Міністерство освіти і науки України
Харківський національний університет імені В.Н. Каразіна
Факультет комп'ютерних наук

Рекомендовано до захисту:
протокол засідання кафедри
№ ___ від ____.____.2025 р.
Завідувач кафедри _____
___Олешко О. І___ _____
(підпис) (прізвище та ініціали)

Пояснювальна записка

до кваліфікаційної роботи бакалавра
на тему Розробка алгоритмів і програмного забезпечення для аналізу кримінальних подій з використанням концепції та технології OSINT

Захищено на засіданні ЕК № _____
протокол № ____ від ____.____.2025 р.
Оцінка _____ / _____
Голова ЕК
_____ _____
(підпис) (прізвище та ініціали)

Виконав:
студент 4 курсу, групи КС- 41
Спеціальності:
122 Комп'ютерні науки
(шифр і назва спеціальності підготовки)
Бублик Владислав Романович
(прізвище, ім'я та по батькові, підпис)
Керівник PhD, старший викладач
Матвєєв О.М.
(ступень, звання, прізвище та ініціали, підпис)
Рецензент к.ф.-м.н., доцент
Севидов С.М
(ступень, звання, прізвище та ініціали, підпис)

Харків – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Харківський національний університет імені В.Н.Каразіна

Навчально-науковий інститут комп'ютерних наук та штучного інтелекту
Кафедра інтелектуальних програмних систем і технологій
Рівень вищої освіти (освітньо-кваліфікаційний рівень) бакалавр
Галузь знань: 12 Інформаційні технології
Спеціальність 122 Комп'ютерні науки
Освітня програма: Комп'ютерні науки

ЗАТВЕРДЖУЮ

Завідувач кафедри
Олег ОЛЕШКО

_____ _____
підпис ініціали, прізвище

"____" _____202__року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ (ПРОЄКТ)

Бублика Владислава Романовича

(прізвище, ім'я, по батькові студента)

1. Розробка алгоритмів і програмного забезпечення для аналізу кримінальних подій з використанням концепції та технології OSINT

керівник роботи Матвєєв Олександр Миколайович, старший викладач ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від "16" червня 2025 року №4101-5/962

2. Строк подання студентом роботи 2 червня 2025 року

3. Перелік питань, які потрібно розробити)

Ознайомитися з особливостями програмного забезпечення, що використовується для збору та обробки відкритої інформації (OSINT) у контексті кримінальних подій; провести огляд існуючих методів та інструментів реалізації систем OSINT-моніторингу; проаналізувати алгоритми обробки текстових даних, зокрема методи фільтрації, класифікації та виявлення кримінальних інцидентів на основі відкритих джерел; ознайомитися з принципами побудови систем з інтеграцією штучного інтелекту для аналізу подій; розробити власні алгоритми збору та підготовки OSINT-даних; здійснити програмну реалізацію прототипу.

4. План роботи

№ з/п	Назви етапів роботи
1.	Формулювання мети дослідження та постановка завдань для реалізації гібридної OSINT-системи з виявлення та аналізу кримінальних подій
2.	Аналіз предметної області кримінальних подій як об’єкту OSINT-дослідження
3.	Визначення методології дослідження та створення вимог до архітектури системи
4.	Створення тестового прототипу системи з інтеграцією NLP-сервісу та реалізацією базових модулів
5.	Впровадження механізму шаблонного парсингу для обходу обмежень зовнішніх API
6.	Проведення експериментального тестування роботи системи на тестових наборах даних
7.	Оформлення результатів дипломної роботи та оформлення пакету документів для процедури захисту КР в екзаменаційній комісії ННІ КН та ШІ.

5. Дата видачі завдання 10 грудня 2024 року

Студент  Бублик В. Р.
підпис ініціали, прізвище

Керівник роботи  Матвєєв О. М.
підпис ініціали, прізвище

АНОТАЦІЯ

Кваліфікаційна робота містить: 61 сторінок, 18 рисунків, 26 джерел.

Метою дослідження є створення програмного забезпечення для автоматизованого збору, обробки та структурування OSINT-даних стосовно кримінальних подій, за допомогою використання мовних моделей та парсерів для семантичного аналізу неструктурованих текстів.

Об’єктом дослідження є процеси обробки текстових масивів, що походять із відкритих джерел, для виявлення ознак кримінальних інцидентів. Предметом дослідження є архітектурні та алгоритмічні рішення, що забезпечують повний цикл обробки — від первинного збору інформації до її семантичного структурування і подальшого збереження у формалізованому вигляді для подальшого аналізу.

У межах роботи реалізовано прототип системи на Java з модулями для імпорту CSV, взаємодії з Groq API, вилучення ключових елементів (тип події, місце, час, особи, об'єкти) та збереження результатів у MySQL. Для підвищення надійності додано модуль очищення пошкоджених файлів.

Система показала стабільну роботу при обробці реальних даних, ефективно розпізнаючи сутності навіть у випадках неповного або синтаксично складного тексту. Результати тестування підтвердили функціональну готовність прототипу до використання в реальних умовах — як для ретроспективного аналізу масивів документів, так і для інтеграції у потокові системи моніторингу. Розроблене рішення має прикладне значення у сфері кримінальної аналітики, OSINT-досліджень, журналістських розслідувань, а також для органів, що здійснюють оперативно-розшукову діяльність.

Ключові слова: OSINT, СЕМАНТИЧНИЙ АНАЛІЗ, КРИМІНАЛЬНІ ПОДІЇ, МОВНІ МОДЕЛІ, JAVA, GROQ API, ПАРСИНГ, MYSQL, РЕЛЯЦІЙНА БАЗА ДАНИХ, РОЗПІЗНАВАННЯ СУТНОСТЕЙ.

ABSTRACT

Qualification work contains: 61 pages, 18 figures, 26 sources.

The purpose of the study is to create software for automated collection, processing and structuring of OSINT data on criminal events, using language models and parsers for semantic analysis of unstructured texts.

The object of research is the processes of processing text arrays originating from open sources to identify signs of criminal incidents. The subject of the research is architectural and algorithmic solutions that provide a full processing cycle - from the initial collection of information to its semantic structuring and further storage in a formalised form for further analysis.

As part of the work, a prototype of the system was implemented in Java with modules for importing CSV, interacting with the Groq API, extracting key elements (event type, place, time, persons, objects) and saving the results in MySQL. To improve reliability, a module for cleaning damaged files was added.

The system showed stable operation when processing real data, effectively recognising entities even in cases of incomplete or syntactically complex text. The test results confirmed the functional readiness of the prototype for use in real-world conditions, both for retrospective analysis of document arrays and for integration into streaming monitoring systems. The developed solution is of practical importance in the field of criminal analytics, OSINT research, journalistic investigations, as well as for authorities engaged in operational and investigative activities.

Keywords: OSINT, SEMANTIC ANALYSIS, CRIMINAL EVENTS, LANGUAGE MODELS, JAVA, GROQ API, PARSING, MYSQL, RELATIONAL DATABASE, ENTITY RECOGNITION.

ЗМІСТ

ПЕРЕЛІК ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

OSINT – Розвідка на основі відкритих джерел

API - АПІ -Програмний прикладний інтерфейс

AI – Штучний інтелект

ML – Машинне навчання

NLP – Обробка природної мови

БД – База Даних

Фейк – Завідомо неправдива інформація, поширена з метою маніпуляції

Парсинг – Автоматизований збір інформації з веб-ресурсів

Фактчекінг– перевірка першоджерела новини, аналіз контексту.

Текстова аналітика – аналіз змісту тексту для виявлення ключових смислів

НПУ – Національна поліція України

Краудсорсинг – передача частини завдань широкому колу людей

ЄДРСР – Єдиний державний реєстр судових рішень.

ВСТУП

У сучасній епохі цифровізації значно зросли обсяги відкритих даних, які можуть стати джерелом цінної інформації для розвідувальних та аналітичних завдань, як в хороших так і в поганих цілях. Розвідка з відкритих джерел передбачає збір та аналіз публічно доступної інформації з метою аналізу та отримання інтелектуальних висновків. OSINT давно визнаний важливим джерелом у роботі правоохоронних органів, оскільки це дає змогу багатозначно розширити контекст розслідувань та робить доступною інформацію, недоступну у внутрішніх базах даних.

З огляду на масштаб соціальних мереж і онлайн-ресурсів, ефективні алгоритми збору та обробки даних є критично важливими для OSINT-розвідки й своєчасного аналізу подій задля безпеки суспільства.

З початком повномасштабної війни проти України, інструменти OSINT отримали нове практичне значення. Завдяки аналізу повідомлень у соціальних мережах, відкритих базах даних, форумах та чатах, українські правоохоронні й воєнні структури змогли не лише виявляти загрози, а й активно запобігати терактам, координувати дії та прогнозувати події в режимі реального часу.

Прикладом може слугувати успішне використання відкритої інформації для коригування ударів по ворожій інфраструктурі або для викриття злочинних дій на тимчасово окупованих територіях. Особливої уваги потребує аналіз кримінальних подій, який базується на даних з різнорідних джерел: новинні портали, державні реєстри, соціальні мережі, форуми, мультимедійні матеріали.

Ця інформація часто є неструктурованою, містить неточності або фрагментарні факти, що вимагає розробки алгоритмів для автоматичної фільтрації, верифікації та класифікації. Проте, її ефективне використання дозволить формувати цілісну аналітичну картину злочинності та оперативно реагувати на події, прогнозувати ризики та розширювати можливості прийняття рішень.

Метою дослідження буде розробка алгоритмів та програмного забезпечення для автоматизованого збору, попередньої обробки й збереженя даних з відкритих джерел для подальшого аналізу інформації про кримінальні події. Створити методичну основу та прототип системи, що оптимізує збір OSINT-даних та готує їх до аналітичних задач.

Об’єкт дослідження: процеси та технології збирання та обробки розвідувальних даних з відкритих джерел, пов’язаних з аналізом кримінальних подій.

Предмет дослідження: Алгоритми та програмні рішення для витягання, попередньої обробки та структурування відкритих даних, що використовуються в системах аналізу кримінальних подій.

Завданням на дослідження буде проведення аналізу наявних джерел відкритої інформації про кримінальні події та їх характеристики, огляд сучасних методів та інструментів проведення розвідки з відкритих джерел OSINT, дослідження архітектурних моделей систем збору та обробки даних й визначення проблем обробки великих обсягів інформації, проекція структури системи даних, розробка й реалізація функціонального алгоритму й програми для збору й підготовки даних.

Після, провести тестування прототипу системи на реальних чи симульованих даних та оцінити її ефективність.

В роботі пропонується новий підхід до інтеграції методів OSINT у збір даних для аналізу кримінальних подій. Особливістю дослідження є розробка спеціалізованих алгоритмів для автоматизованого парсингу інформації з багатьох публічних джерел, які раніше не використовувались спільно, а також побудова архітектури системи, що забезпечує масштабованість та гнучкість при обробці великих масивів даних.

Результати дослідження можуть бути використані правоохоронними органами і аналітичними центрами для підвищення ефективності аналізу злочинності. Це дозволяє доповнити офіційні звіти і бази додатковими відомостями.

1 ДОСЛІДЖЕННЯ СПЕЦИФІКИ ТА ПРОЦЕСІВ ЗБОРУ ДАНИХ ПРО КРИМІНАЛЬНІ ПОДІЇ

1.1 Загальні відомості.

Наразі, у сучасному цифровому суспільстві обсяг доступної інформації, особливо про злочини, суттєво зріс, завдяки розвидку відкритих джерел таких як новинні портали, соціальні мережі, форуми, державні онлайн-реєстри тощо.

Для ефективного аналізу такої інформації потрібні спеціалізовані підходи до її збору, обробки та оцінки. Треба розглянути специфіку кримінальних даних, їх джерела та формат, а також проаналізувати основні проблеми, пов’язані з їх обробкою.

Для забезпечення громадської безпеки, розслідуванні правопорушень та кримінальної аналітики дуже важливу роль відіграє інформація про злочини, скоєні, чи заплановані. У сучасному світі значна частка таких даних знаходиться у відкритих джерелах у вільному доступі, на просторах інтернету, зокрема в новинних порталах, соціальних мережах(Телеграм канали, тощо), блогах та офіційних звітах правоохороних органів.

Ці ресурси дозволяють відстежувати та аналізувати злочини в реальному часі, надаючи можливість запобігти їм або вчитися передбачити запланований акт злочину. Аналіз закономірностей, аналітичні висновки та прогнозування подальших подій в сучасному світі виконуються завдяки переважно інформації з вільних джерел, автори яких, можливо навіть не уявляють, що їх інформація знадобиться для правоохоронців.

Починаючи з 24 лютого 2022 року, українські спецслужби та силові структури завдяки OSINT-розвідці запобігли більше сотням терактів, а пізніше, спостерігаючи за місцевими чатами противника, змогли наносити удари по критичній інфраструктурі та корегувати удари без візуального спостереження за результатом, а підрозділи ЗСУ, завдяки оперативній інформації з боку противника, могли корегувати свої удари для розвитку подальших дій. Загальна тенденція до змін кількості зареєстрованих злочинів в Україні

упродовж останнього десятиліття дозволяє краще зрозуміти актуальність аналітичних підходів до обробки даних. Дані представлені на рисунку 1.1.

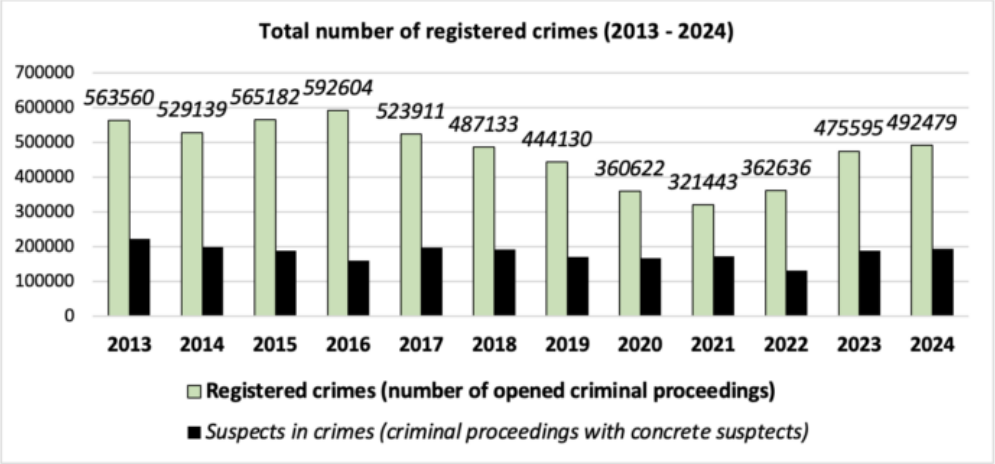


Рисунок 1.1 – діаграма загальної динаміки зареєстрованих злочинів в Україні (період 2013-2024 рр.)

