Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____Комп'ютерних наук_____
(повна назва)

Кафедра_____Інформаційних управляючих систем_____
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти _____другий (магістерський)_____

Виявлення термінів проєкту та їх артефактів, пред'явлених у вимогах до проєкту, за допомогою обробки природної мови
(тема)

Виконав:
студент _2_ курсу, групи _УПГІТм-22-3___
_Лучицький В'ячеслав Володимирович____
(прізвище, ім'я, по батькові)
Спеціальність ____122 Комп'ютерні_____
науки_____
(код і повна назва спеціальності)
Тип програми __освітньо-наукова_____
(освітньо-професійна або освітньо-наукова)
Освітня програма _Управління проектами в галузі інформаційних технологій_____
(повна назва освітньої програми)

Керівник __проф. Максим ЄВЛАНОВ_____
(посада, власне ім'я, прізвище)

Допускається до захисту

Зав. кафедри _____ _Костянтин ПЕТРОВ__
(підпис) (власне ім'я, прізвище)

2024 р.

Харківський національний університет радіоелектроніки

Факультет _____Комп'ютерних наук_____

Кафедра _____Інформаційних управляючих систем_____

Рівень вищої освіти _____другий (магістерський)_____

Спеціальність _____122 Комп'ютерні науки_____
(код і повна назва)

Тип програми _____освітньо-наукова_____
(освітньо-професійна або освітньо-наукова)

Освітня програма_Управління проектами в галузі інформаційних технологій
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_01__»__квітня__ 20 24_ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові _Лучицькому В'ячеславу Володимировичу_____
(прізвище, ім'я, по батькові)

1. Тема роботи _Виявлення термінів проєкту та їх артефактів, пред'явлених у вимогах до проєкту, за допомогою обробки природної мови._____
_____
затверджена наказом університету від _01 квітня 2024_ р. № _258 Cm_
2. Термін подання студентом роботи до екзаменаційної комісії __01__ ___06_____ 2024 р.
3. Вихідні дані до роботи процес групування відмінюваних форм, процес визначення системних вимог, наукова література, публікації та інтернет - джерела з досліджуваної проблеми, матеріали передатестаційна практики._____
_____
_____
_____
_____
_____
4. Перелік питань, що потрібно опрацювати в роботі концептуальні засади аналізу текстових даних у контексті управління проєктами, адаптація процесу групування відмінюваних форм для аналізу текстових даних в вимогах до проєкту, реалізація процесу аналізу текстових даних в вимогах до проєкту, програмна реалізація та апробація сервісу обробки вимог до системи._____
_____
_____
_____

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Аналіз процесів управління IT проєктом	2.04.2024	Виконано
2	Огляд сучасних підходів до аналізу	3.04.2024-5.04.2024	Виконано
3	Опис процесу інтегрованого аналізу	6.04.2024-8.04.2024	Виконано
4	Формалізація елементів процесу	9.04.2024-11.04.2024	Виконано
5	Побудова архітектури системи	12.04.2024-16.04.2024	Виконано
6	Опис технологічного стеку	17.04.2024-19.04.2024	Виконано
7	Реалізація системи та її компонентів	20.04.2024-25.04.2024	Виконано
8	Апробація і опис результатів	26.04.2024-29.04.2024	Виконано
9	Оформлення пояснювальної записки	30.04.2024-4.05.2024	Виконано
10	Захист кваліфікаційної роботи	04.06.2024	Виконано

Дата видачі завдання __01__ ___квітня__2024_ р.

Студент _____
(підпис)

Керівник роботи _____ ___проф. Максим ЄВЛАНОВ_
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка до магістерської кваліфікаційної роботи: 79 сторінок, 4 розділи, 25 рисунків, 5 таблиць, 25 джерел, 2 додатка.

АВТОМАТИЗАЦІЯ, АНАЛІЗ ТЕКСТОВИХ ДАНИХ, ВИМОГИ, ВИЯВЛЕННЯ ТЕРМІНІВ, ДІАГРАМИ, ЛЕМАТИЗАЦІЯ, ОБРОБКА ТЕКСТОВИХ ДАНИХ.

Об'єктом дослідження є процес виявлення та аналізу системних вимог у контексті управління ІТ-проектами. Мета роботи полягає в розробці інформаційної технології для виявлення та класифікації термінів проєкту з використанням обробки природної мови для підвищення ефективності формування та аналізу вимог. Дослідження базується на процесі лематизації що входить до завдань обробки природньої мови.

Результати роботи можуть бути використані в розробці ІТ-проектів, де важлива точність та об'єктивність аналізу вимог. Важливим є також розгляд можливості інтеграції розробленого сервісу виявлення термінів в популярні продукти для управління ІТ-проєктами.

Рекомендується подальше дослідження напрямків вдосконалення виявлення термінів і артефактів з інформаційних систем. Особливу увагу слід приділити розробці підходів, які дозволяють враховувати контекст термінів у вимогах, забезпечуючи їх точне визначення в системі.

ABSTRACT

The explanatory note to the qualification work: 79 pages, 4 chapters, 25 figures, 5 tables, 25 sources, 2 appendices.

AUTOMATION, DIAGRAMS, LEMMATIZATION, REQUIREMENTS, TERM DETECTION, TEXT DATA ANALYSIS, TEXT PROCESSING.

The object of the research is the process of detecting and analyzing system requirements in the context of IT project management. The aim of the work is to develop an information technology for the detection and classification of project terms using natural language processing to enhance the efficiency of requirements formation and analysis. The research is based on the process of lemmatization, which is a task in natural language processing.

The results of the work can be used in the development of IT projects where the accuracy and objectivity of requirements analysis are important. It is also crucial to consider the possibility of integrating the developed term detection service into popular IT project management products.

Further research is recommended in the areas of improving the detection of terms and artifacts from information systems. Special attention should be given to developing methods that consider the context of terms in requirements, ensuring their accurate definition in the system.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ЖЦ – життєвий цикл;

ІС – інформаційні системи;

ІТ – інформаційні технології;

ПЗ – програмне забезпечення;

УП – управління проєктами;

ШІ – штучний інтелект;

AI – Artificial intelligence;

API – Application Programming Interface;

IBM – International Business Machines Corporation;

MS – Microsoft;

POS – Part of Speech;

NLP – Natural Language Processing;

UML – Unified Modeling Language.

ВСТУП

Сучасний стан інформаційних технологій та комп’ютерних наук характеризується стрімким розвитком методів обробки природної мови (NLP), які знаходять застосування в різних галузях, включаючи управління проєктами та аналіз вимог до програмного забезпечення. Прогрес у цій області сприяє ефективнішому збору, аналізу та використанню даних, що становить основу для подальших технічних інновацій.

Світові тенденції в NLP спрямовані на підвищення точності та ефективності аналітичних систем через застосування машинного навчання та глибокого навчання. Це дозволяє автоматизувати процеси, які традиційно вимагали значних людських ресурсів, і забезпечити більш глибоке і точне розуміння текстових даних.

Актуальність роботи визначається високим попитом на ефективні інструменти для виявлення та аналізу системних вимог в управлінні IT-проектами. Сучасний процес формування вимог часто стикається з проблемами, такими як упередженість на основі досвіду та неоднозначність у визначенні вимог, що призводить до ризиків і помилок у проектуванні. Ці виклики підсилюють необхідність розробки більш точних та автоматизованих методів для обробки і аналізу вимог.

Необхідність таких досліджень також обумовлена відсутністю комплексних рішень, які б інтегрували глибоке розуміння текстового контенту з використанням обробки природної мови для системного та об'єктивного аналізу вимог. Застосування методів NLP може значно підвищити точність виявлення ключових термінів та їхніх взаємозв'язків, зменшити упередженість і підвищити загальну надійність процесів управління проектами.

Таким чином, ця робота є актуальною, оскільки вона спрямована на розвиток новітніх технічних рішень, що допомагають усунути існуючі

прогалини в аналізі та обробці системних вимог, тим самим сприяючи підвищенню ефективності і скороченню часу розробки IT-проєктів

Мета даної кваліфікаційної роботи полягає у дослідженні особливостей застосування процесу групування відмінюваних форм при формуванні системних вимог. Робота зосереджена на розробці інформаційної технології для аналізу текстових вимог до IT-проєктів, використовуючи техніки NLP для ефективного виявлення, класифікації та візуалізації проектних термінів та їх зв'язків.

Предметна галузь дослідження охоплює групування відмінюваних форм і особливості його застосування для вирішення формування системних вимог. Робота включає аналіз існуючих підходів та розробку нових підходів для підвищення точності та ефективності процесів аналізу та специфікації вимог у контексті сучасних IT-проєктів.

Для досягнення поставленої мети визначено наступні задачі:

– аналіз сучасних методів NLP, зокрема групування відмінюваних форм, та їх застосування у сфері управління IT-проектами;

– проектування архітектури та визначення технологічного стеку для розробки інформаційної технології;

– розробка механізму зіставлення виявлених сутностей з існуючими елементами системи;

– розробка інформаційної технології для виявлення термінів проєкту з текстових вимог, з акцентом на групування відмінюваних форм;

– апробація розробленої системи на реальних даних.

Пояснювальна записка оформлена згідно методичних вказівок [1] та ДСТУ 3008:2015 [2].

1 КОНЦЕПТУАЛЬНІ ЗАСАДИ АНАЛІЗУ ТЕКСТОВИХ ДАНИХ У КОНТЕКСТІ УПРАВЛІННЯ ПРОЄКТАМИ

Аналіз процесів управління ІТ проєктом

Управління ІТ проєктами в сучасному динамічному технологічному середовищі є складним завданням, яке вимагає системного та структурованого підходу для досягнення успішних результатів. Для ефективного керування такими проєктами важливо мати чітке розуміння процесів, що складають життєвий цикл проєкту. Ці процеси визначені в стандарті ДСТУ ISO/IEC/IEEE , який встановлює уніфікований набір процесів управління проєктами в області ІТ та описує їх основні діяльності і завдання. Цей стандарт є важливим інструментом оскільки він надає структуроване та системне відображення процесів, які необхідно враховувати на кожному етапі проєкту.

Одним із основних процесів управління ІТ проєктами є "Процес визначення вимог до системи", який відіграє ключову роль у розробці успішних інформаційних технологічних рішень. Цей процес спрямований на систематичний збір та аналіз вимог до системи від різних зацікавлених сторін з метою точного визначення функціональних і нефункціональних характеристик, які система повинна втілювати [3]. Візуально процес представлено на рисунку 1.1.

Діяльність «Визначення системних вимог» є центральною у цьому процесі. Під час цієї діяльності проводиться аналіз потреб користувачів та зацікавлених сторін щодо функціональності, продуктивності, безпеки та інших аспектів системи. Результатом завданням діяльності є «Визначення та обґрунтування системних вимог» яке охоплює визначення вимог до системи відповідно до вимог замовлених сторін, функціональних меж, функцій, обмежень, цільових показників вартості, визначених інтерфейсів і критичних характеристик якості.

