НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено

Завідувач кафедри

_____ О.В. Коваль
(підпис) (ініціали, прізвище)

“___”_____2019р.

ДИПЛОМНА РОБОТА

на здобуття ступеня бакалавра

з напряму підготовки 6.050103 “ Програмна інженерія “

на тему “Web-візуалізація онтології предметної області“

Виконав (-ла): студент (-ка) ___4___курсу, групи_ТВ-51_

Кравченко Тетяна Олександрівна _____
(прізвище, ім’я, по батькові) (підпис)

Керівник старший викладач Дацюк О.А. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (вчені ступінь та звання, прізвище, ініціали) (підпис)

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

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

Київ – 2019

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший , бакалаврський

Напрям підготовки 6.050103 “ Програмна інженерія”

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.В. Коваль
(підпис)
”___”_____2019р.

ЗАВДАННЯ
на дипломну роботу студенту
Кравченко Тетяна Олександрівна
(прізвище, ім’я, по батькові)

1. Тема роботи “Web-візуалізація онтології предметної області“

керівник роботи Дацюк Оксана Антонівна, старший викладач
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”__”____ 201__р. № __

2. Строк подання студентом роботи _____

3. Вихідні дані до роботи Для створення системи були використані засоби мов програмування C# та Python, фреймворк ASP.NET MVC та бібліотеки numpy, matplotlib, networkx.

4. Зміст розрахунково-пояснювальної записки (перелік завдань, які потрібно розробити)Проаналізувати засоби візуалізації онтологій предметних областей. Розробити структуру системи. Створити програмний продукт, який дозволяє завантажувати будь-який OWL-файл, проводити навігацію по дереву онтології, підвантажувати документи, які прописані в онтології, будувати граф.

5. Перелік ілюстративного матеріалу
«Архітектура створеної системи», «Діаграма прецедентів», «Функціональна модель системи», «Існуючі програмні рішення», «Алгоритм побудови графу», «Програмні засоби реалізації», «Приклади інтерфейсу користувача», «Висновки»

7. Дата видачі завдання ”_10_”___жовтня_____2018р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Вивчення та аналіз задачі	01.12.2018-31.03.2019	
2	Розробка архітектури та загальної структури системи	01-21.04.2019	
3.	Розробка структур окремих підсистем	22-28.04.2019	
4.	Програмна реалізація системи	29.04-13.05.2019	
5.	Оформлення пояснювальної записки	06.05-01.06.2019	
6.	Захист програмного продукту	25.05.2019	
7.	Передзахист	01.06.2019	
8.	Захист	17.06.2019	

Студент _____ Кравченко Т.О.
(підпис) (прізвище та ініціали,)

Керівник роботи _____ Дацюк О. А.
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Метою роботи було створення програмного засобу для покращення представлення WEB-візуалізації онтології предметної області.

Організація подання даних у форматі, який би був зручним як для програмної обробки, так і для перегляду кінцевими користувачами побудованої для управління ними системи. Веб-додаток, який дає змогу в залежності від типу доступу зберігати, переглядати та завантажувати свої дані в онтологію. Візуалізувати, імплементувати та адаптувати онтологію до вигляду графу.

Записка містить 50 сторінок, 18 рисунків, 1 таблицю та 9 посиланнь.

ANNOTATION

The purpose of the work was to create a software tool for improving the presentation of WEB-visualization of the subject's ontology.

Organization of the presentation of data in a format that would be convenient both for software processing, and for viewing end users of the system built to manage them. A web application that lets you store, view and upload your data to an ontology, depending on the type of content. Visualize, implement, and adapt the ontology to the graph.

The note contains 50 pages, 18 images, 1 tables and 9 references.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

1) OWL (Web Ontology Language) — мова для опису онтологій. Вона дозволяє описувати класи та зв’язки між ними.
2) RDF (Resource Description Framework) — модель для представлення даних (метаданих). Також є лексикою, яка надає набір термінів, які можуть використовуватися для створення загальних/абстрактних описів ресурсів.
3) ASP.NET (Active Server Pages для .NET) — платформа для розробки веб-додатків.
4) API (англ. Application Programming Interface) — прикладний програмний Інтерфейс

ЗМІСТ

ВСТУП

На сьогоднішній день управління знаннями організації стають основою ефективного розвитку і одним з найважливіших факторів конкурентоспроможності. Важливим принципом успішної організації має виступати поширення знань і управління ними для забезпечення значного зростання і розвитку.

Актуальність і важливість проблематики управління знаннями усвідомлюється більшістю фахівців, що займаються корпоративним управлінням і IT-технологіями для цілей управління. Найважливішими елементами (етапами) процесу управління знаннями зізнається: створення, зберігання, пошук, передача (поширення) і використання знань.

На сьогоднішній день формується нова економіка - економіка знань, частиною якої є проблема їх управління. У сучасному світі компанія виробляє не стільки товар або послугу, скільки знання. Це означає, що персонал займається виробництвом знань, інновації стають джерелом новоствореної вартості, організації перетворюються в навчальні. Нехай знання та інтелектуальний капітал (нематеріальні активи компанії) не враховуються в бухгалтерському балансі, тим не менш, вони дають значний економічний ефект, що визначає актуальність теми і логіку даної випускної кваліфікаційної роботи.

В даний час онтології відіграють вирішальну роль при описі знань. Онтологія описує основні концепції предметної області і визначає відносини між ними. Також, онтологія разом з безліччю індивідуальних примірників класів складають базу знань. Для опису онтологій існують різні мови і системи, проте, найбільш перспективним представляється візуальний підхід, що дозволяє безпосередньо візуально створювати онтології, що допомагає наочно сформулювати і пояснити природу і структуру явищ. Візуальні моделі, наприклад, графи мають особливу когнітивну силу, фактично представляючи ресурси когнітивної графіки для структурування інформації. Існує багато програмних пакетів, які можна використовувати як первинний інструмент опису онтологій.

Онтологія визначає загальний словник для вчених, яким потрібно спільно використовувати інформацію в предметної області. Вона включає машинно-інтерпретовані формулювання основних понять предметної області і відносини між ними.

Чому виникає потреба в розробці онтології? Ось деякі причини:

— для спільного використання людьми або програмними агентами загального розуміння структури інформації;

— для можливості повторного використання знань в предметній області;

— для того щоб зробити припущення в предметної області явними;

— для відділення знань в предметній області від оперативних знань;

— для аналізу знань в предметній області.

Спільне використання людьми або програмними агентами загального розуміння структури інформації є однією з найбільш загальних цілей розробки онтологій. Наприклад, нехай, кілька різних веб-сайтів містять інформацію з медицини або надають інформацію про платні медичні послуги, які оплачуються через Інтернет. Якщо ці веб-сайти спільно використовують і публікують одну і ту ж базову онтологію термінів, якими вони всі користуються, то комп'ютерні агенти можуть отримувати інформацію з цих різних сайтів і накопичувати її. Агенти можуть використовувати накопичену інформацію для відповідей на запити користувачів або як вхідні дані для інших додатків.

Забезпечення можливості використання знань предметної області стало однією з рушійних сил недавнього сплеску у вивченні онтологій.

1. Постановка задачі

За багато років існування будь-яких організацій, утворюється безліч документів, великі бази даних, розширюється майно організації, її ресурси тощо. І якщо при створенні організації не було правильно скоректовано її майбутній розвиток – то через декілька років пошук та обробка даних стає досить складною. Знаходити дані у великих базах даних стає все важче, безліч документів зберігаються як в електронних форматах, так і в паперових, і з року в рік цієї інформації стає все більше.

Підходи до організації зберігання та обробки інформації поділяються на два напрямки, де інформація ототожнюється з даними та інформація ототожнюється зі знаннями.

В даний час онтології відіграють вирішальну роль при описі знань. Онтологія описує основні концепції предметної області і визначає відносини між ними. Також, онтологія разом з безліччю індивідуальних примірників класів складають базу знань. Для опису онтологій існують різні мови і системи, проте, найбільш перспективним представляється візуальний підхід, що дозволяє безпосередньо візуально створювати онтології, що допомагає наочно сформулювати і пояснити природу і структуру явищ. Візуальні моделі, наприклад, графи мають особливу когнітивну силу, фактично представляючи ресурси когнітивної графіки для структурування інформації. Існує багато програмних пакетів, які можна використовувати як первинний інструмент опису онтологій.

Тому метою цієї дипломної роботи є організація, класифікація і керування цією великою кількістю даних. Крім того, стоїть задача по оптимізації витрати часу персоналу на пошук, обробку та зміну певних даних. Тому задача включає в себе:

— аналіз предметної області;

— огляд існуючих рішень;

— створення прикладу онтології предметної області;

— створення засобів редагування структури онтології;

— можливість виводити інформацію у зручному вигляді.

