КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра технологій управління

Спеціальність 122 - Комп'ютерні науки
Освітньо-професійна програма «Управління проєктами»

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА
на тему:

«Дослідження технології управління проєктом створення модуля нормалізації адрес в ERP-системі»

Студента 2-го курсу групи УПз-21

Олександра МОРОЗА
(ім'я, прізвище)

(підпис студента)

Науковий керівник:

к.т.н., доцент
(науковий ступінь, вчене звання)

Тетяна ЛАТИШЕВА
(ім'я, прізвище)

(дата) *(підпис)*

Попередній захист:

(Висновок: «До захисту в Екзаменаційній комісії»)

Завідувач кафедри
технологій управління _____ Віктор МОРОЗОВ _____
(підпис) *(ім'я, прізвище)* *(дата)*

Київ - 2025

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій

Кафедра технологій управління
Освітній рівень Магістр
Спеціальність 122 Комп'ютерні науки
Освітньо-професійна програма Управління проєктами

ЗАТВЕРДЖУЮ
Завідувач кафедри
професор Віктор МОРОЗОВ

_____
«29» вересня 2025 року

З А В Д А Н Н Я
НА ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Студента: Олександра МОРОЗА
Група: УПз-21

1. Тема кваліфікаційної роботи: «Дослідження технології управління проєктом створення модуля нормалізації адрес в ERP-системі»

Затверджена Протоколом від 16.06.2025 № 15.

2. Строк подання студентом готової роботи - «10» 12.2025.

3. Цільова установка та вихідні дані до роботи: дослідження проблем якості адресних даних в інформаційних системах державних органів та причин виникнення неструктурованих записів і дублікатів у довіднику адрес ERP-системи НАЗК. Аналіз сучасних методів нормалізації адрес, алгоритмів виявлення дублікатів та інструментів інтелектуального аналізу даних (LLM-моделі, зокрема ChatGPT), а також державних адресних класифікаторів і сервісів (API Укрпошти).

4. Зміст роботи: Обґрунтування актуальності проєкту підвищення якості адресних даних для НАЗК, визначення проблеми наявності застарілих, ненормованих і дубльованих адрес у внутрішній ERP-системі. Формулювання мети, завдань, об'єкта й предмета дослідження. Аналіз існуючих підходів до нормалізації адрес (правила, довідники, ML-підходи, LLM-моделі) та огляд

доступних сервісів (API Укрпошти, інші адресні платформи), оцінка їхніх переваг і обмежень у контексті потреб НАЗК. Побудова дерева проблем та дерева цілей, що відображають поточний стан і бажаний рівень якості адресних даних. Розробка концептуальної моделі інформаційної системи та місця модуля нормалізації адрес у загальній архітектурі ERP. Створення концептуальної та фізичної моделей бази даних для зберігання нормалізованих адрес із використанням окремих довідників (області, райони, населені пункти, вулиці, будинки) та складених унікальних індексів для контролю дублікатів. Обґрунтування вибору технологічного стеку (C#, .NET, WPF/WinForms, SQL, Python, інтеграція з LLM та API Укрпошти). Вибір методів і моделей для нормалізації: опис формалізації задачі, визначення відображення «текстова адреса → структурований запис», опис гібридного підходу, що поєднує правила, звернення до офіційних довідників і використання LLM-моделі для семантичного розбору адрес. Розробка алгоритмів виявлення та обробки дублікатів на основі нормалізованих полів. Опис організації проєкту за гнучкою методологією (Agile / Scrum): формування беклогу, планування спринтів, управління ризиками та ресурсами. Практична частина: розробка модуля нормалізації та інтерфейсу адміністратора для роботи з «legacy»-записами й дублікатами, інтеграція з API Укрпошти та LLM-сервісом, реалізація бізнес-логіки в ERP-системі. Побудова сценаріїв тестування, проведення експериментів з історичними адресними даними НАЗК, аналіз отриманих результатів, оцінка впливу модуля на якість даних, трудомісткість операцій та організаційні ризики. Формулювання висновків щодо ефективності запропонованого рішення й перспектив його масштабування.

5. Перелік графічного матеріалу: Цілі проєкту та основні продукти модуля нормалізації адрес; дерево проблем та дерево цілей щодо якості адресних даних у ERP-системі НАЗК; концептуальна модель інформаційної системи та місце модуля нормалізації в архітектурі ERP; концептуальна модель бази даних адресного модуля; фізична модель бази даних (схема таблиць, ключів, індексів, зв’язків із довідниками); діаграма архітектури програмного забезпечення модуля

нормалізації (інтерфейс користувача, сервіси нормалізації, інтеграція з LLM та API Укрпошти, підсистема виявлення дублікатів); макети інтерфейсу користувача модуля (вкладки «Ненормовані адреси», «Дублі», форми редагування); ілюстративні приклади результатів нормалізації адрес і об’єднання дублікатів.

6. Календарний план виконання роботи

№ з/п	Назва частин роботи	Виконання роботи
1	Вивчення літературних джерел з предмету дослідження	02.10.25-11.10.25
2	Збір і вивчення матеріалів	12.10.25-19.10.25
3	Складання розгорнутого плану кваліфікаційної роботи	20.10.25-23.10.25
4	Ознайомлення наукового керівника з розгорнутим планом кваліфікаційної роботи. Внесення змін	24.10.25-25.10.25
5	Підготовка розділу 1	26.10.25-02.11.25
6	Підготовка розділу 2	03.11.25-14.03.25
7	Підготовка розділу 3	15.11.25-23.11.25
8	Підготовка розділу 4	24.11.25-05.12.25
9	Оформлення кваліфікаційної роботи	25.11.25-02.12.25
10	Передача кваліфікаційної роботи науковому керівникові	08.12.25
11	Попередній захист кваліфікаційної роботи	10.12.25-14.12.25
12	Передача кваліфікаційної роботи рецензенту для рецензування	14.12.25

Дата видачі завдання «30» вересня 2025 р.
Керівник роботи доцент Тетяна ЛАТИШЕВА
(посада, ім’я, прізвище)

_____
(підпис)

Завдання прийняв до виконання студент групи УПз-21
Олександр МОРОЗ
(ім’я, прізвище)

_____
(підпис)

ЗМІСТ

АНОТАЦІЯ

кваліфікаційної роботи магістра на тему
«Дослідження технології управління проєктом створення модуля нормалізації адрес в ERP-системі»

Студента: Мороза Олександра Васильовича

Науковий керівник: Латишева Тетяна Володимирівна

Рік захисту - 2025

Мета кваліфікаційної роботи полягає у дослідженні, проєктуванні та впровадженні модуля нормалізації адрес та виявлення дублікатів у внутрішній ERP-системі Національного агентства з питань запобігання корупції (НАЗК) із використанням сучасних методів інтелектуального аналізу даних (LLM-моделі) та офіційних державних довідників (API Укрпошти).

Ціль проєкту - провести розширений аналіз та дослідження процесів для створення інтегрованого підсистемного модуля, який забезпечує автоматизоване перетворення «сирих» текстових адрес у структурований нормалізований формат, присвоєння їм ідентифікаторів із довідників Укрпошти, а також виявлення й обробку дубльованих записів у довіднику адрес ERP-системи. Це дозволяє підвищити якість адресних даних, зменшити кількість помилок при обробці документів та забезпечити надійну аналітику на рівні НАЗК.

Наукова новизна - побудова концептуально моделі рішення для автоматизованої нормалізації адрес, що інтегрує довідник Укрпошти з методами обробки неструктурованих даних.

Практична цінність роботи полягає у розробці реального прототипу модуля, орієнтованого на впровадження в інформаційне середовище НАЗК. Модуль забезпечує автоматизовану очистку історичних («legacy») адресних записів із застосуванням LLM-моделі ChatGPT 5, синхронізацію з довідниками Укрпошти, контроль унікальності через складені індекси та інтерфейс для

адміністрування дублікатів. Використання запропонованого рішення знижує трудомісткість ручної обробки, скорочує кількість неструктурованих і помилкових адрес у базі даних та створює основу для подальшої автоматизації процесів документообігу й аналітики в агенції.

Кваліфікаційна робота складається з анотації, вступу, основної частини, що включає чотири розділи, висновків та переліку використаних джерел.

У першому розділі розглянуто теоретичні та нормативні засади роботи з адресними даними: проаналізовано вимоги до структури адрес, підходи до нормалізації й унікалізації, а також оглянуто наявні системи та сервіси (зокрема API Укрпошти). Виконано аналіз поточного стану адресного довідника ERP-системи НАЗК, виявлено основні проблеми (наявність застарілих, неструктурованих і дубльованих записів), побудовано дерево проблем і дерево цілей, визначено зацікавлені сторони та сформульовано вимоги до майбутнього модуля.

Другий розділ присвячено методологічним та проєктним засадам створення модуля. Обґрунтовано вибір ітеративного підходу до управління проєктом із використанням елементів Agile/Scrum. Розроблено нові концептуальні моделі інформаційної системи та адресного модуля, формалізовано задачу нормалізації адрес як перетворення текстового рядка у структурований запис із посиланнями на довідники. Окрему увагу приділено вибору технологічного стеку (C#, .NET, MSSQL, Python, LLM) та обґрунтуванню використання LLM-моделі ChatGPT 5 як інструмента для семантичного розбору неструктурованих адрес.

У третьому розділі розроблено архітектуру програмного забезпечення: описано багатошарову структуру модуля нормалізації, концептуальну та фізичну моделі бази даних, механізми інтеграції з ERP-системою та API Укрпошти. Наведено логіку роботи основних компонентів - інтерфейсу адміністратора (вкладки «Ненормовані адреси» та «Дублі»), сервісу нормалізації, підсистеми

виявлення дублікатів, журналювання змін. Сформульовано алгоритм виявлення дублікатів на основі нормалізованих полів та складеного унікального індексу, а також описано концептуальну модель процесу нормалізації legacy-записів із застосуванням LLM-моделі й подальшою валідацією через довідники Укрпошти.

Четвертий розділ містить результати практичної реалізації та дослідження ефективності запропонованого рішення на реальних даних НАЗК. Детально описано процес фільтрації ненормованих записів за допомогою SQL-запитів, підготовки навчальних прикладів, масової обробки адрес за допомогою ChatGPT 5, ручної експертної валідації та присвоєння ідентифікаторів з довідників. Показано, що кількість записів із неповним набором структурних компонентів було скорочено з 19826 до 7094, з яких більшість записів виявилися нерелевантними й підлягають видаленню. Проведено оцінку організаційного та економічного ефекту: зменшення обсягу ручної роботи, зниження ризиків помилок у документах, підвищення якості та цілісності адресних даних.

Робота завершується узагальнюючими *висновками* щодо досягнення поставленої мети й виконання всіх завдань дослідження, а також рекомендаціями стосовно подальшого розвитку модуля: масштабування на інші реєстри, розширення функцій аналітики та використання спеціалізованих моделей, навчених на українських адресних даних.

Кваліфікаційна робота складається з 101 сторінки основного тексту, містить 19 рисунків, 3 формули, 3 додатки.

Ключові слова: *нормалізація адрес, якість даних, ERP-система, НАЗК, адресний довідник, великі мовні моделі, ChatGPT, API Укрпошти, виявлення дублікатів, очищення даних, управління проєктами.*

ВСТУП

У сучасній інформаційній системі державного органу надзвичайно важливо мати високоякісний довідник адрес, що забезпечує достовірність документів, правильність їх доставки та ефективну інтеграцію з іншими системами.

У ERP-системі електронних справ Національного агентства з питань запобігання корупції (далі НАЗК) первісна структура довідника адрес передбачала лише одне текстове поле, що призвело до накопичення нерегламентованих записів і зростання кількості помилок у зв’язку з частими перейменуваннями та змінами адміністративно-територіального устрою України.

Актуальність проєкту: Існуючі механізми перевірки нових записів через API Укрпошти та розширена схема таблиці адрес (код і назва області, району, міста, вулиці, тощо) забезпечили збереження актуальності нових даних, але не вирішили задачу «очищення» історично накопичених вільних текстових записів. Окрім того, недосконаліть інтерфейсу доповнення довідника спричинює появу дублів при реєстрації документів користувачами, що створює додаткове навантаження на адміністративний персонал та знижує загальну якість даних.

Мета кваліфікаційної роботи: Розробити проєкт модуля нормалізації адрес ERP-системи НАЗК, який включатиме:

1. Інтерфейс для відокремлення та візуалізації нерегламентованих записів;
2. Алгоритм парсингу текстового поля на атрибути довідника за стандартами API Укрпошти;
3. Механізм виявлення й усунення дублів через створення унікального індексу та зручний інтерфейс редагування.

Завдання кваліфікаційної роботи:

1. Здійснити аналіз існуючих текстових записів і виокремити типові сценарії помилкових форматів.
2. Спроєктувати архітектуру модуля (компоненти: UI, сервіс парсингу, модуль індексування).
3. Дослідити специфікацію API-інтеграції з сервісом Укрпошти для отримання кодів і назв елементів адреси.
4. Дослідити можливий алгоритм розбору (парсингу) нерегламентованих адрес із використанням комбінованих методів регулярних виразів і довідникових запитів.
5. Спроєктувати та описати структуру бази даних із унікальними індексами для запобігання дублюванню.
6. Створити інтерфейс адміністратора для перегляду, редагування і видалення дублювань із вибором «правильного» варіанту.
7. Створити механізм нормалізації нерегламентованих записів адрес
8. Провести тестування модуля на реальних даних і оцінити його ефективність за метриками точності нормалізації та кількості усунених дублів.

Об’єкт і предмет дослідження: Об’єкт - процеси розробки системи нормалізації та унікалізації адрес у ERP-системі НАЗК. Предмет - конкретні аспекти та характеристики процесів управління проєктом розробки цієї системи (вибір методології, планування, організація робіт, управління ризиками)

Методи дослідження:

- Аналіз вимог та моделювання системи (UML-діаграми);
- Вивчення специфікацій API Укрпошти;
- Дослідження алгоритмів парсингу з використанням регулярних виразів та довідникових запитів;

- Проєктування індексів і структур бази даних;
- Прототипування інтерфейсу й юзабіліті-тестування;
- Експериментальна оцінка на реальних даних і статистичний аналіз результатів.

Очікувані наукові та практичні результати. Передусім буде сформовано концепцію та методику нормалізації адресних записів у державних ERP-системах на основі поєднання офіційних довідникових даних та алгоритмічного розбору ненормованих адрес. Це становить наукову новизну дослідження, оскільки запропонований підхід раніше не застосовувався в межах ERP-систем НАЗК. Практичним результатом роботи стане програмний модуль, інтегрований в ERP-систему НАЗК, який забезпечить очищення існуючих адресних даних від помилок і дублювань та уніфікацію форматів запису адрес. Очікується, що впровадження цього модуля підвищить якість даних у системі, зменшить обсяг ручної обробки адресної інформації та гарантуватиме надійність обміну даними між компонентами інформаційної системи.

Інноваційність теми та обґрунтування її вибору. Інноваційність даного проєкту полягає в розробці спеціалізованого рішення для автоматизованої нормалізації адрес, що інтегрує довідник Укрпошти з методами обробки неструктурованих даних. На відміну від типових підходів, де або використовуються лише сторонні API-сервіси, або проводиться ручне коригування адрес, запропонований модуль поєднує обидва підходи та адаптований до особливостей даних НАЗК. Вибір теми зумовлений потребою підвищити якість даних у системі НАЗК та відсутністю готових інструментів для вирішення проблеми ненормованих адресних записів. Крім того, тема відповідає сучасним тенденціям цифровізації державного управління і дозволяє застосувати методології управління проєктами та розробки програмного забезпечення для розв’язання актуальної практичної проблеми.

РОЗДІЛ 1. ОГЛЯД МЕТОДІВ НОРМАЛІЗАЦІЇ АДРЕС ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Актуальність проблеми та аналіз існуючих підходів до нормалізації адрес

Адресні дані відіграють ключову роль у багатьох інформаційних системах, особливо в державних реєстрах та ERP-системах [1] органів влади. Якість цих даних безпосередньо впливає на достовірність звітності, ефективність пошуку інформації та взаємодію між системами. На жаль, адреси можуть вводитися у різних форматах і стилях, що зумовлює проблему їх неуніфікованості та дублювання. Наприклад, назву тієї самої вулиці можна подати як «вул. Гмирі», «вул. ім. Бориса Гмирі», «вул. Бориса Гмирі» тощо - усі ці варіанти позначають одну й ту саму вулицю, але відрізняються написанням. Таким чином, виникає питання: чи існує процедура нормалізації, яка здатна ототожнити реально однакові адреси, записані по-різному? Відповідь - ствердна, і саме дослідженню таких методів присвячено дану роботу.

Нормалізація поштових адрес [2] (стандартизація) - це процес перетворення довільно заданої адреси до стандартизованого, єдиного формату, визначеного офіційним поштовим відомством країни. За визначенням фахівців, нормалізація включає виправлення помилок у написанні, додавання відсутніх компонентів (наприклад, поштового індексу), скорочення або уніфікацію позначень (наприклад, «вул.» замість «вулиця») і перетворення адреси до єдиної структури. Основна мета такого процесу - підвищити точність і актуальність адресних даних, спростити об’єднання великих наборів адрес (виявлення дублетів) та забезпечити коректне використання адрес у геоінформаційних системах та інших застосунках [2]. Наприклад, Поштова служба США [3] визначає стандартизовану адресу як таку, що містить усі необхідні елементи й використовує встановлені скорочення (Street → St, Avenue → Ave тощо) [3]. Багато країн мають власні стандарти оформлення адрес і офіційні класифікатори.

У Франції, наприклад, цим займається пошта La Poste [2]. В Україні на даний час роль джерела достовірних адресних даних виконує національний поштовий оператор - Укрпошта [4], а також триває створення централізованого державного адресного реєстру - Єдиний державний реєстр адрес [5].

Ненормалізовані адреси в базах даних ускладнюють пошук, порівняння та аналітику. Різні варіанти запису однієї адреси (наприклад, «м. Київ, просп. Перемоги, 10» і «Київ, пр-т Перемоги 10») сприймаються як різні об’єкти, що призводить до дублювання. Часто трапляються помилки, опечатки або некоректні формати (наприклад, «Івано Франківськ» без дефіса чи латиницею), що унеможливлює зіставлення з офіційними довідниками. Додатково дані можуть містити надлишкову або змішану інформацію (наприклад, «біля парку» або «5-12» в одному полі), що ускладнює їх обробку та нормалізацію.

Різні країни та розробники пропонують свої рішення для проблеми нормалізації адрес. У глобальному масштабі одним із найвідоміших інструментів є Libpostal [6] - відкрита бібліотека, призначена для парсингу та нормалізації поштових адрес різних країн та мов [6]. Ця бібліотека використовує статистичні моделі й була створена на основі великої вибірки адресних даних (первинно - проєктом Mapzen у 2016 році). Libpostal [6] дозволяє розбити довільний адресний рядок на компоненти і привести їх до стандартизованої форми, враховуючи мовні особливості. Вона є прикладом підходу на основі NLP (обробки природної мови): модель навчена розпізнавати назви міст, вулиць, номер будинку тощо у тексті і приводити їх до уніфікованого вигляду. Загалом методи нормалізації можна розділити на дві великі групи:

- **Rule-based та словникові алгоритми** - коли використовуються заздалегідь складені словники варіантів назв і правил перетворення. Наприклад, перелік стандартних скорочень («вул.» → «вулиця» або навпаки), база офіційних назв населених пунктів тощо. Такий підхід реалізовано, зокрема, у державних класифікаторах (як-от український

адресний класифікатор Укрпошти [4]). Він надійний для структурованих даних, але вимагає актуальних довідників.

- **Статистичні та нейромережеві алгоритми (NER, ML)** - коли система самонавчається на корпусі адрес і розпізнає структуру на основі ймовірнісних моделей. Застосування Named Entity Recognition [7] (NER) дає змогу виділяти компоненти адрес у тексті та визначати їх тип (місто, індекс, вулиця тощо). Сучасні дослідження демонструють ефективність нейронних мереж, зокрема методом посимвольного або послівного розпізнавання послідовностей з використанням seq2seq-моделей і моделей типу Transformer для мультинаціонального парсингу адрес [7]. Перевага статистичних методів - гнучкість і здатність обробляти нетипові або раніше небачені формати адрес; недолік - потреба у великому обсязі навчальних даних та складність забезпечення 100% точності без донавчання на локальних даних.

Рівень вивченості теми. Слід відзначити, що тема нормалізації адрес є добре опрацьованою в сфері управління якістю даних і геоінформаційних систем. Існують стандарти адресного оформлення (національні поштові норми, класифікатори) та численні програмні реалізації. В Україні традиційно використовувався КОАТУУ, який нині замінюється Єдиним державним реєстром адрес, визнаним базовим державним реєстром згідно із Законом «Про публічні електронні реєстри» [5] і постановою КМУ №254 [8]. Метою є створення достовірного джерела адрес із унікальними ідентифікаторами для уніфікації міжвідомчих записів. Дані реєстру мають стати основою для всіх державних IT-систем, зокрема ERP-системи НАЗК [1].

1.2 Використання API Укрпошти як джерела достовірних адресних даних

До впровадження повноцінного державного реєстру адрес (ЄДРА) основним актуальним джерелом структурованої інформації про адреси в Україні

є **Адресний класифікатор API Укрпошти**. АТ «Укрпошта», як національний оператор поштового зв'язку, веде власну базу даних поштових індексів і об'єктів (областей, районів, населених пунктів, вулиць, об'єктів поштового зв'язку тощо). Для зовнішніх систем надається спеціалізований веб-сервіс - *Address Classifier API*, який дозволяє отримувати офіційні довідникові дані у режимі **реального часу**. Згідно з офіційною інформацією, Укрпошта пропонує клієнтам декілька API для інтеграції: зокрема, API внутрішніх відправлень, міжнародних відправлень, API для листів та **Адресний класифікатор**[9].

Адресний класифікатор Укрпошти є ієрархічним довідником, що відображає офіційну структуру адміністративно-територіальних об'єктів України. Він дозволяє виконувати пошук інформації про область, район, населений пункт, вулицю або будинок за унікальними ідентифікаторами (Ref-кодами) або назвами. API реалізовано за REST-архітектурою: доступ до нього здійснюється за допомогою HTTP-запитів з обов'язковою авторизацією через токен (Bearer Token). Основні ендпоїнти дозволяють отримати:

- список *областей* (GET /address-classifier/regions),
- список *районів* певної області (GET /districts?regionRef=...),
- список *населених пунктів* певного району (GET /settlements?areaRef=...),
- список *вулиць* у населеному пункті (GET /streets?settlementRef=...),
- перелік *будинків* на вказаній вулиці (GET /buildings?streetRef=...).

Кожен запит повертає структуровані дані у форматі JSON: список елементів із потрібними полями (наприклад, для населених пунктів - settlementRef, settlement (назва), settlementType та ін.). Важливо, що дані організовано ієрархічно: наприклад, ендпоїнт вулиць поверне тільки ті вулиці, що

відповідають заданому settlementRef, що гарантує цілісність структури адреси. Кожен об’єкт (область, місто, вулиця тощо) має *унікальний ідентифікатор* (GUID), який дозволяє однозначно посилатися на нього і пов’язувати між різними системами.

Інтеграція з API Укрпошти забезпечує автоматичну актуалізацію адрес завдяки централізованому довіднику, що оновлюється при кожному перейменуванні об'єктів, особливо в умовах адміністративних реформ. Ієрархічна структура вибірки «область-район-населений пункт-вулиця-будинок» гарантує логічну цілісність даних. Унікальні коди (Ref) спрощують інтеграцію та відстеження змін. Завдяки архітектурі мікросервісів API легко вбудовується у зовнішні системи через стандартні HTTP-запити, зокрема для валідації при введенні адрес у формах [9].

Укрпошта особливо підкреслює вигоди від використання адресного класифікатора для бізнесу. В інтеграційних рішеннях (наприклад, для інтернет-магазинів) API **автоматично перевіряє введені користувачем адресні дані і підбирає поштовий індекс**, якщо той не вказаний або некоректний[9]. Це дозволяє уникнути помилок адресування та затримок із доставкою відправлень. Як зазначено на офіційному порталі, **інтеграція адресного класифікатора, що автоматично оновлюється, значно спрощує процес доставки і усуває можливість помилок при формуванні адресного ярлика**[4]. Тобто, у контексті нашого завдання - нормалізації адрес в ERP-системі - підключення до API Укрпошти забезпечує наявність еталонного довідника для звірки та коригування адрес.

Для повноцінної нормалізації недостатньо лише звертатися до API - потрібен і парсинг «сирих» текстових адрес. Оскільки API повертає довідкові дані лише за конкретними запитами, модуль має самостійно аналізувати текст: виділяти ключові елементи (область, місто, вулицю, будинок), зіставляти їх із довідником через API, отримувати Ref-коди та уніфіковані назви, і зберігати в

структурованому вигляді. Це реалізує комбінований підхід, рекомендований у міжнародній практиці: поєднання текстового аналізу з верифікацією через офіційні джерела забезпечує найкращу якість нормалізації.

Отже, огляд інструментів показує, що API Укрпошти є оптимальним джерелом достовірних адресних даних в Україні на даний момент. Його використання в проєкті створення модуля нормалізації адрес забезпечить відповідність записів офіційним класифікаторам та їхню актуалізацію без ручного підтримання довідників. У наступних розділах буде розглянуто, як саме інтегрувати цей API в ERP-систему НАЗК та які алгоритми парсингу необхідно розробити.

1.3 Методи оцінки впливу IT-середовища на успішність проєкту

Розробка модуля в ERP НАЗК відбувається в контексті динамічного IT-середовища - сукупності зовнішніх і внутрішніх факторів, що впливають на проєкт. До зовнішніх належать технічні стандарти, нормативні вимоги, оновлення API (наприклад, Укрпошти), які можуть створювати як нові можливості, так і виклики. Внутрішнє середовище включає технічну інфраструктуру, професійний рівень команди, організаційну культуру, взаємодію зі стейкхолдерами. Зміни в будь-якому з цих елементів можуть впливати на терміни і якість реалізації. Успіх проєкту значною мірою залежить від врахування цих чинників на всіх етапах розробки.

Зважаючи на це, сучасні методології управління проєктами приділяють значну увагу **аналізу та оцінці впливів середовища** на проєкт ще на ранніх етапах. В рамках ініціації проєкту рекомендується проводити аналіз зацікавлених сторін та *PESTLE-аналіз* зовнішнього оточення (Political, Economic, Social, Technological, Legal, Environmental factors), щоб виявити ключові чинники, які можуть вплинути на хід робіт. Для внутрішнього середовища застосовують аналіз сильних і слабких сторін організації (SWOT-аналіз), а також

оцінку **готовності ІТ-інфраструктури** до змін (включно з аудитом наявних систем, що будуть інтегровані з новим модулем). Виявлені фактори слід врахувати при плануванні: закласти резерви часу чи ресурсів, розробити плани реагування на ризики, узгодити очікування з замовником.

Одним із ефективних методів кількісного аналізу впливу середовищних факторів на ІТ-проєкти є застосування апарату нечіткої логіки. У таких моделях на вхід подаються нечіткі оцінки параметрів середовища (наприклад, рівень нестабільності, забезпеченість ресурсами), а на виході формується числова оцінка впливу на критичні параметри проєкту - зокрема, можливі відхилення у термінах реалізації або бюджеті. Це дозволяє керівникові проєкту оперативно моделювати наслідки змін (наприклад, нової політики безпеки) та адаптувати план дій. Дослідники підкреслюють, що проактивне управління змінами та постійний моніторинг середовища є ключовими чинниками успіху ІТ-проєкту.