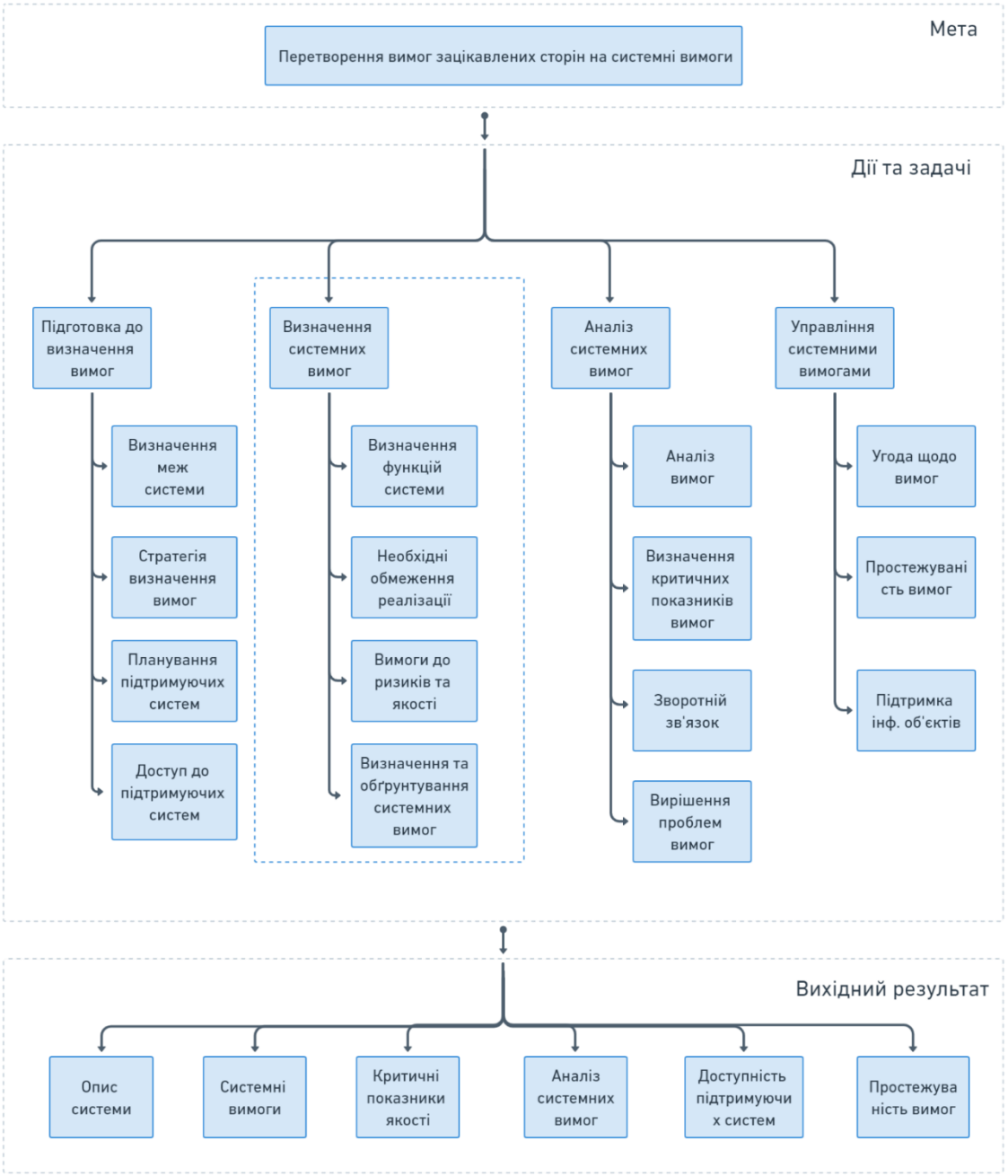

Рисунок 1.1 – Діаграма "Процес визначення вимог до системи"

Вимоги до системи фіксуються у формі, придатній для управління вимогами протягом усього ЖЦ. Ці записи встановлюють базові вимоги до системи та включають відповідне обґрунтування, рішення та припущення. Вони є основою для відстеження інформаційних елементів і наступних елементів системи. Запити на зміну вимог до системи також забезпечують логічне обґрунтування, щоб допомогти у визначенні прийнятності

запропонованої зміни, охоплюючи узгодженість із вимогами зацікавлених сторін [3].

Потрібно відзначити, що практика виконання проєктів, зокрема ІТ-проєктів, підтверджує, що внесення змін до ІС на ранніх етапах їхнього життєвого циклу є найбільш ефективним та економічно доцільним підходом. Це означає, що вирішення проблем та удосконалення системи в самому початковому етапі її розвитку вимагає менших витрат часу, грошей та ресурсів порівняно з такими ж діями на пізніших етапах проєкту [4]. Такий підхід свідчить про важливість ретельного аналізу визначення вимог до системи і важливість врахування потреб користувачів, технічні обмеження, а також вимоги щодо забезпечення безпеки, надійності та інших якостей системи. Для цього можуть використовуватися різноманітні методи, такі як інтерв'ю з користувачами, аналіз існуючих систем.

Крім того, вирішення цих проблем на ранньому етапі є одним з перспективних напрямків науково-прикладних досліджень в галузі інформаційних технологій, що вказує на актуальність і значущість даної проблематики для подальшого розвитку сфери ІТ.

Для успішного вирішення проблеми внесення змін до інформаційних систем на ранніх етапах їхнього ЖЦ є розуміння технічних вимог та складнощів архітектурного проєкту на попередньому досвіді який дозволяє заздалегідь передбачити можливі складнощі та проблеми, які можуть виникнути під час реалізації або внесення змін до системи.

Одним із способів, яким попередній досвід може допомогти у вирішенні цієї проблеми, є аналіз архітектурних особливостей існуючої системи та її зв'язку з іншими компонентами на етапі визначення вимог до проєкту. З ретельним вивченням архітектури системи можна ідентифікувати області, які можуть бути найбільш вразливими до змін, а також визначити потенційні ризики та виклики, пов'язані з цими змінами.

Додатково, попередній досвід також дозволяє оцінити вплив запропонованих змін на вже існуючі компоненти системи та їх функціональні

можливості. Це допомагає уникнути неочікуваних негативних наслідків та забезпечити сумісність нових функцій з вже існуючими.

Попередній досвід, хоча і є важливим джерелом знань, може також викликати упередженість, що може негативно впливати на процес прийняття рішень та розвиток систем. У статті [5] показано, що більшість випадків негативного впливу локальних рішень на загальний дизайн і якість великої програмно-технічної системи усувалося шляхом раннього поділу системи на окремі елементи. Однак, зворотні випадки виникали, головним чином, внаслідок неправильного тлумачення вимог чи упередженості особистого досвіду [5].

Це дозволяє зробити висновок про доцільність використання для ідентифікації конфігурації великих систем людино-машинних чи машинних методів, у яких суб'єктивний вплив окремого аналітика зведено до мінімуму. Такий підхід допомагає уникнути упередженості особистого досвіду та забезпечити об'єктивне та систематичне аналізу складнощів та потенційних проблем в процесі розробки та управління великими системами.

1.2 Огляд сучасних підходів до аналізу текстових даних

Машинний метод, такий як обробка природної мови(NLP), може значно допомогти у зменшенні впливу упередженості особистого досвіду на процес прийняття рішень та розвиток систем. NLP використовується для аналізу, розуміння та генерації природної мови комп'ютерними системами, це може допомогти у виявленні та аналізі вимог користувачів, коментарів, відгуків та інших текстових джерел, що стосуються розробки та управління великими системами. Зокрема, вона може автоматизувати процес аналізу тексту для виявлення ключових сутностей, патернів або проблем, які можуть впливати на розвиток системи. Крім того, NLP може допомогти у створенні об'єктивних

критеріїв та метрик для оцінки якості та ефективності системи, що дозволяє уникнути суб'єктивності та упередженості в процесі оцінки.

Методи обробки природної мови охоплюють широкий спектр алгоритмів і підходів, від базового синтаксичного аналізу до складних моделей глибокого навчання, що здатні вловлювати семантичні зв'язки у тексті. Технології NLP дозволяють автоматизувати процес розпізнавання та класифікації термінів із текстових даних, що може значно підвищити ефективність управління проєктами. Серед доступних методів для класифікації термінів як ключових сутностей і використання їх для подальшого аналізу є методи стемінга та лематизації.

Стемінг є процесом видалення афіксів зі слів, щоб повернутися до кореня або базової форми слова [6]. Цей метод може бути корисним для спрощення текстових даних, але він може призвести до втрати значущої інформації через недоліки у врахуванні словоформ.

Лематизація, у свою чергу, є більш складним процесом, що залучає морфологічний аналіз для перетворення слова на його базову форму або лему. Це процес групування разом різних словоформ одного слова, так щоб їх можна було аналізувати як єдиний елемент, який можна ідентифікувати за лемою слова або його словниковою формою [6].

Далі наведемо аналіз процесу стемінга, що дозволяє розглянути його переваги та недоліки, а також виявити можливості та загрози, які він може мати в контексті обробки текстових вимог. На підставі цього аналізу було створено SWOT-аналіз процесу стемінга (див. табл. 1).

Результат аналізу методом стемінга та основні аспекти:

а) сильні сторони: стемінг працює швидше, ніж багато інших методів обробки тексту, оскільки він заснований на простих алгоритмах відсікання закінчень слів.

б) слабкі сторони:

1) стемінг може призводити до відсікання закінчень, які змінюють семантику слова, що може спотворити значення тексту та знизити точність виявлення сутностей;

більшість алгоритмів стемінга розроблено для англійської мови які можуть не працювати ефективно з іншими мовами, особливо з тими, що мають складну морфологію.

в) можливості:

1) розвиток та оптимізація алгоритмів стемінга для різних мов може розширити їх застосування та покращити точність виявлення сутностей;

2) комбінування стемінга з іншими методами NLP, може п

о) загрози:

розвиток технологій глибокого навчання може зробити стемінг менш релевантним для деяких задач NLP, де важлива висока точність;

збільшення вимог до точності в обробці природної мови може обмежити застосування стемінга, особливо в задачах, де критично важливо розуміння контексту та збереження семантики тексту;

3) неструктуровані дані, такі як текст з соціальних мереж, блогів або форумів, можуть містити велику кількість сленгу, скорочень та інших нестандартних форм вираження. Стемінг може не ефективно впоратися з такими даними, оскільки він не розроблений для глибокого розуміння контексту або різноманітності мовленнєвих форм.

г

а

л

ь

н

у

т

о

Таблиця 1 – SWOT-аналіз процесу стемінга

	Корисні	Шкідливі
Внутрішні	Сильні	Слабкі
	• Швидкість обробки	• Неточність • Мовна залежність
Зовнішні	Можливості	Загрози
	• Розвиток алгоритмів • Інтеграція з іншими методами	• Зміни в обробці природної мови • Зростання вимог до якості обробки тексту • Обмеження в обробці неструктурованих даних

Аналогічно наведемо аналіз лематизації, що дозволяє розглянути її переваги та недоліки, результат який зведено у таблицю 2.

Результат аналізу процесу лематизації та основні аспекти:

а) сильні сторони:

1) лематизація дозволяє точно визначати базову форму слова, що сприяє більш точному виявленню сутностей;

відрізняючись від стемінга, лематизація враховує контекст слова, що дозволяє зберегти його семантичне значення;

3) метод ефективно працює з різними частинами мови, що робить його універсальним інструментом для аналізу тексту;

4) лематизація підтримується численними алгоритмами, які ефективно працюють з різними мовами. Це забезпечує гнучкість у виборі інструментів для обробки специфічних мовних даних, а також робить можливим точне виявлення сутностей для різних лінгвістичних контекстів.

б) слабкі сторони:

лематизація вимагає більше обчислювальних ресурсів порівняно зі стемінгом через необхідність аналізу контексту слова;

2) для ефективної лематизації необхідна складна логіка обробки та великі лінгвістичні бази даних;

3) ефективність методу може залежати від конкретної мови тексту, особливо для мов із складною морфологією.

в) можливості:

1) лематизація може бути інтегрована з іншими методами обробки природної мови для покращення загальної точності виявлення сутностей;

точне виявлення сутностей може покращити тренування моделей машинного навчання, зокрема в задачах класифікації тексту та розуміння природної мови;

3) лематизація може бути застосована у широкому спектрі областей, від семантичного пошуку до автоматичного резюмування тексту.

г) загрози:

1) швидкий розвиток галузі NLP може призвести до того, що лематизація стане застарілою або недостатньо ефективною порівняно з новими методами;

в контексті неструктурованих або сленгових текстів лематизація може зіткнутися з труднощами, що знижує її ефективність.

Таблиця 2 – SWOT-аналіз процесу лематизації

	Корисні	Шкідливі
Внутрішні	Сильні	Слабкі
	• Точність • Контекстна обробка • Універсальність • Підтримка алгоритмами різноманітних мов	• Ресурсоємність • Складність реалізації • Залежність від мови
Зовнішні	Можливості	Загрози
	• Інтеграція з іншими NLP інструментами • Покращення машинного навчання • Розширення застосування	• Зміни в обробці природної мови • Обмеження в обробці неструктурованих даних

Для подальшого дослідження, на основі проведеного SWOT-аналізу, лематизація вибирається як метод виявлення сутностей у тексті через її здатність забезпечувати високу точність аналізу. Цей метод враховує контекст слова, що дозволяє точно ідентифікувати базові форми слів, ця характеристика робить лематизацію особливо цінною для задач, де важлива точність розпізнавання сутностей. Хоча лематизація може вимагати більше обчислювальних ресурсів порівняно зі стемінгом та пред'являти вищі вимоги до реалізації, багато сучасних алгоритмів підтримують українську мову, що свідчить про її придатність у широкому спектрі лінгвістичних середовищ.