Метою розробки є створення програмного додатку, який надає не тільки можливість графічного відображення онтології. А також надає можливість рухатись по дереву онтології, як по системі меню та має можливість працювати з документами, прив’язаними до вузлів дерева.

Така система вимагає від оператора меншої кількості етапів під час введення даних до онтології. Збільшити зручність користування шляхом покращення інтерфейсу та наявності української локалізації.

Призначенням даного програмного засобу є надання зручного інтерфейсу для редагування онтологій інженером з онтологій, як зображено на діаграмі прецедентів (Рисунок 1.1):

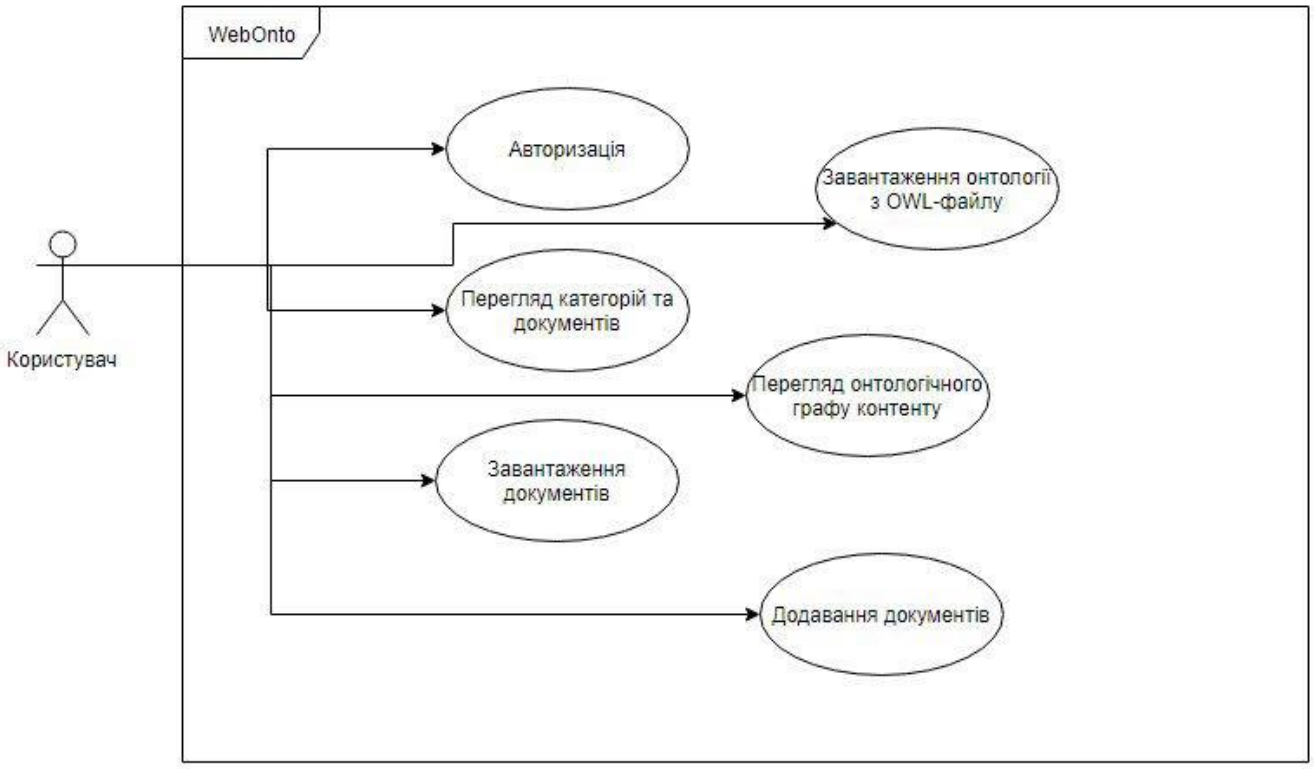

Рисунок 1.1 — Діаграма прецедентів

Так як додавання навіть одного нового елементу онтології вимагає від користувача значної кількості етапів, створення низки елементів для оператора не приносить ніякого задоволення і, навіть, навпаки вимагає багато часу.

1.1 Висновки до розділу

Отже, було розглянуто різні підходи організації знань. Роль онтології в сучасному світі. Принципи роботи онтології, такі як опи знань. Було створено мету даної роботи. Сформовано задачу створення програмного продукту, зобразивши її за допомогою діаграми прецедентів, в якій зображено дії користувача з системою. Дії користувача, такі як проходження авторизації, завантаження файлу онтології з OWL-файлу, перегляд категорій та документів, перегляд онтологічного графу контенту, завантаження та додавання документів.

2. Аналіз проблеми web-візуалізації онтології предметної області

Проблема організації ефективного доступу до інформації завжди відігравала важливу роль для кожної галузі інформатики. Дані у природі зазвичай знаходяться у невпорядкованому стані, однак для них характерні певні закономірності, на основі яких можна побудувати систему класифікації, яка б дозволила полегшити їх організацію у структури даних.

Метою такою організації є подання даних у форматі, який би був зручним як для програмної обробки, так і для перегляду кінцевими користувачами побудованої для управління ними системи.

Пошук оптимального способу класифікації є важливою та складною задачею, вирішення якої залежить від сутності предметної області. Не існує універсальне рішення для будь-якого набору даних.

Створена класифікаційна схема повинна бути інтуїтивно зрозумілою для її безпосередньої візуалізації широким набором спеціалізованих засобів, у тому числі як граф.

Проблема візуалізації у вигляді дерева (різновид графу) може бути легко вирішена тільки для тих схем даних, які мають ієрархічну структуру, тобто містять корінь (у випадку його відсутності, можна створити штучний кореневий каталог, що власне було виконано при побудові системи, яка описується) і не містять циклів.

Таким чином, задача візуалізації зводиться до ефективного використання результатів виконання задач побудови класифікаційної схеми для набору даних, її імплементації та адаптації до вигляду графу.

Для повного розуміння цієї послідовності дій та форматів даних, які при цьому використовуються, необхідно поступово розглянути сутність основних понять онтологічного підходу.

2.1 Поняття онтології предметної області

Під поняттям «онтології» у різних науках розуміють штучну структуру, створену дослідником предметної області для представлення даних про неї у зручному форматі.

З точки зору когнітивної філософії, що вивчає процеси пізнання навколишнього та внутрішнього світу людиною, онтологія є упорядкованою системою, створеною у результаті активної пізнавальної діяльності, яка відображає отримані про природу або більш вузьку предметну область знання у чіткий формат представлення [1].

Знаючи онтологію, можна відтворити основні знання про предметну область, отже вона досить детально описує її складові частини та відношення між ними.

Кожна природня система є динамічною, тому онтологія відображає особливості певного стану предметної області та може потребувати оновлення через певні проміжки часу. Чим більший масштаб системи, тим більшими будуть ці часові проміжки.

Вивчення та розгляд онтології як такої, без орієнтації на можливості її використання, не має сенсу, оскільки це еквівалентно процесу пізнання, у якому результати відкидаються.

Деталі та методи побудови онтологій відрізняються, залежно від конкретних задач, які потрібно вирішити. Можна використовувати як простий опис, що складається з великої кількості послідовних ітерацій (повторень), так і більш складні підходи, в тому числі специфічні для предметної області або науки, яка оброблює отримані результати.

Зазвичай онтології зображуються у вигляді графів (Рисунок 2.1). Граф є математичною структурою, що складається з вершин та ребер, що їх поєднують. Ребра орієнтованого графу мають стрілки в одному зі своїх кінців.

Деревом називають ациклічний (ребра не утворюють циклів) орієнтований граф.

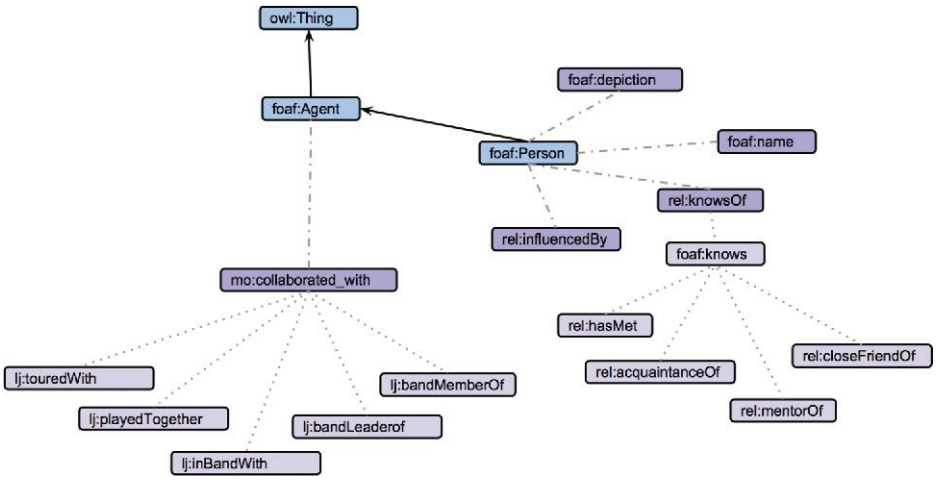

Рисунок 0.1 – Приклад представлення онтології предметної області у вигляді графу з деревоподібною структурою

Зображення класифікаційної системи онтології у вигляді дерева має на меті наочну візуалізацію ієрархічних відношень між її складовими.

