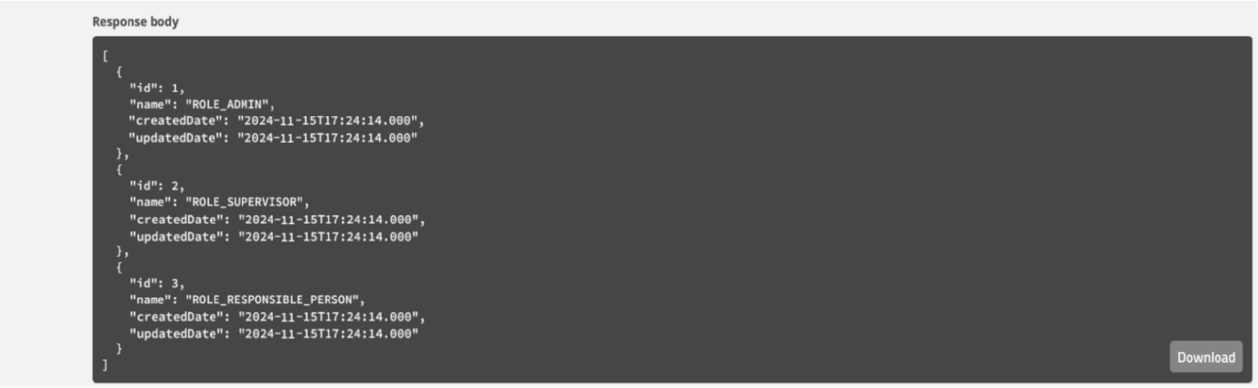

Рисунок 3.20 – Відповідь на запит всіх організацій

Таким чином Swagger допомагає легко тестувати та вдосконалювати розробку контролерів.

РОЗДІЛ 4
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Аналіз законодавчих та нормативно-правових актів з охорони праці при використанні ПК

Зрозуміло, що в основі законодавчих та нормативно-правових актів з охорони праці лежить Конституція України. Під час розробки законодавства про охорону праці широко використовуються Конвенції і Рекомендації МОП, директиви ЄС, досвід нормотворення Німеччини, Великобританії та інших країн світу. Представники країни беруть участь у різноманітних міжнародних проектах.

Закон «Про охорону праці» є найважливішим законодавчим актом. Він визначає основні положення щодо реалізації конституційного права громадян на охорону їх життя і здоров'я в процесі трудової діяльності, регулює відносини між роботодавцем і працівником з питань безпеки праці і встановлює єдиний порядок охорони праці в Україні. Дія Закону поширюється на всі підприємства та організації незалежно від форм власності та видів їх діяльності на усіх громадян, які працюють, а також залучені до праці на цих підприємствах.

Якщо міжнародним договором, згода на обов'язковість якого надана Верховною Радою, встановлено інші норми, ніж ті, що передбачені законодавством України про охорону праці, застосовуються норми міжнародного договору.

Законодавство про охорону праці (рис. 4.1) складається з Законів: «Про охорону праці», Кодексу законів про працю, «Про загальнообов'язкове державне соціальне страхування від нещасних випадків на виробництві та професійні захворювання, які спричинили до втрати працездатності», «Про пожежну безпеку», «Про використання ядерної енергії та радіаційну безпеку»,

«Про забезпечення санітарного та епідемічного благополуччя населення», «Про основні засади державного нагляду у сфері господарської діяльності», «Про дозвільну систему у сфері господарської діяльності».