1.3 Значення обробки природної мови в управлінні проєктами

Революційні підходи в інженерії вимог, зосереджуючись на використанні NLP та генеративного ШІ для підвищення ефективності розробки програмного забезпечення. Як зазначається у статі [7] в середньому 60% помилок у проєктах спричинені уникненнями помилками на етапах визначення та аналізу вимог, що здебільшого виникають через неоднозначності чи плутанину в критеріях прийняття. У контексті існуючих рішень демонструються різні інструменти та методології, які використовують NLP для підвищення якості специфікацій вимог, зменшення помилок і оптимізації робочого процесу розробки. Крім того, розглядається потенціал генеративних моделей ШІ, у автоматизації та вдосконаленні процесу інженерії вимог, передбачаючи значний вплив на зниження кількості погано написаних вимог і збільшення ефективності в проєктах розробки ПЗ.

Для аналізу та управління вимогами у процесі інженерії вимог використовуються такі підходи та інструменти [7]:

а) Visual Narrator [8]: автоматизоване рішення на базі NLP, що виводить концептуальну модель з вимог користувачів, використовуючи такі методи як PoS (Parts of Speech) тегування для ідентифікації лінгвістичних патернів речень. Результат концептуальної моделі — це представлення системи, яке включає концепції, що допомагають людям знати, розуміти або імітувати предмет, який модель представляє. Результатом є "доменна онтологія", яка формує модель, кількісно описуючи об'єкти та відносини між ними таким чином, що програмне забезпечення може їх інтерпретувати;

б) QuARS (Quality Analyzer for Requirements Specification) [9]: представляє собою інструмент або методологію, призначену для оцінки якості специфікацій вимог. Його основна мета полягає у виявленні аспектів, де вимоги можуть бути неясними, непослідовними або неповними, що сприяє покращенню загальної якості вимог та зниженню ризику непорозумінь або

помилок у процесі розробки. Серед особливостей QuARS можна виділити контрольні списки або керівництва для оцінки якості вимог, інструменти для автоматизованого аналізу документів вимог з метою ідентифікації потенційних проблем, а також панелі управління або звіти, які надають уявлення про якість вимог і виокремлюють області для покращення;

в) Qualicen Requirements Scout (QRS) [10]: програмний інструмент, який надає рішення для інженерії та управління вимогами. Допомагає організаціям управляти, відстежувати та аналізувати їхні вимоги від моменту їх створення до остаточного впровадження. Він пропонує централізоване сховище для вимог, які можуть бути організовані, відстеженні та контрольовані протягом часу, забезпечуючи можливість відстеження вимог від їх початкового створення до остаточного впровадження. Таке відстеження гарантує, що вимоги належним чином управляються та що будь-які зміни контролюються та відстежуються.

Крім того, QRS включає функції аналізу вимог, такі як аналіз впливу, аналіз простежуваності та аналіз якості, які допомагають організаціям забезпечити повноту, послідовність та високу якість своїх вимог;

г) Semantha [11]: призначений для порівняння документів на семантичному рівні. Це означає, що Semantha може розуміти ідеї та значення, що перевищують буквальний рівень окремих слів і фраз, дозволяючи виявляти спільні концепції в різних документах, порівнювати їх та виділяти відмінності. Ця здатність робить інструмент надзвичайно корисним для класифікації вимог, ідентифікації пов'язаних ризиків та документування порівняння на семантичному рівні;

д) ReqSuite [12]: особливістю ReqSuite є функція допомоги, яка дозволяє автоматично виявляти кілька ключових проблем у вимогах: концептуальну неповноту (коли важливі вимоги повністю забуваються зацікавленими сторонами або коли вимоги передбачають додаткові вимоги, які відсутні), неточність вимог (коли не зрозуміло, чи є вимога обов'язковою чи опціональною) та суперечності між вимогами.

Цей інструмент допомагає забезпечити, що всі вимоги є повними, точними та не містять внутрішніх суперечностей, що значно знижує ризик виникнення помилок у процесі розробки та впровадження проєктів. Використання ReqSuite може спростити процес управління вимогами, забезпечуючи краще розуміння та узгодженість вимог усіма учасниками проєкту, а також зменшити час та витрати на перегляд і корекцію вимог;

е) IBM Engineering Requirements Quality Assistant (ERQA) [13]: використовує передові можливості обробки природної мови для автоматизованого аналізу вимог. Це дозволяє виявляти потенційні двозначності та генерувати реальні оцінки в реальному часі для оцінки якості вимог. Система оцінювання базується на кількох концепціях, таких як складні вимоги, неточні дієслова, неповнота, відсутність обмежень, відсутні одиниці виміру, негативні твердження, незрозумілі займенники тощо.

Інструмент значно спрощує процес перевірки вимог, дозволяючи швидко ідентифікувати можливі проблеми з якістю та забезпечити, що вимоги є чіткими та однозначними перед початком розробки. Використання IBM ERQA може значно знизити ризики, пов'язані з неправильним тлумаченням вимог, та покращити якість кінцевого продукту, забезпечуючи ефективнішу співпрацю між усіма учасниками проєкту;

ж) генеративний ШІ, такий як GPT-4 від OpenAI, в контексті інженерії вимог може використовуватися для уточнення, ідентифікації потенційних конфліктів або проблем та надання рекомендацій для поліпшення якості вимог, а також генерування тестових випадків і підтримки управління та документації вимог.

На основі аналізу існуючих інструментів і методів в області інженерії вимог, які використовують технології обробки природної мови, можна зробити висновок, що багато із них вирішують завдання аналізу та класифікації вимог, виявлення неоднозначностей, та навіть деякої міри валідації вимог щодо їх повноти та консистентності. Проте, ці інструменти часто обмежуються рамками текстового аналізу та не здатні прямо аналізувати

та інтегрувати виявлені сутності з існуючими системами або візуалізувати зв'язки між сутностями у формі графів, що може відіграти важливу роль у виявлення системних вимог [14] та значно обмежує їх застосування для комплексного розуміння структур даних проєкту.

У контексті кваліфікаційної роботи, відкриваються нові можливості для глибшого розуміння взаємозв'язків між сутностями, що вимагаються у проєкті. Цей підхід не лише дозволяє ідентифікувати та класифікувати сутності в тексті вимог, але й забезпечує засоби для їх зіставлення з існуючими структурами даних та візуального представлення цих зв'язків. Таким чином, розроблений підхід може стати вагомим доповненням до наявних інструментів, заповнюючи існуючі прогалини у візуалізації та аналізі структурних зв'язків між сутностями на основі вимог до ПЗ.

1.4 Висновки проведеного аналізу та формування вимог дослідження

Результати аналізу сучасного стану проблем формування вимог та створення опису архітектури складних ІС дозволяють зробити наступні висновки.

По-перше, існуючі методи та підходи до визначення системних вимог часто стикаються з проблемою неоднозначності та невизначеності. Це призводить до труднощів у визначенні вимог до системи, що може спричинити помилки на ранніх етапах проєктування та збільшити витрати на внесення змін у майбутньому.

По-друге, на сьогоднішній день існує дефіцит повноцінних теоретичних та практичних рішень, здатних на автоматизовану обробку вимог до систем. Особливо це стосується забезпечення глибокої інтеграції виявлених сутностей з існуючими системами та візуалізації їх для детального аналізу. Така ситуація

ускладнює розробку та адаптацію ІС, підвищуючи ризик неврахування важливих аспектів на ранніх етапах проєктування.

По-третє, існує велика потреба в інтеграції досвіду з попередніх проєктів та використання передових практик при визначенні вимог. Недостатня увага до досвіду минулих проєктів та відсутність систематичного підходу до збору та аналізу цього досвіду обмежує можливість врахування вже існуючих рішень та помилок.

По-четверте, в сучасному контексті розробки та модифікації систем, суб'єктивний вплив заснованного на попередньому досвіді може ініціювати упередженість, що, у свою чергу, негативно позначається на процесі ухвалення рішень та прогресу систем.

Дані висновки дозволяють припустити, що процес визначення вимог висвітлюють критичну потребу в розвитку і впровадженні автоматизованих методів аналізу текстових даних для виявлення та розуміння системних вимог, особливо в контексті управління ІТ-проєктами.

Недоліки існуючих методів, такі як неоднозначність та відсутність комплексних рішень для їх автоматизованої обробки, підкреслюють необхідність для фахівців шукати нові підходи та технології. Цей процес також має включати ефективне використання уроків, винесених з минулого досвіду, і водночас застосування сучасних аналітичних інструментів для уникнення упередженості.

Виділені проблеми є наслідком недостатніх теоретичних та практичних досліджень у сфері формування та аналізу вимог до ІС. Основні публікації зарубіжних дослідників [15 - 16] переважно спрямовані на осмислення практичного досвіду у цій області. Проте в цих та інших роботах майже не розглянуто можливості використання автоматизованих інструментів для визначення та аналізу системних вимог.

Незважаючи на значний прогрес у розробці технологій та методологій для управління проєктами, автоматизоване виявлення сутностей з вимог, їх адаптація та зіставлення з елементами системи для подальшого аналізу

залишаються невирішеною проблемою. Це створює вакуум, у якому втрачається потенціал для підвищення ефективності процесу розробки та зменшення ймовірності помилок на ранніх етапах проєктування.

Робота спрямована на заповнення цієї прогалини шляхом розробки технології, який дозволить автоматизувати процес виявлення та аналізу сутностей вимог з допомогою NLP та процесом групування відмінюваних форм (лематизація) з подальшим перетворенням результатів до необхідного набору даних та інтеграцією з вже існуючими елементами системи для подальшого аналізу.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– аналіз сучасних методів NLP, зокрема групування відмінюваних форм, та їх застосування у сфері управління IT-проектами;

– проектування архітектури та визначення технологічного стеку для розробки інформаційної технології;

– розробка механізму зіставлення виявлених сутностей з існуючими елементами системи;

– розробка інформаційної технології для виявлення та класифікації термінів проєкту в текстових документах, з акцентом на групування відмінюваних форм;

– апробація розробленої системи на реальних даних.

АДАПТАЦІЯ ПРОЦЕСУ ГРУПУВАННЯ ВІДМІНЮВАНИХ ФОРМ ДЛЯ АНАЛІЗУ ТЕКСТОВИХ ДАНИХ В ВИМОГАХ ДО ПРОЄКТУ

2.1 Концептуальний опис процесу інтегрованого аналізу вимог на основі обробки природньої мови

У рамках цього дослідження розроблено процес інтегрованого аналізу вимог, який користується передовими досягненнями в галузі обробки природної мови для оптимізації та автоматизації виявлення, класифікації та аналізу вимог до інформаційних систем.

Основою для розробки даного процесу послужили теоретичні положення та методологічні принципи, викладені в монографії [17], де представлено загальний опис процесу управління вимогами до інформаційних систем що виконується протягом всього ЖЦ вимоги. Процес інтегрованого аналізу вимог на основі обробки природньої мови описується наступним чином:

а) назва процесу: "Процес інтегрованого аналізу вимог на основі обробки природної мови" – це процес, який застосовує технології обробки тексту представлений людською мовою з допомогою NLP та процесом групування відмінюваних форм(лематизація) з метою виявлення ключових сутностей та їх взаємозв'язків;

б) очікувані результати: результатом процесу є представлення вимог у вигляді діаграми з виявленими сутностями та їх атрибутами з наявної ІС і зв'язками між визначеними сутностями. Діаграма слугує інтуїтивно зрозумілим засобом для ідентифікації та аналізу ключових сутностей, вказаних у текстових вимогах, та їх взаємодій. Візуалізація сприяє розумінню структури ІС, а також ефективно підтримує комунікацію між розробниками та зацікавленими сторонами, дозволяючи швидко ідентифікувати потенційні проблеми або недоліки у вимогах проєкту;

в) види діяльності:

аналіз вимог і виявлення сутностей: критичним етапом в процесі інтегрованого аналізу вимог є ретельний аналіз текстових матеріалів для ідентифікації ключових сутностей, що містяться у вимогах до інформаційної системи. Основною метою цієї діяльності є впровадження лематизація як метод оптимізації аналітичного процесу. Лематизація дозволяє здійснити групування відмінюваних форм слів до їх лексичної базової форми. Застосування цього методу сприяє формуванню словника сутностей, який включає начальні форми слів, що є ключовими для розуміння та аналізу вимог до інформаційної системи. Цей словник стає фундаментом для подальшої роботи з виявленням зв'язків між сутностями, а також для аналізу та проєктування структури ІС. Впровадження лематизації як елементу процесу інтегрованого аналізу вимог не тільки підвищує точність ідентифікації та класифікації вимог, але й забезпечує більш ефективне використання аналітичних інструментів для розробки відповідних рішень;

) виявлення зв'язків сутностей у ІС: у традиційному процесі управління вимогами аналіз зв'язків між сутностями часто здійснюється вручну та може не враховувати всі можливі зв'язки через обмеженість людського аналізу. Введення цього кроку, який використовує методи для виявлення та аналізу зв'язків, дозволяє створити більш комплексне розуміння як прямих, так і опосередкованих взаємозв'язків між сутностями вимог, підвищуючи якість розробки ІС;

) побудова графа залежностей виявлених сутностей.