Для нашого проєкту основними факторами зовнішнього ІТ-середовища є: еволюція API Укрпошти (оновлення версій, можливі зміни у форматі даних чи політиці доступу), нормативні вимоги НАЗК щодо захисту даних та сумісності систем, а також загальний прогрес у технологіях обробки даних (наприклад, поява нових бібліотек для парсингу адрес). Внутрішнє середовище включає підтримку керівництва НАЗК, доступність компетентних розробників, стан поточної ERP-системи (чи дозволяє вона легко впровадити новий модуль) тощо. На етапі планування ми врахували ці аспекти: передбачено тестування модуля на сумісність з майбутніми оновленнями API, закладено резерв на можливі зміни вимог, узгоджено питання безпеки з ІТ-департаментом НАЗК. Таким чином, застосування методів оцінки впливів середовища дозволило підвищити стійкість проєкту до невизначеностей і збільшити шанси його успішного завершення у задані строки.

1.4 Характеристика об’єкта автоматизації - ERP-система електронних справ НАЗК

Об’єктом впровадження розроблюваного модуля є **ERP-система електронних справ НАЗК**. Національне агентство з питань запобігання корупції (НАЗК) - це центральний орган виконавчої влади зі спеціальним статусом, основним завданням якого є формування та реалізація державної антикорупційної політики[10]. НАЗК виконує функції контролю та перевірки електронних декларацій посадовців, моніторингу способу життя, веде кілька національних реєстрів (зокрема, Єдиний державний реєстр декларацій, Реєстр корупційних правопорушень тощо)[1]. Для ефективного виконання цих завдань Агентство розробило розгалужене інформаційне середовище, яке включає публічні портали та **внутрішні інформаційні системи**. Однією з таких систем є ERP-система електронних справ - внутрішня корпоративна система, що забезпечує електронний документообіг та облік робочих процесів у НАЗК.

ERP-система електронних справ призначена для централізації інформації про всі випадки, що розглядаються НАЗК, та пов’язані з ними дані. В цій ERP ведуться електронні *справи* щодо перевірок декларацій, корупційних ризиків, перевірки повідомлень від викривачів, листування з іншими органами тощо. Кожна така справа містить різнопланові дані, в тому числі і адресну інформацію - наприклад, місце проживання суб’єкта декларування, адреси об’єктів нерухомості, які перевіряються, адреси контрагентів (органів влади або юридичних осіб). Таким чином, **довідник адрес** є невід’ємною складовою ERP НАЗК, оскільки дозволяє структурувати інформацію і уникати багатократного ручного введення адрес в різних модулях системи.

Поточний стан довідника адрес в ERP НАЗК. На момент початку даного проєкту адресний довідник в системі мав низку недоліків відображених у структурній моделі проблем на Рис 1.1.

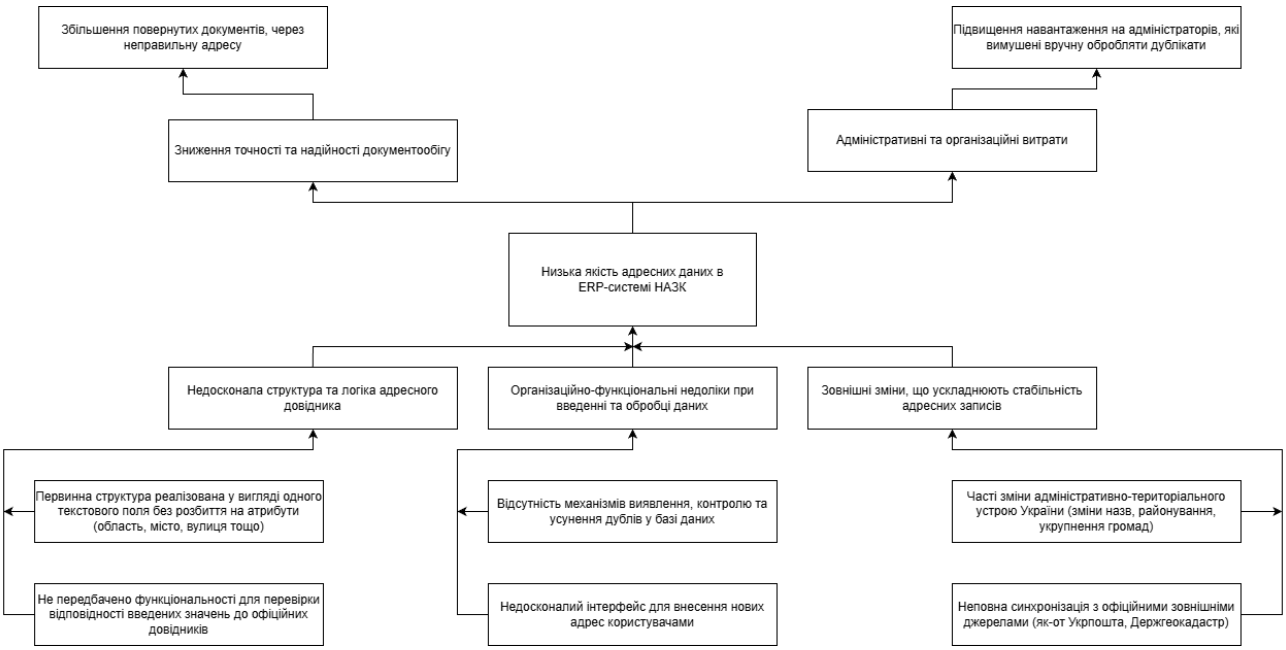


Рис 1.1 Структурна модель проблем проєкту

Існувала єдина таблиця Addresses, поля якої зберігали як *структуровані дані* (коди та назви області, міста тощо), так і *вільний текст* повної адреси. Зокрема, для кожного запису зберігалося: текстове поле raw_address (необроблений рядок адреси, введений користувачем), та окремі поля region_code, region_name, city_code, city_name, street_name, building_number тощо для структури. Ідея полягала в тому, що при додаванні нових записів користувач одночасно вносить адресу і заповнює структуровані поля, використовуючи підказки від API Укрпошти. Однак **історичні записи** (внесені до впровадження інтеграції з API) мали заповненим лише поле raw_address, тоді як всі інші поля були порожніми. Більше того, через відсутність строгих обмежень на рівні БД, навіть нові записи могли містити непослідовні дані (наприклад, неузгоджені *_name і *_code або некоректні комбінації).

Аналіз наявної інформації виявив кілька ключових проблем. По-перше, відсутність нормалізації призводила до того, що назви регіонів, міст і

вулиць зберігалися у вигляді простих текстових значень без зв’язку з довідниками (наприклад, **Regions**, **Cities**), що ускладнювало підтримку цілісності та оновлення даних.

По-друге, система дозволяла довільне введення адрес, що спричинило значну різноманітність форматів (наприклад, «м. Київ, вул. Хрещатик, 1» vs. «Київ Хрещатик 1»), а також неуніфіковані скорочення («область», «obl.», «обл.»), втрати елементів або несистемні пробіли.

По-третє, через відсутність унікальних обмежень дублікати не фіксувалися: записи на кшталт «вул. Лесі Українки, буд. 5» і «Лесі Українки 5» система вбачала різними.

Нарешті, відсутність централізованих механізмів оновлення або ретроконверсії ускладнювала підтримку актуальності довідника, особливо в умовах частих перейменувань населених пунктів та адміністративно-територіальних змін.

Перелічені проблеми сформували потребу в комплексній модернізації адресного модуля. Для визначення напрямів удосконалення була побудована *структурна модель цілей проєкту* (рис. 1.2), яке відображає бажаний стан системи після реалізації запропонованого проєкту.

Рис. 1.2 Структурна модель цілей проєкту

У дереві цілей основна проблема трансформується у ключову **стратегічну мету**: підвищення якості та цілісності адресних даних у ERP-системі НАЗК. Досягнення цієї мети передбачає реалізацію низки підцілей, а саме:

1. структуризація адресних полів відповідно до офіційних атрибутів;
2. автоматична нормалізація «сирих» історичних записів шляхом парсингу та валідації через API Укрпошти;
3. усунення дублювання шляхом виявлення та злиття ідентичних записів;
4. впровадження механізмів контролю цілісності даних (унікальні індекси, обмеження, логіка валідації);
5. удосконалення інтерфейсу введення адрес для зменшення ймовірності помилок з боку користувачів.

Таким чином, **структурні моделі проблем і цілей** у комплексі відображають логіку переходу від виявлених недоліків до системи цілей, яка визначає функціональні та нефункціональні вимоги майбутнього модуля нормалізації адрес.

ERP-система електронних справ НАЗК на сьогодні потребує покращення механізмів роботи з адресними даними. Адресний модуль в її складі є об’єктом модернізації в рамках даного проєкту. Зрозуміння поточного стану (наявних полів, проблем) є відправною точкою для формування вимог до нової підсистеми нормалізації адрес, про що детальніше йдеться далі.

1.5 Постановка задачі і технічне завдання на проєкт

Формулювання проблемної області. На основі вищенаведеного аналізу можна сформулювати проблему: ERP-система НАЗК має значний масив адресних даних, що містять неуніфіковані та частково некоректні записи. Відсутність нормалізації адрес призводить до дублювання інформації, ускладнює автоматичний аналіз та створює ризики помилок при прийнятті рішень (наприклад, під час перевірки декларацій або листування). Отже,

проблемна область охоплює задачі підвищення якості даних, стандартизації та очищення адресної інформації в інформаційно-аналітичній системі державного органу.

Мета дослідження - підвищити якість і цілісність адресних даних в ERP-системі електронних справ НАЗК шляхом розробки та впровадження модуля автоматизованої нормалізації адрес з використанням API адресного класифікатора Укрпошти.

Для досягнення цієї мети необхідно вирішити такі **основні задачі дослідження**:

1. **Аналіз предметної області і вимог:** провести детальний аналіз існуючої структури зберігання адрес в ERP НАЗК, виявити недоліки (дублі, помилки, відсутність полів), вивчити можливості API Укрпошти та інших методів нормалізації адрес. На основі аналізу сформулювати вимоги до нового модуля.
2. **Огляд літератури та напрацювань:** дослідити сучасні методи і алгоритми нормалізації адрес (включно з зарубіжним досвідом, інструментами на зразок Libpostal, підходами НЕР тощо), а також методи оцінки впливів ІТ-середовища на реалізацію проєктів, аби врахувати їх при плануванні впровадження модуля в існуючу систему.
3. **Розробка концепції рішення:** спроєктувати архітектуру модуля нормалізації адрес, визначити його місце в структурі ERP-системи. Розробити алгоритми: (a) парсингу текстових адрес; (b) звернення до API та отримання стандартних даних; (c) виявлення та об’єднання дублюючих адресних записів; (d) ведення логів і звітності про виконану нормалізацію.
4. **Реалізація технічного завдання:** підготувати технічне завдання у формі паспорту проєкту, що включає всі ключові параметри (назва проєкту, замовник, розробник, цілі, функціональні та нефункціональні вимоги, критерії успіху, обмеження, ризики і заходи їх мінімізації, план-графік

виконання тощо). Забезпечити, щоб технічне завдання відповідало методичним вимогам і стандартам (зокрема ДСТУ щодо документації).

5. **Прототипування та тестування:** реалізувати прототип модуля нормалізації на тестовому підмножині даних, провести випробування. Переконатися, що модуль правильно розпізнає і нормалізує адреси, інтеграція з API працює належним чином (в т.ч. витримуються вимоги продуктивності - обробка великої кількості записів у розумні терміни).

6. **Оцінка результатів:** проаналізувати, як впровадження модуля впливає на якість даних в ERP (наприклад, зменшення кількості дублів, частки некоректних записів). Розглянути вплив зовнішніх факторів (напр., чи не змінилися вимоги за час розробки, чи враховано усі оновлення API). Визначити, чи досягнута мета проєкту і сформулювати рекомендації щодо промислового впровадження.

Технічне завдання у формі паспорту проєкту. Нижче наведено узагальнений паспорт проєкту розробки модуля нормалізації адрес для ERP-системи НАЗК:

- **Назва проєкту:** Модуль нормалізації адрес для ERP-системи електронних справ НАЗК.
- **Замовник (ініціатор) проєкту:** Національне агентство з питань запобігання корупції (НАЗК), Відділ електроного документообігу.
- **Розробник (виконавець):** Національне агентство з питань запобігання корупції (НАЗК), Відділ аналітичної роботи та інформаційного розвитку(безпосередні виконавці - головні спеціалісти відділу).
- **Мета проєкту:** Поліпшення якості даних та підвищення ефективності ERP НАЗК шляхом забезпечення єдиного формату зберігання адресної інформації і усунення дублювань.
- **Основні функціональні вимоги:**

- Виявлення «старих» нерегламентованих адресних записів (наявність тексту в raw_address при порожніх структурованих полях).
- Автоматичний **парсинг та нормалізація** таких записів: розпізнавання в тексті компонентів адреси, звернення до API Укрпошти для отримання стандартних назв та кодів, заповнення полів region_code, city_code, street_code тощо відповідними значеннями.
- **Усунення дублювань:** встановлення унікального індексу на сукупність основних полів адреси (код області, код міста, код вулиці, номер будинку, номер квартири) для запобігання внесенню дублів у майбутньому; надання інтерфейсу для пошуку потенційних дублів серед існуючих записів і їх об’єднання (з вибором «правильного» еталонного запису).
- **Логування та інформування користувача:** система повинна формувати журнал змін (які записи нормалізовано, які помилки виникли) та відображати підсумкові повідомлення про успішність операції для оператора.

- **Основні нефункціональні вимоги:**
  - Продуктивність: модуль має обробляти не менше 1000 адресних записів за хвилину без суттєвого погіршення відгуку системи.
  - Безпека: доступ до API Укрпошти здійснювати через захищений протокол HTTPS; токен авторизації зберігати шифровано; забезпечити журналізацію дій для аудиту.
  - Масштабованість та підтримуваність: модуль повинен легко адаптуватися до можливих змін у API (конфігурація кінцевих точок, версії); код має бути документований; передбачити можливість повторного запуску процесу нормалізації для нових даних.

- **Обмеження та припущення:** модуль розробляється для внутрішнього користування НАЗК і працюватиме у межах закритої мережі; передбачається, що інтеграція з API Укрпошти відповідає умовам договору з Укрпоштою про надання доступу до сервісу (наявність актуального токена). Також передбачається наявність резервної копії БД перед масовою нормалізацією даних.

- **Критерії успіху проєкту:** успішна нормалізація не менше 95% історичних адресних записів; виявлення всіх дублюючих адрес після впровадження модуля ; позитивні відгуки користувачів (операторів НАЗК) щодо зручності інтерфейсу; відсутність критичних збоїв протягом 1 місяця експлуатації.

- **Ризики проєкту(Детально розписані у розділі 2.3.2 Аналіз та управління ризиками)**

- **Етапи реалізації:**

1. Детальний аналіз вимог (2 тижні) - включно з погодженням ТЗ із замовником.
2. Розробка та тестування парсеру адрес (4 тижні).
3. Інтеграція з API, відлагодження обміну даними (2 тижні).
4. Розробка функції об’єднання дублів, накладення обмежень у БД (2 тижні).
5. Тестування на реальних даних, виправлення помилок (2 тижні).
6. Впровадження у промислову експлуатацію, моніторинг (1 тиждень). Загальна очікувана тривалість проєкту - ~3 місяці. (Діаграма Ганта була додана до Додатку А)

Цей паспорт проєкту відображає узгоджене бачення замовника і виконавця щодо цілей і шляхів реалізації. Він стане основою для контролю ходу виконання та приймання результатів.

РОЗДІЛ 2. МАТЕМАТИЧНА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

2.1. Розробка нових концептуальних моделей інформаційної системи

У цьому розділі розроблено концептуальну модель модуля нормалізації адрес для ERP-системи. Модуль включає користувацький інтерфейс із двома основними вкладками: **«Дублі»** та **«Ненормовані адреси»**. Вкладка **«Ненормовані адреси»** призначена для відображення всіх записів адрес, що ще не приведені до стандартного формату (тобто не нормалізовані). Користувач може переглядати цей список, аби ідентифікувати «сирі» (legacy) записи адрес та підготувати їх до обробки. Вкладка **«Дублі»** відображає групи адрес, які система вважає дублікативними (повторюваними) - тобто такі, що фактично відповідають одному й тому ж об’єкту, але можуть відрізнятися у записі. Таким чином, інтерфейс модуля дозволяє перемикатися між оглядом ненормалізованих адрес і аналізом можливих дублікатів, забезпечуючи зручний інструментарій для очищення даних адрес.

Робота модуля базується на використанні **зведеної таблиці адрес**, яка об’єднує всі адресні записи з ERP-системи, включно з вже нормалізованими та legacy-записами. Зведена таблиця слугує єдиним джерелом правди для адресної інформації: вона містить як нові записи, що були введені вже у стандартизованому форматі, так і старі записи, отримані з попередніх версій системи або імпортовані ззовні, які можуть мати невпорядкований формат. Кожен запис у цій таблиці має ідентифікатор та атрибути адреси (наприклад, рядок повної адреси, поля для компонентів адреси - область, місто, вулиця тощо). Також може бути передбачено спеціальний атрибут-ознака, який вказує на статус запису: «нормалізований» або «ненормалізований». Модуль оперує цією зведеною таблицею: на вкладці ненормованих адрес відображаються записи зі статусом "ненормалізований", а на вкладці дублів - групування записів

(переважно вже нормалізованих) які мають однакове значення нормалізованої адреси або високий показник подібності.

Для виконання основної функції - перетворення «сирих» адрес у нормалізований вигляд - модуль взаємодіє з компонентом **«Нормалізатор адрес»**. Нормалізатор адрес - це окремий компонент (або зовнішній сервіс/бібліотека), що здійснює алгоритмічну обробку введеного тексту адреси і повертає стандартизовану форму. З погляду модуля, нормалізатор діє як «чорний ящик»: модуль передає йому ненормалізований рядок адреси, а у відповідь отримує нормалізований рядок (або структуру даних з розкладеними полями адреси). Внутрішньо нормалізатор може виконувати такі операції, як **стандартизація форматування** (наприклад, розширення скорочень на зразок «вул.» до «вулиця», перетворення регістру літер), **звірка з довідниками** (наприклад, уточнення назви міста чи області відповідно до офіційних класифікаторів) та **перевірка на повноту** (додавання відсутніх компонентів, як-от поштового індексу, якщо це можливо). Результатом роботи нормалізатора є уніфікований запис, що відповідає правилам оформлення адрес для заданої країни або регіону. Таким чином, модуль нормалізації адрес отримує змогу автоматично приводити всі адреси до єдиного стандарту запису. *Наприклад, якщо в системі є адреса тільки в текстовому форматі «м.Київ, вулиця Хрещатик, 1», нормалізатор може перетворити її на «м. Київ, вул. Хрещатик, буд. 1» і з відповідними індексами Укрпошти.* Це підвищує однорідність даних і спрощує їх подальший аналіз.

Архітектурна схема взаємодії компонентів модуля виглядає таким чином (описово). Користувач ERP-системи, який виконує роль оператора очищення даних, відкриває інтерфейс модуля нормалізації адрес. Інтерфейс звертається до бази даних ERP (MSSQL) і завантажує дані зі зведеної таблиці адрес. Дані розподіляються по вкладках: ненормалізовані адреси відображаються як список записів, що потребують обробки. Коли користувач вирішує виконати нормалізацію, він ініціює взаємодію з нормалізатором (наприклад, натисканням

кнопки «Нормалізувати» для окремого запису або для всього списку). Модуль передає вибрані ненормалізовані адреси до компонента нормалізатора через визначений інтерфейс (це може бути виклик внутрішньої функції або звернення до зовнішнього API-сервісу). Нормалізатор обробляє кожну адресу і повертає нормалізовані дані. Модуль отримує ці результати та оновлює зведену таблицю: для кожного запису проставляється нормалізований варіант адреси (наприклад, заповнюється поле нормалізованої адреси або створюється зв'язок з довідником нормалізованих адрес). Статус записів змінюється на «нормалізовано». Після цього модуль автоматично або за запитом користувача виконує перевірку на **дублікати** серед нормалізованих адрес. Оскільки тепер всі адреси приведені до єдиного формату, виявити дублікати значно легше - достатньо знайти записи з повністю ідентичними нормалізованими рядками або дуже близькими значеннями. Результати перевірки відображаються на вкладці «Дублі»: адреси, що розпізнані як дублікати, можуть бути згруповані разом або представлені списком пар можливих збігів. Користувачеві надається можливість переглянути ці групи і прийняти рішення щодо них (наприклад, об'єднати дублікати, виправити чи видалити зайві записи). Взаємодія між компонентами відбувається таким чином, що база даних виступає центральним посередником: нормалізатор не працює безпосередньо з інтерфейсом, а саме модуль отримує дані з БД, відправляє їх на нормалізацію і назад записує результати. Описана архітектура забезпечує узгоджену роботу всіх компонентів: **користувач** через **інтерфейс модуля** керує процесом, **зведена таблиця адрес** містить актуальні дані, а **компонент нормалізації** виконує спеціалізоване завдання перетворення даних. Такий поділ дозволяє легко підтримувати і оновлювати кожен з елементів системи.

2.1.1 Моделювання архітектури

З метою формального опису роботи модуля нормалізації адрес доцільно побудувати його **концептуальну модель**, яка відображає основні інформаційні потоки, логіку обробки даних та механізми контролю якості. Така модель не прив'язується до конкретної реалізації, але задає загальну структуру взаємодії компонентів.

1. Концептуальна модель обробки потоку адрес

Потік даних у модулі нормалізації може бути подано як послідовність перетворень, що виконуються над множиною вхідних адрес. Формально:

- X- вхідний потік текстових (ненормалізованих) адрес, отриманих із зведеної таблиці ERP;
- f_θ- абстрактний нормалізуючий оператор (модель) з параметрами θ, який реалізує логіку перетворення «сирої» адреси у структурований вигляд (сюди входить як використання LLM, так і інтеграція з довідниками);
- $Y = f_\theta(X)$- результат нормалізації, потік структурованих або стандартизованих адрес;
- D- множина довідників та зовнішніх джерел (класифікатори Укрпошти тощо), до яких звертається нормалізатор у процесі перетворення;
- δ- множина тригерних подій, які ініціюють запуск обробки (натискання кнопки в інтерфейсі, планове завдання за розкладом тощо).

У концептуальній моделі модуль розглядається як **ланцюжок операторів** над потоком X:
очищення й попередня обробка → нормалізація → запис результатів до БД → подальший аналіз (пошук дублікатів). Кожен із цих етапів описується як окрема функція або підмодуль у складі єдиної логічної архітектури.

2. Концептуальна модель контролю, зворотного зв’язку та виявлення дублікатів

На наступному рівні модель доповнюється **контуром контролю та зворотного зв’язку**, що забезпечує адаптивність системи та участь оператора в циклі обробки (human-in-the-loop).

Після отримання нормалізованого потоку Y передбачається запуск підпроцесу **виявлення дублікатів**. У концептуальному вигляді це можна описати як застосування оператора подібності $S(\cdot,\cdot)$до пар або груп адрес у Y, кластеризацію за значенням схожості та перевірку унікальності за ключовими полями (region, district, city, street, building, apartment, postalCode). Результатом є множина кластерів, кожен з яких інтерпретується як потенційна група дублікатів.

Зворотний зв’язок у моделі реалізується через дії оператора-модератора, який у концептуальному вигляді має такі можливості:

- позначати записи, що були нормалізовані некоректно або неповно;
- ініціювати повторну нормалізацію окремих записів або груп адрес;
- коригувати правила обробки, збагачувати довідники, уточнювати параметри моделі f_θ.

У термінах концептуальної моделі це створює **адаптивний цикл**: результати чергової ітерації нормалізації та виявлення дублікатів впливають на налаштування методів і параметрів обробки в наступних ітераціях. Такий цикл відповідає підходу human-in-the-loop для задач із високим ступенем невизначеності та варіативності вхідних даних.

3. Концептуальна модель контролю інформаційних процесів

У контексті управління IT-процесами модуль нормалізації адрес у концептуальному вигляді виконує три ключові функції контролю:

- **Моніторинг якості даних.** Передбачається, що система надає агреговані показники стану адресного довідника: частку нормалізованих записів, кількість ненормалізованих рядків, обсяг потенційних дублікатів, динаміку змін тощо. Ці метрики є елементами інформаційної панелі, яка відображає ефективність функціонування модуля.
- **Аудит та логування.** Концептуальна модель передбачає збереження історії змін адрес: хто, коли і яку нормалізацію або операцію з дублями виконав. Це забезпечує простежуваність усіх трансформацій даних, що є критично важливим для державних інформаційних систем.
- **Планування та запуск обробки.** Тригерні механізми (δ) можуть задаватися як у ручному режимі (ініціація з інтерфейсу користувачем), так і в автоматизованому (розкладні завдання, інтеграційні події). У концептуальній моделі це відображається у вигляді блоку планування обробки, який визначає, які саме адреси, коли і за якими правилами мають бути нормалізовані.

Таким чином, моделювання архітектури та алгоритмів у даному розділі не описує реалізацію на рівні коду, а фіксує **концептуальну структуру** системи: основні сутності (адреси, довідники, оператор нормалізації), потоки їхньої трансформації, контури контролю якості та взаємодію з оператором. Це створює теоретичний каркас, на основі якого у подальших розділах може бути деталізовано проектні рішення та реалізацію модуля в рамках ERP-системи НАЗК.

2.2 Формалізація задачі нормалізації адрес (математична модель)

Для строгого опису задачі нормалізації представимо її у вигляді математичної моделі. Вхідними даними є **нерегламентований текстовий рядок** s, що містить адресу (в довільному форматі), а вихідними даними - **набір структурованих полів** A, упорядкований згідно зі стандартом адрес Укрпошти. Математично можна визначити відображення (функцію):

$$f: S \rightarrow A \quad (1)$$

де S - множина всіх можливих рядків-адрес (над алфавітом символів, допустимих у введенні), а A - множина всіх правильно структурованих адресних записів. Елемент A може бути поданий як кортеж:

$$A = \langle region, district, city, street, building, apartment, postalCode \rangle \quad (2)$$

де кожен компонент є або строкою з нормативною назвою відповідного елемента, або унікальним кодом (ідентифікатором) із довідника. Наприклад, $region$ може бути задано як «Київська область» або кодом 05 (умовно) в системі КОАТУУ, $postalCode$ - це 5-значний поштовий індекс.

Функція нормалізації f не є тотожно визначеною на всьому S (не кожен довільний рядок - валідна адреса). Тому доцільно розглядати її як **часткову функцію**, визначену на підмножині $S_{val} \subset S$ що відповідає адресам, які можуть бути інтерпретовані. Задача нормалізації полягає в тому, щоб для $\forall s \in S_{val}$ знайти $f(s) = A$; якщо $s \notin S_{val}$ (рядок не піддається інтерпретації через брак інформації чи надто велику кількість помилок), то результатом може бути спеціальне значення (наприклад, $f(s) = \emptyset$ або позначка про помилку).

Для визначення S_{val} введемо формальну модель структури української адреси. Будемо вважати, що правильна адреса повинна відповідати певному шаблону (моделі), який базується на ієрархії адміністративно-територіальних об'єктів України. Спрощено, адреса може бути представлена як послідовність компонент у форматі:

$$AddressString \coloneqq [PostalCode,]\ CityOrVillage,\ StreetAddress,\ Region. \quad (3)$$

приклад можливого шаблону, де:

- $PostalCode$ - 5-значний індекс (опційний елемент, може бути на початку або в кінці);
- $CityOrVillage$ - назва населеного пункту з указанням типу (місто, селище, село тощо) або прийнятим скороченням («м.» для міста, «смт» для селища міського типу, «с.» для села і т.д.);
- $StreetAddress$ - назва вулиці з типом (вул., просп., бульв., пров. тощо) та номер будинку (і, за наявності, дробом або літерою), а також номер квартири (позначається як «кв.» або іншими скороченнями);
- $Region$ - назва області (або місто державного підпорядкування), може скорочуватися як «обл.», або не вказуватися для міст центрального підпорядкування (Київ, Севастополь).