У інформатиці під «онтологією» розуміють явний опис множини всіх класів та екземплярів деякої предметної області та відношень між ними, правил їх взаємних перетворень тощо [2].

Відсутність чіткого формулювання, яке було би притаманне конкретному стандарту, означає асбтрактну сутність даного поняття. Однак існує достатньо повне уявлення про склад та методи розробки онтологій для програмних систем [3].

Зазвичай онтологія предметної області складається з

— екземплярів — конкретних або абстрактних одиниць, що представляють найнижчий рівень опису;

— класів — сукупностей екземплярів, які формуються за певними правилами класифікації, що характерні для конкретної предметної області;

— атрибутів — властивостей екземплярів;

— відношень — закономірностей між екземплярами, що поєднують їх у функціональні системи та підсистеми.

Такий формат представлення онтології є найбільш зручним для обробки електронною обчислюваною технікою та відповідає популярним парадигмам предметно-орієнтованого та об’єктно-орієнтованого програмування.

Задача створення онтології є надзвичайно важливою складовою процесу пізнання, оскільки безсистемність останнього може сприяти появі наступних помилок у контексті предметної області:

— не враховано усі важливі екземпляри;

— один і той же екземпляр описано більш ніж раз із різними властивостями;

— враховано занадто багато неважливих з точку зору цілі дослідження екземплярів.

Виникнення цих та інших помилок призводить до появи великої кількості проблем:

— помилки можуть бути своєчасно не виявлені або не виявлені взагалі на етапі розробки, що призведе до серйозних дефектів системи, які впливатимуть на ефективність та успішність використання всієї системи, побудованої з використанням даної онтології;

— необхідно витрачати зайвий час на їх виправлення;

— виникнення стресу серед розробників та занепокоєності, що подібні помилки можуть виникати досить часто та важкі для виявлення та виправлення.

Отже, побудова чіткої та достатньо повної (достатність визначається цілями кінцевої системи) онтологічної схеми для класифікації предметної області є надзвичайно важливою задачею, яку потрібно у повній мірі вирішувати вже на ранніх етапах аналізу та планування.

До створення онтології необхідно активно залучати спеціалістів з предметної області. Класичною помилкою є відділення розробників від таких фахівців із залученням останніх лише на пізніх етапах тестування вже побудованої системи для підтвердження правильності її роботи. Однак виявлення серйозних помилок на кінцевій стадії побудови системи означає дороговартісність її виправлення. Саме тому вкрай важливо грамотно підходити до ранніх етапів аналізу предметної області та використовувати знання про неї кваліфікованих спеціалістів.

Спеціалісти можуть створити онтологію предметної області самостійно, однак при цьому існує можливість ризику, що створена ними онтологіями буде важко інтегруватися до технічної реалізації системи, що створюється. В такому випадку необхідно, щоб розробники та інженери баз даних виправили існуючу онтологію, пристосовуючи її до своїх практичних потреб. При цьому корисною є співпраця зі спеціалістами, оскільки семантика окремих складових онтології може бути сприйнята ними та розробниками системи по-різному.

Отже, створення будь-якої онтології передбачає співпрацю спеціалістів різного фаху, особливо, якщо виникнення помилок у ній буде означати великі ризики фінансових та інших втрат.

Опис онтологій може бути словесним, але обчислювальна техніка потребує свої засоби для обміну, обробки, генерації даних про онтології. Саме тому в останні десятиліття була створена низка стандартів мов для опису онтологій.

Найбільш популярні мови [4]:

— Common Logic,

— DOGMA,

— OIL,

— OWL.

Ці та інші подібні стандарти використовуються для передачі онтологій через мережу Інтернет, її ефективну обробку та редагування машинними засобами, реалізацію уніфікованих алгоритмів візуалізації онтологій у вигляді графів.

Окрім стандартів для мов опису існують також стандарти для форматів, мов запитів, візуалізації онтологічної інформації. Їх спільними особливостями є незалежність від структури конкретної предметної області, що описується, та адаптація до особливостей роботи програмного забезпечення обчислювальних машин з метою підвищення ефективності їх роботи для вирішення різноманітних онтологічних задач.

2.2 Мова опису онтологій OWL

Стандарт OWL (Web Ontology Language) було обрано для використання системою, що була спроектована, як найбільш поширений та простий для сприйняття.

Консорціум World Wide Web рекомендує використання мови OWL для роботи з онтологіями у мережі Інтернет.

Файл онтології на мові OWL має типовий синтаксис.

Кожна онтологія має свою адресу у форматі URL. Наприклад, http://example.org є правильною адресою онтології.

Для опису складових частин отології стандарт OWL використовує універсальну мову документів XML (eXtensible Markup Language).

Формат XML дозволяє створення користувачем своїх схем документів, які складаються з визначених ним елементів-тегів, які, в свою чергу, можуть містити інші елементи або безпосередній текстовий (чи іншого довільного типу) контент. Існує можливість задавати обмеження на вміст елементів, кількість можливих вкладених елементів певного типу.

Для створення та обробки XML-документів існує велика кількість парсерів як загального, так і специфічного призначення.

Все це робить формат XML універсальним для побудови структури документів практично будь-якого рівня складності.

Залежно від рівня деталізації існує 3 підмножини мови OWL (Рисунок 2.2):

— OWL Lite,

— OWL DL,

— OWL Full.

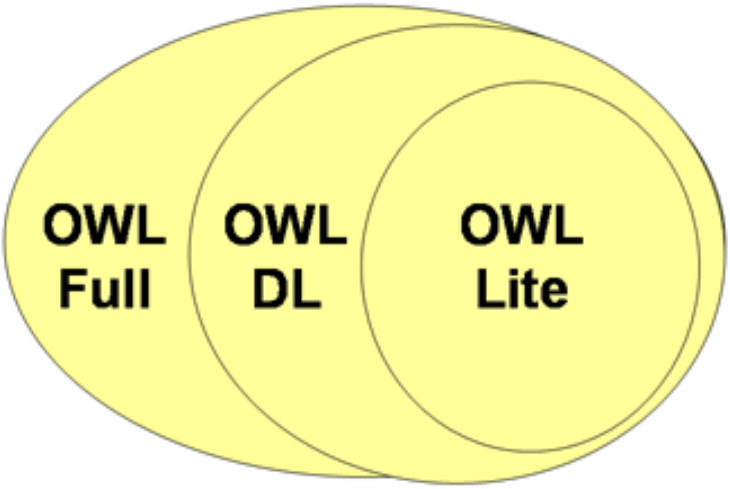

Рисунок 0.2 – Відношення вкладеності між підмножинами мови OWL

Підмножина OWL Lite є функціонально найслабшою та використовується для поверхневого опису предметної області. Вона не передбачає накладання великої кількості обмежень на даних та використовується для вирішення задач простої класифікації.

Підмножина OWL DL полягає у компромісі між обчислювальними можливостями та виразністю мови. Вона є найбільш популярною та розрахована на потреби баз даних сьогодення.

Підмножина OWL Full надає засоби максимальної виразності, однак документи, написані на ній, можуть бути досить важкими для обробки. Повинна використовуватися у крайньому випадку, коли необхідний максимально повний опис предметної області з урахуванням усіх можливих обмежень на дані та спосіб їх представлення.

Мова OWL базується на специфікаціях RDF (Resource Description Framework), розроблених консорціумом World Wide Web для управління метаданими у мережі Інтернет. Ці специфікації часто використовуються у системах баз знань.

Базою знань називають базу даних, що містить інформацію та правила виводу відповідно до знання та накопиченого досвіду стосовно певної предметної області. Поняття «системи бази знань» корелює з формулюванням онтології предметної

області, оскільки остання може виступати у ролі каркасу для наповнення першої змістовною інформацією.

2.3 Мова запитів до онтологій SPARQL

Мова опису онтологій OWL використовує нотацію RDF для опису окремих складових частин онтологічних схем.