г) цілі: процес має за мету збільшення ефективності та точності в процесі управління вимогами проєкту, використовуючи автоматизовані інструменти для аналізу представлених вимог у текстовому вигляді та їх подальшого зіставлення та ідентифікації в контексті ІС;

д) ресурси: основними ресурсами процесу є текстові дані вимог до проєкту, інструменти обробки природної мови, які застосовуються для аналізу цих даних та ІС до якої інтегрується процес;

е) механізми процесу: в центрі процесу інтегрованого аналізу вимог стоять особи, які приймають рішення. Це підкреслює важливість взаємодії між усіма учасниками проєкту у процесі аналізу вимог. Роль цих осіб полягає не лише в формулюванні та верифікації вимог, але й у прийнятті рішень на основі аналізу, проведеного за допомогою обробки природної мови. Вони відповідають за оцінку релевантності виявлених сутностей та визначення важливості зв'язків між ними, що в кінцевому рахунку впливає на прийняття рішень щодо розробки і впровадження системи.

На основі видів діяльності представлено декомпозицію процесу на рисунку 2.1.

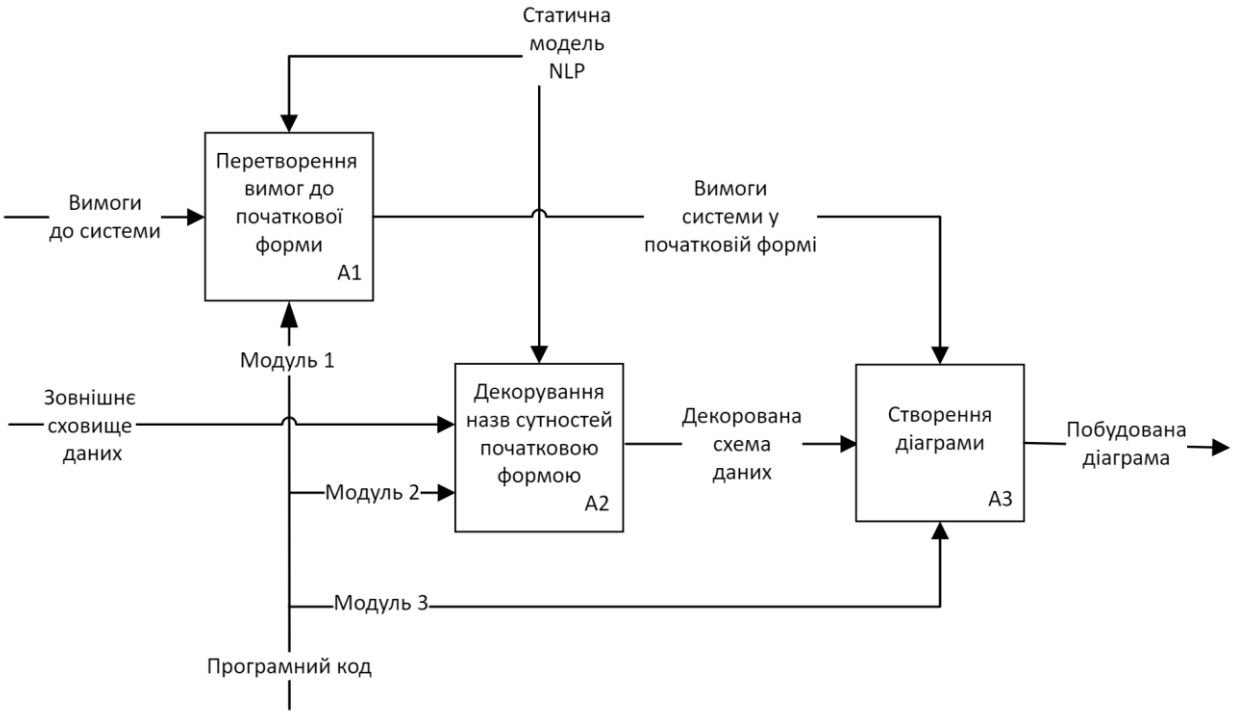

Рисунок 2.1 – IDEF0-діаграма взаємозв’язків основних діяльностей інформаційної технології виявлення сутностей з вимог до проєкту

2.2 Формальний опис елементів процесу

Для лематизації можуть використовуються різні методи та підходи, і вони можуть включати або поєднувати деякі математичні моделі.

Деякі алгоритми обробки природної мови використовують статистичні моделі для визначення леми на основі контексту та інших мовних ознак. Ці моделі навчаються на великих корпусах тексту і використовуються для передбачення леми для кожного слова у тексті.

Також використовуються правила на основі морфології мови для перетворення слів на їх базові форми. Ці правила можуть бути відображені у внутрішній реалізації. Таким чином, хоча деякі з цих математичних підходів можуть бути відображені у внутрішній роботі NLP, користувачам зазвичай надаються інтерфейси високого рівня для використання функціоналу лематизації без необхідності безпосередньої роботи з математичними моделями.

Для подальшого розуміння процесу лематизації та його методів реалізації, розглянемо детальніше кілька аспектів, які включають:

– статистичні моделі;

– правила на основі морфології;

– комбінація статистики та правил.

Кожен аспект розглянемо на прикладі вимоги до системи «Додати можливість для гравців виступати за декілька команд»:

а) статистична модель у контексті обробки природної мови (NLP) є моделлю машинного навчання, яка навчається на великих корпусах тексту з метою вивчення зв'язків між словами та їх лемами. Ці моделі аналізують частоту вживання слів у різних контекстах та інші мовні ознаки, щоб визначити ймовірність того, що дане слово в певному контексті буде мати певну лему:

Припустимо S – множина всіх слів у тексті, C – множина контекстів,

– множина всіх можливих лем у мові, тоді статистичні моделі визначають умовну ймовірність як:

$$p = f(l \mid w, c), \tag{2.1}$$

де $l \in L$ – лема слова $w \in S$ у контексті $c \in C$.

Отже, для прикладу візьмемо вимогу до системи «Додати можливість для гравців виступати за декілька команд»: та для слова «гравців», яке може вказувати на конкретну сутність у системі, ймовірність може виглядати наступним чином:

p = f ("гравець" | "гравців", "Додати можливість для гравців виступати за декілька команд").

У NLP умовна ймовірність *P* використовується для кожної леми, спочатку модель аналізує контекст кожного слова, порівнюючи його з статистичними зв'язками між словами та їхніми лемами. На основі цього аналізу модель визначає умовну ймовірність для кожної можливої базової форми слова у даному контексті. За допомогою цих умовних ймовірностей модель вибирає найбільш ймовірну базову форму слова для кожного слова у тексті;

б) правила на основі морфології використовує правила лематизації, які базуються на морфологічних особливостях мови. Наприклад, якщо українська морфологія передбачає правило про те, що форма «гравців» є родовим відмінком множини від форми «гравець», то це правило може бути використане для визначення леми «гравець» для слова «гравців».

Для цього функція *f* (2.2) повинна використовувати правило лематизації для слова *w* у контексті *c*:

$$p = f(w, c, \{r_i\}), \tag{2.2}$$

де $\{r_i\}$ – множина морфологічних правил, які застосовуються для визначення леми, якщо $\{r_i\} = 0$, то правила ігноруються.

Отже, вираз може виглядати як, *p= f ("гравців", "Додати можливість для гравців виступати за декілька команд", {морфологічне правило});*

в) комбінація статистики та правил поєднує результати статистичної моделі та правила на основі морфології для вибору леми для слова *w* у контексті *c*. Це означає, що використовується умовну ймовірність вибору леми на основі статистичних даних, а також можливість застосування правил для визначення леми, які не враховані статистичною моделлю.

Остаточна лема *l* для слова «гравців» обирається шляхом комбінації статистичної моделі та правил:

$$L^* = arg\ max_{l \in L} f\ (l \mid w, c, \{ri\}), \qquad (2.3)$$

відповідно, $l = arg\ max_{l \in L}\ f\ (l|$*"гравців", "Додати можливість для гравців виступати за декілька команд", {морфологічне правило}).*

Тобто, якщо статистична модель показує, що ймовірність «гравець» у цьому контексті найбільша, відповідно до цієї формули лема для «гравців» буде «гравець». Дана математична модель використовується у процесі «Аналіз тексту» який зображений під номером А1 на рисунку 2.1.

Далі для детального математичного моделювання процесу визначення сутностей у системі, базуючись на лематизованому списку отриманому з вимог, розглянемо наступні концепції:

а) відображення сутностей на таблиці бази даних: визначимо функцію відображення *f*: $E \rightarrow D$,

де *E* – множина сутностей отриманих з вимог;

D – множина сутностей у базі даних.

Ця функція дозволяє встановити відповідність між сутностями з вимог і таблицями бази даних;

б) аналіз атрибутів сутностей: припустимо що матриця *T*, представляє кожен рядок який відповідає таблиці $d_i \in D$, а стовпці представляють атрибути цих таблиць. Елементи матриці *E* вказують на тип даних атрибута *j* таблиці *i*.

Наприклад, якщо в БД є таблиця «гравець» з атрибутами «Ім'я» (текстовий тип) та «Вік» (цілочисельний тип), то це може бути представлено так:

$T_{\text{«гравець», «ім'я»}} = \text{«текст»},$

$T_{\text{«гравець», «вік»}} = \text{«число»};$

в) виявлення залежностей між сутностями моделюємо як орієнтований граф:

$$G = (V, E), \tag{2.4}$$

де *V* – вершини сутностей;

– ребра залежностей між сутностями.

Кожне ребро (v_i,v_j) ідентифікує, що сутність v_i залежить від сутності v_j, наприклад, через зовнішній ключ.

Можна додатково анотувати ребра графа атрибутами, що вказують на тип зв'язку (наприклад, "один до одного", "один до багатьох") [18]:

а) для зв'язку "один до одного" між сутностями *A* та *B*, ребро e_{AB} може мати атрибут тип зв'язку="1:1";

б) для зв'язку "один до багатьох" між *A* та *B*, ребро e_{AB} має атрибут тип зв'язку="1:N".

Процеси визначення сутностей у системі і побудова графа відображена на рисунку 2.1 як процеси А2 та А3.

2.3 Висновки до другого розділу

У розділ представлено розгорнутий аналіз процесу адаптації лематизації для аналізу текстових даних в рамках проєктних вимог. Цей аналіз включає концептуальне визначення та формалізацію процесу, а також розробку та використання математичних моделей для ефективного виявлення сутностей проєктних вимог.

У підрозділ 2.1 зосереджуємось на концептуальному описі цього процесу, де використання методів NLP допомагає оптимізувати ідентифікацію та класифікацію вимог до проєктів, що значно сприяє ефективності проєктного управління. Увесь процес відображається як послідовність різних діяльностей та наведе як діаграма IDEF0 на рисунку 2.1. Завдяки цьому можливе виявлення ключових сутностей та аналіз їх взаємозв'язків, що відображається у формі інтуїтивно зрозумілих діаграм.

Підрозділ 2.2 деталізує формальний опис елементів процесу, у якому обговорюється використання статистичних моделей та морфологічних правил для лематизації словесних форм. Це поєднання дозволяє точніше визначати леми в контексті використання слова, забезпечуючи точність в ідентифікації термінів, які мають значення для системного аналізу вимог.

Таким чином, розглянутий процес адаптації технологій NLP для аналізу текстових даних відкриває нові можливості для підвищення якості та ефективності управління IT-проєктами, дозволяючи краще розуміти та виконувати проєктні вимоги.

РЕАЛІЗАЦІЯ ПРОЦЕСУ АНАЛІЗУ ТЕКСТОВИХ ДАНИХ В ВИМОГАХ ДО ПРОЄКТУ

3.1 Архітектурний опис системи

3.1.1 Огляд архітектури

У цьому розділі представлено детальний опис архітектури розроблюваного сервісу, призначеного для аналізу текстових вимог до проєктів з використанням технологій обробки природної мови. Сервіс розробляється як рішення, що дозволяє його використання в різних сферах управління проєктами.

Для апробації розробленого процесу аналізу текстових даних у вимогах до проєкту використовується монолітний підхід, що у контексті розробки ПЗ дозволяє інтегрувати всі необхідні компоненти обробки, аналізу та візуалізації даних у єдине цілісне рішення. Однак, з огляду на використання новітніх технологій, архітектура системи розроблена таким чином, що в майбутньому вона може бути легко адаптована, наприклад, до мікросервісної моделі для підвищення масштабованості та ефективності обробки запитів.

