МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки та програмної інженерії
Кафедра інженерії програмного забезпечення

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри

Катерина НЕСТЕРЕНКО
“____”_____2024 р.

КВАЛІФІКАЦІЙНА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСНИКА ОСВІТНЬОГО СТУПЕНЯ
“БАКАЛАВР”

Тема: “Веб-застосунок з пошуку персональної інформації у структурованих даних”

Виконавець: Колесник Сергій Олександрович

Керівник: к.т.н Задонцев Юрій Вікторович

Нормоконтролер: Варнавський В’ячеслав Володимирович

Київ 2024

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки та програмної інженерії
Кафедра інженерії програмного забезпечення
Освітній ступінь бакалавр
Спеціальність 121 «Інженерія програмного забезпечення»
Освітньо-професійна програма «Програмне забезпечення систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Катерина НЕСТЕРЕНКО
"___" _____ 2024 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи студента
Колесника Сергія Олександровича

1. Тема проекту: «Веб-застосунок з пошуку персональної інформації у структурованих даних»
затверджена наказом ректора від 08.12.2023 р. № 2483/ст

2. Термін виконання проекту: з 19.01.2024 р. до 29.02.2024 р.

3. Вихідні дані до проекту: програмний продукт розробити у вигляді веб-застосунку з використанням сучасних технологій і методів.

4. Зміст пояснювальної записки:
1. Аналіз сучасних методів і технологій пошуку персональної інформації.
2. Аналіз вимог до веб-застосунку.
3. Реалізація веб-застосунку для пошуку персональної інформації у структурованих даних.
4. Аналіз та узагальнення результатів роботи веб-застосунка.

5. Перелік обов'язкових слайдів презентації:
1. Проблема пошуку персональної інформації в структурованих даних.
2. Аналіз сучасних підходів до розв'язання проблеми.
3. Огляд методів обробки та аналізу структурованих даних.
4. Напрямки розвитку аглоритму.

6. Календарний план-графік

№ пор	Завдання	Термін виконання	Відмітка про виконання
1.	Складання та затвердження графіку кваліфікаційної роботи	19.01.24 – 21.01.24	
2.	Написання 1 розділу “Аналіз сучасних методів і технологій пошуку персональної інформації та допоміжних сторінок (чорновик) - титульної, завдання, графіка, реферат, список скорочень, зміст, вступ, список джерел, 1-й нормо-контроль.	22.01.24 – 24.01.24	
3.	Написання 2 розділу “Аналіз вимог до веб-застосунку”, представлення керівнику	24.01.24 – 28.01.24	
4.	Написання 3 розділу “Реалізація веб-застосунку для пошуку персональної інформації у структурованих даних” , представлення керівнику	29.01.24 – 04.02.24	
5.	Написання 4 розділу “Аналіз та узагальнення результатів роботи веб-застосунка”, розробка тексту доповіді. Оформлення графічного матеріалу для презентації, представлення керівнику	05.02.24 – 11.02.24	
6.	Проходження нормо-контролю, перевірка на антиплагіат, перепліт пояснювальної записки. Отримання відгуку керівника, рецензії.	12.02.24 – 18.02.24	
7.	Передзахист. Допуск до захисту. Рецензування. Подача документів секретарю ДЕК (ПЗ, CD-R з електронними копіями ПЗ, презентації, відгук керівника, рецензія) в папці	19.02.24 – 25.02.24	
8.	ДЕК. Захист Кваліфікаційної роботи	26.02.24 – 29.02.24	

7. Дата видачі завдання 21.01.2024

Керівник: к.т.н. Юрій ЗАДОНЦЕВ

Завдання прийняв до виконання: Сергій КОЛЕСНИК

Дата

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи «Веб-застосунок з пошуку персональної інформації у структурованих даних»: 52 с., 13 рис., 4 табл., 27 інформаційних джерел.

ПЕРСОНАЛЬНА ІНФОРМАЦІЯ, СТРУКТУРОВАНІ ДАННІ, КОНФІДЕНЦІЙНІСТЬ

Об’єкт розробки – веб-застосунок, що дозволяє користувачам ефективно ідентифікувати та аналізувати персональну інформацію у структурованих даних.

Мета роботи – спрощення і автоматизація процесу ідентифікації, фільтрації та управління персональними даними в структурованих даних.

ABSTRACT

Explanatory note to the thesis "Web application for personal information search in structured data": 52 p. , 13 Fig. , 4 table. , 27 information sources.

PERSONAL INFORMATION, STRUCTURED DATA, PRIVACY

Property development – a web application that allows users to effectively identify and analyze personal information in structured data.

Purpose – simplifying and automating the process of identifying, filtering and managing personal data in structured data.

ЗМІСТ

ПЕРЕЛІК ПРИЙНЯТИХ СКОРОЧЕНЬ

PII (Personally Identifiable Information) – Персональна інформація

РБД – Реляційна база даних

SPA (Single Page Application) – Односторінковий застосунок

IoC (Inversion of Control) – Інверсія управління

DI (Dependency injection) – Впровадження залежності

ІК - Інтерфейс користувача

ВСТУП

У сучасному цифровому світі, де обсяги даних, що збираються та зберігаються організаціями, досягають небачених раніше масштабів, виникають значні ризики для конфіденційності та безпеки цих даних. Ці ризики стають особливо важливими, коли йдеться про персональну інформацію, зберігання та обробка якої може призвести до серйозних порушень. Ці порушення можуть мати негативний вплив не тільки на індивідуальні права, але й на корпоративну відповідальність.

Згідно з дослідженням компанії IBM [1] у 2023 році середня вартість витоку персональних даних досягла історичного максимуму і склала 4,45 млн доларів США. Перегляд галузевих особливостей показує, що найдорожчі порушення відбуваються в охороні здоров'я (10,93 млн доларів), фінансах (5,9 млн доларів), фармацевтиці (4,82 млн доларів), енергетиці (4,78 млн доларів) і промисловості (4,73 млн доларів). Це підкреслює актуальність проблеми захисту персональних даних у наш час. Персональна інформація, особливо та, яка може бути використана для ідентифікації особи, є цінною метою для кіберзлочинців та може бути використана для крадіжки особистої ідентичності, шахрайства, шантажу, та інших злочинів.

Одним із ключових способів захисту персональних даних є їх виявлення та видалення з ненадійних систем та джерел [2]. Проте, виконання цього завдання може бути складним через різноманітність форматів та способів представлення персональної інформації.

Ця дипломна робота має на меті створити веб-застосунок для пошуку персональної інформації у структурованих даних, який відповідає вищезазначеним викликам. Завдання, поставлені для досягнення цієї мети, включають:

- Аналіз теоретичних основ пошуку персональної інформації.
- Розробку архітектури веб-застосунку.

- Створення алгоритмів пошуку.
- Реалізацію та тестування застосунку.

Об'єктом дослідження виступають процеси обробки персональної інформації в структурованих даних, а предметом дослідження є алгоритми та методи, використані для їх виявлення та обробки у веб-застосунку.

Методи дослідження включають аналіз наукової літератури, порівняльний аналіз існуючих рішень, проектування та розробку програмного забезпечення, а також тестування програмного забезпечення.

Наукова новизна дипломної роботи полягає у комплексному підході до використання існуючих алгоритмів пошуку персональної інформації в структурованих даних, оптимізації цих алгоритмів для підвищення їх ефективності, точності та швидкості виявлення. Особливу увагу приділено інтеграції різних методів та технік пошуку з метою забезпечення більш глибокого аналізу та кращого розуміння контексту даних. Також особлива увага приділяється розробці ефективних методів фільтрації та обробки результатів пошуку, що дозволяє знизити час обробки та підвищити точність ідентифікації персональної інформації. Це включає вдосконалення механізмів класифікації та оцінки релевантності даних, що відкриває нові перспективи для покращення управління персональною інформацією в організаціях.

Практичне значення отриманих результатів полягає в тому, що розроблений веб-застосунок з пошуку персональної інформації у структурованих даних доповнить існуючі наукові знання в цій галузі, підвищить ефективність та надійність виявлення персональної інформації, і знайде широке практичне застосування в організаціях, які мають потребу в таких рішеннях.

РОЗДІЛ 1.
АНАЛІЗ СУЧАСНИХ МЕТОДІВ І ТЕХНОЛОГІЙ ПОШУКУ ПЕРСОНАЛЬНОЇ ІНФОРМАЦІЇ

1.1. Поняття персональної інформації

Персональна інформація, часто позначається як PII [6], є основоположним поняттям у сфері захисту даних та конфіденційності. Вона включає будь-яку інформацію, яка може бути використана для ідентифікації конкретної особи, як прямо, так і опосередковано. Поняття PII є широким і включає в себе різні види даних, від базових контактних даних до більш чутливої інформації, такої як фінансові або медичні записи. Ця інформація може бути чутливою [4][5] або нечутливою [4][5], залежно від контексту її використання та потенційного ризику для особи.

Чутлива персональна інформація включає дані, які, у разі несанкціонованого доступу, можуть завдати значної шкоди особі, включаючи фінансові збитки, порушення приватності чи інші негативні наслідки. До чутливої персональної інформації належать такі дані, як:

- Ідентифікуюча інформація: ім'я, дата народження, номер соціального страхування, адресу проживання, фотографія та інші дані, які можуть бути використані для ідентифікації особи.
- Медична інформація: медичні записи, інформація про стан здоров'я, психічне здоров'я та інші дані, які можуть бути використані для отримання інформації про здоров'я людини.

- Фінансова інформація: інформація про банківські рахунки, кредитні картки, інвестиції та інші дані, які можуть бути використані для крадіжки грошей або кредитів.
- Інформація про сексуальне життя: інформація про сексуальне здоров'я, сексуальні уподобання та інші дані, які можуть бути використані для шантажу або стигматизації.

Нечутлива персональна інформація, хоча й не вважається прямо шкідливою для особи, все ж вимагає захисту, оскільки її несанкціоноване використання може призвести до порушення приватності. До нечутливої персональної інформації належать такі дані, як:

- Демографічна інформація: стать, вік, освіта, професія, захоплення та інші дані, які не можуть бути використані для ідентифікації особи.
- Історія покупок: інформація про товари та послуги, які людина купувала.
- Історія переглядів веб-сайтів: інформація про веб-сайти, які людина відвідувала.

Важливість захисту персональної інформації не може бути переоцінена. У сучасному цифровому світі, де інформація збирається та зберігається у великих обсягах, ризики витікання персональної інформації стають все більш серйозними. Організації та окремі особи повинні бути свідомі цих ризиків та вживати належних заходів для їх мінімізації.

1.2. Поняття структурованих даних

Структуровані дані [7] – це дані, які організовані в певній логічній структурі. Ця структура визначає, як дані взаємопов'язані між собою.

Структура структурованих даних може бути представлена в різних формах, але вона завжди має певну чітку форму. Наприклад, таблиця

структурованих даних складається з рядків і стовпців, а список структурованих даних складається з послідовності елементів.

Переваги структурованих даних перед неструктурованими даними включають:

- Легкість розуміння і обробки: структуровані дані легше зрозуміти і обробити, ніж неструктуровані дані. Це пов'язано з тим, що структура структурованих даних визначає, як дані взаємопов'язані між собою.
- Придатність для зберігання та пошуку: структуровані дані більш придатні для зберігання та пошуку, ніж неструктуровані дані. Це пов'язано з тим, що структура структурованих даних дозволяє швидко знаходити потрібні дані.
- Підвищена безпека даних: структуровані дані можуть бути легше захищені в порівнянні з неструктурованими або напівструктурованими даними, оскільки доступ до даних може бути контрольованим через протоколи безпеки баз даних.
- Недоліки структурованих даних включають:
- Негнучкість: структуровані дані можуть бути негнучкими в тому плані, що вони обмежені в прийнятті нових типів даних, оскільки будь-які зміни в схемі або моделі даних вимагають значних змін в базі даних.
- Обмежена складність: структуровані дані часто обмежені в термінах складності взаємовідносин між сутностями даних. Це може ускладнити моделювання складних реальних сценаріїв.
- Якість даних: структурований характер даних іноді може призводити до відсутності або неповної інформації, або до даних, які не вписуються чітко у визначену схему, що призводить до проблем з якістю даних.