Для формату RDF було створено потужну мову запитів до онтологій SPARQL (SPARQL Protocol and RDF Query Language), яка має реалізації для різних мов програмування та може бути відтрансльована до інших мов запитів, наприклад SQL (Structured Query Language) для реляційних баз даних.

Для мови SPARQL існує чотири типи можливих запитів:

— SELECT (отримання необроблених результатів);

— CONSTRUCT (отримання результатів у форматі RDF);

— ASK (отримання відповіді на питання, чи є твердження істинним);

— DESCRIBE (отримання опису RDF-ресурсу).

Ідея всіх запитів засновано на тому, що дані формату RDF розглядаються як триплети, що складаються з інформації про

— Subject (підмет);

— Predicate (предикат або присудок);

— Object (об’єкт).

Існують спеціалізовані адаптації SPARQL для окремих типів предметних областей. Зокрема для GIS-систем використовується GeoSPARQL.

2.4 Аналіз існуючих програмних систем

Особливістю сучасного світу є необхідність розуміння відношень та закономірностей для великих множин даних. Це є неможливим без використання

програмних засобів для автоматизації процесу побудови та візуалізації онтологій різноманітних предметних областей.

Існують як старі, так і новітні рішення, що мають свої недоліки та переваги. Більшість з них підтримує лише англійську мову інтерфейсу.

2.4.1 Редактор онтологій WebProtege

Командою дослідників Стенфордського університету (Каліфорнія, США) було розроблено редактор онтологій Protégé та його web-інтерфейс.

Засіб є безкоштовним для використання, але потребує реєстрацію та створення персонального аккаунту для використання.

Інтерфейс користувача зображено на Рисунку 2.3.

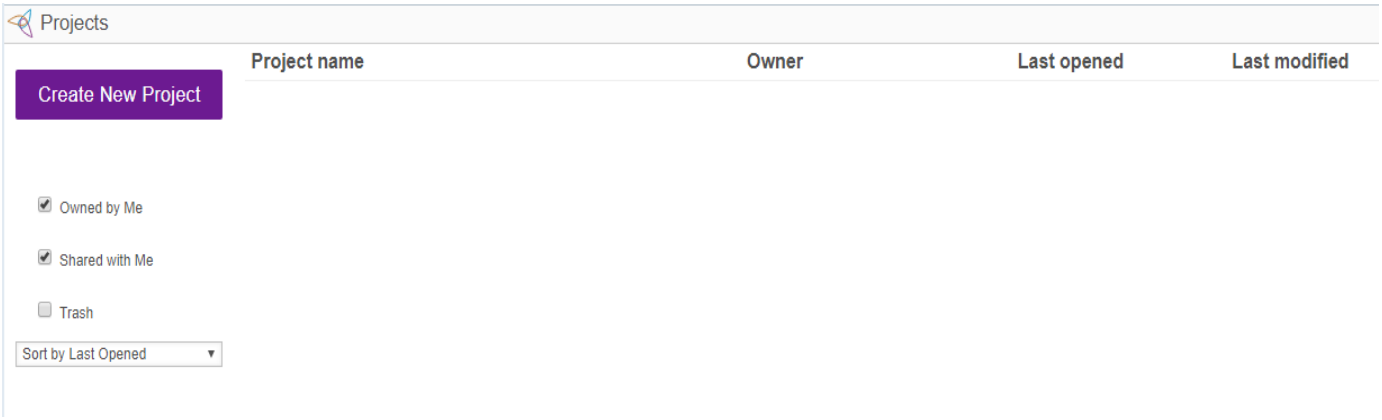

Рисунок 0.3 – Інтерфейс WebProtege

Використання сучасних технологій робить даний застосунок якісним рішенням для роботи з онтологіями, є можливість імпорту та експорту онтологій у різних форматах, в тому числі RDF.

Є можливість побудови онтологічного графу та виконання SPARQL-запитів, що робить даний застосунок популярним при вирішення класичних онтологічних задач.

Підтримується лише англійська мова інтерфейсу. В порівнянні з десктопною версією застосунку Protégé функціонал web-інтерфейсу є дещо меншим, немає можливості встановлення користувацьких плагінів (доповнень).

2.4.2 Формат візуалізації онтологій VOWL

Амбітним проектом галузі семантичного Інтернету є формат візуалізації онтологій VOWL. До проекту входить велика кількість рішень, які споріднює механізм відображення онтологій у вигляді графів.

Приклад такого графу зображено на Рисунку 2.4.

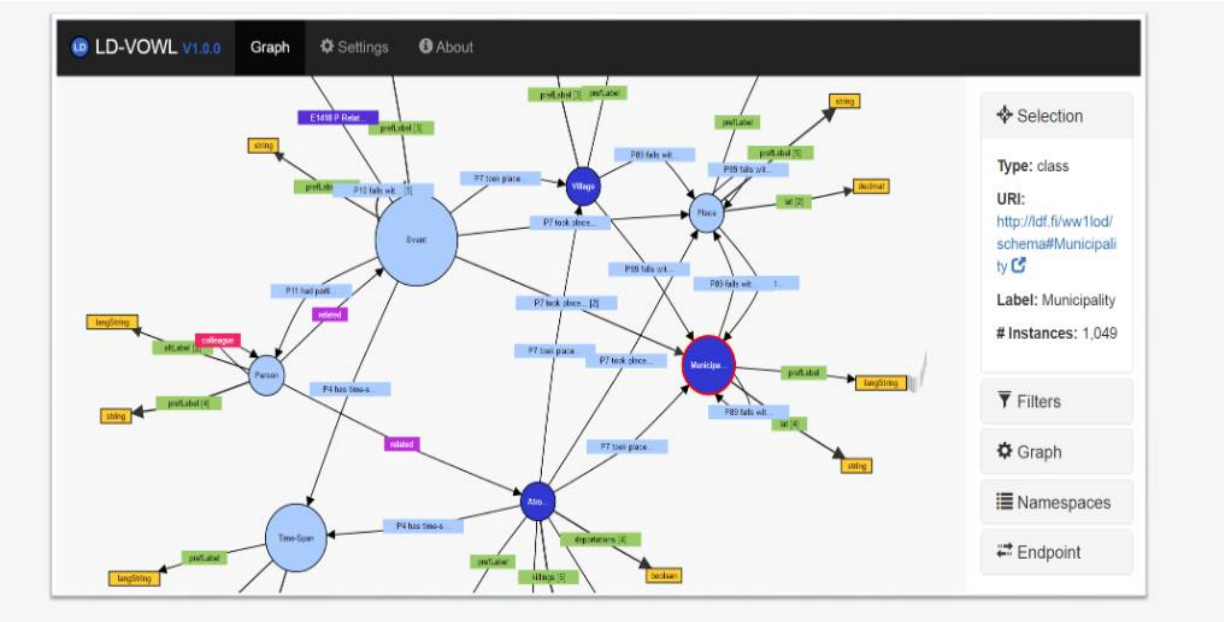

Рисунок 0.4 — Граф візуалізації онтології з використанням засобів VOWL

Інформація для графу отримується з API (Applied Programming Interface), адресу якого надає користувач. Відсутня необхідність створення аккаунту та авторизації. Наявний широкий функціонал для редагування зовнішнього вигляду та структури побудови онтологічного графу.

До недоліків даних засобів можна віднести складний інтерфейс використання, певна незавершеність реалізації (проект знаходиться у стадії розробки). Засоби візуалізації відділені від стандартних операцій роботи з онтологіями, що унеможливлює їх самостійне використання та цим ускладнює процес управління онтологією предметної області. Інтерфейс доступний тільки для англійської мови. Перед початком використання користувач повинен пройти короткий курс навчання роботи з технологією, що потребує виділення додаткового часу.

Вище були розглянуті найпопулярніші додатки, але є і інші, які також користуються популярністю: Apollo, Cmap Tools/COE, DL-Workbench, Ontolingua, OntoStudio та інші.

2.5 Висновки до розділу

Отже, було описано проблеми web-візуалізації онтології предметної області та існуючі програмні рішення для представлення онтології.

3. ЗАСОБИ РОЗРОБКИ ПРОГРАМНОГО ПРОДУКТУ

Створення нового програмного забезпечення у сучасному світі є неможливим без широкого залучення існуючих програмних рішень.

Засоби розробки, що були використані для написання програмного коду та налаштування проекту системи, можна поділити за їх функціональним призначенням та природою на декілька категорій:

- мови програмування (C#, Python),
- мови розмітки (HTML, CSS),
- фреймворки (.NET MVC, OwlDotNetAPI),
- середовища розробки (Visual Studio).