Формально, можна розгорнути цю структуру у вигляді продукцій граматики (нехай не строгої BNF, але як псевдо-БНФ для розуміння):

<address> ::= [<postal_code>] <locality_part> <street_part> [<region_part>]

<postal_code> ::= <DIGIT><DIGIT><DIGIT><DIGIT><DIGIT> (п'ять цифр)

<locality_part> ::= (<city_type> ".")? <locality_name> (наприклад: "м. Київ" або просто "Київ")

<city_type> ::= "м" | "смт" | "с" | "м-ко" | ... (скорочення типів нас. пунктів)

<locality_name> ::= ... (послідовність букв та пробілів, що формують назву з великої літери)

<street_part> ::= (<street_type> ".")? <street_name> <house_number> [<apartment_part>]

<street_type> ::= "вул" | "просп" | "бул" | "пров" | ... (можливі скорочення типів вулиць)

<house_number> ::= <NUMBER> [<LETTER>] ["-" <NUMBER>] (номер будинку, опційно з корпусом або діапазоном)

<apartment_part> ::= ("," | ";")? <apartment_type> <NUMBER> (опційно квартира, відокремлена комою або іншим знаком)

<apartment_type> ::= "кв" | "офіс" | "кв-" | "оф." | ... (скорочення, що передують номеру квартири/офісу)

<region_part> ::= <region_name> ("область" | "обл." | "р-н")? (наприклад: "Київська обл." або "Дніпропетровська")

Наведену спрощену граматику слід розуміти гнучко: реальні адреси можуть відхилятися від цього порядку. Саме тому *задача нормалізації* передбачає не просто жорстке парсинг відповідно до одного шаблону, а й **логічне розпізнавання** з урахуванням різних допустимих варіацій. Математично це можна подати як задачу пошуку такого розбиття вхідного рядка s на підрядки $(s_1, s_2, \dots, s_n)$ і відображення g цих підрядків на відповідні поля $(p_1, p_2, \dots, p_n)$, що:

1. Конкатенація $s_1 + s_2 + \cdots + s_n$ (з пробілами/роздільниками) відновлює вихідний рядок s (критерій повноти розбору);
2. Для кожного i підрядок s_i належить домену допустимих значень для поля p_i (критерій коректності: наприклад, токен "Київська" має відповідати області зі списку, а "12" - числу будинку);

3. Послідовність полів $(p_1, \dots, p_n)$ утворює валідну комбінацію з точки зору адресної ієрархії (критерій узгодженості: наприклад, якщо визначено поле району, то область поля $region$ повинна містити цей район; якщо місто «Київ», то поле області може бути пустим або «м. Київ» - спеціальний статус).

Таким чином, формально задача зводиться до пошуку **оптимального розбиття та відображення** $s_i \to p_i$, що задовольняє зазначеним критеріям. У випадку неоднозначності (коли один і той же рядок може відповідати більш ніж одному набору полів A) потрібно визначити правила вибору *найбільш правдоподібного* розбору. Це можна трактувати як задачу оптимізації або *класифікації*: обрати той результат A , який максимізує деяку функцію впевненості або відповідності. У нашому контексті, критерій оптимальності може базуватися на **повноті збігу з довідником** (скільки компонент вдалося знайти в офіційних класифікаторах) та **мінімальній редакційній відстані** виправлень (наскільки мало відрізняються знайдені довідникові назви від вхідних підрядків).

Наприклад, якщо s = м Київ Хрещатик 1 , то можливі інтерпретації:

- $A_1 = \langle$місто Київ,(без району),м. Київ,вул. Хрещатик, $1, \emptyset, 01001\rangle$ (Київ як місто державного значення, вулиця Хрещатик, буд.1, індекс 01001);
- $A_2 = \langle$область Київська,район Київський,село Хрещатик, $1, \dots\rangle$ - завідомо невідповідний варіант (бо «Київ» інтерпретовано як область, а «Хрещатик» ніби село - такої комбінації не існує в класифікаторі).

Алгоритм повинен віддати A_1 як єдиний правильний результат. Для цього він спирається на базу знань (довідник), де «Київ» знайдено як місто, а «Хрещатик» - як вулицю у цьому місті. Формально, на множині усіх можливих

розборів A_i обирається єдиний A_{best} , що відповідає критеріям валідності та мінімізує невідповідності.

Отже, математична модель визначає функцію нормалізації f через граматичні та логічні правила розпізнавання, накладає обмеження на допустимі комбінації полів і дає формальне підґрунтя для реалізації алгоритму. У наступному підрозділі розглянемо, як ця модель реалізується у вигляді конкретних методів парсингу.

2.3. Управління розробкою проєкту

2.3.1 Аналіз та вибір процесів розробки проєкту і методології управління

Ефективна організація робіт із розробки модуля нормалізації адрес для ERP-системи НАЗК потребує не лише технічних рішень, а й обґрунтованого вибору моделі життєвого циклу програмного забезпечення та методології управління проєктом. Специфіка задачі полягає у високій варіативності вхідних даних (адреси у довільних форматах, наявність значної частки «legacy»-записів, помилки введення, неуніфіковані скорочення), а також у необхідності поєднання кількох технологічних компонентів: парсингу текстових рядків, звірки та збагачення даних через API Укрпошти, процедур унікалізації, а також застосування великої мовної моделі для обробки складних випадків. За таких умов каскадні моделі типу Waterfall, які передбачають повне фіксування вимог на старті та низьку адаптивність у процесі реалізації, є методологічно невиправданими: ймовірність виявлення концептуальних або логічних похибок лише на завершальних етапах є підвищеною, а вартість виправлень у системі, що вже інтегрована в ERP-контур, зростає непропорційно.

З огляду на це, для даного проєкту доцільно застосувати ітеративну модель життєвого циклу з елементами гнучких підходів (Agile). Такий підхід дозволяє декомпозувати складну задачу нормалізації адрес на послідовність керованих ітерацій, у межах яких результат перевіряється на реальних даних і за

необхідності уточнюється. На початковому етапі формується базовий прототип (MVP) підсистеми, який реалізує мінімально достатній набір функцій: первинне розбиття адреси на ключові компоненти, базову інтеграцію з довідниками Укрпошти для валідації кодів, а також початкові сценарії роботи з підозрілими дублями. Подальші ітерації спрямовані на поетапне ускладнення логіки: уточнення правил парсингу, удосконалення механізмів зіставлення з довідниками, включення великої мовної моделі для роботи з «важкими» випадками (неповні/змішані/помилкові записи), а також розширення адміністративного інтерфейсу для контролю дублів і ручної верифікації.

Організацію управління проєктом доцільно будувати як **гібрид Scrum-Kanban**, адаптований до реальної управлінської структури та регулярності взаємодії між підрозділами. Власником продукту в межах ERP-розробок виступає НАЗК як організація, а **замовником** у прикладному сенсі є **відділ контролю та підтримки електронного документообігу**, який формулює потребу та очікуваний результат. Функцію погодження технічного завдання і тижневого плану робіт виконує **керівник департаменту**, який на рівні управління забезпечує пріоритизацію та узгодження робіт із потребами організації. Подальші уточнення щодо технічних рішень - як на етапі формування ТЗ, так і в процесі виконання - здійснюються **керівником відділу аналітичної роботи та інформаційного розвитку** і **головним спеціалістом відділу**, які деталізують вимоги, пропонують реалізаційні підходи та забезпечують виконання робіт у межах технологічних обмежень ERP-платформи.

Практична реалізація гнучкого управління забезпечується сталою **ритмікою комунікацій і артефактів**. На початку кожного тижня проводиться нарада з керівником департаменту (щопонеділка), на якій визначається перелік задач на тиждень, уточнюються пріоритети та очікувані результати. За підсумком ці задачі фіксуються у **Slack Canvas** як погоджений тижневий план (аналог спринт-беклогу/плану ітерації), після чого протягом тижня ведеться

їхній статус: “в роботі”, “виконано”, “перенесено”, із зазначенням причин перенесення або зазначенням результатів виконання задачі. Паралельно, у межах відділу проводяться **щоденні короткі мітинги** (stand-up) щодо задач попереднього дня і плану на поточний день: уточнюються статуси, виявляються блокери, погоджується план дій, а задачі та їхні стани візуалізуються на **фізичній канбан-дошці**. Така комбінація стратегічного тижневого планування та щоденного контролю забезпечує одночасно керованість (через регулярне узгодження з керівництвом) і гнучкість (через швидке реагування на нововиявлені особливості даних та зміну пріоритетів).

У межах цього підходу фактично формалізуються й ключові елементи Scrum, адаптовані під організаційну практику НАЗК. Рольове навантаження розподіляється таким чином, що функції визначення пріоритетів і приймання результатів реалізуються через ланцюг “замовник - керівник департаменту”, а функції координації технічних рішень та усунення перешкод - через керівника профільного відділу. Артефактами управління виступають: (1) погоджений тижневий план задач у Slack Canvas, (2) перелік задач і їхня візуалізація на канбан-дошці у відділі, (3) проміжні результати у вигляді прототипів/покращень компонентів (парсер, інтеграція API, механізм дублів, інтерфейси), які можуть бути продемонстровані та прийняті поетапно. Це дозволяє забезпечити трасованість: від узгодженої потреби -> до конкретної задачі -> до реалізованого фрагмента функціоналу.

Суттєвим елементом запропонованої методології є **вбудоване управління якістю**, інтегроване у кожну ітерацію. На відміну від моделей, де тестування розглядається як фінальна стадія, у даному проєкті перевірка точності нормалізації та виявлення дублювань включається в цикл розробки. Для цього формуються контрольні вибірки адрес (зокрема «проблемні» записи: без розділових знаків, зі змішаними скороченнями, із помилками тощо), для яких визначаються еталонні розбиття та очікувані значення компонентів. На кожному циклі здійснюється вимірювання цільових показників: частка коректно

структурованих записів, кількість виявлених/усунених дублів, відсоток адрес, що потребують ручної корекції. Ці показники використовуються як критерії управлінського контролю: вони дозволяють аргументовано ухвалювати рішення про готовність переходу до наступної ітерації, про необхідність доопрацювання правил або про уточнення вимог замовника.

Окремо в межах обраної методології формалізується процедура **приймання результатів (acceptance)**, що є критичною для державного ІТ-проєкту, де будь-які зміни в ERP-системі впливають на роботу декількох підрозділів. Приймання реалізується на рівні **MVP кожного функціонального елемента** нового модуля (інтерфейсні вкладки, механізм нормалізації, інтеграція з довідниками, сценарії роботи з дублями тощо) і включає погоджені критерії успішності, демонстрацію в тестовому середовищі та ухвалення рішення про доопрацювання або перенесення в промислову експлуатацію.

На практиці процедура приймання організовується у форматі робочої наради з представниками **департаменту Управління документообігу та контролю**, зокрема із керівниками відділів цього управління та залученими головними спеціалістами. Під час наради **керівник відділу аналітичної роботи та інформаційного розвитку** та **головний спеціаліст** демонструють створений функціонал на **dev-середовищі**, виконують коротке навчання користувачів (пояснюють логіку роботи та типові сценарії використання) і показують результати на реальних прикладах даних. Після демонстрації проводиться обговорення: фіксуються зауваження, узгоджуються необхідні покращення, визначаються підрозділи, яким потрібно надати доступ до функціоналу, а також погоджується орієнтовний момент перенесення змін у **prod-середовище**.

У межах такого підходу **acceptance criteria** доцільно описувати у вигляді набору перевірюваних умов, що дають можливість об’єктивно підтвердити готовність функціоналу. Для модуля нормалізації адрес такими критеріями можуть виступати:

1. **функціональна відповідність**: реалізовано всі сценарії, погоджені в ТЗ для конкретного MVP (наприклад, перегляд ненормованих адрес/груп дублів, запуск нормалізації, фіксація результатів, можливість ручного коригування або підтвердження);
2. **коректність на контрольній вибірці**: на заздалегідь узгодженому наборі адрес модуль демонструє прийнятну частку правильного розбиття/ідентифікації компонентів та виявлення дублів;
3. **стабільність роботи на dev-сердовищі**: відсутність критичних помилок під час демонстрації та повторюваність результатів для однакових вхідних даних;
4. **обмеження доступу та ролі**: визначено, яким підрозділам надається доступ і які операції їм дозволені (перегляд, редагування, підтвердження, об’єднання, ініціювання нормалізації);
5. **готовність до перенесення на prod-середовище**: узгоджено перелік змін, що будуть розгорнуті, порядок перенесення та умови відкату/припинення змін у разі виявлення критичних проблем.

Таким чином, управлінська процедура приймання у проєкті виконує дві функції: (1) забезпечує контроль якості через перевірювані критерії, та (2) забезпечує керованість змін через погодження впливу на підрозділи, доступи і план перенесення у продуктивне середовище.

Для підтримки прозорості та відтворюваності процесів розробки доцільно застосовувати сучасний організаційно-технічний інструментарій. Розробка ведеться в середовищі, що підтримує рефакторинг та статичний аналіз, а контроль версій забезпечує фіксацію історії змін, роботу з гілками й можливість рев’ю. Управлінські артефакти (плани, статуси, перенесення задач) фіксуються в Slack Canvas, а оперативний контроль виконання забезпечується канбан-візуалізацією всередині відділу. Така схема дає можливість не лише контролювати завантаженість команди, але й накопичувати доказову базу для

подальшого аналізу виконання проєкту (що було заплановано, що реалізовано, які були блокери та як їх усунено).

Отже, обраний ітеративно-гнучкий підхід до організації життєвого циклу модуля нормалізації адрес є обґрунтованим з огляду на характер задачі та управлінський контекст НАЗК. На відміну від **традиційної каскадної моделі (Waterfall),** де присутня низька гнучкість: фіксація вимог на початку і відсутність можливостей для змін, які можуть призводити до ризику виявлення помилок лише на фінальних етапах та високої вартості по часу, який потрібен на їх усунення, обрана метадологія знижує ризики, пов’язані з невизначеністю та варіативністю вхідних даних, забезпечує можливість послідовного уточнення алгоритмів на основі тестування та зворотного зв’язку, а також створює умови для системного контролю якості і прозорого управління комплексом робіт у межах організаційної структури виконавців і замовника.

2.3.2 Аналіз та управління ризиками

Розробка та впровадження модуля нормалізації адрес в ERP-системі електронних справ НАЗК належить до IT-проєктів із підвищеною критичністю до **якості даних**, **безперервності роботи** та **керованості змін** у продуктивному середовищі. Адресний довідник використовується як опорний елемент у реєстрації документів, формуванні звітності та міжмодульній інтеграції, тому помилки нормалізації або неконтрольовані зміни можуть мати не лише технічні, а й організаційні та регуляторні наслідки. У зв’язку з цим управління ризиками в проєкті розглядається як системний процес, що включає: (1) ідентифікацію ризиків, (2) кількісну (напівкількісну) оцінку, (3) пріоритизацію, (4) планування заходів реагування, (5) моніторинг тригерів і перегляд оцінок у межах ітераційного циклу.

1. Класифікація ризиків проєкту

Для забезпечення повноти аналізу ризики згруповано за категоріями, що найбільш релевантні до специфіки модуля нормалізації адрес:

- **організаційні ризики** (стейкхолдери, зміни процесів, приймання результатів, навчання користувачів);
- **технічні (технологічні) ризики** (сумісність з ERP-архітектурою, продуктивність БД, стабільність інтеграцій);
- **ризики даних та якості інформації** (помилки нормалізації, дублікати, втрата контексту, змішування “неадресних” даних);
- **правові та регуляторні ризики** (актуальність адміністративно-територіальних даних, коректність відображення офіційних назв);
- **ресурсні та часові ризики** (наявність людського ресурсу для ручної валідації, зсув строків через ітерації);
- **ризики використання зовнішнього LLM-сервісу** (залежність від сервісу, точність, відповідальність і контроль даних).

2. Методика кількісної оцінки та пріоритизації ризиків

Для переходу від описового переліку ризиків до керованої системи пріоритетів застосовано напівкількісну методику оцінювання за двома шкалами: **ймовірність (P)** та **вплив (I)**.

Шкала ймовірності (P):

1 - дуже низька (<10%); 2 - низька (10-30%); 3 - середня (30-50%); 4 - висока (50-70%); 5 - дуже висока (>70%).

Шкала впливу (I):

1 - незначний (локальні незручності без впливу на процеси);

2 - помірний (обмежене погіршення якості/часу, яке усувається в межах ітерації);

3 - суттєвий (помітний вплив на якість довідника або роботу підрозділів, потрібні додаткові ресурси);

4 - критичний (порушення ключових процесів, ризик зупинки частини функцій або масові помилки даних);

5 - катастрофічний (істотний збій ERP/масова некоректність довідника з високою ціною відновлення та значними організаційними наслідками).

Інтегральний пріоритет ризику визначається як **R = P × I**.

Для управлінських рішень застосовано такі рівні:

- **низький (R=1-5):** достатньо моніторингу та стандартних заходів;
- **середній (R=6-12):** потрібен план реагування та контроль тригерів;
- **високий (R=13-19):** потрібні превентивні заходи до впровадження та регулярний контроль керівного рівня;
- **критичний (R=20-25):** впровадження можливе лише за наявності резервних сценаріїв, регламенту відкату та посиленого контролю.

У межах проєкту матриця ризиків реалізується як розподіл ризиків за зонами (низька/середня/висока/критична) на основі пар (P, I). Це забезпечує прозору пріоритизацію: у фокусі управління знаходяться ризики з високим впливом на ERP-стабільність, інтеграції та якість даних, тоді як низькі ризики контролюються через спостереження і періодичний перегляд.

3. Реєстр ключових ризиків із оцінками P/I та управлінськими рішеннями

Нижче наведено пріоритетні ризики проєкту з напівкількісними оцінками та запланованими механізмами управління.

R1. Опір змінам з боку користувачів ERP-системи

P=4, I=3 → R=12 (середній).

Сутність ризику полягає у можливому формальному використанні інструменту або спробах обходу правил нормалізації, що зберігатиме низьку якість даних.

Тригери: повторна поява raw-only записів, зростання частки неповних адрес, збільшення звернень у підтримку. Управлінське рішення - керований change-management: навчання, уточнення ролей доступу, контроль якості за вибірками та зворотний зв'язок із підрозділами, які масово вводять адреси.

R2. Недостатня залученість стейкхолдерів і розходження очікувань
P=3, I=4 → R=12 (середній).

Ризик проявляється у невідповідності функціоналу реальним процесам документообігу або у доопрацюваннях після демонстрацій.

Тригери: часті зміни вимог після MVP, суперечливі запити різних відділів.

Управлінське рішення - формалізовані процедури приймання: демонстрації на dev, погоджені acceptance criteria, фіксація рішень щодо доступів, план перенесення у prod.

R3. Деградація продуктивності ERP через зміни БД (індекси/обмеження/тригери)
P=3, I=5 → R=15 (високий).

Оскільки довідник адрес є широко використовуваним, додаткові індекси та перевірки можуть впливати на транзакційність і блокування.

Тригери: зростання часу операцій, блокування, погіршення відгуку інтерфейсу у пікові періоди.

Управлінське рішення - поетапне ввімкнення змін, тестування на dev зі співставними обсягами, контроль SQL-планів, регламент відкату та обмеження на розгортання без позитивних результатів тестування продуктивності.

R4. Збої інтеграцій (API Укрпошти): недоступність, зміна формату, rate limit
P=4, I=4 → R=16 (високий).

Ризик прямо впливає на актуальність довідників і здатність автоматичного зіставлення компонентів адреси.

Тригери: серії помилок інтеграції, нестабільні відповіді, різке уповільнення нормалізації.

Управлінське рішення - резервні сценарії: локальні довідники/кеш, повторні спроби із backoff, журналювання, аварійний режим роботи без API з подальшою синхронізацією.

R5. Некоректна або неповна нормалізація адрес (особливо legacy-записів) P=4, I=4 → R=16 (високий).

Найбільш критичний клас ризиків, оскільки помилки нормалізації створюють хибні зв’язки в довідниках та спотворюють звітність.

Тригери: високий відсоток ручних виправлень, системні помилки у типових шаблонах адрес, скарги користувачів на «не ту адресу».

Управлінське рішення - режим **human-in-the-loop**, контрольні вибірки, накопичення бібліотеки типових помилок, збереження raw та історії трансформацій, а також приймання результатів нормалізації як окремий контроль якості.

R6. Втрата контексту первинного запису під час трансформації P=2, I=5 → R=10 (середній).

Вплив потенційно високий, але ризик керований правильною моделлю даних: збереження первинного рядка та історії змін.

Тригери: неможливість відновити первинну адресу або пояснити джерело змін.

Управлінське рішення - аудит: raw_address + нормалізований варіант + журнал змін із користувачем/часом + можливість відкату.

R7. Змішування адресних і неадресних даних у legacy-рядках P=4, I=3 → R=12 (середній).

Ризик полягає у наявності в одному полі ПІБ/контактів/довільного тексту.

Тригери: виявлення email/телефонів у полі адреси, записи без ознак топонімів.

Управлінське рішення - попередні фільтри, маркування нерелевантних записів, окремий сценарій видалення після перевірки використання у зв'язках.

R8. Невчасне оновлення офіційних назв та АТО
P=3, I=4 → R=12 (середній).

Наслідки проявляються в юридично чутливих документах.

Управлінське рішення - регламент оновлення довідників, контроль змін, протокол оновлення та перевірка узгодженості.

R9. Дефіцит ресурсу для ручної валідації нормалізації
P=4, I=3 → R=12 (середній).

Ризик безпосередньо впливає на строк очищення довідника.

Управлінське рішення - включення валідації як окремих задач у тижневий план, пріоритизація адрес, що реально використовуються, пакетна обробка з контролем якості.

R10. Перевищення строків через повторні ітерації та дефекти інтеграції
P=3, I=3 → R=9 (середній).

Управлінське рішення - резерв часу, обмеження scope на ітерацію, чіткі acceptance criteria на MVP, ранні демонстрації.

4. Ризики використання LLM та управлінська відповідальність

Використання LLM (ChatGPT 5) у проєкті має специфічні управлінські наслідки, оскільки мова йде про залежність від зовнішнього сервісу і про обмеження на автоматичне прийняття рішень. У даній роботі LLM розглядається не як компонент, що вносить зміни у продуктивний довідник автоматично, а як інструмент первинної структуризації legacy-масиву, який не інтегрується в саму

ERP-систему, з обов’язковою перевіркою людиною та подальшим зіставленням із довідниками.

R11. Залежність від зовнішнього LLM-сервісу (доступність/умови/вартість) P=3, I=4 → R=12 (середній).

Управлінське рішення - винесення LLM-обробки у позаопераційний контур, документування результатів, можливість повторної обробки або переходу на альтернативний сценарій (правила/ручна обробка пріоритетних адрес) без блокування базових процесів ERP.

R12. Ризик помилкового розбору LLM та відповідальність за наслідки P=4, I=4 → R=16 (високий).

Управлінське правило: результат LLM не є автоматично правильним, а лише гіпотезою для перевірки. Приймання результатів нормалізації відбувається через модератора/відповідального працівника. Лише після валідації та прив’язки до довідників (зокрема через дані Укрпошти) запис переводиться у статус нормалізованого. Це знижує управлінський ризик непомітних помилок, які потрапляють у основний довідник.

R13. Ризик неправомірної обробки/витоку даних при роботі з LLM P=2-3, I=5 → R=10-15 (середній/високий залежно від режиму). Оскільки legacy-рядки можуть містити неадресні фрагменти, управління ризиком передбачає: попереднє очищення і фільтрацію (email/телефон/явні персональні маркери), мінімізацію змісту пакетів, контроль доступів до файлів експорту, а також журналювання етапів підготовки даних і відповідальних осіб.

5. План реагування: prevention / mitigation / contingency та моніторинг

Для кожного ризику визначаються три рівні управлінських дій:

1. **Prevention (превентивні заходи)** - дії, які зменшують ймовірність настання ризику (напр., тестування на dev, навчання користувачів,

регламент оновлення довідників, обмеження прав доступу, збереження raw та історії).

2. **Mitigation (пом'якшення наслідків)** - дії, які знижують вплив, якщо ризик реалізувався (наприклад, швидке виправлення правил/промптів, розширення контрольної вибірки, ручна валідація особливих випадків, корекція індексів).
3. **Contingency (резервні сценарії)** - дії на випадок критичних відмов (наприклад, робота без API з кешем, відкат змін БД, тимчасове вимкнення унікальних обмежень з подальшим очищенням у офлайн-режимі, обмеження функціоналу до базового).

Моніторинг ризиків реалізується у рамках регулярного управлінського циклу: на початку тижня ризики високої зони ($R \geq 13$) переглядаються керівним рівнем разом із пріоритетами робіт, а в межах щоденних коротких обговорень у відділі фіксуються поточні тригери (збої інтеграцій, накопичення черги валідації, системні дефекти нормалізації). Результати демонстрацій MVP на dev та приймання результату із підрозділами документообігу виконують функцію додаткового контролю - фактично це перевірка, що ризики якості і придатності функціоналу не переходять у критичну фазу перед розгортанням у prod.

Таким чином, система управління ризиками у проєкті модуля нормалізації адрес ґрунтується на поєднанні напівкількісної оцінки (P/I), зонування ризиків за пріоритетом, превентивних і резервних сценаріїв та регулярного моніторингу у межах ітераційної організації робіт. Окремо формалізовано управлінські обмеження застосування LLM: використання як інструмента попередньої структуризації з обов'язковою людською валідацією та контролем даних, що дозволяє отримати практичний ефект при збереженні керованості та відповідальності в державному IT-контурі.