У розрізі кваліфікаційної роботи ми будемо використовувати такі джерела структурованих даних:

- Реляційні бази даних
- Електронні таблиці

Реляційні бази даних (РБД) є одним із найпоширеніших типів баз даних. Вони складаються з таблиць, які містять рядки та стовпці. Рядки таблиці називаються записами, а стовпці таблиці називаються полями. Кожна таблиця в реляційній базі даних має певну структуру, яка визначає, які поля містяться в таблиці та які типи даних ці поля можуть містити.

РБД мають ряд переваг перед іншими типами баз даних. Вони є відносно простими у використанні та розумінні. Вони також є ефективними для зберігання та пошуку даних.

Електронні таблиці є ще одним поширеним типом структурованих даних. Вони складаються з рядків та стовпців, які аналогічні записам та полям у реляційних базах даних. Кожна електронна таблиця має певну структуру, яка визначає, які дані містяться в ній.

Електронні таблиці є відносно простими у використанні та розумінні. Вони також є ефективними для зберігання та обробки даних.

1.3. Алгоритми або методи пошуку персональної інформації

Список заборон (deny-list) - це простий алгоритм ідентифікації персональної інформації, який використовує список заборонених слів. Цей список містить слова, які зазвичай зустрічаються в персональній інформації, такі як імена, адреси, номери телефонів і електронні адреси. Алгоритм сканує текстові дані, шукаючи ці слова. Якщо слово зустрічається в списках, алгоритм ідентифікує його як персональну інформацію [8].

Принцип роботи

Принцип роботи «списку заборон» можна описати наступним чином:

- Створення списку заборонених слів. Спочатку створюється список заборонених слів. Цей процес вимагає ретельного аналізу домену даних.
- Сканування текстових даних. Потім текстові дані скануються по слову.

- Ідентифікація персональної інформації. Якщо слово зустрічається в списку заборонених слів, воно ідентифікується як персональна інформація.

Переваги

Список заборон має ряд переваг, зокрема:

- Проста реалізація. Список заборон є простими у реалізації, що робить його доступними для широкого кола користувачів.
- Контрольованість. Надає користувачам контроль над тим, які дані блокуються.
- Ефективність для виявлення відомих шаблонів. Список заборон може бути ефективним для виявлення відомих шаблонів персональної інформації, таких як раса, стать, релігія, звання і т. ін..

Недоліки

Список заборон також має ряд недоліків, зокрема:

- Неефективність для виявлення невідомих шаблонів. Список заборон може бути не таким ефективним для виявлення невідомих шаблонів персональної інформації, таких як псевдоніми, зашифровані дані, номери карток і т. ін..
- Потреба в постійному оновленні. Списки потрібно регулярно оновлювати, що може бути трудомістким.
- Не враховує контекст. Списки заборон можуть не розпізнавати важливий контекст, у якому використовується термін.

Складність реалізації

Складність реалізації список заборон залежить від того, як створюється список заборонених слів. Якщо список створюється вручну, то його реалізація є відносно простою. Однак це може бути трудомістким завданням, особливо якщо список повинен бути великим. Якщо список створюється за допомогою

машинного навчання, то його реалізація може бути більш складною, але вона може забезпечити більшу точність.

Витрати

Витрати, пов'язані з використанням списку заборон, залежать від того, як створюється список заборонених слів. Якщо список створюється вручну, то витрати є відносно низькими. Однак це може бути трудомістким завданням, особливо якщо список повинен бути великим.

Регулярні вирази (regex) - це потужний інструмент, який можна використовувати для пошуку певних даних у текстових даних. Вони можуть бути використані для виявлення персональної інформації, наприклад, номера кредитних карток, поштових кодів, номерів телефонів і електронних адрес. [9]

Принцип роботи

Регулярний вираз - це шаблон, який використовується для пошуку певних даних у текстових даних. Алгоритм на основі регулярних виразів сканує текстові дані, шукаючи регулярні вирази, які відповідають шаблонам персональної інформації.

Переваги

Регулярні вирази мають ряд переваг, зокрема:

- Ефективність для виявлення невідомих шаблонів. Регулярні вирази можуть бути ефективними для виявлення невідомих шаблонів персональної інформації, таких як псевдоніми або шифровані дані.
- Точність. Регулярні вирази можуть бути досить точними, якщо вони правильно створені.
- Гнучкість. Можливість налаштування під специфічні вимоги або формати даних.
- Універсальність. Працюють у багатьох програмних мовах і середовищах.
- Можливість комплексного пошуку. Можна конструювати складні шаблони для виявлення даних, які відповідають кільком критеріям одночасно.

Недоліки

Регулярні вирази також мають ряд недоліків, зокрема:

- Складність реалізації. Регулярні вирази можуть бути складними у реалізації, що може ускладнити їх використання для широкого кола користувачів.
- Ризик помилкових спрацьовувань. Можуть ідентифікувати некоректну інформацію як персональні дані.
- Час виконання. Регулярні вирази можуть бути повільними, якщо текстові дані великі.

Складність реалізації

Складність реалізації алгоритму на основі регулярних виразів для пошуку персональної інформації полягає у необхідності точного визначення патернів, які мають відповідати різноманітним форматам даних. Це вимагає детального розуміння даних, з якими працює система, та може бути часомістким через необхідність тестування та налагодження виразів для забезпечення їх ефективності та точності. Підтримка та оновлення виразів також вимагає регулярного перегляду, щоб вони залишалися актуальними у змінних умовах використання даних.

Час виконання

Час виконання алгоритму на основі регулярних виразів залежить від розміру текстових даних і складності регулярного виразу. Якщо текстові дані великі або регулярний вираз складний, то час виконання може бути значним.

Регулярні вирази - це потужний інструмент, який можна використовувати для виявлення персональної інформації. Вони можуть бути ефективними для виявлення як відомих, так і невідомих шаблонів персональної інформації. Однак важливо розуміти їхні переваги та недоліки, щоб використовувати їх ефективно.

Правило-орієнтований (rule-based) - це алгоритм ідентифікації персональної інформації, який використовує правила. Правило - це умова, яка визначає, чи є певні дані персональною інформацією. [8]

Принцип роботи

Принцип роботи можна описати наступним чином:

- Створення набору правил. Спочатку створюється набір правил, які визначають, чи є певні дані персональною інформацією.
- Аналіз текстових даних. Потім текстові дані аналізуються на відповідність визначеним правилам.
- Ідентифікація персональної інформації. Якщо дані відповідають правилу, вони ідентифікуються як персональна інформація.

Переваги

Правило-орієнтовані алгоритми з пошуку персональної інформації мають ряд переваг, зокрема:

- Точність: Може точно виявляти визначені шаблони та структури.
- Прозорість. Чітке розуміння того, як відбувається ідентифікація та обробка даних.
- Контрольованість: Користувачі мають повний контроль над логікою виявлення.
- Гнучкість. Здатність адаптувати систему до нових вимог шляхом зміни або додавання правил.

Недоліки

Правило-орієнтований алгоритм персональної інформації також має ряд недоліків, зокрема:

- Обмежена гнучкість: Може не враховувати нові або непередбачені шаблони даних.
- Потреба в постійному оновленні: Правила потрібно регулярно оновлювати, щоб відповідати змінам у даних або вимогах.

- Ресурсомісткість: Розробка і підтримка складних наборів правил може бути ресурсоємною.
- Ризик помилкових позитивних або негативних результатів: Може помилково ідентифікувати або упускати дані через жорсткість правил.

Складність реалізації

Складність реалізації правило-орієнтованого алгоритму для пошуку персональної інформації полягає в потребі ретельної розробки та налаштування правил для відповідності різноманітним форматам даних. Це включає аналіз специфіки даних, визначення шаблонів і структур, що мають виявлятися, та розробку гнучкої системи правил, яка може адаптуватися до змін. Крім того, важливим є регулярне тестування та оновлення правил для забезпечення точності і актуальності системи.

Час виконання

Час виконання правило-орієнтованого алгоритму для пошуку персональної інформації може варіюватися в залежності від кількох факторів, зокрема складності та кількості правил, обсягу даних для обробки та ефективності самої системи. Для невеликих даних з простими правилами виконання може бути швидким. Проте, у випадках з великими обсягами даних або складними наборами правил, процес може зайняти значно більше часу через необхідність детального аналізу та перевірки кожного елемента даних.

Обробка природної мови (NLP) – це алгоритми ідентифікації персональної інформації, які використовують природну мовну обробку (NLP). NLP - це галузь комп'ютерної науки, яка стосується розуміння та обробки природної мови. [10]

Принцип роботи

Принцип роботи NLP алгоритмів для пошуку персональної інформації можна описати наступним чином:

- Фразовий розбір. Спочатку текстові дані розбираються на фрази.

- Ідентифікація граматичних конструкцій. Потім ідентифікуються граматичні конструкції, які можуть містити персональну інформацію.
- Виявлення імен, адрес, номерів телефонів тощо. На основі ідентифікованих граматичних конструкцій виявляються імена, адреси, номери телефонів тощо.

Переваги

NLP алгоритми персональної інформації мають ряд переваг, зокрема:

- Глибоке розуміння контексту. Розпізнавання нюансів та семантики мови для точнішого виявлення персональних даних..
- Точність. NLP алгоритми можуть бути досить точними, якщо вони навчені на великому наборі даних. Це пов'язано з тим, що NLP-алгоритми можуть використовувати статистичні методи, щоб визначити, які ознаки є найбільш характерними для персональної інформації.
- Масштабованість. Ефективна обробка великих обсягів тексту.

Недоліки

NLP алгоритмів для пошуку персональної інформації також мають ряд недоліків, зокрема:

- Складність реалізації. NLP алгоритми можуть бути складними у реалізації, що може ускладнити їх використання для широкого кола користувачів. Це пов'язано з тим, що NLP алгоритми вимагають значних знань у галузі комп'ютерної науки та лінгвістики.
- Потреба в обширних тренувальних даних. Необхідність великої кількості анотованих даних для ефективного навчання моделей.
- Час виконання. Обробка великих обсягів даних може бути часозатратною, особливо для складних моделей NLP.

Складність реалізації

Складність реалізації NLP алгоритму персональної інформації залежить від того, як вони навчаються. Якщо NLP алгоритм навчаються на наборі даних,

який містить лише відомі шаблони персональної інформації, то їх реалізація може бути відносно простою. Однак це може призвести до зниження точності розпізнавання невідомих шаблонів. Якщо NLP алгоритм навчаються на наборі даних, який містить як відомі, так і невідомі шаблони персональної інформації, то їх реалізація може бути більш складною, але це може призвести до підвищення точності розпізнавання невідомих шаблонів.

Таблиця 1.1.

Порівняння характеристик алгоритмів виявлення персональної інформації.

Характеристика	Список заборон	Регулярні вирази	Правило-орієнтований	Обробка природної мови
Принцип роботи	Порівнює слова в текстових даних із списком заборонених слів.	Використовує регулярні вирази для пошуку шаблонів персональної інформації в текстових даних.	Використовує правила для визначення того, чи є певні дані персональною інформацією.	Використовує природну мовну обробку для розуміння контексту та виявлення персональної інформації.
Переваги	Проста реалізація, контрольованість, ефективність для виявлення відомих шаблонів.	Ефективність для виявлення невідомих шаблонів, точність, гнучкість, універсальність, можливість комплексного пошуку.	Точність, прозорість, контрольованість, гнучкість.	Глибоке розуміння контексту, точність, масштабованість.
Недоліки	Неефективність для виявлення невідомих шаблонів, потреба в постійному оновленні, не враховує контекст.	Складність реалізації, ризик помилкових спрацьовувань, час виконання.	Обмежена гнучкість, потреба в постійному оновленні, ресурсоємність, ризик помилкових позитивних або негативних результатів.	Складність реалізації, потреба в обширних тренувальних даних, час виконання.
Складніст	Відносно	Відносно складна,	Відносно	Більш складна, ніж

ь реалізації	проста, якщо список заборонених слів створюється вручну. Більш складна, якщо список створюється за допомогою машинного навчання.	якщо регулярні вирази створюються вручну. Більш складна, якщо регулярні вирази створюються за допомогою машинного навчання.	складна, якщо правила створюються вручну. Більш складна, якщо правила створюються за допомогою машинного навчання.	інші алгоритми.
Витрати	Відносно низькі, якщо список заборонених слів створюється вручну.	Відносно низькі, якщо регулярні вирази створюються вручну.	Відносно низькі, якщо правила створюються вручну.	Відносно високі, якщо алгоритм навчається на великому наборі даних.