3.1 Мова програмування C# та фреймворки для роботи з нею

Для реалізації переважної частини модулів системи було обрано мову програмування C#[5].

Мова C# відрізняється широким спектром застосування та підтримкою великої кількості парадигм програмування, що робити її використання ефективним для великих та малих проектів із залученням бібліотек, створених іншими користувачами, різних технологій, інших мов та систем контролю версій.

Нововведенням C# стала можливість легшої взаємодії, порівняно з мовами-попередниками, з кодом програм, написаних на інших мовах, що є важливим при створенні великих проектів. Якщо програми на різних мовах виконуються на платформі .NET, .NET бере на себе клопіт щодо сумісності програм (тобто типів даних, за кінцевим рахунком).

Станом на сьогодні C# визначено флагманською мовою корпорації Microsoft, бо вона найповніше використовує нові можливості .NET. Решта мов програмування,

хоч і підтримуються, але визнані такими, що мають спадкові прогалини щодо використання .NET.

Присутність або відсутність тих або інших виразних особливостей мови диктується тим, чи може конкретна мовна особливість бути трансльована у відповідні конструкції CLR. Так, з розвитком CLR від версії 1.1 до 2.0 значно збагатився і сам C#; подібної взаємодії слід чекати і надалі. (Проте ця закономірність буде порушена з виходом C# 3.0, що є розширеннями мови, що не спираються на розширення платформи .NET.) CLR надає C#, як і всім іншим .NET-орієнтованим мовам, багато можливостей, яких позбавлені «класичні» мови програмування. Наприклад, збірка сміття не реалізована в самому C#, а проводиться CLR для програм, написаних на C# точно так, як і це робиться для програм на VB.NET, J# тощо.

Одиницею функціоналу мови є клас, який згідно основних принципів об’єктно-орієнтованого програмування, інкапсулює дані (сутність) та методи роботи з ними (поведінку).

Класи належать до просторів імен, в межах яких повинні мати унікальну назву. Простори імен зазвичай ототожнюються з бібліотеками, які можуть бути імпортовані до різних проектів. Для імпорту використовуються менеджери пакетів, наприклад NuGet (Рисунок 3.1).

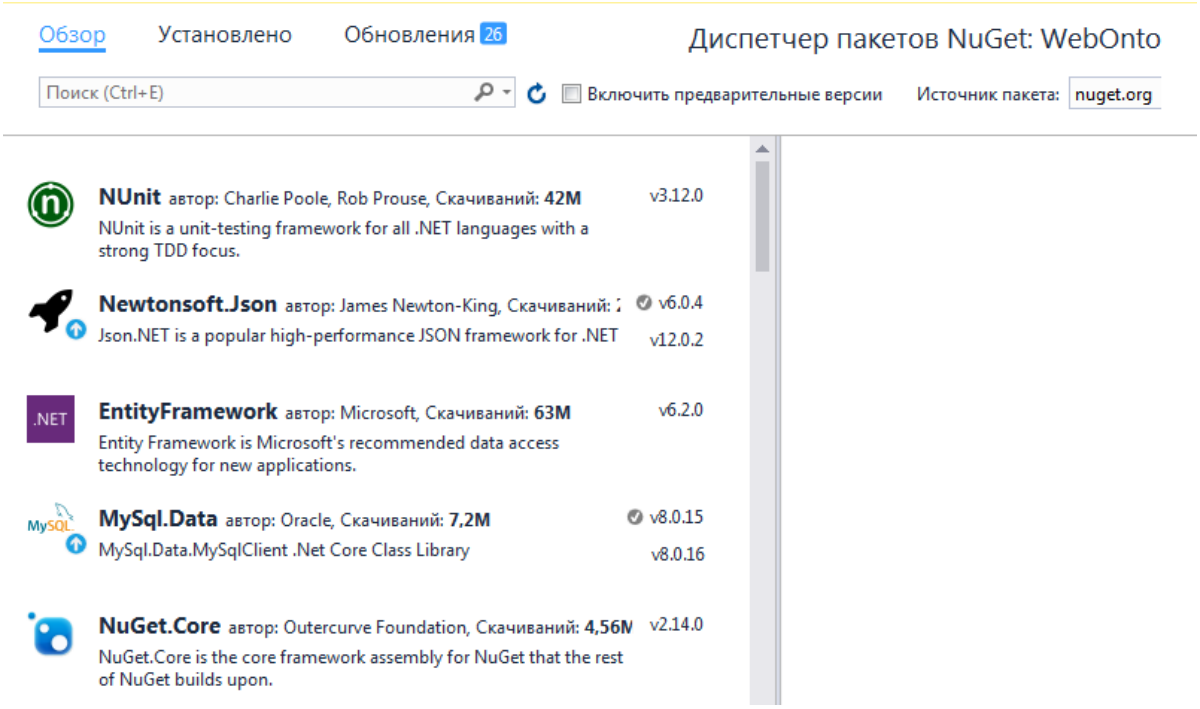

Рисунок 3.1 — Інтерфейс менеджеру пакетів NuGet

Рішення ASP .NET є потужним фреймворком, який часто використовується для побудови web-застосунків. У якості серверної мови може використовуватися Visual Basic або C#. Всі мови, що підтримуються, використовують спільний набір загальних бібліотек, які ефективно реалізовані для досягнення найкращих показників при виконанні програм, побудованих з їх використанням.

Створена система орієнтована на web-інтерфейс. Саме тому було обрано ASP .NET MVC фреймворк, основними принципами якого є

- використання шару моделі для інкапсуляції сервісів, які працюють з чистими даними;
- залучення логіки шаблонів у шарі представлення для відображенні результатів роботи моделі та функціональних запитів до користувача;
- поєднання моделі та представлення у контролері, який оброблює HTTP-запити користувача.

Фреймворк також підтримує велику кількість різних технологій, у тому числі використання HTML, CSS та JavaScript у шаблонах шару представлення.

Для обробки OWL-файлів було використано ефективне готове рішення від Microsoft, що має назву OwlDotNetAPI[6].

Дана бібліотека, яку можна завантажити через менеджер пакетів NuGet у складі проекту dotNetRDF (Рисунок 3.2), має методи для автоматичного парсингу онтологій у різних форматах, причому результати подаються у вигляді графу, для якого можна легко будувати та виконувати SPARQL-запити[7].

Засоби бібліотеки дозволяють виконувати ітерацію по вершинах графу, обробляючи їх та трансформуючи у внутрішнє представлення конкретної програмної системи.

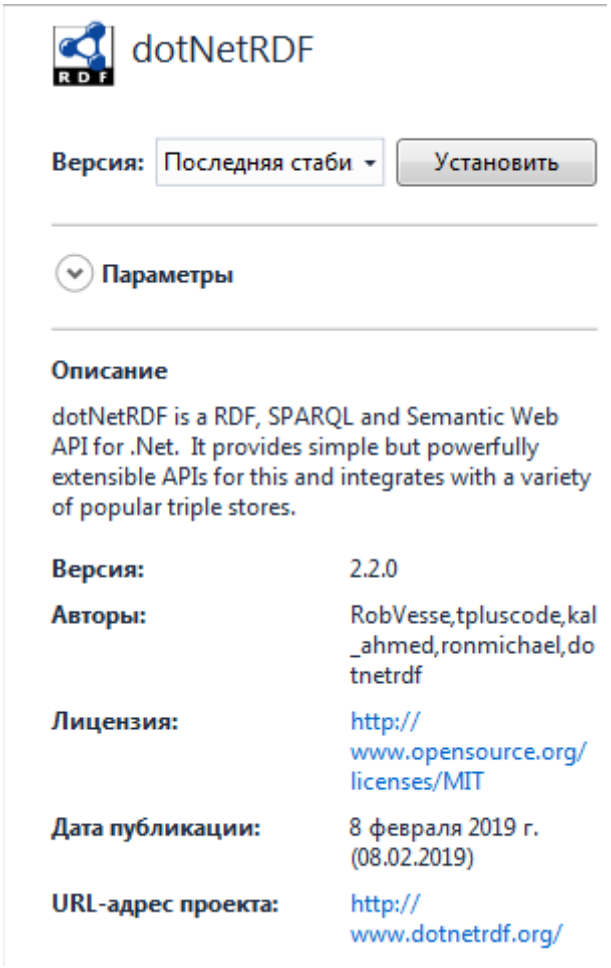

Рисунок 3.2 — Опис проекту dotNetRDF у вікну менеджеру пакетів NuGet

Отже, засоби OwlDotNetAPI були використані для роботи з файлами формату OWL, оскільки вони реалізують готові та ефективні рішення стандартних задач роботи з онтологіями у вигляді графів.