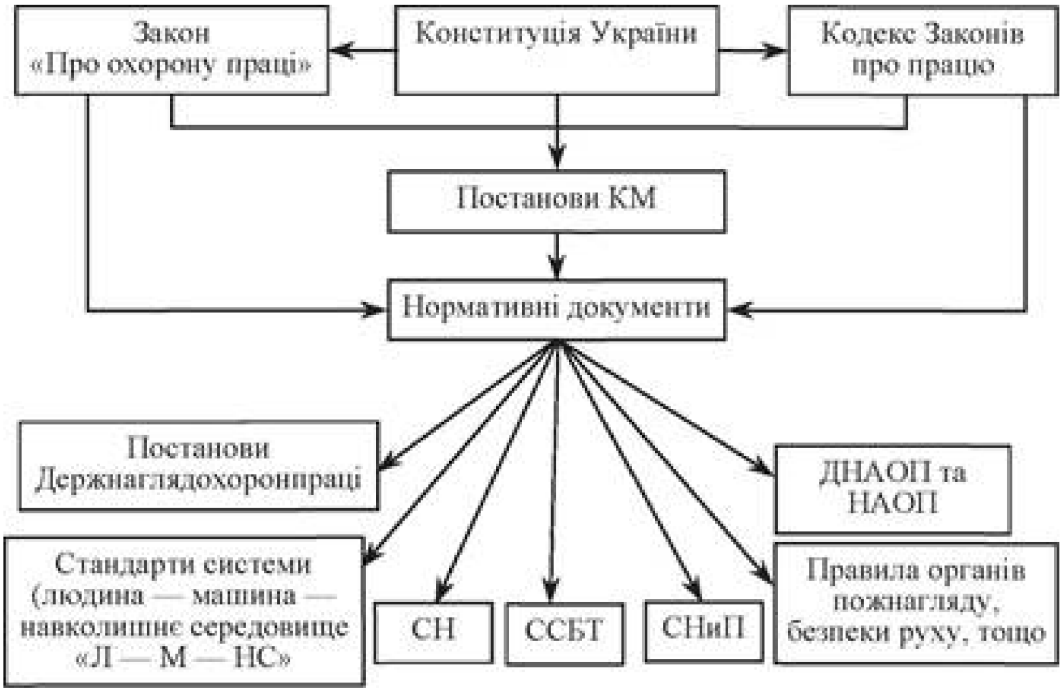

Рисунок 4.1 – Законодавчі та нормативно-правові акти з охорони праці

Трудові відносини між працівниками і роботодавцями в Україні регулюються Кодексом законів про працю (КЗпП), відповідно до якого права працюючої людини на охорону праці охороняються всебічно. КЗпП містить розділ XI «Охорона праці» (ст. 153-173) та розділ XVIII «Нагляд і контроль за дотриманням законодавства про працю».

Сьогодні на території України діє широкий спектр нормативних документів з охорони праці - від міждержавних (ГОСТ, ССБТ) до нормативних документів конкретних організацій (підприємств).

До нормативно-правових актів з охорони праці відносяться, згідно зі ст. 27 Закону, правила, норми, регламенти, положення, стандарти, інструкції та інші документи, обов'язкові до виконання.

Система стандартів безпеки праці - комплекс взаємопов'язаних стандартів, які містять вимоги, норми і правила, що направлені на забезпечення

безпеки праці, збереження життя, здоров'я і працездатності людини у процесі трудової діяльності.

До Державного реєстру НПАОП включено нормативні акти з охорони праці, затверджені відповідними органами нагляду протягом останніх років, внесено офіційні зміни і доповнення, що містяться в інформаційних покажчиках.

Питання з виробничої санітарії на сьогодні містять не тільки ГОСТ і ССБТ, але і державні санітарні норми. Так, наприклад, розглянемо вимоги до наступних виробничих факторів – шум, вібрація, мікроклімат виробничих приміщень.

Нормотворення в галузі охорони праці в Україні найтісніше пов'язане з аналогічним процесом в інших країнах СНД, активно розвивається. Всі зміни і доповнення до діючих норм і правил періодично публікуються в офіційному розділі науково-виробничого журналу «Охорона праці» Держгірпромнагляду.

Розпорядженням Кабінету Міністрів України від 31 серпня 2011 р. схвалена концепція загальнодержавної цільової програми поліпшення стану безпеки, гігієни праці та виробничого середовища на 2012-2016 рр.

В кожній галузі, зокрема в освітній, розроблені програми поліпшення стану безпеки, гігієни праці та виробничого середовища. Також заключена галузева угода між МОНМС та ЦК Профспілки працівників освіти і науки України.

На підприємствах, в установах, організаціях розробляються стандарти з безпеки праці, створюються також інструкції з охорони праці для кожної професії. Робітники і службовці повинні дотримуватись вимог інструкцій, які встановлюють правила виконання робіт і поведінки у виробничих приміщеннях і на території підприємства.

Норми охорони праці повинні органічно входити до правил внутрішнього розпорядку організацій і підприємств, які працівники мають виконувати .