Подібна оперативна інформація при правильному використанні може стати ‘’мечем’’ для зловмисника, та ‘’щитом’’ для правоохоронців, тому разом з спецслужбами, вмілі злочинці теж користуються вільними джерелами інформаціями.

Таким чином, ефективний аналіз кримінальних подій потребує застосування спеціалізованих алгоритмів для збору, фільтрації та оцінки достовірності даних. Це дозволить автоматизувати обробку інформації та отримати більш точну картину криміногенної ситуації в реальному часі.

1.1.1 Специфіка даних про злочини.

Основними джерелами в таких випадках можуть бути: Новинні портали та ЗМІ є найбільш доступним та простим джерелом інформації, офіційні ЗМІ та великі інформаційні агентства публікують оперативні дані про злочини, спираючись на офіційні джерела та звіти слідчих, деколи, на журналістські розслідування. Офіційні ресурси правоохоронних органів, такі як поліція, прокуратура, державний реєстр судових рішень або інтерпол, публікують

звіти про злочини, кримінальну статистику та аналітичні матеріали. Соціальні мережі та форуми поширюють багато інформації про злочини, якщо офіційні джерела не встигають оперативно реагувати, простий міський чат в публічному доступі може дати оперативну інформацію швидше, ніж хтось встигне викликати поліцію, адже в першу чергу, інформацію поширюють живі люди, а учасники соціальних мереж перевіряють збіги в коментарях, в залежності від почутого або побаченого. Darknet та спеціалізовані бази даних, під час витоку інформації, не зовсім приносять оперативну інформацію, але дозволяють оцінити та проаналізувати вже скоєне, що б запобігти потенціальним злочинам, що теж є корисним джерелом інформації. Деякі витоки інформації допомагають розкривати злочини, які були нерозкриті протягом десятиліть, а пізніше подібні випадки, як приклад, використовують для вивчення в академіях внутрішніх справ.

З основних джерел можна зрозуміти, що інформація про злочини є специфічним типом даних, яка характеризується високим ступенем чутливості, варіативністю джерел і необхідністю ретельної верифікації. Маючи різне походження, та відрізняючись рівнем деталізації, така інформація може також містити як об’єктивну інформацію, так і суб’єктивні оцінки. Кожне з зазначених джерел вище має власні особливості щодо точності, оперативності та об’єктивності інформації.

Як зазначено вище, кожне з цих джерел має власні особливості щодо точності, оперативності та об’єктивності інформації, та через специфіку злочинної діяльності, інформація про правопорушення часто має неструктурований характер. Дані можуть бути представлені у вигляді текстових звітів, відеоматеріалів, аудіозаписів, картографічних даних, тощо. Це вимагає від розробників алгоритмів працювати не тільки над аналізом тексту, а й розпізнаванням облич та інших особливих прикмет, таких як номерні знаки, також аналіз геолокаційних даних для виявлення місця злочину.

Таким чином, специфіка даних про злочини обумовлює необхідність використання методів штучного інтелекту, автоматичної класифікації та оцінки достовірності інформації.

1.1.2 Проблематика роботи з інформацією про злочини.

Сама по собі обробка інформації про злочини є доволі складним завданням, що пов’язане з низкою технічних, правових та організаційних проблем. Інформація про злочини може надходити з різних джерел, містити неточності або бути виправленою в залежності від джерела, бути фрагментованою або суб’єктивною.

Додаткову складність створює динамічний характер злочинної активності, що вимагає аналіз даних в реальному часі й не дозволяє довго зупинятися на одній справі. Саме це є причиною більшості нерозкритих злочинів, від дрібних до важких в аналізі, правоохоронцю немає часу на тлі нових потенційних, актуальних або важливіших злочинів.

Як було описано раніше, з ключових проблем є відсутність єдиних стандартів щодо формату збереження та представлення даних про злочини. Інформація може бути подана у вигляді:

-Текстових звітів (поліцейські звіти, статті у ЗМІ)

-Відео та фотофайлів (записи камер спостереження, зйомки очевидців)

-Графічних матеріалів (карти злочинів, схеми злочинних угруповань)

Кожен із цих форматів вимагає окремих методів обробки, що ускладнює автоматизацію аналізу.

Також відсутність централізованої бази даних є проблемою роботи з інформацією. Дані про злочини часто ‘’розпорошені’’ між різними джерелами, зазначеними вище: державними органами, медіа, та громадськими організаціями.

Відсутність централізованого сховища інформації ускладнює швидкий доступ до необхідних даних і гальмує процес аналітики.

В залежності від розкриття злочину або зміни кваліфікації правопорушення, виникає проблема зміни статусу злочинів. Ця інформація не завжди оперативно оновлюється в базах даних. Крім того, одне й те саме правопорушення може висвітлюватися у різних джерелах з різними подробицями, що призводить до дублювання інформації і теж ускладнює аналіз.

Варто розуміти ще те, що може виникати не просто дублювання інформації, а і спотворення її, або фактів під час скоєння злочину.

Нерідко інформація про злочини використовується для маніпуляції суспільною думкою. Вона може бути подана з викривленням фактів або перебільшенням деталей, для створення паніки або хибного уявлення про ситуацію.

Основні ризики поширення неправдивої інформації:

Політичні маніпуляції – для дискредитації певних груп, партій чи влади.

Дезінформація у соціальних мережах – поширення чуток та непідтверджених повідомлень для створення паніки

Свідоме перебільшення злочинності – теж вид створення панічних настроїв з комерційною або політичною метою.

Щоб уникнути маніпуляцій, необхідно використовувати алгоритми перевірки інформації:

Фактчекінг – перевірка першоджерела новини, аналіз контексту

Лінгвістичний аналіз – автоматичне визначення ознак маніпулятивних текстів.

Порівняльний аналіз джерел – зіставлення даних із кількох незалежних джерел.

Законодавство багатьох країн обмежує розголошення особистої інформації про потерпілих, підозрюваних та свідків.

Проте відкриті джерела нерідко можуть порушувати законодавство, маючи дані ПІБ осіб, причетних до злочину, їх місце проживання, або фотографії/відео з місця подій.

Робота з такою інформацією вимагає ретельного контролю та дотримання правових норм, адже без потрібних повноважень, з такими даними можна самому стати злочинцем. Тобто, для дотримання юридичних норм(в випадку якщо OSINT розвідкою займається не уповноважена особа) потрібно використовувати лише публічні джерела, анонімізувати повністю дані та використовувати агреговані статистичні дані без персональних деталей.

1.2 Постановка задачі .

Об’єктом дослідження у цій роботі є інформація про кримінальні події, яка поширюється у відкритих джерелах, зокрема: новинні сайти, соціальні мережі, форуми, офіційні портали державних установ, публічні бази даних, а також мультимедійні матеріали.

Предметом дослідження є алгоритми збору, обробки та перевірки достовірності цієї інформації з відкритих джерел, які можуть бути використані для аналізу криміногенної ситуації в межах OSINT-розвідки з відкритих джерел.

Мета роботи полягає у розробкі гібридної системи збору та підготовки інформації з відкритих джерел для подальшого аналізу кримінальних подій, яка поєднує методи обробки природньої мови, машинного навчання, аналізу зображень та геоаналітики. Для досягнення поставленої мети роботи, необхідно вирішити наступні завдання:

Провести аналіз відкритих джерел даних, які містять інформацію про кримінальні події, та визначити їх специфіку, формат і ступінь достовірності. Дослідити сучасні підходи до обробки неструктурованої інформації, зокрема застосування методів NLP (Natural Language Processing), аналізу зображень, відео, метаданих та геолокацій, та розробити гібридний підхід до збору, фільтрації, класифікації та перевірки даних.

Цей підхід із поєднанням таких алгоритмів: NLP для витягування сутностей, виявлення ключових слів для класифікації типу злочину, сентимент-аналіз для емоційної оцінки повідомлень, розпізнавання облич і

номерних знаків для фото або відео-аналізу, та аналіз дій на відео та геоприв’язка подій.

Реалізувати прототип системи на основі Java та open-source бібліотек для збору, обробки і зберігання інформації. Протестувати систему на змодельованих або реальних наборах даних та оцінити її ефективність за метриками точності, повноти та продуктивності.

Актуальність дослідження обумовлена зростанням кількості злочинів, які висвітлюються у цифровому середовищі, та нагальною потребою в ефективних засобах автоматизованого аналізу такого роду інформації. Через відсутність інтегрованих рішень, що могли б об’єднувати дані з різних відкритих джерел, існує потреба у створенні системи, яка дозволить швидко виявляти критично важливу інформацію, прогнозувати ризики та підвищити ефективність аналітичних підрозділів правоохоронних органів.

Розроблена система матиме здатність не лише автоматизувати збір великої кількості інфорації, а й виявляти ключові закономірності, що сприяє оперативному реагуванню та прийняттю рішень. Практична реалізація такого підходу дозволяє покращити ситуаційну обізнаність аналітичних підрозділів, зменшити вплив людського фактору та підвищити ефективність розслідувань та запобігання злочинності.

2 АНАЛІЗ МОДЕЛЕЙ СПЕЦИФІКИ ПІДХОДІВ ДО ПОШУКУ ТА ЗБОРУ КРИМІНАЛЬНОЇ ІНФОРМАЦІЇ

2.1 Аналітична роль інформації у аналізі кримінальних подій.

З розвитком OSINT-методів з’явилася потреба у створенні програмних рішень, які дозволяють автоматично здійснювати збір, можливо обробку, збереження та аналіз інформації з різних відкритих джерел. Такі системи за своєю структурою подібні до IT-проєктів аналітичного типу, що обслуговують конкретні задачі – у нашому випадку, аналіз кримінальних подій. Тому доцільним є розгляд існуючих моделей реалізації подібних програмних систем.

Існуючі програмні засоби OSINT-розвідки можна умовно поділити на централізовані, децентралізовані, гібридні та модульні моделі.

Система централізованих моделей зосереджена на єдиному серверному рішенні, де відбувається збір, обробка та аналіз усієї інформації. Така архітектура дозволяє підтримувати високий рівень контролю та безпеки, однак зменшує гнучкість та масштабованість системи.

Децентралізовані моделі передбачають розподіл функцій між кількома вузлами – наприклад, один модуль відповідає за парсинг, інший – за NLP-аналіз, ще інший – за візуалізацію. Такі моделі підходять до створень більш гнучких та адаптивних систем, проте ускладнює керування загальним середовищем та вимагає додаткових засобів синхронізації між собою.

Гібридна модель поєднує в собі переваги централізованих і децентралізованих моделей: загальна база даних чи хмарна частина об’єднує оброблену інформацію, в той час як окремі компоненти можуть функціонувати незалежно або у вигляді мікросервісів. Підхід через гібридну модель вважається одним із найперспективніших у розробці систем OSINT-розвідки, оскільки дозволяє ефективно масштабувати рішення, підключати сторонні сервіси для кращого аналізу будь-якого формату інформації (аналіз зображень, переклад з інших мов) та мінімізувати навантаження на окремі вузли, що дасть оптимізацію під час роботи. Архітектура системи формує

фундамент для реалізації її компонентів. Як система розгортається за модульною схемою, враховуючи зовнішні та мікросервіси, показано на рисунку 2.1

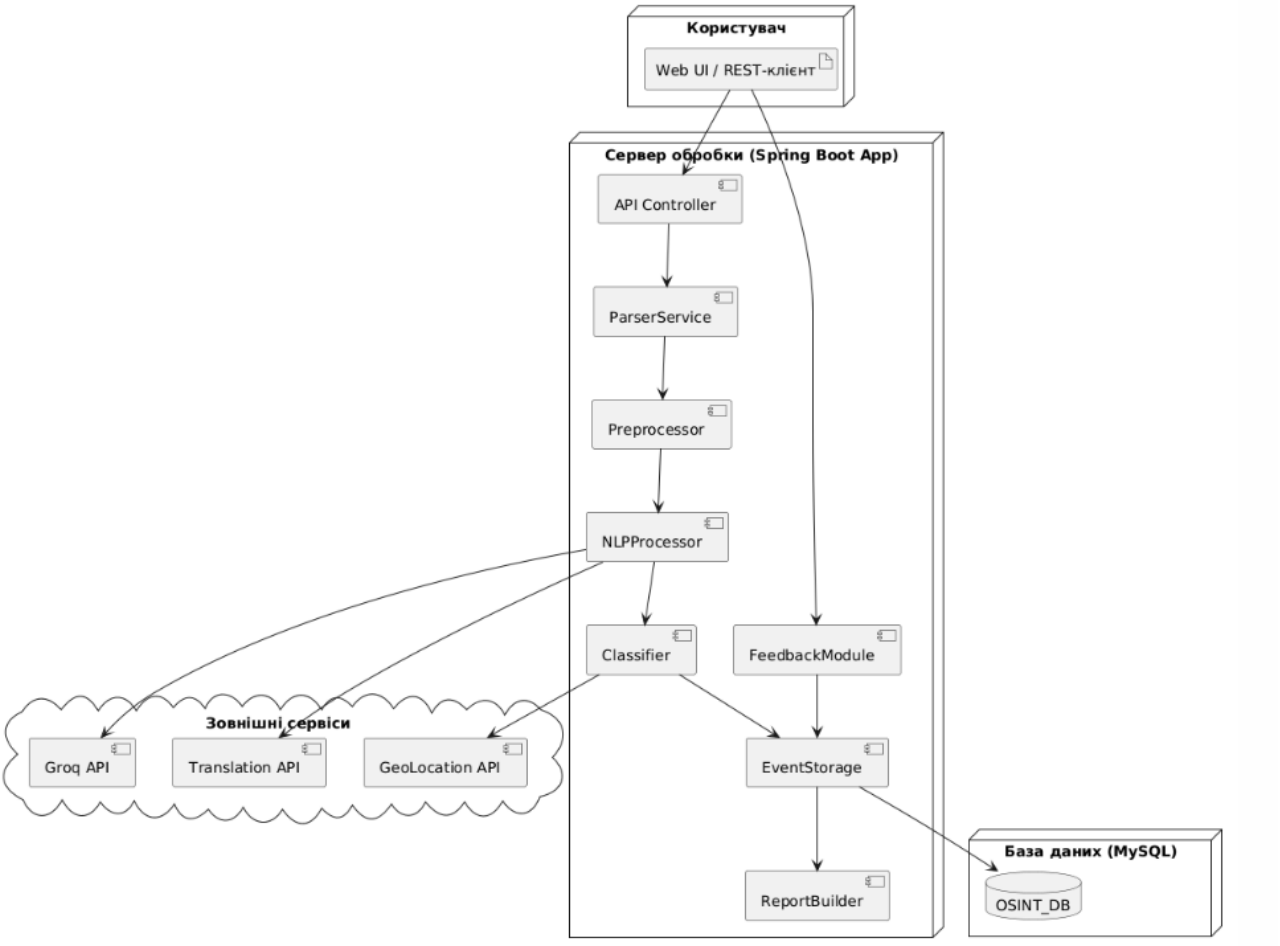


Рисунок 2.1 – Діаграма розгортання

Враховуючи задачі, які ставляться перед системою, доцільно вибирати модель, яка найкраще відповідає вимогам надійності, швидкості, масштабованості, безпеки та гнучкості оновлення.

