НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
(повна назва інституту/факультету)

Кафедра автоматики та управління в технічних системах
(повна назва кафедри)

«На правах рукопису»
УДК 681.03

«До захисту допущено»

Завідувач кафедри
_____ Ролік О. І.
(підпис) (ініціали, прізвище)

“ 17 ” грудня 2018 р.

Магістерська дисертація

зі спеціальності (спеціалізації): 121 «Інженерія програмного забезпечення»
(код і назва спеціальності)

на тему: Система автоматизованого пошуку інформації фармацевтичного характеру

Виконав: студент 2 курсу, групи ІТ-73МП
(шифр групи)

Корнілов Іван Станіславович _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник к.т.н, доцент каф. АУТС, Дорогий Я. Ю. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (науковий ступінь, вчене звання, , прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2018 року

РЕФЕРАТ

Магістерська дисертація на тему «Система автоматизованого пошуку інформації фармацевтичного характеру» містить 100 аркушів пояснювальної записки, 44 таблиці, 16 рисунків, 8 креслеників та 20 бібліографічних посилань на використані джерела.

Актуальність теми магістерської дисертації зумовлена широкою популярністю систем контролю обліку фармацевтичної продукції, які користуються значним попитом серед компаній виробників лікарських засобів та державних установ у сфері охорони здоров'я та необхідністю у пошуку інформації по лікарських засобах у автоматичному режимі.

Метою роботи є розробка системи, яка б дозволила спеціалістам фармацевтичних компаній налаштовувати та виконувати пошук інформації стосовно лікарських засобів в автоматичному режимі з можливістю подальшої обробки результатів пошуку.

Об'єктом дослідження є дані про лікарські засоби, які зберігаються в системах фармацевтичного нагляду.

Наукова новизна полягає в розробці системи, яка б дозволила виконувати пошукові запити в автоматичному режимі. Аналогів систем такого роду на даний момент не існує.

Результати розробки системи впроваджено до ліцензованого програмного продукту «База даних з фармаконагляду та безпеки лікарських засобів DSBase» виробництва ТОВ «ІРІС». Впровадження результатів підтверджуються актом впровадження.

Ключові слова: система, пошук, видання, лікарські засоби, служба, автоматизація.

SUMMARY

The master's thesis: “Automated search system for pharmaceuticals-related information” contains 99 sheets of explanatory note, 44 tables, 16 pictures and 20 bibliographic references on used literary sources.

The urgency of the theme of master's thesis predetermined by the wide popularity of control systems for the accounting of pharmaceutical products, which are in great demand among the companies of pharmaceutical manufacturers and state institutions in the field of health care and the need for the search for information on medicines in automatic mode.

The aim of the work is to develop a system that would allow pharmaceutical companies to customize and search for information about medicines in an automatic mode with the possibility of further processing of search results.

The subject of the research is data on medicinal products stored in pharmacovigilance systems.

The scientific novelty is to develop a system that would allow doing searches in automatic mode on. Analogues of systems of this kind do not exist now.

The results of the system development introduced into the licensed software product "Database on Pharmacovigilance and Safety of Drugs DSBase" produced by "IRIS Ltd.". Implementation of the results confirmed by the implementation act.

Key words: system, search, publication, medicines, service, automation.

ЗМІСТ

ВСТУП

Актуальність теми магістерської дисертації зумовлена широкою популярністю систем контролю обліку фармацевтичної продукції, які користуються значним попитом серед компаній виробників лікарських засобів та державних установ у сфері охорони здоров'я. Основним завданням роботи подібних систем є збереження інформації стосовно відповідності призначення лікарських засобів та результатів практичного застосування препарату.

За допомогою ефективного пошуку, збору та обробки інформації можна значно полегшити систематизацію фармацевтичної інформації для виробників та споживачів препаратів. Основним інструментом таких систем є популярні пошукові канали, на кшталт «Google», які надають можливість відслідкувати всі ресурси, які лежать в полі інтересів споживачів та виробників лікарських засобів. На сьогоднішній день спеціалісти фармацевтичних компаній виконують пошук інформації в ручному режимі. Цей процес потребує чимало ресурсного забезпечення: втрати часу, велика кількість персоналу, які могли б замітини автоматизовані системи пошуку, багато похибок, людський фактор недоопрацьованості масивів інформації, неможливість охопити увесь наявний матеріал.

Саме тому наразі існує реальна потреба в створенні автоматизованої системи пошуку, яка б надала можливість користувачам більш раціонально та ефективно розпоряджатися своїм ресурсним капіталом. В процесі обробки запиту дана система знаходить необхідну інформацію і завантажує її в базу даних фармацевтичних компаній в необхідному форматі.

Метою роботи є розробка системи, яка б дозволила спеціалістам фармацевтичних компаній налаштовувати та виконувати пошук інформації стосовно лікарських засобів в автоматичному режимі з можливістю подальшої обробки результатів пошуку.

Об'єктом дослідження є дані про лікарські засоби, які зберігаються в системах фармацевтичного нагляду.

Предметом дослідження є пошук інформації про лікарські засоби в автоматичному режимі.

В ході дослідження були поставлені наступні задачі:

- розробка компонентів автоматизації пошуку;
- розробка компонентів візуалізації результатів пошуку;
- розробка компонентів безпосереднього виконання пошукових запитів.

З усього вище сказаного однозначно зрозуміла необхідність створення автоматизованої системи пошуку інформації фармацевтичного характеру з використанням сучасних технологій розробки ПЗ.

Наукова новизна полягає в розробці системи для автоматизації пошуку інформації та її подальшої обробки при тому, що аналогів такого типу систем на момент написання роботи не існує.

Практична значимість розробленої системи полягає у застосуванні системи виробниками фармацевтичної продукції для знаходження інформації про лікарські засоби, які вони виробляють у автоматичному режимі.

Апробація результатів роботи.

Результати розробки системи впроваджено до ліцензованого програмного продукту «База даних з фармаконагляду та безпеки лікарських засобів DSBase» виробництва ТОВ «ІРІС». Впровадження результатів підтверджуються актом впровадження.

1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

Предметною областю застосування системи автоматизованого пошуку є системи, які застосовуються для обліку, аудиту та систематизації інформації про фармацевтичні препарати. Виходячи з цього необхідно зібрати та проаналізувати інформацію про існуючі системи пошуку та існуючі інформаційні системи для фармацевтичного нагляду. Також потрібно визначити точку інтеграції системи пошуку та системи фармацевтичного нагляду для їх спільної коректної роботи.

1.1 Системи фармацевтичного нагляду

Поява інформаційних систем нагляду за використанням фармацевтичних препаратів на території Уаїни була більше необхідністю, ніж просто забаганкою міністерства охорони здоров’я. Ця необхідність пояснюється появою більшої кількості постачальників лікарських препаратів і неможливістю їх контролювати в ручному режимі. Також зростають об’єми інформації, яку треба зберігати. Тому дані системи дозволили б вирішити ряд питань і проблем в сфері збору та обробки даних фармацевтичного характеру, а саме:

- усунення помилок, спричинених людським фактором;
- підвищення якості та повноти інформації;
- підвищення швидкості обробки масивів даних;
- сортування, групування інформації, пошук та аналіз необхідних даних;
- складання звітності за періоди часу.

Компанії-виробники лікарських засобів почали впроваджувати в свою роботу інформаційні системи для того, щоб вирішити питання, викладені раніше. Такого роду системи дозволити їм систематизувати та автоматизувати введення даних про лікарські засоби та здійснювати перевірку даних щодо відсутності ефективності лікарських засобів або наявність побічних реакцій, отриманих на основі звернень пацієнтів за допомогою спеціальних форм на сайті.

Однією з таких систем є система, яка на сьогоднішній день застосовується в державному експертному центрі (рис 1.1). Ця система призначена для реалізації фармацевтичного нагляду територіально-розподілених фармацевтичних компаній та їх представництв в інших країнах. [1]

Рисунок 1.1 – Головна сторінка системи з фармаконагляду

Робота з системою відбувається за допомогою веб-браузеру. Система націлена на створення даних реєстрів, тому систему можна легко кастомізувати і створювати нові реєстри достатньо легко і швидко. Таку можливість дає спеціалізована структура бази даних, яка застосовується в системі.

Система має наступний функціонал:

- ведення реєстру лікарських засобів;
- реєстрація нового лікарського засобу;
- перереєстрація лікарського засобу, внесення змін, додавання інструкції, перегляд історичності;
- реєстрація побічних реакцій;
- перегляд діючих речовин контретного лікарського засобу;
- забезпечення захищеного документообігу;
- формування звітів згідно із завантаженими шаблонами звітних форм.

1.2 Принципи роботи пошукових систем

На сьогоднішній день існує декілька потужних пошукових систем. Однією з них с пишукова система Google. Користувач бачить роботу з системою наступним чином. Він вводить ключові слова або фрази в строку пошуку пошукової системи і написка кнопку пошуку. У відповідь він отримує сторінку визначеного формату, яка складається із посилань на джерела знайденої інформації.

Стандартна пошукова система побудована на трьох основних модулях: пошуковий робот, модуль індексування контенту, модуль графічного інтерфейсу.[2]

Спочатку пошуковий робот автоматично у фоновому режимі проходить по всіх відомих йому посиланнях та знаходить не відомий йому контент. Далі такого роду контент потрапляє в модуль індексування, де він аналізується. В процесі аналізу виділяються ключові слова і вся ця інформація разом з веб-сторінкою зберігається в індексній базі даних. Індекс дозволяє швидко знаходити інформацію, що є дуже критичним в пошукових системах. Тому останній модуль пошукової системи працює вже з індексованими даними. Схема роботи пошукової системи представлена на рисунку 1.2. [3]

Рисунок 1.2 – Схема роботи пошукової системи

Коли користувач робить запит, ввівши строку запиту у вигляді ключових слів, система перевіряє індекс і видає користувачу найбільш підходящі результати з заголовком, посиланням на ресурс та анотацією. Але постає питання

порядку розстановки посилань в результуючому наборі. Таким порядком називається релевантність, тобто семантична відповідність контенту знайденого результату з заданою строкою пошуку.[4]

На сьогоднішній день комерційні компанії, які постачають послуги пошуку не надають детальної інформації щодо сучасних алгоритмів ранжування результатів пошуку. Раніше релевантність сайту визначалася кількістю ключових слів на сторінці та кількістю посилань на даний сайт з інших сайтів, які мали високу релевантність. Проте зараз такими методами майже не користуються і переходять на алгоритми визначення релевантності за допомогою відслідковування поведінки користувача. Такі пошукові системи як Google використовують наступні засоби для визначення релевантності: фіксування поведінкових факторів з аналітики, використання власного браузера Google Chrome, який теж збирає необхідну про користувача інформацію, а також із власної соціальної мережі та власних сервісів.[5]

Висновки до розділу

В даному розділі було оглянуто предметну область системи, а саме було розглянуто існуючі системи фармацевтичного нагляду, функціонал даних систем, основні типи даних, які зберігаються і обробляються в такого роду системах. Також було оглянуто опис систем пошуку, оновні компоненти, алгоритми та властивості.

2 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

Аналогів систем пошуку в автоматичному режимі у відкритому доступі не існує. Якщо такі системи існують, то вони скоріш за все закритого типу й інформації про них немає. Але можна розглянути в якості альтернатив аналоги окремих підсистем або модулів системи. Автоматизація роботи системи базується на використанні компоненту планування задач виконання пошукових запитів. Тому розглянемо існуючі аналоги компоненту планування.

2.1 Планувальник Hangfire

Hangfire – багатопотоковий та масштабований планувальник задач, побудований за принципами клієнт-серверної архітектури на стеку технологій .NET з проміжним збереженням задач в базі даних. Дана реалізація планувальника безкоштовна з відкритим вихідним кодом.[6]

Суть роботи hangfire полягає в тому, що клієнт системи додає задачу в базу даних, а сервер системи періодично опитує базу даних та виконує задачі. Задачі перед передачею в БД серіалізуються та передаються в БД в серіалізованому вигляді. Опис роботи hangfire представлений на рисунку 2.1. Виконання задачі відбувається з використанням рефлексії. Для цього потрібно, щоб була присутня збірка, в якій міститься код задачі. [6]

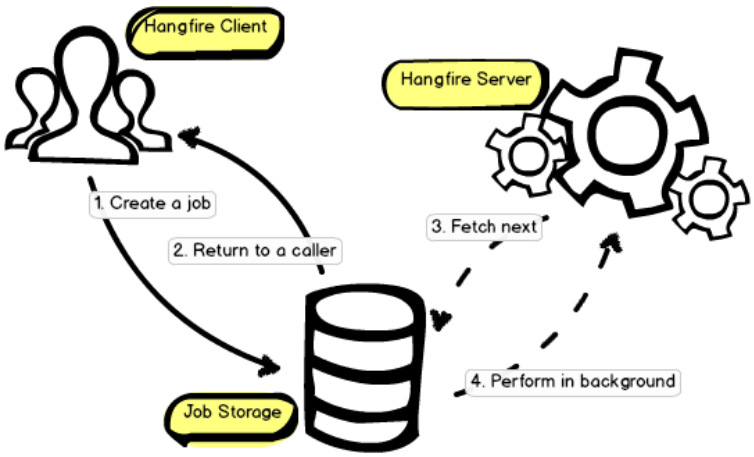


Рисунок 2.1 – Опис роботи hangfire

Основними особливостями hangfire є:

1. Задачі формуються клієнтом та зберігаються у сховищі у вигляді серелізованого коду, який потрібно виконати на стороні сервера.
2. Обидві сторони (клієнт та сервер) мають доступ до БД у режимі запису.
3. Існує можливість, за необхідності, розподілу навантаження серверної частини між екземплярами служб виконання задач (горизонтальне масштабування).
4. Необхідність сховища, для збереження задач на виконання.
5. Миттєве виконання задач відсутнє.
6. Запускати можна як служби Windows, за допомогою IIS та за допомогою контейнерів.
7. Присутня можливість використовувати вирази CRON для задавання періодичності виконання задачі.

Отже, Hangfire пропонує непоганий сервіс для виконання запланованих задач, дозволяє виконувати задачі з певною періодичністю та не обмежений засобами хостингу сервісу. Перевагами є легкість використання, зрозумілий програмний інтерфейс, можливість використання CRON виразів. Недоліками є необхідність використання бази даних та неможливість розширення кількості підписників на отримання результатів виконаної задачі.

2.2 Планувальник Quartz.NET

Планувальник Quartz.NET є більш низкорівневим рішенням для планування періодичного виконання задач різноманітного характеру. Постачається він у вигляді бібліотеки класів та позиціонується як компонент для розробки систем автоматизованого виконання задач загального характеру.

Перевагами даної реалізації є:

1. Не має жорстких обмежень по використанню таких як обов’язкова наявність сховища.
2. Просте рішення для нескладних задач.

Недоліками даної реалізації є:

1. Високий поріг входження (більш низькорівнева реалізація).
2. Використання старої моделі багатопотоковості.
3. Пропонується як компонент, але не як сервіс. Немає можливостей масштабування.

Отже, такая реалізація планування виконання задач непогана, але підходить для більш простих задач. Від розробника вимагається побудувати інфраструктуру навколо даного компоненту для можливості масштабування та можливостей моніторингу роботи.

Висновки до розділу

Система автоматизованого пошуку не має аналогів на даний момент часу. Тому було розглянуто окремі компоненти системи, а саме засоби планування виконання задач. Було розглянуто дві відомі реалізації, а саме Hangfire та Quartz.Net. Розглянуті засоби були проаналізовані, виділені основні переваги та недоліки.

3 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ СИСТЕМИ

Система призначена для виконання спеціалізованого пошуку в автоматичному та ручному режимі для отримання результатів моніторингу певного джерела інформації по заданим ключовим словам.

Система застосовується в фармацевтичних компаніях для пошуку будь-якої інформації по лікарським засобам, референтним лікарським засобам, та по діючим речовинам.

Клієнтська частина системи дозволяє користувачам створювати видання, створювати та конфігурувати пакети моніторингу, створювати та налаштовувати план-графіки, створювати результати моніторингу.

Серверна частина (бібліотека) дозволяє отримувати результати пошуку, обробляти їх, аналізувати отримані в результаті пошуку публікації, віддавати результати пошуку клієнтам, реєструвати підписки клієнтів на певні видання.

Компонент виконання роботи (в даному випадку пошук) з використанням таймеру (планувальник) може використовуватися в інших системах, де потрібно виконувати роботу в певний час.

Компонент пошуку може використовувати не обмежену кількість провайдерів пошуку (такі як Google, Bing та інші) та необмежену кількість адаптерів, через які результати пошуку віддаються клієнтам.

4 ВИЗНАЧЕННЯ ВИМОГ ДО СИСТЕМИ

4.1 Вимоги до функціоналу клієнту системи пошуку

Перелік вимог:

- забезпечити автентифікація та авторизацію користувачів в систему;
- забезпечити зберігання інформації в базі даних та занесення всіх змін і маніпуляцій в базу даних;
- можливість створення, редагування, перегляду та видалення видань;
- можливість створення, редагування, перегляду та видалення результатів моніторингу;
- можливість створення, редагування, перегляду та видалення пакетів моніторингу;
- можливість створення, редагування, перегляду та видалення план-графіків;
- можливість отримання миттєвих результатів моніторингу;
- можливість автоматизованого виконання моніторингу (в фоновому режимі);
- можливість налаштування виконання план-графіку на рік;
- зберігання всіх операцій змінення, видалення та внесення нової інформації в інформаційні об’єкти системи в історичність;
- можливість приймати результати пошуку від служби пошуку та зберігати ці результати у вигляді результатів моніторингу.

4.2 Вимоги до функціоналу електронної бібліотеки

Перелік вимог:

- забезпечити автентифікація та авторизацію користувачів в систему;

- забезпечити зберігання інформації в базі даних та занесення всіх змін і маніпуляцій в базу даних;
- можливість створення, редагування, перегляду та видалення видань;
- можливість створення, редагування, перегляду та видалення результатів пошуку;
- можливість створення, редагування, перегляду та видалення публікації, які були знайдені в результаті пошуку;
- можливість створення, редагування, перегляду та видалення правил аналізу видань;
- можливість створення, редагування, перегляду та видалення колекцій підписки на видання;
- можливість керування довідниковою інформацією;
- можливість приймати результати пошуку від служби пошуку та зберігати ці результати;
- можливість передавати на аналіз знайдені публікації та віддавати готові публікації клієнтам моніторингу.
- можливість зберігати матеріали пошуку для подальшої їх індесації.
- можливість індексувати матеріали для пошуку цих матеріалів службою пошуку.

4.3 Вимоги до функціоналу компоненту пошуку:

Перелік вимог:

- виконання миттєвого пошуку;
- виконання запланованого пошуку;
- стійкість до високого навантаження;
- можливість використання різних провайдерів пошуку;

- можливість використання декількох адаптерів віддачі результатів пошуку;
- можливість отримання вмісту веб-сторінки за її веб-адресою;
- можливість довгострокового планування пошуку.

5 ОПИС СЦЕНАРІЇВ ВИКОРИСТАННЯ СИСТЕМИ

5.1 Сценарії використання план-графіків

У даному сценарії використання діють 2 актори – це користувач системи (працівних фармацевтичної компанії) та служба планування пошукових запитів. Діаграму прецедентів сценаріїв використання план-графіків зображено у додатку Д. Опис сценаріїв використання представлено у вигляді таблиць 5.1 - 5.9

Таблиця 5.1 – Сценарій створення план-графіку

Короткий опис	Створення план-графіку моніторингу літератури
Первинні актори	Працівник фармацевтичної компанії
Передумова	Відкрита сторінка план-графіків. Користувач має права на створення план-графіків.
Післяумова	Створений план-графік відображається у загальній таблиці сторінки.
Головний сценарій	
№	**Дія**
1	Користувач натискає на кнопку «Створити план-графік».
2	Система перенаправляє користувача на сторінку створення план-графіку.
3	Користувач заповнює всі необхідні поля сторінки.
4	Користувач натискає на кнопку «Створити».
5	Система перенаправляє користувача на головну сторінку план-графіків, де в таблиці видно створений запис.

Таблиця 5.2 – Сценарій копіювання план-графіку

Короткий опис	Копіювання план-графіку моніторингу літератури з вже існуючого в системі
Первинні актори	Працівник фармацевтичної компанії
Передумова	Відкрита сторінка план-графіків. Інсує хоч один запис в таблиці. Користувач має права на копіювання план-графіків.
Післяумова	Копію план-графіку відображається у загальній таблиці сторінки.
Головний сценарій	
№	**Дія**
1	Користувач обирає в таблиці запис, який треба скопіювати.
2	Система перенаправляє користувача на сторінку створення план-графіку.
3	Користувач змінює, якщо необхідно поля запису.
4	Користувач натискає на кнопку «Створити».
5	Система перенаправляє користувача на головну сторінку план-графіків, де в таблиці видно скопійований запис.

Таблиця 5.3 – Сценарій видалення план-графіку

Короткий опис	Видалення план-графіку моніторингу літератури.
Первинні актори	Працівник фармацевтичної компанії
Передумова	Відкрита сторінка план-графіків. Інсує хоч один запис в таблиці. Користувач має права на видалення план-графіків.
Післяумова	Видалений план-графіку не відображається у загальній таблиці сторінки.
Головний сценарій	
№	**Дія**
1	Користувач обирає в таблиці запис, який треба видалити.
2	Система запитує у користувача підтвердження наміру видалити план-графік.
3	Користувач погоджується на видалення запису.
4	Система оновлює таблицю на сторінці, в якій вже не буде видаленого запису.

Таблиця 5.4 – Сценарій активування план-графіку

Короткий опис	Активування план-графіку моніторингу для запуску сеансів пошуку.
Первинні актори	Працівник фармацевтичної компанії, служба планування.
Передумова	Відкрита сторінка план-графіків. Інсує хоч один запис в таблиці. Користувач має права на активацію план-графіків.
Післяумова	План-графік активовано. Пошукові запити зареєстровано у службі планування. План-графік має статус «активований» та колір запису «зелений».
Головний сценарій	
№	**Дія**
1	Користувач обирає в таблиці план-графік, який треба активувати та натискає кнопку «Активувати».
2	Система пропонує користувачу обрати параметри пошуку.
3	Користувач налаштовує параметри пошуку та натискає на кнопку «активувати».
4	Система створює пошукові запити в базі даних.
5	Пошукові запити відправляються на службу планування для реєстрації.
6	Таблиця план-графіків оновлюється, запис змінює статус на «активований» та змінює колір на «зелений».