Порядок опрацювання і затвердження власних нормативних актів з охорони праці, тобто тих, що діють на підприємстві, визначений НПАОП

0.006.03-93 «Порядок опрацювання та затвердження власником нормативних актів про охорону праці, що діють на підприємстві».

Положення про організацію системи управління охороною праці в галузі освіти затверджене наказом МОН України 01.08.2001. (зміни внесені 20.11.2006). Воно складається з 8 розділів.

В Україні видаються Покажчик нормативно-правових актів з питань охорони праці (НПАОП), якій постійно оновлюється і поповнюється. Остання його редакція містить біля 800 документів з охорони праці.

4.2 Дія електромагнітного випромінювання на організм людини, та його нормування

Електромагнітні поля негативно впливають на організм людини, яка безпосередньо працює з джерелом випромінювання, а також на населення, яке мешкає поблизу джерел випромінювання. Встановлено, що переважна частина населення знаходиться в умовах підвищеної активності ЕМП. Можна вважати, що в діапазоні промислових частот (у тому числі 50 Гц) допустимо розглядати вплив на біологічний об'єкт електричної і магнітної складових поля роздільно. В будь-якій точці ЕМП промислової частоти енергія магнітної складової поля, яка поглинається тілом людини, майже в 50 разів менша від енергії електричної складової цього поля, що поглинається тілом. Це дає змогу зробити висновок, що в діапазоні промислових частот дією магнітної складової поля на біологічний об'єкт можна знехтувати, а негативний вплив на організм обумовлений електричною складовою поля.

Ступінь впливу електромагнітних випромінювань на організм людини взагалі залежить від діапазону частот, тривалості опромінення, характеру опромінення, режиму опромінення, розмірів поверхні тіла, яке опромінюється, та індивідуальних особливостей організму.

У результаті дії ЕМП на людину можливі гострі та хронічні форми порушення фізіологічних функцій організму. Ці порушення виникають в результаті дії електричної складової ЕМП на нервову систему, а також на структуру кори головного та спинного мозку, серцево-судинної системи.

У більшості випадків такі зміни в діяльності нервової та серцево-судинної системи мають зворотній характер, але в результаті тривалої дії вони накопичуються, підсилюються з плином часу, але, як правило, зменшуються та зникають при виключенні впливу та поліпшенні умов праці. Тривалий та інтенсивний вплив ЕМП призводить до стійких порушень та захворювань.

На початку 60-х років у науково-технічній літературі з'явилися перші відомості про те, що люди, опромінені імпульсом НВЧ коливань, можуть постійно чути якийсь звук. Залежно від тривалості та частоти повторень імпульсів цей звук сприймається як щебет, цвірінчання чи дзюркіт у деякій точці всередині чи ззаду голови. Це явище викликало інтерес вчених, які розпочали систематичні дослідження на людях та тваринах. Під час дослідів люди повідомляли про свої відчуття.

Отже, електромагнітне випромінювання як хвороботворний чинник слід розглядати на підставі клінічних та експериментальних матеріалів. Сумісну дію цих випромінювань широкого діапазону можна класифікувати як окрему радіохвильову хворобу. Тяжкість її наслідків знаходиться у прямій залежності від напруженості ЕМП, тривалості впливу, фізичних особливостей різних діапазонів частот, умов зовнішнього середовища, а також від функціонального стану організму, його стійкості до впливу різних чинників можливостей адаптації.

Поряд із радіохвильовою хворобою (як специфічним результатом дії ЕМП) зростає ризик виникнення загальних захворювань, захворювань органів дихання, травлення тощо. Це відбувається також і за дуже малої інтенсивності ЕМП, яка незначно перевищує гігієнічні нормативи. Ймовірно, що причиною тут є порушення нервово-психічної діяльності як головної у керуванні всіма функціями організму.

У результаті дії на організм людини електромагнітних випромінювань в діапазоні 30 кГц - 300 МГц спостерігається: загальна слабкість, підвищена втома, сонливість, порушення сну, головний біль та біль в ділянці серця. З'являється роздратованість, втрачається увага, сповільнюються рухово-мовні реакції. Виникає ряд симптомів, які свідчать про порушення роботи окремих органів - шлунку, печінки, підшлункової залози. Погіршуються харчові та статеві рефлекси, діяльність серцево-судинної системи, фіксуються зміни показників білкового та вуглеводневого обміну, змінюється склад крові, зафіксовані зміни на рівні клітин.