Таким чином, гібридна, багатомодульна структура з можливістю інтеграції сторонніх сервісів, масштабуванням та автоматизованою перевіркою достовірності зібраних даних, буде найбільш ефективним рішенням, враховуючи задачу збору та підготовки даних для аналізу кримінальних подій.

Модель модульної архітектури передбачає чітке розділення функціональності за модулями.

2.1.1 Аналіз ресурсів.

Так як ресурси для збору даних про кримінальні події відіграють ключову роль в підходах OSINT-розвідки, правильний вибір та класифікація цих ресурсів є критично важливими для забезпечення повноти, актуальності та достовірності зібраних даних. Від ресурсу залежить не лише якість, настрій або контекст отриманої інформації, а й технічна можливість її обробки. Ресурси – джерела, з яких безпосередньо здійснюється отримання інформації.

Як правило, найбільш достовірними та офіційними джерелами є державні ресурси, оскільки вони публікують інформацію, засновану на первинних документах або перевірених даних від правоохоронних структур.

Основним джерелом з державних ресурсів буде Єдиний державний реєстр судових рішень[11], крім нього, до таких ресурсів належать:

- Сайт НПУ (Національної поліції України) [1]
- Сайт Офісу Генерального прокурора України [2]
- Головне управління ДСНС у Львівській області [3]
- Львівська обласна державна адміністрація [4]
- Львівська міська рада [5]
- Сайт СБУ (Служби безпеки України) [6]

Перевагою таких ресурсів, буде юридичне підтвердження фактів та абсолютно офіційнй характер.

Серед недоліків – повільне оновлення через щільну перевірку інформації, відсутність структурованих API, що є проблематикою автоматичного збору даних, та складність навігації.

Новинні сайти є основним джерелом оперативної інформації, включно з кримінальними.

- ТСН (Телевізійна служба новин) [7]
- Українська правда [8]

- УНІАН [9]

- BBC Україна [10]

Вони оновлюються швидко, мають зручну структуру сторінок, але водночас можуть містити неточності, перебільшення або елементи "клікбейту". Також трапляється відсутність структурованих даних, що ускладнює збір.

Існують спеціалізовані сервіси, що створені спеціально для моніторингу криміногенної ситуації, а також публічні бази даних:

-OpenCrimeMap (переважно США, але як приклад структури);

-Bellingcat (розслідувальна спільнота з методами OSINT);

-OSINT Framework – каталог інструментів;

-GeoJSON-бази для геоаналітики.

Схожим прикладом в Україні є сервіс DeepState.ua – моніторинг лінії зіткнення, що здійснюється OSINT-розвідкою.

DeepState відрізняється тим, що нерідко бере інформацію з краудсорсингу та інших неформальних джерел, в той час як приклад структур з США, чекає на офіційні звіти або оперативну інформацію від ЗМІ.

‘’Неформальними джерелами’’ можна назвати канали, форуми в соціальних мережах, власники яких, за своєї ініціативи, повідомляють про події, що дізнаються від свідків, місцевих, в випадку бойових дій, це може бути корисним та оперативним джерелом інформації, але самі власники DeepState просять не планувати поїздки або плани евакуації використовуючи інформацію про переміщення військ з сайту, та позиціонує себе просто як моніторинговий сервіс лінії зіткнення.

Альтернативні джерела неофіційної інформації, можуть бути раніше за офіційну, проте мають найнижчий рівень достовірності та вимагає глибокої модерації або будь-якої перевірки. Ефективна система збору інформації для OSINT-аналізу кримінальних подій повинна враховувати такий широкий спектр джерел: від офіційних до неформальних, від структурованих до повністю хаотичних.

Оптимальним буде підхід, при якому для кожного типу ресурсу застосовуються індивідуальні методи збору, перевірки та обробки.

2.1.2 Методи збору інформації з ресурсів.

Збір даних для аналізу кримінальних подій у рамках OSINT передбачає використання різноманітних відкритих джерел (вебсайти, новинні портали, соціальні мережі, форуми, урядові реєстри тощо) та відповідних технік збору. Типові формати даних у відкритих джерелах включають HTML-сторінки, тексти, PDF-файли, таблиці (CSV, Excel), API (JSON, XML), RSS-стрічки новин, мультимедійні дані (зображення, відео) тощо. Наприклад, інформацію про кримінальні події можуть містити як текстові анонси у новинах (HTML, JSON-логи сайтів) так і документи судових рішень у PDF чи HTML-форматах (іноді опубліковані в Єдиному державному реєстрі судових рішень).

Техніки збору включають: API-запити: багато ресурсів надають публічні (або напівпублічні) API [13]. Так, соціальні мережі (Twitter, Facebook, YouTube) мають API для програмної вибірки даних (стрічки дописів, профілі користувачів, метадані постів тощо). Використовуючи стандартні HTTP-запити до API, отримують структуровані відповіді у форматах JSON чи XML (наприклад, запит до Twitter API для вилучення твітів за ключовими словами). Крім соціальних мереж, урядові органи часто надають API доступ до відкритих реєстрів: прикладом є API сервісу “Суд на долоні”, що дозволяє шукати судові рішення у форматі JSON

Веб-скрапінг (парсинг): коли API відсутнє або обмежене, використовують «витягування» інформації безпосередньо з HTML-сторінок. Для цього застосовують бібліотеки на кшталт BeautifulSoup, Scrapy або Selenium у Python. Наприклад, можна зробити HTTP-запит до сторінки з рішенням суду або новиною, а потім видобути необхідні поля (дата, місце події, опис) з HTML. Для зручності деякі ресурси мають RSS/Atom-канали новин, які можна обробити через парсери XML.

Пошукові запити та індексація: для початкового збору використовують загальні пошукові системи або спеціалізовані агрегатори. Наприклад, пошукові API Google чи Bing (хоч і обмежені) можуть надати посилання на релевантні статті чи блоги за ключовими словами «кримінал + регіон». Додатково застосовують веб-архіви (Wayback Machine) для отримання старих версій сторінок із інформацією про події, якщо потрібні історичні дані.

Спеціалізовані OSINT-інструменти: існують готові платформи, що інтегрують різні джерела. Наприклад, Maltego або SpiderFoot забезпечують зручний інтерфейс для «машинного» збору даних з відкритих джерел (пошук зв'язків між IP-адресами, доменами, електронними адресами тощо) [14]. Вони мають вбудовані «машини» або модулі для взаємодії з численними API та веб-ресурсами, автоматично будують граф зв'язків. Хоча такі інструменти націлені на кібернетичну безпеку, принцип їх роботи аналогічний збору будь-яких даних з відкритих джерел.

Для підготовки зібраних даних часто застосовують автоматизовані скрипти та ETL-процеси. Наприклад, після отримання списку URL-адрес рішень суду, можна по черзі виконати запити, зберегти текстові версії документів, перевести їх у структурований вигляд (через регулярні вирази або NLP для вилучення ключових полів). Часто поєднують різні методи: спочатку API-запитом отримують список ідентифікаторів, потім парсером завантажують їх текст. Усе це супроводжується збереженням метаданих (дата, джерело) та контролем актуальності. У підсумку збір інформації здійснюється усіма можливими відкритими шляхами: API, веб-скрапінг (парсери на Python, Node.js), аналіз медіа (пошук за зображеннями, відео, геолокація).

Методи збору даних для OSINT базуються на комбінації доступних інтерфейсів та протоколів: офіційних API відкритих реєстрів, RSS-каналів, веб-скрапінгу та пошукових технологій. Завдяки цьому інформація з різних відкритих джерел (соцмережі, новинні сайти, публічні бази) стає доступною для автоматичної обробки. Головною перевагою такого підходу є велика

кількість джерел і мізерні матеріальні витрати (адже все відкрито і безкоштовно)

2.2. Проблеми узгодження інформації з різних ресурсів.

Зібрані з відкритих джерел дані часто мають неоднорідний формат і структуру, що ускладнює їх інтеграцію. Наприклад, один сайт публікує статистику у вигляді HTML-таблиці, інший – у PDF-файлі, третій – через JSON-API. Щоб об’єднати ці дані, потрібно уніфікувати формати: перетворити PDF або HTML в уніфікований таблицю (з графічних документів – OCR/PDF-парсинг) і заповнити спільні поля. Складність полягає у відмінностях у найменуваннях полів (наприклад, «дата», «Дата_публікації», «timestamp»), одиницях виміру, мовних варіантах (різні мови чи транскрипції імен) тощо.

Також можуть бути різні часові пояси, формати дат чи несинхронізовані індекси (наприклад, різні ID подій в різних системах). Ще одна проблема – дублювання та суперечливість даних. Одну і ту ж кримінальну подію можуть висвітлювати кілька джерел з різними деталями (наприклад, різні рівні деталізації в ЗМІ та офіційних звітах). Під час інтеграції необхідно зіставляти записи і «зливати» їх у єдине уявлення про подію. Тут застосовують алгоритми пошуку схожих записів (fuzzy matching) за ключовими ознаками: дати, місце, ключові слова. Етимологія назв також ускладнює узгодження: назва організації чи прізвище можуть записані по-різному. Тому часто використовують словники відповідностей або напівавтоматичні рішення з людино-машинною перевіркою, щоб «зшити» інформацію з різних джерел.

Крім того, дані з відкритих джерел можуть бути неповними або застарілими. Оскільки багато реєстрів оновлюються з затримкою через експертизу (наприклад, сайт судової влади може публікувати рішення лише після їх перевірки), виникають розриви в актуальності. Відкриті дані часто розміщують в архівах (наприклад, на data.gov.ua опубліковано річний зріз ЄДРСР у вигляді XLSX/PDF), що не дає змоги отримати свіжі дані «на льоту». Також намагання автоматично збирати інформацію з офіційних сайтів іноді

блокується захистом від скрапінгу (CAPTCHA, обмежена швидкість запитів). Наприклад, Реєстр судових рішень вимагає введення капчі, що ускладнює написання повністю автоматичного парсера.

У підсумку різнорідність форматів, відсутність стандартизованих API, різні частоти оновлення та захист від автоматизації створюють серйозні труднощі при узгодженні даних з багатьох джерел.

2.2 Сучасний стан та шляхи удосконалення систем збору OSINT-даних

2.2.1 Аналіз існуючих рішень.

Існують різні сервіси та проекти, які забезпечують збір або уніфікацію інформації з відкритих джерел. Серед них можна виділити як загальні OSINT-інструменти, так і спеціалізовані українські сервіси.

Наприклад, Opendatabot – українська компанія, що надає агрегований доступ до даних кількох державних реєстрів (підприємства, судові рішення, борги, нерухомість тощо) через веб-інтерфейс та API. Через їхній API можна отримати дані про судові рішення, зареєстровані компанії тощо, консолідовано з різних джерел. Ще один приклад – проєкт «Суд на долоні», який реалізує власні RESTful API до реєстру судових рішень [11].

Його ендпоінти дозволяють отримувати кількість рішень і деталі конкретного рішення за ідентифікатором. Цей сервіс оновлюється щодня і базується на відкритих даних Державної судової адміністрації.

Крім того, офіційний сайт Єдиного державного реєстру судових рішень сам по собі є джерелом даних – ним користуються пошукові сервіси (наприклад, Національної поліції чи органів прокуратури), хоча прямого публічного API для масового збирання не має (захищений кабінетом та CAPTCHA). Також на data.gov.ua (державному порталі відкритих даних) публікують готові набори даних, наприклад, архіви реєстру судових рішень за 2016 та 2019 роки у вигляді Excel/PDF.

Для загального OSINT-аналітика також корисні фреймворки та посібники: існує каталог безкоштовних OSINT-ресурсів (OSINT Framework), онлайн-курс чи рекомендації з логічними операторами пошуку. Серед інших прикладів можна назвати проекти Bellingcat – міжнародна спільнота онлайн-розслідувачів, що використовує відкриті джерела для виявлення фактів (є блог та інструментарій перевірки даних), а також дослідницькі платформи (OpenArchives, OCCRP Aleph тощо). В Україні виділяють сервіси моніторингу (наприклад, DeepState.ua – збирання повідомлень свідків бойових дій) та аналітичні проєкти, що забезпечують зшивання даних із соцмереж, чатів, публічних карт. Усі ці рішення націлені на збирання інформації із широкого спектру каналів, але часто потребують подальшої обробки та верифікації.

2.2.2 Виявлення проблем в існуючих рішеннях.

Хоча згадані сервіси надають зручні можливості, у них є суттєві недоліки. По-перше, тимчасова розбіжність і частота оновлення: наприклад, API «Суд на долоні» оновлюється лише раз на день або двічі на тиждень, тоді як сучасний аналітик потребує максимально свіжих даних. Те саме стосується архівних датасетів на data.gov.ua – вони статичні і не містять поточних доданих документів.

По-друге, формат даних: як видно з прикладу відкритих наборів, реєстр судових рішень представлено у різних форматах – XLSX (2019 рік) і PDF (2016 рік).

PDF-файли погано піддаються автоматичній обробці, що ускладнює витяг інформації. Через це навіть офіційні дані можуть знадобитися додаткового перетворення (OCR або конвертації). По-третє, обмеження доступу: як уже відзначалося, наявність захисту від скриптів (CAPTCHA) і відсутність відкритого API змушують розробників писати «криві» парсери або використовувати дорогі платні рішення. Крім того, відкриті дані часто не містять персональних ідентифікаторів – наприклад, у ЄДРСР заборонено публікувати ПІБ учасників справ. Це унеможливлює пошук рішень по

прізвищу підозрюваного чи потерпілого, що суттєво обмежує можливості об’єднати дані з соцмереж чи телефонних баз. Ще одна проблема – різнорідність термінології й структур. Різні реєстри або ЗМІ можуть використовувати власні категорії чи умовні позначення (наприклад, формулювання «кримінальна стаття», «відділення поліції», «рег. номер справи»). У відсутності уніфікованого словника потрібно застосовувати ручну нормалізацію (налаштовувати відповідності) або машинне навчання для автозіставлення.

Також на практиці виникає ненадійність та маніпуляції у відкритих джерелах: новинні сайти можуть містити неточні або перекручені дані (клікбейт), а неофіційні канали поширюють чутки. Ці фактори теж входять до «проблем існуючих рішень» – дані доводиться додатково фільтрувати та перевіряти вручну. Нарешті, у деяких проектів є обмежена масштабованість: багато інструментів розраховані на певний набір джерел або працюють тільки в межах однієї країни/тематики і їх важко адаптувати для інших наборів даних. Отже, загальні недоліки полягають у низькій оперативності оновлень, неоднорідності форматів даних, недостатній автоматизації через API, імовірному недостовірному вмісті та необхідності ручної інтеграції.