Таблиця 5.5 – Сценарій деактивування план-графіку

Короткий опис	Деактивування план-графіку моніторингу для припинення сеансів пошуку.
Первинні актори	Працівник фармацевтичної компанії, служба планування.
Передумова	Відкрита сторінка план-графіків. Інсує хоч один запис в таблиці. Користувач має права на деактивацію план-графіків.
Післяумова	План-графік деактивовано. Пошукові запити видалено в службі планування. План-графік має статус «деактивований» та колір запису «синій».
Головний сценарій	
№	**Дія**
1	Користувач обирає в таблиці план-графік, який треба деактивувати.
2	Користувач натискає кнопку «Деактивувати»
3	Система надсилає службі планування повідомлення про видалення пошукових запитів.
4	Користувач отримує повідомлення про успішну деактивацію план-графіку.
5	Таблиця план-графіків оновлюється, запис змінює статус на «деактивований» та змінює колір на «синій».

Таблиця 5.6 – Сценарій виконання миттєвого пошуку

Короткий опис	Виконання миттєвого пошуку у реальному часі.
Первинні актори	Працівник фармацевтичної компанії, служба планування.
Передумова	Відкрита сторінка план-графіків. Інсує хоч один запис в таблиці. Користувач має права на виконання миттєвого пошуку.
Післяумова	Повідомлення про успішну операцію.
Головний сценарій	
№	**Дія**
1	Користувач обирає в таблиці план-графік, по якому треба виконати миттєвий пошук.
2	Користувач натискає кнопку «Виконати зараз».
3	Система пропонує користувачу обрати параметри пошуку.
4	Користувач обирає необхідні параметри пошуку і натискає кнопку «Виконати зараз».
5	Система надсилає пошукові запити на службу планування.
6	Інтерфейс користувача блокується на час виконання пошуку.
7	Після успішного виконання, користувач отримує відповідне повідомлення.

Таблиця 5.7 – Сценарій перегляду пакетів моніторингу

Короткий опис	Перегляд пакетів моніторингу пов’язаних з план-графіком та сеансами пошуку.
Первинні актори	Працівник фармацевтичної компанії.
Передумова	Відкрита сторінка план-графіків. Інсує хоч один запис в таблиці. Користувач має права на перегляд пакетів моніторингу.
Післяумова	Користувач отримує необхідну інформацію про пакет моніторингу.
Головний сценарій	
№	**Дія**
1	Користувач обирає в таблиці план-графік, пакети якого він хоче перегдянути.
2	Користувач натискає кнопку «Перегляд пакетів моніторингу».
3	Система виводить діалог, в якому відображена таблиця з усіма пов’язаними пакетами моніторингу.

Таблиця 5.8 – Сценарій додавання сеансів пошуку

Короткий опис	Додавання сеансів пошуку до існуючого план-графіку.
Первинні актори	Працівник фармацевтичної компанії.
Передумова	Відкрита сторінка план-графіків. Інсує хоч один запис в таблиці. Користувач має права на редагування план-графіку.
Післяумова	До план-графіку додано необхідну кількість сеансів пошуку.
Головний сценарій	
№	**Дія**

1	Користувач обирає в таблиці план-графік, до якого необхідно додати сеанс пошуку.
2	Користувач натискає кнопку «Редагувати план-графік».
3	Система перенаправляє користувача на сторінку редагування план-графіку.
4	Користувач натискає кнопку «Додати сеанс пошуку».
5	Система виводить діагог створення новго запису сеансу пошуку.
6	Користувач заповнює всі необхідні поля та натискає кнопку «Створити».
7	Користувач натискає кнопку «Зберегти» на сторінці редагування план-графіку.

Таблиця 5.9 – Сценарій експорту план-графіків в Excel

Короткий опис	Експорт у формат Excel всіх план-графіків відображених у таблиці.
Первинні актори	Працівник фармацевтичної компанії.
Передумова	Відкрита сторінка план-графіків. Інсує хоч один запис в таблиці. Користувач має права на експорт план-графіків у формат Excel.
Післяумова	Користувач отримує файл формату Excel з всією інформацією про план-графіки.
Головний сценарій	
№	**Дія**
1	Користувач натискає над таблицею нкопку «Експорт в Excel».

2	Система виводить діалог, у якому пропонує користувачу обрати ім’я та розташування збереженого файлу.
3	Користувач натискає кнопку діалогу «Зберегти».

Висновки до розділу

В даному розділі було описано сценарії використання план-графіків у системі. Опис проводився згідно з діаграмою прецедентів у додатку Д. Було описано 9 сценаріїв використання план-графіків користувачем.

6 ВИБІР ТЕХНОЛОГІЙ ТА РЕАЛІЗАЦІЙ ПЗ

Керуючись власним досвідом та знаннями у сфері розробки програмного забезпечення та корелюючи їх із завданням та вимогами до системи було обрано .Net орієнтований стек технолоній, а саме:

- платформа .Net Framework 4.5 у якості середовища виконання програмного коду;
- мова програмування C#;
- система упраління БД MSSQL Server;
- у якості хостингу обрано IIS;
- для реалізації веб-додатків обрано Asp.Net MVC;
- у якості реалізації служб обрано технологію WCF;

6.1 Платформа .NET 4.5

Microsoft .NET framework представляє собою програмну платформу для створення програмних продуктів різної складності та різного призначення. В основі платформи лежить середовище виконання CLR (Common Language Runtime). Особливістю платформи є те, що вона підтримує різні мови програмування, такі як C#, Visual Basic, F#.[7]

Платформа має широку бібліотеку базових класів. Кожен додаток, створений з використанням платформи .NET складається зі збірок. Збірки складаються з керуємих модулів. В керуємому модулі присутні метадані, які описують всі типи даних та метадані, які описують члени типів даних. Кожна збірка має маніфест, який описує набір файлів, які входять в збірку.[7]

Середовище виконання оперує інструкціями CIL коду. Компілятори таких мов як C# та Visual Basic компілюють вихідний код в проміжний код, яким оперує CLR. JIT компілятор призначений для компіляції CIL коду в машинний код. Модель виконання коду представлена на рисунку 5.1.[7]

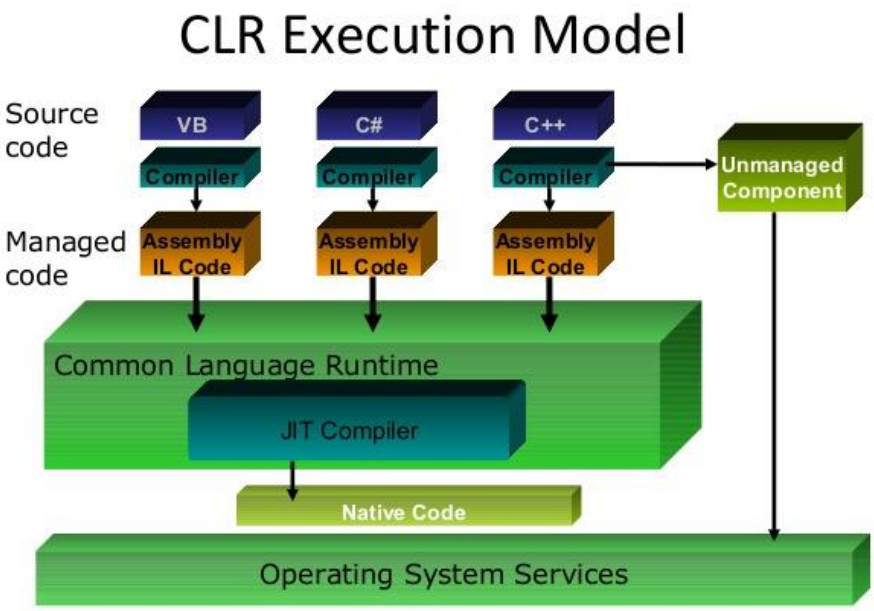


Рисунок 5.1 – Модель виконання коду с CLR

Програми, написані на платформі .NET називають керуємими, тому що CLR контролює всю пам’ять виділену в процесі виконання програми. Керуванням пам’яті займається механізм GC (Garbage Collector).[7]

6.2 Мова програмування C#

В якості мови програмування було обрано C#, тому що мова є потужною, строгою та простою в розумінні. C# представляє парадигму об’єктно-орієнтованого програмування, але також є можливість писати в процедурному та функціональному стилі, а також застосовувати парадигму компонето-орієнтованого програмування.[9]

Код мови C# компілюється в проміжний код, який можна виконувати на будь якому комп’ютері, на якому встановлено .NET Framework необхідної версії. Потужність мови полягає в тому, що є можливість використання таких конструкцій як: лямбда-вирази, методи розширення, LINQ-запити, некеруємий коду, анонімні типи, часткові класи.[9]

6.3 Система управління базами даних MS SQL 2012

Вибір системи управління базами даних обумовлений відмінною інтеграцією з платформою .NET. MS SQL використовує свою реалізацію мови SQL – T-SQL. Мова запитів T-SQL дозволяє створювати процедури та користувацькі функції.[10]

MS SQL підтримує дзеркалювання та кластеризацію баз даних. Також сервер підтримує розподілене управління базами даних та повноцінну модель транзакцій.[10]

6.4 Веб-сервер IIS

Роль веб-сервера – забезпечити хостинг застосунків ASP.Net, WCF, SharePoint. Даний веб-сервер було обрано тому що це єдина реалізація веб-сервера для ASP.NET застосунків в операційній системі Windows.[11]

Веб-сервер розміщує кожен веб-сайт в окремому адресному просторі. Для кожного веб-сайту створюється пул застосунків, що ізолює веб-сайт від інших застосунків, які розміщуються на веб-сервері.[11]

Коли браузер звертається до веб-ресурсу, розміщеному на веб-сервері IIS, запит перехвачує драйвер операційної системи http.sys. Він отримує необхідну конфігурацію від служби WAS. Далі WAS запускає процес W3WP.exe для пулу застосунків. В цьому процесі в окремому домені знаходиться сам веб-сайт, якому передається запит. Запит оброблюється веб-сайтом і відповідь передається назад драйверу http.sys. Драйвер http.sys передає відповідь браузеру. Загальний опис роботи IIS представлено на рисунку 5.2.[11]

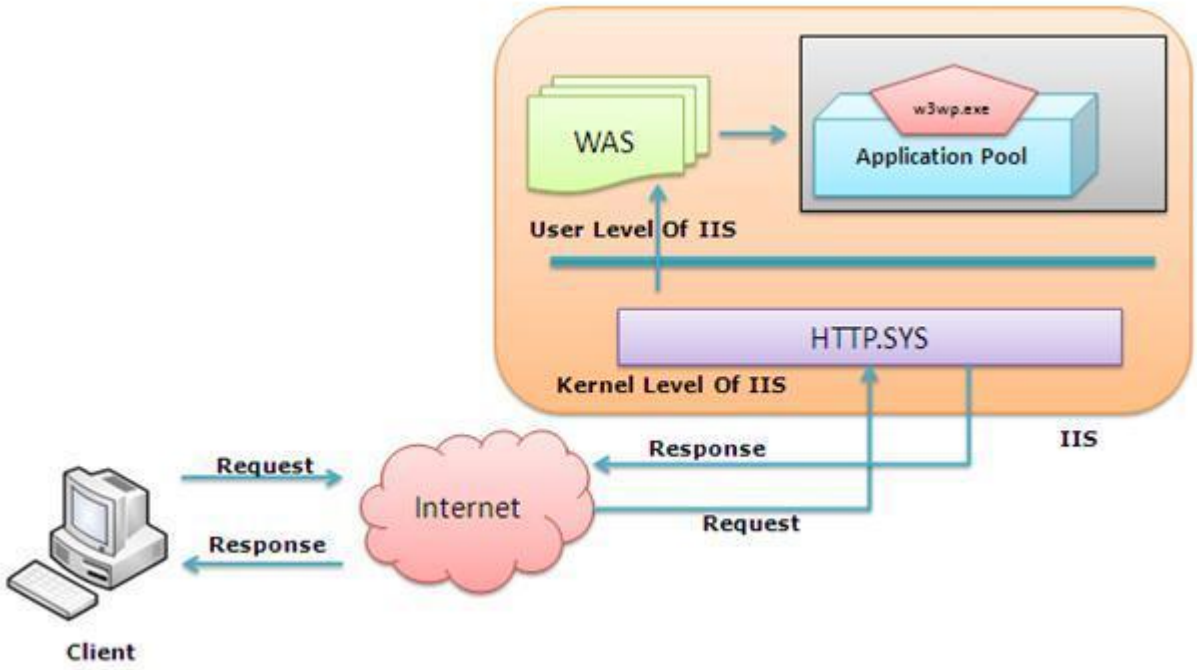


Рисунок 5.2 – Загальний опис роботи IIS

6.5 Набір бібліотек ASP NET MVC

ASP.NET MVC – це платформа для розробки веб-додатків від Microsoft, яка поєднує в собі ефективність та простоту архітектури «модель-представлення-контролер. Вона являє собою повномасштабну альтернативу традиційної технології ASP.NET Web Forms, надаючи суттєві переваги для всіх проектів веб-розробки, такі як розділення логіки відображення від самого відображення, можливість створення restful API. [12]

MVC – архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення. Цей шаблон поділяє систему на три частини. Цими частинами є доменна модель, відокремленні відображення та відокремлений модуль управління (рисунок 5.1). Мета шаблону – гнучкий дизайн програмного забезпечення, який повинен полегшувати подальші зміни чи розширення програм, а також надавати можливість повторного використання окремих компонентів програми.[12]

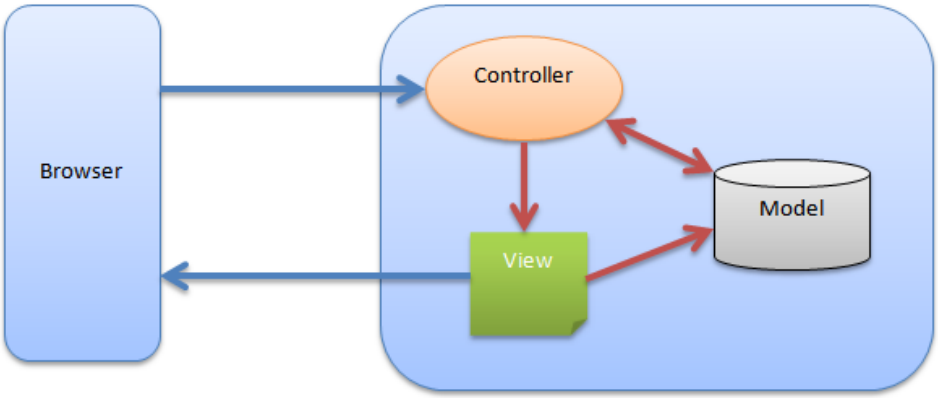


Рисунок 5.1 — Структурні частини архітектурного шаблону MVC

Контролер (controller) представляє клас, що забезпечує зв'язок між користувачем і системою, відображенням і сховищем даних. Представлення (view) – це власне візуальна частина або призначений для користувача інтерфейс програми. Як правило, HTML сторінка, яку користувач бачить, зайшовши на сайт. Модель (model) представляє клас, який описує об’єкти предметної області та всю бізнес-логіку використання даних.[12]

Платформа ASP.NET MVC має механізм візуалізації Razor, який дозволяє прив’язувати модель до представлення з використанням строгої типізації та застосовувати спеціальні методи для генерації сторінок HTML.[12]

6.6 Набір бібліотек LINQ to SQL

LINQ to SQL – це інструмент ORM початкового рівня, що дозволяє виконувати потужні SQL запити. Підтримка LINQ реалізована на рівні мови програмування C#.[13]

Запити LINQ to SQL називаються відкладеними запитами, тому що вони виконуються тільки після запиту на безпосереднє отримання даних від бази даних. Запити зберігаються у вигляді типу ExpressionTree. Кожна нова умова до запиту додається в існуюче дерево виразів.[13]

DataContext – це клас, який встановлює з'єднання з базою даних. Він надає інтерфейс доступу до даних бази даних, вставки, отримання, видалення даних. Також цей клас відслідковує всі зміни в об’єктах отриманих від бази даних.[13]

6.7 Набір бібліотек WCF

Windows Communication Foundation (WCF) — набір клієнтських бібліотек, що дозволяють застосункам на базі відкритої платформи .NET Core взаємодіяти з сервісами WCF, відправляючи повідомлення між сервісами в асинхронному режимі. WCF робить можливою побудову безпечних і надійних систем через спрощену уніфіковану програмну модель взаємодії.[14]

WCF являється реалізацією службово-орієнтованої архітектури (SOA). Суть SOA полягає в розробці програмного забезпечення, при якому система організована у вигляді виділених організованих програмних компонентів із заданою поведінкою, які називаються службами.[14]

Служба – це набір методів із загальними функціональними вимогами, які декларуються у вигляді публічного інтерфейсу та представляють єдиний компонент коду. Служби викликаються іншими модулями, які використовують функціонал служби як «чорний ящик».[14]

Обмін повідомленнями в WCF відбувається за декількома сценаріями:

1) «Запит-відповідь», коли клієнт служби робить запит з деякими параметрами або без них та отримує відповідь.

2) Односторонні повідомлення, коли клієнт викликає методи служби та не чекає від неї відповіді.

3) Дуплексний обмін повідомленнями, коли служба може також викликати методи клієнта, але для цього потрібно створювати додатковий контракт (інтерфейс) взаємодії. Такі способи взаємодії необхідно використовувати, коли потрібно налаштувати взаємодію клієнтів служби між собою.[14]

Для під’єднання до служби WCF клієнти використовують метадані, які служба публікує під час хостингу. Метадані служби публікуються в форматах WSDL, схем XML та WS-Policy. За допомогою цих метаданих клієнти служб маються можливість автоматично налаштовуватись для доступу до служб та виклику їхніх методів.[14]

WCF використовує контракти даних для передачі даних між клієнтом та службою. Дані передаються в серіалізованому вигляді, а контракт даних публікується разом з метаданими служби, що дає можливість клієнтам створювати проксі-класи даних автоматично. Контракти даних описують кожен параметр повідомлення, яке може бути створено чи використано службою. Так як параметри повідомлень представлені в форматі XML, то це дає можливість будувати клієнтів служби на відмінному від .NET технологічному стеку. Єдиною вимогою є можливість клієнта зчитувати XML.[14]

По замовчуванню WCF конвертує всі дані в XML та передає по стандарту SOAP, але є можливість передавати дані у форматах XML, JSON, ATOM по стандарту REST.[14]

Рівень обміну повідомленнями складається з каналів. Канали оперують повідомленнями та їх заголовками. Існують канали транспорту та канали протоколів. Канали транспорту передаються повідомлення по мережі, які представлені у вигляді XML. Прикладами каналів транспорту є HTTP, іменовані канали, TCP, MSMQ. Канали протоколів реалізують протоколи обробки повідомлень за допомогою зчитування додаткових заголовків в повідомленнях. Рівень обміну повідомленнями демонструє можливі формати даних і їх шаблони обміну.[14]

Служби можуть розміщуватися в звичайному виконуваному файлі. Наприклад це може бути консольне застосування чи застосування Windows Form. Також служби можуть розміщуватися в виконуваному файлі, який управляється IIS, WAS або Windows Service.[14]

6.8 Бібліотека логування Log4Net.

Логування є необхідним компонентом інфраструктури будь-якого додатку. Коли наступають критичні моменти в роботі системи, такі як непердбачувані збої, помилки служб, некоректна робота додатку, компонент логування відіграє ключову роль. Від обраного компоненту залежить детальність інформації про помилку і як наслудок швидкість її виправлення. Тому для логування в системі автоматизації пошуку було використано Log4Net. Даний компонент має детальну документацію, зрозумілий інтерфейс та підтримку розробників.

Log4Net дозволяє виводити логи у різні місяця призначення: плоскі файли, база даних, хмара. Бібліотека допомагає розробникам знаходити точне місцезнаходження помилки в коді програми. Log4Net запроектовано таким чином, що він не впливає на продуктивність основного коду програмного продукту. Однією з найбільших перевал логера є поняття ієрархічне логування. Це означає що кожен модуль або підсистема великої системи може мати свій власний логер.[15]

Бібліотека Log4Net має наступні переваги:

1. Можливість виводу логів в різні місця призначення.
2. Ієрархічне логування.
3. Можливість XML конфігурації в конфігураційному файлі.
4. Динамічна конфігурація.
5. Висока продуктивність.
6. Модульна архітектура.

Існує три етапи що, стосуються Log4Net: конфігурація, встановлення, виклик. Конфігурація проводиться в конфігураційному файлі, але створення, налаштування логера проводяться в коді. Після створення логера і реєстрації його в DI контейнері, можна його використовувати для логування дій в системі. [15]

Існує 5 рівнів логування: Fatal, Error, Warn, Info, Debug. Правил до використання цих рівнів немає, тому кожен розробник сам вирішує, які рівні логування до яких дій застосовувати.[15]

Для успішної роботи логера в конфігураційному файлі в тезі «configSections» повинна бути визначена секція для логера, наприклад секція з ім’ям «log4net». Далі в тезі «log4net» потрібно визначити апендери, щоб записувати логи, наприклад у базу даних. Можна визначити декілька апендерів одночасно. Всередині тегу «appender» потрібно визначити тег «layout», щоб задати формат виведення логу.[15]

Для форматування строки виведення логу існують наступні параметри:

- «date» – виведення дати. Можливо використовувати фігурні дужки для форматування дати.
- «utcdate» – виведення дати в універсальному форматі.
- «exception» – повідомлення про помилку (якщо вона є).
- «level» – рівень логу.
- «message» – передане повідомлення для логування.
- «newline» – перехід на нову строку.
- «timestamp» – кількість мілісекунд від старту застосування.
- «thread» – ім’я потоку.
- «identity» – ім’я користувача (виводится значення властивості Principal.Identity.Name).
- «location» – адреса виклику методу логування.
- «line» – номер строки.
- «method» – назва методу.
- «username» – ім’я користувача (властивість WindowsIdentity).