2.3.3 WBS проєкту

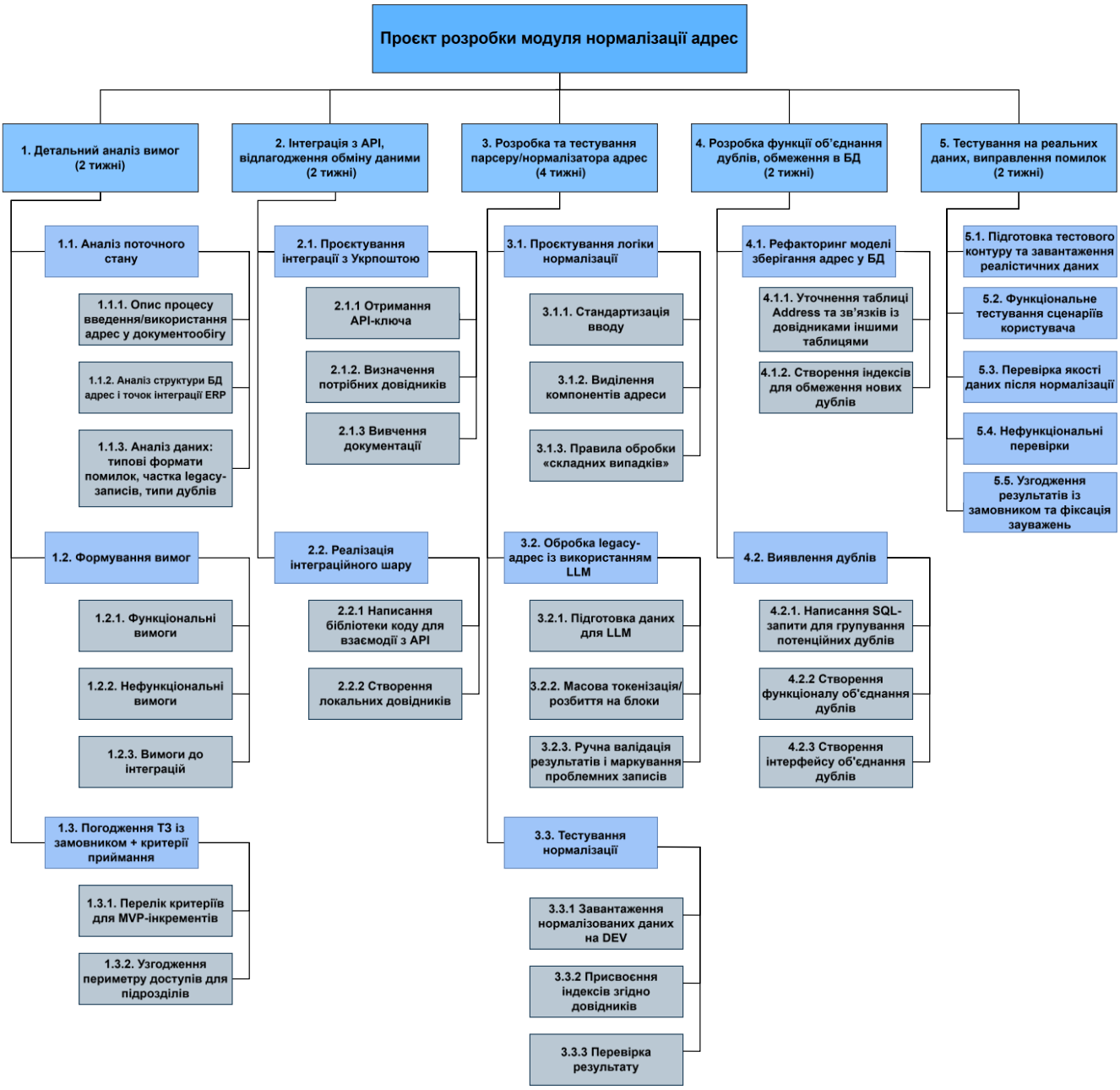


Рис. 2.1 WBS-схема проєкту

Для забезпечення керованості робіт із розробки модуля нормалізації адрес у ERP-системі електронних справ НАЗК у проєкті застосовано ієрархічну структуру робіт (Work Breakdown Structure, WBS). WBS у даному випадку виконує роль базового інструмента управління змістом: вона формалізує межі проєкту, задає логіку декомпозиції до рівня керованих пакетів робіт, слугує основою для календарного планування та визначення контрольних точок, а також забезпечує прозорий контроль виконання і змін. Принципово важливо, що

WBS дозволяє встановити однозначну відповідність між запланованими роботами та очікуваними результатами, а також уніфікує комунікацію між виконавцями і стейкхолдерами (кожен пакет робіт має чітку мету, вхідні дані та вимірюваний вихід).

Побудову WBS виконано за підходом deliverable-oriented decomposition, тобто через декомпозицію за результатами: кожен елемент структури прив'язаний до конкретного проміжного продукту (артефакту) або функціонального інкременту - наприклад, узгоджених вимог і критеріїв приймання, інтеграційного шару з API Укрпошти, механізмів парсингу/нормалізації та індексування, підсистеми роботи з дублями, а також комплексу тестування й процедур впровадження. Такий підхід є доцільним для проєкту, в якому значна частина ризиків пов'язана з якістю даних і залежністю від інтеграцій: орієнтація на результати дозволяє поетапно перевіряти готовність компонентів, обмежувати вплив змін на суміжні частини ERP та забезпечувати кероване нарощування функціональності через приймання інкрементів. Загальну структуру WBS наведено на рисунку (Рис. 2.1).

2.3.4 Організаційна структура проєкту

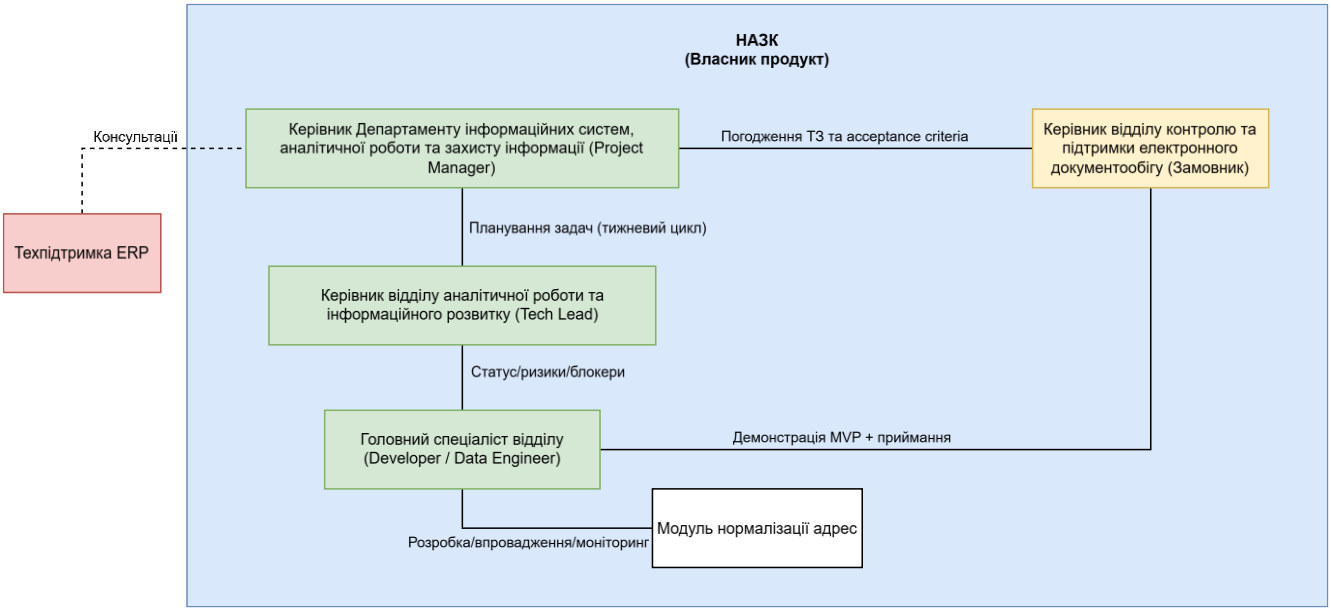


Рис 2.2 OBS-схема проєкту

Організаційна структура проєкту (Рис. 2.2) визначає ролі учасників та їх зони відповідальності, а також забезпечує однозначне зіставлення організаційних ролей із пакетами робіт WBS. У проєкті застосовано принцип розмежування відповідальності: **керівник департаменту** є відповідальним за управління змістом, пріоритетами та погодженням результатів на рівні стейкхолдерів; **керівник відділу аналітики** відповідає за технічну реалізацію та якість інкрементів; **головний спеціаліст** виконує розробку і підготовку результатів до демонстрації; **підрозділ-замовник (Відділ контролю та підтримки електронного документообігу)** здійснює експертну оцінку та приймання за критеріями acceptance criteria; **техпідтримка ERP** забезпечує відповідність змін вимогам експлуатації, продуктивності та безпеки.

У межах WBS відповідальність розподілено таким чином:

- **Пакети аналізу вимог і погодження ТЗ** (збір вимог, формалізація, погодження): відповідальний керівник департаменту; технічне опрацювання та уточнення забезпечує керівник відділу аналітики і головний спеціаліст; замовник підтверджує коректність вимог і критеріїв приймання.
- **Пакети проєктування та розробки** (архітектура модуля, зміни БД, інтерфейси, інтеграція з API/довідниками, механізми унікалізації): відповідальний керівник відділу аналітики; виконавець - головний спеціаліст; техпідтримка ERP консультує щодо експлуатаційних обмежень і контролю впливу на продуктивність.
- **Пакети нормалізації legacy-даних та контролю якості** (пакетна обробка, перевірки, журналювання, контрольні вибірки): відповідальний керівник відділу; виконавець - головний спеціаліст; замовник бере участь у валідації результатів і визначенні правил обробки спірних випадків.

- **Пакети тестування та приймання MVP**: демонстрація здійснюється керівником відділу та головним спеціалістом на dev-середовищі; приймання проводиться представниками Управління документообігу та контролю на основі попередньо погоджених acceptance criteria; рішення щодо доступів і перенесення в prod погоджується на рівні керівництва.
- **Пакети впровадження та моніторингу**: техпідтримка ERP відповідає за консультацію, щодо перенесення змін у промислове середовище й технічний моніторинг; керівник відділу та головний спеціаліст відповідають за усунення дефектів і супровід; керівник департаменту контролює досягнення управлінських цілей і стабільність процесу.

Таким чином, OBS у проєкті використовується не лише як опис складу учасників, а як інструмент управління відповідальністю: кожному пакету WBS відповідає визначена роль відповідального та ролі залучених учасників, що забезпечує трасованість між планом робіт і організаційною структурою виконавців.

2.4. Обґрунтування вибору методів інтелектуального аналізу даних та алгоритмів реалізації

Вибір методів інтелектуального аналізу даних та програмно-алгоритмічного забезпечення для модуля нормалізації адрес має базуватися не лише на теоретичній придатності окремих підходів, а й на реальних обмеженнях проєкту: часових, ресурсних та технологічних. У випадку ERP-системи НАЗК додатковим визначальним фактором є використання наявної low-code/enterprise-платформи, що задає допустимий стек технологій для реалізації інтерфейсів та бізнес-логіки.

Аналіз типової якості історичних адресних записів показав, що дані містять велику кількість «шуму»: змішані формати, розмовні скорочення, орфографічні помилки, неповні або некоректні конструкції. Для досягнення високої точності нормалізації у таких умовах класичні алгоритми, орієнтовані на регулярні вирази

чи жорстко задані правила, потребували б дуже великого обсягу ручної розробки та постійного супроводу. Розгортання ж повноцінної моделі машинного навчання (HMM, CRF, спеціалізовані трансформери тощо) потребувало б формування великої розміченої вибірки адрес, виділення обчислювальних ресурсів для навчання та подальшого моніторингу моделі. За наявних організаційних обмежень (дефіцит часу та людських ресурсів, необхідність швидкого впровадження результатів у продуктивне середовище) та з урахуванням того, що задача є допоміжною в рамках ширшої ERP-системи, такий шлях був визнаний малопрактичним.

У зв'язку з цим як основний інструмент інтелектуального аналізу для нормалізації легасі-адрес було обрано **велику мовну модель ChatGPT 5 (LLM)**. Ця модель здатна виконувати семантичний розбір природномовного тексту без спеціального донавчання на локальних даних, працюючи у режимі *zero-shot* або *few-shot* із використанням лише ретельно сформульованих підказок (prompts). Змістовно це дозволяє покласти на LLM найтрудомісткіший етап - інтерпретацію довільного текстового рядка адреси (з урахуванням контексту, порядку слів, типових помилок) та поділ його на логічні компоненти: область, населений пункт, вулиця, номер будинку тощо. Таким чином, складна задача розроблення й підтримки вузькоспеціалізованої моделі NER (Named Entity Recognition) замінюється побудовою чітко регламентованого сценарію взаємодії з вже існуючою високопродуктивною LLM.

Водночас імовірнісний характер відповіді великої мовної моделі та відсутність гарантій 100-відсоткової точності робить неприйнятним автоматичне застосування її результатів без додаткового контролю. Тому обрано **гібридний підхід «LLM + експертна верифікація»**. На першому етапі ChatGPT 5 використовується як інструмент попередньої нормалізації: для кожної «сирої» адреси формується структурована пропозиція (набір полів із потенційно правильними значеннями). На другому етапі ці дані підлягають обов'язковому перегляду та затвердженню модератором (аналітиком/адміністратором

довідника), який за потреби коригує компоненти адреси, звіряє їх із довідниками Укрпошти та ухвалює остаточне рішення. Така схема «людина в контурі» (human-in-the-loop) дозволяє поєднати високу продуктивність LLM з вимогами до надійності та юридичної коректності даних у державній установі.

Технічна реалізація повного модуля нормалізації детермінується архітектурою наявної ERP-платформи, що базується на технологіях екосистеми **.NET**. Це обумовлює використання **C#** як основної мови бізнес-логіки та побудови інтерфейсів. Для реалізації робочих форм і віконних інтерфейсів застосовується поєднання **WPF-класів** (для більш гнучких, сучасних UI-компонентів) та **WinForms** (для типових форм введення/редагування даних, що вже існують у системі та добре інтегровані з іншим функціоналом). Такий підхід забезпечує спадкоємність із наявними екранами ERP, мінімізує витрати на навчання користувачів та дозволяє інтегрувати нові можливості в уже усталені робочі процеси.

Робота з базою даних виконується через вбудовану **ORM-бібліотеку C#**, що абстрагує SQL-запити у вигляді об’єктних моделей та методів. Це спрощує підтримку коду, знижує ризик помилок у ручних SQL-конструкціях і забезпечує єдиний підхід до доступу до таблиць (зокрема, до оновленої таблиці адрес із новою структурою полів та індексів). На рівні самої СУБД застосовуються обмеження цілісності, унікальні індекси та зовнішні ключі, які реалізують формальні критерії однозначності адрес і підтримують математичну модель дедуплікації, описану у попередніх підрозділах.

Додаткову роль у програмному стеку відіграє **Python**. У межах проєкту його пропонується використовувати для задач, де потрібні гнучкі засоби обробки даних або побудови умов відображення даних у інтерфейсах, а також для написання допоміжних скриптів. До таких задач належать, наприклад, підготовка вибірок адрес для відправки до LLM, аналіз статистики якості нормалізації, автоматизоване формування проміжних файлів (JSON/CSV) для

імпорту в ERP-систему, а також окремі елементи логіки, які зручніше реалізувати в середовищі наукових бібліотек Python. Результати роботи Python-скриптів інтегруються з основним модулем через обмін файлами або через проміжні таблиці в базі даних.

Таким чином, обраний підхід до алгоритмічного та технологічного забезпечення модуля нормалізації адрес є компромісом між теоретично можливою «ідеальною» моделлю та практичними обмеженнями державної ERP-системи. Використання LLM ChatGPT 5 дозволяє суттєво скоротити трудомісткість обробки неструктурованих легасі-адрес, тоді як ручна експертна перевірка гарантує відповідність результатів вимогам до якості та надійності даних. Поєднання C#, SQL (через ORM) та Python у межах існуючої low-code-платформи забезпечує технічну сумісність, можливість подальшого розширення функціоналу та зручність супроводу системи силами внутрішніх IT-фахівців НАЗК.

Модуль нормалізації адрес є типовим прикладом задачі, де поєднуються класичне моделювання, методи ШІ та підходи до керування інформаційними потоками. Завдяки математичній формалізації, нечітким правилам та активному зворотному зв’язку із користувачем досягається висока точність обробки даних, що особливо важливо для держсектору та систем із великим обсягом гетерогенних записів.

РОЗДІЛ 3. РОЗРОБКА ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ ПРОЄКТУ

3.1. Розробка концептуальної моделі інформаційної системи

Концептуальна модель інформаційної системи визначає основні **джерела даних**, **потоки інформації** та **взаємодію компонентів** у проєкті модуля нормалізації адрес. На верхньому рівні можна виділити такі ключові елементи і процеси даної системи:

- **Джерело даних (вхідні дані)** - це ERP-система, а саме її база даних, в якій містяться адреси. Вхідними даними для нашого модуля є *сукупність адресних записів із ERP*. Дані можуть надходити потоком: наприклад, при запуску модуль завантажує всі адреси, які потребують нормалізації (нові чи змінені записи). Також джерелом можуть бути legacy-дані, перенесені зі старих систем у загальну таблицю адрес.
- **Вихідні дані (результати)** - це інформація, яка формується модулем нормалізації і надається користувачу або іншим компонентам системи. До вихідних даних відносяться: **Список (звіт) дубльованих адрес** - результати аналізу дублікативності, де зазначено, які записи є потенційними дублями. Цей список може бути представленим у вигляді груп адрес або пар збігів, і він, по суті, є інструментом для користувача по очистці: на його основі можна приймати рішення про об’єднання записів або інші дії. Вихідні дані можуть зберігатися у БД (наприклад, таблиця, що фіксує ідентифікатори дубльованих записів, чи відмітка біля кожного запису про знайдені дублікати), а також відображатися інтерактивно в інтерфейсі.
- **Потоки даних** між компонентами: **потік даних** починається з отримання з ERP-бази ненормалізованих адрес при відкритті модуля. Після цього користувач може ініціювати процес нормалізації - адреси надсилаються нормалізатору, а у відповідь повертаються нормалізовані

дані, які записуються назад у базу. Наступний потік - **потік виявлення дублікатів**. Модуль аналізує оновлені адреси, застосовує алгоритми пошуку схожих записів (наприклад, через SQL-запити чи функції подібності), та формує списки потенційних дублікатів. **Зворотний зв'язок** від користувача відіграє ключову роль: підтвердження дублікатів, ручне редагування або спростування збігів породжує **тригерні події** - наприклад, повторну нормалізацію або помітку запису як такий, що потребує злиття. Такий механізм забезпечує повноцінну інтерактивність та адаптивність системи.

- **Тригерні події в інтерфейсі** виникають також при надходженні нових даних. Припустімо, в ERP-системі додається новий клієнт з адресою - ця подія (створення нового запису адреси) може бути тригером для модуля нормалізації, який автоматично (або через чергу завдань) підхопить цю адресу і спробує її нормалізувати, додавши результат до таблиці адрес. Якщо модуль інтегровано тісно з ERP, такий тригер може бути реалізований як виклик процедури нормалізації при вставці запису. У разі більш автономної роботи модуля, нові адреси з'являться у списку «ненормованих» для подальшої обробки оператором.

Отже, концептуальна модель відображає **повний цикл** роботи з адресними даними: від моменту, коли «сирі» дані потрапляють у систему, до моменту, коли вони виходять у вигляді корисної інформації (стандартизовані адреси та очищені від дублювань списки). Взаємозв'язок компонентів можна підсумувати так: **ERP-джерело адрес -> модуль нормалізації (UI + логіка) -> нормалізатор (функція/сервіс) -> база даних (оновлення) -> модуль нормалізації (аналіз дублікатів) -> користувач (результати та дії)** Рис. 3.1 . Така потокова схема забезпечує узгодженість даних і актуальність інформації: база даних постійно оновлюється результатами нормалізації, а користувач через інтерфейс отримує як вхідні, так і вихідні дані практично в реальному часі.

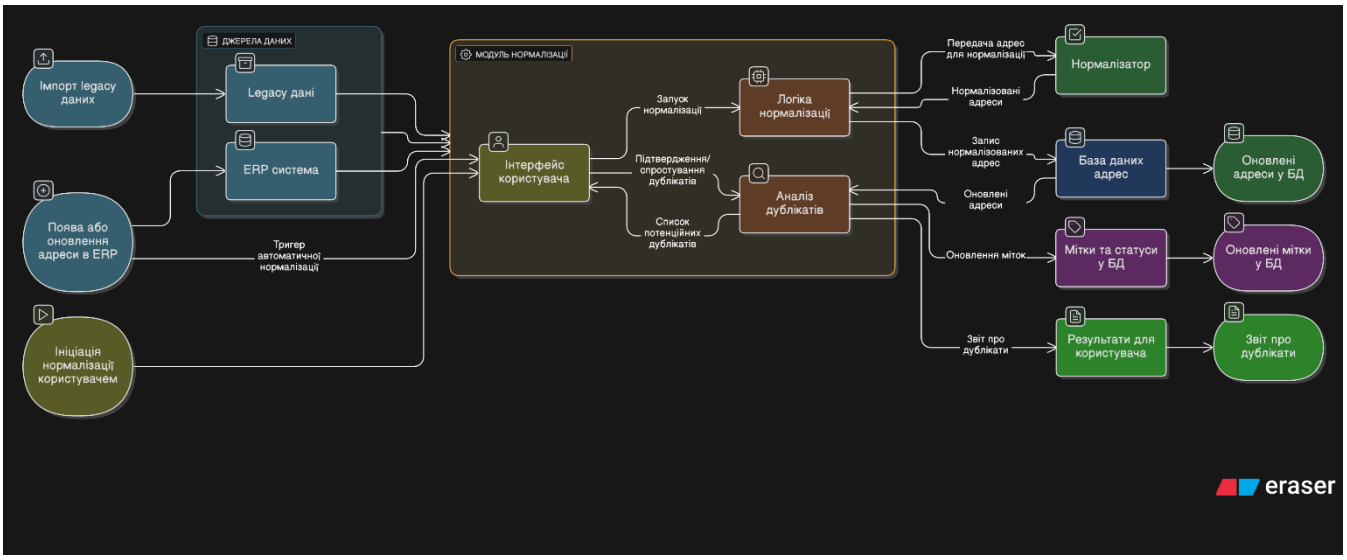


Рис. 3.1 Концептуально модель модуля нормалізації адрес

3.1.1. Визначення функціональних та нефункціональних вимог

На основі концептуальної моделі та очікуваної поведінки системи сформулюємо **функціональні вимоги** до модуля нормалізації адрес. Функціональні вимоги визначають, *що саме повинна робити система*, які операції і послуги надавати користувачу. Основні функції модуля такі:

1. **Перегляд ненормованих адрес**

Модуль повинен відображати список усіх ненормалізованих записів на вкладці «Ненормовані адреси». Дані подаються у табличному вигляді з можливістю сортування та фільтрації (наприклад, за містом). Кожен запис має містити текст адреси, дату додавання та джерело.

2. **Редагування адрес**

Користувач може вручну редагувати адреси через окрему форму. Після збереження запис переводиться в статус «ненормалізований» для подальшої обробки. Це дозволяє виправити помилки перед нормалізацією.

3. **Запуск нормалізації**

Можлива обробка однієї або кількох адрес через кнопки «Нормалізувати» чи «Нормалізувати всі». Результати оновлюються в БД. Адреси, що не вдалося обробити, позначаються відповідно. Може відображатися індикатор прогресу.

4. **Виявлення дублікатів**

Функція активується автоматично після нормалізації або вручну. Система знаходить адреси з однаковими або дуже схожими нормалізованими даними. Вкладка «Дублі» показує групи можливих дубльованих записів з ідентифікаторами та контекстною інформацією. Доступні дії: об’єднання або виключення зі списку дублікатів.

Окрім функціональних, система повинна відповідати ряду **нефункціональних вимог**, які характеризують якість роботи, сумісність та інші властивості:

1. Продуктивність

- Інтерфейс має бути чутливим до дій користувача, з мінімальними затримками при відображенні великої кількості записів (наприклад, через пагінацію або часткове завантаження).
- Нормалізація однієї адреси виконується за секунди, масова - у фоновому режимі.
- Пошук дублікатів має бути оптимізований (наприклад, через SQL-індекси), без повного перегляду таблиці.

2. Сумісність з MSSQL

- Структура БД і всі запити повинні бути реалізовані на T-SQL з урахуванням специфіки MSSQL (типи NVARCHAR, порівняння рядків, індекси тощо).

- Модуль інтегрується з існуючою ERP-інфраструктурою, допускається використання CLR-функцій.
- Заборонено застосування несумісних технологій.

3. Безпека

- Доступ до функцій обмежується ролями користувачів ERP.
- Всі дії (редагування, об’єднання тощо) мають журналюватися.
- При взаємодії з зовнішнім сервісом передача даних здійснюється через HTTPS із можливим знеособленням.
- Забезпечується захист від SQL-ін’єкцій, перевірка довжини полів та обробка некоректного введення.

Окрім зазначених, можуть існувати інші нефункціональні вимоги: **надійність** (щоб модуль не призводив до збоїв ERP, мав механізми відновлення після помилок), **масштабованість** (можливість розширення на більші обсяги даних або додавання нових функцій), **юзабіліті** (інтуїтивність інтерфейсу для кінцевого користувача) тощо. Але в контексті даного проєкту особливо виділяються саме вищеперелічені - продуктивність, сумісність, безпека та інтернаціоналізація, як ключові показники успішності впровадження модуля.

3.1.2. Формування Use Case елементів до функціональних вимог та побудова Use Case Diagram

На основі функціональних вимог можна описати основні **сценарії використання (Use Case)** для модуля нормалізації адрес. Кожен сценарій представляє собою завершену послідовність дій користувача і системи, що веде до досягнення певної мети. Наведемо ключові Use Case-сценарії:

1. **Перегляд ненормованих адрес.**

Актор: Користувач (оператор очищення даних).

Передумови: У системі є ненормалізовані адреси; користувач має доступ до модуля.

Основний сценарій: Користувач відкриває вкладку «Ненормовані адреси» в інтерфейсі ERP. Система виконує запит до БД і отримує список записів, що позначені як ненормалізовані. Список адрес відображається на екрані. Користувач може прогортати список, здійснювати пошук по тексту адреси або відфільтрувати записи (наприклад, по місту). Таким чином, актор отримує повну картину адрес, які потребують нормалізації.

Результат: Користувач проінформований про обсяг робіт з нормалізації; жодних змін у системі не відбувається, лише читання даних.

2. Запуск нормалізації адрес.

Актор: Користувач.

Передумови: Є вибрані адреси для нормалізації (або користувач вирішує нормалізувати всі); нормалізатор доступний.

Основний сценарій: У вкладці «Ненормовані адреси» користувач обирає записи та натискає «Нормалізувати» або «Нормалізувати всі». Система надсилає адреси до нормалізатора, отримує нормалізовані дані й оновлює відповідні записи: зберігає стандартну форму, змінює статус на «нормалізовано», прибирає їх зі списку. У разі помилки або кількох варіантів - залишає записи з відповідним повідомленням для подальшого перегляду.

Розширення (альтернативний потік): Якщо нормалізація масова і займає час, можливий підсценарій: система відображає прогрес-бар або повідомлення «Виконується нормалізація...», дозволяючи користувачу чекати або виконувати інші дії. По завершенні - повідомлення «Нормалізацію завершено успішно (N адрес оброблено)».

Результат: Обрані адреси збережені у нормалізованому форматі. БД оновлена, список ненормалізованих скоротився. В системі потенційно з’явилися нові нормалізовані записи, готові до використання і для перевірки на дублікати.

3. Перегляд дубльованих адрес.

Актор: Користувач.

Передумови: У системі є хоча б дві адреси, що дублюються (після нормалізації).

Основний сценарій: Користувач відкриває вкладку «Дублі». Система або зчитує попередньо знайдені дублікати, або виконує пошук у реальному часі. Виводиться перелік груп дублікатів (наприклад: «Адреса X - ID1, ID2, ID3»). Користувач може переглянути деталі груп, порівняти записи й обрати дію: «Об’єднати дублікати» (злиття або позначення) чи «Позначити як унікальні» (виключення з пошуку дублікатів).

Результат: Користувач отримує інформацію про дублікати і може вжити заходів для очистки. Сама по собі операція перегляду не змінює дані, окрім випадків, коли користувач ініціює об’єднання/позначення (тоді відбуваються зміни: записи зливаються або помітки дубль/не дубль записуються в БД).

4. Оновлення адреси (ручне виправлення).

Актор: Користувач.

Передумови: Користувач виявив помилку в адресі або отримав зовнішні коригування (наприклад, уточнення від клієнта).

Основний сценарій: Користувач відкриває запис адреси (через пошук або список), вносить зміни у формі (наприклад, уточнює вулицю чи додає корпус) і натискає «Зберегти». Система валідує дані, оновлює БД і змінює статус запису

на «ненормалізований». За потреби одразу запускається повторна нормалізація та перевірка на дублікати (автоматично або вручну).

Результат: В базі даних оновлено інформацію про конкретну адресу. Дані актуалізовано. Якщо були запущені супутні процеси (повторна нормалізація, перевірка дублів), їх результати теж відображаються відповідно. Користувач виправив помилку, забезпечивши більш коректні дані для системи.