2.2.3 Можливі варіанти вирішення існуючих проблем.

Для подолання виявлених обмежень можна запропонувати кілька підходів. По-перше, уніфікація форматів через ETL-процеси: створення конвеєрів, які перетворюють дані у спільну структуру. Це може бути, наприклад, перенесення всіх даних у реляційну БД або NoSQL-структуру з чітко визначеним набором полів (дата, місце, опис тощо). Для PDF і зображень використовують OCR та шаблонні алгоритми, а для HTML – бібліотеки парсингу. Таким чином, вихідні дані нормалізуються: цифри приведені до єдиного формату дат, єдині позначення статей, уніфіковані геолокації (наприклад, використання стандартних геокодів). Для відображення логіки

збереження структурованих даних застосовується ER-модель, як показано на рисунку 2.2.3.1 Вона демонструє основні сутності та їх зв’язки в базі даних.

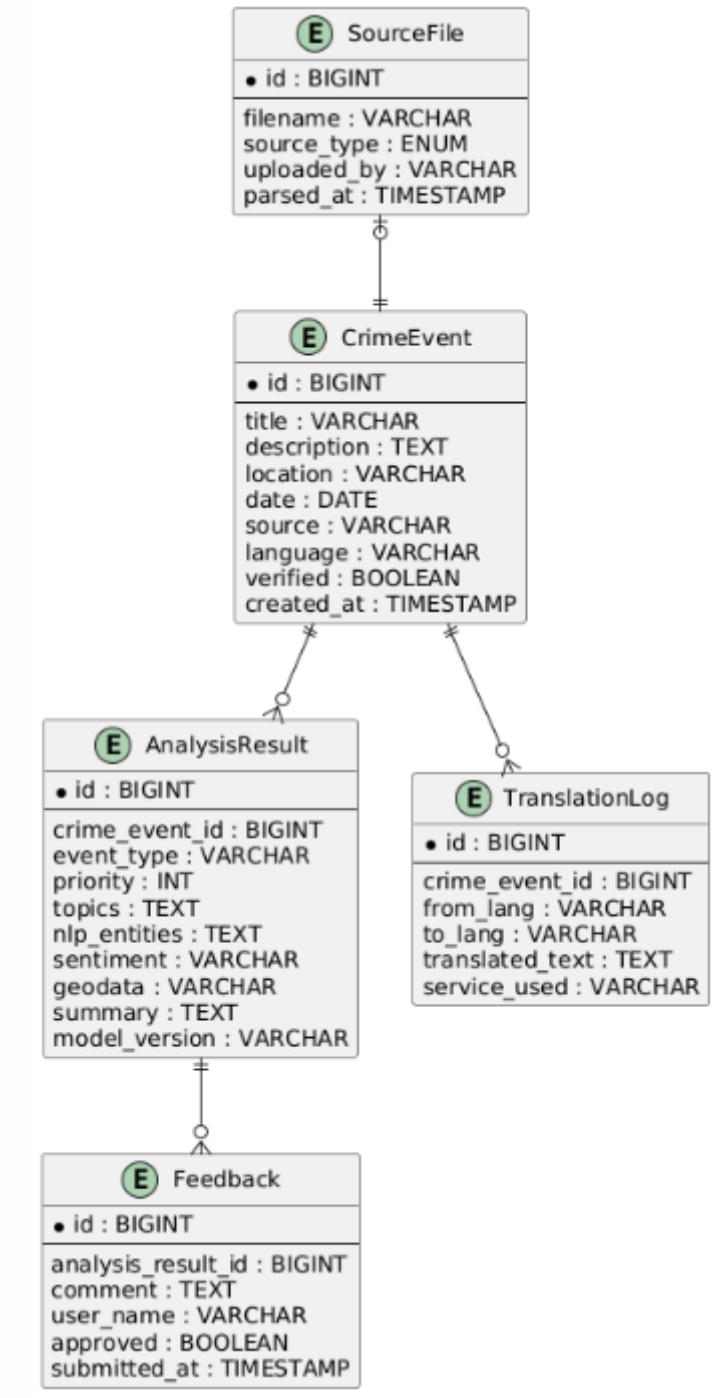


Рисунок 2.2.3.1 – ER-діаграма

По-друге, використання технологій обробки природної мови (NLP): для розпізнавання імен, географічних назв, дат з необробленого тексту рішень або новин можна застосовувати готові моделі розпізнавання сутностей (NER). Це

дозволить автоматично витягнути ключові сутності з повідомлень і порівнювати їх із базою (наприклад, зі списком усіх населених пунктів або базою даних суддів). NLP-аналітика допоможе знаходити однакові події в двох джерелах, навіть якщо вони описані по-різному.

Третє – графові моделі знань та семантичні технології: можна будувати єдині графи об’єктів (люди, організації, події), де зв’язки формуються на основі атрибутів (збіг імен, адрес, номерів справ). Для узгодження форматів пропонується використовувати загальні стандарти (наприклад, схеми JSON-LD, RDF або національні класифікації). По можливості, варто інтегрувати всі джерела у спільну аналітичну платформу з потужним API та інтерфейсом. Наприклад, продовжувати розробку таких проектів як «Суд на долоні» з відкритим кодом, які вже поєднують різні реєстри в єдиному API.

Нарешті, важливо розширювати використання краудсорсингу та верифікації: залучати користувачів або співробітників правоохоронних органів до підтвердження даних (фактчекінг). Це можна зробити через внутрішні механізми звітності (feedback loops) при злитті джерел, або через штучний інтелект, який знаходитиме суперечності між джерелами і проситиме людину уточнити. Такий гібридний підхід (машина + людина) дозволить оперативно відфільтровувати помилкові вкиди і бути готовим до швидких змін.

Для вирішення існуючих проблем слід комбінувати технічні методи (ETL, NLP, API) та організаційні підходи (коштівний доступ, співпраця з джерелами, верифікація). Оптимальною є побудова модульної системи: для кожного типу джерела (ZMI, реєстр, соцмережі) – своя «ланка» збору та нормалізації, а потім централізований компонент зведення і аналізу. Це дозволить автоматизувати інтеграцію даних і поліпшити якість узгоджених результатів.

2.3 Проблематика автоматизованого збору даних та подолання технічних бар'єрів

Модуль парсеру повинен виконувати базову задачу зі збором даних для забезпечення автоматичного збору відкритої інформації (OSINT). Він має виконувати початкове завантаження, структуризацію та збереження даних для подальшої обробки. Такий парсер може працювати різними методами, що підходять під різні задачі.

Парсинг веб-сторінок через Selenium є одним з найгнучкіших варіантів, та може бути написаний на Java або Python. Він імітує реальну поведінку користувача в браузері, що дозволяє завантажувати JS-контент, взаємодіяти з функціоналом сайту: кнопками, вкладками, формами та отримувати динамічні таблиці.

Наприклад, при зборі звітів з сайтів судових реєстрів або з новинних порталів, які використовують AJAX. В таких випадках, Selenium дозволяє «проскролити» сторінку, дочекатись завантаження, витягнути таблиці й зберегти їх у форматі CSV. Недоліком такого підходу є порівняно низька швидкість та високе ресурсоспоживання.

Також комп’ютерний тест на робота Captcha може заважати правильному використанню Selenium та в цілому є однією з головних перешкод у зборі OSINT-даних.

При спробі автоматичного доступу до інформаційних ресурсів з використаням Selenium (або подібних засобів автоматизації браузера), часто виникає проблема блокування — ресурси впроваджують CAPTCHA щоб відрізнити бота від людини.

Для обходу такого захисту можна використати спеціальні сервіси антикапча.

Це спеціальні платформи, які вирішують CAPTCHA за вас. Бот надсилає зображення або токен на сервіс, а у відповідь отримує розгадану капчу.

Як приклад можна використати сервіси які використовують ШІ для обходу капчі: CapMonster, DeathByCaptcha, 2Captcha, Anti-Captcha [13]

З Selenium це буде працювати так:

Елемент капчі зчитується на сторінці

Передає токен до API сервісу.

Отримує відповідь і вставляє її у форму сторінки

Сайт завантажується й Selenium готовий до подальшої взаємодії

Ці сервіси платні, але ефективні й автоматизовані, через HTTP-запит можуть бути інтегрованими до самого Selenium.

З недоліків: велике використання ресурсів та платний окремий сервіс. З переваг: найефективніший засіб обходу капчі.

Також є методи моделювання поведінки користувача, виконуючи дії з відповідною затримкою, тощо, але сучасна CAPTCHA враховує на правильне виконання задачі, а не зчитує поведінку курсору користувача, тому не всюди може бути ефективним.

Є метод з використанням проксі + обхід капчі через зміну IP + cookies

Деякі ресурси блокують повторні запити з однієї адреси, або виявляють сліди Selenium через специфічні ознаки.

Однак використовуючи антидетект-браузери або VPN сервіси можна обійти подібний захист, що не є всюди дуже ефективним, але може бути варіантом для обходу захистів на деяких сайтах.

В методі з Selenium, захист капча є великим недоліком, адже крім зборів даних, його можуть використовувати для DDOS-атак, тому сайти забезпечують себе відповідним захистом. Проте, для збору даних можна використовувати готові файли зі звітами, реєстрами формату CSV/JSON, де знаходяться всі дані з таблиць для парсингу, вони не є динамічними або активними, та використовуються як архів, але є варіантом для OSINT збору даних, так як не потребують проходження капчі, і лише потрібен готовий архів з документом.

3 РОЗРОБКА МОДЕЛЕЙ ТА АЛГОРИТМІВ ДЛЯ РЕАЛІЗАЦІЇ ГІБРИДНОЇ МОДЕЛІ АНАЛІЗУ ДАНИХ

3.1 Архітектура та реалізація модуля парсингу в OSINT-системі

У цьому проєкті має бути реалізовано кілька функціональних компонентів: модулі збору даних з відкритих джерел (веб-скрейпери, API-запити до соціальних мереж і новинних порталів), модуль підготовки та очищення даних, у випадку якщо база даних не підійде для формату роботи програми, модуль фільтрації та попередньої класифікації, а також блоки машинного навчання (класифікаційний ядро) і сховище результатів. Залучення зовнішніх сервісів дозволяє виконувати, наприклад, переклад тексту або додатковий аналіз зображень, зменшуючи навантаження на центральну систему. Код проєкту написаний на Java з використанням Maven (pom.xml містить залежності), а його архітектура модульна: кожен пакет відповідає за окрему частину (збір, зберігання, обробка, аналіз). Може бути використаний також Python. Як будуть виглядати об’єкти системи та зв’язки між ними описано у вигляді діаграмі класів на рисунку 3.1.

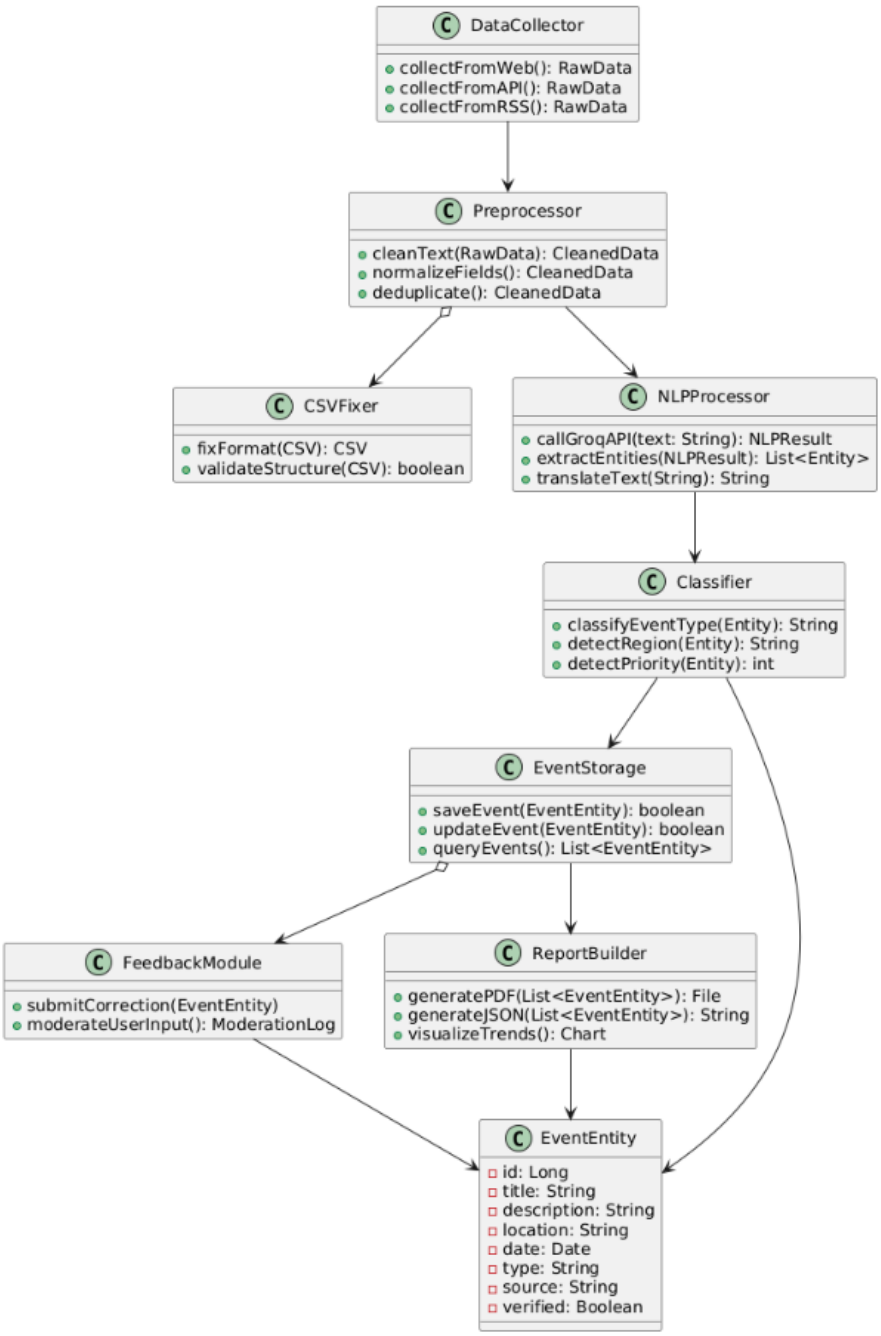


Рисунок 3.1 – Діаграма класів

Якщо відкриті джерела надають готові масиви даних (в нашому випадку це статистика з сайту МВС, відкриті дані з Єдиного реєстру судових рішень), то їх можна автоматично завантажувати та обробляти через модуль парсера [11]. У цьому випадку програма читатиме файли у форматі CSV, JSON, XML або навіть XLSX, нормалізує назви колонок, усуває дублі, виправляє кодування (UTF-8/BOM), а потім зберігає у внутрішньому структурованому

форматі. Такий парсер легко реалізується через стандартні бібліотеки Java (OpenCSV, Jackson, Apache POI) або Python (pandas, csv, json, openpyxl).