3.2 Мови розмітки HTML та CSS

Однією з головних завдань "всесвітньої павутини" є надання інформації в найкоротші терміни. Тому створення Web-сторінок і розміщення їх в мережі InterNet не тільки ознайомить користувачів з потрібною їм інформацією, але і послужить рекламою для діяльності Вашої компанії, що в кінцевому результаті зможе принести прибуток і збільшити власний заробіток, причому без особливих витрат.

Максимально в стислі терміни для створення Web-проектів підходить лише мова гіпертекстової розмітки HTML, який дозволяє оформляти наповнення Web-сторінки і контролювати її зовнішній вигляд з вікна абсолютно будь-якого браузера.

Базовим функціональним елементом кожного web-інтерфейсу є сторінка. Для створення зовнішнього вигляду Інтернет-сторінки існують стандарти HTML5 та CSS3.

Мова розмітки HTML5 використовується для розміщення елементів інтерфейсу на сторінці. Кожен елемент задається парою тегів (відкриваючим та закриваючим). Все, що знаходиться між тегами, є контентом елементу. Для відкриваючого тегу є можливість задавати атрибути, однак ознакою зручного та естетично збалансованого вигляду сторінки є використання таблиць стилів.

До елементів можна звертатися, вказуючи їх ідентифікатори. Іншим способом їх групування є визначення класів. Якщо один і той самий ідентифікатор може мати лише один елемент, то одного класу може належати багато елементів.

Отже, мова HTML була обрана для опису сторінок системи, оскільки вона має ряд позитивних якостей:

- простота та надзвичайна виразність у використанні;
- широкі можливості забезпечення доступу до окремих елементів та створення класів елементів, що повинні оброблятися однаковим чином;
- високий рівень стандартизації;
- популярність серед web-розробників, що викликало появи великої кількості стандартних шаблонів.

Каскадні таблиці стилів CSS дозволяють зручно задавати графічні атрибути як для окремих елементів сторінок, так і для їх груп (класів, елементів одного типу тегу).

Використання HTML та CSS надає змогу швидкого створення дизайну web-інтерфейсу. Існує можливість зміни вигляду сторінки у залежності від фізичних розмірів екранного пристрою кінцевого користувача.

Таким чином, сучасні потужні рішення web-дизайну дозволяють легко розробити зовнішній вигляд сторінок та заощадити час на реалізацію технічних деталей їх взаємодії з користувачами та бізнес-логіки системи.

3.3 Мова програмування Python та модуль network

Для побудови онтологічного графу контенту було використано можливості мови програмування Python.

Існуючі рішення побудови графів на мові C# можна охарактеризувати як складні для імплементації та орієнтовані більше на модель, ніж на представлення. В той же час, мова програмування Python містить зручні готові рішення, які дозволяють побудувати зображення графу відповідно до вхідних даних та зберегти його у довільному форматі.

Мова програмування Python часто використовується для створення невеликих скриптових програм, що виконують обробку або аналіз даних.

Існує велика кількість модулів для мови Python, написаних іншими розробниками. Для побудови онтологічного графу були використані бібліотеки

- numpy;
- matplotlib;
- networkx.

Модуль numpy призначений для швидкої обробки великих масивів даних. Він написаний на мові C з використанням ефективних технологій, у тому числі

паралельної обробки даних. Використання даного модулю дає відчутні переваги по відношенню до класичних структур даних, що наявні у мові програмування Python.

Модуль matplotlib є аналогом стандартної бібліотеки tkinter, але надає більш широкі можливості графічного зображення геометричних фігур, діаграм та графиків. Існує можливість збереження результатів до файлів різних форматів.

Модуль networkx було створено як універсальне рішення для побудови мережевих графів. Вершини графу можуть мати різні стратегії щодо автоматичного розміщення. Також існує можливість створення власного розміщення, окремо для вершин, ребер та міток (назв вершин).

Використання засобів мови програмування Python спростило побудову онтологічного графу інформаційних ресурсів.

3.4 Середовище розробки Visual Studio

Оскільки переважна частина програмного коду системи написана на мові програмування C# із використанням фреймворку .NET найкращим вибором середовища розробки став програмний продукт Visual Studio.

Під час розробки даного програмного забезпечення було використано версію редакції Enterprise. До складу цієї версії входить модуль візуального проектування GUI-інтерфейсу Swing UI Designer, XML-редактор, редактор регулярних виразів, система перевірки коректності коду, система контролю за виконанням завдань і доповнення для імпорту та експорту проектів з Eclipse. Доступні засоби інтеграції з системами відстеження помилок JIRA, Trac, Redmine, Pivotal Tracker, GitHub, YouTrack, Lighthouse.

Особливості Visual Studio 2017:

- покращена інтеграція з хмарною технологією Azur. Розробки Microsoft в цьому напрямку дозволяють полегшити створення, налагодження, розміщення і публікацію ваших додатків в хмарі Azure прямо з IDE, надаючи

до того ж вбудовані інструменти для роботи цими додатками, а також з Docker-контейнерами, .NET Core додатками і так далі;

- розробка для мобільних пристроїв. Розробники отримали поліпшені інструменти налагодження і профілювання, інструменти генерації модульних тестів. Також з’явилася можливість створювати кросплатформені додатки;
- офіційно доступна нова версія Visual studio Team Foundation Server 2017. У цей випуск on-premise платформи для організації спільної роботи команд включили давно очікувані можливості, наприклад, нові шаблони процесів, поліпшене керування доступом до репозиторіїв, pull-реквестамі і багато іншого;
- розробники отримують доступ до додаткових сервісів для оптимізації і створення DevOps циклу всередині своєї організації, таким як хмарний CI-сервер, інструменти навантажувального тестування в хмарі і навіть персонального DevOps навчання;
- випуск нового інструментарію, доступного .NET Core в складі Visual Studio 2017.
- У CLI додалися додаткові команди і можливість вибору власних шаблонів проекту. Також був анонсований приклад реалізації мікросервісной архітектури, який ви можете знайти в репозиторії GitHub;
- Нові можливості торкнулися структури проекту, заснованої на .csproj, що забезпечує сумісність з build-системами для .NET, заснованими на MSBuild. Додатково, формат.csproj значно спрощує розробникам можливості редагування файли для оголошення залежностей, target-платформ і властивостей проекту.

Середовище Visual Studio (Рисунок 3.3) створене для легкої розробки проектів на мовах, що підтримуються .NET[8], та надає потужний функціонал із різноманітних засобів, у тому числі

— вбудований сервер для локального запуску web-проектів;

— засоби відлагодження помилок;

— аналіз ефективності коду;

— підсвітку синтаксису та механізм автодоповнення для всіх мов, що підримуються.

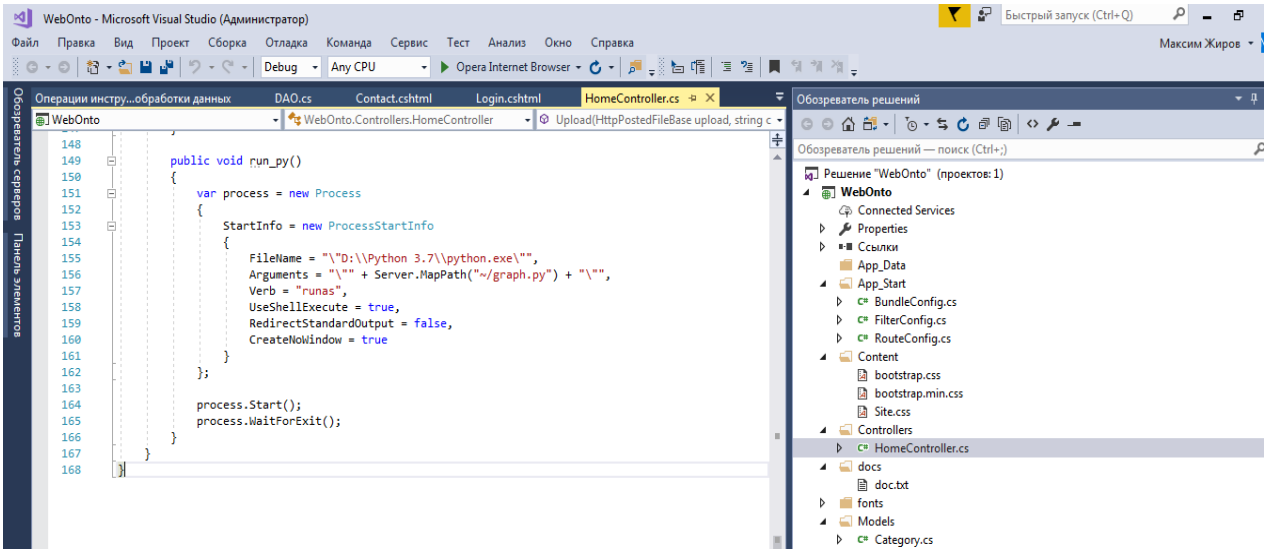

Рисунок 3.3 — Інтерфейс Visual Studio 2017

Таким чином, середовище Visual Studio є найкращим серед аналогів для створення складних у налаштуванні web-проектів на мові програмування C#[9].