На основі наведених сценаріїв побудовано **діаграму варіантів використання (Use Case Diagram)** Додаток Б для модуля. У цій діаграмі відображено взаємодію між акторами та випадками використання. **Актори**, залучені в нашій системі, такі:

- **Користувач** - основний актор, що взаємодіє з модулем. Це може бути, наприклад, співробітник відділу даних або адміністратор ERP, відповідальний за чистоту довідників. Користувач ініціює всі розглянуті вище use case: перегляд списків, запуск процесів, редагування. На діаграмі він зображений зліва від системи і має зв’язки (асоціації) з кожним з чотирьох основних варіантів використання.
- **Нормалізатор адрес** - другорядний актор, який представлений як зовнішня система або служба. Хоча з точки зору реалізації це компонент нашої системи, на діаграмі його доцільно відобразити як окремий актор, оскільки він виконує роль постачальника послуги нормалізації. Use case «Запуск нормалізації адрес» пов’язаний асоціацією з актором «Нормалізатор адрес», що відображає: в рамках цього сценарію система звертається до зовнішнього сервісу. Іншими словами, «Нормалізатор» тут грає роль, схожу на актор - він не ініціатор, але учасник взаємодії, відповідає на запити системи.
- **Система** - на діаграмі варіантів використання під системою розуміється сам модуль нормалізації адрес (або ширше - ERP-система з інтегрованим модулем). Як правило, система в UML-діаграмі не

зображається як актор, а є межою (boundary), всередині якої знаходяться use case. Однак, у нашому описі зазначено «актор система», маючи на увазі, ймовірно, що деякі сценарії можуть ініціюватися системою автоматично. Наприклад, **тригерні події**: система (без прямої дії користувача) може сама запустити нормалізацію для нової адреси або оновити список дублів після зміни даних. Такі фонові дії можна інтерпретувати як окремі варіанти використання, де актором виступає таймер або система. На спрощеній діаграмі можна не виводити їх окремо, але текстово слід згадати. Отже, актор «Система» можна трактувати як внутрішній планувальник чи процес, що теж може викликати use case «Запуск нормалізації» (наприклад, за розкладом о 12:00 щоночі) чи «Пошук дублікатів» (після кожного оновлення).

- **API Укрпошти** (зовнішня система, з якою модуль взаємодіє для валідації та отримання даних). - другорядний актор у вигляді зовнішньої системи, яка використовується іншими модулями

У підсумку, Use Case Diagram дає наочне уявлення про функціональність: **«Користувач»** взаємодіє з **«Модулем нормалізації адрес»** через чотири основних варіанти використання, а **«Модуль»** у свою чергу співпрацює з **«Нормалізатором»** при виконанні завдання нормалізації. Це відповідає нашим вимогам і забезпечує, що всі передбачені дії користувача враховані у проєкті.

3.1.3. Розробка концептуальної та логічної моделей бази даних проєкту

На завершення проєктування інформаційного забезпечення потрібно спроєктувати структуру бази даних, що підтримуватиме роботу модуля нормалізації адрес. Це включає побудову **концептуальної моделі даних** і її конкретизацію до **логічної моделі** (таблиці, зв’язки, атрибути, ключі), а також визначення **фізичної моделі** для MSSQL (типи даних, індекси, ключі).

Концептуальна модель бази даних проєкту модуля нормалізації адрес ERP-системи складається з двох основних сутностей:

1. Основна таблиця адрес (Addresses)

Ця сутність є єдиним сховищем для всіх адресних записів, включаючи нормалізовані адреси, ненормалізовані (легасі-записи) та дублікати. Вона забезпечує гнучке і повне представлення адресних даних в єдиній структурованій формі.

Основні атрибути сутності Addresses:

- **AddressID** - унікальний ідентифікатор адресного запису.
- **RegionID** - ідентифікатор регіону (області).
- **Region** - назва регіону (області).
- **DistrictID** - ідентифікатор району.
- **District** - назва району.
- **CityID** - ідентифікатор населеного пункту (міста, селища).
- **City** - назва населеного пункту (міста, селища).
- **StreetID** - ідентифікатор вулиці.
- **Street** - назва вулиці.
- **Building** - номер будинку (включає корпус за наявності).
- **Apartment** - номер квартири або офісу (опціонально).
- **PostalCode** - поштовий індекс.
- **Text** - оригінальний текстовий запис адреси, як вона надійшла в систему (ненормалізований).
- **DateAdded** - дата додавання запису.
- **LastModified** - дата останньої зміни запису.
- **ModifiedBy** - ідентифікатор користувача чи системи, яка внесла зміни.

Адреси, що мають однакові значення полів (RegionID, Region, DistrictID, District, CityID, City, StreetID, Street, Building, Apartment, PostalCode) після нормалізації, розглядаються як дублікати.

2. **Таблиця історії нормалізації (AddressNormalizationHistory)**

Друга сутність є журналом, що фіксує всі зміни, пов’язані з операціями нормалізації, що проводяться над записами таблиці Addresses. Вона забезпечує повний контроль і прозорість операцій, які виконуються модулем нормалізації адрес.

Основні атрибути AddressNormalizationHistory:

- **HistoryID** - унікальний ідентифікатор запису історії.
- **AddressID** - зовнішній ключ, що вказує на відповідний запис у таблиці Addresses.
- **OperationDate** - дата і час проведення операції.
- **PerformedBy** - користувач чи процес, що виконав нормалізацію.
- **OriginalText** - вихідний текст адреси до нормалізації.
- **NormalizedText** - текст адреси після нормалізації (структурована форма).
- **OperationType** - тип операції (автоматична нормалізація, ручна зміна, виявлення дубля).
- **OperationStatus** - результат операції (успішно, помилка, потребує уточнення).

Логічна модель бази даних (для MSSQL)

Рис. 3.2 Логічна модель таблиць модуля нормалізації адрес

Логічна модель Рис 3.2 чітко структурує описані сутності у вигляді конкретних таблиць, атрибутів, зв'язків та індексів для реалізації на платформі MSSQL.

Таблиця Addresses (основна таблиця адрес)

Поля таблиці:

- AddressID - первинний ключ, BIGINT IDENTITY(1,1).
- RegionID - INT, ID регіону.
- Region - NVARCHAR(100), назва регіону.
- DistrictID - INT, ID району.
- District - NVARCHAR(100), назва району.

- CityID - INT, ID міста.
- City - NVARCHAR(100), назва міста.
- StreetID - INT, ID вулиці.
- Street - NVARCHAR(100), назва вулиці.
- Building - NVARCHAR(50), номер будинку.
- Apartment - NVARCHAR(20), номер квартири (NULL, якщо немає).
- PostalCode - NVARCHAR(20), поштовий індекс.
- Text - NVARCHAR(1000), оригінальний текст адреси.
- NormalizationStatus - INT (0 - ненормалізована, 1 - нормалізована, 2 - дубль).
- IsVerified - BIT, підтвердження нормалізації.
- SourceSystem - NVARCHAR(100), джерело запису.
- DateAdded - DATETIME2, дата створення запису.
- LastModified - DATETIME2, дата останньої зміни.
- ModifiedBy - NVARCHAR(100), користувач або система, що змінила запис.

Індекси та обмеження:

- Первинний ключ: AddressID.
- Некластерний індекс на полях (RegionID, DistrictID, CityID, StreetID, Building, Apartment, PostalCode) для швидкого пошуку і визначення дублікатів.

Таблиця AddressNormalizationHistory (журнал змін)

Поля таблиці:

- HistoryID - первинний ключ, BIGINT IDENTITY(1,1).
- AddressID - зовнішній ключ на таблицю Addresses.
- OperationDate - DATETIME2, дата і час операції.
- PerformedBy - NVARCHAR(100), автор операції.

- OriginalText - NVARCHAR(1000), текст адреси до нормалізації.
- NormalizedText - NVARCHAR(1000), нормалізований текст адреси після операції.
- OperationType - NVARCHAR(50), тип операції (автоматична, ручна, дублювання).
- OperationStatus - NVARCHAR(50), статус виконання операції.

Зовнішні ключі та обмеження:

- Первинний ключ: HistoryID.
- Зовнішній ключ AddressID → Addresses(AddressID), ON DELETE NO ACTION.

Індекси:

- Кластерний індекс за OperationDate для зручності вибірки історії за часом.
- Некластерний індекс по AddressID для швидкого отримання історії конкретної адреси.

Логічні зв’язки та цілісність:

У логічній моделі бази даних чітко визначено зв'язок між таблицями:

- Зв’язок між Addresses та AddressNormalizationHistory типу один-до-багатьох (один адресний запис має багато записів історії нормалізації).
- Виявлення дублікатів забезпечується порівнянням нормалізованих атрибутів: групуванням записів за ключовими полями адреси (RegionID, CityID, StreetID, Building, Apartment, PostalCode).

Представлена модель БД забезпечує потреби модуля: ми можемо зберігати оригінальні і нормалізовані адреси, швидко фільтрувати ненормалізовані (для інтерфейсу), знаходити історію нормалізації через спільний NormalizedID,

відслідковувати історію змін для аудиту. В MSSQL ця модель може бути фізично реалізована шляхом виконання скриптів CREATE TABLE з зазначеними полями, ключами та індексами. Наприклад, схема таблиці Addresses та NormalizedAddresses відображатиме зв’язок один-до-багатьох.

3.2 Архітектура та розробка програмного забезпечення

Модуль нормалізації адрес інтегрується в ERP-систему НАЗК на платформі .NET з урахуванням принципу low-code - більшість функціональності реалізується через наявні інструменти ERP без значного обсягу коду. Новий інтерфейс конфігурується стандартними засобами системи, а складні задачі, як-от парсинг адрес, передаються зовнішнім службам.

Модуль складається з чотирьох основних компонентів: інтерфейсу користувача для роботи з адресами, сервісу нормалізації (парсинг і стандартизація), механізму виявлення дублікатів та інтеграції з API Укрпошти. Кожен компонент виконує окрему функцію, що забезпечує модульність і гнучкість системи.

ERP-система містить реляційну базу даних, структуру якої було розширено для підтримки нового модуля. Замість зберігання адрес у системі реалізовано нормалізовану структуру БД: адреси зберігаються через зв’язки на окремі довідники (області, міста, вулиці тощо), а не одним текстовим полем. Це забезпечує цілісність і спрощує оновлення даних.

Для уникнення дублювання в таблиці Address створено складений унікальний індекс на ключові поля (region_id, district_id, city_id, street_id, building_number, apartment_number). Це запобігає внесенню ідентичних записів і дозволяє виявляти дублікати. Історичні повтори визначаються окремим програмним методом.

Модуль нормалізації інтегрується з ERP через абстрактний інтерфейс парсера, що приймає текст адреси та повертає структуровані компоненти. Підтримується реалізація як на C# (.NET), так і через REST API (наприклад, на Python). Завдяки цьому парсер можна змінювати без модифікації основної системи.

Модуль працює асинхронно, використовуючи чергу завдань для обробки адрес. Усі дії адміністраторів журналюються з фіксацією користувача і часу. Передбачено повторні спроби при збоях та сповіщення. Архітектура гарантує надійність і контроль змін.

Також реалізована інтеграція з API Укрпошти [9] для автоматичного оновлення довідників Region, City, District, Street. За потреби він запитує нові або змінені елементи (наприклад, перейменовані вулиці) та синхронізує їх із локальною базою, забезпечуючи актуальність адресних даних.

Отже, архітектура програмного забезпечення модуля нормалізації адрес є багатошаровою і гнучкою. Вона поєднує можливості low-code платформи для швидкого створення інтерфейсів і налаштувань, із можливістю підключення високоспеціалізованих зовнішніх компонент (парсерів) для реалізації складних функцій. Структура рішення (рис. 3.1 у розділі 3) побудована таким чином, щоб ізолювати змінні частини (алгоритми парсингу, правила перевірки) від стабільних (структура даних, інтерфейси), забезпечуючи тим самим легкість подальшої підтримки та розширення функціональності.

3.2.1 Алгоритм виявлення дублікатів

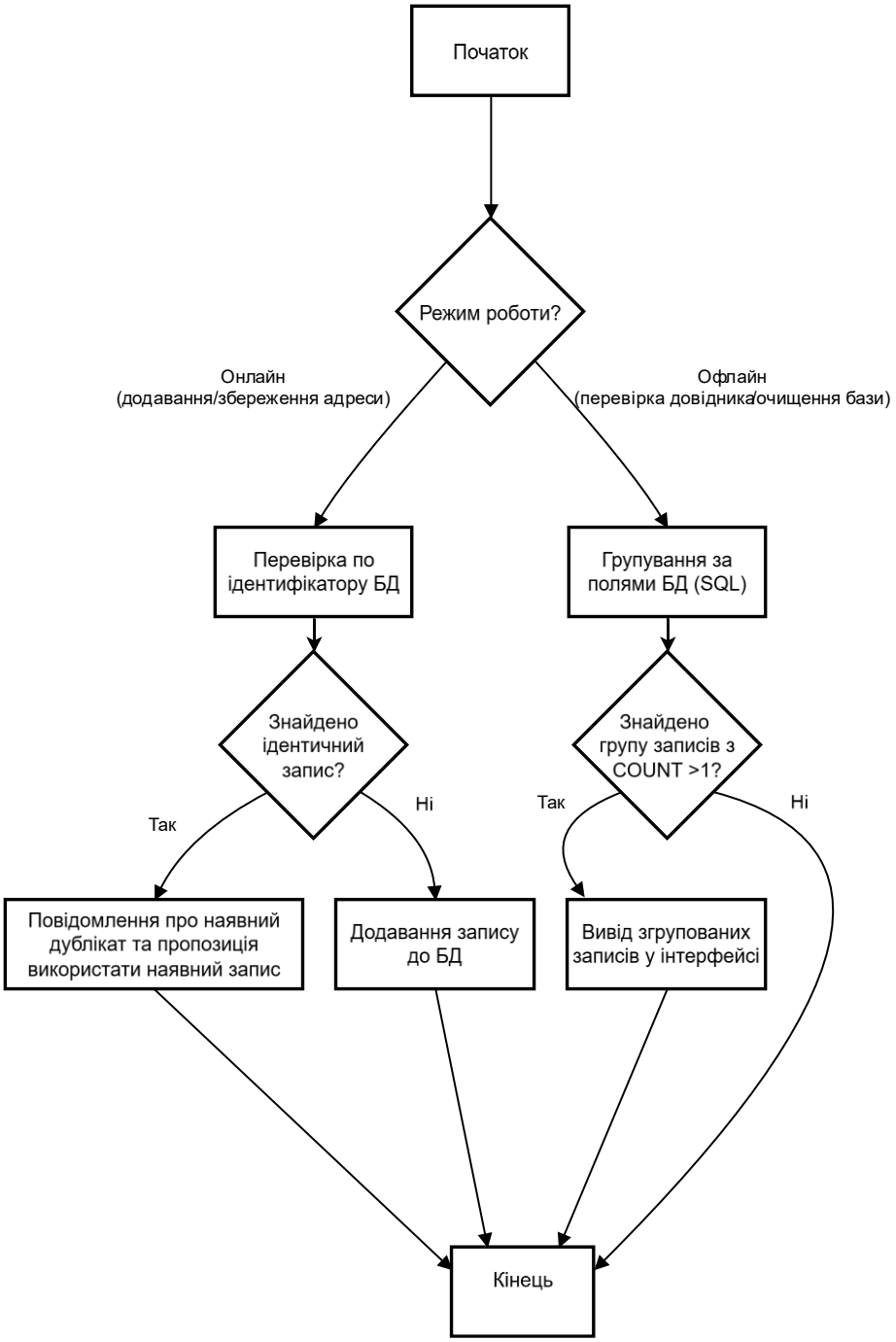


Рис.3.3 Алгоритм виявлення дублікатів

Механізм виявлення дублікатів адрес (Рис. 3.3) реалізовано в рамках модуля з метою **очищення історичних даних** та запобігання повторному введенню ідентичних адрес. Під дублікатом в даному контексті розуміється два або більше записи довідника, що представляють ту саму фактичну адресу. У нормалізованій моделі даних (коли адреса розбита на поля і посилається на довідники) визначення дубліката зводиться до порівняння ключових полів: якщо *region_id*,

district_id, *city_id*, *street_id*, *building_number*, *apartment_number* збігаються у двох записів - це **дублікат**. Саме таке визначення закладено у складений унікальний індекс на таблиці Address. Отже, алгоритм виявлення дублікатів фактично шукає групи записів з однаковими значеннями цих атрибутів.

Пошук дублікатів може працювати у двох режимах: *онлайн* (динамічно при додаванні нового запису) та *офлайн* (перевірка всієї бази). В онлайн-режимі, як згадувалося, перевірка відбувається одразу після спроби збереження нової адреси. Компонент DuplicateDetector (або відповідна процедура на рівні бази) виконує запит: «Чи існує запис у Address з такими самими значеннями полів?». Здійснюється це засобами ORM у коді C# і одночасно SQL-запитом з відповідним фільтром. Якщо результат знаходиться, новий запис не додається до БД, а користувач отримує повідомлення про наявний дублікат (у випадку жорсткого застосування унікальності).

В офлайн-режимі, який використовується для первинного очищення довідника, алгоритм сканує всі наявні записи. Це можна виконати SQL-запитом із групуванням:

```
SELECT region_id, district_id, city_id, street_id, building_number, apartment_number, COUNT(*)
FROM Address
GROUP BY region_id, district_id, city_id, street_id, building_number, apartment_number
HAVING COUNT(*) > 1;
```

Такий запит поверне всі комбінації, що зустрічаються більше одного разу, тобто потенційні дублікати. Далі за кожною такою комбінацією вибираються конкретні записи (їхні ідентифікатори, повні текстові адреси, пов’язані документи тощо) для подальшого аналізу і відображення у відповідному інтерфейсі.

Важливо, що пошук дублів здійснюється **лише на нормалізованих даних**. Тобто алгоритм довіряє тому, що якщо дві адреси мають однакові посилання на довідники і номери, то це справді той самий об’єкт. У випадку, якщо якісь записи ще не нормалізовані (містять лише raw_address), їх дублювання автоматично не виявиться, доки не буде виконана їх нормалізація. Проте на етапі аналізу вхідних даних можна було визначити очевидні текстові дублікати за повним збігом рядків - такі випадки також враховано при формуванні сценаріїв очищення (наприклад, дві абсолютно ідентичні строки адреси в старій системі вважаються дублікатом ще до нормалізації).

Обробка виявлених дублікатів. Алгоритм не видаляє дублікати автоматично без підтвердження, щоб не втратити важливі пов’язані дані (наприклад, документи, що посилаються на адресу). Натомість після виявлення дублюючих записів виконується напівручний процес об’єднання, докладно описаний у розділі 3.3 Інтерфейс модуля. В контексті алгоритму достатньо зазначити, що знайдені дублі обліковуються створенням запису у таблиці Duplicates із зазначенням двох ідентичних адрес. Надалі адміністратор через інтерфейс вирішує, який із записів залишити як основний, а який - видалити чи об’єднати. Такий підхід відповідає кращим практикам забезпечення якості даних: автоматичне виявлення і **флагування дублікатів** поєднується з ручним підтвердженням перед їх усуненням. Відомо, що навіть після формальної нормалізації деякі дублікати можуть вимагати перевірки людиною (наприклад, якщо два записи відрізняються лише скороченнями у назві, система вважає їх одним адресом, але потрібне підтвердження, що це не різні об’єкти).

Алгоритм виявлення дублікатів реалізований у прототипі модуля. Його робота перевірена на тестовому наборі даних: скрипт успішно виявив всі контрольні дублювання, що були навмисно додані в БД для перевірки (див. результати тестування в розділі 4.4). Таким чином, механізм унікалізації даних працює на двох рівнях: превентивно (через унікальні індекси, що

унеможливлюють дублювання при введенні) та реактивно (через пошук і об’єднання дублікатів, що історично існували).

Варто зазначити, що продуктивність алгоритму є достатньою для наявних обсягів даних: запит з групуванням виконується за частки секунди на кількадесят тисяч записів, а перевірка при вставці нового запису здійснюється через індекс за O(1). Тому рішення масштабується на базу довідника адрес НАЗК без проблем. Після очищення даних і впровадження контролю унікальності очікується суттєве підвищення якості довідника та зменшення навантаження на персонал, який раніше виправляв дубльовані адреси вручну і витрачав багато часу на пошук документів, де використані дубльовані адреси.

3.2.2 Нормалізація застарілих записів адрес

Окремим критично важливим завданням у межах розробки модуля нормалізації адрес стала очистка та нормалізація великого масиву історичних (legacy) адресних записів, що тривалий час накопичувалися у внутрішній ERP-системі НАЗК у неструктурованому вигляді. Такі записи, як правило, містили дані лише у полі «повна адреса» (raw_address), тоді як усі структурні атрибути (область, район, населений пункт, вулиця, номер будинку) залишалися незаповненими. Це робило неможливим їх використання у формальних запитах, аналітиці, а також у механізмах контролю дублікатів.

З огляду на обсяг даних та їхню різнорідність, процес нормалізації був реалізований як **гібридна методика**, що поєднує класичні засоби роботи з базами даних (SQL), інструменти штучного інтелекту (LLM ChatGPT 5) та обов’язкову ручну експертну валідацію результатів. Такий підхід дозволив досягти балансу між автоматизацією та надійністю, уникнувши як надмірної трудомісткості, так і неконтрольованих помилок автоматичної обробки.

Етапи очищення та нормалізації

1) Первинна фільтрація та експорт даних

Першим кроком було формальне виділення множини «проблемних» записів із загального масиву адрес. Для цього застосовано спеціалізований SQL-запит, який відбирав тільки ті адреси, що:

- не мали жодних заповнених структурних полів (region_id, district_id, city_id, street_id, building_number тощо);
- містили дані виключно в одному текстовому полі «повна адреса».

Таким чином формувалася чітка підмножина адресних записів, що підлягали нормалізації, без втручання в уже належним чином структуровані дані. Відібраний масив ненормованих адрес було експортовано у формат електронної таблиці (Excel) Рис 3.4, що забезпечило зручність подальшого аналізу, групування та підготовки прикладів для моделей обробки.

1	KKEA_5	ADRES
14438		lemberg_sv@meta.ua
14439		leibinetssulyk@gmail.com
14440		KushchSH@email.ua
14441		kostiaslipchenkoua@gmail.com
14442		kirillsholydko@gmail.com
14443		john.engstrom2@usdoj.gov
14444		info@guardiancapital.com
14445		info@finport.com.ua
14446		hubkabob@i.ua
14447		hrs@police.gov.ua
14448		ermolenkovita@ukr.net
14449		buh_orz_rada@ukr.net
14450		berezovskaa999@gmail.com
14451		autistasobrio@gmail.com
14452		artur293@ukr.net
14453		alinanosylya@i.ua
14454		acn@oecd.org
14455		90351, Закарпатська обл., Виноградівський р-н, селище міського типу Вилок(з), вул.Гагаріна, будинок 8
14456		81092 вул. Сонячна 2 с. Бірки Яворівський р-н Львівська обл
14457		0667444716@ukr.net
14458		пр-т Гагаріна, 12-А, офіс No1407, м. Одеса, 65039
14459		Промислова зона, буд. 35, м. Вараш, Рівненська обл., 34400
14460		а/с No 12, м. Ізюм, Харківська
14461		vasiliy.slyuta@metinvestholding.com
14462		tsybulskyya@gmail.com
14463		themistokleous.sotiris@gmail.com
14464		lduray@wanadoo.fr
14465		мікрорайон Синєочко, с. Шкло, Яворівський р-н, Львівська обл., 81050
14466		вул. Тиха, 5, с. Лісне, Дергачівський р-н, Харківська обл., 62000
14467		вул. Юності, 20а, кв.82, смт Північне, м. Торецьк, Донецька обл., 85280
14468		вул. Осівка, 11, с. Швайківка, Бердичівський р-н, Житомирська обл., 13331
14469		вул. Вокзальна площа, 3, м. Київ, 01032
14470		вул. Кожум'яцька, буд. No 22-а, кв. 2, м. Київ, 04071

Рис 3.4 Фрагмент Excel таблиці ненормованих записів

2) Аналіз стилів запису та формування еталонних прикладів

На другому етапі було проведено ґрунтовний аналіз стилів запису адрес у історичних даних. Було виявлено, що:

- широко використовуються різноманітні скорочені варіанти написання (наприклад, «м. Київ», «Київ м.», «смт», «сел.» тощо);
- часто відсутні або використані некоректно розділові знаки (коми, тире);
- порядок компонентів адреси є нестандартизованим (наприклад, «Хрещатик 1, Київ» замість «м. Київ, вул. Хрещатик, 1»);
- зустрічаються поєднання елементів різних рівнів (область/район/населений пункт) в одному фрагменті.

Для систематизації цих випадків була сформована окрема таблиця, додана у Додатку В , у якій для типових «сирих» рядків наведено приклади правильного розбиття на логічні блоки (область, район, населений пункт, вулиця, будинок). Ця таблиця виконувала роль еталонного набору даних, на який надалі орієнтувалися як під час налаштування запитів до LLM, так і в процесі ручної перевірки результатів.

3) Токенізація та семантичний розбір за допомогою LLM ChatGPT 5

З метою масової обробки великого масиву ненормованих адрес було застосовано функціонал глибокого дослідження великої мовної моделі (LLM) ChatGPT 5. Модель використовувалася як інтелектуальний парсер, здатний:

- виконувати токенізацію адресних рядків (поділ на окремі лексичні елементи);
- здійснювати семантичний розбір, визначаючи, який елемент відповідає місту, селу, вулиці, номеру будинку тощо;

- класифікувати виявлені фрагменти за наперед визначеними ролями (тип населеного пункту, назва вулиці, номер будинку, індекс, додаткові реквізити).

При формуванні запитів до LLM у підказці (prompt) вказувалися як сам текст адреси, так і приклади правильних розбиттів (з еталонної таблиці), що дозволяло моделі орієнтуватися на бажаний формат вихідних даних. Результатом роботи ChatGPT 5 була структурована таблиця, де для кожної вихідної адреси пропонувалися заповнені поля «область», «район», «населений пункт», «вулиця», «номер будинку» тощо. Частину таких результатів продемонстровано на Рис. 3.5.

INDEX_A	NAIMKEA_1	NAIMKEA_2	NAIMKEA_3	NAIMKEA_4	NAIMKEA_5	NAI	ADRES
04071	Україна			Київ	Кожум'яцька	22	вул. Кожум'яцька, буд. No 22-а, кв. 2, м. Київ, 04071
01032	Україна			Київ	Вокзальна площа	3	вул. Вокзальна площа, 3, м. Київ, 01032
13331	Україна	Житомирська	Бердичівський	Швайківка	Осівка	11	вул. Осівка, 11, с. Швайківка, Бердичівський р-н, Житомирська обл., 13331
85280	Україна	Донецька		Торецьк	Юності	20а	вул. Юності, 20а, кв.82, смт Північне, м. Торецьк, Донецька обл., 85280
62000	Україна	Харківська	Дергачівський	Лісне	Тиха	5	вул. Тиха, 5, с. Лісне, Дергачівський р-н, Харківська обл., 62000
81050	Україна	Львівська	Яворівський	Шкло			мікрорайон Синєочко, с. Шкло, Яворівський р-н, Львівська обл., 81050
	Україна			Ізюм		12	а/с No 12, м. Ізюм, Харківська
34400	Україна	Рівненська		Вараш	Промислова зона	35	Промислова зона, буд. 35, м. Вараш, Рівненська обл., 34400
65039	Україна			Одеса	Гагаріна	12	пр-т Гагаріна, 12-А, офіс No1407, м. Одеса, 65039
81092	Україна	Львівська	Яворівський	Бірки	Сонячна		81092 вул. Сонячна 2 с. Бірки Яворівський р-н Львівська обл
90351	Україна	Закарпатська	Виноградівський		Гагаріна	8	90351, Закарпатська обл., Виноградівський р-н, селище міського типу Вилок(з), ву
02167	Україна			Київ		41	а/с 41, м. Київ, 02167

Рис 3.5 Фрагмент таблиці нормалізованих адрес за допомогою ChatGPT 5

4) Ручна валідація та коригування результатів парсингу

Незважаючи на високу здатність LLM до роботи з природномовними текстами, гарантованої безпомилковості вона не забезпечує. Тому наступним обов'язковим етапом стала ретельна ручна експертна валідація отриманих результатів.