Кожен метод має свої особливості, переваги та обмеження, тому важливо вибрати найбільш підходящий під конкретні потреби та обставини. Залежно від конкретної ситуації, можна застосувати один з методів, або ж їх комбінацію для досягнення оптимальної точності та ефективності в ідентифікації та захисті персональних даних. Враховуючи стрімкий розвиток технологій та зміни в структурі даних, постійне оновлення та удосконалення цих методів є ключовим для забезпечення довгострокової ефективності в захисті персональної інформації.

Висновки до розділу 1.

У сучасному цифровому світі особиста інформація є цінним активом. Вона може бути використана для ідентифікації людей, навмисного вторгнення в їхню приватність або навіть для здійснення злочинів. Тому важливо мати ефективні методи пошуку персональної інформації, щоб захистити людей від цих загроз.

Вивчення різноманітних методів пошуку персональної інформації підкреслює можливості їхньої інтеграції для створення більш ефективних систем ідентифікації. Комбінування різних підходів, від структурованих списків заборонених слів до складних алгоритмів обробки природної мови, дозволяє забезпечити більш глибокий та всебічний аналіз даних. Це не тільки підвищує точність ідентифікації персональної інформації, але й зменшує ризик помилок, наприклад, помилкових спрацьовувань або пропусків даних.

Інтеграція різних методів також відкриває шлях для адаптивності системи до змінних умов та нових викликів, забезпечуючи гнучкість у виявленні персональних даних у різних форматах та контекстах. Такий багатоаспектний підхід може бути надзвичайно корисним у виявленні складних шаблонів та у захисті інформації в умовах постійного технологічного прогресу.

У підсумку, об'єднання різних методів пошуку персональної інформації в одну координовану систему представляє собою стратегічний підхід, який може значно покращити якість та ефективність ідентифікації персональних даних, одночасно забезпечуючи гнучкість та адаптацію до постійно змінюваних вимог і умов.

РОЗДІЛ 2.
АНАЛІЗ ВИМОГ ДО ВЕБ-ЗАСТОСУНКУ

2.1. Аналіз бізнес-вимог до веб-застосунку

Цільовий ринок та користувачі

Сегменти ринку: Застосунок призначений для малого, середнього, та великого бізнесу, а також урядових організацій. Він має бути універсальним інструментом для будь-яких організацій, що бажають керувати та контролювати персональні дані в їхніх системах.

Користувачі: Користувачі веб-застосунку включають:

- Адміністраторів, які відповідають за конфігурацію та управління застосунком.
- Користувачів, які використовують застосунок для пошуку, аналізу та звітності про персональні дані.

Основні функції та можливості

Пошук персональної інформації:

- Інтеграція з різними типами баз даних та файловими сховищами для забезпечення всебічного пошуку та класифікації персональних даних.
- Реалізація алгоритмів для ідентифікації типових та неочевидних випадків персональної інформації.

Аналіз та звітність:

- Розробка інструментів для агрегування даних пошуку, їх аналізу та представлення у зручних форматах для користувачів. Це включає в себе:

- Візуалізацію структур даних
- Інтерактивні діаграми чутливості
- Відображення статистики знайдених даних

Вимоги до інтеграції

Інтеграція з джерелами даних:

- Підтримка популярних систем управління базами даних (наприклад, PostgreSQL, MySQL, MariaDB) та файлових сховищ (наприклад, CIFS).
- Розробка API та плагінів для забезпечення легкої інтеграції з новими та існуючими системами клієнтів.

Безпека та конфіденційність

Захист та шифрування:

- Реалізація надійних методів шифрування для зберігання чутливої інформації.
- Впровадження політик безпеки для управління доступом до функцій застосунку та даних.
- Забезпечення відповідності до міжнародних та місцевих нормативних вимог щодо захисту даних, таких як GDPR.

Технічні та технологічні обмеження

Розгортання:

- Врахування необхідності розгортання на серверах компанії, забезпечення безпеки, високої доступності, та надійності сервісу.
- Планування ресурсів для масштабування та обслуговування застосунку.

Масштабованість та продуктивність

Великі об'єми даних:

- Забезпечення здатності ефективно обробляти великі датасети, включаючи бази даних з мільйонами записів.
- Оптимізація продуктивності для можливості одночасного сканування та аналізу декількох джерел даних.

Користувацький інтерфейс

Інтуїтивний дизайн:

- Розробка користувацького інтерфейсу, який є зрозумілим та ефективним для різних типів користувачів.
- Врахування сучасних трендів дизайну та забезпечення адаптивності для різних пристроїв.

Інтерактивність:

- Включення інтерактивних елементів, таких як підказки та контекстно-залежна допомога, для поліпшення користувацького досвіду.

2.1. Аналіз функціональних вимог

Інтеграція з різними джерелами даних:

Підтримка баз даних:

- **Конектори до баз даних:** Розробити спеціалізовані конектори для популярних систем управління базами даних. Кожен конектор повинен вміти автоматично виявляти схеми, таблиці, та поля, а також мати можливість читання даних з урахуванням специфіки кожної СУБД.
- **Безпечне з'єднання:** Впровадження заходів безпеки для з'єднань, включаючи шифрування, управління доступом, та моніторинг активності. Забезпечення, що конфіденційність і цілісність даних не буде порушена під час з'єднання.

Підтримка файлових форматів:

- **Аналізатори файлів:** Розробка або інтеграція існуючих аналізаторів для розпізнавання та обробки структурованих файлових форматів. Включення можливостей для розбору складних файлів, таких як вкладені таблиці, змішані типи даних, і т.д.
- **Обробка великих файлів:** Оптимізація для обробки великих файлів, забезпечення ефективного парсингу та аналізу без значного впливу на продуктивність системи.

Конфігурація інтеграції:

- **Інтерфейс налаштувань:** Надання користувачам зручного інтерфейсу для введення параметрів інтеграції, включаючи вибір джерел даних, налаштування з'єднань, та специфікацію об'єктів для сканування.
- **Адаптивність інтеграції:** Забезпечення можливості легкої модифікації і додавання нових джерел даних у майбутньому без значного переписування коду або архітектурних змін.

Пошук та ідентифікація персональних даних:

Алгоритми ідентифікації:

- **Виявлення шаблонів:** Розробка алгоритмів для виявлення типових шаблонів персональних даних, як-от номерів телефонів, адрес, імен, ідентифікаційних номерів тощо. Використання регулярних виразів, машинного навчання, або інших методів для підвищення точності та зменшення помилкових спрацьовувань.
- **Контекстний аналіз:** Впровадження можливостей контекстного аналізу для розуміння змісту даних та відокремлення персональної інформації від несуттєвої. Розробка алгоритмів, які можуть аналізувати семантику та структуру даних.

Класифікація даних:

- **Система категорій:** Створення чіткої системи категоризації для персональних даних, яка відповідає як загальноприйнятим стандартам, так і специфіці бізнесу користувача. Можливість налаштування категорій та додавання нових за потребою.
- **Автоматична класифікація:** Розробка методів автоматичної класифікації знайдених даних, забезпечення відображення інформації за відповідними категоріями в звітах і на інтерфейсі.

Параметри пошуку:

- **Гнучкі налаштування:** Надання користувачам можливості гнучкого налаштування параметрів пошуку, включаючи вибір конкретних полів для сканування, визначення пріоритетних областей, та інших критеріїв для вдосконалення результатів.
- **Інтерактивність:** Розробка інтерактивних елементів для роботи з параметрами пошуку, таких як слайдери, чекбокси, та випадаючі списки для зручності користувачів.

Безпека даних та управління доступом:

Шифрування даних:

- **Захист на рівні бази даних:** Впровадження шифрування на рівні баз даних та файлів для забезпечення конфіденційності збережених даних. Використання передових методів шифрування для забезпечення високого рівня захисту.
- **Безпечна передача даних:** Забезпечення безпеки даних під час їх передачі між застосунком та джерелами даних, включаючи використання захищених протоколів передачі.

Авторизація та аутентифікація:

- **Рівні доступу:** Розробка диференційованих рівнів доступу для різних типів користувачів, забезпечення, що кожен користувач має доступ лише до тих даних і функцій, які дозволені його роллю.
- **Система аутентифікації:** Впровадження надійної системи аутентифікації з підтримкою багатофакторної автентифікації, сесійного управління, та інших сучасних методів забезпечення безпеки.

Аудит та журналування:

- **Журнали дій:** Ведення детальних журналів всіх дій користувачів, запитів до системи, та змін у даних. Це дозволяє забезпечити відстежуваність дій та спростити розслідування в разі інцидентів.
- **Інструменти для аудиту:** Розробка інструментів та інтерфейсів для перегляду, аналізу та управління журналами дій. Забезпечення можливості швидкого доступу до історії дій для адміністраторів та аудиторів.

Візуалізація та звітність:

Інструменти візуалізації:

- **Детальні графіки та мапи:** Розробка інструментів для створення інтерактивних графіків, мап, та інших візуальних представлень, що дозволяють користувачам легко інтерпретувати результати сканування та аналізу даних.
- **Кастомізація відображення:** Надання можливості користувачам кастомізувати візуалізації згідно з їхніми перевагами та потребами, включаючи зміну масштабу, кольорів, та інших параметрів представлення даних.

Генерація звітів:

- **Автоматизовані звіти:** Автоматична генерація звітів на основі результатів сканування, з можливістю кастомізації структури та змісту звітів.
- **Експорт даних:** Підтримка експорту звітів у різноманітні формати, такі як Excel або CSV, для зручності подальшого аналізу, ділення з колегами, або архівування.

Інтерактивність:

- **Інтерактивне управління даними:** Розробка інтерфейсів, які дозволяють користувачам інтерактивно працювати з даними, включаючи фільтрацію результатів, детальний перегляд конкретних елементів, та зміну параметрів відображення на льоту.

2.2. Аналіз не функціональних вимог

Продуктивність та ефективність:

Оптимізація обробки:

- **Час відгуку:** Визначення та забезпечення максимально допустимого часу відгуку на запити користувачів. Застосунок повинен виконувати сканування, аналіз, та відображення результатів в межах заданого часу, забезпечуючи гладке та ефективне використання.
- **Обробка великих об'ємів даних:** Система має бути спроектована та оптимізована для швидкої обробки великих датасетів. Використання технік паралельної обробки, кешування, та розподілених систем даних для підвищення швидкості та ефективності.

Масштабування:

- **Горизонтальне та вертикальне масштабування:** Система має підтримувати як горизонтальне, так і вертикальне масштабування для вирішення зростаючих потреб в обробці даних. Впровадження

автоматичного масштабування може бути розглянуто для хмарних розгортань.

Масштабованість:

Впорядковане зростання:

- **Здатність до зростання:** Система має бути спроектована таким чином, щоб легко адаптуватися до збільшення обсягу даних та кількості користувачів. Це включає в себе використання модульної архітектури, легко масштабованих баз даних, та інших компонентів, які можна масштабувати незалежно.

Паралельне оброблення:

- **Множинність обробки:** Система має забезпечувати можливість одночасного оброблення декількох запитів, забезпечуючи паралельне сканування, аналіз, та звітність без втрати продуктивності або стабільності.

Надійність та доступність:

Стабільна робота:

- **Відновлення після збоїв:** Розробка механізмів для швидкого відновлення системи після збоїв, включаючи резервне копіювання та відновлення даних, автоматичне переключення на резервні системи.
- **Розподіленість та відмовостійкість:** Використання розподілених архітектур та відмовостійких компонентів для забезпечення безперервної роботи застосунку навіть у випадку відмови окремих елементів системи.