З недоліків лише те, що переважна кількість таких даних є архівами або звітами за певний проміжок часу, й не оновлюються пізніше, не є активною таблицею, тому цей метод не підійде для швидкого аналізу злочину, що щойно відбувся, або про який щойно звітували в реєстрі. Попри затримку, такі дані можна використовувати для тестування або вивчення моделі, що б до актуальних новин вона видавала кращий результат.

API-збір з відкритих джерел

Багато платформ (наприклад, поліція, новини, соцмережі) мають публічні REST API, через які можна отримувати структуровані дані у JSON/CSV-форматі [24]. Парсер у цьому випадку виконує запити (наприклад, GET /*назва даних для пошуку*) і перетворює відповіді на уніфікований набір полів (дата, місце, опис, тип події). Перевагою є стабільність формату, але часто потрібна авторизація через токени.

Для Java також є окремий метод парсингу HTML-таблиць через Jsoup (для Java), не для динамічних сторінок, які містять HTML таблиці, є зручним інструментом бібліотека Jsoup. Вона дозволяє витягнути дані за CSS-селекторами (наприклад, table > tbody > tr > td) без запуску браузера. Переваги – швидкість та простота реалізації, особливо для структурованих реєстрів чи таблиць звітів.

Загалом, модуль парсера є критично важливою частиною OSINT-системи, бо саме він формує вхідні дані для подальшої фільтрації, класифікації та перевірки достовірності.

Отже парсер має збирати інформацію та формувати вхідні дані, передаючи їх на модуль, що виконуватиме основну роботу над даними.

3.2 Розробка функціональних та нефункціональних вимог

3.2.1 Функціональні вимоги.

Функціональні вимоги OSINT-системи визначають функції та послідовність дій, необхідних для збору, фільтрації, класифікації та обробки відкритих даних. Процес формування вимог до системи, та кращого відображення взаємодії користувача з ключовими функціями системи зображено на рисунку 3.2.1.1

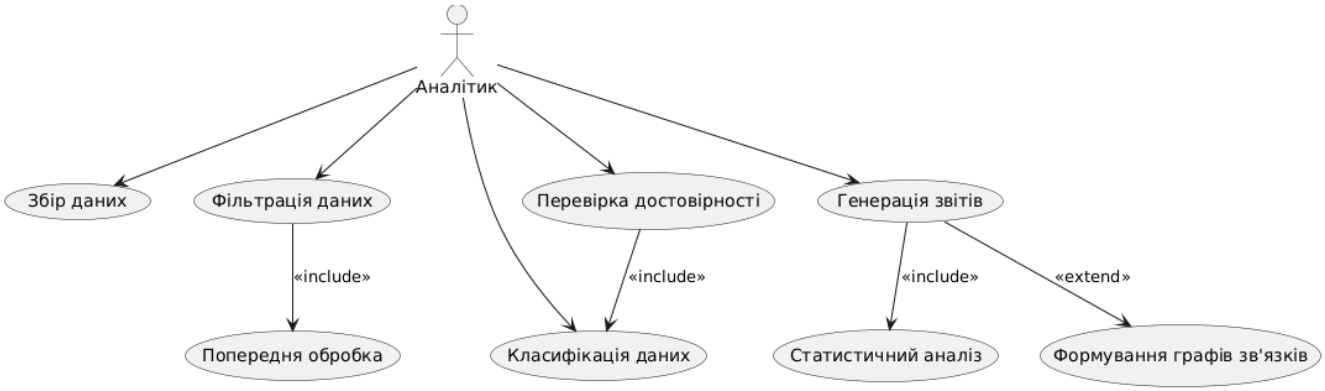


Рисунок 3.2.1.1 – Use Case діаграма взаємодії користувача з системою

Основні модулі та їх можливі реалізації повинні включати збір даних, фільтрацію, класифікацію та обробку.

Загалом, типова архітектура OSINT-прототипу формулюється як конвеєр з етапами «збір→фільтрація→класифікація→аналіз», що узгоджується з відомими підходами до обробки інформаційних потоків.

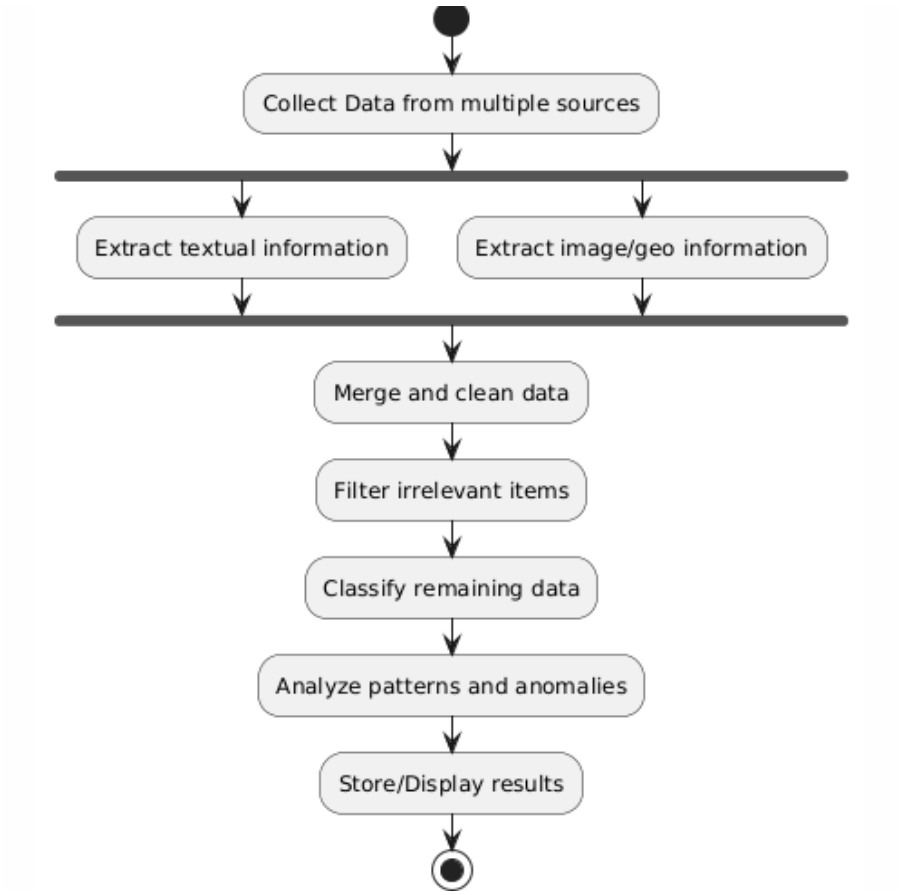


Рисунок 3.2.1.2 – Діаграма послідовності обробки даних у системі збору OSINT

Такий ‘’конвеєр’’ можна реалізувати послідовними мікросервісами або агентами, що обмінюються даними через API чи черги повідомлень (наприклад, RabbitMQ, Kafka).

Для збору даних широко використовується веб-скрапінг (бібліотеки Scrapy, BeautifulSoup, Selenium), API соціальних мереж і пошукових систем (Twitter API, Google Custom Search API) та RSS-стрічки для автоматичного збору інформації [14].

Сюди належать також спеціалізовані OSINT-сервіси (Maltego, Shodan тощо) та ручні пошукові методи. До переваг відносять здатність швидко збирати великі обсяги публічних даних, недоліком – необхідність обходу обмежень доступу (CAPTCHA, ліміти API, зміни структури сайтів), можливі юридичні обмеження (наприклад, поважання robots.txt) [20].

Фільтрація проводиться за різними критеріями – ключовими словами, мовою, датою, джерелами тощо. Простим підходом є пошук за запитами, фільтрація за ними, складнішими ж - ML-моделі для виявлення нерелевантного чи неприйнятного контенту (наприклад, фільтр ненормативної лексики або спаму).

Сервіси ELK Stack (Elasticsearch + Kibana) можуть використовуватися для індексації і фільтрації за запитами. Як перевага – швидке відсіювання великої кількості шуму, недолік – ризик втратити релевантну інформацію при надто грубих фільтрах.

Далі за суть класифікації. Мета - призначити отриманим даним категорії за змістом чи типом. Можна застосовувати правила за ключовими словами, або машинне навчання через ШІ, SVM, сучасні NLP-моделі (GPT) тощо.
Прикладами є класифікація новин за темами або ідентифікація загроз у текстах форумів за ключовими словами.

Наївний Байєс (Naive Bayes) – ймовірнісний класифікатор, який на основі моделі умовних незалежностей обчислює ймовірність приналежності тексту до певного класу (наприклад, «фейк» або «реальна новина»). В основі лежить формула Баєса:

$$P(c \mid x) = \frac{P(x|c)\,P(c)}{P(x)} \quad (3.1)$$

Де P(x) – набір ознак;

P(c) – клас події.

Такий підхід часто використовується для текстової класифікації (наприклад, спам-фільтри) завдяки простоті й швидкості розрахунку.

Логістична регресія – бінарний класифікатор, де ймовірність належності документа до відповідного класу задається логістичною функцією:

$$\sigma(z) = \frac{1}{1+e^{-z}}, z = w^{\mathrm{T}}x + b \text{ (3.2)}$$

Де сигма(z) – логістична функція;

x – вхідний вектор ознак;

w – ваги;

b-зсув.

Під час навчання визначаються параметри, які максимізують імовірність правильного передбачення. Цей метод дає інтерпретовані оцінки (ймовірності) та часто використовується як базовий класифікатор.

Формули LSTM (Long Short-Term Memory) – різновиду рекурентної нейронної мережі для роботи з послідовними даними (текст, часові ряди).

$$f_t = \sigma\big(W_f[h_{t-1}, x_t] + b_f\big), i_t = \sigma(W_i[h_{t-1}, x_t] + b_i), o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \text{ (3.3)}$$

$$\widetilde{C_t} = \tan h\,(W_c[h_{t-1}, x_t] + b_c) \text{ (3.4)}$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \widetilde{C_t}, h_t = o_t \odot \tan h\,(C_t) \text{ (3.5)}$$

Де x_t- вхід у мережу на момент часу t;

h_{t-1}- прихований стан попереднього кроку, тобто з моменту часу t – 1;

f_t – функція, що вирішує, яку частину минулого стану забути;

i_t – сигнал входу, визначає, які нові значення потрібно додати до комірки пам’яті;

o_t- сигнал виходу, визначаючий, що буде виведено з комірки пам’яті;

$\widetilde{C_t}$- проміжне значення, що може бути додане до стану;

C_t- значення стану комірки пам’яті на момент t;

C_{t-1}- стан комірки пам’яті попереднього кроку;

h_t- вихідний вектор на момент часу t;

$W_{(f,i,o,c)}$- вагові матриці для відповідних гейтів та кандидатів;

$b_{(i,f,o,c)}$- відповідні зміщення;

σ- сигма функція активації, обмежує знеачення між 0 і 1;

$\tan h$- тангенс гіперболи, що масштабує значення між -1 і 1;

$\odot$ - поелементне множення.

Він здатен запам’ятовувати довгі залежності, що корисно для розпізнавання тексту новин.

LSTM моделі застосовують до аналізу послідовності слів у новинах і можуть відрізняти реальну хронологію подій від випадкового набору фраз.

Перевага такого підходу це гнучкість і точність при достатньому тренуванні моделі, але велика потреба в розміченому наборі даних та ризик випадкових хибних класифікацій, що викликають ‘’незрозумілість’’ висновків.

Для прикладу можна використовувати фреймворки scikit-learn, TensorFlow/PyTorch чи готові NLP-бібліотеки (spaCy, NLTK) для реалізації моделей.

У задачах OSINT та обробки текстової інформації часто використовуються різні метрики для оцінки якості класифікаційних моделей: точність (accuracy), прецизійність (precision), повнота (recall), F1-міра тощо. Для порівняння моделей застосовують бар-діаграми або лінійні графіки.

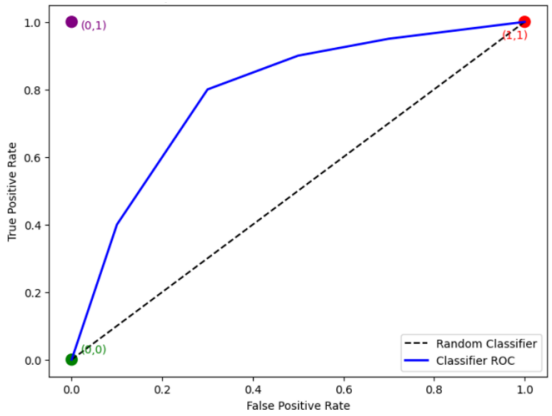


Рисунок 3.2.1.3 – Приклад ROC-кривої класифікаційної моделі.

Ми бачимо абсцис (FPR) та ординат (TPR), які демонструють зміну показників при різних порогах від 0 до 1. Ідеальний класифікатор («прапор над» діаграми) мав би TPR=1 при FPR=0.

Особливо для бінарних задач із незбалансованими класами використовуються precision-recall(влучність та повнота) криві. Ці графіки з осею абсцис – повнота та ординат – влучність ілюструє компроміс між виявленням усіх позитивних та малою кількістю хибно позитивних.

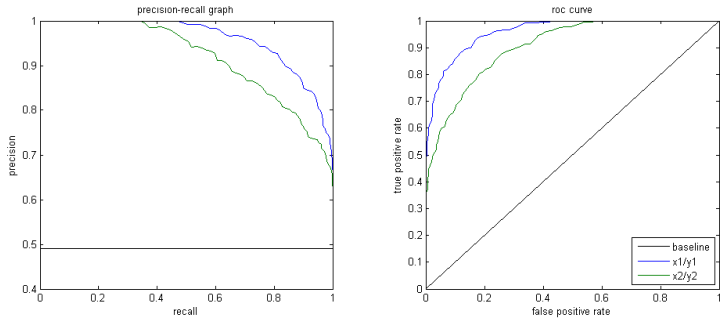


Рисунок 3.2.1.4 - Приклад оцінки якості класифікації подій на основі Precision-Recall та ROC-кривих

Обробкою та підготовкою даних, а саме операціями очищення даних (видалення пропусків, дублікатів, нормалізація тексту та приведення її в потрібний формат для роботи з даними) можуть займатися інструменти імпорту та експорту CSV, в Java це Apache, або інші сервіси типу Hadoop для великих обсягів, OpenRefine для візуального очищення.

Наприклад, Pandas надає методи для знаходження і видалення пропусків чи дублікатів.

Підвищення якості даних є перевагою для подальшого спрощення аналітики. Але з недоліків – витрати ресурсів на обробку даних при особливо масштабних розмірів файлів, що потребує розподілених рішень.

3.2.2 Нефункціональні вимоги.

Для визначення меж нашого проекту, якісні характеристики системи та обмеження її роботи, потрібно визначити нефункціональні вимоги.