3.5 Висновки до розділу

В розділі було описано основні програмні засоби розробки, такі як мови програмування, мови розмітки, фреймворки та середовища розробки, що були використані для написання програмного коду та налаштування проекту системи, зазначено їх аналоги та переваги. Засоби розробки є популярними та актуальними в наш час. Для реалізації переважної частини модулів системи було обрано мову програмування C#. Середовищем розробки було обрано Visual Studio, так як вона

найкраще підходить для реалізації проектів, написаних мовою C#. Для обробки OWL-файлів було використано ефективне готове рішення OwlDotNetApi. Для зовнішнього вигляду Інтернет-сторінки використовувалися мови розмітки HTML5 та CSS3. Для побудови онтологічного графу були використані бібліотеки numpy, matplotlib, networkx.

4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Програмна система була побудована як комплекс, що складається з декількох модулів. Кожний модуль має свою архітектуру та використовується для реалізації чітко визначеного переліку функцій. Окремі модулі взаємодіють між собою за допомогою спеціалізованих інтерфейсів.

Загальний компоненти системи показано на Рисунок 4.1.

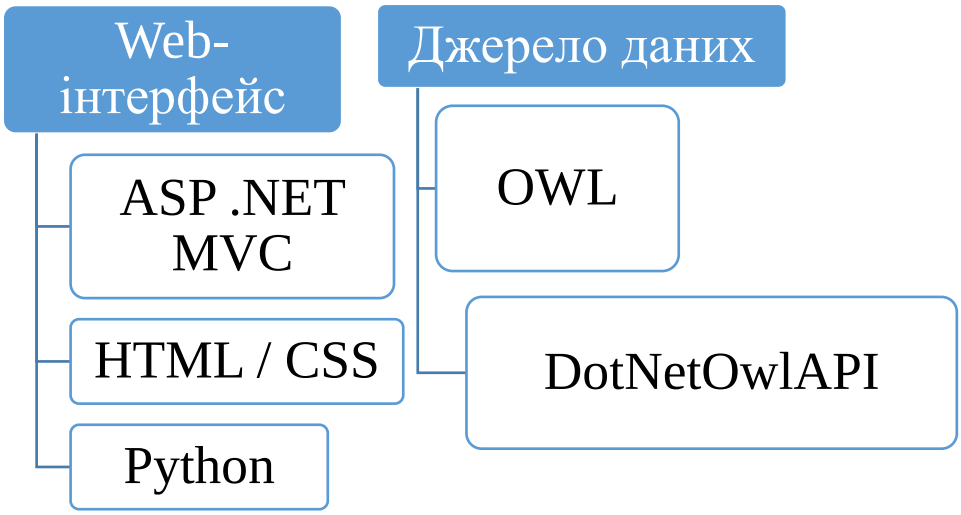

Рисунок 4.1 — Компоненти створеної системи

Для створення кожного з модулів були використані технології, що не залежать від конкретної операційної системи чи апаратної платформи, тому вони можуть виконуватися для практично будь-якої з них.

Модулі обробки джерела даних, взаємодії з OWL-файлами та більшість програмного коду web-інтерфейсу були написані на мові програмування C#.

Найбільш складну структуру має частина реалізації web-інтерфейсу. Це обумовлено широким спектром задач, які вона покликана вирішувати.

Головні відношення між окремими компонентами програмної системи показано на Рисунку 4.2.

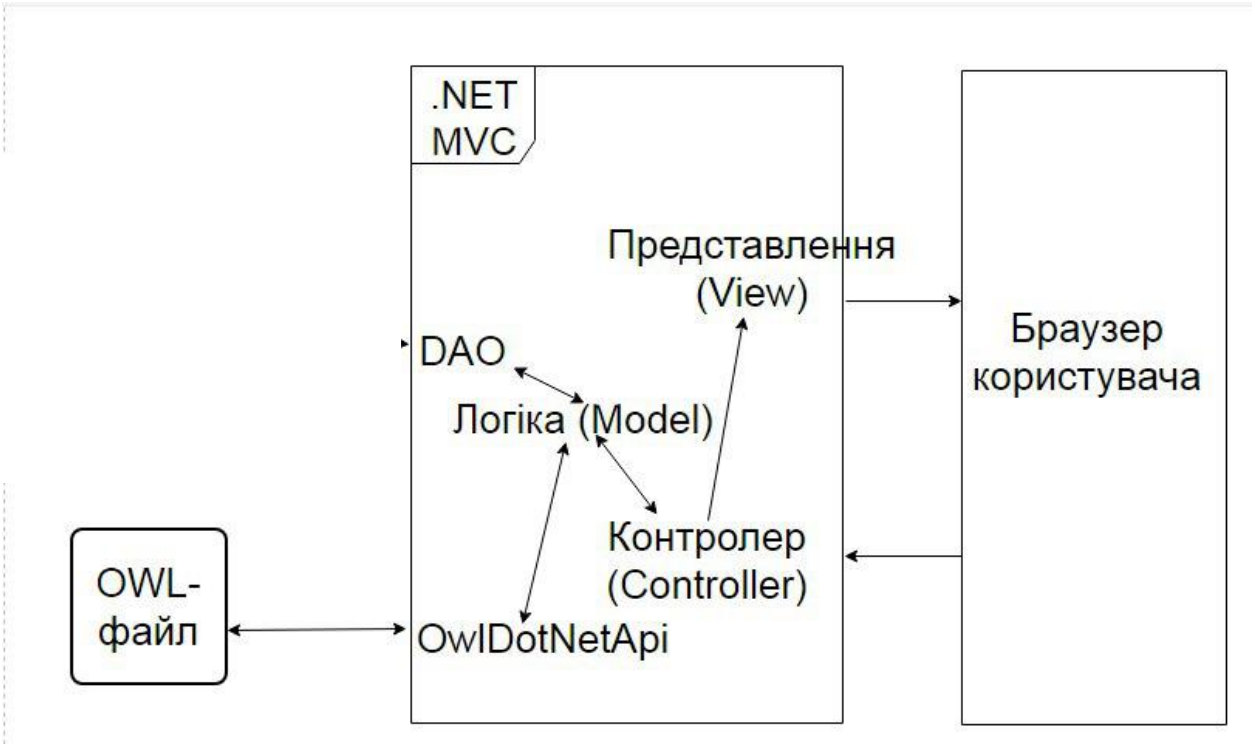

Рисунок 4.2 — Архітектура комунікації компонентів системи.

Структуру системи також можна зобразити відповідно до сценарію дій користувача, які ініціюють її відклики та зміни внутрішнього стану (Рисунок 4.3).

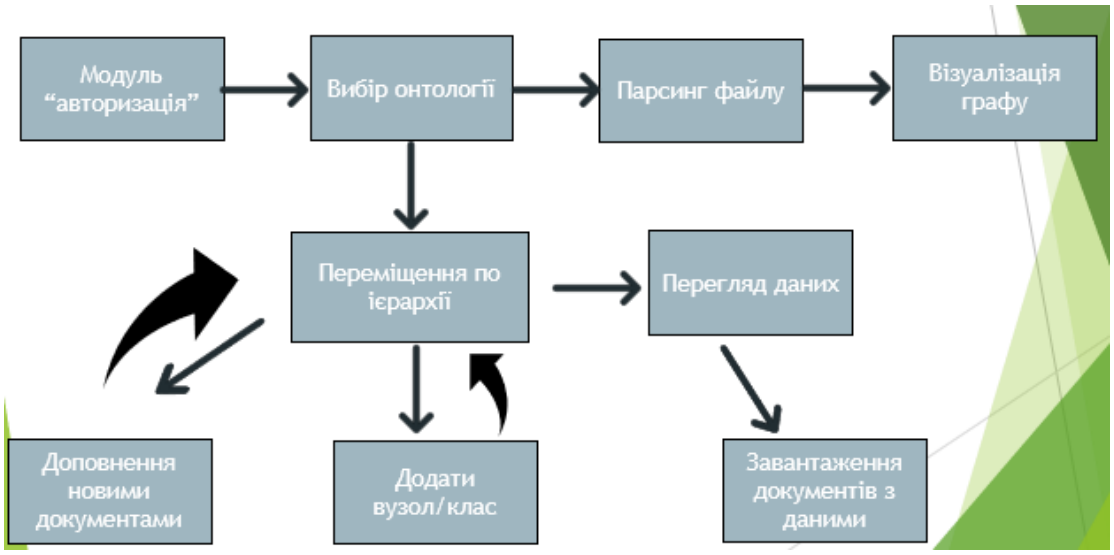

Рисунок 4.3 — Функціональна структура системи

В залежності від вибору користувача, можна асинхронно рухатися між блоками, вибираючи потрібну дію.