Доступність:

- **24/7 Доступність:** Забезпечення високого рівня доступності системи, включаючи можливість доступу до функціоналу застосунку в будь-який час з будь-якої точки світу.
- **Балансування навантаження:** Впровадження рішень для балансування навантаження, щоб оптимізувати розподіл ресурсів та забезпечити стійку роботу при високому навантаженні.

Інтуїтивний користувацький інтерфейс:

Зручність використання:

- **Чистий та простий дизайн:** Розробка інтерфейсу, який є зрозумілим та легким у використанні для користувачів різного рівня технічної підготовки. Включення навігації, яка дозволяє легко переходити між різними функціями.
- **Адаптивність:** Створення дизайну, який адаптується до різних розмірів екранів та пристроїв, включаючи настільні комп'ютери, ноутбуки, та мобільні пристрої.

Допомога та підтримка:

- **Інтерактивні підказки:** Впровадження інтерактивних підказок, контекстних меню, та інших форм підтримки користувачів для полегшення процесу освоєння системи.
- **Документація та навчальні матеріали:** Надання докладної документації, відеоуроків, FAQ, та інших навчальних ресурсів для допомоги користувачам у вирішенні проблем та підвищенні ефективності використання системи.

Відповідність стандартам та законодавству:

Законодавча відповідність:

- **Дотримання приватності та захисту даних:** Забезпечення, що система відповідає всім відповідним міжнародним, національним та місцевим законодавчим актам щодо захисту даних, таким як GDPR, HIPAA тощо.
- **Регулярні оновлення та аудити:** Впровадження процедур регулярних оновлень та аудитів системи для забезпечення її актуальності та відповідності змінам у законодавстві та технологіях.

Висновки до розділу 2.

У розділі 2 ми провели всебічний аналіз бізнес-вимог до веб-застосунку, вивчивши цільові ринки та користувачів, основні функції та можливості застосунку, а також вимоги до інтеграції, безпеки, технічних обмежень, масштабованості та користувацького інтерфейсу. Ми визначили ключові аспекти для ефективної роботи застосунку: від пошуку та класифікації персональних даних до забезпечення високої продуктивності та надійності сервісу. Отримані висновки допоможуть нам розробити веб-застосунок, який не тільки задовольняє потреби бізнесу, але й відповідає всім стандартам безпеки та приватності.

РОЗДІЛ 3.
РЕАЛІЗАЦІЯ ВЕБ-ЗАСТОСУНКУ ДЛЯ ПОШУКУ ПЕРСОНАЛЬНОЇ ІНФОРМАЦІЇ У СТРУКТУРОВАНИХ ДАНИХ

3.1. Вибір технологій

Вибір технологій для реалізації веб-застосунку для пошуку персональної інформації у структурованих даних є критично важливим аспектом процесу розробки. Це не лише визначає технічні характеристики та функціональність програмного продукту, але й має прямий вплив на ефективність, швидкість розробки, зручність підтримки та можливість майбутнього розширення застосунку. Досягнення успіху в цій сфері вимагає уважного аналізу потреб, вимог та можливостей, що робить вибір правильних технологій критично важливою задачею.

При виборі технологій для розробки веб-застосунку враховувались такі ключові фактори:

- Функціональні можливості веб-застосунку.
- Простота та зручність використання.
- Масштабованість та продуктивність.
- Безпека та надійність.
- Вартість та доступність технологій.

Вибір фронтенд фреймворка

Для розробки фронтенд частини веб-застосунку було обрано фреймворк Angular. Angular - це JavaScript фреймворк з відкритим кодом, що використовується для побудови односторінкових веб-застосунків (SPA).

Angular має ряд переваг, що роблять його вдалим вибором для даного веб-застосунку:

- Модульність: Angular дозволяє розбивати веб-застосунок на модулі, що робить код більш організованим та зручним для обслуговування.
- Компоненти: Angular використовує компоненти для візуалізації даних та реалізації логіки веб-застосунку.
- Data binding: Angular використовує data binding для синхронізації даних між моделлю та представленням веб-застосунку.
- Routing: Angular використовує routing для навігації між різними сторінками веб-застосунку.
- TypeScript: Angular використовує TypeScript, що є надмножиною JavaScript з додатковими можливостями, такими як статична типізація.

Використання Angular для розробки фронтенд веб-застосунку має ряд переваг, таких як:

- Зменшення часу розробки: Angular дозволяє розробникам швидше створювати веб-застосунки завдяки наявності готових компонентів та модулів.
- Підвищення продуктивності: Angular використовує TypeScript, що дозволяє писати більш якісний код.
- Зручність обслуговування: Angular дозволяє легко розбивати веб-застосунок на модулі.

Вибір мови програмування

Для розробки бекенд веб-застосунку було обрано мову програмування Java. Java – це популярна мова програмування, що має ряд переваг, які роблять її вдалим вибором для цієї мети:

- Об'єктно-орієнтованість: Java – це об'єктно-орієнтована мова програмування, що робить код більш організованим та зручним для обслуговування.
- Платформна незалежність: Java-код може запускатися на будь-якій платформі, що має Java Virtual Machine (JVM).
- Велика спільнота: Java має велику та активну спільноту розробників, що робить доступними багато бібліотек, інструментів та навчальних матеріалів.
- Безпека: Java пропонує ряд функцій для забезпечення безпеки веб-застосунків, таких як контроль доступу, шифрування та перевірка помилок.
- Продуктивність: Java – це високопродуктивна мова програмування, що робить її ідеальною для масштабованих веб-застосунків, які повинні обробляти велику кількість даних та запитів.

Вибір бекенд фреймворка

В якості фреймворку для розробки бекенд веб-застосунку було обрано Spring. Spring - це фреймворк, що значно полегшує розробку застосунків, та пропонує ряд переваг, таких як:

- Високий рівень абстракції: Spring надає високий рівень абстракції, що спрощує розробку та підтримку веб-застосунків. Він дозволяє розробникам концентруватися на бізнес-логіці, уникнувши деталей інфраструктури.
- Широкий спектр модулів: Фреймворк має велику кількість додаткових модулів, які надають різноманітні можливості для роботи з різними аспектами розробки, такими як безпека (Spring Security), робота з базами

даних (Spring Data), робота з REST API (Spring Web MVC) та багато інших.

- Інверсія управління (IoC) та впровадження залежності (DI): Spring використовує принципи інверсії управління та внедрення залежностей, що дозволяє розробникам створювати відокремлені та легко тестовані компоненти.
- Велике співтовариство: Spring має велике та активне співтовариство, що робить доступними багато ресурсів та допомоги. Ви можете знайти багато інформації та допомоги в Інтернеті, що робить процес розробки більш простішим.
- Масштабованість: Spring – це масштабований фреймворк, який може бути використаний для розробки веб-застосунків будь-якого розміру. Ви можете бути впевнені, що ваш веб-застосунок буде масштабуватися разом з вашим бізнесом.

Вибір системи обробки повідомлень

Для обробки повідомлень у бекенд веб-застосунку обрано систему обробки повідомлень Apache Kafka. Apache Kafka – це відома масштабована та розподілена система, спроектована для ефективної передачі даних між різними компонентами веб-застосунків. Її архітектура базується на розподілених та реплікованих кластерах серверів, що дозволяє забезпечити високу доступність та стійкість до відмов. Однією з ключових переваг Apache Kafka є її здатність опрацьовувати великі обсяги даних з високою швидкістю та низькою затримкою, що робить її ідеальним вибором для веб-застосунків, де потрібна швидка та ефективна обробка потокової інформації. Крім того, Apache Kafka має багато вбудованих функцій, таких як можливості реплікації даних, масштабованість та гарантує доставку повідомлень у правильному порядку, що робить її потужним інструментом для вирішення поставлених завдань.

Вибір бази даних

Для збереження даних буде використовуватись реляційна база даних PostgreSQL. PostgreSQL – це реляційна система керування базами даних з відкритим кодом, що володіє широким спектром функцій та можливостей. Вона використовується для розробки веб-застосунків, enterprise-систем, аналітики даних та багато іншого.

Переваги PostgreSQL:

- Масштабованість: PostgreSQL може бути розгорнута на кластері комп'ютерів, що робить її здатною обробляти великі обсяги даних.
- Надійність: PostgreSQL має високу доступність та стійкість до відмов.
- Безпека: PostgreSQL пропонує ряд функцій для забезпечення безпечного зберігання даних.
- Гнучкість: PostgreSQL підтримує різні типи даних, такі як JSON і XML.
- Простота використання: PostgreSQL має простий у використанні API.
- Відкритий код: PostgreSQL – це безкоштовний та відкритий код, що робить його доступним для всіх.

3.2. Архітектура веб-застосунку

Веб-застосунок буде побудований на основі мікросервісної подійно-орієнтованої архітектури. Цей тип архітектури має ряд переваг, що роблять його ідеальним вибором для веб-застосунків, які потребують високої масштабованості, гнучкості та надійності.

Мікросервісна архітектура – це стиль розробки програмного забезпечення, де система розбивається на невеликі, незалежні та самодостатні сервіси. Ці сервіси, або мікросервіси, спілкуються між собою через за допомогою шини пофідомлень.

Подійно-орієнтована архітектура – це стиль розробки програмного забезпечення, де система реагує на події, що генеруються в системі або зовні.

Мікросервісна подійно-орієнтована архітектура поєднує в собі переваги мікросервісної та подійно-орієнтованої архітектур. Цей тип архітектури ідеально підходить для веб-застосунків, які потребують:

- Високої масштабованості: Мікросервіси можна легко масштабувати, щоб задовольнити зростання попиту.
- Гнучкості: Мікросервіси можна легко розробляти, тестувати та розгортати, що дозволяє швидко вносити зміни до веб-застосунку.
- Надійності: Подійно-орієнтована архітектура робить веб-застосунок більш стійким до відмов.
- Простоти обслуговування: Мікросервіси простіше обслуговувати та оновлювати, ніж монолітні програми.

Мікросервісна подійно-орієнтована архітектура складається з наступних компонентів:

- Мікросервіси: Невеликі, незалежні та самодостатні служби, що виконують певну задачу.
- Події: Асинхронні повідомлення, що використовуються для комунікації між мікросервісами.
- Шина подій: Система, що використовується для маршрутизації подій до відповідних мікросервісів.

Взаємодія між бекенд мікросервісами відбувається через шину подій. Мікросервіс публікує подію, коли щось відбувається, а інші мікросервіси можуть підписатися на цю подію, щоб отримати повідомлення про неї. Взаємодія між бекенд і фронтенд сервісами буде відбуватись за допомогою REST API.

Веб-застосунок буде використовувати ряд технологій для реалізації мікросервісної подійно-орієнтованої архітектури, таких як:

- Spring Boot: Фреймворк для розробки мікросервісів на Java.
- Angular: Фреймворк для розробки веб інтерфейсу на TypScript.

- Kafka: Система обробки подій.
- PostgreSQL: Система зберігання даних.

На цьому етапі веб застосунок буде складатись з трьох компонентів. 3 мікросервісів. Data Source manager UI, Data Source manager і Database scanner.