Однією з вимог до системи є конфігурація сервісу для аналізу сховища даних та її сутностей, що дозволяє налаштувати систему для обробки вимог незалежно від мови користувача. У випадку, якщо вимоги подані українською мовою, конфігурація включатиме підключення до бази даних, де сутності визначені українською мовою, а для обробки вимог завантажується статична модель, відповідна до мови користувача.

3.1.2 Компоненти системи

Систему, яка реалізує процес інтегрованого аналізу вимог на основі обробки природньої мови пропонується представити як сукупність таких компонентів:

а) модуль перетворення вимог до початкової форми: модуль відповідає за токенізацію вхідної рядкової вимоги, розбиваючи її на окремі слова, які служать вхідними даними для процесу лематизації. У результаті лематизації кожне слово в реченні перетворюється на свою базову форму, що сприяє уніфікації та стандартизації тексту для подальшого аналізу вимог;

б) модуль декорування схеми бази даних: працює з зовнішньою схемою сховища даних, де виконується токенізація та лематизація назв таблиць і колонок. Після цього модуль використовує отримані леми для декорування сутностей вхідної схеми сховища даних;

в) модуль створення діаграми зв'язків: відповідає за встановлення відповідностей між лемами вимог користувача, отриманими з модуля 1, та лемами схеми бази даних, обробленими модулем 2 та генерації діаграми для аналізу системи.

Візуалізація архітектури системи

Для ілюстрації архітектури системи використовуються діаграми компонентів, які демонструють взаємодію між основними частинами сервісу, та діаграми послідовностей, що відображають процес обробки запиту від моменту його надходження до генерації відповіді.

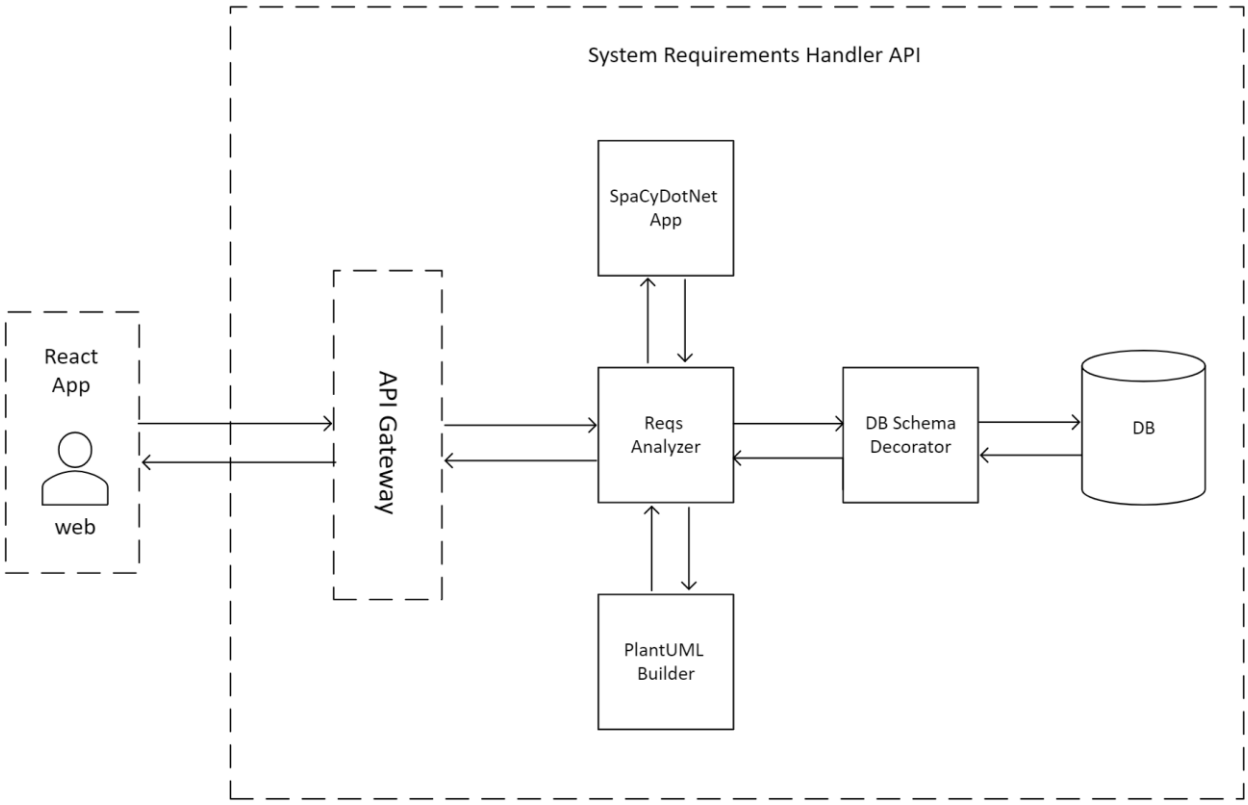

Рисунок 3.1 – Схема архітектури системи обробки вимог і зв’язки між компонентами

3.2 Технологічний стек основних модулів системи

3.2.1 Модуль обробки природної мови

У розробці сервісу для аналізу текстових даних в вимогах до проєкту ключову роль відіграють вибрані технології та інструменти, кожен з яких виконує специфічну функцію у загальній архітектурі системи.

Написання сервісу для обробки вимог до системи відбувається з допомогою мови програмування C# та кросплатформеної технології .NET Core [19] від MS.

Основою для обробки природної мови є мова програмування Python, яка підтримує широкий спектр бібліотек. Серед цих бібліотек spaCy [20] є ключовими інструментами який надає розширені можливості для лематизації

та морфологічного аналізу, включаючи підтримку багатьох мов включаючи українську мову. Для використання вище зазначених інструментів та досягнення створення загального цілісного рішення пропонується інтеграція середовищі .NET, зокрема в .NET Core, і базується на Python.NET для забезпечення взаємодії між Python та .NET.

3.2.2 Модуль інтеграції зі сховищем даних

Модуль інтеграції зі сховищем даних забезпечує з'єднання між API сервісом та базою даних для аналізу існуючих схем і відповідностей. Використовуючи технології MS SQL Server та бібліотеку Microsoft.SqlServer.Management.Smo, модуль дозволяє встановити з'єднання з базою даних та виконати аналіз її схеми. Це включає вилучення та аналіз інформації про таблиці, стовпці, типи даних та інші метадані, що необхідні для кореляції із лематизованими токенами отриманими з вимог проєкту. Цей модуль спрощує процес виявлення відповідностей між проєктними вимогами та існуючими елементами в базі даних, забезпечуючи важливу основу для подальшого аналізу та візуалізації зв'язків.

Модуль створення діаграми зв'язків

Для генерації діаграм, що представляють зв'язки між знайденими сутностями у схемі бази даних, використовується інструмент PlantUML [22] – він призначений для створення діаграм програмного забезпечення, який

дозволяє описувати структуру програми за допомогою текстового опису та автоматично генерувати візуальні діаграми на основі цього опису використовує простий синтаксис, що базується на мові UML.

4 Клієнтська частина для користувача

Для реалізації клієнтської частини системи, що забезпечить взаємодію користувача з сервісом для можливості подання вимог до обробки, заплановано використання фреймворку React [23]. Який був обраний за рахунок свого розширеного набору готових до використання стилізованих компонентів. Зокрема, бібліотека компонентів BlueprintJs [24] буде використана для створення інтуїтивного інтерфейсу.

Цей технологічний стек був обраний з огляду на здатність до швидкої розробки, підтримки різних людських мов для наочності та демонстрації підходу, а також легкості інтеграції між різними компонентами.

3.3 Висновки до третього розділу

Третій розділ присвячений розробці архітектури та компонентів системи для аналізу текстових вимог до проєктів з використанням технологій обробки природної мови. Система спроектована з використанням монолітного підходу, що дозволяє інтегрувати всі компоненти обробки, аналізу та візуалізації даних в єдине цілісне рішення. Архітектура системи розроблена таким чином, що в майбутньому вона може бути легко адаптована до мікросервісної моделі для підвищення масштабованості та ефективності обробки запитів.

Система складається з кількох ключових компонентів. Модуль перетворення вимог до початкової форми забезпечує токенізацію та лематизацію вхідних текстових вимог, що сприяє уніфікації та стандартизації тексту для подальшого аналізу. Модуль декорування схеми бази даних виконує токенізацію та лематизацію назв таблиць і колонок, що дозволяє встановити відповідності між лемами вимог користувача та лемами схеми бази даних. Модуль створення діаграми зв'язків генерує діаграми для аналізу системи, які представляють зв'язки між знайденими сутностями у схемі бази даних.

Для реалізації системи використовуються сучасні технології. Зокрема, використання Net технологій. Інтеграція з базою даних за допомогою MS SQL Server та бібліотеки Microsoft.SqlServer.Management.Smo забезпечує надійне з'єднання та аналіз існуючих схем і відповідностей. Використання PlantUML для генерації діаграм забезпечує наочне представлення зв'язків між сутностями у схемі бази даних, що полегшує їх аналіз. Створення клієнтської частини системи за допомогою фреймворку React забезпечує зручний та інтуїтивний інтерфейс для користувача.

ПРОГРАМНА РЕАЛІЗАЦІЯ ТА АПРОБАЦІЯ СЕРВІСУ ОБРОБКИ ВИМОГ ДО СИСТЕМИ

Реалізація системи та її компонентів

У третьому розділі було зазначено, що однією з основних вимог до системи є її здатність обробляти вимоги користувачів на різних мовах, зокрема англійською та українською. Для задоволення цієї вимоги система була спроектовано з використанням гнучкої конфігурації API, що дозволяє легко інтегрувати підтримку додаткових мов.

Конфігураційний файл системи містить окремі секції для кожної мови, де визначаються параметри для обробки тексту, взаємодії з базою даних, ідентифікації колонок та регулярних виразів для обробки тексту, що дозволяє адаптувати систему під конкретні мовні особливості. Моделі для обробки тексту використовують можливості бібліотеки spaCy та може бути обрана будь яка мова підтримувана цією бібліотекою. Для апробації налаштовано використання української мови з допомогою попередньо навченої моделі uk_core_news_sm, а для англійської – модель en_core_web_sm.

```
"DataSet": {
  "ua": {
    "LangModel": "uk_core_news_sm",
    "ConnectionDb": "Data Source=localhost;Initial Catalog=ФутбольнаЛіга;User ID=nure;
    "ColumnIdentifierSuffix": "Код",
    "ColumnSplitRegex": "[А-ЯЁЇІЄҐ][^А-ЯЁЇІЄҐ]*"
  },
  "en": {
    "LangModel": "en_core_web_sm",
    "ConnectionDb": "Data Source=localhost;Initial Catalog=FootbalLeague;User ID=nure;
    "ColumnIdentifierSuffix": "Id",
    "ColumnSplitRegex": "[A-Z][^A-Z]*"
  }
}
```

Рисунок 4.1 – Фрагмент конфігурації мовних моделей інформаційної технології

Такі параметри як LangModel, ConnectionDb, ColumnIdentifierSuffix та ColumnSplitRegex забезпечують не тільки адекватне розпізнавання та обробку мовних структур, але й належну взаємодію з базами даних та адаптацію до специфічних мовних ідентифікаторів у даних.

Для демонстрації підтримки багатомовності, зокрема обробки даних на українській та англійській мовах, у межах системи було створено два окремі сховища даних (рисунок 4.2). Кожне сховище налаштовано відповідно до своєї мовної конфігурації, що дозволяє оптимізувати обробку та зберігання даних відповідно до специфічних мовних вимог. Ці сховища названі ФутбольнаЛіга для української версії та FootballLeague для англійської версії, які було розроблено на платформі Microsoft SQL Server.

Як було зазначено вище, конфігурації цих баз даних відповідають кожній мові кожній навченої моделі, де ФутбольнаЛіга використовує модель uk_core_news_sm для обробки українського тексту, та відповідно, FootballLeague використовує модель en_core_web_sm для обробки тексту англійською мовою. Це забезпечує системі гнучкість та ефективність при роботі з даними, що надходять на різних мовах.