Для OSINT-прототипу важливими є: performance, scalability, reliability, usability, security.

Продуктивність (performance): швидкість збору і обробки потоків даних (частина за секунду/денний обсяг). Враховується необхідність обробляти великі набори текстів/медіа в реальному часі. При високих вимогах рекомендують асинхронні та паралельні рішення (async-запити, багатопотоковість) або розподілені системи (Hadoop, Spark). Недолік масштабності – збільшення часу відгуку.

Масштабованість (scalability): здатність системи ефективно працювати із збільшенням обсягу даних або кількості джерел. Наприклад, мікросервісна архітектура з Docker/Kubernetes дозволяє «горизонтальне» масштабування. Недолік – складність розгортання та супроводу.

Надійність і стійкість (reliability, fault tolerance): мінімізація простоїв і втрат даних у разі збоїв мережі чи апаратури. Забезпечується резервуванням компонентів, резервним копіюванням, чергами повідомлень для запобігання втраті даних. Наприклад, система повинна продовжувати роботу навіть при відмові окремого агенту збору.

Безпека та законодавчі обмеження: хоч OSINT і працює з відкритими джерелами, необхідно дотримуватися умов доступу (захист облікових даних API, обмеження за частотою запитів) та норм GDPR/захисту ПДВ. Використання проксі і VPN може забезпечити анонімність запитів. Недолік – додаткові ресурси на впровадження безпеки.

Юзабіліті та підтримка (usability, maintainability): зрозумілий інтерфейс або API для користувача/аналітика, модульний код для легкого оновлення та розширення (додавання нових джерел або моделей). Це знижує вартість супроводу, але може потребувати додаткового проектування (конфігураційні файли, документація).

Відповідно до принципів інженерії вимог, нефункціональні властивості повинні бути чітко задокументовані (наприклад, максимальна затримка відповіді – 1 секунда) та перевірені під час тестування. В нашому випадку,

використовуючі безкоштовні сервіси, затримка відповіді має бути більшою, для влаштовування обмежень моделі (Наприклад: 30 запитів на хвилину, тощо). Узагальнення цих вимог допомагає обрати оптимальні технології та архітектуру на практиці.

3.3 Опис архітектури реалізованої системи.

Розроблений прототип є консольною Java-програмою, основною метою якої є автоматизоване перетворення текстових матеріалі (судових рішень) на структуровану інформацію, придатну для аналітичного дослідження. Архітектура реалізованої системи відповідає принципам модульності та розділення обов'язків між компонентами. Це дає змогу гнучко масштабувати рішення, оновлювати окремі функціональні блоки без зміни загальної логіки обробки.

Загалом архітектуру можна описати як багатошаровий програмний каркас, що охоплює відповідні компоненти, такі як: Модуль зчитування вхідних даних, модуль взаємодії з API (Groq), модуль парсингу результатів, збереження даних та конфігураційний шар.

Модуль зчитування вхідних даних відповідає за обробку файлів формату CSV або JSON, які містять сирі текстові фрагменти, наприклад, судових рішень з Єдиного реєстру судових рішень України (будь-яка обрана область або архів всіх судових рішень та постанов за відповідний рік). Програма читає записи, у яких фіксується лише частина даних, що не підлягає під формат для взаємодії з API, та надає файлу належну структуру або розмітку перед самою роботою.

Цей модуль реалізований як утилітний клас із функціональністю зчитування рядків та передавання їх до обробки.

Модуль взаємодії з API (Groq) є центральною складовою логіки обробки. Саме тут відбувається виклик зовнішнього API через HTTP-запит із передачею тексту про аналіз, як запиту до мовної моделі. Текстове повідомлення, сформоване з вхідних даних, передається до Groq API, яке генерує

структуровану відповідь, після ця відповідь парситься та зберігається у проміжній моделі.

Парсинг результатів відповідає за вилучення ключових елементів з отриманої відповіді, зокрема: тип подіїх, місце, дата, правопорушники, об’єкти правопорушення тощо. Розбір відповіді реалізовано з урахуванням того, що результат надходить у форматі, подібному до JSON, але потребує додаткової обробки на кшталт відсікання службових символів або перевірки відповідності формату відповіді мовної моделі.

Реалізоване з’єднання з базою даних MySQL. Структурована інформація зберігається в реляційній таблиці, забезпечуючи можливість зберігати результати обробки з прив’язкою до оригінального запису CSV, збереження унікальних ідентифікаторів, а також підтримку подальших аналітичних запитів до бази. А конфігураційний шар містить необхідні параметри підключення до бази даних, API-ключі, шлях до файлів судового реєстру та структуру запитів з фільтраційними критеріями та логіку повторного надсилання запитів у разі помилки. Це дозволяє переналаштувати логіку програми без зміни основного коду, що дозволяє модифікувати архітектуру під різні задачі та роботу з різними джерелами.

Таким чином, програмна архітектура реалізованого рішення є прикладом pipeline моделі, де вхідні дані послідовно проходять етапи: зчитування → обробка через API → парсинг → збереження. Всі модулі реалізовано як незалежні класи або функції з чітким призначенням, що відповідає принципам високої когезії та низької зв’язаності. У майбутньому це дозволяє легко адаптувати систему до обробки інших типів текстів, доповнювати модульну архітектуру новими модулями роботи з інформацією або перенести логіку у вебінтерфейс.

3.3.1 Використані технології

У реалізації прототипу використано перевірені та ефективні інструменти, що забезпечують зручність розробки, стабільну роботу програми та можливість масштабування в подальшому.

Технологічний стек був підібраний з урахуванням вимог до обробки текстових даних, інтеграції з зовнішніми API та збереження інформації у структурованому вигляді. Спершу, мовою програмування була обрана Java замість потенційного Python, адже має широкі можливості для обробки текстових даних, багату екосистему бібліотек і стабільну роботу в серверному середовищі. Java забезпечує достатню продуктивність при роботі з файлами та об’єктами, а також підтримку сучасних концепцій на кшталт потокової обробки, лямбда-виразів.

Формат вхідних даних: було розрахунок на CSV як на стандартний табличний формат, зручний для зберігання великої кількості однотипних записів, а також цей табличний формат використовувався в архівах Єдиного державного судового реєстру. У програмі реалізовано зчитування CSV з урахуванням специфіки структури — обробка символів табуляції, лапок, роздільників. Для парсингу застосовано стандартні засоби Java: BufferedReader, String.split(), з можливістю розширення через зовнішні бібліотеки.

Зв’язок з Groq API є зовнішнім мовним сервісом, отримуючим текстовий запит і повертаючи змістовану структуровану відповідь. Сам Groq було обрано для прототипу через його безкоштовні модулі з невеликим лімітом в 30 запитів на 3 хвилини, в той час як інші компанії давали таку кількість запитів на добу.

Запити надсилаються через HTTP, а автентифікація виконується за токеном, зазначеним у конфігурації. Формат відповіді підлягає обробці регулярними виразами або парсерами, залежно від складності шаблону.

База даних MySQL для збереження структурованих результатів у вигляді реляційних таблиць. Для ключових полів, з якими працює API, реалізовано таблиці тексту запиту, отриманої відповіді, категорії події, часу

фіксації та службової інформації такої як ідентифікаційний номер події та її статус обробки.

Для взаємодії з БД використано стандартний JDBC-драйвер. Запити реалізовано у вигляді SQL-інструкцій на стороні Java, із попередньою перевіркою з’єднання та обробкою помилок.

Фреймворк для логування SLF4J + Logback забезпечено централізоване логування процесів, запитів, перевірок тощо. Це дозволяє вести журнал обробки, здійснювати налагодження та забезпечує прозорість виконання програми.

Конфігураційний файл з критичними параметрами, шляхом до CSV, API ключом та базою даних внесено в окремий файл налаштувань. Використання цього технологічного стека дозволило реалізувати працездатний прототип з можливим контролем кожного етапу обробки — від імпорту даних до збереження результатів. Система легко адаптується до інших типів задач, залишається модульною, зберігаючи при цьому простоту налаштування та стабільність виконання.

3.3.2 Технології для використання

Для реалізації системи обрали сучасні програмні платформи та бібліотеки що забезпечують ефективний збір і аналіз даних. Зокрема, за основу певних модулів можна взяти мову Python як просту та універсальну, яка широко застосовується для створення OSINT-інструментів. Python добре підходить для веб-скрапінгу, обробки тексту та машинного навчання, тому вибір цієї мови дозволить швидко реалізувати функціональність. Для реалізації потенційного веб-інтерфейсу можна скористатися фреймворками, для творення RESTful API на Java можна використати Spring Boot, що дозволить зовнішнім системам надсилати дані на обробку, отримувати результати та керувати запитами через браузер або інші програми.

На основі JavaScript-фреймворку, таких як React або Vue.js, які забезпечують зручне введення, візуалізацію результатів обробки, фільтрацію по подіях, локаціях, датах та інше. Це зробить систему доступною для

нефахових користувачів, та дасть можливість аналізувати сам проект або використати його в аналітичних цілях.

Система пошуку на основі Apache Lucene — для забезпечення повнотекстового пошуку по сутностях, подіях, іменах або ключових словах, що не реалізується ефективно в стандартній реляційній БД. Що б прив’язати події до карти.

Також, для географічної прив’язки подій або їх відстеження за мапою є відповідні бібліотеки для геолокації, такі як GeoTools на Java або інтеграція з зовнішніми API

Існуючих мап (OpenStreetMap, Google Maps).

Для масового імпорту або експорту результатів CSV формату достатньо, можлива підтримка XML в разі інтеграції напряму з державними реєстрами або юридичними системами. Проте, для структурованого передавання результатів через API переважно використовується JSON.

Також проект має включати в себе засоби безпеки для обмеження доступу до сервісу та захист API-запитів від несанкціонованого використання, адже це не тільки рятує систему та її дані, а ще й може зберегти потенційні витрати в повноцінному проекті з не локально розгорнутою API, для зняття обмежень з якої потрібно витратити відповідні кошти.

Для підвищення якості та надійності коду можливе підключення модулів тестування — наприклад, JUnit для модульного тестування логіки обробки, Mockito для створення тестових залежностей, а також Postman для інтеграційного тестування REST API в разі реалізації веб-сервісу.

Такі засоби дозволять верифікувати функціональність системи на різних рівнях: від окремих модулів до повного циклу обробки даних.

У подальшому система може адаптуватись до обробки інших форматів файлів крім CSV, а й інших джерел: підключення до відкритих REST API новинних сайтів, агрегаторів, RSS-стрічок тощо. За допомогою бібліотек Java для HTTP запитів можна реалізувати динамічний збір інформації у реальному часі, для цього також знадобляться парсери JSON.

Таким чином, обрана технологічна база забезпечує міцний фундамент для роботи з текстовими даними, а також створює умови для гнучкого доповнення та інтеграції нових можливостей типу геоаналітики, вебінтерфейсу та інтелектуального пошуку.

Реалізовний підхід відповідає сучасним тенденціям у сфері OSINT-розробки, де ключовими цінностями є масштабованість, автоматизація та структуроване представлення даних.

3.3.3 Можливості масштабування архітектури

У поточному вигляді прототип є лише локальним застосунком, який послідовно обробляє текстові дані та зберігає результати в базу даних MySQL. Такий підхід зручний для початкового аналізу, розробки та налагодження алгоритмів, проте система може бути масштабована та розширена у кількох напрямах.

Спочатку, у разі збільшення обсягів вхідних даних необхідно реалізувати паралельну обробку запитів. Поточна архітектура допускає розподілення обробки файлів на декілька потоків або окремих інстансів програми, кожен з яких може працювати з частиною даних, формуючи свої окремі запити, що дає змогу зменшити час на загальну обробку. У випадку використання брокера повідомлень, типу Apache Kafka або RabbitMQ можна реалізувати асинхронну модель обробки, де кожен новий фрагмент даних потрапляє в чергу, а незалежні обробники виконують виклики до API та запис результатів у базу. Цей підхід вже значно підвищує продуктивність системи при великій кількості запитів.

Є можливість інкапсуляції у мікросервіси окремих компонентів системи, таких як обробка запитів до мовної моделі, парсинг відповідей або збереження даних в бази. Вони можуть бути винесені у самостійні мікросервіси. Кожен сервіс реалізує вузьку функцію, має власний API, конфіг та масштабування.

Приклади: умовний сервіс text-parser — приймає текст і повертає розпізнану структуру, event-writer — зберігає дані до БД, classifier — аналізує

і класифікує тип події. Такий підхід підвищує надійність системи, адже у разі помилки одного компонента не ламається вся архітектура, та інші лишаються працездатними, а також цей підхід дозволяє масштабувати критичні частини незалежно.

В разі використання проекту та розширення прототипу слід задуматись про розгортання в хмарному середовищі для зниження навантаження на локальні ресурси та забезпечення стійкості до збоїв. Виклики до API можуть здійснюватися з серверних функцій, зберігання даних на хмарних SQL, а логіка маршрутизації на Kubernetes-кластерах. Це дасть змогу ефективно реагувати на зростання кількості користувачів або навантаження без ручного втручання в разі потреби. Хмарна інфраструктура нехай є більш вразливою ніж локально розгорнута, але дозволяє інтегрувати зовнішні сервіси та взаємодіяти з ними без необхідності розгортання додаткового ПЗ на своїх серверах.

Також проект не можна лишати без візуалізації та інтерфейсу.

Для забезпечення доступу до результатів обробки стороннім користувачам треба заздалегіть передбачити створення вебінтерфейсу. Його реалізація цілком можлива за допомогою Spring Boot для прикладу, фронтенду React або Vue.js для інтерактивного перегляду подій, їх пошуку або фільтрації.

Візуалізовані геоданні на карті через Google Maps, та умовні діаграми на Chart.js або Apache Superset для підвищення інформативності в звітах через графіки, діаграми тощо.

Реалізована архітектура системи вже закладає основу для подальшого масштабування, її модульність дає одразу декілька напрямків розширення та адаптації під різні задачі, тому розширення функціональності у відповідності до потреб сучасної аналітики безпеки є лише питанням часу.

3.4 Обґрунтування вибору платформи розробки та інструментальних засобів

3.4.1 Вибір мови програмування та архітектурного підходу

Мова програмування Java була обрана з урахуванням її стабільності, потужної стандартної бібліотеки, широкої підтримки сторонніх інструментів і звісно наявності вже перевірених рішень для інтеграції з базами даних, обробки файлів, побудови запитів та парсингу. Крім того, Java дає достатній рівень контролю над потоками виконання, що є корисним у перспективі переходу на паралельну або асинхронну обробку запитів в випадку розширення функціоналу проекту.

Архітектурно прототип реалізований як консольний застосунок із чітко визначеними модулями: зчитування даних, виклик мовної моделі, аналіз відповіді та збереження результатів. Такий поділ дає змогу підтримувати розширюваність і підвищує модульність та гнучкість структури.