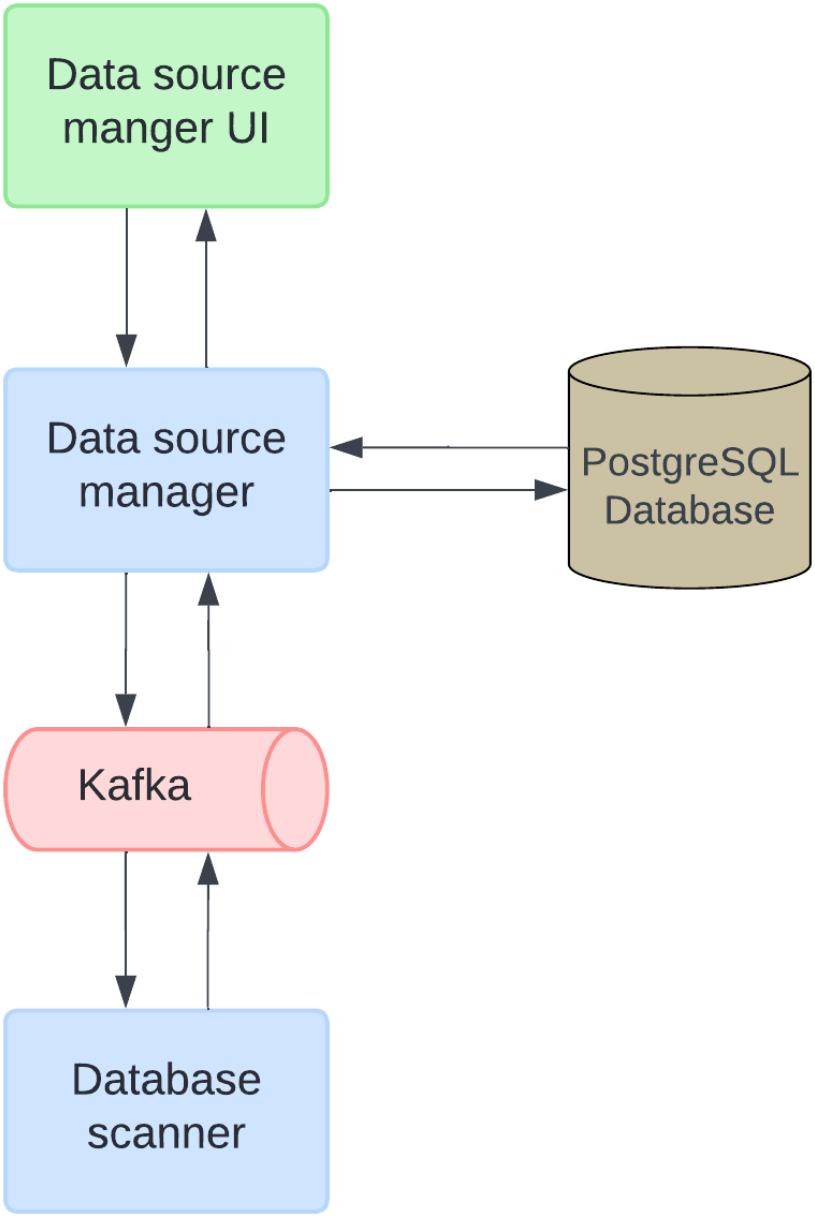

Рис. 3.2.1. Мікросервісна подійно-орієнтована архітектура веб-застосунку.

Data Source Manager UI - це фронтенд-сервіс, що відповідає за візуалізацію користувацького інтерфейсу.

Data Source Manager - це бекенд-сервіс, що керує джерелами даних та зберігає результати їх класифікації. На даному етапі, сервіс підтримує лише тип джерела даних "База даних". Але структура бази даних і інтерфейсів для взаємодії розроблена таким чином, щоб у подальшому її можна було легко розширити і додати потрібний тип даних. Наприклад центральне сховище або хмарне сховище.

База даних буде складатись з 4 таблиць data_source, database_classification, database_classification_table, database_classification_column. Таблиця data_source відповідає за збереження інформації про джерело даних. В свою чергу database_classification,database_classification_table, database_classification_column буде зберігати результати класифікації для джерел з типом даних база даних. Детальна модель «сутність-зв'язок» бази даних зображені на рис. 3.2.2..

Рис. 3.2.2. Модель «сутність-зв'язок» бази даних data_source_manager.

Архітектура передбачає наявність сервісів для сканування джерел різного виду. Так звані “Data source scanner”. Взаємодія між цими сканерами і менеджером джерел даних буде виконуватись за допомогою шини подій. Тому важливо початково мати гнучкі контракти повідомлень, які у майбутньому зможуть підтримувати різні типи джерел даних.

Так як початково система підтримує тільки тип джерела даних база даних, відповідно буде реалізовано сканер баз даних.

Database Scanner - бекенд-сервіс, що обробляє дані з джерел типу "база даних". Архітектура мікросервісу побудована на гнучких інтерфейсах реалізуючи які, можна отримати підтримку різних баз даних. Для демонстрації буде реалізовано підтримка баз даних PostgreSQL і MySQL.

Для підтримки взаємодії між мікросервісами шина даних буде включати в себе такі канали для повідомлень:

data_source_command – відповідає за повідомлення з командами для керування процесом аналізу джерел даних. Реалізація включає в себе 2 команди, start і stop.

Контракт повідомлення команди “start”

```
{
  "command": "start",
  "dataSourceId": "c02d04d8-7107-48a1-a87e-28bf11492b7e",
  "dataSourceType": "database",
  "product": "postgresql",
  "connectionProperties": {
    "type": "database",
    "jdbcUrl": "jdbc:postgresql://localhost:5432/test_db",
    "credentials": {
      "type": "basic",
      "username": "user",
      "password": "password"
    }
  }
}
```

Контракт повідомлення команди “stop”

```
{
  "command": "stop",
  "dataSourceId": "c02d04d8-7107-48a1-a87e-28bf11492b7e",
  "dataSourceType": "database",
  "product": "postgresql"
}
```

data_source_status – відповідає за повідомлення з оновленням статусу аналізу джерела даних.

```
{
  "dataSourceId": "c02d04d8-7107-48a1-a87e-28bf11492b7e",
  "analysisStatus": "DONE",
  "analysisAt": 1707343521
}
```

database_classification – відповідає за повідомлення з результатами класифікації баз даних.

```
{
  "dataSourceId": "c02d04d8-7107-48a1-a87e-28bf11492b7e",
  "schema": "test_schema",
  "tables": [
    {
      "name": "business_cust_diginfo",
      "columns": [
        {
          "name": "bc_id",
          "ordinalPosition": 1,
          "sqlType": "NUMERIC",
          "entityType": "OTHER"
        },
        {
          "name": "email",
          "ordinalPosition": 3,
          "sqlType": "VARCHAR",
          "entityType": "EMAIL_ADDRESS"
        },
        {
          "name": "phone_number",
          "ordinalPosition": 8,
          "sqlType": "VARCHAR",
          "entityType": "PHONE_NUMBER"
        }
      ]
    }
  ]
}
```

Якщо розглядати систему високорівнево, то взаємодія з нею передбачає три ключові процеси: додавання нових джерел даних, їх подальший аналіз, та

огляд результатів класифікації цих джерел. На рисунках 3.2.3., 3.2.4. і 3.2.5 зображена високорівнева діаграма послідовності цих процесів.

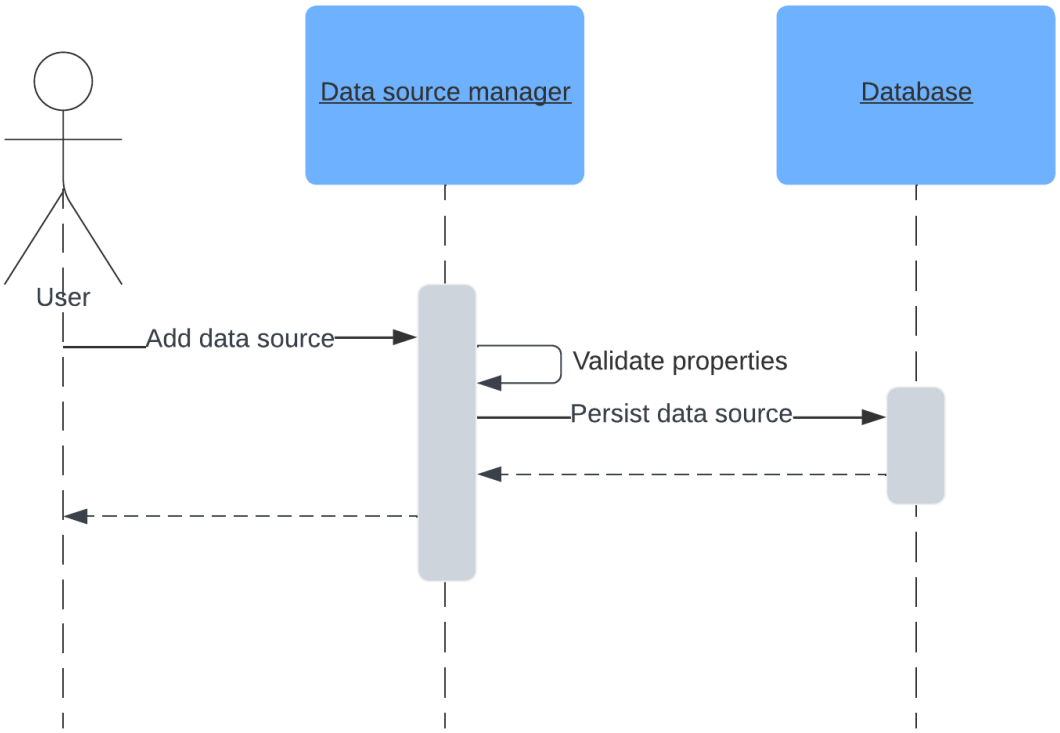

Рис. 3.2.3. Високорівнева діаграма послідовності додавання джерела даних.

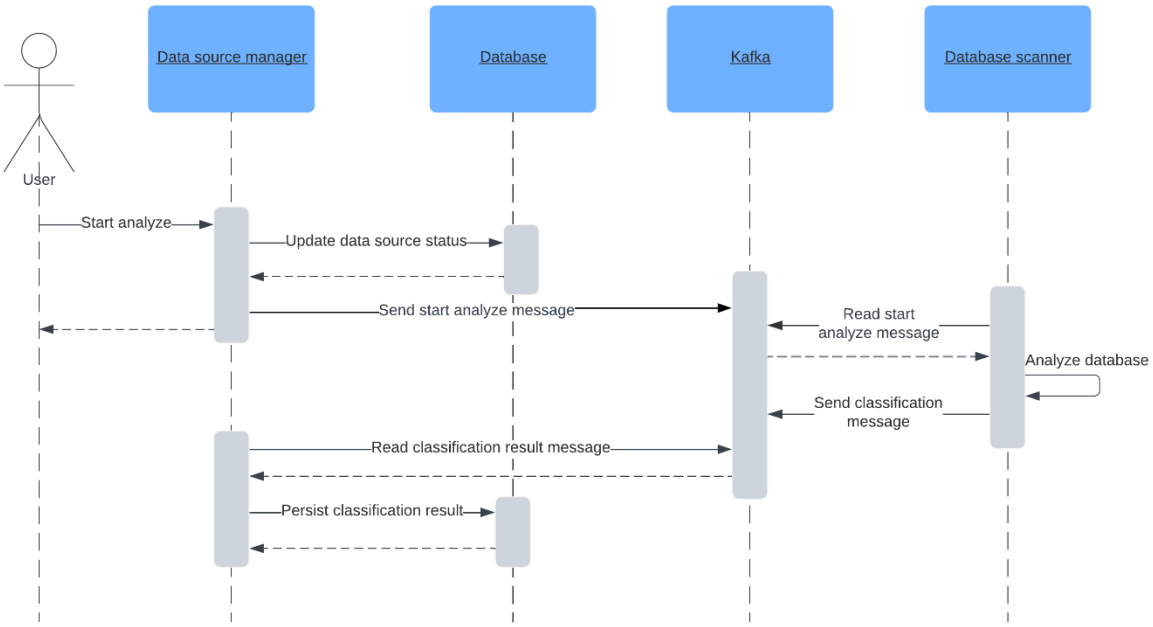

Рис. 3.2.4. Високорівнева діаграма послідовності класифікації джерела даних.