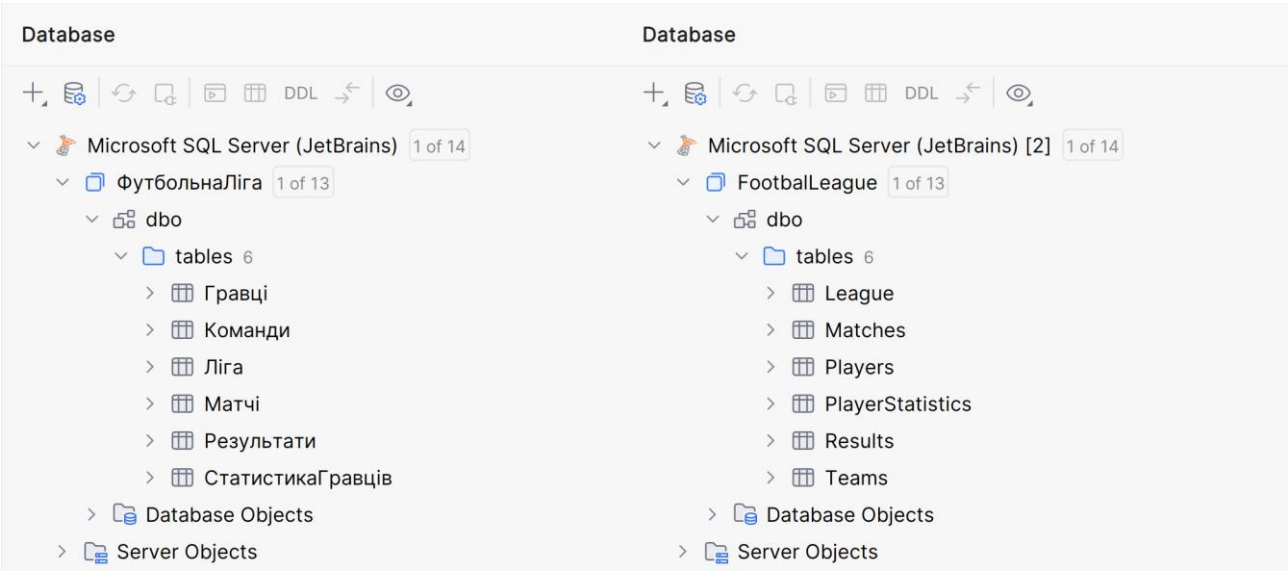

Рисунок 4.2 – Структура сховищ даних для апробації обробки вимог на різних мовах

В рамках реалізації модулів, зазначених на рисунку 3.1, а саме для визначення лем з тексту вимог та DB Schema Decorator для визначення і декорації лемами схеми бази даних, використовується метод nlp з бібліотеки spaCy. Цей метод є частиною стандартного інтерфейсу обробки тексту в spaCy.

Проте, в нашій системі ми використовуємо SpacyDotNet, обгортку для .NET, яка дозволяє інтегрувати функціональність spaCy у додатки, написані на C# тому в нашому випадку ми використовуємо клас Lang і його метод який повертає об'єкт Doc (рисунок 4.3), він містить структуровану інформацію про оброблений текст, і його властивості такі як Text, Lemma, POS, Tag і т.д.

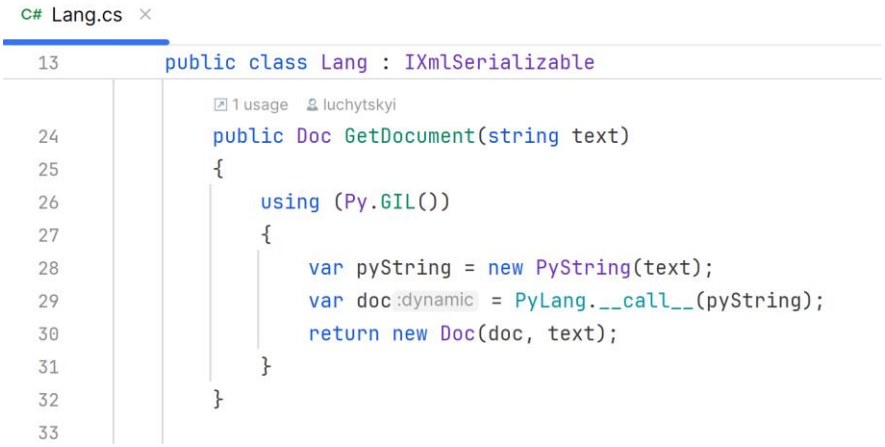

Рисунок 4.3 – Загальний метод для обробки тексту

Далі наведено програмний код класу DbEntitiesLemmatizer, який декорує схему бази даних лемами на основі сутностей і їх назв:

```
using System.Text.RegularExpressions;
using Microsoft.IdentityModel.Tokens;
using Microsoft.SqlServer.Management.Smo;
using ReqsHandler.Core.Configuration;
using ReqsHandler.Core.Services.Models;

namespace ReqsHandler.Core.Services;

public class DbEntitiesLemmatizer(ISpacyInstance spacyInstance, ICurrentContext context)
   : IDbEntitiesLemmatizer
{
```

Р
и
с
у
н

```
private Regex ColumnSplitRegex => new Regex(context.DataSet.ColumnSplitRegex);

public IEnumerable<ReqsTable> DecorateEntitiesWithLemma(TableCollection collection)
{
  var tables = new List<ReqsTable>();
  foreach (Table table in collection)
  {
    var isSplitName = SplitAndLemmatize(table.Name, out var splitList, out var lemmas);
    tables.Add(new ReqsTable
    {
      Name = table.Name,
      Lemmas = lemmas,
      IsSplitName = isSplitName,
      SplitNames = splitList,
      BaseEntity = table,
      Columns = MapColumnLemma(table.Columns)
    });
  }

  return tables;
}

private bool SplitAndLemmatize(string name, out IList<string> splitList, out IEnumerable<string> lemmas)
{
  splitList = Array.Empty<string>();
  lemmas = Array.Empty<string>();
  if (name.IsNullOrEmpty())
  {
    return false;
  }

  var isSplitName = SplitIfNeed(name, out splitList);
  var doc = spacyInstance.GetDocument(string.Join(" ", splitList));
  lemmas = doc.Tokens.Where(t => !t.IsPunct).Select(t => t.Lemma.ToLower());

  return isSplitName;
}

private bool SplitIfNeed(string entityName, out IList<string> result)
{
  var isSplit = false;
  result = Array.Empty<string>();
  var parts = ColumnSplitRegex.Matches(CleanUpName(entityName));
  if (parts.Count > 1)
  {
    isSplit = true;
  }

  result = new List<string>();
  foreach (Match part in parts)
  {
    result.Add(part.Value);
  }

  return isSplit;
}
```

Рисунок 4.4, аркуш 2

```
  private IEnumerable<ReqsColumn> MapColumnLemma(ColumnCollection tableColumns)
  {
    var columns = new List<ReqsColumn>();
    foreach (Column column in tableColumns)
    {
      var isSplitName = SplitAndLemmatize(column.Name, out var splitList, out var lemmas);
      columns.Add(new ReqsColumn
      {
        Name = column.Name,
        Lemmas = lemmas,
        BaseEntity = column,
        IsSplitName = isSplitName,
        SplitNames = splitList
      });
    }

    return columns;
  }

  private string CleanUpName(string name)
  {
    return name.Replace(context.DataSet.ColumnIdentifierSuffix, "");
  }
}
```

Рисунок 4.4, аркуш 3

Основний публічний метод цього класу, DecorateEntitiesWithLemma, призначений для обробки колекцій таблиць. Він приймає колекцію об'єктів типу Microsoft.SqlServer.Management.Smo.TableCollection і повертає колекцію об'єктів типу ReqsTable, кожен з яких містить леми для відповідних елементів таблиці.

Процес визначення лем відбувається в методі SplitAndLemmatize. Цей метод аналізує назви об'єктів бази даних, використовуючи регулярні вирази, визначені в конфігурації, яка показана на рисунку 4.1, для визначення необхідності розбиття складних назв на окремі слова. Наприклад, назва таблиці «СтатистикаГравців» (рисунок 4.2) в одне слово не дозволяє spaCy ефективно токенізувати її, оскільки spaCy розроблений для роботи зі словами, що мають звичні міжсловесні розділові знаки. Тому, метод SplitAndLemmatize перш за все перевіряє, чи потрібно розділити назву на більш малі фрагменти, що значно покращує точність подальшої обробки вимог.

Також треба зазначити, що, перед тим як розбивати або аналізувати текст, використовується метод CleanUpName, який з допомогою зазначеного в

конфігурації властивості ColumnIdentifierSuffix видаляє суфікс яким зазвичай помічають в структурованих базах даних поле з ідентифікатором, в наведеній схемі це суфікс «Код» для української локалізації та «Id» відповідно для англійської мови.

Після того як леми визначені для тексту, який надав користувач, і схеми бази даних починається процес побудови діаграми.

Клас PlantUmlBuilder відіграє ключову роль у створенні коду для візуалізації діаграми з допомогою PlantUML. Основний метод цього класу BuildUml, який створює текстове представлення діаграми, організовуючи таблиці та їхні взаємозв'язки (рисунок 4.5).

Особливість PlantUmlBuilder полягає в його здатності деталізувати кожну сутність за допомогою лем, отриманих з тексту вимог, що дозволяє відобразити можливі структурні і логічні зв'язки між компонентами системи. Такий підхід забезпечує більшу зрозумілість і корисність діаграм, які використовуються для аналізу і планування системних вимог.

C# PlantUmlBuilder.cs

```
8   public class PlantUmlBuilder(List<TableDto> tables)

        1 usage  Viacheslav Luchytskyi +1
28      private string BuildUml(IList<string> inputLemmas, HashSet<string> includedTables)
29      {
30          var diagram = new StringBuilder();
31          diagram.AppendLine("@startuml");
32          diagram.AppendLine(DefinedFunctions);
33          diagram.AppendLine();
34
35          // Definition of tables
36          var result:string = BuildTableEntities(inputLemmas, includedTables);
37          if (result.IsNullOrEmpty())
38          {
39              return string.Empty;
40          }
41
42          diagram.Append(result);
43
44          // Definition of relationships between tables
45          diagram.Append(BuildDependentTables(inputLemmas, includedTables));
46
47          diagram.AppendLine("@enduml");
48          return diagram.ToString();
49      }
```

Рисунок 4.5 – Фрагмент коду побудови UML діаграми

Як зазначено у пункті 3.1.3, архітектура системи для обробки вимог користувачів по виявленню атрибутів та артефактів з природньої мови передбачає впровадження різних модулів системи, які виконують свої функції для досягнення кінцевого результату. Всі модулі забезпечують доступ до функціональності через REST API, реалізовані на платформі ASP.NET Core, що забезпечує високу продуктивність і масштабованість системи.

Для оптимізації процесів налагодження та документації кожного модуля було вирішено забезпечити доступ до API інтерфейсу та впровадити використання Swagger [25], як це представлено на рисунку 4.6. Цей інструмент дозволяє тестувати API інтерфейси та водночас служить засобом їх документації, значно спрощуючи інтеграцію та перевірку взаємодії між модулями.

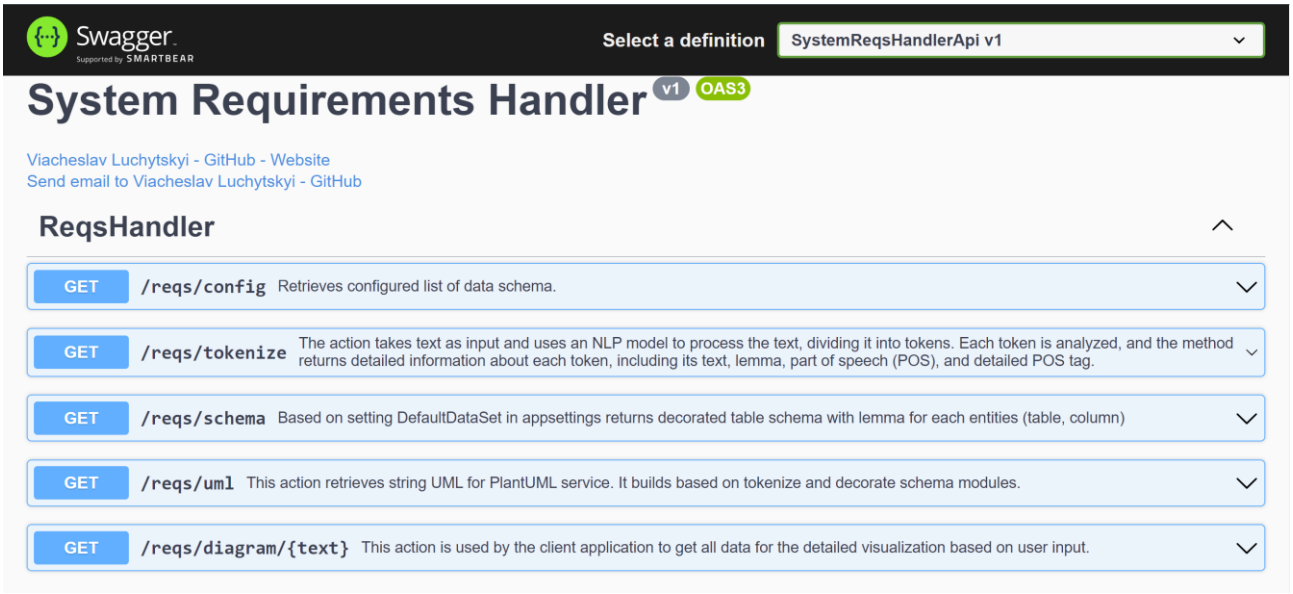

Рисунок 4.6 – Інтерфейс для тестування API системи

Апробація результатів на основі реалізації

Згідно з архітектурою системи, описаною у третьому розділі, для доступу до сервісу пропонується використовувати клієнтський інтерфейс, для чого було розроблено просте користувацьке середовище, де є вибір сховища даних для аналізу, поле для вводу вимоги та список результатів аналізу вимоги:

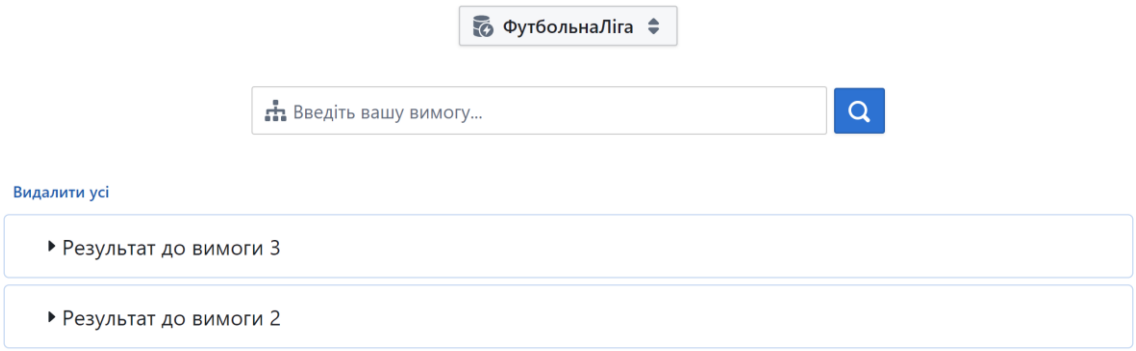

Рисунок 4.7 – Користувацький інтерфейс для обробки вимоги

Далі розглянемо результати апробації системи на прикладі конкретних вимог. На рисунку 4.8 продемонстровано результат для наданої вимоги «Додати можливість для гравців виступати за декілька команд». В ході аналізу вибраної системи були виявлені та виділені зеленим кольором сутності, що мають відношення до даної вимоги, зокрема таблиці та поля, що представляють гравців і команди, а також виявленні залежності між таблицями, що відображають взаємозв'язки знайдених сутностей з іншими сутностями систем.

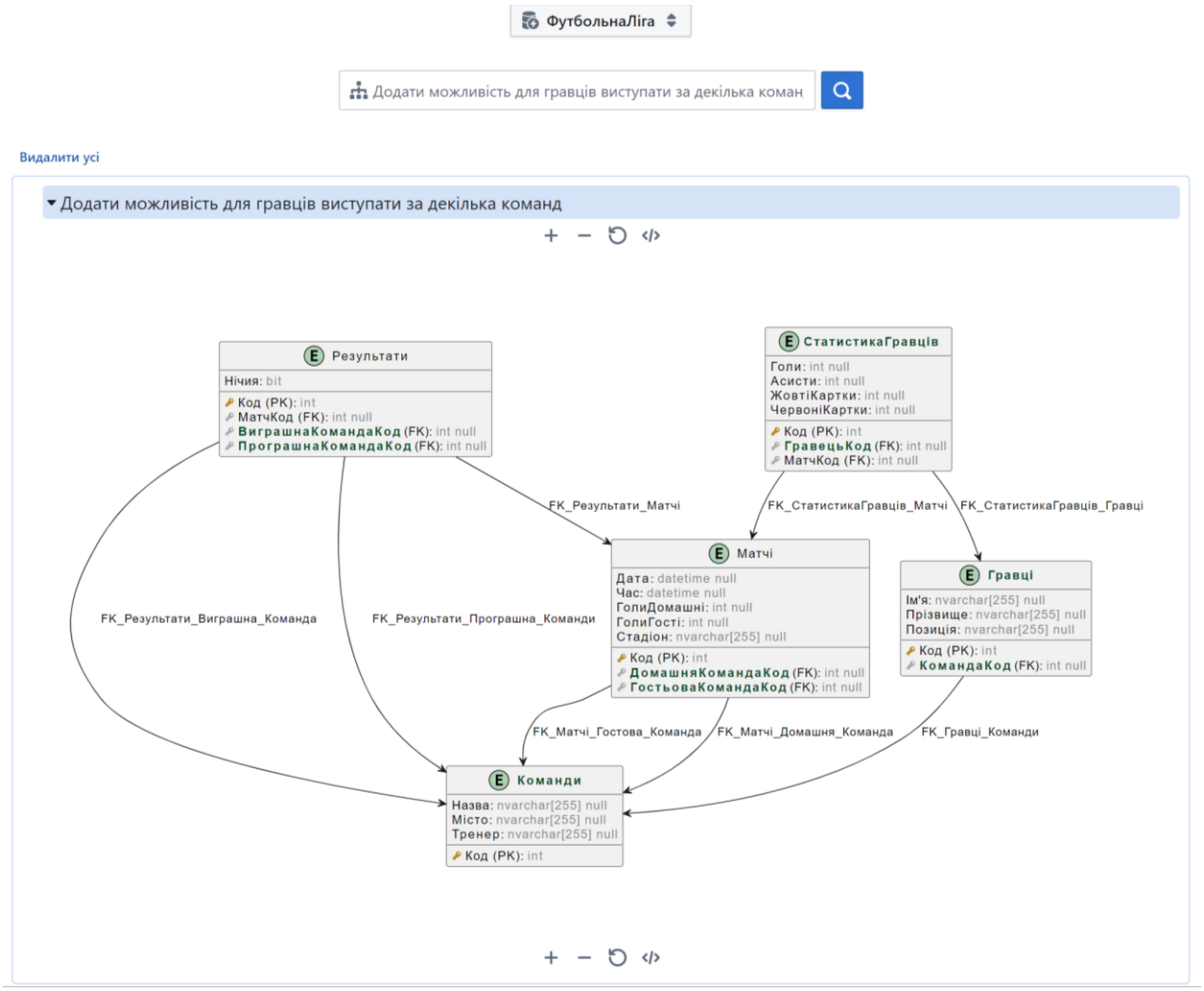

Рисунок 4.8 – Результат аналізу вимоги

Далі переведемо вимогу на англійську і продемонструємо результат, попередньо змінивши на відповідну схему бази даних зі списку рисунок 4.9. Як бачимо система так само визначила начальні форми для слів «players» та та візуалізувала їх на діаграмі.

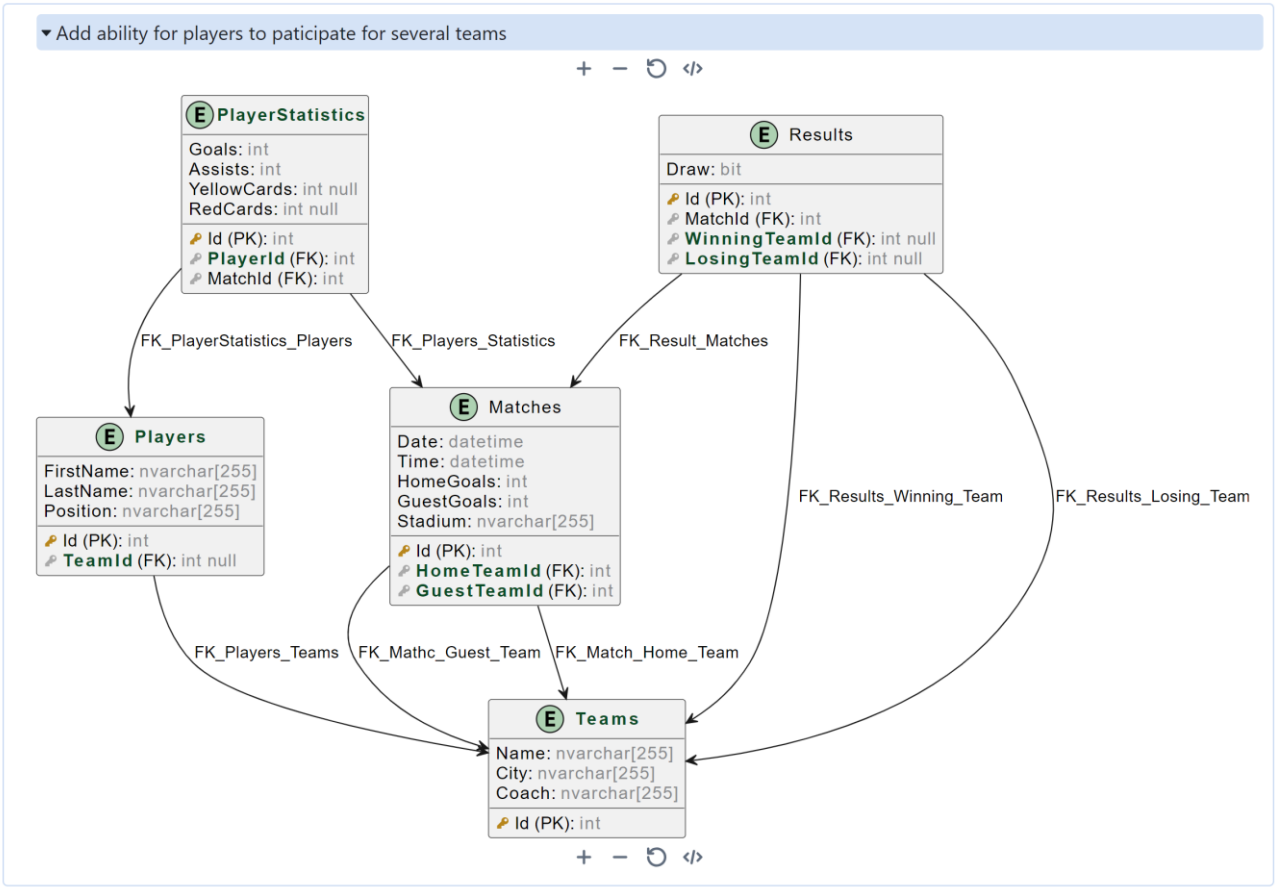

Рисунок 4.9 – Результат аналізу вимоги наданий англійською мовою

Результат апробації показав, що система ефективно ідентифікує ключові сутності у текстах вимог і класифікує їх з достатньою точністю. Це дозволяє забезпечити високу якість аналізу вимог та оптимізувати процес управління проектами, знижуючи ризики пов’язані з неправильним тлумаченням вимог. Але є і недоліки котрі були знайдені в процесі апробації системи, серед яких можна виділити наступні:

– визначення коректного контексту для артефактів знайдених у вимогах, наприклад, для вимоги «Increase the number of characters for the team name» атрибут «name» буде визначено системою для кожної сутності, а не тільки для сутності «team», яка зазначена у вхідному тексті вимоги, дивитись рисунок

– визначення правильних типів зв’язків як «один до одного», «один до багатьох», потребує можливо додаткової імплементації в системі для виконання запитів на стороні бази даних.