За допомогою тегу «filter» можна визначити які рівні потрібно логувати.

6.9 Засоби контейнеризації Docker

Docker – це набір програмного забезпечення для автоматизації розгортання застосувань в середовищі віртуалізації на рівні операційної системи. Основна ідея полягає в тому, що можна створити образ застосування зі всіма залежностями та розгортати його в операційній системі, де встановлено програмні засоби Docker. Процес розгортання означає створення контейнеру,

який запускається в окремому процесі операційної системи та ізольований від інших процесів.[20]

Висновки до розділу

В даному розділі було розглянуто та обрано технологічний стек розробки системи. Було обрано стек технологій .Net, СУБД MSSQL. Стек технологій .Net є сучасним, потужним рішенням для розробки застосувань. Для розробки веб-частини системи було обрано Asp.Net MVC. Для розробки служб було обрано програмні засоби WCF. Хостинг застосувань буде відбуватися за допомогою серверу IIS, Windows служб та засобів Docker.

7 АРХІТЕКТУРА ТА ПРОЕКТУВАННЯ СИСТЕМИ

7.1 Загальна архітектура системи

Система автоматизованого пошуку складається з багатьох компонентів. Структурну схему системи відображено у додатку А. Але умовно система поділяється на 2 чатини: клієнт та сервер.

Клієнтська частина складається з наступних модулів:

1) Веб-додаток.
2) Компонент виконання запланованих задач.
3) Приймач результатів моніторингу.

Серверна частина складається з наступних модулів:

1) Веб-додаток бібліотеки.
2) Компонент безпосерднього пошуку.
3) Компонент виконання скриптів аналізу даних.
4) Приймач результатів пошуку.
5) Компонент одержання завантажених та проаналізованих публікацій.
6) Компонент клієнтської підписки на видання бібліотеки.
7) Служба фонового завантаження та індексації видань бібліотеки.

7.2 Архітектура клієнтської частини

7.2.1 Трирівнева архітектура веб-додатку клієнтської частини

Для побудови невисоконавантажених веб-додатків тритівнева архітектура є свого роду стандартом і передбачає поділ додатку на 3 зони відповідальності: доменна логіка (моделі даних, сервіси тощо), представлення, база даних (або інше сховище даних). Абстракцією трирівневої архітектури є шаблон проектування MVC. Реалізацією є проект на основі Asp.Net MVC фреймворку. Веб-додаток клієнтської частини побудовано на основі шаблону MVC.

Рівень роботи з доменною локігою розподіляється на наступні типи даних: моделі даних предметної області, інтерфейси сервісів, реалізації сервісів, класи роботи з базою даних (репозиторії), файли ресурсів, хелпери.

На цьому рівні знаходиться вся бізнес-логіка, яка стосується створення роботи з виданнями, пакетами, план-графіками, запуску автоматичного моніторингу лікарських засобів. Репозиторії необхідні для отримання даних з БД та для проведення транзакційних операцій. Сервіси виступають у якості медіаторів, вони реалізовують логіку отримання даних з репозиторіїв, запуску бізнес-логіки в доменних моделях та видачі оброблених даних на рівень контроллерів.

Рівень представлення включає в себе файли відображення, моделі представлення, файли скриптів, в яких зосереджена логіка відобрадення даних.

Рівень сховища даних складається з розгорнутої БД, яка складається з таблиць, відображень, процедур, необхідних для виконання операцій, пов’язаних з бізнес-логікою додатку.

Архітектура веб-додатку була побудована з огляду на використання шаблонів проектування таких як: сінгтон, фасад, стратегія, одиниця роботи, репозиторій. Також автивно застосовувалися шаблони інверсії залежностей та ін’єкції залежностей. При проектуванні використовувалися правила SOLID.

Приймач результатів моніторингу спроектовано таким чином, що він представляє собою звичайний контроллер, який на вхіж приймає дані, на основі, яких формується команда. Команда викливає необхідні класи, щоб зберегти результати у базі даних та оновити дані в системі.

7.2.2 Архітектура служби планування

Архітектурно служба планування позиціонується як сервіс, який повинен працювати безперервно та без збоїв. У випадку непередбачуваних помилок, він повинен мати можливість автоматичного перезапуску. Тому для розробки використовувалася технологія WCF та хостинг як Windows-служба.

Служба планування виступає помередником між клієнтською частиною та службою безпосереднього пошуку. Тому саме вона після виконання пошуку віддає результати на сторону клієнта. Передача даних відбувається за допомогою спеціалізованих класів, які виконують роль передавачів даних на контретний екземпляр клієнту. В рамках служби планування ці класи навиваються адаптерами та конфігуруються вони у конфігураціфному файлі. В конфігураційному файлі задаються секції (по одній на кожного клієнта). В основному для одного клієнта необхідно одна служба планування, але іноді бувають випадки використання однієї служби планування декількома клієнтами одночасно. Тому дана конфігурація є достатньо гнучкою і дозволяє без проблем підключати нових клієнтів до служби планування.

Ядро служби планування побудоване у вигляді збірки ExSearch.Service. Дана збірка побудована з використанням принципів розробки SOLID, а саме: класи виконують тільки одну задачу, в проекті застосовується інверсія залежностей (IoC) та ін’єкція залежностей (DI).[15] Збірка містить в собі папку з абстракціям, де присутній інтерфейс логування дій в службі планування, а також інтерфейс репозиторію для абстрагування роботи з базою даних. У внутрішній базі даних зберігаються необхідні дані про пошукові запити, правила пошуку та системні логи.

Структура проекту ExSearch.Service в середовищі розробки Visual Studio зображена на рисунку 6.1.

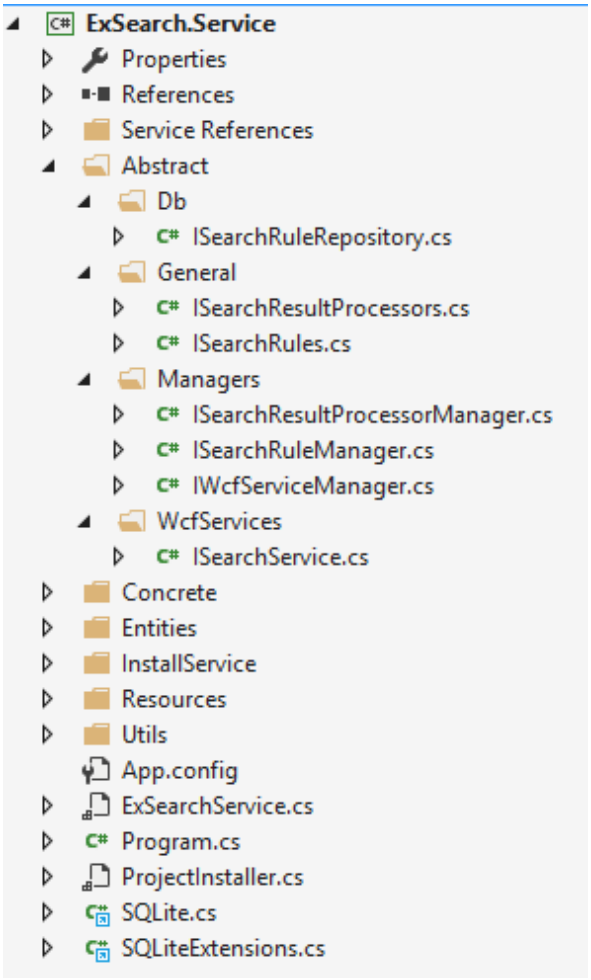


Рисунок 6.1 – Структура проекту ExSearch.Service

Проект поділено на папки для зручної навігації. Проект має наступну структуру папок:

- папка для абстрактних сутностей;
- папка для реалізацій абстракцій;
- папка з сутностями доменної моделі;
- папка з файлами розгортання служби;
- папка з ресурсами;
- папка з допоміжними класами та хелперами.

Папки з реалізаціями та з сутностями предметної області розбиті на підпапки згідно із зонами відповідальності класів, які в них знаходяться. Діаграма класів служби відображена у додатку Г.

Служба розроблялася за принципом розділення обов’язків між класами, а також з використанням інферсії залежностей. В архітектурі уе відображалося тим, що є доменні сутності та доменні сервіси, які представляють собою основну

бізнес-логіку служби, а навколо них побудована інфраструктура служби. Для досягнення високого рівня розділення обов'язків між класами, застосовувалися шаблони проектування, такі як фабричний метод, адаптер, шаблонний метод, проксі, абстрактна фабрика та декоратор.

Використання шабонів проектування значно полегшує розробку системи, розширення її функціоналу у відповідності до принчипу відкритості/закритості, а також проведення модульного тестування, коли можна зробити заглушки для тих компонентів системи, які є сторонніми в даному контексті, а тестувати лише цільовий код. Наприклад виклористання шаблону фабрики абстрагує створення об'єктів в системі, що значно зменшує прямі зв'язки між класами. Шаблон ін'єкції залежностей є основним при проектування, тому що якраз він дозволяє послабити зв'язки між типами системи, а обов'язки отримання залежностей покласти на контейнер інверсії залежностей. Реалізації інтерфейсів будуть постачатися контейнером при надходженні запитів на виконання команд служби в залежності від законфігурованого типу часу життя конкретної залежності. Шаблон проксі застосовується у якості створення клієнта для виклику команд сторонніх служб (наприклад служби пошуку).

Папка General складається з двох програмних інтерфейсів. Інтерфейс ISearchResultProcessors декларує один метод для отримання хендлерів результатів пошуку. На вхід йому приходить ім'я хендлеру. Інтерфейс ISearchRules декларує сигнатури методів управління планувальниками пошуку, такі як: активація, деактивація, CRUD операції над планувальниками. В сумі інтерфейс містить 6 методів.

Інтерфейс ISearchRuleRepository призначено для отримання правил пошуку з бази даних. Він містить методи отримання даних, створення правила пошуку, видалення та оновлення.Також в інтерфейсі присутня властивість, за допомогою якої можна отримати всі правила пошуку окремого планувальника.

Для роботи з базою даних було створено інтерфейс репозиторію ISearchRuleRepository, який призначено для виконання операцій з правилами пошуку такими як: створення, оновлення, видалення. Інтерфейс містить

властивість для отримання правил пошуку в БД, а також шість методів для виконання операцій CRUD стосовно правил пошуку.

Папка Managers містить в собі три інтерфейси: ISearchResultProcessorManager, ISearchRuleManager та IWcfServiceManager. Інтерфейс ISearchResultProcessorManager застосовується для запуску та для зупинки хендлерів результатів пошуку. Інтерфейс ISearchRuleManager призначений для управління запуском та зупинкою планувальників пошуку. Інтерфейс IWcfServiceManager призначений для управління життєвим циклом служб WCF, які визначені в даному файлі збірки.

Інтерфейс ISearchService, який знаходиться у папці WcfServices, є основним інтерфейсом взаємодії в цій збірці. Він представляє собою публічний інтерфейс веб-служби та викликається для здійснення моментального та автоматичнго пошуку. Інтерфейс складається з десяти методів. Методи викликаються зовнішнім клієнтами служби для створення інформації про правила пошуку, їх оновлення та видаляння. Також в інтерфейсі присутні методи для активізації та деактивізації правил пошуку з переданим конкретним ідентифікатором правила пошуку. За допомогою цього інтерфейсу також можна запросити необхідні дані правила пошуку передавши його ідентифікатор. Присутня можливість отримати перелік правил пошуку передавши ім’ям обробника пошуку. Основним є метод виконання миттєвого пошуку.

Збірка ExSearch.Core містить в собі базові інтерфейси для ведення логування, реєстрації типів об’єктів в контейнері Unity та програмні інтерфейси для роботи адаптерів (передавачів результатів пошуку). Для запуску та роботи адаптерів представлено два інтерфейси: IDispatcher, ISearchResultProcessor та три абстрактних класи Dispatcher, SearchResultProcessor, SearchResultProcessorFactory. Інтерфейс IDispatcher та його абстрактна реалізація використовуються для отримання екземплярів обробників результатів пошуку (адаптерів) та для реєстрації типів у контейнері Unity. Інтерфейс ISearchResultProcessor та його абстрактна реалізація описують роботу обробника результатів пошуку і містить собі три методи, які призначені для виконання певних дій після оперативного пошуку, після планового автоматичного пошуку,

після отримання помилки під час пошуку. Абстрактний клас SearchResultProcessorFactory представляє собою фабрику для створення обробників результатів пошуку. Створення обробників результатів пошуку відбувається із застосуванням конфігураційних секцій в конфігураційному файлі.

В якості адаптеру виступає збірка ExSearch.Adapter.DSBase. Збірка містить в собі один інтерфейс IDSBaseSearchResultClient, який описує роботу зі службою завантаження результатів пошуку на сторону клієнта моніторингу. Інтерфейс містить методи ініціалізації настройок служби завантаження результатів пошуку, метод з'єднання, від'єднання, перевірки досяжності, відправки результатів автоматичного пошуку та результатів миттєвого пошуку. Для налаштування адаптеру існує клас DSBaseConfigurationSection, який зчитує з локального конфігураційного файлу інформацію та налаштовує адаптер на роботу.

7.3 Архітектура серверної частини

Архітектура веб-модулю серверної частини повністю ідентична веб-модулю клієнтської частини. Служби прийому результатів пошуку, підписки на видання та завантаження публікацій теж архітектурно співпадає зі службою прийому результатів моніторингу на клієнтській частині. Єдиними відмінностями є сутності, які використовуються у веб-модулі серверної частини системи.

7.3.1 Архітектура служби пошуку

Служба пошуку представлена у вигляді набору збірок ExSearch.Engine.Service та ExSearch.Engine.Core. З точки зору структури проектів служба пошуку схожа на службу планування, при чому реалізація базової збірки служби планування майже повторює реалізацію базової збірки служби пошуку. Для передачі результатів пошуку на сторону електронної

бібліотеки використовуються обробники результатів пошуку. Для безпосереднього пошуку використовуються провайдери (обробники) пошуку. Провайдери пошуку можуть використовувати можливості служби браузера для рендерингу результату пошуку.

Основними вимогами до служби є стійкість до високого навантаження, контроль виконання пошуку, гарантоване отримання результату пошуку. Служба розміщується в процесі операційної системи Windows в якості Windows service. Для реалізації клієнт-серверної технології застосовується WCF.

Збірка має інтерфейс для логування дій в модулі пошуку та інтерфейс репозиторію для збереження необхідних даних в базу даних. В базі даних зберігаються системні логи.

В папці General зосереджено два інтерфейси: ISearchProcessors та ISearchResultLoaders. Інтерфейс ISearchProcessors призначений для отримання інтерфейсу обробника результатів пошуку і містить один метод, який повинен повернути інтерфейс обробника результатів пошуку за його ім’ям. Інтерфейс ISearchResultLoaders призначений для отримання інтерфейсу екземпляру завантажника за його ім’ям та для отримання всіх завантажників і містить в собі два методи.

В папці Managers зосереджено три інтерфейси, що стосуються служби пошуку: ISearchProcessorManager, ISearchResultLoaderManager та IWcfServiceManager. Інтерфейс ISearchProcessorManager призначений для запуску та зупинки обробників пошуку. Інтерфейс ISearchResultLoaderManager призначений для запуску та зупинки завантажників пошуку. Інтерфейс IWcfServiceManager призначений для запуску та зупинки всіх служб WCF, які визначені в даній збірці.

Інтерфейс ISearchCoreService, що визначений в папці WcfServices є основним в цій збірці. Цей інтерфейс представляє публічний інтерфейс веб-служби і дозволяє здійснювати пошук. Інтерфейс містить в собі один метод. Метод дозволяє зовнішнім клієнтам служби робити пошук передавши об’єкт пошукового запиту та назву обробника пошуку.

Збірка ExSearch.Engine.Core містить базові інтерфейси для логування, реєстрації екземплярів об’єктів в контейнері Unity та інтерфейси для роботи обробників та провайдерів пошуку. Для роботи обробників представлено два інтерфейси: IDispatcher, ISearchResultLoader та три абстрактних класи Dispatcher, SearchResultLoader, SearchResultLoaderFactory. Інтерфейс IDispatcher та його абстрактна реалізація призначені для отримання екземплярів інтерфейсів обробки результатів пошуку обробника та для реєстрації додаткових класів в контейнері Unity. Інтерфейс ISearchResultLoader та його абстрактна реалізація описують роботу обробника результатів пошуку і містить собі 2 методи, які призначені для завантаження результатів в електронну бібліотеку після оперативного пошуку та після планового автоматичного пошуку. Абстрактний клас SearchResultLoaderFactory представляє собою фабрику для створення завантажників результатів пошуку. Створення завантажників результатів пошуку відбувається із застосуванням конфігураційних секцій в конфігураційному файлі.

В якості завантажника виступає збірка ExSearch.Engine.Handler.DSBase, яка призначена для відправки результатів пошуку на сторону електронної бібліотеки. Збірка побудована на основі збірки ExSearch.Adapter.DSBase і має ідентичний інтерфейс та способи налаштування.

В якості провайдера пошуку виступає збірка ExSearch.Engine.Provider.Google, яка виконує функції безпосереднього пошуку за допомогою пошукової системи Google. Збірка містить в собі два однакових за структурою інтерфейси: ICseGoogleClient та IWebGoogleClient. Інтерфейси мають метод ініціалізації настройок та метод безпосереднього пошуку. Інтерфейс ICseGoogleClient призначений для пошуку з використанням Google custom search API, а інтерфейс IWebGoogleClient для пошуку з використанням служби браузерів та стандартного веб-пошуку Google. Настоювання провайдера пошуку відбувається за схожим сценарієм з настройкою завантажників, тобто за допомогою класів-конфігураторів та конфігураційних секцій локального конфігураційного файлу.

7.3.2 Архітектура служби браузера

Служба браузера представлена у вигляді збірок ExSearch.Engine.Browser, ExSearch.Engine.Service та ExSearch.Engine.Core. У збірці ExSearch.Engine визначено інтерфейс IBrowserManager, який призначено для запуска та зупинки всіх браузерів.

Інтерфейсом служби браузерів є IBrowserService, який призначений для загрузки контенту з мережі Інтернет. Він містить в собі 2 методи, які призначені для завантаження часткового контенту, тобто тільки сторінка HTML або будь-який інший один об’єкт та метод повної загрузки сторінки зі всіма залежними сутностями.

Головною для роботи браузера є збірка ExSearch.Engine.Browser. Структура проекту представлена на рисунку 6.2.

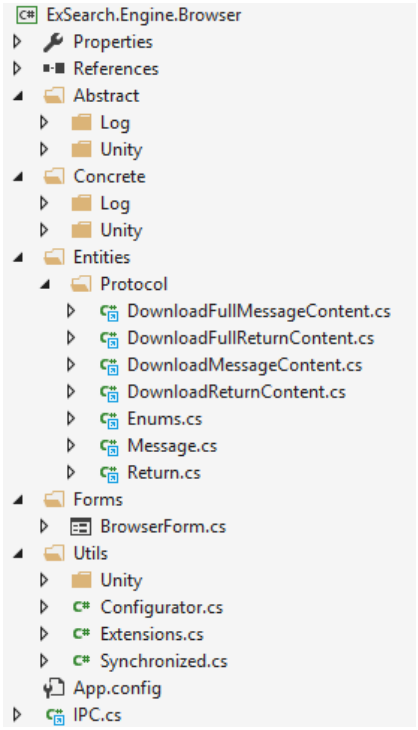


Рисунок 6.2 – Структура проекту ExSearch.Engine.Browser

Для взаємодії браузерів та служби браузерів використовується протокол взаємодії IPC.

7.3.3 Архітектура служби відпрацювання скриптів Python

Служба відпрацювання скриптів Python представлена у вигляді двох збірок: ExSearch.Python.Processor та ExSearch.Python.Service. У збірці ExSearch.Python.Processor представлено інтерфейс для запуску та зупинки сервера відпрацювання скриптів, інтерфейс для логування, інтерфейс для реєстрації об’єктів в DI-контейнері Unity. Також представлено інтерфейс IPythonAPI, який призначено для виклику методів збірки pythonXX.dll, де XX означає версію збірки. Інтерфейс має 2 властивості: отримання версії Python та отримання шляху до збірки Python.

Для взаємодії служби відпрацювання скриптів Python та самими обробниками скриптів використовується протокол IPC та іменовані канали взаємодії. Принцип роботи схожий на роботу служби браузерів.

Збірка ExSearch.Python.Service містить дванадцять інтерфейсів. В папці Db зосереджено інтерфейс IScriptContainerRepository для роботи з базою даних. В ньому присутні методи додавання, оновлення, отримання та видалення скриптів по заданим ідентифікаторам.

В папці General зосереджено чотири інтерфейси: IFileSystem, IPythonProcessor, IPythonScriptSet, IPythonSessions. Інтерфейс IFileSystem призначено кешування наборів сктриптів у файловій системі. Інтерфейс має метод для отримання списку повних шляхів до файлів за маскою регулярного виразу, метод для отримання файлу за його повним шляхом, метод збереження файлу на диск, метод видалення файлу з диску, методи створення, видалення та очищення директорії. Інтерфейс IPythonProcessor призначено для виконання скриптів і має один метод, який запускає на виконання скрипти Python з передачею необхідних параметрів. Інтерфейс IPythonScriptSets призначений для управління наборами скриптів. Інтерфейс має 4 методи, які створюють, оновлюють, видаляють та запускають набори скриптів на виконання. Інтерфейс IPythonSessions призначений для управління сесіями виконання скриптів і містить в собі метод запуску сесії з передачею вхідних даних та метод завершення сесії.

Папка Log містить в собі інтерфейси для логування дій в модулі. Папка Managers містить менеджери управління життєвим циклом процесів відпрацювання скриптів Python. Папка Unity містить інтерфейс для реєстрації об’єктів в DI-контейнері Unity.

Інтерфейс IPythonService є інтерфейсом служби відпрацювання скриптів Python і містить в собі п’ять методів, а саме метод додавання набору скриптів з отриманням ідентифікатору набору, метод отримання набору скриптів за ідентифікатором, метод отримання всіх наборів скриптів, метод видалення скриптів та метод запуску скриптів на виконання.