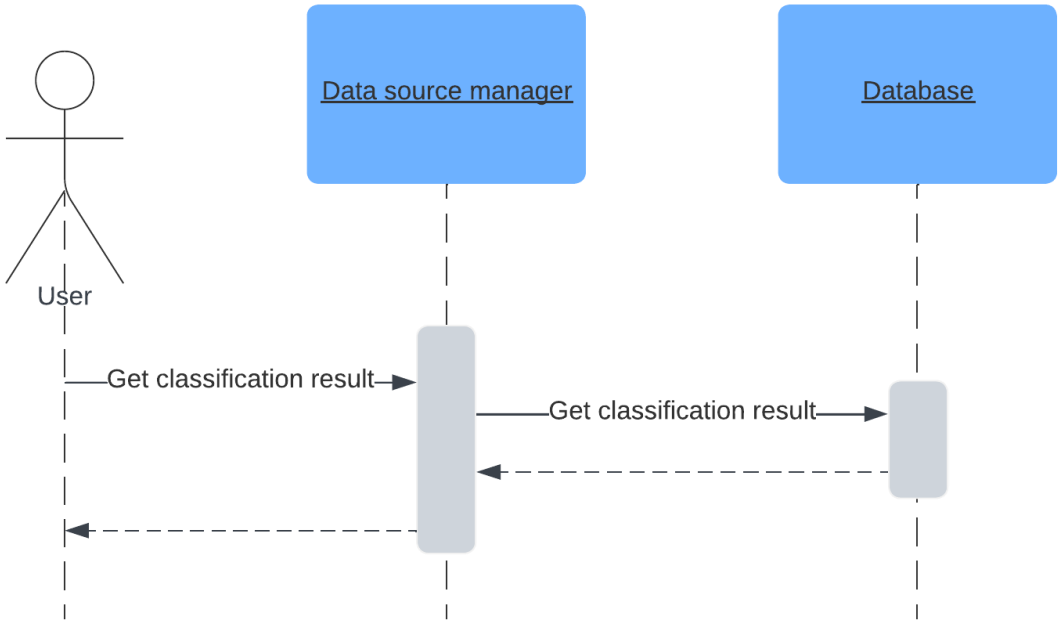

Рис. 3.2.5. Високорівнева діаграма послідовності результатів класифікації.

3.3. Розробка інтерфейсу користувача

Інтерфейс користувача (ІК) є одним з ключових компонентів будь-якої системи, адже він забезпечує взаємодію користувача з системою. У цьому розділі буде описано проектування ІК для системи пошуку персональної інформації, який буде відповідати наступним основним функціональним можливостям:

- Відображення списку джерел даних: ІК буде показувати список доступних джерел даних, з яких користувач може вибрати те, з яким він хоче працювати.
- Спливаюче вікно додавання/редагування джерела даних: ІК буде мати спливаюче вікно, де користувач може додати нове джерело даних або редагувати існуюче.
- Відображення результатів класифікації для обраного джерела: ІК буде показувати результати класифікації даних для джерела, яке вибрав користувач.

Детальний опис кожної з цих функціональних можливостей буде надано далі в цьому розділі.

Відображення списку джерел даних

Список джерел даних, рис. 3.3.1., буде представлено у вигляді таблиці, де кожен рядок буде відповідати одному джерелу даних. Таблиця буде містити наступні стовпці:

- Id: Ідентифікатор джерела даних.
- Name: Назва джерела даних визначне користувачем.
- Type: Тип джерела даних (наприклад Database, Central Storage, Cloud Storage).
- Product: Тип продукту джерела даних (наприклад PostgreSQL, OneDrive, Amazon S3).
- Analysis Status: Статус аналізу джерела даних.

Користувач буде мати можливість додавати або видаляти джерело даних, продивлятись статус аналізу, управляти процесом аналізу а також продивлятись його результат.

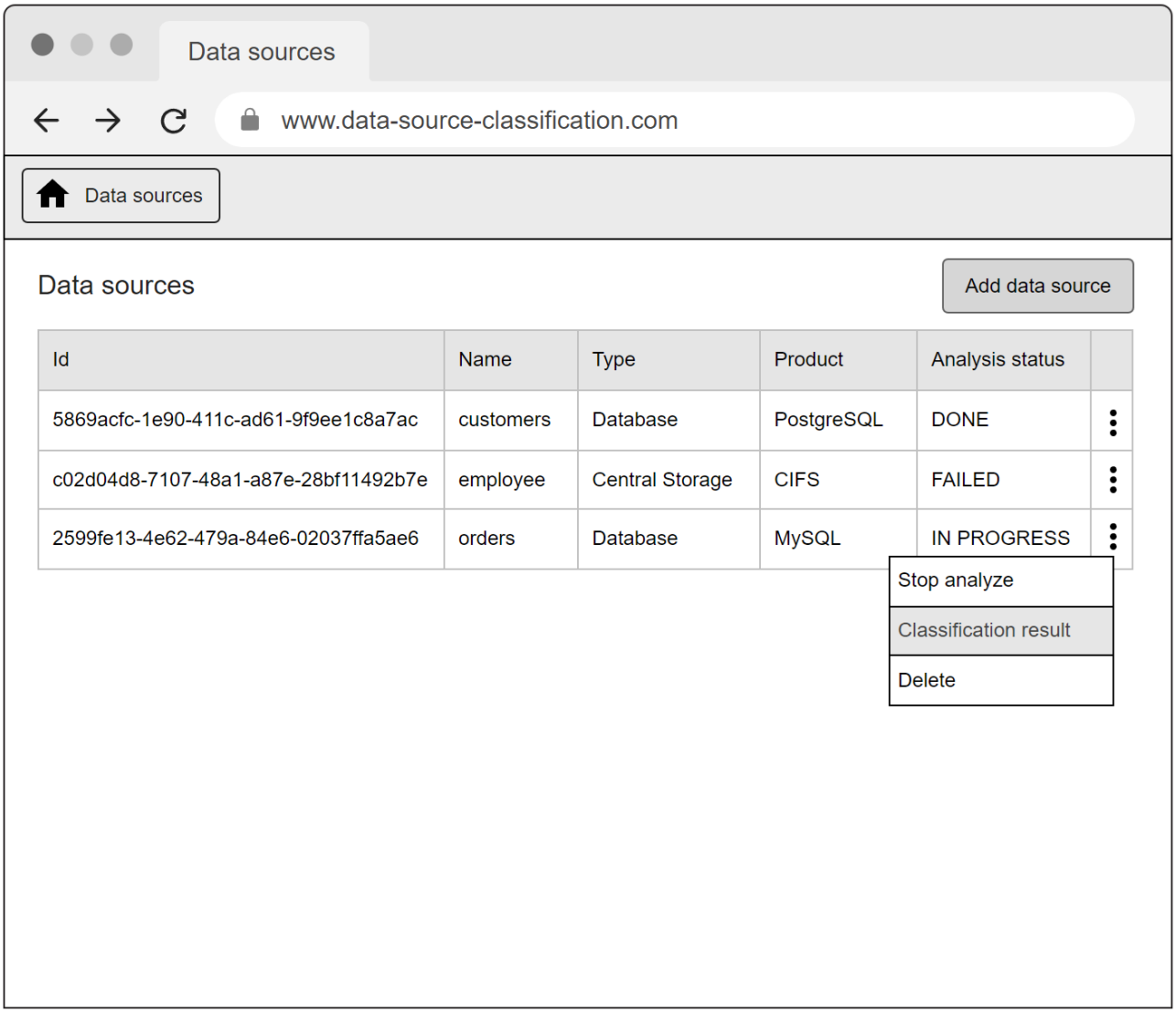

Рис. 3.3.1. Макет списку джерел даних.

Спливаюче вікно додавання/редагування джерела даних

Спливаюче вікно, рис. 3.3.2., буде містити поля для введення наступної інформації джерела даних:

- Name: Назва джерела даних.
- Type: Тип джерела даних (наприклад Database, Central Storage, Cloud Storage).
- Product: Тип продукту джерела даних (наприклад PostgreSQL, OneDrive, Amazon S3).
- Connection type: Тип підключення (Наприклад підключення до баз даних за допомогою Jdbc URL)

Поля для ведення облікових даних будуть відображатись в залежності від обраного типу підключення. У випадку з підключення до бази зданих за допомогою URL форма буде містити наступні поля:

- Jdbc URL: URL-адреса JDBC для встановлення зв’язку між нашим додатком Java і базою даних.
- Username: Ім’я користувача облікового запису
- Password: Пароль облікового запису

Користувач буде мати можливість зберегти або скасувати зміни.

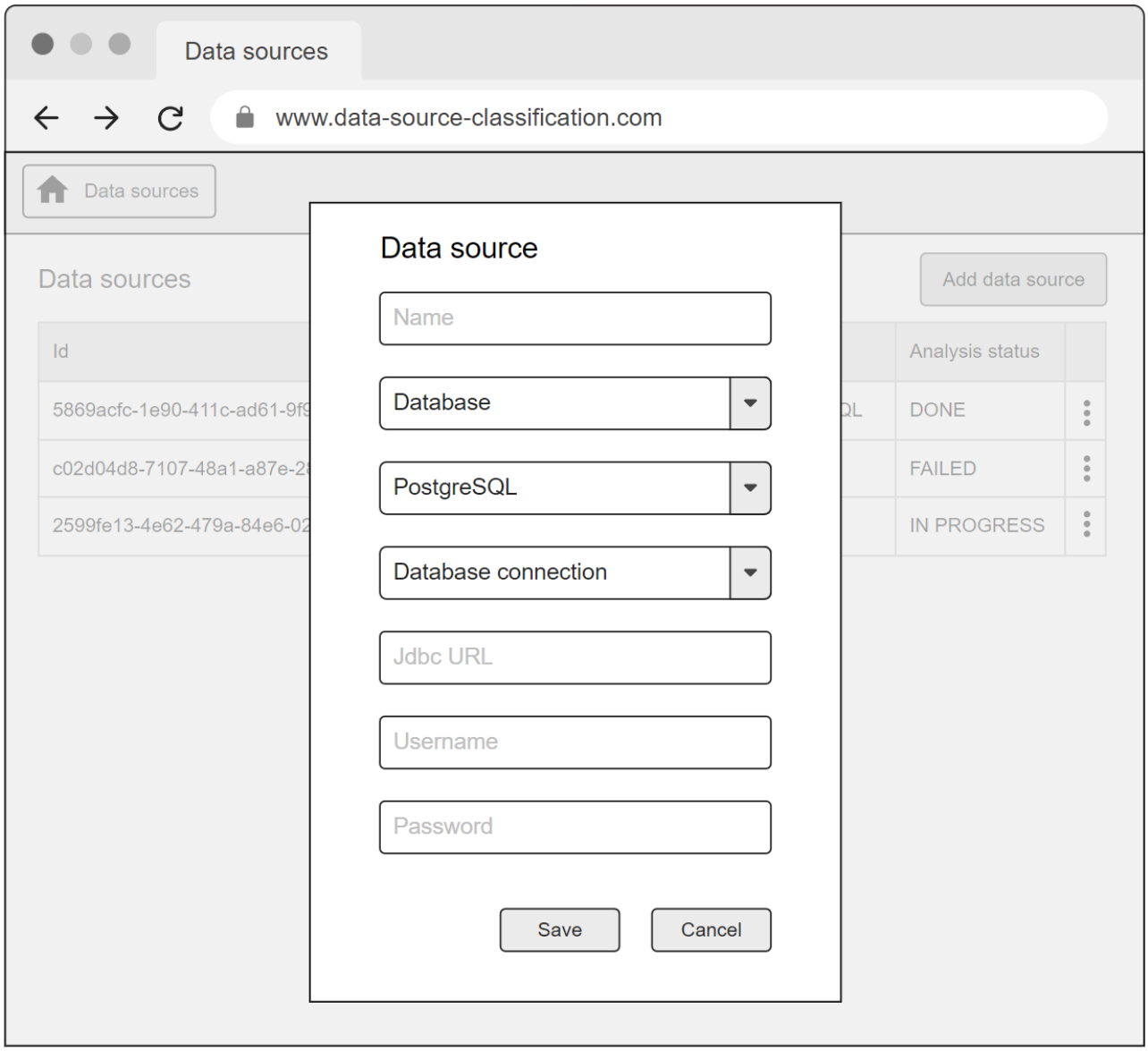

Рис. 3.3.2. Макет додавання/редагування джерела даних.

Відображення результатів класифікації для джерела типу база даних

Результати класифікації даних для джерела типу база даних, рис. 3.3.3., будуть представлені у вигляді таблиці з вкладками, де кожна вклада буде відповідати одній таблиці в схемі або каталозі бази даних. Таблиця буде містити інформації про класифікацію колонок таблиці. Вона буде містити наступні поля:

- Name: Назва колонки.
- SQL Type: Тип колонки у форматі SQL.
- Entity type: Класифікований тип даних.