При систематичній дії ЕМП високої та надвисокої частоти на організм людини спостерігається підвищення кров'яного тиску, трофічні явища (випадіння волосся, ламкість нігтів). ЕМП викликають зміну поляризації молекул та атомів, які є складовою частиною клітин, в результаті чого виникає небезпечний нагрів. Надмірне тепло може нанести шкоду як окремим органам, так і всьому організму людини. Професійні захворювання виникають у працівників при тривалому та інтенсивному опроміненні.

Вплив випромінювань надвисокої частоти (НВЧ) на організм людини привертає увагу великої кількості дослідників і відображається у численних наукових доповідях і публікаціях. В одній із них наведені відомості про клінічні прояви дії НВЧ залежно від інтенсивності опромінення. При інтенсивності близько 20 мкВт/см2 спостерігається зменшення частоти пульсу, зниження артеріального тиску, тобто явна реакція на опромінення. Вона сильніша й може навіть виражатися у підвищенні температури шкіри в осіб, які раніше потрапляли під дію опромінення.

Із ростом інтенсивності відбуваються електрокардіографічні зміни, при хронічному впливі - тенденція до гіпотонії, до змін у нервовій системі. Потім спостерігається прискорення пульсу, коливання об'єму крові.

При інтенсивності 6 мВт/см2 помічені зміни у складі крові, помутніння кришталика. Випромінювання інтенсивністю до 100 мВт/см2 викликають стійку гіпотонію, стійкі зміни серцево-судинної системи, двосторонню

катаракту. Подальше опромінення помітно впливає на тканини, викликає больові почуття. Якщо інтенсивність перевищує 1 Вт/см2, це спричинює дуже швидку втрату зору, що є одним із серйозних ефектів дії НВЧ на організм людини. На більш низьких частотах такі ефекти не відбуваються, і тому їх треба вважати специфічними для НВЧ діапазону. Ступінь пошкодження залежить, в основному, від інтенсивності та тривалості опромінення.

Інтенсивне НВЧ опромінення відразу викликає сльозотечу, подразнення, звуження зіниці ока. Після короткого (1-2 доби) прихованого періоду спостерігається погіршення зору, що посилюється під час повторного опромінення і свідчить про кумулятивний характер пошкоджень. Спостереження за людьми доводять існування механізму відбудови пошкоджених клітин, який вимагає тривалого часу (10-20 діб). Зі зростанням часу та інтенсивності впливу пошкодження набувають незворотного характеру.

У разі прямого впливу на око випромінювання відбувається пошкодження рогівки. Але серед усіх тканин ока найбільшу чутливість в діапазоні 1-10 ГГц має кришталик. Сильні пошкодження кришталика зумовлені тепловим впливом НВЧ (при щільності потоку енергії понад 100 мВт/см2). За малої інтенсивності помутніння спостерігаються тільки у задній ділянці, за великої - по всьому об'єму кришталика.

Для зменшення впливу ЕМП на персонал та населення, яке знаходиться у зоні дії радіоелектронних засобів, потрібно вжити ряд захисних заходів. До їх числа можуть входити організаційні, інженерно-технічні та лікарськопрофілактичні.

Організаційні заходи здійснюють органи санітарного нагляду. Вони проводять санітарний нагляд за об'єктами, в яких використовуються джерела електромагнітних випромінювань.

Інженерно-технічні заходи передбачають таке розташування джерел ЕМП, яке 6 зводило до мінімуму їх вплив на працюючих, використання в умовах виробництва дистанційного керування апаратурою, що є джерелом випромінювання, екранування джерел випромінювання, застосування засобів

індивідуального захисту (халатів, комбінезонів із металізованої тканини, з виводом на заземлюючий пристрій). Для захисту очей доцільно використовувати захисні окуляри ЗП5-90. Скло окулярів вкрито напівпровідниковим оловом, що послаблює інтенсивність електромагнітної енергії при світлопропусканні не нижче 75%.