На цьому етапі:

- порівнювалися вихідні текстові рядки та запропоновані LLM розбиття на компоненти;
- виправлялися типові помилки, пов'язані, зокрема, з українською специфікою написання назв населених пунктів (апострофи, дефіси,

складені назви): наприклад, уточнювалися варіанти на кшталт «Камянець Подільський», «Олександрівка першотравнева» тощо;

- перевірялося, чи не було переплутано ролі елементів (коли вулиця помилково трактувалася як населений пункт або навпаки);
- вилучалися записи, які фактично не містили адресної інформації, але з технічних причин потрапили до поля адреси (ПІБ клієнта, електронна адреса, телефонні номери тощо).

У ході такої валідації було ідентифіковано та відфільтровано близько 2500 нерелевантних записів, що не могли вважатися адресами. Окремо було виділено близько 2000 записів, які навіть після обробки LLM залишилися структурно неповними або занадто неоднозначними для автоматичної інтерпретації; вони були позначені як такі, що потребують додаткового втручання або можуть бути вилучені за відсутності зв’язків з іншими сутностями системи.

5) Присвоєння довідникових ідентифікаторів та інтеграція з класифікаторами

Після ручного коригування очищений і структурований масив адрес було завантажено у тестове середовище ERP-системи НАЗК для технічної інтеграції з довідниковою підсистемою. На цьому етапі для кожного розпізнаного компонента адреси виконувалося зіставлення з локальними довідниками, синхронізованими з API Укрпошти:

- для області, району, населеного пункту, вулиці та номера будинку здійснювався пошук відповідного запису в таблицях Region, District, City, Street, Building;
- у разі знаходження відповідника компоненту призначався унікальний ідентифікатор (ID), який надалі використовувався як зовнішній ключ у таблиці Address;

- фрагмент коду, що реалізує присвоєння ідентифікаторів блоками адрес, наведено на Рис. 3.6.

```
-------------------------------------------------------------
-- 1. ОНОВЛЕННЯ ОБЛАСТЕЙ (KKEN_2)
-------------------------------------------------------------
UPDATE KDAPTST_
SET KKEN_2 = (
    SELECT TOP 1 rg.id
    FROM UAREG_22 AS rg
    WHERE LOWER(LTRIM(RTRIM(KDAPTST_.naimkea_2))) = LOWER(LTRIM(RTRIM(rg.name_ua)))
)
WHERE EXISTS (
    SELECT 1
    FROM UAREG_22 AS rg
    WHERE LOWER(LTRIM(RTRIM(KDAPTST_.naimkea_2))) = LOWER(LTRIM(RTRIM(rg.name_ua)))
);

-------------------------------------------------------------
-- 2. ОНОВЛЕННЯ РАЙОНІВ (KKEN_3)
-------------------------------------------------------------
UPDATE KDAPTST_
SET KKEN_3 = (
    SELECT TOP 1 ds.id
    FROM UADIS_22 AS ds
    WHERE LOWER(LTRIM(RTRIM(KDAPTST_.naimkea_3))) = LOWER(LTRIM(RTRIM(ds.name_ua)))
      AND (KDAPTST_.KKEN_2 IS NULL OR ds.idrel = TRY_CAST(KDAPTST_.KKEN_2 AS INT))
)
WHERE EXISTS (
    SELECT 1
    FROM UADIS_22 AS ds
    WHERE LOWER(LTRIM(RTRIM(KDAPTST_.naimkea_3))) = LOWER(LTRIM(RTRIM(ds.name_ua)))
      AND (KDAPTST_.KKEN_2 IS NULL OR ds.idrel = TRY_CAST(KDAPTST_.KKEN_2 AS INT))
);

-------------------------------------------------------------
-- 3. ОНОВЛЕННЯ НАСЕЛЕНИХ ПУНКТІВ (KKEN_4)
-------------------------------------------------------------
UPDATE KDAPTST_
SET KKEN_4 = (
    SELECT TOP 1 ct.id
    FROM UACITY_22 AS ct
    WHERE LOWER(LTRIM(RTRIM(KDAPTST_.naimkea_4))) = LOWER(LTRIM(RTRIM(ct.name_ua)))
      AND (KDAPTST_.KKEN_3 IS NULL OR ct.idrel = TRY_CAST(KDAPTST_.KKEN_3 AS INT))
      AND ct.id <> 16056
)
WHERE EXISTS (
    SELECT 1
    FROM UACITY_22 AS ct
    WHERE LOWER(LTRIM(RTRIM(KDAPTST_.naimkea_4))) = LOWER(LTRIM(RTRIM(ct.name_ua)))
      AND (KDAPTST_.KKEN_3 IS NULL OR ct.idrel = TRY_CAST(KDAPTST_.KKEN_3 AS INT))
      AND ct.id <> 16056
);
```

Рис. 3.6 Фрагмент коду присвоєння ідентифікатору блокам адрес

Для випадків, коли конкретний номер будинку був відсутній у довіднику (наприклад, у новозбудованих мікрорайонах або за наявності нетипового формату нумерації), застосовувалася **логіка наслідування**: ідентифікатори добиралися на основі найбільш схожих уже наявних записів за тією ж вулицею

та населеним пунктом. При цьому такі відповідності могли додатково маркуватися як такі, що потребують подальшої перевірки або уточнення.

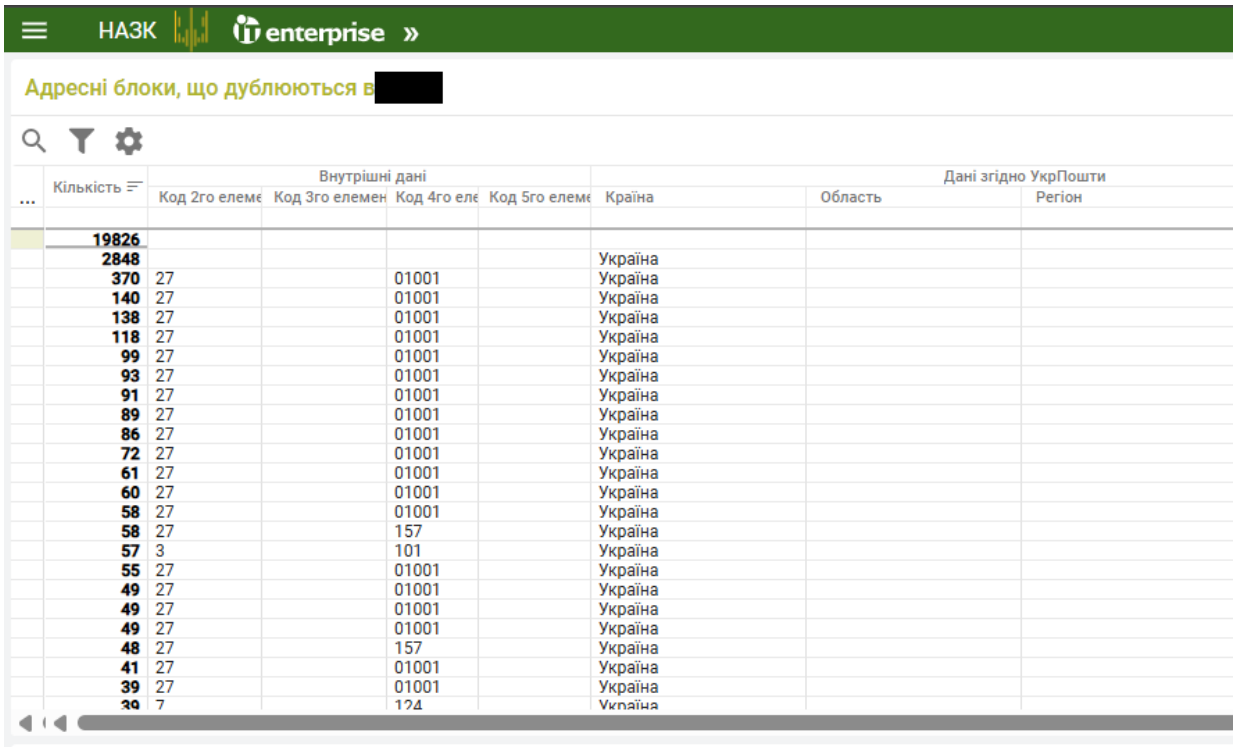
6) Фінальна інтеграція з основним довідником та оцінка результатів

На завершальному етапі обробки нормалізовані та збагачені довідниковими ID дані були інтегровані з основною таблицею адрес ERP-системи за допомогою транзакційних SQL-операцій злиття (MERGE).

Цей етап включав:

- оновлення існуючих записів шляхом заповнення раніше порожніх структурних полів (region_id, city_id, street_id, building_number тощо);
- зміну статусу записів із «ненормалізований» на «нормалізований» для забезпечення коректного відбору у подальшій роботі модуля;
- виявлення та маркування записів, що не пов’язані з жодними клієнтськими чи документними сутностями, як кандидатів на видалення або архівацію.

У результаті виконаної роботи кількість записів із неповним набором блоків адреси було зменшено з 19826 до 7094 (Рис. 3.7 та Рис. 3.8).


НАЗК enterprise »

Адресні блоки, що дублюються в [redacted]

...	Кількість	Внутрішні дані				Дані згідно УкрПошти		
		Код 2го елеме	Код 3го елемен	Код 4го еле	Код 5го елеме	Країна	Область	Регіон
	19826							
	2848					Україна		
	370	27		01001		Україна		
	140	27		01001		Україна		
	138	27		01001		Україна		
	118	27		01001		Україна		
	99	27		01001		Україна		
	93	27		01001		Україна		
	91	27		01001		Україна		
	89	27		01001		Україна		
	86	27		01001		Україна		
	72	27		01001		Україна		
	61	27		01001		Україна		
	60	27		01001		Україна		
	58	27		01001		Україна		
	58	27		157		Україна		
	57	3		101		Україна		
	55	27		01001		Україна		
	49	27		01001		Україна		
	49	27		01001		Україна		
	49	27		01001		Україна		
	48	27		157		Україна		
	41	27		01001		Україна		
	39	27		01001		Україна		
	39	7		124		Україна		

Рис. 3.7 Кількість застарілих записів до нормалізації

Кількість	Внутрішні дані				Дані згідно УкрПошти				
	Код 2го елеме	Код 3го елемен	Код 4го еле	Код 5го елеме	Країна	Область	Регіон	Населений пункт	6-го елеме
7094									
2848					Україна				
370	27		01001		Україна				5
140	27		01001		Україна				3
138	27		01001		Україна				1
118	27		01001		Україна				10
99	27		01001		Україна				4
93	27		01001		Україна				6
91	27		01001		Україна				2
89	27		01001		Україна				8
86	27		01001		Україна				9
72	27		01001		Україна				7
62									1
61	27		01001		Україна				26
60	27		01001		Україна				11
58	27		01001		Україна				14
57	3		101		Україна				1
55	27		01001		Україна				19
49	27		01001		Україна				12
49	27		01001		Україна				28
49	27		01001		Україна				36
41	27		01001		Україна				16
39	27		01001		Україна				15
39	7		124		Україна				164
37									2

Рис 3.8 Кількість застарілих записів після нормалізації

При цьому близько 4 000 записів було визначено як такі, що не використовуються в жодній із пов’язаних таблиць даних контрагентів, і, відповідно, можуть бути безпечно вилучені після завершення формальних процедур.

3.3 Інтерфейси модуля

3.3.1 Інтерфейс виявлення дублікатів

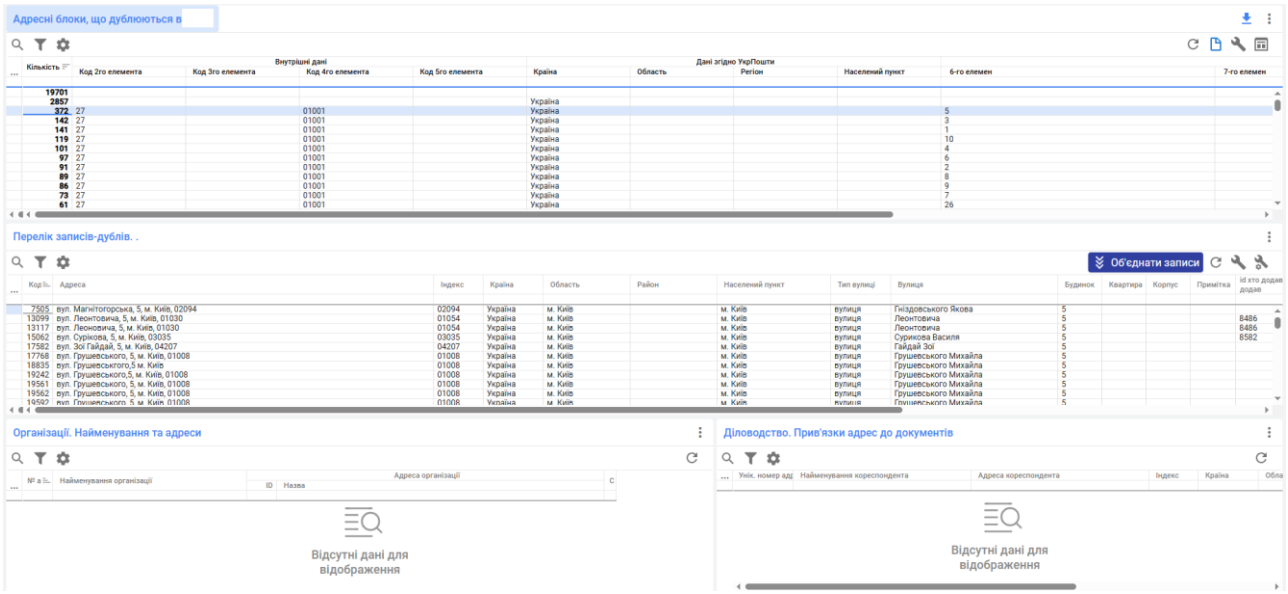


Рис. 3.9 Фрагмент інтерфейсу дублів адрес

Для роботи з виявленими дублікатами було розроблено спеціалізований **адміністративний інтерфейс** Рис. 3.9. Цей інтерфейс призначений для уповноважених користувачів (адміністраторів даних), які відповідають за підтримку якості адресного довідника. Основні функції інтерфейсу - перегляд списку потенційних дублікатів, деталізований порівняльний перегляд дублюючих записів, список документів, де використовуються такі адреси та виконання операції об’єднання або видалення дубліката.

Відображення дублікатів. Після запуску інтерфейсу система обчислює дублі й виводить їх у таблиці: кожен рядок - група дублюючих адрес із ключовими полями та кількістю повторів. Адміністратор може переглянути деталі кожної групи.

Порівняння та вибір основного запису. У деталях відображаються всі дублікати - зазвичай 2-3 записи зі співпадаючими адресними полями. Інтерфейс показує їх поруч для зручного порівняння. Адміністратор оцінює відмінності (повноту, час додавання тощо) й обирає еталон.

Об’єднання (мердж) дублікатів. Після вибору основного запису адміністратор запускає об’єднання. Система оновлює зовнішні посилання на новий ID, видаляє дублікати з таблиці Address і фіксує дію в журналі аудиту. Група дублів оновлюється або зникає.

Інші можливості інтерфейсу. Інтерфейс виявлення дублікатів оснащено фільтрами за регіоном, містом і датою створення, що дозволяє зручно розподіляти завдання між адміністраторами та ефективно пріоритезувати обробку записів.

Інтерфейс реалізовано як окрему форму ERP-модуля, що через методи бекенду отримує групи дублікатів і обробляє дії користувача (об’єднання, видалення), з виведенням повідомлень про результат.

Інтерфейс орієнтований на зручність і безпеку: усі незворотні дії підтверджуються, доступ мають лише адміністратори, а всі зміни фіксуються в аудиті з зазначенням користувача, часу та затронутих записів.

За результатами впровадження, інтерфейс виявлення дублікатів значно спрощує підтримку чистоти даних. Адміністратори більше не мусять вручну виконувати складні SQL-запити або переглядати експортовані списки - система сама сигналізує про проблему та пропонує засоби її вирішення. Це мінімізує людський фактор і прискорює процес очищення даних в десятки разів. У наступному підрозділі наведено результати тестування цього інтерфейсу на реальних даних, що підтверджують його коректність і корисність.

3.4 Результати тестування реалізованих компонентів

Основне тестування зосереджено на модулі виявлення дублікатів, який реалізовано разом з інтерфейсом. Наведено результати перевірки, сценарії, методики, проблеми та показники.

3.4.1 Сценарії та методи тестування

Для перевірки працездатності інтерфейсу виявлення дублікатів було підготовлено декілька **тестових сценаріїв**, що відображають як типові, так і граничні випадки використання:

- **Сценарій 1: Відсутність дублікатів.**

Метою було переконатися, що при спробі додати нову адресу, яка точно дублює існуючу, система або видає повідомлення про дублювання (при ввімкненому унікальному індексі) Рис. 3.10 або, якщо індекс тимчасово вимкнено - цей запис з’являється в інтерфейсі як потенційний дубль. Обидва підвипадки пройшли успішно: при активному індексі користувач отримав

негайне попередження, при вимкненому - запис було додано і модуль відобразив його як дубль до пари існуючого.

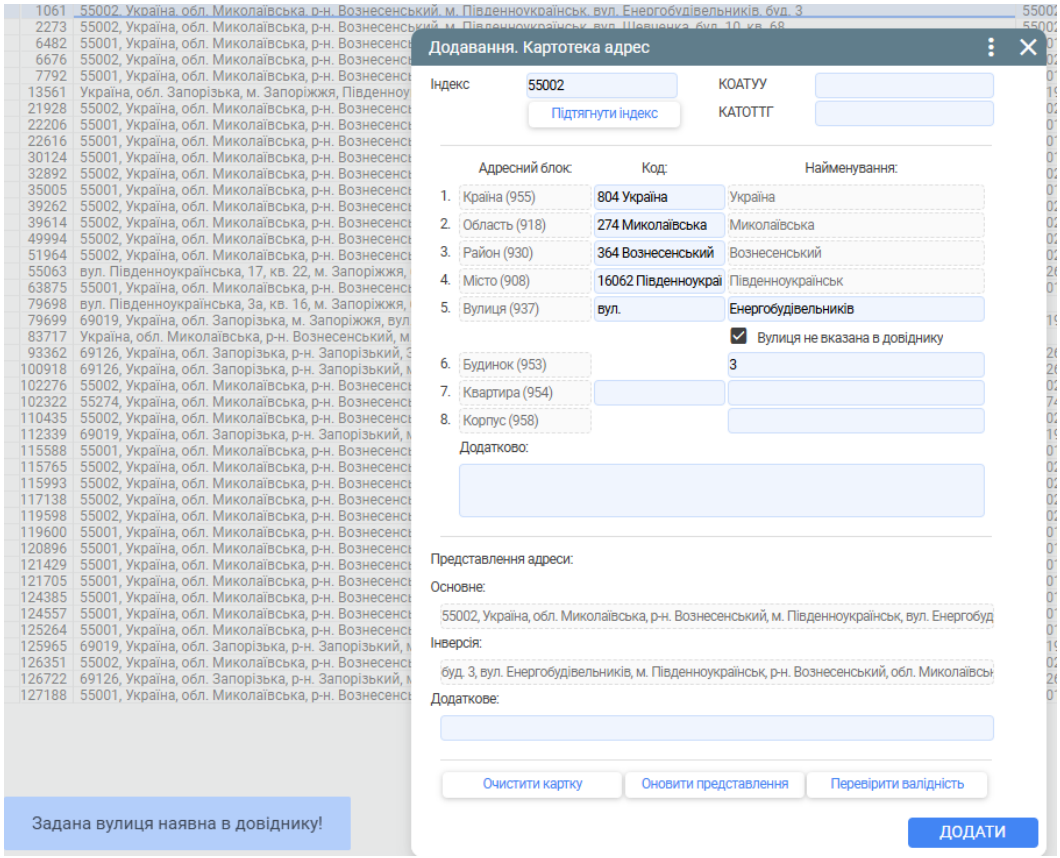


Рис 3.10 Повідомлення про вже існуючу адресу

- **Сценарій 2: Прямий дублікат (повний збіг).**

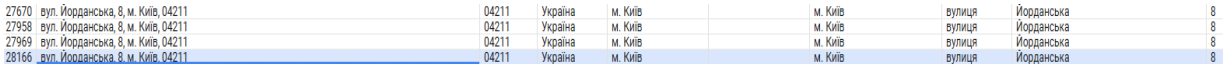


Рис 3.11 Фрагмент додаваня однакових записів

У базу було навмисно внесено 5 ідентичних адрес. Наприклад, додано запис «вул. Йорданська, 8, м. Київ, 04211». Після запуску пошуку модуль знайшов ці дублікати і відобразив їх в інтерфейсі. При переході до деталей видно всі записи, які мають однакові значення всіх полів. Рис 3.11. Адміністратор виконав об’єднання, перед яким висвітилось попередження (Рис. 3.12): обрано перший запис як основний Рис. 3.13. Після підтвердження операції система виконала

видалення дублюючого запису. Повторна перевірка бази даних показала, що залишився лише один запис «вул. Йорданська, 8, м. Київ, 04211» і всі посилання (у даному випадку документів не було, але перевірялися загальні посилання) спрямовані на нього. Сценарій пройшов успішно - дублі усунуто, основний запис збережено.

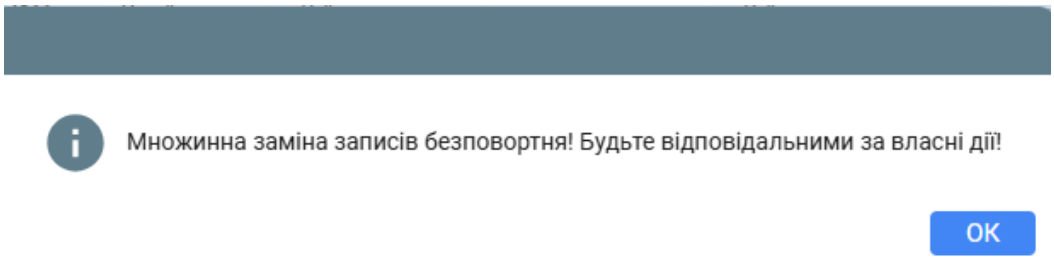


Рис 3.12 Попередження при об’єднанні дублів

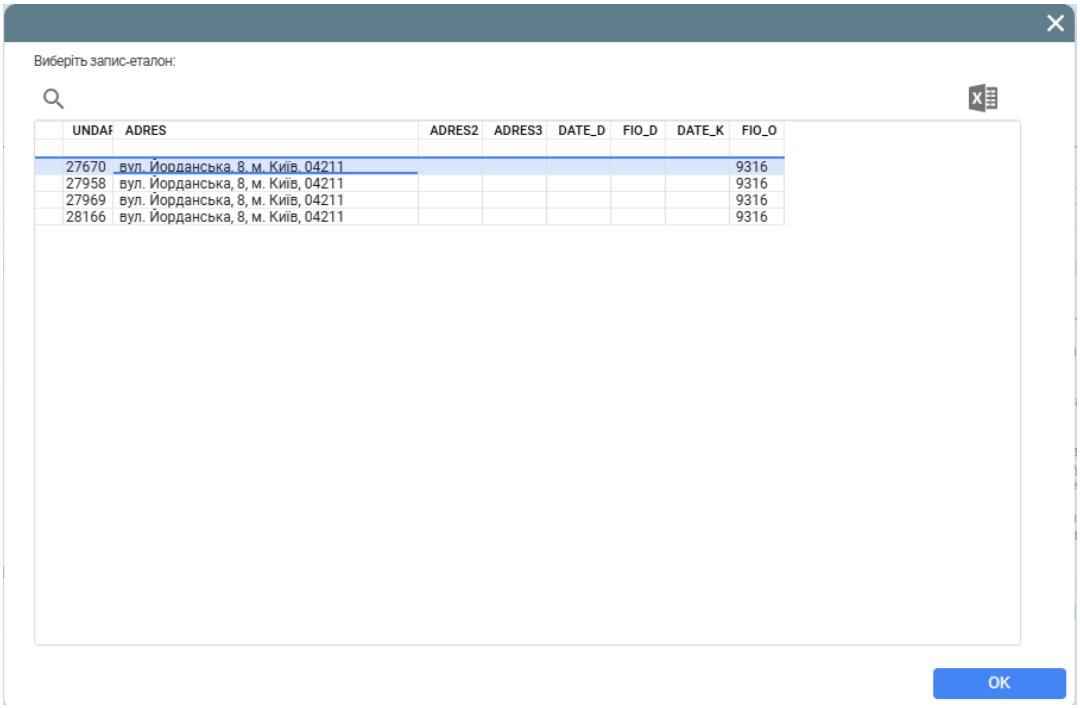


Рис 3.13 Вибір еталоної адреси

- **Сценарій 3: Часткове дублювання / варіації написання.**

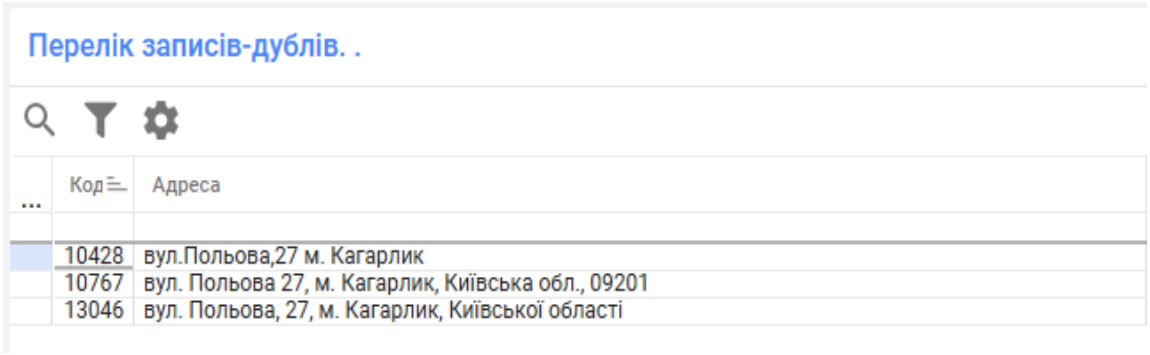


Рис 3.14 приклад дублів адрес з різним написанням

Цей сценарій мав на меті перевірити, як система поводиться, якщо два або більше записів відповідають одній адресі, але були введені трохи по-різному, модуль повинен розпізнати їх як дубль. Досить швидко вдалося знайти такий випадо, зображений на Рис 3.14 де є одразу три різні варіації написання адреси: вул.Польова,27 м. Кагарлик, вул. Польова 27, м. Кагарлик, Київська обл., 09201 та вул. Польова, 27, м. Кагарлик, Київської області. Цей тест продемонстрував, що **система здатна виявляти дублювання навіть при різницях у введенні**.

Для проведення тестування використовувалися такі методи: **функціональне тестування** (перевірка відповідності поведінки вимогам на кожному сценарії), **тестування на граничних випадках** (множинні дублікати, відсутність дублів), та елементів **юзабіліті-тестування** (оцінка зручності інтерфейсу адміністратором). Тестування здійснювалося на копії бази даних, ізольованій від прод середовища, щоб гарантувати безпеку даних.