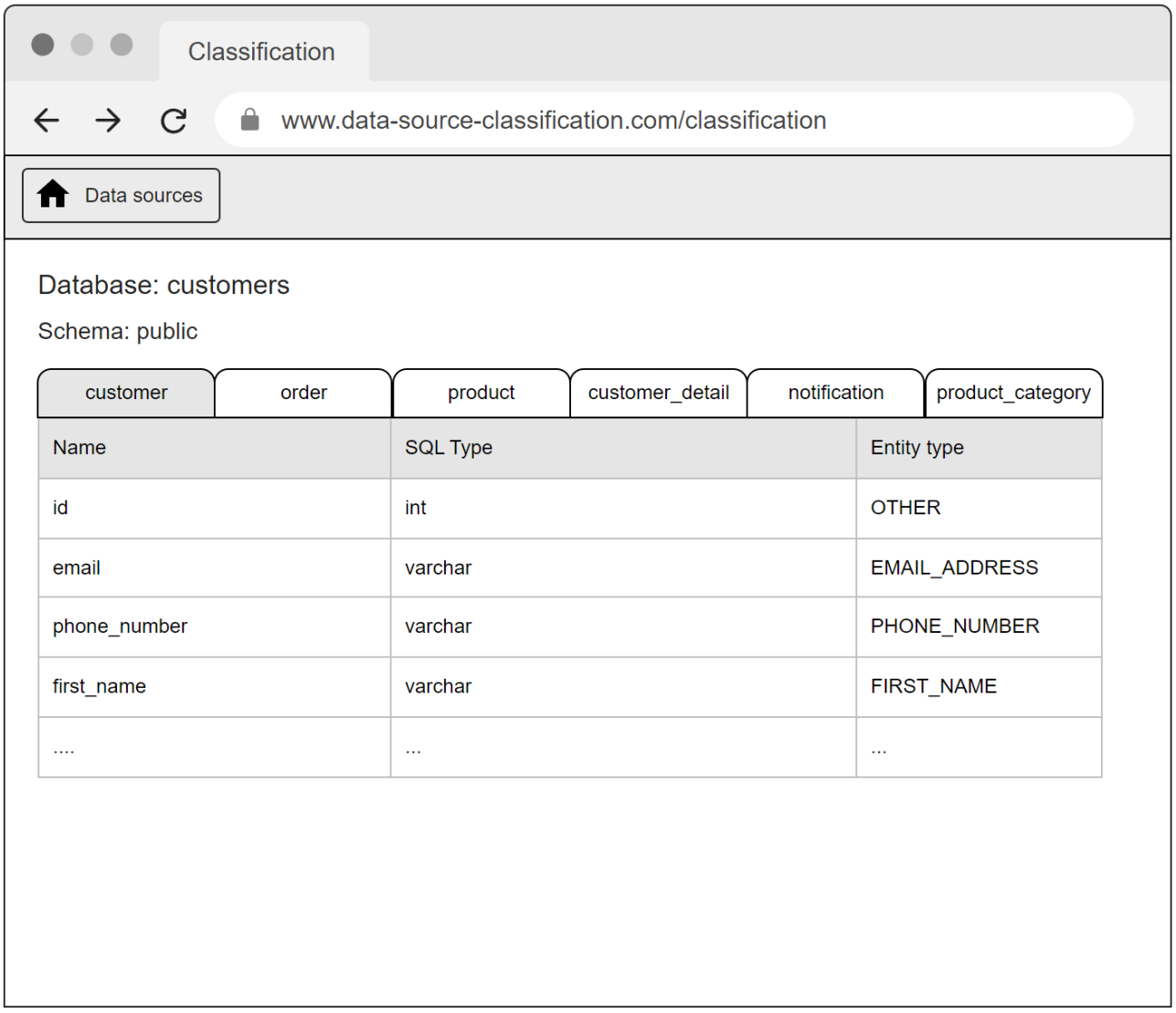

Рис. 3.3.3. Результати класифікації даних.

Окрім вищезазначених функціональних можливостей, ІК також буде мати:

- Панель навігації, яка буде дозволяти користувачу переходити до різних розділів системи.
- Панель інструментів, яка буде містити кнопки для виконання різних дій, таких як запуск класифікації даних, експорт результатів класифікації тощо.

Дизайн ІК буде простим, зрозумілим та зручним для користування.

3.4. Реалізація серверної частини

У цьому розділі ми зосередимося на створенні стійкого фундаменту для нашого веб-застосунку, використовуючи Java як основну мову програмування та Spring Framework як основний фреймворк для побудови серверної частини.

Відповідно до розробленої високорівневої архітектури реалізуємо два основні мікросервіса, data source manager і database scanner.

Сервіс Data Source Manager

Data Source Manager - це компонент системи, який відповідає за управління джерелами даних та аналіз результатів їх класифікації. Він використовується для:

- Централізованого доступу до інформації про джерела даних.
- Управління процесами аналізу даних.
- Зберігання та візуалізації результатів класифікації.

Відповідно до визначеної архітектури, на 4 чітко визначені рівні:

- Рівень контролера
- Сервісний рівень

- Рівень репозиторія
- Рівень обміну повідомленнями.

Рівень контролера обробляє HTTP-запити від користувачів і координує взаємодію користувача з сервісом. До нього входить клас DataSourceController.

DataSourceService і DatabaseClassificationService виступає як сервісний шар, який містить бізнес-логіку яка відповідає за додавання, оновлення та видалення джерел даних, запуск та зупинка аналізу, обробку та отримання результатів класифікації.

DatabaseClassificationService та DatabaseClassificationRepository належать до рівня репозиторію, вони відповідають збереження та отримування даних про джерела даних та класифікації в базі даних.

DataSourceCommandMessagingService у рівні обміну повідомленнями управляє комунікацією з іншими сервісами або компонентами через протоколи обміну повідомленнями, у цьому випадку використовуючи топіки Kafka. Основною задачою цього класа є відправка команд для різних сканерів. Слухачі, такі як DataSourceStatusListener та DatabaseClassificationListener, є компонентами, що реагують на події, забезпечуючи відповідність та актуальність системи.

Діаграма класів на рисунку 3.4.1. детально ілюструє структуру та взаємозв'язки між різними компонентами системи, виокремлюючи рівні відповідальності та забезпечуючи зрозуміле уявлення про архітектурну організацію веб-застосунку.

Рис. 3.4.1. Діаграма класів сервіса data source manager.

Сервіс Database Scanner

Database scanner - це сервіс, який автоматично класифікує дані в базах даних. За допомогою гнучкої архітектури і визначених інтерфейсів він забезпечує можливість роботи з різними базами даних, і надає можливість роз.

На рис. 3.4.3. зомражена діаграма класів, яка дає чітке розуміння як компоненти сервіса взаємодіють один з одним.

Основною точкою входу є слухач DataSourceCommandListener, який реагує на командні події та управляє процесом класифікації за допомогою цих подій. Реалізації AbstractDataSourceCommandProcessor допомагають визначити, як реагувати на команду, і відфільтровують команди, які не входять до зони їх відповідальності. Наприклад, StartDataSourceCommandProcessor відповідає за команду старту класифікації бази даних та оновлення статусу для відповідного джерела даних.

DatabaseClassificationService за допомогою DatabaseScannerProvider знаходить відповідний до типу бази даних побудовник з'єднань і будує на його основі відповідний сканер.

Реалізація DatabaseScanner є однією з ключових для цього сервісу. Відповідно до інтерфейсу DatabaseScanner, вона має надати можливість:

- Отримувати всі схеми або каталоги бази даних.
- Отримувати структуру бази даних з визначеними типами і назвами елементів.
- Отримувати зразки для кожної колонки відповідної таблиці і схеми.

RecognizerFacade інкапсулює в собі всі реалізації Recognizer і відповідає за класифікацію зразків бази даних. Кожна реалізація Recognizer в свою чергу відповідає за реалізацію класифікації конкретного типу сутності. Наприклад, PhoneNumberRecognizer відповідає за визначення номеру телефону, а EmailRecognizer за визначення email-адрес.

RecognizerFacade аналізує кожен зразок всіма доступними розпізнавачами і відфільтровує результати відповідно до зазначеного порогового значення. З залишених результатів обирається класифікація з найвищою оцінкою. На рис. 3.4.2. зображено детально роботу алгоритму класифікації даних.

Після того як отримано всі результати, результат структури бази даних і результат класифікації обєднуються і відправляються у якості повідомлення за допомогою DatabaseClassificationMessagingService.

Діаграма послідовності на рис. 3.4.4. дає чітке уявлення про процес класифікації даних і допомагає зрозуміти, як всі ці етапи взаємопов'язані і як вони впливають на загальний процес класифікації даних.

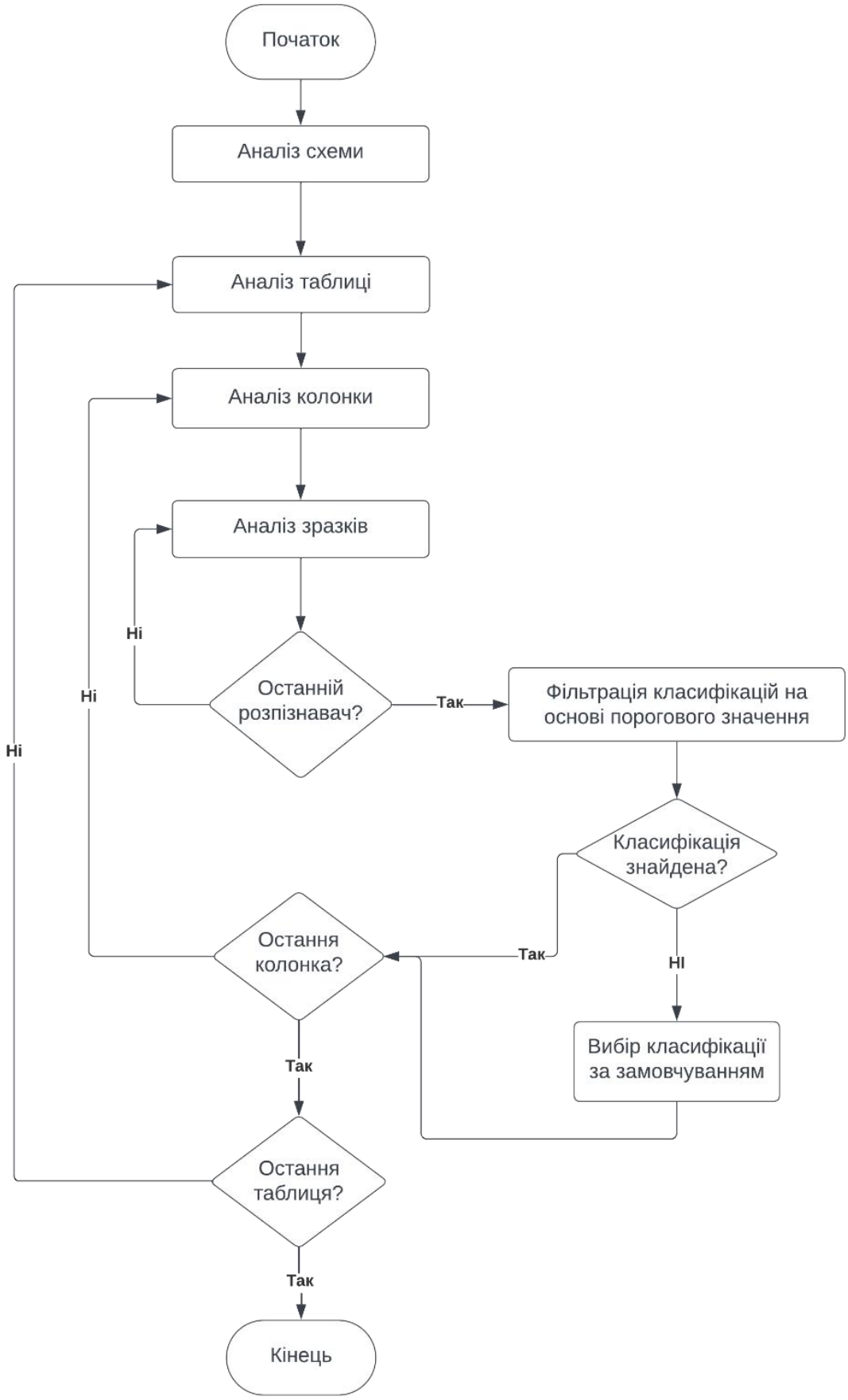

Рис. 3.4.2. Алгоритм класифікації даних.