Взагалі, засоби індивідуального захисту необхідно використовувати лише тоді, коли інші захисні засоби неможливі чи недостатньо ефективні: при проходженні через зони опромінення підвищеної інтенсивності, при ремонтних і налагоджувальних роботах в аварійних ситуаціях, під час короткочасного контролю та при зміні інтенсивності опромінення. Такі засоби незручні в експлуатації, обмежують можливість виконання трудових операцій, погіршують гігієнічні умови.

У радіочастотному діапазоні засоби індивідуального захисту працюють за принципом екранування людини з використанням відбиття і поглинання ЕМП. Для захисту тіла використовується одяг з металізованих тканин і рідіопоглинаючих матеріалів. Металізовану тканину роблять із бавовняних ниток з розміщеним всередині них тонким проводом, або з бавовняних чи капронових ниток, спірально обвитих металевим дротом. Така тканина, наче металева сітка, при відстані між нитками до 0,5 мм значно послаблює дію випромінювання. При зшиванні деталей захисного одягу треба забезпечити контакт ізольованих проводів. Тому електрогерметизацію швів здійснюють електропровідними масами чи клеями, які забезпечують гальванічний контакт або збільшують ємнісний зв'язок неконтактуючих проводів.

Лікарсько-профілактичні заходи передбачають проведення систематичних медичних оглядів працівників, які перебувають у зоні дії ЕМП, обмеження в часі перебування людей в зоні підвищеної інтенсивності електромагнітних випромінювань, видачу працюючим безкоштовного лікарсько-профілактичного харчування, перерви санітарно-оздоровчого характеру.

РОЗДІЛ 5
ЕКОНОМІЧНА ЕФЕКТИВНІСТЬ

5.1 Економічна ефективність використання вебзастосунку «Розумне місто»

Вебзастосунок інформаційної системи «Розумне місто» є інструментом, що сприяє автоматизації процесів управління міськими ресурсами, підвищенню якості життя мешканців та раціоналізації витрат. Розглянемо ключові аспекти економічної ефективності такого рішення:

Завдяки централізованому збору й аналізу даних, система дозволяє ефективніше розподіляти ресурси. Знижує витрати на енергію шляхом оптимізації роботи освітлення, систем опалення та водопостачання. Зменшує витрати на утримання інфраструктури через використання прогнозного аналізу для технічного обслуговування.

Інтеграція цифрових рішень у міське управління. Скорочує час на ухвалення рішень завдяки доступу до аналітичних даних у реальному часі. Зменшує адміністративні витрати через автоматизацію бюрократичних процесів (наприклад, електронні заявки на послуги, оплата штрафів і податків онлайн).

Залучення інвестицій. Сучасні технології в управлінні містом приваблюють інвесторів. Поліпшення транспортної інфраструктури, комунальних послуг та екологічних показників створює сприятливі умови для бізнесу. Інноваційні проєкти часто підтримуються міжнародними грантами та програмами фінансування.

Розумні рішення, такі як смарт-лічильники та системи моніторингу споживання. Скорочує несанкціоновані витрати води, газу чи електроенергії.

Оптимізовує витрати на транспортування сміття за рахунок впровадження розумних контейнерів із сенсорами заповнення.

Прозора система управління фінансами та цифровізація сплати податків і штрафів. Збільшує надходження від податкових платежів. Скорочує корупційні ризики.

Інформованість та залученість громадян. Підвищення довіри мешканців до муніципальної влади. Скорочення витрат мешканців на транспорт, комунальні послуги чи адміністративні послуги завдяки зручному доступу до інформації.

Система "Розумне місто" може оптимізувати міську мобільність. Зменшення заторів завдяки інтелектуальному управлінню світлофорами. Зниження витрат на обслуговування транспорту за рахунок аналізу даних про використання маршрутів.

Вебзастосунок інформаційної системи "Розумне місто" має значний потенціал для економії міських ресурсів та підвищення доходів, водночас покращуючи якість життя громадян. Інвестиції в такі системи є вигідними в довгостроковій перспективі, оскільки сприяють сталому розвитку міської інфраструктури.

5.2 Бізнес-план розробки програмного забезпечення

Коротке найменування: Розробка вебзастосунку «Розумне місто».

Видом послуг є надання послуг в інформатизації та створенні відповідного сайту системи управління службами конкретного міста.

КВЕД 62.01 Комп'ютерне програмування.

КВЕД 62.02 Консультування з питань інформатизації.