7.3.4 Архітектура служби фонового завантаження та індексації видань бібліотеки

Служба у вигляді зовнішнього прикладного інтерфейсу WebAPI для виклику зі сторони сторонніх служб та застосувань та внутрішніх компонентів, таких як: планувальник задач, індексатор, загрузчик контенту, та база даних бібліотеки.

Для планування задач частково використовуються наробки служби планування пошукових запитів.

Для загрузчика контенту використовується служба браузерів, які у фоновому режимі будуть завантажувати контент.

Компонент індексації відповідає за пошук відповідних ключових слів в контенті та збереження цієї інформації в таблицях БД.

7.4 Проектування бази даних

Стандартним підходом до створення структури бази даних є створення на кожну сутність бізнес логіки окремої таблиці та приведення структури бази даних в третю нормальну форму. Але при збільшенні кількості сутностей в системі, буде рости кількість таблиць в базі даних, що в майбутньому може призвести до складності підтримки структури бази даних. Тому було розглянуто

альтернативний підхід до створення структури бази даних. За даним підходом вся структура бази даних описується вигляді записів в деяких базових таблицях. В цих таблицях зберігаються метадані сутностей системи та їх значення. Основними таблицями є: типи даних, об’єкти предметної області (таблиці), записи об’єктів, атрибути об’єктів, значення атрибутів об’єктів. Недоліками такої структури є те, що такі структури не завжди можна привести до третьої нормальної форми. Також запити для отримання значень об’єктів будуються з використанням великої кількості з’єднань, що відображається на швидкості виконання.

Проте перевагами є те, що час на підтримку таких структур скорочується так як для додавання нової сутності в систему достатньо додати запис в таблицю сутностей, а також додати записи атрибутів сутності в таблицю атрибутів.

Для реалізації веб-додатку клієнтської та серверної сторони системи застосовується три реляційні БД. Перша база даних призначена для збереження всіх даних бізнес-логіки системи, а саме пити об’єктів, їх атрибути та значення, довідникова інформація. Друга база даних потрібна для роботи самого веб-додатку і містить інформацію про користувачів системи, їх ролей та дозволів на відображення інформації. Ця база даних є доробкою стандартної бази даних, яка створюється засобами ASP.NET. Третя база даних призначена для зберігання файлів в системі.

При проектуванні першої бази даних використовується підхід у проектуванні баз даних, описаний на початку розділу. Перелік основних сутностей бази даних зображено у таблиці 7.1.

Таблиця 7.1 — Перелік сутностей та їх опис

Назва сутності	Опис
Dossier	Представляє типи об’єктів системи
Dossier_Attribute	Представляє атрибути об’єктів
Dossier_List	Представляє інформацію про запис об’єкту та його дочірніх об’єктів
Dossier_Value	Представляє значення об’єктів системи
Dossier_History	Представляє всі версії значень об’єкту, якщо він змінювався
Dossier_Link	Дозволяє створювати зв’язки n-n між об’єктами
Dictionary	Містить інформацію про словники та їх структуру

Назва сутності	Опис
Dossier	Представляє типи об’єктів системи
Dossier_Attribute	Представляє атрибути об’єктів
Dossier_List	Представляє інформацію про запис об’єкту та його дочірніх об’єктів
Dossier_Value	Представляє значення об’єктів системи
Dossier_History	Представляє всі версії значень об’єкту, якщо він змінювався
Dossier_Link	Дозволяє створювати зв’язки n-n між об’єктами
Dictionary	Містить інформацію про словники та їх структуру
Dictionary_Value	Містить значення словників
Grid_Value	Містить значення масивів даних, вкладених у об’єкт
Grid_History	Містить всі версії значень масивів даних, вкладених у об’єкт, якщо об’єкт змінювався в процесі роботи системи.
Client	Представляє клієнтів системи
Drug_registration	Представляє лікарські засоби
Active_substances	Представляє інформацію про діючі речовини
Reference_drug	Представляє референтні лікарські засоби
Session_request	Представляє запити на пошук інформації
Sysdict	Представляє системні словники та їх значення

Після проектування концептуальної моделі БД відбувається перехід до даталогічної моделі, де кожна сутність перетворюється в таблицю з необхідними атрибутами. Даталогічну модель бази даних відображено у додатку В.

Перших одинадцять сутностей в таблиці 7.1 формують основу бази даних на основі метаданих.

Таблиця Dossier призначена для зберігання всіх типів об’єктів в системі та їх ієрархій. По своїй суті об’єкт для веб-модулю клієнтської та серверної частини

– це реєстраційна картка, яка містить безліч вкладок, але не менше однієї (головна вкладка).

Таблиця Dossier містить сім полів. Ці поля описані в таблиці 7.2. Поле DOS_IDP – це порядковий номер типу об’єкта. За допомогою поля DOS_LID реалізується ієрархічний зв’язок, в полі міститься NULL, якщо це головний запис, в іншому випадку там зберігається ідентифікатор батьківського запису. Поле LLS_IDP зберігає мову типу об’єкта. Поле DOS_UID є глобальним ідентифікатором без врахування мови. Поле CLI_IDP містить ідентифікатор клієнта, який є власником типу об’єкта.

Таблиця 7.2 – Опис таблиці Dossier

Назва поля	Тип поля	Опис
DOS_IDP (PK)	NUMERIC	Ідентифікатор типу об’єкта
DOS_LID (FK)	NUMERIC	Ідентифікатор батьківського типу об’єкта
CLI_IDP (FK)	NUMERIC	Ідентифікатор клієнта
LLS_IDP (FK)	NUMERIC	Ідентифікатор мови
DOS_LNAM	NVARCHAR	Повна назва
DOS_SNAM	NVARCHAR	Скорочена назва
DOS_UID	NUMERIC	Глобальний ідентифікатор

Таблиця Dossier_Attribute призначена для збереження атрибутів об’єктів предметної області. Таблиця має 9 полів, які описані в таблиці 7.3.

Таблиця 7.3 – Опис таблиці Dossier_Attribute

Назва поля	Тип поля	Опис
DAT_IDP (PK)	NUMERIC	Ідентифікатор атрибуту
DAT_LID (FK)	NUMERIC	Ідентифікатор батьківського атрибуту
DOS_IDP (FK)	NUMERIC	Ідентифікатор типу об’єкта
DIC_IDP (FK)	NUMERIC	Ідентифікатор словника
LLS_IDP (FK)	NUMERIC	Ідентифікатор мови

DAT_LNAM	NVARCHAR	Повна назва атрибуту
DAT_SNAM	NVARCHAR	Скорочена назва атрибуту
DAT_FNAM	NVARCHAR	Назва поля в таблиці Dossier_Value
DAT_UID	NUMERIC	Глобальний ідентифікатор

Поле DIC_IDP містить ідентифікатор словника, якщо поле є словниковим. Поле DAT_FNAM містить назву поля в таблиці Dossier_Value, в якому зберігається значення для даного атрибута.

Таблиця Dossier_List призначена для зберігання відомостей про записи об’єктів. При створені нової картки спочатку створюється запис в цій таблиці, а вже потім в таблицях Dossier_Value та Grid_Value. Для кожного об’єкта в таблиці міститься як мінімум 2 записи: головний запис та запис однієї вкладки. Таблиця має одинадцять полів, що описуються в таблиці 7.4.

Таблиця 7.4 – Опис таблиці Dossier_List

Назва поля	Тип поля	Опис
DLS_IDP (PK)	NUMERIC	Ідентифікатор запису
DLS_DID (FK)	NUMERIC	Ідентифікатор типу об’єкту головного запису

Продовження таблиці 7.4 – Опис таблиці Dossier_List

Назва поля	Назва поля	Назва поля
DLS_PID (FK)	NUMERIC	Ідентифікатор типу об’єкт вкладки
DLS_LID (FK)	NUMERIC	Ідентифікатор батьківського запису
DLS_CDAT	DATETIME	Дата створення запису
DLS_DAUT	NVARCHAR	Ім’я користувача, який створив запис
DLS_EDIT	INT	Признак реагування запису в поточний момент часу
DLS_EDAT	DATETIME	Дата останнього редагування запису, якщо редагування відбувається в поточний момент часу

DLS_NEDI	NVARCHAR	Ім’я користувача, який останній редагував запис, якщо редагування відбувається в поточний момент часу
DLS_DDAT	DATETIME	Дата видалення запису
CLI_IDP (FK)	NUMERIC	Ідентифікатор клієнта, користувач якого створив запис

Поля DLS_EDIT, DLS_EDAT та DLS_NEDI призначені для блокування запису під час його редагування. Це означає, що якщо один користувач редагує запис, інший може відкрити його тільки на перегляд. Поле CLI_IDP містить в собі ідентифікатор клієнта та призначене для відокремлення записів по ідентифікатору клієнта. Система спроектована таким чином, що в тій частині бази даних, яка основана на метаданих дані з таблиці Dossier_List та Dossier_Value не видаляються. Для того, щоб не показувати користувачу видалені дані в таблиці є поле DLS_DDAT, яке показує одночасно дату видалення запису і є ознакою видалення запису.

Таблиця Dossier_Value призначена для збереження даних об’єктів системи. Таблиця містить поля всіх типів даних, щоб можна було зберегти будь-яку інформацію про об’єкт. Таблиця має 104 поля. При неповний опис полів описаний в таблиці 7.5.

Таблиця 7.5 – Опис таблиці Dossier_Value

Назва поля	Тип поля	Опис
DVL_IDP (PK)	NUMERIC	Ідентифікатор запису
DVL_USR	NVARCHAR	Ім’я користувача, який створив запис
DVL_DAT	DATETIME	Дата створення запису
DLS_IDP (FK)	NUMERIC	Ідентифікатор запису в таблиці Dossier_List
DVL_TEXT1	NVARCHAR	Текстове поле
…	…	…
DVL_TEXT50	NVARCHAR	Текстове поле

DVL_DATE1	DATETIME	Поле збереження дати
…	…	…
DVL_DATE15	DATETIME	Поле збереження дати
DVL_NUM1	NUMERIC	Числове поле
…	…	…
DVL_NUM25	NUMERIC	Числове поле
DVL_FILE1	VARBINARY	Поле для збереження файлів
…	…	…
DVL_FILE10	VARBINARY	Поле для збереження файлів

В основному значення в таблиці Dossier_Value записуються тільки для вкладок.

Таблиця Dossier_History призначена для ведення історичності змінення об'єктів системи. Кожна нова версія об’єкту зберігається в таблиці Dossier_Value, а попередня версія зберігається в таблиці Dossier_History. Таблиця має 105 полів. ЇЇ опис представлено в таблиці 7.6.

Таблиця 7.6 – Опис таблиці Dossier_History

Назва поля	Тип поля	Опис
DVH_IDP (PK)	NUMERIC	Ідентифікатор запису
DVL_IDP (FK)	NUMERIC	Ідентифікатор запису в таблиці Dossier_Value, який містить поточну версію значень об’єкта
DVH_USR	NVARCHAR	Ім’я користувача, який створив запис
DVH_VER	INT	Номер версії запису
DVH_DAT	DATETIME	Дата створення запису
DVH_TEXT1	NVARCHAR	Текстове поле
…	…	…
DVH_TEXT50	NVARCHAR	Текстове поле
DVH_DATE1	DATETIME	Поле збереження дати
…	…	…
DVH_DATE15	DATETIME	Поле збереження дати
DVH_NUM1	NUMERIC	Числове поле
…	…	…
DVH_NUM25	NUMERIC	Числове поле
DVH_FILE1	VARBINARY	Поле для збереження файлів
…	…	…
DVH_FILE10	VARBINARY	Поле для збереження файлів

Таблиця Dossier_Link призначена для зв’язку об’єктів системи між собою. Дана таблиця використовується, коли потрібно отримати зв’язок «багато до багатьох». Таблиця має 4 поля, опис яких представлено в таблиці 7.7.

Таблиця 7.7 – Опис таблиці Dossier_Link

Назва поля	Тип поля	Опис
DLI_IDP (PK)	NUMERIC	Ідентифікатор зв’язку
DLI_LI1 (FK)	NUMERIC	Ідентифікатор запису в таблиці Dossier_List
DLI_LI2 (FK)	NUMERIC	Ідентифікатор запису в таблиці Dossier_List
DLI_DAT	DATETIME	Дата створення зв’язку

Таблиця Dicionary містить в собі інформацію про словники, які використовуються в системі. Таблиця має 5 полів, опис яких представлено в таблиці 7.8.

Таблиця 7.8 – Опис таблиці Dictionary

Назва поля	Тип поля	Опис
DIC_IDP (PK)	NUMERIC	Ідентифікатор словника
DIC_LID (FK)	NUMERIC	Ідентифікатор батьківського довідника
LLS_IDP (FK)	NUMERIC	Ідентифікатор мови словника
DIC_NAME	NVARCHAR	Назва словника
DIC_UID	NUMERIC	Глобальний ідентифікатор словника

Словники можуть бути ієрархічними з безліччю рівнів вкладеності, про що свідчить поле DIC_LID, яке містить ідентифікатор батьківського довідника. Поле DIC_UID містить ідентифікатор словника без прив’язки до мови. Це значить, що у одного словника на різних мовах поля DIC_IDP містять різні значення, а поля DIC_UID – однакові.

Таблиця Dictionary_Value призначена для збереження значень довідників з таблиці Dictionary. Таблиця є ієрархічною і містить 10 полів, опис яких представлено в таблиці 7.9.

Таблиця 7.9 – Опис таблиці Dictionary_Value

Назва поля	Тип поля	Опис
DICV_IDP (PK)	NUMERIC	Ідентифікатор значення

DIC_IDP (FK)	NUMERIC	Ідентифікатор словника
DICV_LID (FK)	NUMERIC	Ідентифікатор батьківського значення
LLS_IDP (FK)	NUMERIC	Ідентифікатор мови
DICV_LNAM	NVARCHAR	Повна назва значення словника
DICV_SNAM	NVARCHAR	Скорочена назва значення словника
DICV_SDAT	DATETIME	Початкова дата дії значення словника
DICV_EDAT	DATETIME	Кінцева дата дії значення словника
DICV_UID	NUMERIC	Глобальний ідентифікатор значення словника
DICV_SORT	NUMERIC	Порядок значень словника

Значення словників, як і самі словники, є ієрархічними в залежності від ієрархії словників. Поля DICV_SDAT та DICV_EDAT призначені для контролю актуальності значення словника. При зміненні значення словника чи видаленні значення словника, змінюється поле DICV_EDAT таким чином, щоб користувачу більше не показувалося дане поле, або показувалося з ознакою неактуальності. Поле DICV_SORT призначене для контролю порядку значень в словнику. Порядок значень встановлюється один раз при створенні словника і більше не змінюється. Дане поле більше призначене для розробників системи, щоб не зав’язуватися на конкретні id значень словників, а використовувати їх відносний порядок.

Таблиця Grid_Value призначена для збереження спискових значень об’єктів системи (вкладок). Таблиця має 74 поля, опис яких представлено в таблиці 7.10.

Таблиця 7.10 – Опис таблиці Grid_Value

Назва поля	Тип поля	Опис
GRD_IDP (PK)	NUMERIC	Ідентифікатор значення
DVL_IDP (FK)	NUMERIC	Ідентифікатор запису значень об’єкту системи в таблиці Dossier_Value
GRD_TEXT1	NVARCHAR	Текстове поле
…	…	…
GRD_TEXT30	NVARCHAR	Текстове поле
GRD_DATE1	DATETIME	Поле для збереження дати
…	…	…
GRD_DATE10	DATETIME	Поле для збереження дати
GRD_NUM1	NUMERIC	Числове поле
…	…	…
GRD_NUM15	NUMERIC	Числове поле
GRD_FILE1	VARBINARY	Поле для збереження файлів
…	…	…
GRD_FILE10	VARBINARY	Поле для збереження файлів
GRD_LID (FK)	NUMERIC	Ідентифікатор типу спискових значень

Поле GRD_LID вказує на ідентифікатор типу спискових значень, що зберігається в таблиці Dossier_Attribute в полі DAT_IDP, при цьому значення поля DAT_FNAM в цій таблиці має бути пустим.

Таблиця Grid_History призначена для ведення історичності зміни записів в таблиці Grid_Value. Таблиця має 74, опис яких описано в таблиці 7.11.

Таблиця 7.11 – Опис таблиці Grid_History

Назва поля	Тип поля	Опис
GRH_IDP (PK)	NUMERIC	Ідентифікатор значення
DVH_IDP (FK)	NUMERIC	Ідентифікатор запису значень об’єкту системи в таблиці Dossier_History
GRD_TEXT1	NVARCHAR	Текстове поле
…	…	…
GRD_TEXT30	NVARCHAR	Текстове поле
GRD_DATE1	DATETIME	Поле для збереження дати
…	…	…
GRD_DATE10	DATETIME	Поле для збереження дати
GRD_NUM1	NUMERIC	Числове поле
…	…	…
GRD_NUM15	NUMERIC	Числове поле
GRD_FILE1	VARBINARY	Поле для збереження файлів
…	…	…
GRD_FILE10	VARBINARY	Поле для збереження файлів
GRD_LID (FK)	NUMERIC	Ідентифікатор типу спискових значень

Таблиця Client призначена для збереження інформації про клієнтів системи, їх підприємствах. Дані про клієнтів використовуються при підписці клієнта на видання. А дані про підприємства використовуються при відмежуванні даних між різними підприємствами одного клієнта. Таблиця має 5 полів, опис яких представлено в таблиці 7.12.

Таблиця 7.12 – Опис таблиці Client

Назва поля	Тип поля	Опис
CLI_IDP (PK)	NUMERIC	Ідентифікатор клієнта
CLI_NAME	NVARCHAR	Коротка назва клієнта

Продовження таблиці 7.12 – Опис таблиці Client

Назва поля	Тип поля	Опис
CLI_NAME_FULL	NVARCHAR	Повна назва клієнта
CLI_PID (FK)	NUMERIC	Ідентифікатор батьківського запису клієнта
CLI_IDP_Synonym	NVARCHAR	Ідентифікатор підприємства, синонімом якого є запис

Таблиця Client є ієрархічною, тому що за предметною областю системи фармацевтичного нагляду клієнт може мати декілька підприємств, які в свою чергу можуть мати по декілька синонімів.

Таблиця Drug_registration призначена для зберігання інформації про лікарські засоби. Дана таблиця є ключовою в системі з фармацевтичного нагляду. В модулі моніторингу вона використовується для створення запитів до служби планування, а потім до служби пошуку. Таблиця має 36 полів. В таблиці 7.13 описані основні поля таблиці Drug_registration, які потрібні для виконання пошуку.

Таблиця 7.13 – Опис полів таблиці Drug_registration

Назва поля	Тип поля	Опис
id (PK)	NUMERIC	Ідентифікатор лікарського засобу
trade_name	NVARCHAR	Назва лікарського засобу
form	NVARCHAR	Форма випуску лікарського засобу
dic_declarer_id (FK)	NUMERIC	Ідентифікатор заявника ЛЗ
Назва поля	Тип поля	Опис

Продовження таблиці 7.13 – Опис полів таблиці Drug_registration

country_drug_identification_id (FK)	NUMERIC	Ідентифікатор ЛЗ в країні
lls_idp	NUMERIC	Ідентифікатор мови

Таблиця Active_substances призначена для збереження діючих речовин. Таблиця має вигляд звичайного довідника з ідентифікатором значення та його

текстовим представленням. Діючі речовини використовуються при моніторингу лікарських засобів (їх можна включити в ключові слова пошуку), а також окремо здійснювати моніторинг по діючим речовинам.

Таблиця Reference_drug призначена для збереження інформації про референтні лікарські засоби. Система дозволяє здійснювати моніторинг по референтним лікарським засобам. Таблиця має 5 полів, опис яких представлено в таблиці 7.14.

Таблиця 7.14 – Опис таблиці Reference_drug

Назва поля	Тип поля	Опис
id (PK)	NUMERIC	Ідентифікатор референтного лікарського засобу
trade_name	NVARCHAR	Назва референтного лікарського засобу
form	NVARCHAR	Форма випуску референтного лікарського засобу
manufacturer	NUMERIC	Виробник референтного лікарського засобу
lls_idp	NUMERIC	Ідентифікатор мови

Таблиця Session_request призначена для збереження інформації про запити на пошук. Запити на пошук формуються перед початком сеансу пошуку або після завершення попереднього сеансу пошуку. Таблиця має 11 полів, опис яких представлено в таблиці 7.15.

Таблиця 7.15 – Опис полів таблиці Session_request

Назва поля	Тип поля	Опис
id (PK)	INT	Ідентифікатор запиту
Session_ID (FK)	INT	Ідентифікатор сесії
Scheduled_plan_ID (FK)	INT	Ідентифікатор план-графіку
Planner_ID	UNIQUEIDENTIFIER	Ідентифікатор планувальника на стороні служби планування
Object_search_id (FK)	INT	Ідентифікатор об’єкта пошуку
Request_state (FK)	TINYINT	Ідентифікатор стану запиту
Session_scheduled_date_and_running_time	DATETIME	Запланована дата виконання запиту
Session_actual_date_and_running_time	DATETIME	Фактична дата виконання запиту
Cron_expression	NVARCHAR	Значення крон-виразу
Error_text	NVARCHAR	Текст помилки, якщо вона виникла під час виконання запиту
Keywords	NVARCHAR	Ключові слова в JSON форматі
Object_search_type_id (FK)	INT	Тип об’єкта пошуку

Після запуску план-графіку створюються сеансові запити для найближчого сеансу моніторингу. В таблиці Session_request заповнюються поля всі поля окрім полів: Planner_ID, Session_actual_date_and_running_time, Error_text. Ці поля заповнюються після успішного виконання пошуку і отримання результатів клієнтом моніторингу. Якщо під час пошуку виникла помилка, то в поле Error_text записується її повний опис. При створенні запиту, поле Request_state,

має ідентифікатор стану запиту, що відповідає не запущеному запиту. Після запуску це поле отримує значення, яке показує, що запит виконується. Коли приходить результат пошуку, то запит редагується і поле Request_state буде містити ідентифікатор завершеного стану запиту, а в поле Session_actual_date_and_running_time записується дата виконання запиту.

Таблиця Dictionaries призначена для зберігання інформації про системні словники. Таблиця містить 3 поля, опис яких представлено в таблиці 7.16.