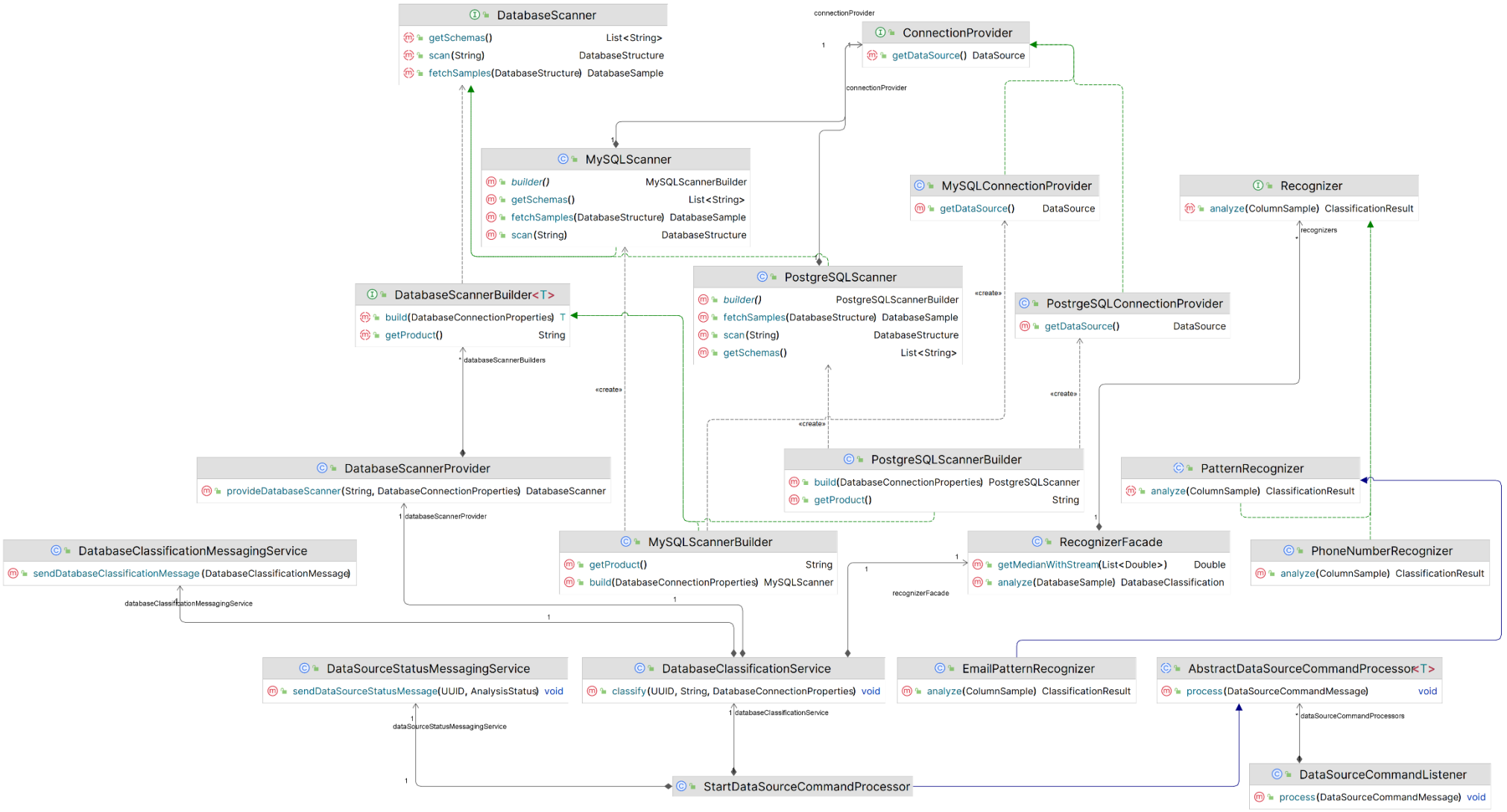

Рис. 3.4.3. Діаграма класів сервіса database scanner.

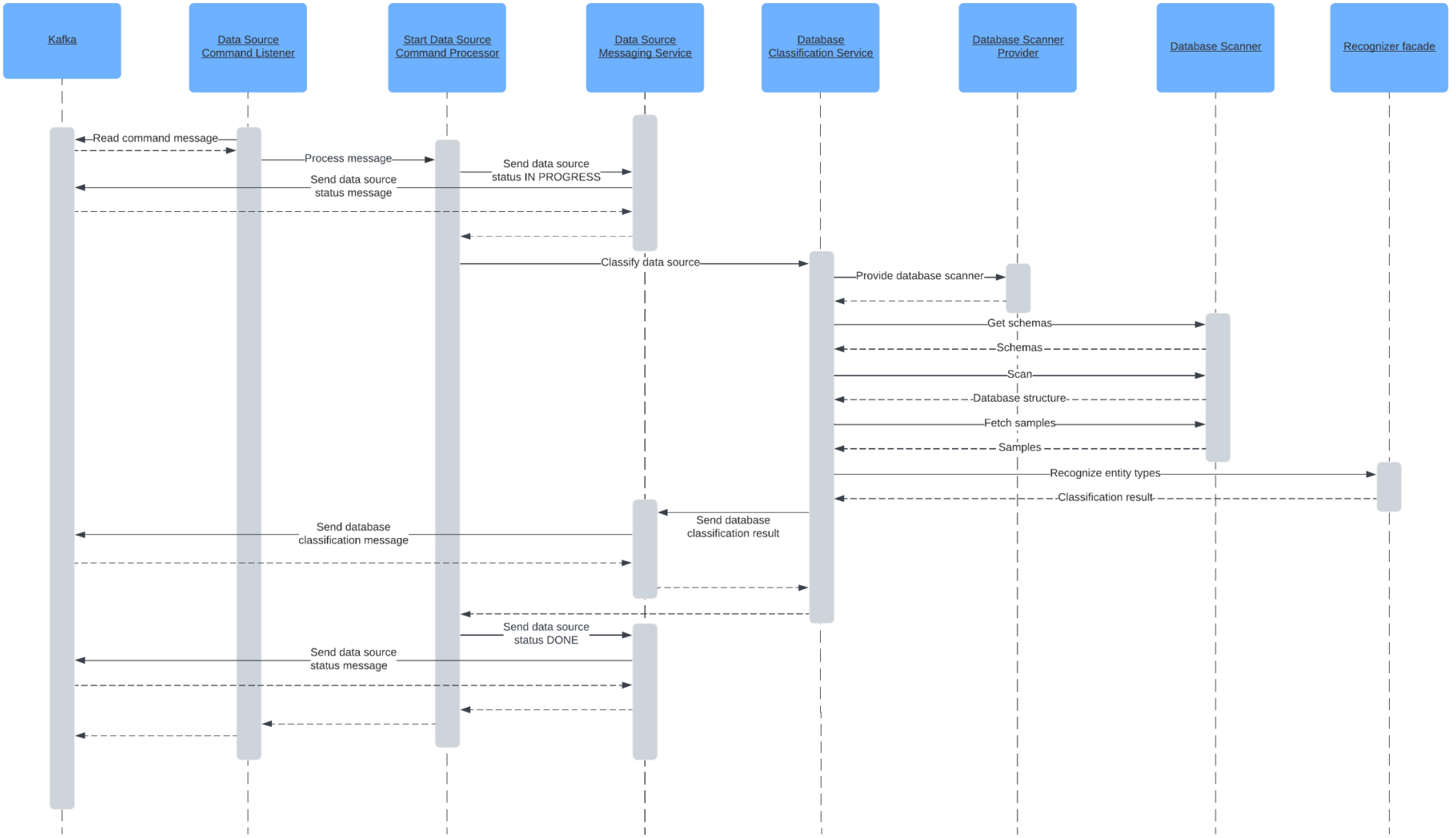

Рис.3.4.4. Діаграма послідовності класифікації даних.

Висновки до розділу 3.

Процес реалізації веб-застосунку виявив значний вплив вибору технологій на всі аспекти системи. Застосування Angular, Java, Spring, Kafka та PostgreSQL створило фундамент для надійного, безпечного продукту, який легко масштабується. Мікросервісна архітектура внесла гнучкість у розвиток проекту, дозволяючи керувати окремими частинами системи незалежно, що сприяє швидкій адаптації до змін.

Інтеграція різних сервісів і компонентів через шину подій Kafka та REST API виявила себе як ефективний спосіб забезпечення високої продуктивності і стабільності веб-застосунку. Це підтвердило вибір архітектури та підхід до розробки, які дозволяють системі гнучко реагувати на змінні обсяги даних та запитів.

Розроблений веб-застосунок, хоча й володіє обмеженим набором функцій, завдяки вдалій архітектурі здатен швидко розвиватися та розширюватися. Існуючі інтерфейси дозволяють легко додавати нові функції, роблячи систему гнучкою та адаптивною до потреб користувачів, що дозволяє їй стати ще більш цінним інструментом для роботи з даними.

РОЗДІЛ 4.
АНАЛІЗ ТА УЗАГАЛЬНЕННЯ РЕЗУЛЬТАТІВ РОБОТИ ВЕБ-ЗАСТОСУНКА

4.1. Тестування веб-застосунка

4.2. Напрямки розвитку роботи алгоритму

Висновки до розділу 4.

ВИСНОВКИ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ponemon Institute LLC. Cost of a Data Breach Report [Електронний ресурс] / Ponemon Institute LLC. – 2023. – Режим доступу до ресурсу: https://www.ibm.com/reports/data-breach.
2. Lê Đức Phong. Data Breach: Analysis, Countermeasures, and Challenges [Електронний ресурс] / Lê Đức Phong, Haruna Isah, Sajjad Dadkhah // International Journal of Information and Computer Security. – 2023. – Режим доступу до ресурсу: https://www.researchgate.net/publication/366761866_Data_Breach_Analysis_Countermeasures_and_Challenges.
3. European Parliament. REGULATION (EU) 2016/679 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL [Електронний ресурс] / European Parliament, Council of the European Union. – 2016. – Режим доступу до ресурсу: https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32016R0679.
4. Brown J. Assembly Bill No. 375 [Електронний ресурс] / Jerry Brown // California Civil Code. – 2018. – Режим доступу до ресурсу: https://leginfo.legislature.ca.gov/faces/billTextClient.xhtml?bill_id=201720180AB37.
5. Belen Saglam R. Personal information: Perceptions, types and evolution [Електронний ресурс] / R. Belen Saglam, J. R.C. Nurse, D. Hodges // Journal of Information Security and Applications. – 2022. – Режим доступу до ресурсу: https://www.sciencedirect.com/science/article/pii/S2214212622000503.
6. Schwartz P. The PII Problem: Privacy and a New Concept of Personally Identifiable Information [Електронний ресурс] / P. Schwartz, D. Solove // New York University Law Review. – 2011. – Режим доступу до ресурсу: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=1909366.
7. Structured vs. Unstructured Data: What's the Difference? [Електронний ресурс] // IBM Cloud Education. – 2021. – Режим доступу до ресурсу: https://www.ibm.com/blog/structured-vs-unstructured-data.
8. Somchart F. Enabling Efficient Personally Identifiable Information Detection with Automatic Consent Discovery [Електронний ресурс] / F. Somchart, S. Pattavee // Vol. 17 No. 2: ECTI Transactions on CIT. – 2023. – Режим доступу до ресурсу: https://doi.org/10.37936/ecti-cit.2023172.252270.
9. Ikechukwu O. Developing Smart Web-Search Using RegExSING REGEX [Електронний ресурс] / O. Ikechukwu, O. Stanley, O. Ebele // Vol 11. International Journal on Natural Language Computing. – 2022. – Режим доступу до ресурсу: https://aircconline.com/ijnlc/V11N3/11322ijnlc03.pdf.

10. Brown C. Simple and Efficient Identification of Personally Identifiable Information on a Public Website [Електронний ресурс] / C. Brown, C. Morisset // IEEE International Conference on Big Data. – 2022. – Режим доступу до ресурсу: https://ieeexplore.ieee.org/document/10020584.