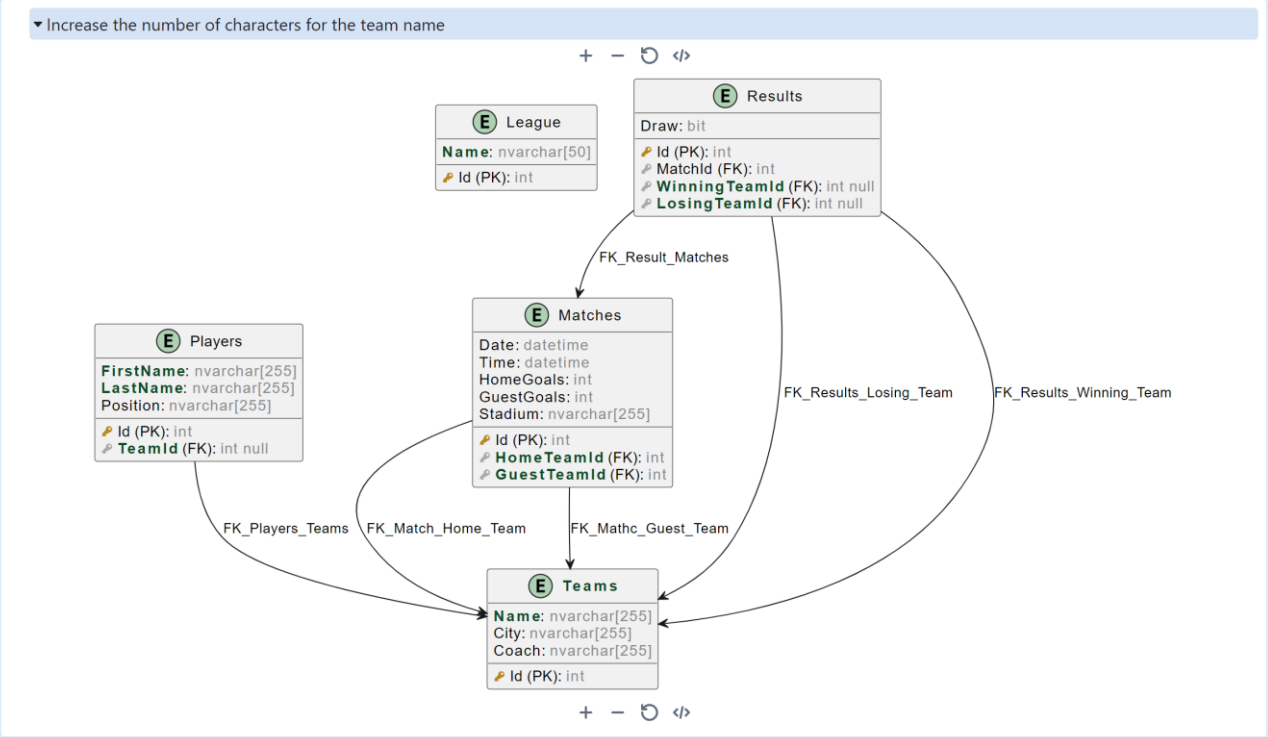

Рисунок 4.10 – Результат визначення артефакту з невизначеною сутністю у контексті вимоги

Інші приклади аналізу вимог і інструменти для оцінювання результатів системи наведені у Додатку А.

Висновки до четвертого розділу

Четвертий розділ присвячений програмній реалізації та апробації сервісу обробки вимог до системи. У цьому розділі описано практичну реалізацію компонентів системи, включаючи їх верифікацію та документування.

Розробка системи здійснювалася з урахуванням багатомовної підтримки, що дозволяє обробляти вимоги користувачів на різних мовах. Конфігураційний файл системи містить окремі секції для кожної мови, де визначаються параметри для обробки тексту, взаємодії з базою даних та ідентифікації колонок, що дозволяє адаптувати систему під конкретні мовні особливості. Для обробки тексту використовуються моделі бібліотеки spaCy, зокрема статична українська модель uk_core_news_sm та англійська модель

Було проведено апробацію системи на реальних даних, що включала тестування її компонентів та оцінку їх ефективності. Система продемонструвала здатність виявляти такі терміни системи як сутності таблиць з баз даних і їх артефакти у вигляді типів та різних властивостей виявлених сутностей, а також візуалізувати зв'язки між ними у вигляді діаграм. Це підтвердило її ефективність у підвищенні точності та швидкості аналізу системи на основі наданих вимог до проєкту.

ВИСНОВКИ

У рамках магістерської роботи розглянуто методи виявлення термінів проєкту та їх артефактів за допомогою обробки природної мови. Цей підхід може використовуватись для поліпшення процесу визначення вимог до проєкту, згідно з вимогами стандарту ДСТУ ISO/IEC/IEEE 15288:2016, сприяючи ефективному керуванню процесами розробки, впровадження та підтримки програмного забезпечення. Основна увага була спрямована на розв'язання таких проблем як ефективного використання попереднього досвіду та уникнення упередженості, заснованому на особистому досвіді. Для досягнення цієї мети було сформульовано та вирішено наступні завдання з автоматизації виявлення вимог на основі обробки природної мови (NLP):

а) аналіз сучасних підходів до аналізу текстових даних: проведено порівняльне дослідження стемінга та лематизації для визначення оптимальної методики обробки природної мови в контексті проєктних вимог. Лематизація, яка враховує морфологічні особливості мови, забезпечує більш точний аналіз, що є критично важливим для точного виявлення та класифікації термінів проєкту;

б) інформаційна технологія для виявлення сутностей на основі текстових даних: спроектовано архітектуру та визначено технологічній стек, на основі бібліотеки яка використовується для обробки природньої мови і включає процес лематизації, з метою розпізнавання сутностей у базах даних, які є одним з компонентів інформаційної системи;

в) розробка механізму зіставлення виявлених та адаптованих сутностей з існуючими елементами системи: Було розроблено програмний функціонал для виявлення лем сутностей схеми БД, таких як таблиць та колонок, що дозволило декорувати схему БД уніфікованим списком лем для кожної сутності. Такий підхід сприяв ефективній кореляції між структурним

елементами бази даних та текстовими даними вимог для їх подальшого зіставлення;

г) розробка інструменту для візуалізації взаємозв'язків між ідентифікованими сутностями у системі, для графічного представлення структури та залежностей: реалізовано підхід, що включає генерацію коду UML на основі аналізу текстових вимог та зіставлення їх з елементами схем баз даних. Цей процес дозволяє автоматично створювати діаграми, які візуально представляють структуру проєкту та залежності між різними сутностями.

Така візуалізація може не тільки сприяти глибшому розумінню взаємозв’язків між компонентами проєкту, але й полегшує процес ідентифікації потенційних проблем та оптимізації робочих процесів. Це стає особливо корисним для проєктних менеджерів і команд розробників, які можуть використовувати ці діаграми для планування, виконання та моніторингу проєктних завдань;

д) апробація результатів дослідження: було проведено детальне тестування та оцінка системи. Апробація включала верифікацію функціональності системи з точки зору точності виявлення та зіставлення сутностей, а також здатності системи обробляти текст на українській та англійській мовах. Було встановлено, що система ефективно впоралася з аналізом текстових даних, демонструючи точність у виявленні та класифікації сутностей, що відповідає встановленим вимогам.

Також було виявлено, що інтеграція візуалізації взаємозв’язків сутностей може підвищувати зрозумілість структурних залежностей між компонентами проєкту. Результати апробації підтвердили, що розроблена система може бути ефективно використана в різних проєктних середовищах, що забезпечує важливий вклад у практику управління проєктами з використанням технологій обробки природної мови.

Треба також зазначити, що одним із ключових аспектів дослідження було впровадження та аналіз процесу інтегрованого аналізу вимог на основі

обробки природної мови. Цей процес включає систематичний підхід до автоматизації виявлення, класифікації та аналізу текстових вимог до проєктів, використовуючи технології обробки природної мови. Він демонструє ефективність аналізу вимог порівняно з традиційними підходами. Завдяки застосуванню обробки природної мови, процес визначення вимог став більш автоматизованим, що дозволяє зменшити ризики, пов'язані з людським фактором і неточностями у традиційних методах.

Цей інтегрований процес сприяє підвищенню якості кінцевих проєктних результатів, забезпечуючи більш ефективне управління проєктами та оптимізацію ресурсів. Впровадження цього новаторського підходу відкриває шлях для подальших досліджень та розвитку в галузі управління проєктами та інженерії вимог, надаючи значний потенціал для майбутньої оптимізації і адаптації процесів вимог у різних сферах IT-індустрії.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

етодичні вказівки щодо розробки та оформлення кваліфікаційної роботи другого (магістерського) рівня вищої освіти за освітньо-науковою програмою «Управління проектами в галузі інформаційних технологій» / Упоряд.: Петров К.Е., Левикін В.М., Чалий С.Ф., Євланов М.В., Міхнов Д.К., Міхнова А.В., Чала О.В. – Харків: ХНУРЕ, 2024. – 24 с.

ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки і техніки. Структура і правила оформлювання. – Чинний від 22.06.2015. – Київ: ДП «УкрНДНЦ», 2016. – 31 с.

СТУ ISO/IEC/IEEE 15288:2016 Інженерія систем і програмного забезпечення. Процеси життєвого циклу систем. [Електронний ресурс] // С

тандарт з управління проєктами та Настанова до зводу знань з управління йроєктами (Настанова PMBOK) – Сьоме видання. – Newton Square, Pennsylvania: Project Management Institute, Inc., 2021. – 130 с. (дата звернення 02.04.2024)

, V. Andrikopoulos, P. Avgeriou, P. Chris Broekema. System and software Дrchitecting harmonization practices in ultra-large-scale systems of systems: A ponfirmatory case study // Information and Software Technology, 150, 2022. № увернення 03.04.2024)

к

aymond S. T. Lee. Natural Language Processing A Textbook with Python Implementation. – United International College Beijing Normal University-Hong Kong Baptist University Zhuhai, China. – 2024. (дата звернення 03.04.2024)

evolutionising Requirements Engineering: Unleashing the Power of NLP and G isualNarrator Tool | [Електронний ресурс] // Сайт «GitHub». – Режим в

е

д

а

доступу: https://github.com/MarcelRobeer/VisualNarrator (дата звернення

uARS: A Tool for Analyzing Requirements | [Електронний ресурс] // Сайт «QUARS». – Режим доступу: https://www.quars.it/, вільний (дата звернення

enning Femmer. Requirements Quality Defect Detection with the Qualicen Requirements Scout. – Technical University Munichand Qualicen, Munich, Germany | [Електронний ресурс]. – Режим доступу: https://ceur-ws.org/Vol-2075/NLP4RE_paper2.pdf, вільний (дата звернення 04.04.2024)

Semantha | [Електронний ресурс] // Сайт «Semantha». – Режим доступу: https://www.semantha.de/semantha-requirements/(дата звернення 04.04.2024)

ReqSuite | [Електронний ресурс] // Сайт «ReqSuite». – Режим доступу: https://www.osseno.com/en/requirements-management-tool/, вільний (дата звернення 04.04.2024)

IBM Engineering Requirements Quality Assistant | [Електронний ресурс] // Сайт «IBM». – Режим доступу: https://www.ibm.com/docs/en/erqa?topic=assistant-overview, вільний (дата звернення 05.04.2024)

14. Defining Terms Used to Describe Requirements | [Електронний ресурс] // Сайт «Learn Microsoft». – Режим доступу: https://learn.microsoft.com/en-us/previous-versions/visualstudio/visual-studio-2015/modeling/model-user-requirements?view=vs-2015&redirectedfrom=MSDN#RequirementsClasses, вільний
15. Candase Hokanson, Karl Wiegers. Software Requirements Essentials: Core Practices for Successful Business Analysis 1st Edition. Addison-Wesley Professional. – 2023. – 208 p. (дата звернення 05.04.2024)
16. Phillip A. Laplante, Mohamad Kassab. Requirements Engineering for Software and Systems (Applied Software Engineering Series) 4th Edition. Taylor & Francis. – 2022. – 404 p. (дата звернення 05.04.2024)

17. Левикін В.М. Паттерни проектування вимог до інформаційної системі: моделювання та застосування [Текст]/В.М. Левикін, М.В. Євланов, М.А. Керносов: монографія. – Харків: ТОВ «Компанія ЗМІТ», 2014. – 320 с. (дата звернення 06.04.2024)
18. Relationships in object models | [Електронний ресурс] // Сайт «Learn Microsoft». – Режим доступу: https://learn.microsoft.com/en-us/ef/core/modeling/relationships, вільний (дата звернення 08.04.2024)
19. Introduction to .NET | [Електронний ресурс] // Сайт «Learn Microsoft». – Режим доступу: https://dotnet.microsoft.com/en-us/platform/support/policy/dotnet-core, вільний
20. Industrial-Strength Natural Language Processing | [Електронний ресурс] // Сайт «SpaCy». – Режим доступу: https://spacy.io/, вільний (дата звернення

21. SpacyDotNet .NET wrapper | [Електронний ресурс] // Сайт «GitHub». – Режим доступу: https://github.com/AMArostegui/SpacyDotNet, вільний
22. Drawing UML with PlantUML| [Електронний ресурс] // Сайт «PlantUML». – Режим доступу: https://plantuml.com/guide, вільний
23. The library for web and native user interfaces | [Електронний ресурс] // Сайт «React». – Режим доступу: https://react.dev/learn, вільний
24. Blueprint React-based UI toolkit for the web | [Електронний ресурс] // Сайт «
25. &OpenAPI Guide» | [Електронний ресурс] // Сайт «Swagger». – Режим доступу: https://swagger.io/docs/specification/about/, вільний (дата звернення

u
e
p
r
i
n
t
J
s
»