Таблиця 7.16 – Опис таблиці Dictionaries

Назва поля	Тип поля	Опис
id (PK)	INT	Ідентифікатор значення словника
code	NVARCHAR	Кодова назва словника
name	INT	Назва словника

Таблиця Sysdict призначена для зберігання значень системних словників. Таблиця має 4 поля, опис яких представлено в таблиці 7.17.

Таблиця 7.17 – Опис полів таблиці Sysdict

Назва поля	Тип поля	Опис
id (PK)	INT	Ідентифікатор значення словника
value	NVARCHAR	Текстове значення словника
dictionary_id (FK)	INT	Ідентифікатор словника
sort_order	INT	Порядок значень в словнику

Таблиця LibraryContent призначена для збереження завантаженого контенту видання. Таблиця має 7 полів, опис яких представлено в таблиці 7.18.

Таблиця 7.18 – Опис полів таблиці LibraryContent

Назва поля	Тип поля	Опис

id (PK)	INT	Ідентифікатор контенту
literature_id	INT	Ідентифікатор видання
url	NVARCHAR	Посилання на контент
hash	INT	Хеш-код контенту
u_date	DATETIME	Дата оновлення
version	INT	Версія
is_indexed	INT	Стан індексації

Таблиця ContentKeyword призначена для збереження інформації про кількість входження ключових слів в статті. Таблиця має 3 поля, опис яких представлено в таблиці 7.19.

Таблиця 7.19 – Опис полів таблиці ContentKeyword

Назва поля	Тип поля	Опис
kw_id	INT	Ідентифікатор ключового слова
content_id	INT	Ідентифікатор контенту
count	INT	Кількість входжень

Висновки до розділу

В даному розділі було розглянуто архітектурне рішення щодо розробки системи вцілому. Система поділяється на клієнтську та серверну частини. Клієнтська частина робить запити на пошук до серверної частини та отримує на прийомники результати пошуку. Клієнтська частина складається з модуля планування та веб-додатку, у якому відображається вся інформація. Серверна частина складається з електронної бібліотеки у якості веб-додатку, служби пошуку та служби обробки результатів скриптами Python. Проектування системи відбувалося із дотриманням правил SOLID та з використанням шаблонів проектування. Структура бази даних реляційна і налічує 19 основних таблиць.

8 РЕАЛІЗАЦІЯ СПРОЕКТОВАНИХ СИСТЕМ

8.1 Узагальнені принципи роботи системи

Функціональна схема системи у вигляді діаграми взаємодії зображена у додатку Б. Вона відображає взаємодію між модулями та їх логічні зв'язки.

Для запуску пошуку потрібно мати як мінімум одне видання, завантажене з бібліотеки та один лікарський засіб. Якщо ці дані присутні в системі, то потрібно створити пакет моніторингу. Після цього потрібно створити план-графік для обраного видання. [19]

Після створення план-графіку у користувача є можливість створити сеанс моніторингу і запустити план-графік на автоматичне виконання або виконати миттєвий пошук.

Розглянемо алгоритм автоматичного пошуку.

Автоматичний пошук починається в той момент, коли запускається план-графік на автоматичне виконання. В цей момент в базі даних створюються записи про сеансові запити. Ці запити містять інформацію про ключові слова пошуку, лікарський засіб та джерело пошуку у вигляді URL-адреси. Під час роботи веб-застосунку працює таймер, який в по зазначеному в конфігураційному файлі параметрі тайм-ауту здійснює запити в базу даних для пошуку сеансових запитів, які ще не були запущені та не є простроченими. Схема алгоритму зображена у додатку Є.

Після отримання набору сеансових запитів, вони відправляються на службу планувальника у вигляді правил пошуку. На службі планувальника правила пошуку зберігаються в базі даних і їм надається ідентифікаційний код, який віддається клієнту моніторингу і зберігається в базі даних в таблиці сеансових запитів. [19]

На стороні служби планування запускаються на кожне правило пошуку таймери і в необхідні періоди часу (які задаються клієнтом) виконують запит до служби пошуку. Служба пошуку отримує необхідну інформацію від служби планування і в залежності від провайдера пошуку формує строку пошуку або

передає необхідні параметри. В даній системі провайдерами пошуку є Google та Google Search API.

Отримавши строку пошуку чи необхідні параметри служба пошуку конфігурує провайдера пошуку та передає йому необхідну інформацію. У випадку використання Google Search API відбувається виклик через цей інтерфейс і служба отримує результати пошуку в специфічній структурі даних.

У випадку використання звичайної пошукової системи Google система пошуку застосовує службу браузера. В цьому випадку в службу браузера передається строка запиту. Далі служба за допомогою внутрішніх механізмів отримує веб-сторінку з результатами пошуку та віддає її назад в службу пошуку.

Після отримання результатів пошуку службою пошуку, вона знаходить хендлер передачі результату в електронну бібліотеку. Прийом результатів пошуку електронною бібліотекою відбувається за допомогою служби прийому результатів пошуку. В електронній бібліотеці результати зберігаються в базі даних. Після цього результати пошуку аналізуються за допомогою служби виконання скриптів Python. Скрипти пишуться для кожного видання окремо для отримання необхідної інформації про знайдену публікацію. Після аналізу скриптами, відбувається фінальна обробка результатів пошуку електронною бібліотекою. [19]

В цей же час служб пошуку паралельно з передачею результатів пошуку в електронну бібліотеку, передає результати в службу планувальників. Служба планувальників конфігурує необхідний адаптер для передачі результатів на клієнт моніторингу видань. Передача результатів відбувається за допомогою служби завантаження результатів моніторингу. На клієнті результати зберігаються частково в базі даних. Після збереження результатів клієнт моніторингу видань робить запит до служби отримання публікацій електронної бібліотеки, а також робить маніпуляції для очищення бази даних від непотрібних сеансових запитів та створення нових сеансових запитів. Отримання публікації відбувається по ідентифікаційному коду публікації. Після отримання проаналізованої публікації до клієнта моніторингу можна вважати процес автоматичного пошуку завершеним. [19]

Алгоритм роботи миттєвого пошуку дуже схожий на алгоритм автоматичного виконання пошуку, але має деякі відмінності. По-перше при натисканні кнопки миттєвого пошуку, створюється сеанс пошуку, для сеансу пошуку створюються сеансові запити. Далі ці запити віддаються на службу планувальників, де на кожен запит створюється одноразовий таймер (після відпрацювання таймер утилізується), далі повторюється алгоритм автоматичного пошуку. По-друге, при завантаженні результатів пошуку клієнту з бази видаляються дані про сеанс та сеансовий запит, а також видаляється інформація з бази даних служби планувальника. Весь перебіг даних між компонентами системи зображено у додатку Е.

Після розгляду узагальненого алгоритму роботи системи, взаємодії модулів, потрібно розглянути кожен модуль більш детально.

8.2 Реалізація клієнтської частини

8.2.1 Сутності предметної області

Основними сутностями предметної області системи є: лікарські засоби, референтні лікарські засоби, діючі речовини, видання, результати моніторингу, пакети моніторингу, план-графіки, сеанси, сеансові запити, правила пошуку.

Видання є представленням тих джерел інформації, які потрібно використовувати під час пошуку. Картка має 16 полів, які повністю описують видання. Всі поля сутностей описуються в таблиці Dossier_Attribute, а їх значення в таблиці Dossier_Value. Такі поля як електронна адреса видання, періодичність виходу видання та ідентифікатор в електронній бібліотеці використовуються при створенні план-графіків і подальшому виконанні моніторингу.

Пакети моніторингу використовуються для створення план-графіків і подальшому створенні сеансових запитів. Пакет моніторингу складається з набору видань, набору лікарських засобів або референтних лікарських засобів або діючих речовин та ключових слів. Пакети моніторингу мають початкову та кінцеву дати дії.

План-графіки є ключовими сутностями системи. План-графіки створюються для конкретних видань на певний рік. Вони містять пакети моніторингу, за якими буде здійснюватися моніторинг. Також план-графіки можна налаштувати на виконання з певні періоди часу за допомогою сеансів план-графіку. Сеанси можна створити автоматично на весь рік починаючи від поточної дати. При розрахунках планових дат виконання план-графіку буде прийматися періодичність видання. Також є можливість в ручному режимі створювати сеанси на необхідні користувачу дати. Сеанс має признак виконання, тому користувач завжди буде знати стан виконання сеансу. План-графік також має стан виконання як і сеанс. Сторінка план-графіку відображена на рисунку 8.1.

План - графики мониторинга литературы

Предприятие:
ООО "Фарма Старт"
Название план-графика:
Сгенерированный для пакета мониторинга: "Пакет по 4 ЛС на все издания" -11

Дата создания:	Издание:	Периодичность выхода издания:	Год:	Состояние активации:
19.06.2017	PubMed	Ежемесячно	2017	Активирован

Комментарий:

Пакеты мониторинга литературы

Полное название пакета	Тип пакета	Объект мониторинга	Дата включения пакета в план-график	Пользователь
Пакет по 4 ЛС на все издания	Лекарственное средство (ЛС)	ПАРОКСИН/Paroxin, ЕСЦИТАМ 10/ESCITAM 10, ДІОКОР 80/DIOCOR 80, КВЕТИРОН 25/Quetiron 25	19.06.2017	Адмін Адмін Адмін

Сеансы план-графика

Плановая дата выполнения сеанса	Плановое время выполнения сеанса	Количество дней	Признак формирования записи	Пользователь	Признак выполнения сеанса	Фактическая дата выполнения	Фактическое время выполнения	Комментарий
17.11.2017	00:00	30	Программа	Адмін Адмін Адмін	Не активный			
18.10.2017	00:00	30	Программа	Адмін Адмін Адмін	Не активный			
18.09.2017	00:00	30	Программа	Адмін Адмін Адмін	Не активный			
19.08.2017	00:00	30	Программа	Адмін Адмін Адмін	Не активный			
20.07.2017	00:00	30	Программа	Адмін Адмін Адмін	В ожидании			

Рисунок 8.1 – Сторінка план-графіка

Після запуску план-графіку для найближчого сеансу створюються сеансові запити. Сеансовий запит містить в собі набір ключових слів пошуку, джерело пошуку та крон-вираз. Крон-вираз дозволяє в компактній формі представляти дату виконання пошуку.

Правила пошуку отримуються із сеансових запитів та містять в собі інформацію про ключові слова, дату пошуку, стартовий індекс сторінки пошуку, адрес сайту, ім’я обробника пошуку, ім’я провайдера пошуку, признак активації, додаткові параметри.

Після виконання успішного пошуку служба отримання результатів моніторингу отримує результати пошуку та зберігає їх в таблиці результатів

моніторингу. Результат моніторингу містить посилання на реальний пошуковий запит Google та посилання на видання. Також результат містить знайдені публікації. Публікацією вважається кожне посилання результату пошуку Google. Публікація описується основними полями такими як: назва, реферат, автори, дата публікації. За можливості до публікації прикріплюється файл у форматі HTML, якщо результатом пошуку є веб-сторінка, або файл у форматі PDF, якщо було знайдено файл у форматі PDF.

8.2.2 Реалізація основних інтерфейсів, сервісів та контролерів веб-модулю

Веб-модуль реалізовано згідно з шаблоном MVC з використанням технологія ASP.NET MVC. Архітектуру модулю було представлено в розділі 7.2.1. За цією архітектурою інтерфейси декларують основні операції з сутностями системи, сервіси їх реалізовують. Контролери використовуючи сервіси відображають інформацію в представленнях. Інстанціація репозиторіїв та сервісів відбувається з використанням IoC контейнера Unity. В сервісах та контролерах використовуються інтерфейси, реалізація яких впроваджується контейнером Unity. В даному випадку використовується п’ятий принцип SOLID, а саме впровадження залежностей.

8.2.3 Принцип роботи автоматизованого план-графіку

Для запуску автоматичного план-графіку потрібно спочатку завантажити видання з електронної бібліотеки. Для того, щоб завантажити видання, необхідно попередньо підписатися на потрібні видання електронної бібліотеки. Далі потрібно створити пакет моніторингу, який включатиме це видання, перелік об’єктів пошуку та перелік ключових слів. Після цього можна створювати план-графік. Для створення план-графіку потрібно обрати видання, створений пакет моніторингу та визначити як мінімум один сеанс моніторингу.

Як тільки створення план-графіку завершилося успішно, його можна активувати. Після активації план-графіку створюються сеансові запити на кожен об’єкт пошуку, які вказано в пакетах моніторингу до план-графіку.

В автоматизації відправки запитів на сторону служби планування у веб-модулі було створено таймер, який стартує під час старту веб-застосунку. Кожен період часу, який встановлюється в конфігураційному файлі, таймер викликає метод пошуку доступних для відправки на службу планування сеансових запитів. В таблиці Session_request робиться пошук всіх запитів, планова дата яких більша за поточну дату і менша встановленого тайм-ауту. В даному випадку тайм-аут встановлено у розмірі 30 днів. Таке обмеження потрібно для зменшення навантаження на службу планування. Після вибору сеансових запитів, вони групуються за ідентифікатором план-графіку. Далі кожна група сортується по даті планового виконання і вибирається по одному першому запиту з кожної групи. По ідентифікатору сесії цих запитів вибираються інші запити та додаються до загального списку.

Відібрані сеансові запити перетворюються в правила пошуку. Правила пошуку містять все необхідну інформацію для виконання пошуку. Ключові слова перетворюються в список, сайт пошуку береться з даних видання, ім’я обробника пошуку та ім’я провайдера пошуку береться з конфігураційного файлу. Додаткові параметри містять: ідентифікатор видання в електронній бібліотеці, ім’я підприємства, мова план-графіку, тип та назва об’єкта пошуку. Додаткові параметри необхідні для створення записів про результати пошуку на стороні електронної бібліотеки.

Далі сформовані правила пошуку поділяються на порції по 32 правила і відправляються на службу планування. Служба планування на кожне правило пошуку повертає ідентифікатор в форматі GUID. Цей ідентифікатор зберігається в полі Planner_ID таблиці Session_request і пов’язує сеансові запити з планувальниками на стороні служби планування. Коли це поле заповнене, то наступні сеанси, вже не будуть створювати нові планувальники на стороні служби, а оновлювати дані існуючих планувальників.

Алгоритм роботи служби планування та інших компонентів пошуку і прийому результатів описано в наступних підрозділах.

8.2.4 Принцип роботи миттєвого пошуку

Принцип роботи миттєвого пошуку відрізняється від автоматичного тим, що план-графік не обов’язково повинен мати створений сеанс пошуку. При запуску миттєвого пошуку одразу створюються сеанс моніторингу та сеансові запити. Далі сеансові запити перетворюються на правила пошуку і відправляються на службу планування. Далі алгоритм роботи нічим не відрізняється від алгоритму при автоматичному пошуку. Єдина відмінність є при отриманні результату, бо тоді викликається метод отримання результату миттєвого пошуку.

8.2.5 Реалізація служби планування та принципи її роботи

Служба планування реалізована за допомогою засобів WCF. Для початку службу потрібно встановити за допомогою файлу ExSearch.Service.Install.bat. Після встановлення служби за допомогою Windows Service Manager потрібно запустити службу. Після першого запуску служби створюється файл бази даних і виконується скрипт створення структури бази даних. База даних створюється за принципом CodeFirst. Це означає, що всі сутності бази даних конфігуруються в коді і представлені у вигляді наступних класів: Log, SearchQuery, SearchResult, SearchRule.

Перед запуском служби інстанціюються всі сервіси та менеджери за допомогою контейнера Unity в тому числі клас для логування, який інстанціюється першим. При інстанціації менеджера планувальників із бази даних витягаються всі правила пошуку і зберігаються в потокобезпечному списку для подальшого запуску таймерів. Далі безпосередньо запускається сама служба викликом методу ServicBase.Run. Старт служби запускає обробники результатів пошуку, планувальники пошуку та служби WCF. Обробники пошуку

запускаються і конфігуруються виходячи із конфігураційних секцій конфігураційного файлу. Налаштуваннями конфігураційної секції приведені в таблиці 8.1.

Таблиця 8.1 – Опис параметрів секції налаштування обробника пошуку

Назва параметру	Опис
name	Назва секції налаштувань
addressSearchResultService	Повна адреса служби завантаження результатів
proxyConfigFromWCFSection	Признак конфігурування проксі-налаштувань із секції налаштувань клієнтів WCF
proxyEnable	Признак увімкнення використання проксі-сервера
proxyHost	Ім’я проксі-сервера
proxyPort	Порт проксі-сервера

Після запуску обробників пошуку, запускаються планувальники пошуку для кожного правила пошуку. По своїй суті планувальники виглядають як таймери. В даному випадку використовується CronTimer. На кожне правило пошуку створюється і запускається таймер. При створенні таймеру, йому передаються параметри такі як: обробник результату, тип запуску таймера та крон-вираз. На подію спрацювання таймеру підписано метод, який створює об’єкти запитів на пошук і викликає метод пошуку служби пошуку. Запит на пошук представлено у вигляді класу SearchRequest, опис полів якого представлено в таблиці 8.2.

Таблиця 8.2 – Опис полів класу SearchRequest

Назва поля	Опис
Keywords	Ключові слова пошуку
WebSite	Веб-сайт, на якому потрібно здійснювати пошук
StartSearchDate	Початкова дата періоду пошуку
FinalSearchDate	Кінцева дата періоду пошуку
ExcludeKeywords	Ключові слова, які потрібно виключити з пошуку
Ordering	Признак сортування

SearchCountry	Країна пошуку
SearchLanguage	Мова пошуку
StartIndex	Індекс сторінки з результатами пошуку
FileFormat	Формат файлу пошуку, якщо здійснюється пошук файлів
IsCaseSensitive	Признак точного пошуку
IsSafeSearch	Признак безпечного пошуку
AdditionalRequestParams	Додаткові параметри пошуку

Клас SearchRequest відображає можливі фільтри налаштування пошуку Google в сукупності з розширеним пошуком Google. На рисунку 8.1 зображено фільтри пошуку на сторінці Google, а на рисунку 8.2 – розширені налаштування пошуку.

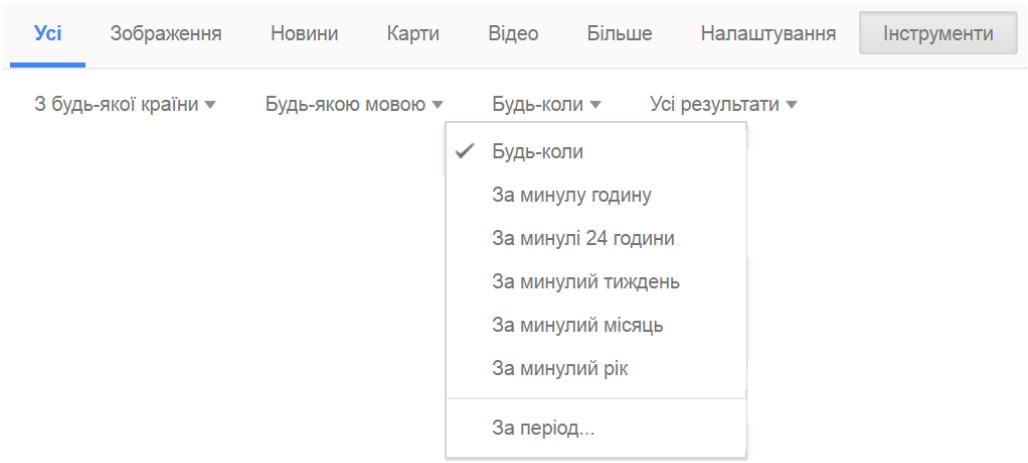


Рисунок 8.1 – Фільтр пошуку Google

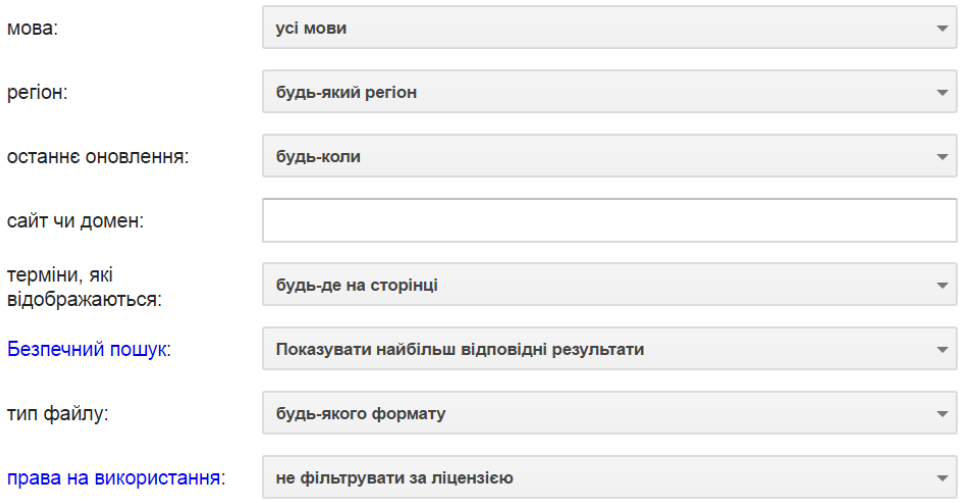


Рисунок 8.2 – розширені налаштування пошуку Google

У відповідь на виклик служби пошуку отримуємо відповідь на пошуковий запит у вигляді об’єкту типу SearchResponse який містить відомості про результати пошуку на сторінці пошуку Google. Опис типу SearchResponse наведено у таблиці 8.3.

Таблиця 8.3 – Опис типу SearchResponse

Назва поля	Опис
Description	Опис відповіді на пошуковий запит
Items	Список результатів пошуку
ItemsPerPage	Кількість результатів на сторінці

Продовження таблиці 8.3 – Опис типу SearchResponse

Назва поля	Опис
Link	Посилання на пошуковий запит
ExcludeKeywords	Ключові слова, які потрібно виключити з пошуку
Request	Пошуковий запит
ResultGuid	Ідентифікатор відповіді на пошуковий запит
StartIndex	Стартовий індекс сторінки результатів
Title	Ім’я провайдера пошуку
TotalResults	Загальна кількість результатів пошуку

Далі цей результат передається обробнику результатів пошуку, який в свою чергу викликає службу прийому результатів моніторингу, що знаходиться разом з веб-модулем клієнтської частини системи.