3.4.1 Виявлені помилки та отримані результати

В результаті тестування було виявлено декілька незначних недоліків у реалізації, які успішно усунуті в процесі налагодження:

- **Відсутність попередження при об’єднанні.** Виявлено, що при натисканні "Об’єднати записи" не було модального діалогу попередження користувача. Це могло призвести до випадкового об’єднання запису, якщо адміністратор помилково натиснув кнопку і кнопку підтвердження. Варіант допрацьовано і вже є повідомлення з попереднім застереженням Рис. 3.12 .
- **Некоректне відображення при великій кількості груп.** При тесті на 100 однакових дублів було помічено, що при окремому виборі кожного дубля список на сторінці трохи перевантажений і прокрутка працює не плавно. Але якщо відсортувати адреси за назвою то можна застовувати множинний вибір комбінацією Shift + ЛКМ.

Щодо самого **алгоритму виявлення дублікатів**, тестування підтвердило його правильність: усі задані дублювання були знайдені, можливі хибні спрацьовування фільтруються вже самим користувачем інтерфейсу, але таких попередньо не знайдено. Цьому сприяє чітке визначення дублю (повний збіг ключових полів), тому помилкових випадків, щоб система помічала різні адреси як одну, не траплялось.

Результати тестування засвідчили успішну роботу модуля. Інтерфейс дублікатів пройшов усі сценарії, отримав схвальні відгуки за зручність, а очищення тестового набору дало 100% видалення повторів. Частка дублікатів знижена до 0%, кількість унікальних адрес відповідає реальним. Очікується подібний ефект при впровадженні на реальних даних НАЗК (близько 20 тис. дублів і 19 тис. ненормалізованих записів). За результатами тестування система була впровадженна в робоче середовище ERP-системи.

РОЗДІЛ 4. АНАЛІЗ АДЕКВАТНОСТІ ТА ЕФЕКТИВНОСТІ ВПРОВАДЖЕНИХ МОДЕЛЕЙ, МЕТОДІВ І ЗАСОБІВ

4.1 Оцінка ефективності впровадження модуля нормалізації адрес

У цьому розділі представлено аналіз того, наскільки ефективним та доцільним виявилося впровадження розробленого модуля нормалізації адрес у внутрішню ERP-систему НАЗК. Аналіз охоплює як якісні, так і кількісні показники, порівнюючи стан адресного довідника та пов’язаних процесів до та після реалізації рішення.

Метою є з’ясувати, чи досягнуто поставлених цілей - покращення якості даних, усунення дублювань, зменшення ручної роботи та підвищення загальної продуктивності системи. Особливу увагу приділено адекватності застосованих методів нормалізації (комбінація використання офіційного довідника Укрпошти та парсингу за допомогою LLM) та їх ефективності в контексті практичного використання в НАЗК.

Як зазначалося раніше, першопочатково адресні дані в ERP НАЗК зберігалися у вигляді одного текстового поля, через що накопичилося багато неуніфікованих (нерегламентованих) записів і дублікатів. Недосконаліть інтерфейсу для введення адрес призвела до появи значної кількості повторів - одні і ті самі адреси могли вводитися користувачами по-різному, що створювало зайве навантаження на персонал та знижувало якість даних. Впроваджений модуль нормалізації адрес був покликаний вирішити ці проблеми шляхом автоматизованого очищення історичних записів, стандартизації їх формату та усунення дублів. Під «адекватністю» рішень мається на увазі відповідність обраних моделей і методів специфіці даних та потребам системи, а під «ефективністю» - досягнення відчутних покращень у показниках якості даних і продуктивності процесів.

Для оцінки ефективності використано низку метрик.

По-перше, аналізується якість адресного довідника до і після впровадження: частка нормалізованих записів, кількість дублікатів, рівень помилкових або неповних адрес.

По-друге, оцінено продуктивність обробки даних: швидкість опрацювання адресної інформації, час, витрачений операторами на ручну перевірку та виправлення адрес, частота втручань адміністратора для підтримки довідника.

Також враховано точність роботи модуля нормалізації - який відсоток адрес він правильно нормалізує автоматично і скільки випадків потребують ручного втручання. Додатково розглянуто зручність роботи користувачів з оновленим інтерфейсом та вплив змін на їхні робочі процеси.

Нарешті, проаналізовано організаційний ефект: як покращення адресних даних відобразилося на суміжних задачах (логістиці документів, аналітиці, обслуговуванні громадян) та чи може набутій досвід бути масштабований на інші системи.

Важливим аспектом оцінки є зіставлення результатів із очікуваннями, сформульованими на етапі проєктування. Передбачалося, що впровадження модуля підвищить якість даних у системі, зменшить обсяг ручної обробки адресної інформації та забезпечить надійність обміну даними між компонентами інформаційної системи.

У цьому розділі перевіряється, наскільки ці очікування підтвердилися на практиці, та робляться висновки щодо доцільності такого рішення для НАЗК. Крім того, розглядається, чи є запропоноване рішення достатньо гнучким і ефективним, щоб бути рекомендованим до використання в інших державних органах, які стикаються зі схожими проблемами якості адресних даних.

4.2 Порівняльні показники якості даних до і після реалізації

Першим кроком аналізу стало порівняння стану адресного довідника до впровадження модуля і після його інтеграції. Для цього були зібрані ключові показники якості даних та ефективності процесів за два періоди: (1) до початку виконання робіт з нормалізації та (2) після завершення очищення історичних записів і запуску нового механізму на поточних даних. Основні результати цього порівняння зведено в таблиці та проілюстровано на графіках.

4.2.1 Якість адресного довідника: структурованість та унікальність записів

До реалізації проєкту адресний довідник НАЗК містив значну кількість неструктурованих записів. За оцінками, близько **20%** усіх адрес були введені у довільному форматі (нерегламентовано), тобто не розкладені по відповідних полях (область, місто, вулиця тощо). Такі записи були результатом історичного накопичення даних в одному текстовому полі і різнотипного вводу інформації користувачами. Відсутність єдиного стандарту призводила до варіативності написання адрес, що утруднювало пошук і обробку даних. Окрім того, у довіднику нараховувалося велика кількість дублюючих записів - одна і та сама адреса могла фігурувати кілька разів з незначними відмінностями у написанні або скороченнях. Наявність таких дублів створювала надмірність даних та потенційну неоднозначність при їх використанні. Наприклад, записи «м. Київ, вул. Академіка Глушкова, 1» і «Київ, Ак. Глушкова 1» до нормалізації сприймалися системою як різні, хоча представляють ту саму реальну адресу.

Після впровадження модуля нормалізації ситуація кардинально покращилася. Всі наявні адреси було автоматично розібрано на стандартизовані атрибути та звірено з офіційним довідником Укрпошти. У результаті частка нерозкладених (ненормалізованих) записів зменшилася з 20% до 7%(більша частина з яких підлягає видаленню або відокремленню від основної таблиці адрес), оскільки кожен запис набув структурованої форми або був

виправлений/об’єднаний з існуючим записом. Що стосується дублікатів, то цей процес потребує тривалого часу так як, щоб не видалити потрібні записи, адміністратор має вручну перевірити ці дані. Для запобігання повторній появі дублів запроваджено механізм унікального індексу на рівні бази даних. Таким чином, кількість нових явних дублів у довіднику знизилася до нуля. Це означає, що кожна реальна нова поштова адреса тепер представлена у системі лише одним, уніфікованим записом. Відповідно, довідник адрес став повністю унікалізованим і узгодженим із офіційними назвами.

Варто відмітити, що досягнення такого рівня якості стало можливим завдяки поєднанню двох підходів: використання авторитетного довідника (адресного класифікатора Укрпошти) та алгоритмічного парсингу для зіставлення неформатованих рядків з офіційними еталонами. Згідно з кращими практиками, валідація та очищення адресних даних через звіряння з авторитетними джерелами є дієвим способом забезпечити їхню достовірність і консистентність. Автоматизовані засоби, такі як API-інтерфейси для перевірки адрес, істотно спрощують цей процес. У нашому випадку інтеграція з API Укрпошти дозволила нормалізувати старі записи за встановленими стандартами: назви областей, населених пунктів, вулиць були приведені у відповідність до офіційних класифікаторів. Такий адресний нормалізатор гармонізував формати і виправив розбіжності, що було критично для подальшої коректної роботи системи та аналізу даних.

Крім усунення дублів, було також виявлено і виправлено інші якісні дефекти даних: відсутні або неповні частини адрес доповнено, помилки в написанні (опечатки) виправлено, застарілі назви актуалізовано. Наприклад, зміни адміністративно-територіального устрою (перейменування населених пунктів, вулиць) тепер відображені у довіднику коректно. До впровадження такі зміни часто не враховувалися, що призводило до появи застарілих адрес у базі. Наразі ж підтримується відповідність із поточними офіційними даними за рахунок регулярного оновлення довідника та контролю змін. Це означає, що

адресний довідник не лише очищено, а й надалі підтримується у актуальному стані, що важливо для довгострокової якості даних.

4.2.2 Продуктивність обробки даних та зниження трудомісткості процесів

Окрім покращення безпосередньо якості даних, впровадження модуля нормалізації адрес значно вплинуло на операційну ефективність процесів, пов’язаних зі введенням та обробкою адресної інформації. У цій підсекції порівнюються показники швидкості та трудозатрат до і після реалізації рішення.

Швидкість обробки нових адрес. Раніше, коли користувачі (оператори) додавали нові адреси до системи, була необхідність ручної перевірки: адміністратор або відповідальна особа мала переглядати введений текст і звіряти його з відомими довідниками, особливо якщо адреса виглядала нетипово. Це уповільнювало процес реєстрації нових документів, оскільки на верифікацію адреси могло витрачатися додатковий час (у середньому до декількох хвилин на запис у складних випадках). Після впровадження модуля, перевірка і стандартизація адрес відбувається в реальному часі автоматично - при введенні адреси система через API шукає відповідний стандартний запис і пропонує його користувачу. У результаті, додавання адреси тепер не потребує пост-фактум перевірки: швидкість введення адресного запису наблизилася до часу, необхідного лише на введення користувачем (кілька секунд). Загалом, середній час обробки адресної інформації при введенні нового документу скоротився, за оцінкою, в 2 рази. Автоматична нормалізація усуває затримки, пов’язані з очікуванням ручної перевірки, що підвищує загальну пропускну здатність процесу реєстрації даних.

Очищення історичних даних. За відсутності автоматизованого рішення, приведення до ладу старих адресних записів вимагало б значних ручних зусиль. Фактично, щоб обробити тисячі нерегламентованих адрес, команді адміністраторів довелося б витратити багато годин на пошук кожної адреси у

довідниках, виправлення помилок та об’єднання дублів. Впроваджений модуль виконав цю роботу значно швидше: основне парсинг-скрипт і запити до API обробили весь масив (~13 тисяч записів) за кілька хвилин в автоматичному режимі. Звичайно, певний час було витрачено на одноразову розробку та налаштування алгоритмів, проте безпосереднє виконання зайняло мінімум часу порівняно з ручною роботою. Таким чином, організація зекономила ресурси та час співробітників, які були перенаправлені на більш пріоритетні задачі, замість рутинного очищення даних.

Зменшення ручних втручань. Показник, тісно пов’язаний з попередніми, - це кількість випадків, коли потрібне втручання людини для виправлення або підтвердження адреси. До впровадження, практично кожна нова адреса, введена в довідник, потенційно могла вимагати уваги адміністратора (особливо якщо це новий населений пункт чи нетиповий формат). За оцінками, близько 15-20% введених записів потребували виправлень або уточнень вручну. Після впровадження модуля цей показник різко знизився. Система сама відловлює підозрілі чи неповні адреси та допомагає користувачу їх коригувати на етапі введення. У результаті частка адрес, що *необхідно виправляти вручну*, впала до декількох відсотків. Більше 95% записів зараз нормалізуються правильно **без втручання людини**. Лише поодинокі випадки (≈5%) потребують ручного перегляду - наприклад, якщо адреса не знайдена в довіднику Укрпошти через помилки у джерелі або якщо є декілька можливих варіантів відповідності. На Рис. 4.1 подано розподіл адрес за способом обробки: автоматично нормалізовані та ті, що вимагали ручної корекції.

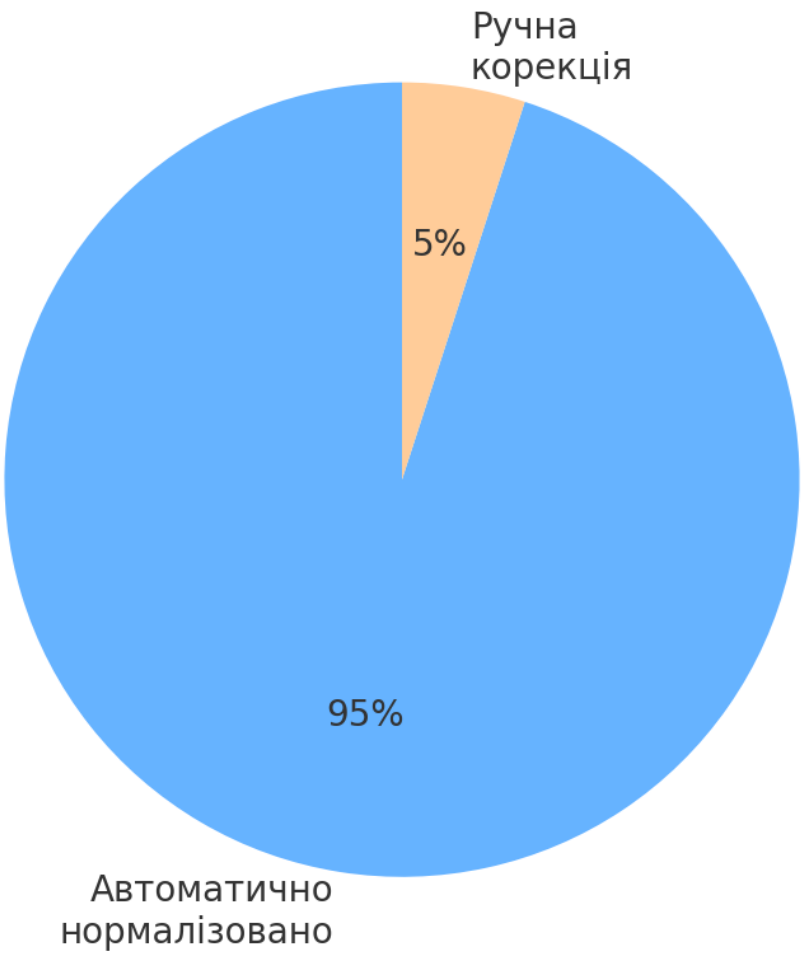


Рис. 4.1. Розподіл результатів нормалізації адрес: частка автоматично оброблених записів і тих, що потребували ручної корекції (у %).

Як видно з Рис. 4.1, переважна більшість адресних записів (близько 95%) були успішно розпізнані і уніфіковані модулем автоматично. Це свідчить про високу точність алгоритмів нормалізації та адекватність використаних джерел даних. Лише 5% випадків перейшли в режим ручної обробки - ці записи були винятковими сценаріями, що не піддалися автоматичній розбивці (наприклад, дуже нестандартний формат або відсутність адреси у довіднику). Таким чином, можна оцінити точність нормалізації приблизно в *95%*. Цей показник є доволі високим для подібних систем і засвідчує коректність обраного підходу. В контексті інформаційних систем показник автоматичної обробки 90-95%

вважається дуже добрим результатом, оскільки повна автоматизація без жодної помилки майже неможлива через різноманітність вхідних даних. До того ж, наявність механізму ручної перевірки для решти 5% означає, що жоден проблемний випадок не залишився поза увагою - користувач або адміністратор міг виправити чи підтвердити його, забезпечивши стовідсоткову чистоту даних після завершення процедури.

Для повноти аналізу варто навести конкретні числові показники до і після впровадження, які підсумовують вищевказані результати:

- **Кількість ненормалізованих (неструктурованих) адресних записів:** до - ~19000 записів (20% від загальної кількості); після - ~7000 записів (більшість з яких не використовуються в системі або не є записами адреси та підлягають видаленню).
- **Середній час перевірки та внесення однієї адреси:** до - ~2 хвилини (включно з можливим ручним виправленням); після - <20 секунд (адреса одразу нормалізується при введенні, більшість вводиться без затримок).
- **Частка адрес, що вимагають ручного виправлення:** до - ~20%; після - ~5%.
- **Обсяг ручної роботи для очищення історичних даних:** до - десятки людино-годин (за оціночними прикидками для обробки всіх записів вручну); після - <1 людино-година (переважно на контроль та перевірку результатів роботи скрипта, автоматизація виконала основну частину роботи).

Наведені цифри чітко демонструють позитивний ефект від реалізації проєкту. Показники якості даних значно покращилися (довідник став повністю нормалізованим і без дублів), а показники продуктивності - зросли (менший час і зусилля на обробку інформації). Зменшення частки ручних втручань означає, що людський фактор у введенні адрес мінімізовано, що своєю чергою скорочує ймовірність помилок та непослідовностей.

Усі ці результати узгоджуються з основною метою - підвищити якість даних та ефективність робочих процесів ERP-системи НАЗК завдяки модернізації адресного довідника.

4.3 Моніторинг впровадження та підтримки процесу

Для довготривалого успіху будь-якого IT-рішення важливо не лише досягти разового покращення, але й забезпечити стабільність та контроль його роботи надалі. У випадку модуля нормалізації адрес було впроваджено низку заходів моніторингу та супроводу, щоб гарантувати його ефективну роботу в режимі реального часу і оперативно реагувати на можливі збої чи відхилення. Цей підрозділ описує, як здійснюється аудит і логування процесу нормалізації, які механізми повторної перевірки запроваджено та як система реагує на нестандартні ситуації.

Логування та аудит.

Модуль нормалізації інтегровано з підсистемою логування ERP-системи. Кожна дія - виклик до API Укрпошти, знайдений результат чи повідомлення про помилку - реєструється у відповідному журналі. Це дозволяє адміністраторам відстежувати роботу модуля у деталях. Зокрема, ведеться статистика: скільки разів API повернув кілька варіантів (тобто адреса вимагала вибору), скільки разів не знайдено відповідності. Такий аудит допомагає виявити потенційні проблемні місця.

Механізми повторної перевірки.

Модуль налаштовано на відстеження актуальності вже нормалізованих адрес. Оскільки адресні дані можуть змінюватися (перейменування вулиць, населених пунктів тощо), передбачено механізм періодичного звіряння всіх записів довідника з оновленим довідником Укрпошти. Наприклад, раз на місяць (або за потребою при виході нової версії класифікатора) запускається фоновий процес, який проходить по всіх адресах у системі і перевіряє, чи немає змін в офіційному довіднику. Якщо виявлено, що якась назва вже не актуальна

(наприклад, вулицю перейменовано), система може помітити таку адресу як застарілу і запропонувати оновити згідно з новим стандартом. Ця функція забезпечує динамічну підтримку якості даних: адресний довідник НАЗК не старіє, а еволюціонує разом із зовнішніми змінами.

Реакція на збої та виняткові ситуації.

В процесі експлуатації можуть виникати ситуації, коли автоматизація дає збій: недоступність API Укрпошти, мережеві проблеми, тощо. Для таких випадків реалізовано декілька рівнів захисту. По-перше, у разі недоступності зовнішнього сервісу, система не блокує роботу користувача - адреса може бути тимчасово збережена з використанням локальних довідників. По-друге, про усі критичні збої негайно **сповіщаються** адміністратори (внутрішнє оповіщення). Це дозволяє швидко втрутитися, якщо, наприклад, змінилося API або сталася непередбачена помилка. Таким чином, навіть у разі збоїв дані не втрачаються і процес роботи не зупиняється.

4.4 Вплив на користувачів: зручність інтерфейсу та зміни в робочих процесах

Важливим критерієм успішності впровадження IT-рішення є його прийняття кінцевими користувачами та вплив на щоденну роботу персоналу. У випадку модуля нормалізації адрес користувачами є, з одного боку, оператори, які вводять дані (реєструють електронні справи, вносять адреси), а з іншого - адміністратори, що підтримують систему. Розглянемо, як змінився досвід цих користувачів після реалізації проекту, наскільки зручним є новий функціонал та як він вплинув на робочі процеси.

Інтерфейс та поведінка користувачів при введенні адрес.

Однією з складових проекту був розроблений дружній інтерфейс для введення адрес, який інтегровано в існуючі форми ERP-системи. Тепер, коли користувач починає вводити адресу, система підказує варіанти (autocomplete) на основі офіційного довідника. Наприклад, достатньо ввести поштовий індекс, як

випадає список можливих варіантів. Користувач може вибрати потрібний, і решта полів автоматично заповняться (область, район тощо). Цей механізм не тільки прискорює введення (менше друкувати), але й запобігає появі помилок - немає ризику опечатки або невірного формату, оскільки вибирається вже перевірений варіант. За рахунок цього користувацький досвід суттєво покращився: операторам більше не потрібно згадувати або шукати правильний формат написання адреси, вони покладаються на підказки системи.

З точки зору поведінки, перед запуском було проведено коротке навчання та демонстрація можливостей підказок і автоматичної нормалізації. Після декількох днів практики більшість операторів позитивно оцінили нововведення. З'ясувалося, що автоматичні підказки не лише економлять час, а й зменшують стрес, пов'язаний з необхідністю бути уважним до кожної дрібниці у форматі адреси. Якщо раніше користувач хвилювався, чи правильно він скорочує назву області або чи актуальна назва вулиці, то тепер ці питання зняті - система все робить за нього. Відтак людський фактор помилок при введенні практично усунуто, а користувачі можуть зосередитися на інших аспектах своєї роботи.

Інтерфейс для роботи з дублями.

Окремо був реалізований режим адміністратора для перегляду та редагування дубльованих записів (про нього йшлося в розділі 3). Після масового очищення дублів основна робота цього інтерфейсу - *реагувати на поодинокі випадки*, якщо раптом з'явиться потенційний дубль. Завдяки унікальним індексам, таких випадків майже не трапляється, але інтерфейс все одно служить корисним інструментом. Адміністратори відзначили, що цей модуль значно спрощує контроль: раніше, щоб знайти дублікати, потрібно було вручну запускати запити до бази, звіряти списки. Тепер же система сама сигналізує про можливий дубль (напр. якщо користувач ввів адресу, яка дуже схожа на існуючу). Адміністратор заходить у спеціальне вікно, де зібрані підозрілі записи, і одним кліком може об'єднати або відхилити дублювання. Таким чином, підтримка цілісності даних стала менш трудомісткою і більш інтерактивною. В

цілому, і оператори, і адміністратори зауважили зменшення рутини у своїй роботі: багато операцій, які раніше вимагали ручної перевірки, тепер автоматизовані або винесені в зручний інтерфейс.

Вплив на робочі процеси. Після впровадження модуля нормалізації було відзначено декілька позитивних змін у щоденній роботі підрозділів, пов'язаних з введенням та використанням адресної інформації:

- Час на оформлення однієї електронної справи (що включає введення адреси) скоротився, як згадувалося, на 10-20%. Це означає більш швидке обслуговування внутрішніх процесів і, відповідно, можливість обробляти більший обсяг роботи за той самий час.
- Знизилося навантаження на відділ контролю електроного документообігу: раніше діловодам надходило багато звернень щодо виправлення адрес у документах та довідниках контрагентів. Тепер таких звернень майже немає, оскільки система сама підказує і виправляє неправильні записи.
- Задоволеність користувачів зросла. Хоча це якісний показник, він проявився у зворотному зв'язку: співробітники відзначають, що новий інструмент полегшує їхню роботу. Менше рутини та повторюваних дій підвищують мотивацію та дозволяють сфокусуватися на важливіших аспектах (наприклад, аналізі змісту документів, а не виправленні некоректних адрес).
- Робочий процес став більш стандартизованим. Якщо раніше різні співробітники могли по-різному вводити адресні дані, що ускладнювало колективну роботу з ними, то тепер усі слідують одному формату, продиктованому системою. Це полегшує взаємозамінність персоналу і розуміння даних: будь-який працівник, відкривши запис, бачить знайому структуровану адресу, а не довільний текст.

Таким чином, впровадження модуля нормалізації адрес мало позитивний вплив на кінцевих користувачів ERP-системи НАЗК. Інтерфейс став зручнішим

і «розумнішим», зменшилась кількість помилок та затримок при роботі з адресами, а робочі процеси стали більш продуктивними та узгодженими. Людський фактор, як джерело помилок, значною мірою нейтралізовано, що вивільнило час і зусилля працівників для більш значущих задач. Це все є важливою складовою загальної ефективності впровадженого рішення.

4.5 Покращення якості довідника адрес та його роль у діяльності організації

У цьому підрозділі розглянемо, яке значення мають досягнуті покращення адресного довідника для ширшого контексту діяльності НАЗК та суміжних процесів. Якісні адресні дані - це не самоціль, вони безпосередньо впливають на інші аспекти роботи організації: логістику документів, аналітичну обробку інформації, взаємодію з громадянами та іншими установами. Проаналізуємо, як саме модернізований довідник адрес сприяє підвищенню точності та ефективності цих напрямів.

Точність логістики та комунікацій.

НАЗК, як державний орган, здійснює значну кількість комунікацій, у тому числі поштових розсилок, надсилання офіційних повідомлень, взаємодіє з регіональними підрозділами, іншими установами та громадянами. В усіх цих випадках *адреса* є критичним реквізитом. Раніше наявність помилок або дублів у адресах могла призводити до збоїв у доставці: листи поверталися через некоректну адресу, відбувалися плутанини коли дві різні справи мали ту саму адресу, тощо. Тепер, з уніфікованим довідником, такі випадки практично виключені. Достовірні адресні дані забезпечують правильність доставки документів і повідомлень з першого разу. За рахунок автоматичного виправлення та перевірки адрес, організація уникає зайвих витрат часу і ресурсів на повторні відправлення чи розслідування, чому кореспонденція не дійшла до адресата. Це особливо важливо в контексті контролю та превенції корупції, де часто потрібне оперативне і точне донесення інформації до відповідальних осіб.

Крім того, внутрішня логістика (переміщення справ, архівів) також виграє від впорядкованих адрес: легше знайти потрібні матеріали, згрупувати їх за регіонами, відстежити рух документів.

Аналітика та прийняття рішень.

Одним із завдань ERP-системи електронних справ є надання даних для аналітичних звітів, статистики, прийняття управлінських рішень. Адресні дані можуть використовуватися для географічного аналізу - наприклад, розподіл певних категорій справ за регіонами, виявлення «проблемних зон» тощо. Коли адреси неуніфіковані, такий аналіз утруднений або дає викривлені результати (через дублювання одні регіони могли рахуватися двічі, або частина адрес випадала з аналізу через неправильний формат).

Після нормалізації даних аналітична обробка стала значно точнішою. Тепер можна з більшою впевненістю агрегувати дані по областях, районах чи містах, не переживаючи, що частина записів «не потрапить» у вибірку. Це підвищує достовірність звітності та аналітики, що формуються на основі ERP-системи.

Керівництво НАЗК отримало змогу приймати рішення, спираючись на *якісні дані*, а не на приблизні оцінки.

Зросла і взаємодія з зовнішніми аналітичними системами: оскільки адреси тепер мають стандартизовані коди (наприклад, коди КОАТУУ/КОАТТП або інші ідентифікатори з довідника Укрпошти), їх можна без проблем зіставляти з даними інших державних реєстрів. Це відкриває можливості для збагачення даних та більш комплексного аналізу. Як вказують фахівці, повнота і узгодженість адресних даних є передумовою ефективного прийняття рішень та успішної інтеграції інформаційних систем.

Обслуговування громадян та користувачів.