3.4.2 Інтеграція з мовною моделлю через Groq API

Вибір пав на Groq API. Одним із ключових рішень стало використання зовнішнього API для обробки природньої мови. У реалізації застосовано прямий виклик Groq моделі через HTTP-запит з формуванням запитів і обробкою відповідей у форматі JSON.

Для тестування прототипу це забезпечує прозору логіку взаємодії із зовнішньою службою, дозволяє гнучко керувати параметрами запитів і зменшує залежність від сторонніх SDK або платформ. Також це спрощує масштабування системи для подальшого її розширення. Простота реалізації не суперечить вимогам до стабільності й точності.

3.4.3 Робота з джерелами даних

Формат CSV є стандартним, загальноприйнятим при зберіганні масивів правових або статистичних даних. Бібліотека OpenCSV(Java) забезпечує стабільне зчитування файлів незалежно від їхнього розміру чи формату, проте

файли для обробки та HTTP-запиту до зовнішнього API все ж таки обробляються під єдиний формат для спрощення обробки та занесення їх в БД. Загалом, CSV як основне джерело зумовлено не лише зручністю, а й відповідністю реальним умовам: масиви судових рішень, що підлягають обробці, на практиці часто надаються саме в такому вигляді. Це дозволило обійтись без складних парсерів, зосередившись безпосередньо на семантичному аналізі вмісту.

3.4.4 Збереження результатів та обґрунтування вибору СУБД

MySQL забезпечує транзакційну надійність, структуроване зберігання даних, можливість виконання складних запитів(що є корисно в роботі з кримінальними подіями), а також фільтрація та агрегацію результатів. Основна таблиця містить поля для збереження як первинного тексту, так і ключових параметрів, отриманих внаслідок обробки.

Також база підтримує розширення — у вигляді додаткових таблиць (наприклад, для сутностей, геоприв’язки, логів обробки), що дозволяє масштабувати систему без зміни логіки основного модуля.

3.4.5 Гнучкість реалізації

Винесені дані до зовнішнього конфігураційного файлу дозволяють розгорнути програму в різних середовищах без перекомпіляції.

Код має модульну структуру: класи обробки, зчитування, збереження в базу даних не перетинаються логікою і можуть бути протестовані окремо. Це робить модернізацію системи простіше та дозволить інтегрувати додаткові функції. Таким чином, вибір Java як платформи, Groq API як інструменту семантичного аналізу, MySQL як системи зберігання та CSV як джерела даних сформував збалансовану, гнучку та практично орієнтовану архітектуру, яка відповідає як поточним, так і майбутнім вимогам до OSINT-систем обробки кримінальних подій.

3.5 Тестування прототипу

3.5.1 Мета та підхід до тестування

Тестування реалізованого прототипу мало на меті перевірку її працездатності, стабільності при обробці вхідних даних, коректності взаємодії з зовнішнім API, правильності структурування отриманих результатів та їх збереження у базу даних. Особлива увага приділялась сценаріям циклу зчитування →відправка запитів →аналіз відповідей → збереження в БД.

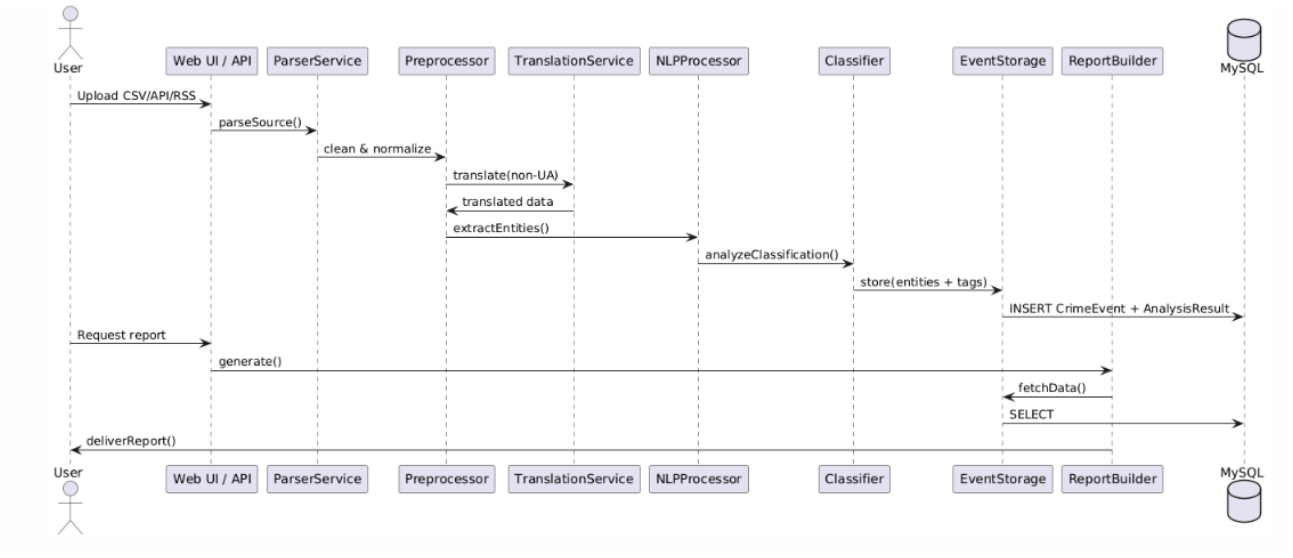


Рисунок 3.5.1 – Діаграма повного процесу обробки події

Тестувати будемо в межах загального процесу, не ізольовано, починаючи з імпорту тексту і завершуючи успішною вставкою запису в таблицю, для доцільного тестування систем, що виконують послідовну обробку даних.

3.5.2 Перевірка коректності зчитування CSV-Файлів

Першим етапом стало тестування модуля обробки вхідних файлів. Було завантажено тестову вибірку з файлів Єдиного реєстру судових рішень України, які містили як валідні тексти судових рішень, так і пусті записи. Вигляд категорії справ цих даних показано на рисунку 3.4.2.1

Код категорії	Назва
2036	Захоплення заручників
3017	звільнення з публічної служби
4095	Землекористування
2041	Зловживання військовою службовою особою владою або службовим становищем (стара категорія)
2076	Зловживання владою або службовим становищем
2001	Злочини проти життя та здоров'я особи
2004	Злочини у сфері господарської діяльності
2012	Злочини проти авторитету органів державної влади, органів місцевого самоврядування, об'єднань громадян та злочини проти журналістів
2016	Злочини проти безпеки виробництва
2017	Злочини проти безпеки руху та експлуатації транспорту
2028	Злочини проти власності
2035	Злочини проти волі, честі та гідності особи
2039	Злочини проти встановленого порядку несення військової служби (військові злочини)
2054	Злочини проти громадської безпеки
2051	Злочини проти громадського порядку та моральності
2060	Злочини проти миру, безпеки людства та міжнародного правопорядку

Рисунок 3.5.2.1 – Фрагменти категорії справ судових рішень

Для забезпечення стабільної обробки вхідних CSV-файлів та пізнішої взаємодії з API, файли у випадок пошкодження структури або її відсутності мають пройти через допоміжний модуль CSVFixer, що переробить файл в формат, здатний взаємодіяти з API. Його завданням є попереднє очищення та вирівнювання структури файлу перед його передачею в основний модуль обробки.

Рисунок 3.5.2.2 – Отримання обробленого файлу через CSV-Fixer

3.5.3 Тестування запитів до Groq API

Наступнім кроком стала перевірка взаємодії з Groq API. У процесі тестування аналізувалися сформовані HTTP-запити, формат payload, обробка токенів а також відповідність отриманих відповідей очікуваному формату. У разі помилок, наприклад перевищення ліміту символів для безкоштовної LLM моделі або відсутності ключа API система належно фіксувала виняток і позначала запис необробленим, виводячи помилку в консоль, не зупиняючи загальний процес.

В звичайному ж випадку передбачено логування події та маркування помилки у відповідному полі бази даних.

Рисунок 3.5.3.1 - Вивід зчитаних рядків із CSV-файлу

Наступнім чином буде виглядати запит до Groq API:

```
private static String analyzeWithGroq(HttpClient client, String text) throws Exception {  1 usage   bublykcodes
    String prompt = "Оціни серйозність та можливі наслідки наступної події, а також надайте короткий аналіз: " + text;
    String requestBody = """
            {
                "model": "meta-llama/llama-4-scout-17b-16e-instruct",
                "messages": [
                    {
                        "role": "user",
                        "content": "%s"
                    }
                ],
                "max_tokens": 200
            }
            """.formatted(prompt.replace( target: "\"",  replacement: "\\\""));
```

Рисунок 3.5.3.1.1 – фрагмент коду запиту до Groq API

Перед надсиланням запитів до API парсер виконує задачу по обробці вказаної кількості рядків з документу, та після завантаження надсилає запит на обробку до Groq API.

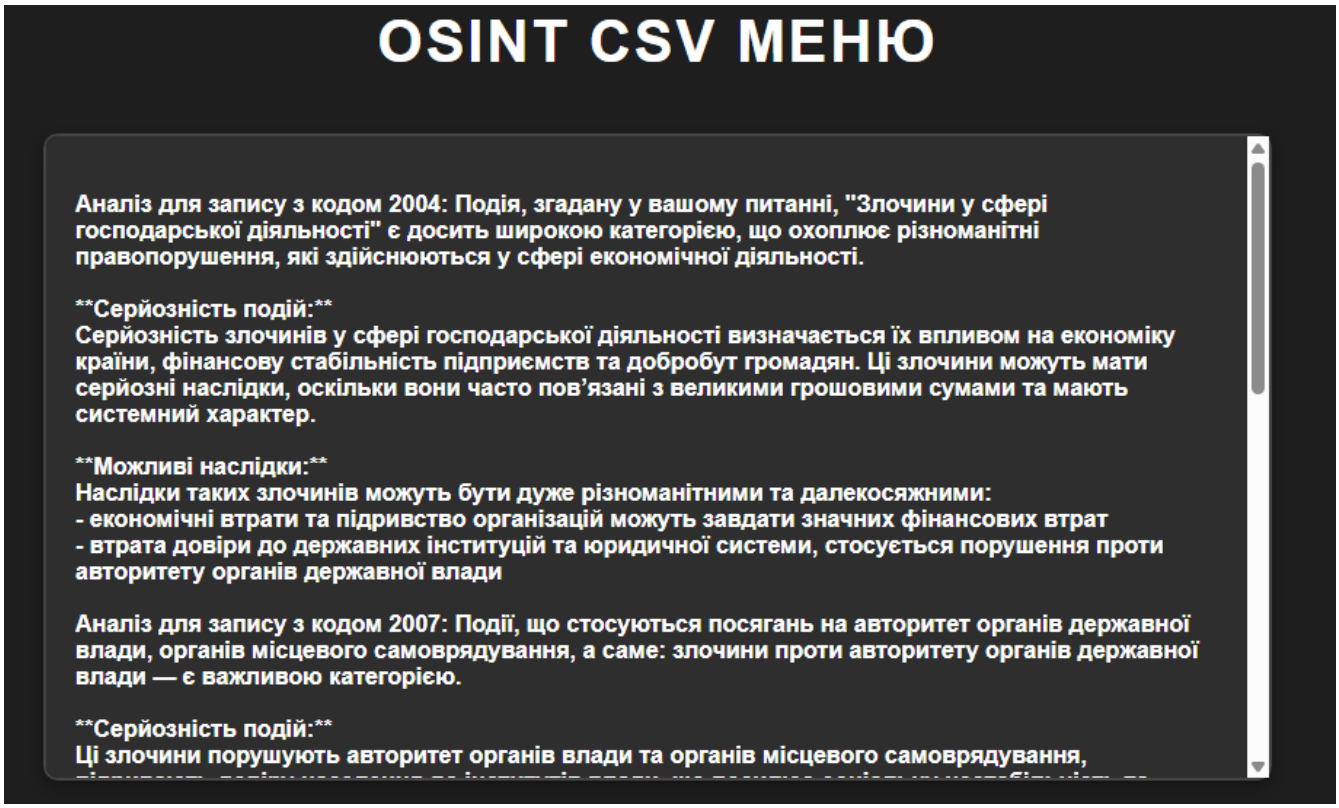
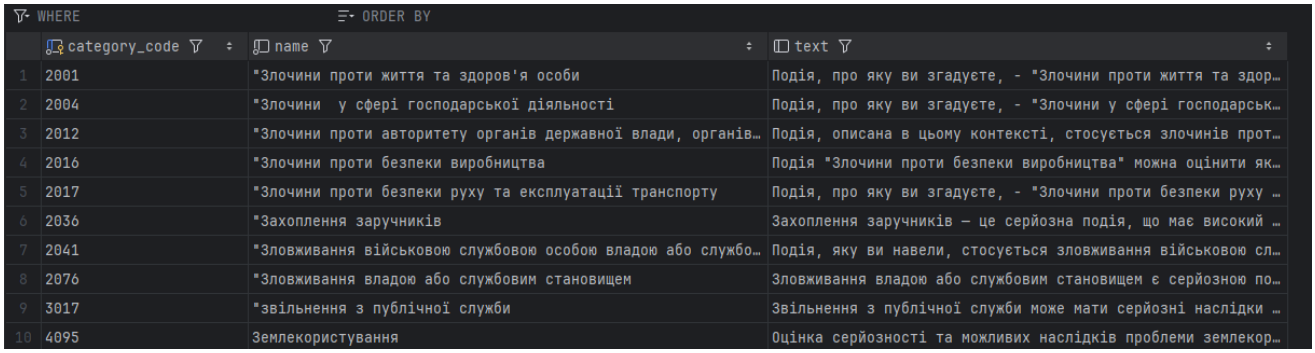


Рисунок 3.5.3.2 – Отримання відповіді від Groq API у вигляді JSON-подібного тексту

3.5.4 Перевірка збереження результатів у базу даних

Після успішного парсингу та отримання відповіді від API тестувалась функція запису результатів до реляційної бази даних. Усі структуровані поля зберігаються у відповідні колонки таблиці criminal_events, а також додається оригінальний текст і повна відповідь від моделі. При повторному запуску та повторній перевірці тих самих даних в разі невеликих змін система перезаписувала та вставляла запис знову в залежності від змінених параметрів.