Коли служба вже в запущеному стані, вона представляє для веб-модулю клієнтської частини програмний інтерфейс для взаємодії. Служба представляє методи створення, оновлення, видалення, активації, деактивації, отримання правил пошуку.

Під час роботи автоматизованого план-графіку, служба планування періодично приймає виклики методів створення або оновлення правил пошуку. Після створення правил пошуку відбувається запуск таймеру для правила пошуку і алгоритм, описаний вище, повторюється. Якщо правило оновлюється, то змінюється крон-вираз таймеру і він запускається знову. Після повного виконання план-графіку, веб-модуль викликає метод видалення правил пошуку на стороні служби.

Якщо користувач запускає миттєвий пошук, то на стороні служби планування спочатку реєструється правило пошуку, а веб-модулю повертається ідентифікатор планувальника. Після цього веб-модуль ініціює виклик миттєвого пошуку, вказавши ідентифікатор планувальника. При цьому на стороні служби миттєво спрацьовує таймер і виконується виклик методу пошуку служби пошуку. Після отримання результатів від служби пошуку, служба планування відправляє їх на службу отримання результатів моніторингу за допомогою виклику методу SendResultInstantClient.

8.2.6 Реалізація служби прийому результатів моніторингу

Служба прийому результатів моніторингу має 2 методи на прийом результатів: метод прийому результатів автоматичного пошуку та метод прийому результатів миттєвого пошуку.

Метод прийому результатів миттєвого пошуку дещо відрізняється від методу прийому результатів автоматичного пошуку.

Відмінності полягають в тому, що після отримання результату миттєвого пошуку з бази даних вибирається сеансовий запит і на основі результату пошуку і сеансового запиту створюється результат моніторингу, а сеансовий запит як і сеанс моніторингу видаляються з бази даних. Коли приймається результат автоматичного пошуку необхідно в базі даних змінити статуси сеансового запиту, сеансу та план-графіку. Якщо всі сеансові запити завершились, то вважається що сеанс також закінчився. Якщо закінчуються всі сеанси план-графіку, то вважається що план-графік також завершено. Ще однією відмінністю прийому результатів автоматичного пошуку є формування нових сеансових запитів та видалення старих після збереження результату моніторингу. Також після цього запускається метод пошуку доступних на виконання сеансових запитів.

Спільним між методами прийому результатів пошуку є процес створення результату моніторингу. Спочатку створюється результат моніторингу в базі даних на основі сеансового запиту та результатів пошуку. Після створення результату моніторингу видаляється сеансовий запит і якщо до даного сеансу більше немає пов’язаних сеансових запитів, то видаляється й сеанс із бази даних. Після цього запускається процес отримання публікації до результату моніторингу. Публікації отримуються із електронної бібліотеки за допомогою служби отримання публікацій. Публікації отримуються за ідентифікатором публікації. Ідентифікатор публікаціям визначила служба планування перед відправкою результатів на сторону веб-модулю. Для забезпечення стабільної роботи служби завантаження публікацій було створено семафор на 16 потоків на кожне ядро процесору. Після отримання публікації від служби завантаження публікацій, вона зберігається в базі даних, оновивши існуючу публікації з таким самим ідентифікатором. На цьому прийом результатів моніторингу закінчується.

8.3 Реалізація серверної частини

8.3.1 Сутності предметної області

Основними сутностями предметної області системи є: видання, результати пошуку, публікації, правила аналізу та колекції підписки.

Видання в електронній бібліотеці потрібні для створення підписок клієнтів на моніторинг певних видань, а також гарантують використання при виконанні моніторингу клієнтом. Картка має 16 полів, які повністю описують видання. Всі поля сутностей описуються в таблиці Dossier_Attribute, а їх значення в таблиці Dossier_Value.

Результати пошуку відображають всі результати пошуків, які виконувалися службою пошуку. Результат пошуку містить інформацію про об'єкт пошуку, видання пошуку та знайдені публікації.

Публікації в електронній бібліотеці показують конкретну інформацію, знайдену за допомогою служби пошуку. Основними полями публікації є: назва, автори, реферат, ключові слова. До публікацій за можливості прикріпляється файл публікації у форматі HTML або PDF.

8.3.2 Реалізація основних інтерфейсів, сервісів та контролерів

Веб-модуль реалізовано згідно з шаблоном MVC з використанням технологія ASP.NET MVC. Так як веб-модулі клієнтської та серверної частини знаходяться в одному рішенні, тому реалізація дуже схожа і концептуально та архітектурно нічим не відрізняється.

8.3.3 Реалізація служби пошуку та принципи її роботи

Служба реалізована за допомогою технології WCF. Встановлення та запуск служби відбувається таким самим чином як і вже описаної служби планування. При запуску служби запускаються всі браузери (вони використовуються при пошуку), запускаються обробники пошуку, запускаються завантажники пошуку та стартують служби WCF (в тому числі і служба браузерів).

Служба пошуку представляє єдиний метод Search, який приймає пошуковий запит описаний в таблиці 7.2 та ім'я провайдера пошуку. Існує два

провайдера пошуку в системі: пошук через Google custom search API та через веб-пошук з використанням браузерів.

Першою реалізовувалася взаємодія з Google custom search API. Для взаємодії використовувався набір бібліотек Rest4Net, а також тип SearchParameters, який пропонує провайдер пошуку. Обов'язковими параметрами є ключові слова та сайт пошуку. Для того, щоб здійснювати пошуку через цього провайдера потрібно мати ключ-ідентифікатор додатку Google, а також ідентифікатор користувацького пошуку Google. Після конфігурування провайдера пошуку, та передачі в його метод пошуку запиту отримуємо у відповідь об'єкт типу SearchResult, який також пропонує провайдер пошуку.

Під час тестування реалізації пошуку через Google custom search API було виявлено деякі недоліки даного провайдера пошуку. Провайдер пошуку не дозволив виконувати пошук за період дат, а лише за останні дні. Також якість пошуку дуже сильно відрізнялася від звичайного пошуку Google через веб-інтерфейс. Тому було вирішено реалізувати пошук через веб-інтерфейс. Для цього реалізовувалася служба браузерів, яка дозволяла перейти на сторінку, виконати всі скрипти на цій сторінці та завантажити її у форматі HTML.

Пошук за допомогою служби браузерів відбувається за наступним сценарієм. Спочатку перевіряється наявність ключових слів та веб-сайту для пошуку. Потім починається формуватися строка пошуку. Правила формування строки пошуку надає провайдер пошуку (Google). Спочатку в строку вставляється адреса провайдера пошуку. За адресом провайдера пошуку йде основна частина строки пошуку у форматі який сприймає Google. Ключові слова склеюються через оператор «OR» строки пошуку. Потім до строки додається адреса веб-сайту пошуку. Далі додається період пошуку. Також можна в строку додати країну пошуку, мову пошуку та інші параметри, які зазначені в таблиці 7.2. Після формування строки пошуку викликається метод служби браузерів Download якому передається ця строка. Метод Download повертає контент сторінки пошуку у форматі HTML.

Після отримання контенту, він форматується у формат XML за допомогою класу SgmlReader. Після цього з файлу результату пошуку видаляються всі

скрипти та стилі, тому що вони не потрібні в результатах пошуку. Отримавши коректний XML файл, служба викликає метод парсингу результатів пошуку у формат, який потребує служба прийому результатів пошуку в електронну бібліотеку. Готові результати пошуку у вигляді об’єкта типу SearchResponse віддаються на службу прийому результатів пошуку електронної бібліотеки.

8.3.4 Реалізація та принципи роботи служби браузера

Служба браузера стартує разом зі службою пошуку. Там же стартують всі браузери. Кожен браузер знаходиться в окремому процесі. Кількість запущених браузерів залежить від кількості ядер процесору. На кожне ядро призначено по 2 браузера. Взаємодія браузерів відбувається по протоколу IPC. Під час старту процесу браузера, стартують IPC прийомник та IPC відправник.

У якості реалізації браузеру використовується бібліотеки ChomiumFX та ChromiumWeb. Коли викликається метод Download служби браузерів, служба формує повідомлення і делегує це повідомлення вільному браузеру. Браузер в залежності від команди, вказаній в повідомленні, виконує певні дії. Може бути 3 команди: пінг, завантаження та вихід. Пінг призначений для перевірки досяжності браузера. Команда завантаження призначена для переходу на необхідну сторінку і її завантаження. Команда виходу призначена для виходу з браузеру. Для забезпечення успішного завантаження сторінки, синхронізуються всі події браузеру, які виникають при завантаженні сторінки.

Результат завантаження спочатку отримує служба браузерів, а потім служба пошуку.

8.3.5 Реалізація служби отримання результатів пошуку

Після виконання пошуку службою пошуку, результат передається в службу отримання результатів пошуку на стороні електронної бібліотеки. Служба має 2 методи: прийом результатів автоматичного пошуку, прийом

результатів миттєвого пошуку. Реалізація методів майже однакова, за винятком різних типів виконання пошуку (миттєвий, автоматичний).

На основі прийнятого результату пошуку службою пошуку, в базу даних зберігається запис про результат моніторингу та записи про знайдені публікації. Далі для кожної публікації за допомогою служби браузерів завантажується сторінка і у вигляді файлу прикріплюється до публікації. На основі цього файлу і ключових слів, які використовувалися при виконанні пошуку, формується реферат публікації. Далі файл публікації змінюється для підсвічення знайдених ключових слів. Реферат формується за принципом: речення, в якому зустрічається ключове слова плюс попереднє та наступне речення. Після цього викликається метод аналізу публікації служби виконання скриптів Python. В результаті роботи цього методу публікація аналізується і з прикріпленого файлу за попередньо сформованими правилами витягається назва, автори та дата публікації. Весь процес роботи з публікацією виконується в окремому потоці синхронізованому на семафорі. Такий вид синхронізації потрібен для забезпечення стійкості служби.

Після проведення всіх цим маніпуляцій процес прийому результату пошуку вважається завершеним, а публікація готова для завантаження через службу отримання публікацій.

8.3.6 Реалізація служби отримання публікацій

В підрозділі 7.2.6 було описано процес отримання результатів пошуку у веб-модуль клієнтської частини системи. Наприкінці процесу було описано, що публікації отримуються за ідентифікатором з електронної бібліотеки за допомогою служби отримання публікацій.

Служба отримання публікацій реалізована таким чином, що коли веб-модуль клієнтської частини викликає метод служби на отримання публікації, поток виконання очікує в службі доки публікація не пройде наступні етапи:

1) завантаження файлу публікації;
2) отримання реферату публікації;

3) підсвічування знайдених ключових слів;
4) аналіз публікації набором скриптів Python.

Після проходження цих етапів публікація стає доступною, потом виконання відпускається і веб-модуль клієнтської частини отримує готову публікацію.

8.3.7 Реалізація служби підписки на видання

Служба підписки на видання призначена для підписки клієнтів на видання. Підписка здійснюється на основі ідентифікатора клієнта. Процес підписки працює наступним чином. На стороні електронної бібліотеки створюється колекція підписки для конкретного клієнта. Колекція складається з набору видань. Далі колекція активується, і признак активації зберігається в базі даних. Служба має 2 методи. Перший метод призначений для перевірки наявності доступних видань. Другий метод призначений для завантаження видань з електронної бібліотеки в клієнтську частину системи.

Коли користувач, який працює з веб-модулем клієнтської частини, заходить на сторінку видань, робиться виклик методу перевірки доступних видань. Якщо доступні видання є, то у веб-інтерфейсі розблоковується кнопка завантаження видань з електронної бібліотеки.

Після натиснення на кнопку завантаження викликається метод завантаження публікацій і публікації завантажуються в базу даних клієнта.

8.3.8 Реалізація служби виконання скриптів Python

Служба виконання скриптів Python призначена для аналізу знайдених публікацій за допомогою спеціальних правил, які містяться в скриптах Python. Служба реалізована за допомогою технології WCF і являє собою багатопотокову обгортку навколо середовища виконання скриптів Python.

Процес встановлення та запуску служби однаковий зі службою пошуку. В якості реалізації середовища виконання скриптів Python обрано клас

Python27API. Після запуску служби запускаються обробники інтерпретатора Python та запускається служба WCF.

Взаємодія обробників інтерпретатора Python відбувається з використанням протоколу IPC. Принципи взаємодії по даному протоколу описані в підрозділі 8.3.4.

Служба має методи додавання, оновлення, видалення, отримання набору скриптів. Також служба має метод запуску скриптів на виконання. Всі файли скриптів зберігаються в файловій системі, а відомості про ці файли зберігаються в локальній базі даних. Кожен набір скриптів має унікальний ідентифікатор, який при виклику методу додавання скриптів повертається стороні, яка його викликала.

Основний метод запуску набору скриптів на виконання приймає на вхід наступні параметри: ідентифікатор набору скриптів, файл для аналізу у вигляді байтового масиву та параметри для скриптів. Безпосереднє виконання скриптів делегується класу Python27API. Після виконання скрипта, файл публікації оновлюється і з ним можна працювати далі. На цьому робота служби виконання скриптів Python закінчується.

8.3.9 Реалізація служби фонового завантаження та індексації видань бібліотеки

Служба фонового завантаження та індексації видань бібліотеки призначена для періодичного завантаження контенту веб-сторінок сайту, їх збереження у базі даних та індексації відносно ключових слів для забезпечення альтернативного варіанту пошуку інформації. Схема компонентів підсистеми фонового завантаження та індексації зображена у додатку Ж.

Принцип роботи служби достатньо простий та дієвий. Періодично (залежить від настройок в конфігураційному файлі) служба заходить на головну сторінку видання та починає завантажувати сторінку за допомогою служби браузерів. Далі відбувається парсинг всіх посилань на сторінці, які відносяться потосного домену. Рекурсивно служба завантажує контент по посилання і

зберігає його у базі даних. Після завантаження сторінки служба перевіряє чи індексувалася дана сторінка раніше. Якщо так, то вираховується хеш код і якщо він відрізняється від відповідного у базі даних, то сторінка індексується відносно ключових слів та зберігається у базі даних. Старі дані індексації видаляються.

Процес індексації контенту полягає у наступному. В базі даних є записані базові ключові слова, за якими ймовірно буде відбуватися пошук (назви лікарських засобів, хвороб, тощо). При активному використанні системи ключові слова будуть поповнюватися тими, що приходять від клієнтів під зас пошукових запитів. Служба шукає кількість входжень конкретного ключового слова у публікації і записує цю інформацію в базі даних. Далі ці дані будуть використовуватися при пошукових запитах клієнтів і будуть комбінуватися з іншими результатами пошуку від провайдерів пошуку (наприклад Google).

При виконанні дуже активно використовуються можливості багатопоточності та асинхронності .Net фреймворку, так як операції завантаження та індексації достатньо ресурсоємкі по часу виконання та часу очікування результату.

8.3.10 Розгортання служб за допомогою Docker

Для забезпечення високої доступності та гнучкості роботи служб було використано засоби Docker. Для бази данних та кожної служби було створено образ і при розгортанні за допомогою команди «docker compose» було запущено по екземпляру контейнеру на кожну службу.

Так як запити на служби приходять нерівномірно, то було використано «docker swarm» для забезпечення масштабування та балансування навантаження. Тому на деякі служби було запущено по декілька екземплярів контейнерів та налаштовано балансування навантаження. Це означає, що запити на служби будуть розподілятися рівномірно між контейнерами.

Для початку було сворено образ для розгортання бази даних. Розгортання бази даних у контейнері дозволить зменшити час розробнику для налаштування даних для тестування, тому що можна мати різні образи БД для різних випадків

тестування. Також цей крок дозвлить без проблем змінювати структуру бази даних та змінювати самі дані без особливої обережності. Якщо щось піде не так як було заплановано, завжди можна швидко запустити ще один екземпляр контейнеру, а старий знищити.

Для свторення образу бази даних було обрано базовий образ microsoft/windowsservercore, поверх нього було встановлено сервер БД та приєднано файли існуючої чистої БД. Приклад dockerfile-скрипту наведено на рисунку 8.3.

```
FROM microsoft/windowsservercore
LABEL maintainer "Ivan Kornilov"
ENV sql_express_download_url "https://go.microsoft.com/fwlink/?linkid=829176"
ENV sa_password="_" \
    ACCEPT_EULA="_" \
SHELL ["powershell", "-Command", "$ErrorActionPreference = 'Stop'; $ProgressPreference = 'SilentlyContinue';"]

# make install files accessible
ARG source
WORKDIR /mssql
COPY start.ps1 /
COPY dsbase.mdf
COPY dsbase_log.ldf

RUN Invoke-WebRequest -Uri $env:sql_express_download_url -OutFile sqlexpress.exe ; \
        Start-Process -Wait -FilePath .\sqlexpress.exe -ArgumentList /qs, /x:setup ; \
        .\setup\setup.exe /q /ACTION=Install /INSTANCENAME=SQLEXPRESS /FEATURES=SQLEngine /UPDATEENABLED=0 /SQLSVCACCOUNT='NT
        Remove-Item -Recurse -Force sqlexpress.exe, setup

RUN stop-service MSSQL`$SQLEXPRESS;

CMD .\start -sa_password $env:sa_password -ACCEPT_EULA $env:ACCEPT_EULA -attach_dbs="[{'dbName':'DSBase','dbFiles':['./dsbase.
```

Рисунок 8.3 – Dockerfile-скрипт для створення образу БД

Маючи такий файл потрібно побудувати образ за допомогою команди “dokcer build», і вже після цього розгортати контейнер з базою даних. Якщо в процесі роботи необхідно буде доповнити образ необхідною інформацією, то можна створити ще один окремий dockerfile, або виконати команду «docker commit». Дана команда створить образ на основі контейнеру. Для запуску контейнеру потрібно виконати команду «docker run». Команду запуску контейнеру представлено на рисунку 8.4.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\WINDOWS\system32> docker run -it -p 1433:1433 -m 3000M --ip 172.19.96.21 dsbase
```

Рисунок 8.4 – Запуск контейнеру з БД

Для створення образів WCF-служб треба пройти такі ж самі кроки як і для створення образу БД. Обираємо базовий образ microsoft/wcf:4.6.2, задаємо робочу папку, копіюємо бінарні файли служби. Далі вказуємо файл, який буде вхідною точкою з службу. Приклад скрипта створення образу служби наведено на рисунку 8.5.

```
FROM microsoft/wcf:4.6.2
WORKDIR ExSearch
EXPOSE 9007
COPY ExSearch/ .
ENTRYPOINT .\bin\Release\ExSearch.exe
```

Рисунок 8.5 – Скрипт створення Docker-образу служби

Після створення образу служби, можна запустити контейнер за допомогою команди «docker run», але якщо служб декількі і вони посилаються одна на одну, то є більш зручний спосіб запуску через команду «docker compose». На рисунку 8.6 наведено приклад файлу docker-compose.yml.

```
networks:
  service-proxy:
    driver: bridge
  database:
    driver: bridge

volumes:
  exsearch_logs:
  exsearch_core_logs:
  exsearch_python_logs:
  exsearch_indexer_logs:

services:
  exsearch:
    image: exsearch
    ports:
      - 9007:9007
    volumes:
      - exsearch_logs:C:\Logs\ExSearch\ExSearch.log
    networks:
      - service-proxy

  exsearch_core:
    image: exsearch_core
    ports:
      - 9008:9008
    volumes:
      - exsearch_core_logs:C:\Logs\ExSearch\ExSearchCore.log
    networks:
      - service-proxy