Цільова аудиторія продукту – керівники міст, зацікавлені в розвитку безпеки та привабливості свого міста.

Для реалізації проекту слід залучити інвестиції в розмірі 1 500 000.00 грн (Один мільйон п’ятсот тисяч гривень, 00 копійок). Джерелами фінансування проекту будуть адміністрації міст та можливі інвестори. За перший рік не плануються продажі. Вони стануть можливими лише на 2 рік після повної реалізації продукту першому клієнту.

Для створення продукту потрібні фронтенд команда в кількості 2 працівники та бекенд команда в кількості 4 працівники та один менеджер проекту. Також бізнес аналітик та команда промоутерів – 5 працівників.

Спочатку місто має замовити продукт, та надалі оплачувати підтримку системи. Тому оплата планується частинами, різниця в оплаті буде лише в кількості служб міста. Середня ціна реалізації системи під конкретне місто буде становити близько мільйону гривень, підтримка системи оцінюється в 100000 грн за місяць. Середня ціна системи під одне місто – 1 мільйон гривень (час реалізації близько року). Загальний прибуток (за місяць) без податків 100 тисяч грн. Рентабельність: 1.9.

Для того щоб задовольнити клієнта потрібна зручна система. Достатньо часу має буде присвячено розробці дизайну зручного інтерфейсу. Також, швидкодія та надійність є важливими пунктами. Команда бекенд має докласти зусилля щоб код працював швидко та відлагоджено, що немало важливо є також читабельність такого коду. Тому потрібні фахівці високого класу.

Головним недоліком послуги є велика ціна та незацікавленість міст в отриманні доступу до даної системи.

Аналогів послуги немає в країні. Альтернативних рішень аплікації немає, тому встановлювати ціну у порівнянні з ними - неможливо.

Вартість послуги вираховується на основі кількості організацій та співробітників відповідно. Вартість однієї служби в середньому буде становити 250 000 грн. Тому мінімумом для міста буде приблизно 4 служби.

Просуванням системи буде займатись команда із промоутерів. Будуть створені реклами на сайтах, проінформовані голови міст. Також планується створення реклами для кожної зі служб міста.

Багато спеціалістів працюють дистанційно. Тому офіс не відіграє ролі. Режим роботи – повний робочий день. Перші фінансові вкладення потрібні будуть на оплату фахівців та пошук кадрів.

Юридичний статус – Фізична особа-підприємець 3 групи.

Переваги ФОП:

– низький розмір штрафів;

– низьке держмито;

– свобода в розподілі доходів;

– спрощена ліквідація;

– звільнення від сплати майнового податку на використовуване майно;

– можливість працювати без відкриття банківського рахунку;

– можливість вести діяльність без атрибутів юрособи – печатки, штампа, статуту;

– робота по схемі єдиного податку.

Джерелом фінансування будуть інвестиції(1500000 грн) або кредитні кошти. Витрати на старт проекту близько 1000000 грн.

Перший прибуток буде лише приблизно через рік, після створення першого екземпляру системи для міста.

Існують деякі ризики провалу проекту в плані незацікавленості в послузі. В даному випадку буде виділено більше коштів на рекламу системи. В плані покриття збитків можливе підвищення цін на послугу.

ВИСНОВКИ

Проектування вебзастосунку інформаційної системи "Розумне місто" є актуальним і складним завданням, що вимагає інтеграції різноманітних технологій і врахування широкого спектру вимог. Після проведення аналізу та розробки концепції вебзастосунку для такої системи можна зробити такі висновки:

Значення інтеграції різних систем: Для забезпечення ефективної роботи інформаційної системи "Розумне місто" необхідна інтеграція різних міських служб і платформ, таких як управління енергоресурсами, транспортом, безпекою та комунікаціями. Це дозволяє створити єдину екосистему, яка забезпечує оперативний обмін даними між усіма елементами міської інфраструктури.

Сучасні технології для надійної архітектури: Впровадження Інтернету речей (IoT), великих даних (Big Data), хмарних обчислень і штучного інтелекту (AI) дозволяє забезпечити високу продуктивність і масштабованість вебзастосунку. Ці технології дозволяють обробляти великі обсяги даних у реальному часі, що є критичним для підтримки актуальності і точності інформації.