WHERE ORDER BY

	category_code	name	text
1	2001	"Злочини проти життя та здоров'я особи	Подія, про яку ви згадуєте, - "Злочини проти життя та здор…
2	2004	"Злочини у сфері господарської діяльності	Подія, про яку ви згадуєте, - "Злочини у сфері господарськ…
3	2012	"Злочини проти авторитету органів державної влади, органів…	Подія, описана в цьому контексті, стосується злочинів прот…
4	2016	"Злочини проти безпеки виробництва	Подія "Злочини проти безпеки виробництва" можна оцінити як…
5	2017	"Злочини проти безпеки руху та експлуатації транспорту	Подія, про яку ви згадуєте, - "Злочини проти безпеки руху …
6	2036	"Захоплення заручників	Захоплення заручників — це серйозна подія, що має високий …
7	2041	"Зловживання військовою службовою особою владою або службо…	Подія, яку ви навели, стосується зловживання військовою сл…
8	2076	"Зловживання владою або службовим становищем	Зловживання владою або службовим становищем є серйозною по…
9	3017	"звільнення з публічної служби	Звільнення з публічної служби може мати серйозні наслідки …
10	4095	Землекористування	Оцінка серйозності та можливих наслідків проблеми землекор…

Рисунок 3.5.4.1 – Вміст таблиці MySQL після завершення обробки

Також окремо тестувались сценарії, що моделюють типові помилки: відсутність з’єднання, неправильний API-ключ, або порожній рядок у CSV. Система в усіх випадках не завершувала роботу аварійно, навпаки, фіксувала помилку в логах та переходила до наступнього запису. Це підтвердило стійкість логіки обробки до часткових збоїв, що критично для автоматизованих систем збору даних.

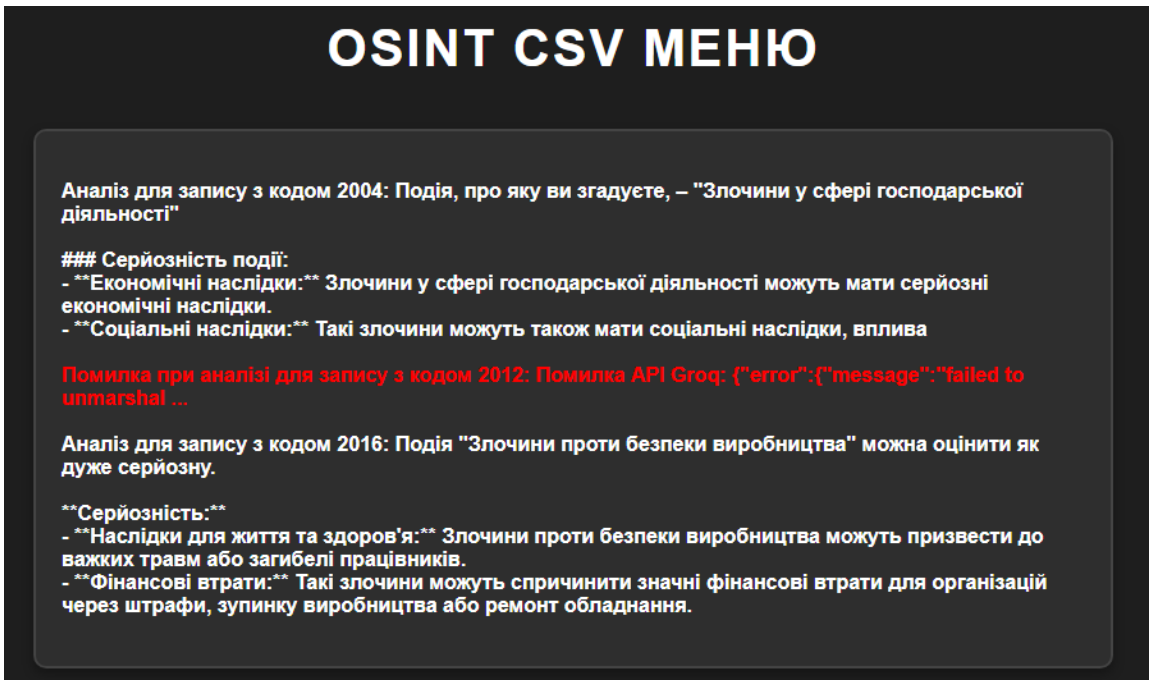


Рисунок 3.5.4.2 – Реакція системи на помилку взаємодії з пустим рядком CSV

Як ми бачимо, система продовжила далі працювати, та занесла пустий рядок в базу даних, після оновлення рядку з кодом ‘’2012’’ та спроби повторно занести його в базу даних, його було успішно занесено (див. Рисунок - Вміст таблиці MySQL після завершення обробки). Тестовий документ виглядає так, як на рисунку 3.5.4.2.1

Категорії справ

Код категорії	Назва	Текст
2036	Захоплення заручників	
3017	звільнення з публічної служби	
4095	Землекористування	
2041	Зловживання військовою службовою особою владою або службовим становищем (стара категорія)	
2076	Зловживання владою або службовим становищем	
2001	Злочини проти життя та здоров'я особи	
2004	Злочини у сфері господарської діяльності	
2012	!!!@@@###	Непридатний текст
2016	Злочини проти безпеки виробництва	
2017	Злочини проти безпеки руху та експлуатації транспорту	
2028	Злочини проти власності	
2035	Злочини проти волі, честі та гідності особи	
2039	Злочини проти встановленого порядку несення військової служби (військові злочини)	
2054	Злочини проти громадської безпеки	
2051	Злочини проти громадського порядку та моральності	
2060	Злочини проти миру, безпеки людства та міжнародного правопорядку	
2063	Злочини проти правосуддя	
2065	Злочини проти статевої свободи та статевої недоторканності особи	
2067	Злочини у сфері використання електронно-обчислювальних машин (комп'ютерів), систем та комп'ютерних мереж і мереж електрозв'язку	

Рисунок 3.5.4.2.1 – CSV документ з недопустимими символами в коді ‘’2012’’

Також рисунок 3.5.4.3 показує реакцію системи на неправильний ключ API, тобто відсутність зв’язку з моделлю.

Рисунок 3.5.4.3 – Реакція системи на помилку взаємодії з API, неправильний ключ

3.5.5 Узагальнення результатів тестування

Проведене тестування підтвердило працездатність прототипу, кожного з етапів системи в умовах реального завантаження. Основні критерії ефективності – стабільна обробка вхідних даних, коректне формування запитів, структурованість відповідей та успішне збереження результатів – були дотримані. Усі критичні помилки належним чином фіксувалися, проте, програма продовжувала обробку чергових записів.

Система успішно обробляє десятки записів за хвилину без затримок, враховуючи обмеження безкоштовної моделі LLM, демонструючи надійність при роботі з великими CSV-файлами, а також адаптивність до різної якості вхідних даних.

Всього було оброблено 2 тестових документи формату CSV по 466 окремих злочинів та 4103 рядків в кожному. На обробку запиту та надсилання відповіді LLM витрачає приблизно 3 секунди, враховуючи обмеження в 30 запитів на добу, безперервно безкоштовна модель може працювати 1,5 хвилини або 90 секунд. Всього, не враховуючи обмеження, модель обробила 932 запити за 46 хвилин з перервами на зняття обмежень.

ВИСНОВКИ

У ході виконання дослідження було досягнуто поставлену мету, та розроблено методичний підхід і функціональний модульний прототип системи, орієнтованої на автоматичний збір, попередню обробку та структурування даних про кримінальні події з відкритих джерел. А також, здатність прототипу до потенційного розширення функціоналу в різних напрямах.

Тема має високу актуальність у зв’язку із зростанням інформаційного навантаження на аналітичні служби, потребою в оперативному реагуванні на події а також складної ситуації в умовах війни та інформаційної гібридності.

Об’єктом дослідження стали процеси збору та обробки OSINT-даних із відкритих онлайн-ресурсів.

Ці джерела є різнорідними за структурою та змістом – від новинних сайтів і державних реєстрів до форумів і соціальних мереж, що є складністю перед обробкою без попереднього структурування. Основною проблемою є фрагментарність, недостовірність або нестабільність відкритої інформації, яка вимагає застосування спеціальних алгоритмів для її фільтрації та класифікації.

У наявних рішеннях під час аналізу було виявлено низку недоліків типу відсутності єдиного API до державних баз, захищенність реєстрів від автоматизації (CAPTCHA, обмеження частоти доступу), слабка структурованість даних у відкритих джерелах, обмежена можливість перевірки достовірності повідомлень. Проте, запропоновано шляхи подолання цих проблем, за рахунок модульної архітектури та використання різних засобів парсингу в залежності від поставленої задачі, а також використання сервісів машинного навчання для класифікації контенту та виявлення фейкових повідомлень.

Розроблений програмний прототип, наразі, має реалізацію консольного застосунка з модульною архітектурою, що складається з окремих модулів:

імпорту файлів, виклику зовнішнього API, парсингу відповідей та збереження результатів до реляційної бази даних.

Система здатна обробляти масиви текстів, витягувати ключові елементи подій а також відмічати випадки помилок чи нечітких відповідей, при цьому, не зупиняючи свою роботу. Така автоматизована обробка забезпечує базовий рівень структурованості вхідних текстів, які у своєму вигляді є непридатними для прямого аналітичного використання, маючи різний формат в залежності від джерела інформації.

Після тестування, можемо зробити висновок, що система стабільно функціонує на реальних або змодельованих наборах даних, коректно фільтрує порожні або пошкоджені записи, є стійкою до відмов зовнішнього API та здатна фіксувати виняткові ситуації. За допомогою допоміжних утиліт, які підганяли неструктурований файл під структуру для роботи з API вдалося підвищити ефективність, стабільність та надійність обробки нестандартних або зруйнованих джерел.

Це є надзвичайно важливим для практичних умов реального використання, де якість даних не гарантована зовнішніми джерелами.

Прототип є модульним та демонструє свою високу гнучкість – її можна масштабувати на більші обсяги даних, адаптувати під різні задачі та інші джерела інформації, інтегрувати до більш комплексних платформ OSINT. Структура програмного коду дозволяє реалізувати локальні рішення та хмарні або розподілені системи збору й оцінки нових інцидентів у режимі реального часу.

Результати дослідження мають вагоме практичне значення для автоматизації процесів OSINT-розвідки, зокрема в секторі правопорядку, медіамоніторингу, аналізу кримінальних подій та журналістських розслідувань, для забезпечення миттєвого реагування.

Подальший розвиток цієї системи може включати розширення на мультимедійні дані, покращення моделей класифікації, інтеграцію геопростору та відповідних сервісів і реалізацію веб-інтерфейсу для не

технічних користувачів. Таким чином, розробка підтверджує свою актуальність, корисність і прикладну цінність у сфері інформаційної безпеки та кримінального аналізу.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Національна поліція України. “Сайт НПУ (Національної поліції України)” // mvs.gov.ua. URL: https://mvs.gov.ua/contacts/national-police-ukraine (Дата звернення: 22.03.2025)
2. Офіс Генерального прокурора України. “Офіційний сайт” // gp.gov.ua. URL: https://gp.gov.ua (Дата звернення: 22.03.2025)
3. Головне управління ДСНС у Львівській області. “Офіційний сайт” // lv.dsns.gov.ua. URL: https://lv.dsns.gov.ua (Дата звернення: 22.03.2025)
4. Львівська обласна державна адміністрація. “Офіційний сайт ЛОДА” // loda.gov.ua. URL: https://loda.gov.ua (Дата звернення: 22.03.2025)
5. Львівська міська рада. “Офіційний портал міста” // city-adm.lviv.ua. URL: https://city-adm.lviv.ua (Дата звернення: 22.03.2025)
6. Служба безпеки України. “Сайт СБУ” // ssu.gov.ua. URL: https://ssu.gov.ua (Дата звернення: 22.03.2025)
7. ТСН. “Телевізійна служба новин” // tsn.ua. URL: https://tsn.ua (Дата звернення: 22.03.2025)
8. Українська правда. “Новини України” // pravda.com.ua. URL: https://www.pravda.com.ua (Дата звернення: 22.03.2025)
9. УНІАН. “Інформаційне агентство УНІАН” // unian.ua. URL: https://www.unian.ua (Дата звернення: 22.03.2025)
10. BBC News Україна. “BBC Українською” // bbc.com. URL: https://www.bbc.com/ukrainian (Дата звернення: 22.03.2025)
11. Єдиний державний реєстр судових рішень. “Судовий реєстр України” reyestr.court.gov.ua. URL: https://reyestr.court.gov.ua/ (Дата звернення: 22.03.2025)
12. DeepState. “Інтерактивна карта бойових дій” // deepstatemap.live. URL: https://deepstatemap.live/#6/49.4383200/32.0526800 (Дата звернення: 29.05.2025)

13. The Cyber Huntress. “10 Python OSINT Tools for Investigating Disinformation and Extremism” // Medium. URL: https://medium.com/@thecyberhuntress/10-python-osint-tools-for-investigating-disinformation-and-extremism-c83b2f33fee6 (Дата звернення: 29.04.2025)
14. Bazzell, M. “Open Source Intelligence Techniques” // Self-Published. 2021. 9th Edition. ISBN: 979-8594564633.
15. Oracle. “The Java™ Tutorials” // Oracle Docs. URL: https://docs.oracle.com/javase/tutorial/ (Дата звернення: 26.04.2025)
16. GeeksForGeeks. “Java Programming Language – Tutorials and Practice” // geeksforgeeks.org. URL: https://www.geeksforgeeks.org/java/ (Дата звернення: 25.04.2025)
17. Baeldung. “Learn Java and Spring Online” // baeldung.com. URL: https://www.baeldung.com/ (Дата звернення: 22.04.2025)
18. Вікіпедія. “Розвідка на основі відкритих джерел (OSINT)” // Wikipedia. URL: https://uk.wikipedia.org/wiki/Розвідка_на_основі_відкритих_джерел (Дата звернення: 22.03.2025)
19. Фіглузі, Ф. “Система ФБР. Кодекс досконалості наймогутнішого відомства США” // Харків: Віват, 2022. ISBN: 978-966-982-248-6.
20. Хіджазі, Р. “Open Source Intelligence Methods and Tools: A Practical Guide to Online Intelligence” // Apress, 2020. ISBN: 978-1-4842-5360-2.
21. The Village Україна. “Як правильно шукати інформацію з відкритих джерел: розпитали фахівця з OSINT” // the-village.com.ua. URL: https://www.village.com.ua/village/life/mediahramotnist/342503-yak-pravilno-shukati-informatsiyu-z-vidkritih-dzherel-rozpitali-fahivtsya-z-osint (Дата звернення: 19.04.2025)
22. Prometheus. “OSINT: Open Source Intelligence (безкоштовний курс)” // prometheus.org.ua. URL: https://prometheus.org.ua/prometheus-free/osint-open-source-intelligence/ (Дата звернення: 22.04.2025)

23. YouControl. “Як використовувати сайт і IP-адресу у OSINT-розслідуванні” // youcontrol.com.ua. URL: https://youcontrol.com.ua/articles/website-and-ip-intelligence-for-OSINT/ (Дата звернення: 22.02.2025)
24. k, R. M. “Intelligence Collection” // CQ Press, 2013. ISBN: N/A. (Дата звернення: 21.03.2025)
25. Weimann, G. “Terror on the Internet: The New Arena, the New Challenges” // United States Institute of Peace Press, 2006. ISBN: N/A. (Дата звернення: 22.03.2025)
26. Anderson, J. G., & Rainie, L. “The Future of Truth and Misinformation Online” // Pew Research Center, 2017. ISBN: N/A. (Дата звернення: 22.02.2025)