  exsearch_python:
```

Рисунок 8.6 – Конфігурація docker compose

Після запуску команди «docker compose» файл зчитується і на кожен сервіс створюється контейнер згідно з конфігурацією.

Висновки до розділу

В даному розділі було розглянуто реалізацію та основні алгоритми роботи системи. Головними можливостями системи є створення пакетів та план-графіків на основі пакетів та виконання сеансів пошуку. Сеанси пошуку можуть бути миттєвими (коли результат віддається відразу) та автоматичними (коли пошукові запити відбуваються за встановленим графіком). Служба пошуку виконує пошук за допомогою фонового браузеру та пошукової системи Google. Результати отримує клієнт та бібліотека. Клієнт отримує публікації в асинхронному режимі лише після обробки відповідною службою обробки результатів. Служби можна хостити як служби Windows так і розгортати їх за допомогою Docker.

9 ТЕСТУВАННЯ СИСТЕМИ

Тестування програмного забезпечення – це процес виміру якості та працездатності розроблюваного програмного продукту. Процес тестування повинен починатися на початку проектування системи. Спочатку треба зробити валідацію та верифікацію вимог до програмного продукту. Далі тестування проводиться під час усього циклу розробки програмного забезпечення.

Тестування поділяється на наступні види:

1) Приймальне тестування. Даний вид тестування проводиться під час передачі розробленого і протестованого програмного продукту замовнику. Замовник проводить кейси тестування і визначає чи відповідає розроблений продукт його вимогам.

2) Функціональне тестування. Даний вид тестування проводиться для визначення відповідності функціоналу та поведінки програми вимогам функціональної специфікації.

3) Тестування безпеки. Проводиться для унеможливлення заходу сторонніх осіб в програму, виконання злоякісних скриптів, порушення цілісності даних.

4) Навантажувальне тестування. Даний вид тестування проводиться для визначення межі можливостей програмного продукту. Наприклад, визначення кількості користувачів, які одночасно можуть працювати з системою.

З точки зору виконання тестування, воно буває ручне (manual testing) або автоматизоване (automated testing).

З точки зору застосування тестування, воно буває, модульне, інтеграційне та системне.

9.1 Автоматизоване тестування.

Суть автоматизованого тестування полягає в написанні коду, який буде перевіряти функціональність системи на відповідність вимогам. Тести запускаються кожен раз перед запуском або публікацією системи. Середовище розробки Visual Studio дозволяю у зручному вигляді контролювати процес виконання тестів (рис. 8.1).

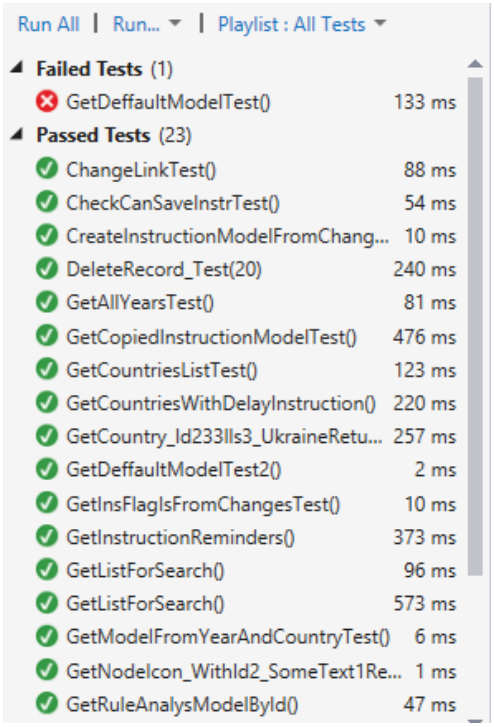


Рисунок 9.1 – Вікно відображення тестів в середовищі Visual Studio

В даній системі автоматизоване тестування застосовувалося для тестування сервісів веб-модулю клієнтської та серверної частини. Модульними тестами покривалися окремі функції сервісів. Інтеграційними тестами покривалися сервіси при взаємодії зі службами. Автоматизоване тестування значно полегшило розробку функціоналу.

Для перевірки стійкості служб до навантаження було проведено навантажувальне тестування на перевірку кількості можливих одночасних запитів на служби. За допомогою даного виду тестування було усунуто проблему падіння служби при високому навантаженні на неї.

9.2 Ручне тестування

Ручне тестування проводиться самим розробником для знаходження тих помилок, які не в змозі виявити автоматизоване тестування. Розробник імітує користувача системи і намагається знайти всі вразливі місця системи.

При розробці системи автоматизованого пошуку для фармацевтичних компаній було проведено ручне тестування, в тих частинах системи, де було проблематично провести автоматизоване тестування. Для тестування використовувалися техніки налагодження коду за допомогою точок зупину і використання слідкування за станом критично важливих змінних. Також в системі широко застосовувалося логування виконання методів системи.

В службах дії методів логуються в окремі файли та в базу даних окремо для кожної служби. В конфігураційних файлах служб знаходяться налаштування формату виводу логів. Після деякого часу відпрацювання системи, файли з логами перечитуються та визначаються вразливі місця системи. Для полегшення пошуку було застосовано мітки логів, такі як ERROR, INFO, DEBUG, WARN. Особливу увагу потрібно звертати на мітку ERROR, бо в повідомленні біля цієї мітки знаходиться інформація про помилку яка виникла в системі.

Переваги застосування логування починають з’являтися, коли система вже знаходиться в експлуатації і немає можливості запустити автоматизовані тести чи провести налагодження коду. В цьому випадку логи дозволяють оперативно знайти помилку, виправити її та представити оновлену версію системи замовнику.

Також тестування проводилося за окремими сценаріями, особливо перевірялося кількість знайдених результатів службою та їх відповідність результатам, які знайшли вручну за допомогою пошукової системи Google. Проводилася перевірка точного завантаження всіх необхідних видань по підписці та можливість виконання пошуку з їх використанням.

Висновки до розділу

Було проведено тестування модулів системи. Тестування відбувалося в автоматизованому режимі (за допомогою юніт тестів та інтеграційних тестів) та в ручному режимі. Для визначення несправностей системи застосовується логування. Система повністю протестована та готова до експлуатації.

10 РОЗРОБЛЕННЯ СТАРТАП-ПРОЕКТУ

10.1 Опис ідеї проекту

Таблиця 10.1 Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Система автоматизованого пошуку фармацевтичної інформації	1. Пошук інформації про ліки	Можливість здійснювати пошук інформації про лікарський засів. Зручне подання у вигляді реєстрів. Пост-обробка інформації. Стисле подання результатів пошуку.
	2. Автоматизація пошуку	Можливість створення запланованих сеансів пошуку. Зручний контроль стану виконання пошукових запитів.
	3. Ведення обліку веб-ресурсів (видань) та їх матеріалів	Всі необхідні об’єкти системи структуровані та легкодоступні. Інформація подається у зручному вигляді. Індексація знайдених матеріалів.

Таблиця 10.2 Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№ п/п	Техніко-економічні характеристики ідеї	(Потенційні) товари/концепції конкурентів / Мій проект	W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
1.	Вартість експлуатації	5 тис. грн/міс.	-	-	+
2.	Вартість обслуговування	5 тис. грн/міс.	-	-	+
3.	Можливість підключення декількох провайдерів пошуку	Присутня	-	-	+
4.	Можливість підключення декількох бібліотек на одну службу планування	Відсутня	+	-	-
5.	Можливість підключення декількох клієнтів до днієї служби планування	Присутня	-	-	+
6.	Забезпечення обробки результатів пошуку	Присутнє	-	-	+
7.	Забезпечення планування сеансів пошуку	Присутнє	-	-	+
8	Наявність веб-інтерфейсу	Присутній	-	+	-
9.	Можливість створювати звіти та проводити аналітику	Присутній	-	-	+

№ п/п	Техніко-економічні характеристики ідеї	(Потенційні) товари/концепції конкурентів / Мій проект	W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
10.	Наявність програмного коду у відкритому доступі	Відсутня	-	+	-
11.	Легке створення об’єктів у БД	Присутнє	-	-	+
12.	Можливість інтеграції з іншими системами	Присутня	-	-	+

10.2 Технологічний аудит ідеї проекту

Таблиця 10.2 Технологічна здійсненність ідеї проекту

№ п/п	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1.	Реляційна БД	MSSQL Server	Наявні	Доступні
2.		Oracle	Наявні	Доступні
3.		MySql	Наявні	Доступні
Обрана технологія реалізації ідеї проекту: MSSQL Server.				
4.	Мова програмної реалізації	Компільована у машинний код (C, C++)	Наявні	Доступні
5.		Компільована у байткод (C#)	Наявні	Доступні
6.		Скриптова/інтерпретована (Python, JavaScript)	Наявні	Доступні
Обрана технологія реалізації ідеї проекту: Компільована у байткод (C#)				
7.	Операційна система	Windows	Наявні	Доступні
8.		Linux	Наявні	Доступні
9.		MacOS	Наявні	Доступна
Обрана технологія реалізації ідеї проекту: Windows				

10.3 Аналіз ринкових можливостей запуску стартап-проекту

Таблиця 10.3 Попередня характеристика потенційного ринку стартап-проекту

№ п/п	Показники стану ринку (найменування)	Характеристика
1.	Кількість головних гравців, од	0
2.	Загальний обсяг продаж, грн/ум.од	10000 грн/ум. од.
3.	Динаміка ринку (якісна оцінка)	Зростає
4.	Наявність обмежень для входу (вказати характер обмежень)	Недискримінаційні якісні
5.	Специфічні вимоги до стандартизації та сертифікації	Відсутні
6.	Середня норма рентабельності в галузі (або по ринку), %	65%

Таблиця 10.4 Характеристика потенційних клієнтів стартап-проекту

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1.	Автоматизація пошуку інформації	1. Середній бізнес 2. Великий бізнес	Різні об’єми та характеристики ресурсів	Підвищення продуктивності бізнесу. Зменшення часу на пошук інформації.
2.	Обробка результатів пошуку	1. Середній бізнес 2. Великий бізнес	Різна деталізація обробки інформації	Зменшення часу на перегляд знайденої інформації. Зручне подання інформації.
3.	Індексація ресурсів	1. Малий бізнес 2. Середній бізнес	-	Більш якісний пошук. Отримання більшої кількості результатів пошуку

Таблиця 10.5 Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1.	Блокування пошукових запитів	Блокування пошукових запитів провайдером пошуку через надмірну кількість пошукових запитів з однієї IP адреси	Балансування пошукових запитів. Використання резервних провайдерів пошуку (в тому числі власного).
2.	Отримання несанкціонованого доступу сторонніми особами	Хакерська атака що може призвести до зупинення роботи служби пошуку	Залучення спеціалістів з інформаційної безпеки. Використання засобів оркестрації.
3.	Відсутність ринку	Відсутність шляху збуту товару внаслідок помилкового орієнтування	Ретельний розгляд проблем потенційних клієнтів. Залучення експертів та менторів. Консультації із спеціалістами.
4.	Недостача капіталовкладень	Витрачені усі кошти до моменту виходу на ринок	Пошук нових джерел інвестицій.

Таблиця 10.6 Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1.	Отримання інвестицій	Отримання капіталу що необхідний для реалізації продукту	Розробка продукту