Хоч ERP-система електронних справ НАЗК і є внутрішньою, її результати відображаються назовні - у взаємодії агентства з громадянами та іншими органами. Якісний адресний довідник сприяє кращому обслуговуванню запитів громадян. Наприклад, якщо громадянин подає звернення чи декларацію і вказує адресу, тепер ця адреса одразу нормалізується і надалі використовується послідовно. Це означає, що якщо громадянин потім звернеться для уточнення чи в пошуку інформації, оператор легко знайде всі пов’язані записи за уніфікованою адресою (а не буде шукати серед декількох варіантів написання). Також, у випадку розсилки повідомлень про результати перевірок чи інших офіційних листів громадянам, ймовірність помилки в адресі мінімізована, що підвищує рівень довіри до роботи агентства (не буде ситуацій, коли лист не дійшов через банальну помилку у адресі).

Крім того, інші відділи чи суміжні організації, які отримують інформацію від НАЗК, також виграють від уніфікованих адрес. Наприклад, якщо дані з ERP НАЗК передаються до державного реєстру чи статистичного відомства, стандартні адресні поля дозволяють автоматично імпортувати інформацію без попереднього трудомісткого очищення. Фактично, НАЗК, впорядкувавши власний адресний довідник, стає більш надійним партнером у міжвідомчому обміні даними. Це узгоджується з загальною тенденцією до підвищення якості даних у державному секторі: чим вища якість базових даних, тим ефективніше працюють електронні послуги, реєстри і система державного управління в цілому.

Підтримка цілісності інформаційної системи.

Одним із важливих практичних результатів проекту стало забезпечення більшої цілісності даних у ERP-системі. Раніше, через розрізненість та неоднорідність адрес, могли виникати помилки при інтеграції модулів системи: наприклад, модуль відправки електронних листів не міг коректно обробити

адресу. Тепер усі компоненти ERP-системи працюють з одним узгодженим довідником адрес, що гарантує надійність обміну даними між компонентами. Відсутність дублювання усуває можливість розходжень: якщо в різних місцях системи посилалися на одну адресу, тепер це одна й та сама сутність, а не дві різні строкові записи. Це спрощує підтримку системи, резервне копіювання, міграцію даних. Інформаційна взаємодія з іншими державними системами також стає надійнішою - стандартизовані адреси легше транслювати між базами даних, виключаються помилки відповідності. Таким чином, модуль нормалізації адрес зробив внесок не лише у власне довідник, а й у покращення якості всієї інформаційної системи.

Отже, зміни у довіднику адрес, досягнуті завдяки реалізації проекту, мали позитивний системний ефект на діяльність НАЗК. Підвищилася точність логістичних і комунікаційних процесів, покращилася аналітика та звітність, спростилася взаємодія з громадянами та органами влади, зміцнилася інтеграція інформаційних систем. Усе це підтверджує, що інвестиція в якість даних окуповується багатократно - чисті й стандартизовані адресні дані стали надійним фундаментом для багатьох функцій ERP-системи та роботи агентства в цілому.

У четвертому розділі було проведено всебічний аналіз впровадженого модуля нормалізації адрес та оцінено результати його роботи в ERP-системі електронних справ НАЗК. Аналіз підтвердив, що поставлені цілі досягнуті повною мірою. Рішення виявилося адекватним - обрані методи (API-валізація та алгоритмічний парсинг) добре підійшли до характеру даних і проблеми, а також ефективним - показники якості даних та продуктивності процесів суттєво покращилися після його реалізації.

ВИСНОВОК

У ході виконання кваліфікаційної роботи було досягнуто поставленої мети - розроблено проєкт модуля нормалізації адрес для ERP-системи електронних справ НАЗК, спрямований на підвищення якості, цілісності та унікальності адресних довідників, а також на зниження ручного навантаження на персонал і ризиків, пов’язаних з використанням некоректних адресних даних.

Мета дослідження реалізована через виконання комплексу взаємопов’язаних завдань, визначених у вступі, а саме:

1. **Виконано аналіз існуючих текстових записів і типових помилкових форматів.**

   Було здійснено системний аналіз історично накопичених текстових адресних записів у ERP-системі НАЗК. На основі вибірок ненормованих адрес виокремлено типові сценарії помилок: варіативність форматів запису (різні варіанти однієї й тієї ж адреси), неуніфіковані скорочення, відсутність або перестановка обов’язкових компонентів, орфографічні помилки, змішування службової інформації (ПІБ, контакти) з адресою тощо. Проведений аналіз дав змогу формалізувати основні класи проблем і визначити вимоги до алгоритмів парсингу та нормалізації.

2. **Виконанно проєктування архітектури модуля (UI, сервіс парсингу, модуль індексування).**

Спроєктовано багатошарову архітектуру модуля нормалізації адрес, що включає:

- користувацький інтерфейс з окремими вкладками для роботи з ненормованими записами, дублями та пов’язаними документами;
- сервіс парсингу та нормалізації, відповідальний за трансформацію «сирих» текстових адрес у структурований вигляд;

- індексування та виявлення дублікатів, що працює на основі нормалізованої структури БД та складених унікальних індексів;
- інтеграційний шар з ERP-платформою та зовнішніми сервісами.

Така архітектура забезпечує модульність, розширюваність і можливість незалежної еволюції окремих компонентів.

3. **Виконано дослідження специфікації API-інтеграції з сервісом Укрпошти.**

Проведено детальне вивчення адресного класифікатора API Укрпошти як еталонного джерела достовірних адресних даних. Проаналізовано структуру REST-ендпоїнтів, формати запитів і відповідей, ієрархію «область - район - населений пункт - вулиця - будинок», систему унікальних ідентифікаторів. На основі цього сформовано вимоги до інтеграційного шару модуля: механізми отримання, кешування та актуалізації довідників, а також правила зіставлення результатів парсингу з відповідними записами Укрпошти.

4. **Виконано дослідження алгоритмів розбору (парсингу) нерегламентованих адрес.**

Було проаналізовано можливі підходи до розбору нерегламентованих адрес: класичні rule-based методи на основі регулярних виразів і словників, статистичні та ML-підходи, а також використання великих мовних моделей. За результатами аналізу обґрунтовано вибір гібридного підходу: поєднання попереднього структурного розбору, довідникових запитів до API Укрпошти та застосування LLM ChatGPT 5 для нормалізації legacy-записів, що мають особливо складні або нестандартні формати. Такий підхід дозволяє досягти прийнятної якості розпізнавання при обмежених ресурсах часу та навчальних даних, зберігаючи можливість подальшого розширення алгоритмічної частини.

5. **Виконано проєктування структури бази даних із унікальними індексами.**

На основі вимог до нормалізованого зберігання адрес спроєктовано структуру бази даних модуля: розділено атрибути адрес на окремі довідники (області, райони, населені пункти, вулиці тощо), запроваджено зв’язки «багато-до-одного» між записами адрес і відповідними елементами довідників. У таблиці Address запропоновано складений унікальний індекс за ключовими полями (region_id, district_id, city_id, street_id, building_number, apartment_number), що унеможливлює створення нових дублікатів у нормалізованій моделі даних. Також описано допоміжні індекси для підвищення продуктивності пошуку та перевірки унікальності.

6. **Створенно інтерфейс адміністратора для роботи з дублями.**

Розроблено концепцію адміністративного інтерфейсу, який забезпечує повний цикл роботи з підозрюваними дубльованими записами. Інтерфейс дозволяє:

- переглядати згруповані дублікати, сформовані на основі нормалізованих атрибутів;
- порівнювати деталі записів (у тому числі пов’язані документи та об’єкти в ERP);
- обирати «еталонний» запис, об’єднувати або позначати дублікати для видалення;
- реєструвати всі дії в журналі змін для подальшого аудиту.

Такий інтерфейс реалізує принцип «людина в контурі» (human-in-the-loop), поєднуючи автоматичне виявлення дублів з контрольованим, керованим прийняттям рішень.

7. **Створенно механізм нормалізації нерегламентованих записів адрес.** Описано та обґрунтовано поетапний механізм нормалізації застарілих («legacy») текстових адрес:

    - первинна фільтрація ненормованих записів за допомогою SQL-запитів із відбором рядків, у яких відсутні структурні атрибути;
    - експорт і аналітична обробка в середовищі електронних таблиць із виділенням типових патернів запису;
    - масова токенізація й семантичний розбір адрес за допомогою LLM ChatGPT 5 з отриманням проміжних структурованих представлень;
    - ручна валідація і коригування результатів, видалення нерелевантних записів;
    - зіставлення отриманих компонентів із довідниками Укрпошти, присвоєння їм унікальних ідентифікаторів, формування нормалізованих записів для завантаження в ERP.

    Запропонована процедура поєднує автоматизовану обробку з експертною перевіркою і може розглядатися як прототип технологічного процесу виправлення історичних даних у державних реєстрах.

8. **Виконанно тестування модуля на реальних даних і оцінки його ефективності.**

Проведено тестування розроблених алгоритмів і процедур на реальному фрагменті адресного довідника ERP-системи. Продемонстровано можливість суттєвого скорочення обсягу некоректних та неповних записів: кількість записів із неповним набором блоків адреси зменшено з 19826 до 7094, при цьому частина записів ідентифікована як нерелевантна й підлягає видаленню. За результатами тестування підтверджено, що запропонована архітектура, структура БД та механізми нормалізації забезпечують значне підвищення структурованості,

узгодженості та унікальності адресних даних, а також створюють основу для подальших автоматизованих аналітичних та логістичних процесів у НАЗК.

Узагальнюючи, можна зробити висновок, що всі поставлені завдання кваліфікаційної роботи виконано в повному обсязі. Сформовано цілісну концепцію модуля нормалізації адрес для ERP-системи НАЗК, обґрунтовано вибір алгоритмічних, інтеграційних і архітектурних рішень, спроєктовано структуру бази даних та адміністративні інтерфейси, розроблено механізм масової нормалізації історичних записів і продемонстровано його ефективність на реальних даних.

Отримані результати мають як наукову складову (методика нормалізації та унікалізації адресних даних у державних інформаційних системах), так і виразну практичну значущість для підвищення якості даних та цифрової стійкості процесів у НАЗК.

Перспективами подальших досліджень є розширення використання методів машинного навчання для повністю автоматизованого парсингу українських адрес, інтеграція з Єдиним державним реєстром адрес та створення єдиної адресної платформи для міжвідомчої взаємодії.

ПЕРЕЛІК ВИКОРИСТАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ:

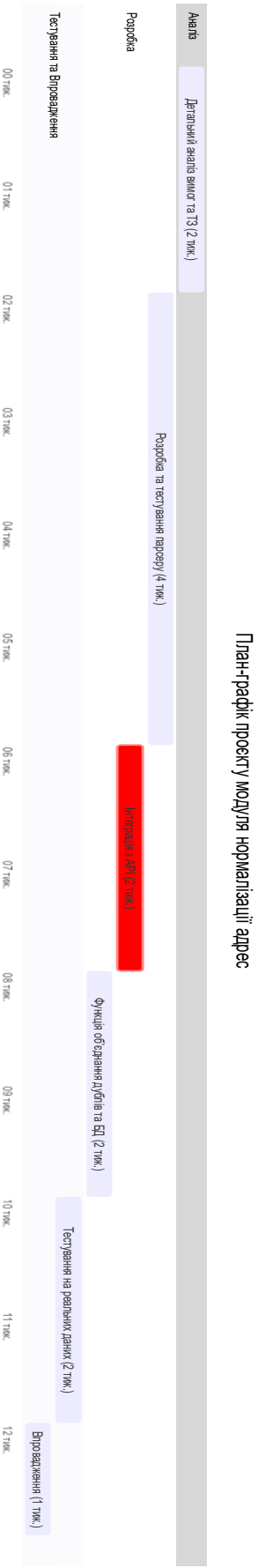
1. Реєстри. НАЗК. URL: https://nazk.gov.ua (дата звернення: 05.06.2025).
2. Address Standardization (Normalization). SmartyStreets Blog. URL: https://smarty.com (дата звернення: 05.06.2025).
3. USPS Postal Addressing Standards. Publication 28. U.S. Postal Service, 2021. URL: https://smarty.com (дата звернення: 05.06.2025).
4. Переваги інтеграції з API. Адресний класифікатор. Укрпошта. URL: https://dev.ukrposhta.ua (дата звернення: 05.06.2025).
5. Закон України. Про публічні електронні реєстри : Закон України від 06.07.2021 № 1689-IX. URL: https://e-construction.gov.ua (дата звернення: 05.06.2025).
6. What Is Libpostal? How It Works & Why It Matters. Senzing. URL: https://senzing.com (дата звернення: 05.06.2025).
7. Multinational Address Parsing: A Zero-Shot Evaluation. arXiv:2112.04008, 2021. URL: https://arxiv.org (дата звернення: 05.06.2025).
8. Деякі питання створення базових державних реєстрів : Постанова КМУ від 07.04.2021 № 254. URL: https://e-construction.gov.ua (дата звернення: 05.06.2025).
9. Документація API адресного класифікатора. Укрпошта. URL: https://dev.ukrposhta.ua (дата звернення: 05.06.2025).
10. Про запобігання корупції : Закон України від 14.10.2014 № 1700-VII. URL: https://nazk.gov.ua (дата звернення: 05.06.2025).
11. ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання. Вид. офіц. Київ : УкрНДНЦ, 2016. 16 с.
12. Тарасюк Г.М. Управління проєктами. Київ : Каравела, 2006. 319 с.
13. Васильків Н.М., Моружко А.В. Вплив середовища на виконання ІТ-проєкту. Сучасні інформаційні технології, 2019. С. 8-12.

14. Моружко А.В. Внутрішнє середовище IT-проєкту : тези доп. конф. «Розвиток науки в XXI столітті», Харків, 2019. С. 46-49.
15. Ukrposhta: інтеграція сервісу доставки для e-commerce. Brander.ua. URL: https://brander.ua (дата звернення: 05.06.2025).
16. Адресний реєстр: розпочато верифікацію адрес. Міністерство цифрової трансформації. URL: https://e-construction.gov.ua (дата звернення: 05.06.2025).
17. Нормалізація схем баз даних. Посібники ВНТУ. URL: https://web.posibnyky.vntu.edu.ua (дата звернення: 05.06.2025).
18. Data Duplication Implications and Solutions. Oracle. URL: https://oracle.com (дата звернення: 05.06.2025).
19. Find duplicate addresses in database. Stack Overflow. URL: https://stackoverflow.com (дата звернення: 05.06.2025).
20. Ballesteros, J. R. *ChatGPT for parsing addresses and geocoding*. - Medium, 1 лютого 2023. - Електронний ресурс. - Режим доступу: https://medium.com
21. Зюзюн В., Кубявка Л. Застосування сучасних інформаційних технологій в управлінні проєктами [Електронний ресурс] : навч. посіб. Київ, 2025. 212 с.
22. Кубявка, Л. Б., & Латишева, Т. В. (2024). Концепція побудови і принципи управління проєктного та продуктового IT-менеджменту. *Управління розвитком складних систем*, (57), 45-50. https://doi.org/10.32347/2412-9933.2024.57.45-50
23. I. Teslia, I. Khlevna, O. Yehorchenkov, N. Yehorchenkova, O. Grigor, Y. Kataieva, T. Latysheva, T. Prokopenko, Y. Tryus, and A. Khlevnyi, "Development of the concept of building project management systems in the context of digital transformation of project-oriented companies," *Eastern-Eur. J. Enterprise Technol.*, vol. 6, no. 3 (120), pp. 14-25, Dec. 2022.
24. Application of Project Management Techniques for Timeline and Budgeting Estimates of Startups. // Електронний ресурс. Режим доступу: https://www.mdpi.com/2071-1050/15/21/15526

25. Morozov V., Kalnichenko O., Timinsky A., Liubyma I. Projects change management in based on the projects configuration management for developing complex projects // 2017 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS). - Bucharest, 2017. - Vol. 2. - P. 939-942.
26. Тімінський О. Г., Оберемок І., Оберемок Н. Development of methodology of efficiency estimation of management technology of project-oriented organizations // Technology Audit and Production Reserves. - 2017. - Vol. 2, No. 2 (34). - P. 24-29.
27. Бушуєв С. Д., Морозов В. В. Динамічне лідерство в управлінні проектами : монографія. - К. : Українська асоціація управління проектами, 1999. - 312 с.
28. Морозов В. В. Формування, управління та розвиток команди проекту : навч. посіб. / уклад. : В. В. Морозов, А. М. Чередниченко, Т. І. Шпильова. - К. : Таксон, 2009. - 464 с.
29. Бушуєв С. Д., Бушуєв Д. А., Бушуєва Н. С. Управління проєктами розробки програмного забезпечення на основі гнучких методологій Agile та Scrum. *Сучасний стан наукових досліджень та технологій в промисловості*. 2019. № 1 (7). С. 15-24.
30. Бушуєва Н. С. Методологія управління проєктами інноваційного розвитку в умовах високої невизначеності. *Вісник ЧДТУ*. 2020. С. 45-52.

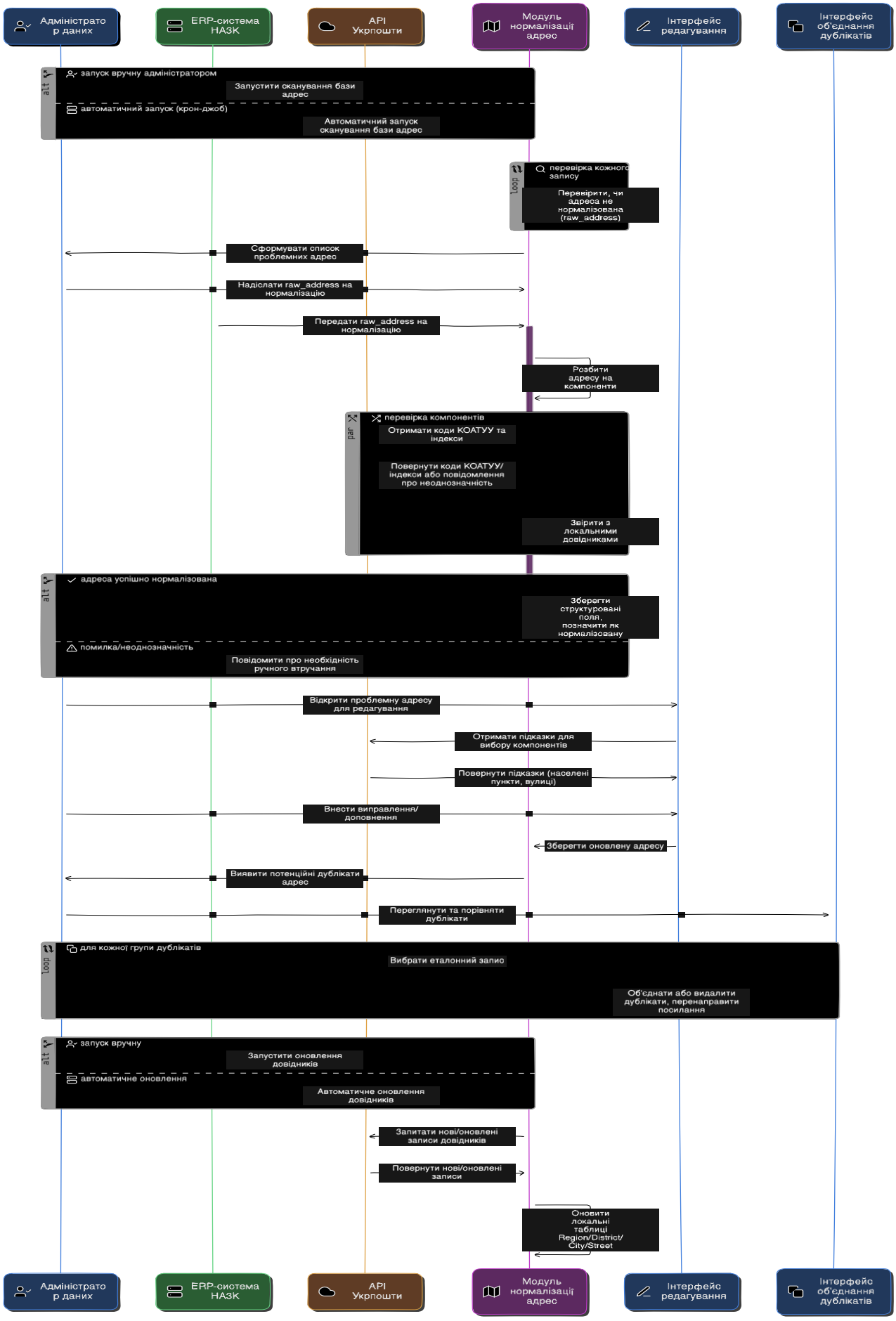
ДОДАТОК А

Діаграма Ганта



ДОДАТОК Б

Use Case діаграма



ДОДАТОК В

Еталонний приклад нормалізації адрес

NAIMKEA_2	KKEA_3	KKEN_3	NAIMKEA_3	KKEA_4	KKEN_4	NAIMKEA_4	KKEA_5	KKEN_5	NAIMKEA_5	KKEA_6	KKEN_6	NAIMKEA_6	KKEA_7	KKEN_7	NAIMKEA_7	ADRES
Київ					Київ			Ярославська			20				12	Ярославська,20, кв.12, м. Київ, 04071
					Мізяч			Яворницька Тараса								Яворницька Тараса, 12, смт. Мізяч, 35740
					Южноукраїнські			Виноградна			29					Южноукраїнськ, вул. Виноградна, 29
Полтавська			Миргородський		Гоголеве											Полтавська обл., Миргородський р-н, селище міського типу Гоголеве, вул.Горова, будинок 27
Черкаська			Шполянський													Шполянський р-н, Черкаська обл
					Антонівка			В'ячеслава Чорновола			266					Шосе В'ячеслава Чорновола, 266, смт. Антонівка, м. Херсон, 73000
Херсонська					Антонівка			В'ячеслава Чорновола								шосе В'ячеслава Чорновола, буд. 266, смт.Антонівка, м. Херсон, Херсонська область, 73000
Київ					Київ			Котельникова			37-А				43	Шосе Дарниця Сергіївна вул. Котельникова, 37-А, кв. 43, м. Київ, 03115
Київська			Рокитнянський		Бирюки			Шкільна			17					Шкільна, 17, с. Бирюки, Рокитнянський р-н, Київська обл., 09630
Закарпатська			Хустський		Вишково			Шанк			57					Шанк №57, смт Вишково, Хустський район, Закарпатська область, 90454
Полтавська			Решетилівський		Любачі			Київське			11					ш. Київське, 11, Любачі, Решетилівський район, Полтавська область, 38420
Полтавська					Кременчук			Чумацький			7					Чумацький шлях, 7, м. Кременчук, Полтавська область, 39621
					Теплик			Чорнобиль			4ж					Чорнобиль, 4ж, смт. Теплик, 23800
Черкаська			Черкаський		Новий Білоус			СВИРИДОВСЬКОГО								Черкаська обл., Черкаський р-н, село Новий Білоус, ВУЛИЦЯ СВИРИДОВСЬКОГО, будинок 54
Черкаська			Черкаський		Олешівка			ЧЕРКАСЬКА								Черкаська обл., Черкаський р-н, селище міського типу Олешівка, ВУЛИЦЯ ЧЕРКАСЬКА, будинок 3
Черкаська			Черкаський					СОБОРНОСТІ								Черкаська обл., Черкаський р-н, селище міського типу Козелець, вул. Соборності, будинок 27
Черкаська			Сновський		Сновськ			СВОБОДИ								Черкаська обл., Сновський р-н, місто Сновськ, ВУЛИЦЯ СВОБОДИ, будинок 141
Чернігівська			Ріпкинський		Розсудів			ЛІСНА			5					Чернігівська обл., Ріпкинський р-н, село Розсудів, ВУЛИЦЯ ЛІСНА, будинок 5
Чернігівська					Чернігів			ШЕВЧЕНКА			51а					Чернігівська обл., місто Чернігів, ВУЛИЦЯ ШЕВЧЕНКА, будинок 51А
Чернігівська					Чернігів			ШЕВЧЕНКА			42					Чернігівська обл., місто Чернігів, ВУЛИЦЯ ШЕВЧЕНКА, будинок 42
Чернігівська					Чернігів			ШЕВЧЕНКА			34					Чернігівська обл., місто Чернігів, ВУЛИЦЯ ШЕВЧЕНКА, будинок 34
Чернігівська					Чернігів			МАЛИНОВСЬКОГО			38					Чернігівська обл., місто Чернігів, ВУЛИЦЯ МАЛИНОВСЬКОГО, будинок 38
Чернігівська					Чернігів			МАГІСТРАТСЬКА			7					Чернігівська обл., місто Чернігів, ВУЛИЦЯ МАГІСТРАТСЬКА, будинок 7
Чернігівська					Чернігів			КОЦЮБИНСЬКОГО			70					Чернігівська обл., місто Чернігів, ВУЛИЦЯ КОЦЮБИНСЬКОГО, будинок 70
Черкаська					Умань			Крамаренка			16					Черкаська обл., місто Умань, вул.Крамаренка, будинок 16
Черкаська					Золотоноша			ШЕВЧЕНКА			105а					Черкаська обл., місто Золотоноша, ВУЛИЦЯ ШЕВЧЕНКА, будинок 105А
Черкаська			Канівський		Баландине			ШЕВЧЕНКА			21					Черкаська обл., Канівський р-н, село Баландине, ВУЛИЦЯ ШЕВЧЕНКА, будинок 21
Черкаська			Канівський		Камянка			НЕКРАСОВА			10-а					Черкаська обл., Канівський р-н, місто Камянка, ВУЛ.НЕКРАСОВА, будинок 10-А
Черкаська			Звенигородський		Звенигород			ШЕВЧЕНКА			68					Черкаська обл., Звенигородський р-н, місто Звенигорода, ПРОСПЕКТ ШЕВЧЕНКА, будинок 68
Черкаська			Городищенський		Старосілля			ГРУШЕВСЬКОГО			2					Черкаська обл., Городищенський р-н, село Старосілля, ВУЛИЦЯ ГРУШЕВСЬКОГО, будинок 2
					Новомосково			Червоний кут			1					Червоний кут, 1, Новомосковськ, 51206
								Шоковського			8				5	Шоковського,8, кв. 5
					Чернівці			Центральна			1					Центральна площа, 1, м Чернівці, 58002
Волинська			Любешівський		Ветли			Стромище			2					хутір Стромище, 2, с. Ветли, Любешівський район, Волинська область, 44200
Волинська			Любешівський		Ветли			Стромище			2					хутір Стромище, 2 с. Ветли, Любешівський район, Волинська область, 44200
Рівненська			Дубенський		Варковичі			Бинок			3					хутір Бинок, 3, с. Варковичі, Дубенський район, Рівненська область, 35612
Київ					Київ			Хрещатик			2					Хрещатик, 2, м. Київ
					Карпівка			Холодноярська			4					Холодноярська, 4, м. Карпівка, 39500
Тернопільська			Кременецький		Ходаки											Ходаки, Шумський, Тернопільська, 47114
								Хмельницький			204				39к	Хмельницький, вул. Подільська, 39к, 204, 290013
Хмельницька			Шепетівський		Улашківці			Перемоги			4					Хмельницька обл., Шепетівський р-н, село Улашківці, вул.Перемоги, будинок 4
Хмельницька			Шепетівський		Плесна											Хмельницька обл., Шепетівський р-н, село Плесна
Хмельницька			Шепетівський		Полонне			Декабристів			50					Хмельницька обл., Шепетівський р-н, місто Полонне, вул.Декабристів, будинок 50
Хмельницька			Хмельницький		Олешин											Хмельницька обл., Хмельницький р-н, село Олешин
Хмельницька			Хмельницький		Лісові Гринівці			ЦЕНТРАЛЬНА								Хмельницька обл., Хмельницький р-н, село Лісові Гринівці, ВУЛИЦЯ ЦЕНТРАЛЬНА, будинок 4