Зручність та доступність для користувачів: Успішний вебзастосунок для розумного міста повинен бути зручним і інтуїтивно зрозумілим для широкого кола користувачів. UX/UI-дизайн має відповідати принципам доступності, щоб забезпечити зручне використання незалежно від рівня цифрової грамотності користувачів.

Забезпечення безпеки та конфіденційності даних: Розумні міста збирають значну кількість персональних даних, тому вебзастосунок повинен мати надійні механізми для захисту інформації та конфіденційності. Це вимагає впровадження передових засобів кібербезпеки, включаючи шифрування, аутентифікацію користувачів та багаторівневий захист даних.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Каралью А., Дель Бо К., Нейкамп П. (2011). Розумні міста в Європі. Журнал міських технологій, 18(2), 65-82.

2. Батті М., Аксгаузен К. В., Джіаннотті Ф., Позднухов А., Баццані А., Вахович М., Португалі Ю. (2012). Розумні міста майбутнього. Європейський фізичний журнал, 214(1), 481-518.

3. Харрісон К., Доннеллі І. А. (2011). Теорія розумних міст. Матеріали 55-ї щорічної зустрічі ISSS-2011.

4. Нам Т., Пардо Т. А. (2011). Концептуалізація розумного міста з вимірами технологій, людей і установ. Матеріали 12-ї щорічної конференції з цифрового уряду, 282-291.

5. Анджелідоу М. (2015). Розумні міста: поєднання чотирьох сил. Міста, 47, 95-106.

6. Гашем І. А. Т., Чанг В., Аннуар Н. Б., Адеволе К., Якоб І., Гані А., Чирома Х. (2016). Роль великих даних у розумних містах. Міжнародний журнал управління інформацією, 36(5), 748-758.

7. Дзанелла А., Буї Н., Кастеллані А., Ванжеліст Л., & Цорзі М. (2014). Інтернет речей для розумних міст. Журнал Інтернету речей IEEE, 1(1), 22-32.

8. Аль-Нуаймі Е., Аль-Нейяді Х., Надер М., Аль-Джаруді Дж. (2015). Застосування великих даних у розумних містах. Журнал інтернет-сервісів та застосунків, 6(1), 1-15.

9. Шафферс Х., Комнінос Н., Паллот М., Трусс Б., Нільссон М., Олівейра А. (2011). Розумні міста та інтернет майбутнього: до кооперaційних рамок відкритих інновацій. Асамблея інтернету майбутнього, 431-446.

10. Кітчін Р. (2014). Місто в реальному часі? Великі дані та розумний урбанізм. ГеоЖурнал, 79(1), 1-14.

11. ГОСТ 19.003-80 ЄСПД. Схеми алгоритмів і програм. Позначення умовні графічні.

12. ГОСТ 19.105-78 ЄСПД. Загальні вимоги до програмних документів.

13. ГОСТ 19.401-78 ЄСПД. Текст програми. Вимоги до змісту і оформлення.

14. ДСТУ 3008-95 Документація. Звіти у сфері науки і техніки. Структура і правила оформлення.

15. НД ТЗІ 1.1-003-99. Термінологія у області захисту інформації в комп'ютерних системах від несанкціонованого доступу. // Департамент спеціальних телекомунікаційних систем і захисту інформації Служби безпеки України. – Київ, 1999.

16. Leitão J.; Gil P.; Ribeiro B.; Cardoso A. A survey on home energy management. IEEE Access 2020, 5722 p.

17. Yahia Z.; Pradhan A. Optimal load scheduling of household appliances considering consumer preferences: An experimental analysis. Energy 2018, 163 p.

18. Chen Y.Y.; Lin Y.H. A smart autonomous time- and frequency-domain analysis current sensor-based power meter prototype developed over fog-cloud analytics for demand-side management. Sensors 2019, 4443 p.

19. Evans A.M.; Campos A. Open government initiatives: Challenges of citizen participation. J. Policy Anal. Manag. 2013, 32, 172–185.

20. Yeh H. The effects of successful ICT-based smart city services: From citizens'perspectives. Gov. Inf. Q. 2017, 34, 556–565.

21. Coe A.; Paquet G.; Roy J. E-governance and smart communities: A social learning challenge. Soc. Sci. Comput. Rev. 2001, 19, 80–93.