2.	Успішна маркетингова політика	В результаті проведеної маркетингової політики отримана висока зацікавленість користувачів	Підтримка стабільної роботи системи та проведення масштабування системи Збільшення цін на використання сервісу Використання подібної маркетингової стратегії надалі для залучення нових користувачів
3.	Поглинання конкурентами	Пропозиція купівлі проекту	Розвиток розроблених технологій Оцінка вартості розроблених технологій

Таблиця 10.7 Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
Олігополія	Незначна кількість конкурентів Велика ринкова сила Схожість використовуваних технологій	Інформування ринку щодо появи нової система автоматизованого пошуку
Галузевий	Загроза появи нових конкурентів Виркова влада споживачів Висока потреба у товарі	Інформування ринку щодо якості використовуваної новаторської технології Пропозиція гнучких цін
Внутрішньогалузева	Діяльність в одній галузі економіки Надання сервісів одного типу	Зменшення вартості сервісу Примноження каналів розподілу
Товарно-видова	Надання різних сервісів одного типу	Маркетингова політика
Цінова	Використання цін для покращення економічних умов збуту	Зменшення вартості системи Використання нових каналів розподілу
Марочна	Пропозиція схожої системи Спільна цільова аудиторія	Інформування ринку щодо появи нової система автоматизованого пошуку

10.4 Розроблення ринкової стратегії проекту

Таблиця 10.10 Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1.	Працівники підприємств по випуску лікарських засобів	Висока	65%	Середня	Низькі бар'єри входу
2.	Контролюючі органи (ДЕЦ)	Середня	78%	Середня	Низькі бар'єри входу
3.	Власники підприємств по випуску лікарських засобів	Низька	35%	Середня	Високі бар'єри входу
Які цільові групи обрано: працівники підприємств по випуску лікарських засобів, контролюючі органи (ДЕЦ)					

Таблиця 10.11 Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку*
1	Надання системи середньому та великому бізнесу	Вибірковий розподіл	Здатність протистояти прямим конкурентам Низькі витрати Ефективна співпраця	Стратегія диференціації

Таблиця 10.12 Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати	Чи буде компанія копіювати основні характеристики	Стратегія конкурентної поведінки*

		існуючих у конкурентів?	товару конкурента, і які?	
1	Так	Шукати нових	Ні	Стратегія лідера. Розширення первинного попиту

10.5 Розроблення маркетингової програми стартап-проекту

Таблиця 10.13 Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити
1	Пошук інформації про ліки	Отримує результати пошуку по заданим ключовим словам	Якість надання послуг Інноваційність технологій що використовуються Простота використання Цінова перевага
2	Автоматизація пошуку	Налаштувати автоматичне виконання сеансів пошуку	Якість надання послуг Інноваційність технологій що використовуються Простота використання Цінова перевага
3	Ведення обліку веб-ресурсів (видань) та їх матеріалів	Можливість стуктурувати дані та результати пошуку	Якість надання послуг Інноваційність технологій що використовуються Простота використання Цінова перевага

Висновки до розділу

В даному розділі було описано у вигляді табличних даних основну ідею стартапу системи автоматизованого пошуку. Було оцінено перспективи

реалізації ідеї проекту. Було виконано аналіз сильних сторін та слабких сторін ідеї. Було проведено SWOT аналіз проекту. Проект має великі перспективи на ринку, тому що на даний момент не існує прямих конкурентів.

ВИСНОВКИ

Під час виконання магістерської дисертаціїї було проаналізовано предметну область, яка відноситься до створення пошукових систем, досліджено існуючі рішення в галузі та визначего вимоги до системи автоматизованого пошуку інформації фармацевтичного характеру, було окреслено основні задачі системи.

Виходячи із сформованих вимог до функціоналу системи, визначених завдань та з використанням сучасного стеку технологій .NET Framework було створено клієнт-серверну систему, в якій клієнтською частиною виступає система фармацевтичного нагляду, а серверною частиною виступає електронна бібліотека. На стороні клієнта встановлюється компонент планування виконання пошуковиї запитів та приймач результатів пошуку, а для управління використовується веб-модуль, який побудовано з використанням трирівневої архітектури. На стороні електронної бібліотеки встановлюється служба безпосереднього пошуку, служба фонових браузерів, служба виконання скриптів обробки результатів пошуку, компонент завантаження результатів пошуку у вигляді публікацій в електронну бібліотеку, служба отримання публікацій та компонент клієнтської підписки на видання бібліотеки.

Система виконує всі поставленні на початку виконання роботи завдання, а саме виконання автоматизованого та миттєвого пошуку інформації про певний лікарський засіб на визначеному веб-джерелі. Для запуску пошуку користувачі повинні створити пакети моніторингу, які складаються з лікарських засобів та видань, та на основі цих пакетів створити план-графіки виконання пошуків. Після запуску план-графіків користувачі отримають необхідні результати моніторингу.

Під час реалізації системи проводилося регулярне тестування всіх модулів, що у підсумку підтвердило цілодобову роботу та стійкість до навантаження системи.

Отже, з вище сказаного виходить, що система реалізована в повному обсязі та виконані всі поставлені задачі на магістерську дисертацію в повному

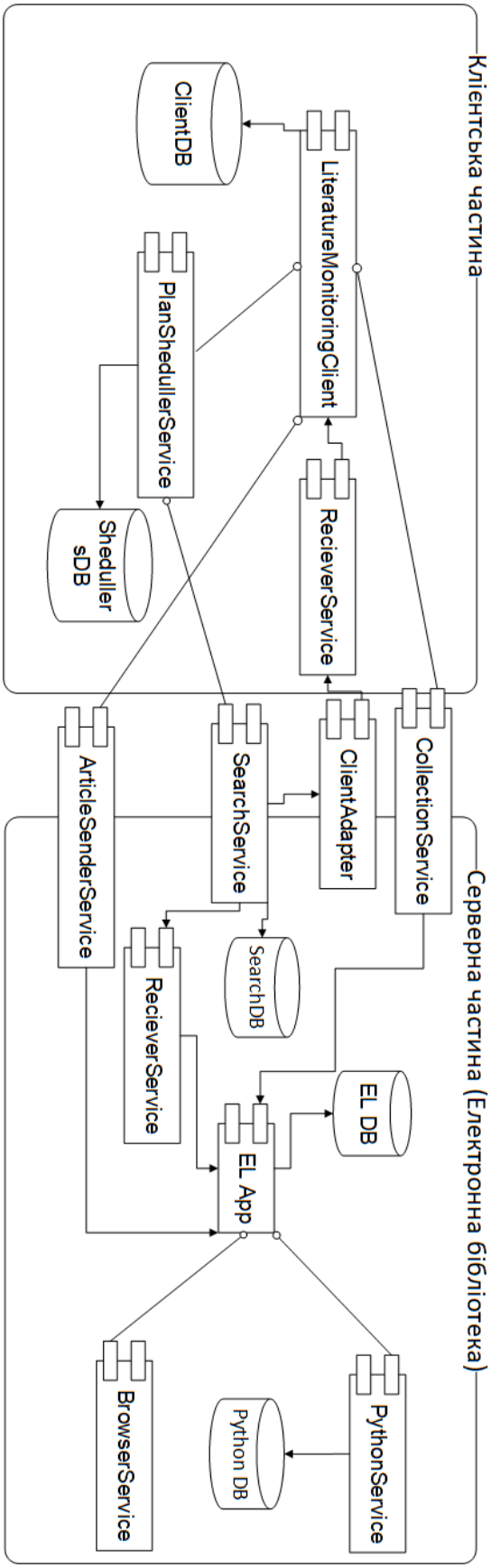
обсязі. Результати магістерської дисертації апробовано та впроваджено у комерційне виробництво ТОВ «ІРІС». Акт впровадження додається.

ПЕРЕЛІК ПОСИЛАНЬ

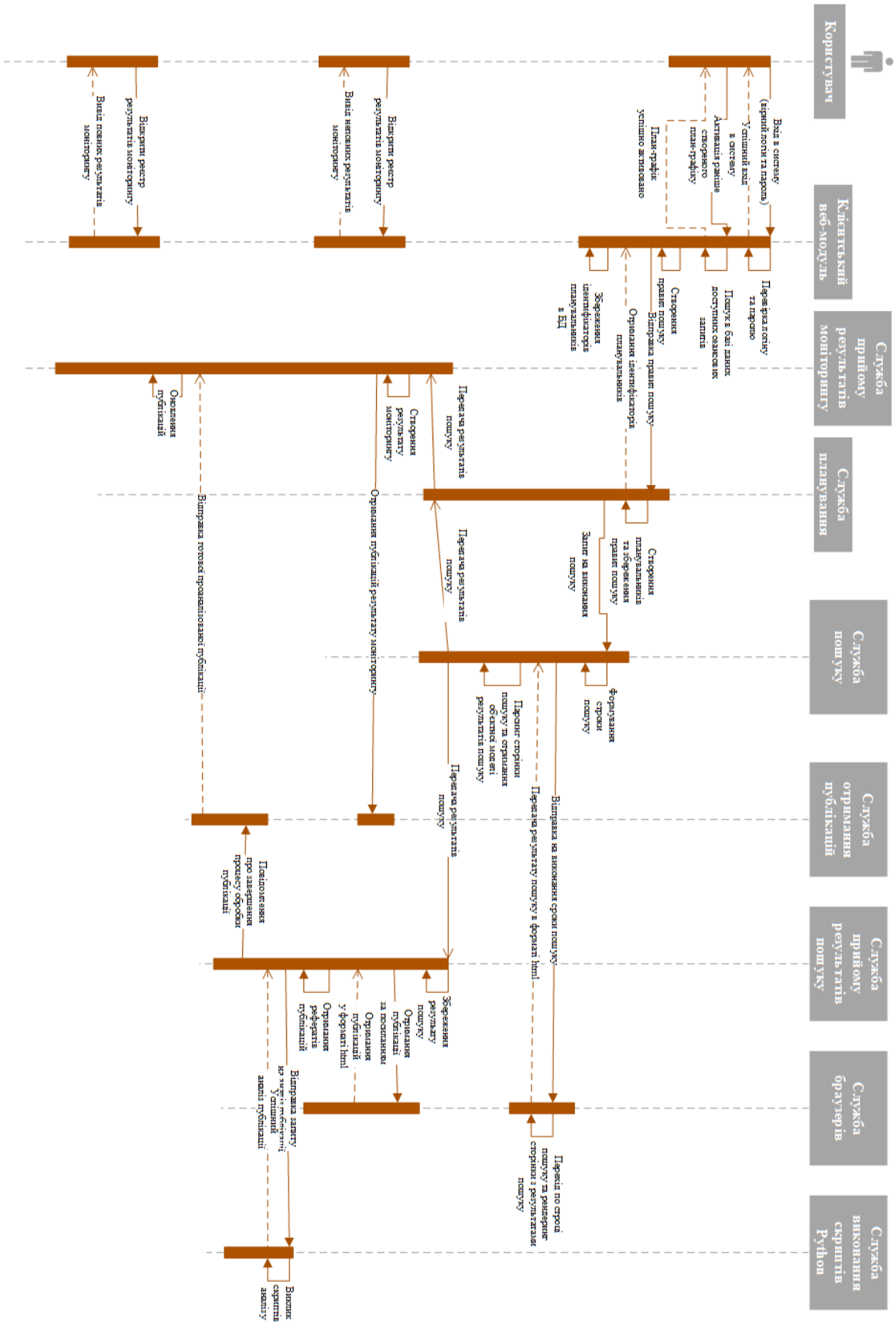
1. Офіційний сайт державного експертного центру України [Електронний ресурс] – Режим доступу до ресурсу: http://www.dec.gov.ua.
2. Segev E. Google and the Digital Divide: The Bias of Online Knowledge / Elad Segev. – Oxford: Chandos Publishing, 2010. – 221 с.
3. Ашманов И. С. Продвижение сайта в поисковых системах / И. С. Ашманов, А. А. Иванов. – М.: Вильямс, 2007. – 304 с.
4. Эволюция алгоритмов поисковых систем [Електронний ресурс] – Режим доступу до ресурсу: https://habrahabr.ru/company/altweb/blog/231531.
5. Дорнфест Р. Секреты Google. Трюки и тонкая настройка / Р. Дорнфест, Р. Бош, Т. Калишейн. – М.: Русская редакция, 2008. – 510 с.
6. Офіційна сторінка фреймворку Hangfire [Електронний ресурс] – Режим доступу до ресурсу: https://www.hangfire.io/overview.html.
7. Троелсен Е. Язык программирования C# 5.0 и платформа .NET 4.5 / Ендрю Троелсен. – М.: Вильямс, 2015. – 1312 с.
8. Рихтер Д. CLR via C#. Программирование на платформе Microsoft.NET Framework 4.5 на языке C# / Д. Рихтер. – П.: Питер, 2017. – 896 с.
9. Скит Д. C# для профессионалов. Тонкости программирования / Джон Скит. – М.: Вильямс, 2017. – 608 с.
10. Ицик Б. Microsoft SQL Server 2012. Основы T-SQL / Бен-Ган Ицик. – М.: Эксмо, 2015. – 400 с.
11. Адамс К. Администрирование сервера IIS 7 / Крис Адамс. – М.: Бином-Пресс, 2013. – 362 с.
12. Фримен А. ASP.NET MVC 4 с примерами на C# 5.0 для профессионалов / Адам Фримен. – М.: Вильямс, 2013. – 688 с.
13. Фримен А. LINQ. Язык интегрированных запросов в C# 2010 для профессионалов / А. Фримен, Д. Раттц. – М.: Вильямс, 2011. – 656 с.
14. WCF 4: Windows Communication Foundation и .NET 4 для профессионалов / П.Сибраро, К. Клайс, Ф. Коссолино, Й. Грабнер. – М.: Вильямс, 2011. – 464 с.

15. Симан М. Внедрение зависимостей в .NET / Марк Симан. – С.-П.: Питер, 2013. – 464 с.
16. ДСТУ 2392-94.
17. Інформаційні системи і технології. Методичні вказівки щодо виконання дипломного проекту для студентів спеціальностей 1-40 01 02 «Інформаційні системи та технології» / уклад .: О.І. Наранович. - Барановичі: РІО Барген, 2009. - 39 с.
18. Основи розробки веб-додатків. Навчальний посібник / В.В. Осадчий, В.С. Круглик – Мелітополь: ТОВ «Видавничий будинок ММД», 2012. – 540 с.
19. Пріоритетні наукові напрями та найважливіші проблеми: від теорії до практики. Матеріали міжнародної науково-практичної конференціїї. – Одеса: ГО «Інститут інноваційної освіти», 2017. – 164 с.
20. Что такое Docker и технология контейнеров Linux [Електронний ресурс] – Режим доступу до ресурсу: https://vps.ua/blog/docker-and-linux-containers.

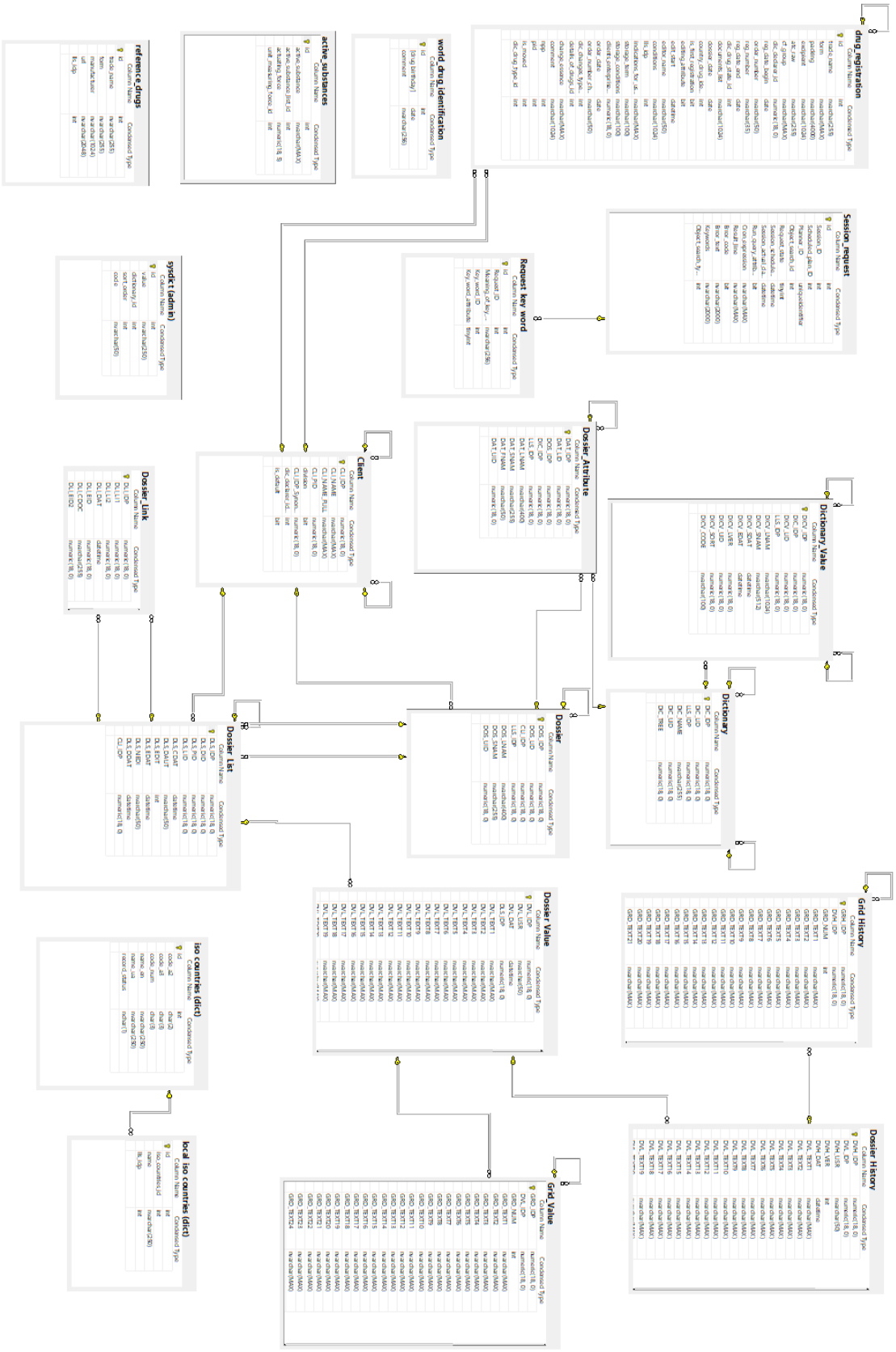
ДОДАТОК А. СТРУКТУРНА СХЕМА СИСТЕМИ



ДОДАТОК Б. ФУНКЦІОНАЛЬНА СХЕМА СИСТЕМИ

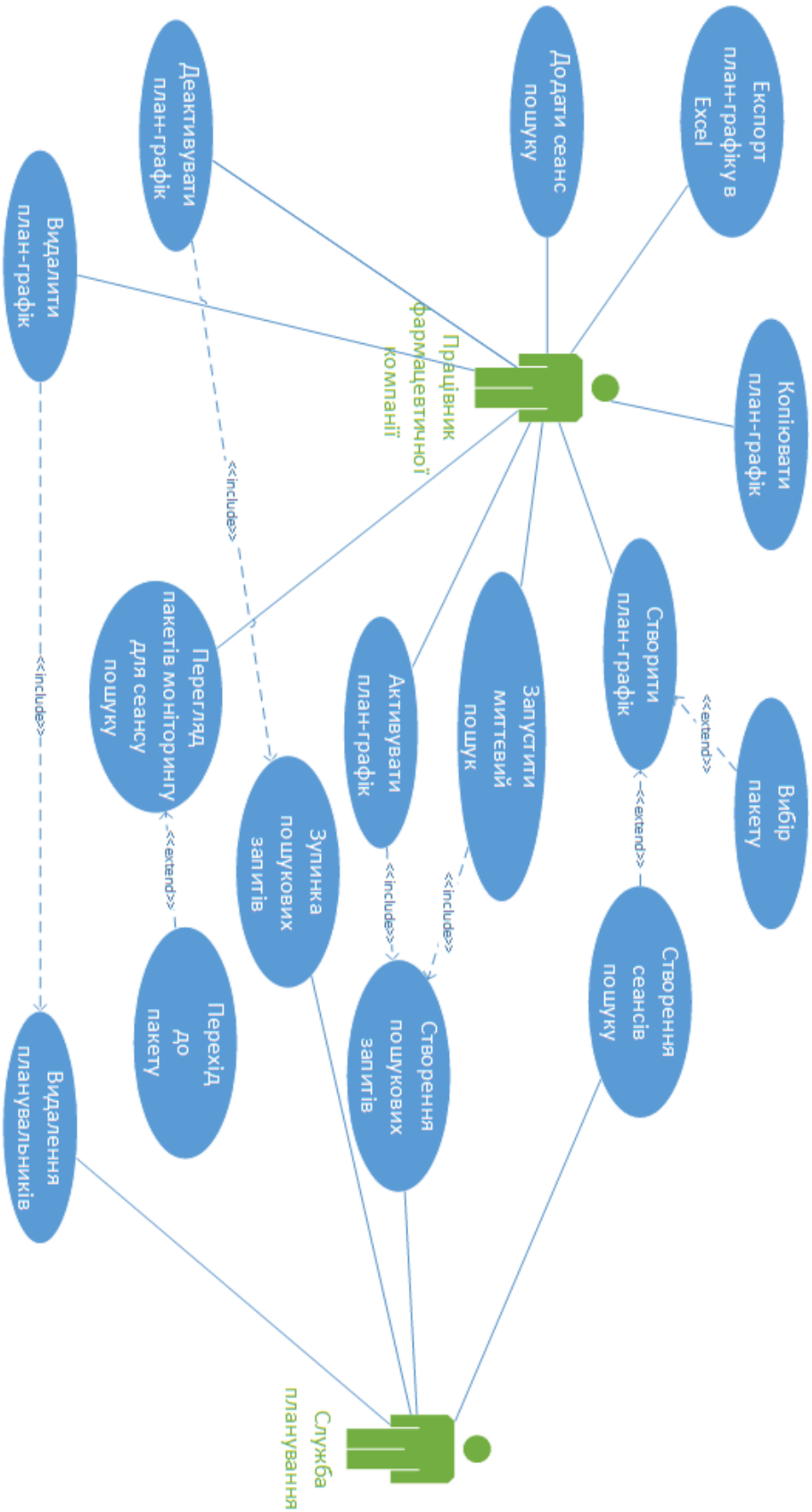


ДОДАТОК В. ДАТАЛОГІЧНА МОДЕЛЬ БД

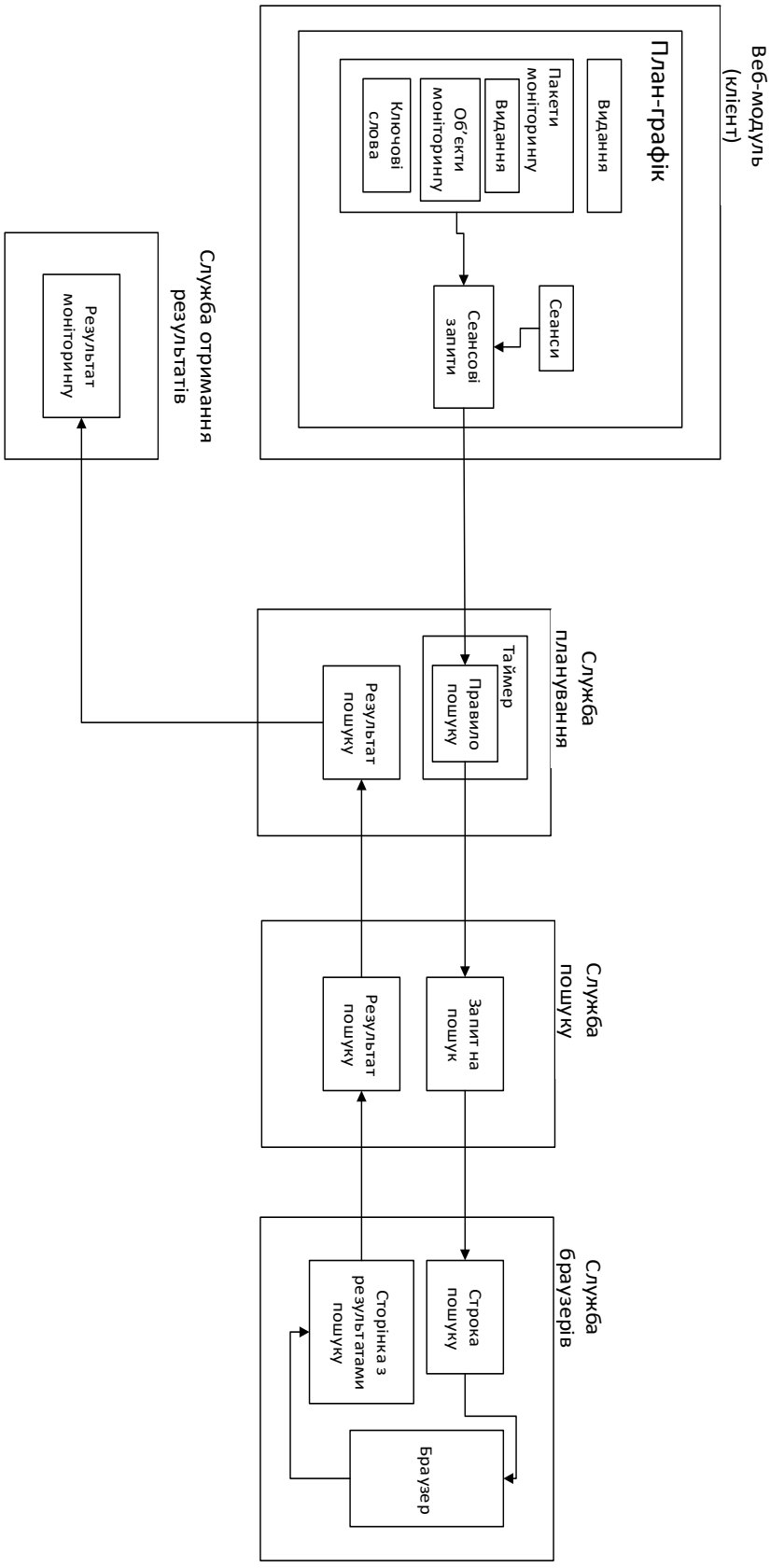


ДОДАТОК Г. ДІАГРАМА КЛАСІВ

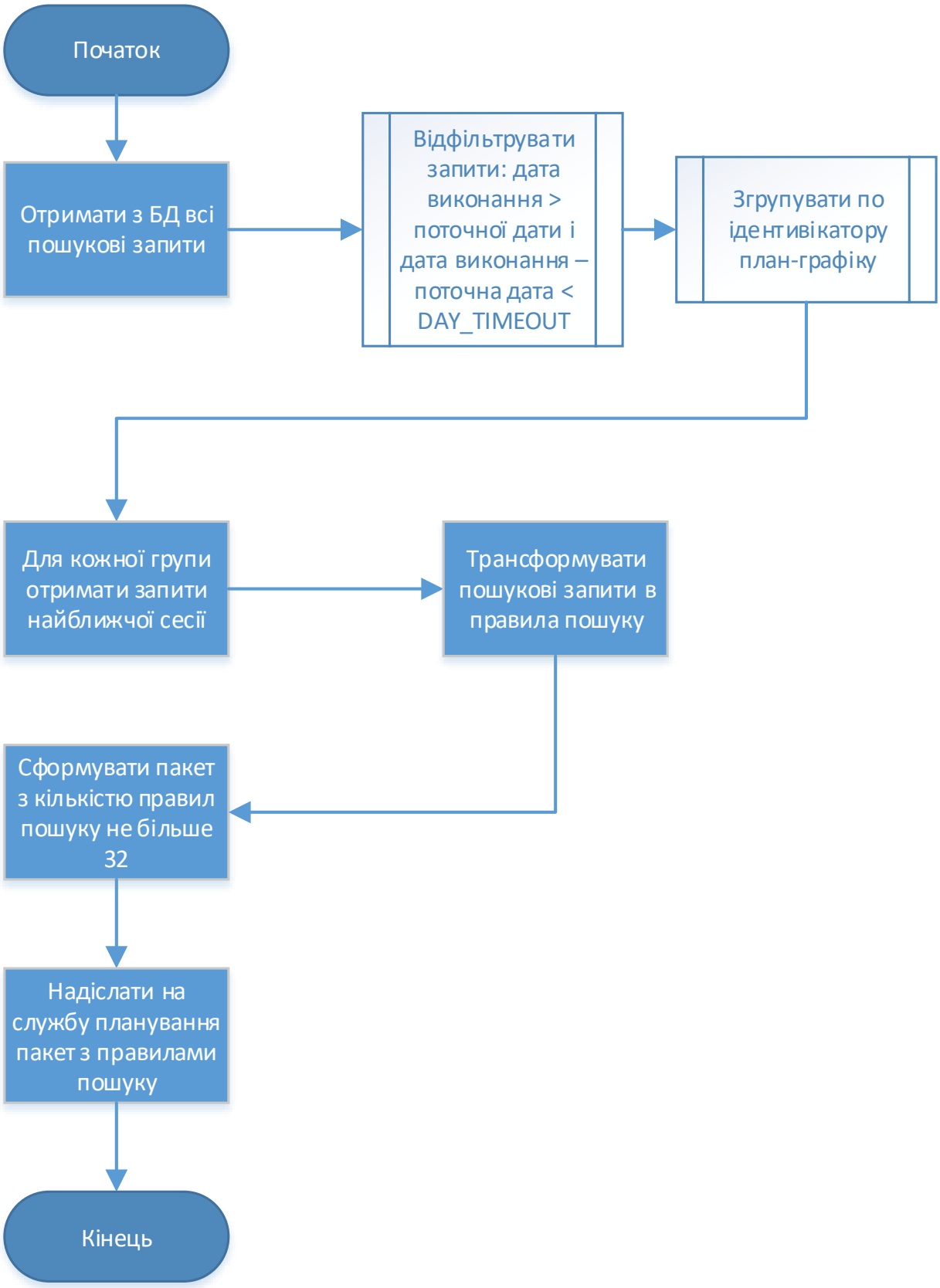
ДОДАТОК Д. ДІАГРАМА ПРЕЦЕДЕНТІВ СТОРІНКИ ПЛАН-ГРАФІКІВ



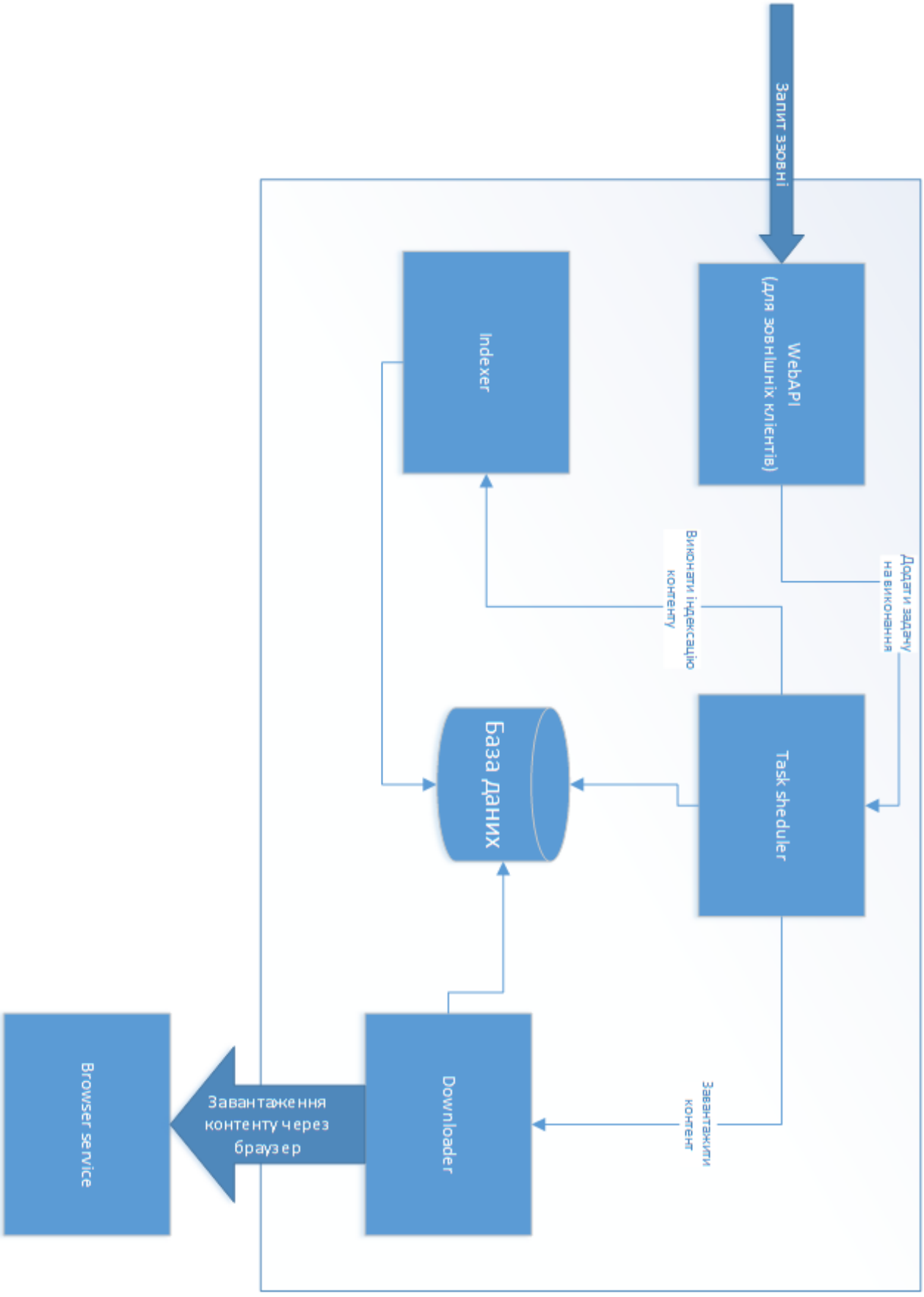
ДОДАТОК Е. ДІАГРАМА ПЕРЕТВОРЕННЯ ДАНИХ



ДОДАТОК Є. СХЕМА АЛГОРИТМУ АВТОМАТИЧНОГО ПОШУКУ



ДОДАТОК Ж. ДІАГРАМА КОМПОНЕНТІВ СЛУЖБИ ІНДЕКСАЦІЇ



ДОДАТОК З. АКТ ВПРОВАДЖЕННЯ

ІРІС
Інститут розробки інформаційних систем

Україна, 03055, м. Київ, пр. Перемоги, 29
Код ЄДРПОУ 39682689
телефон 380 (44) 592-40-33
www.iisd.com.ua, iisd@iisd.com.ua

Вих. № 71 від 03.12.2018 р.
На № _____ від _____ р.

АКТ
про впровадження результатів дипломного проекту

ТОВ «Інститут розробки інформаційних систем» цим Актом про впровадження результатів дипломного проекту підтверджує, що результати магістерської дисертації студента НТУУ «КПІ ім. І. Сікорського» Факультету інформатики та обчислювальної техніки групи ІТ-73мп Корнілова Івана Станіславовича на тему «Система автоматизованого пошуку інформації фармацевтичного характеру» впроваджено до ліцензійного програмного продукту «База даних з фармаконагляду та безпеки лікарських засобів DSBase».

Програмний продукт DSBase є комп'ютеризованою інформаційно-аналітичною системою, призначеною для реалізації правил належної практики з фармаконагляду, автоматизації процесів ведення реєстраційної інформації та здійснення моніторингу безпеки лікарських засобів фармацевтичної компанії Заявника (Держателя реєстраційних посвідчень), а також підтримки формування звітності, що стосується фармаконагляду. Впровадження результатів дипломного проекту дозволило розширити функціонал існуючої системи щодо моніторингу літературних джерел для пошуку інформації про лікарські засоби.

Генеральний директор А.В.Чадюк