4.1 Структура модуля обробки OWL-файлу

У якості прикладу роботи системи було вирішено працювати з онтологією інформаційних ресурсів школи.

Для обраної предметної області під інформаційними ресурсами розуміють організовану сукупність інформації, інформаційних продуктів, призначених для інформаційного забезпечення роботи школи.

Приклад онтології «Школа» подано на Рисунку 4.4. Особливістю використання такої онтології є те, що крім класифікації інформаційних ресурсів потрібно зберігати посилання на самі файли з ресурсами та мати можливість завантаження цих файлів.

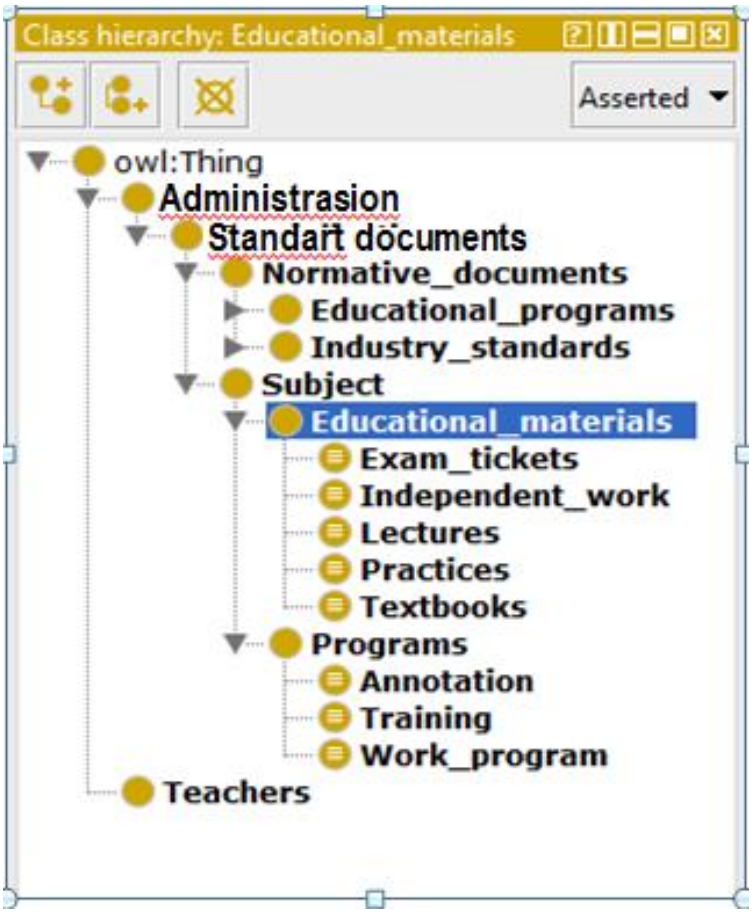

Рисунок 4.4 — Спрощене дерево онтології «Інформаційні ресурси Школи»

Система отримує дані про інформаційні ресурси у вигляді зовнішнього файлу формату OWL.

При завантаженні файлу онтології створюється новий екземпляр класу Graph, дані для якого зчитуються екземпляром службового класу OpenFileDialog та форматуються відповідним чином.

Клас Graph має метод LoadFromFile(filePath: String), який реалізує парсинг OWL-файлів (та подібних форматів онтологій). Даний метод було використано у програмній системі для заповнення графу та збереження його наповнення у вигляді колекції класів Category та Document без втрати ієрархічної структури.

Дані про атрибути та відношення онтології ігноруються, оскільки не відіграють важливої ролі у контексті переліку інформаційних ресурсів.

Клас Document містить конструктор за замовчуванням та наступні поля, реалізовані як властивості мови програмування C#:

— public string ID { get; set; }, унікальний ідентифікатор документу;

— public string Name { get; set; }, назва документу;

— public string Content { get; set; }, зміст (власне текст) документу;

— public Category Category { get; set; }, категорія, до якої даний документ належить.

Клас Category містить конструктор за замовчуванням, в якому відбувається ініціалізація списків, та наступні поля, реалізовані як властивості мови C#:

— public string ID { get; set; }, унікальний ідентифікатор категорії;

— public string Name { get; set; }, назва категорії;

— public List<Category> Supercategories { get; set; }, перелік надкатегорій для конкретної категорії;

— public List<Category> Subcategories { get; set; }, перелік підкатегорій цієї категорії;

— public List<Document> Documents { get; set; }, перелік документів категорії.

Інформація про онтологічну схему конвертується з формату OWL до внутрішнього представлення системи у вигляді структур класів Category та Document. Для цього у програмному модулі інтерфейсу користувача відбувається ітеративне проходження через всі точки завантаженого з OWL-файлу графу. З назви класів та документів видаляється назва онтологічної схеми, після чого відповідна інформація записується до внутрішньої структури категорій та документів у моделі.

4.2 Структура web-інтерфейсу

Web-інтерфейс системи складається з чіткого набору директорій, в кожній з яких знаходяться файли, що виконують подібні завдання та мають схожу структуру:

— Views (шаблони сторінок);

— App_Start (інформація про розгортання системи на сервері);

— bin (динамічні бібліотеки, створені при компіляції проекту);

— Models (класи бізнес-логіки);

— Controllers (контролери);

— Scripts (код на мові Python для побудови онтологічного графу контенту).

Шаблони сторінок написані на мовах HTML, CSS та C# та утворюють безпосередній інтерфейс для кінцевого користувача, приймаючи вхідні дані, відображаючи їх та сприймаючи відповіді.

Сайт системи містить 3 сторінки:

— головну (Index),

— авторизації (Login),

— відображення онтологічного графу контенту (Graph).

Модель складається з класів, які реалізують бізнес-логіку системи та не залежать від деталей реалізації графічного інтерфейсу. До пакету моделей застосунку належать раніше описані класи Category, Document, DAO та клас побудови структури даних для відображення у вигляді графу OntoGraph.

Система містить лише один контролер HomeController, відповідальний за обробку усіх HTTP-запитів, які є дозволеними для користувача. Всі можливі запити подано у Таблиці 4.1.

Таблиця 4.1 — Обробка HTTP-запитів у контролері системи.

Шлях	HTTP-метод доступу	Аргументи методу контролера
Index	GET	String id
Index	POST	String id, String docname
Search	GET	
Login	GET	String login, String pass
Login	POST	
Upload	POST	HttpPostedFileBase upload, String category
Graph	GET	

Код на мові програмування Python знаходиться у скриптовому файлі graph.py.

Виконання даного коду ініціюється за умови GET-запиту за шляхом Graph у приватному методі контролера run_py() з використанням стандартної методики виклику командного рядка у програмах на мові C#.

Отже, web-інтерфейс створеної системи має достатньо насичену структуру, яка відображає функціонал, доступний кінцевому користувачу.

4.3 Завантаження нових документів на сервер

Запит методу POST для шляху UPLOAD ініціює процедуру збереження документу, який було завантажено на сторінку.

Процедура обробки файлу відбувається у 2 етапи.

Спочатку файл користувача записується до спеціальної директорії серверу. При цьому зберігаються його назва та зміст.

На другому етапі для сутності поточної категорії створюється новий документ. Потім відбувається зчитування змісту нещодавно завантаженого файлу з метою збереження інформації про нього у створеному екземплярі класу Document.

Онтологічний граф при оновленні буде містити на один вузол більше, що дозволить користувачеві пересвідчитися в успішному збереженні нового файлу у онтологічній схемі.

4.4 Побудова онтологічного графу контенту

Дані для графу отримуються з таблиць бази даних. Безпосередню їх візуалізацію забезпечує статичний метод Draw() класу Model.OntoGraph(). Пари вершин, що сполучаються ребрами графу, записуються до текстового файлу graph.txt відповідно до таблиці ієрархічних відношень. При виконанні GET-запиту до сторінки графу запускається на виконання скриптовий файл graph.py, який парсить зазначений текстовий файл та заносить результату до списку кортежей (пар), які разом з інформацією про розміщення вершин, ребер та міток у візуалізації структури передаються до об'єкту класу орієнтованого графу бібліотеки networkx (Рисунок 4.4).

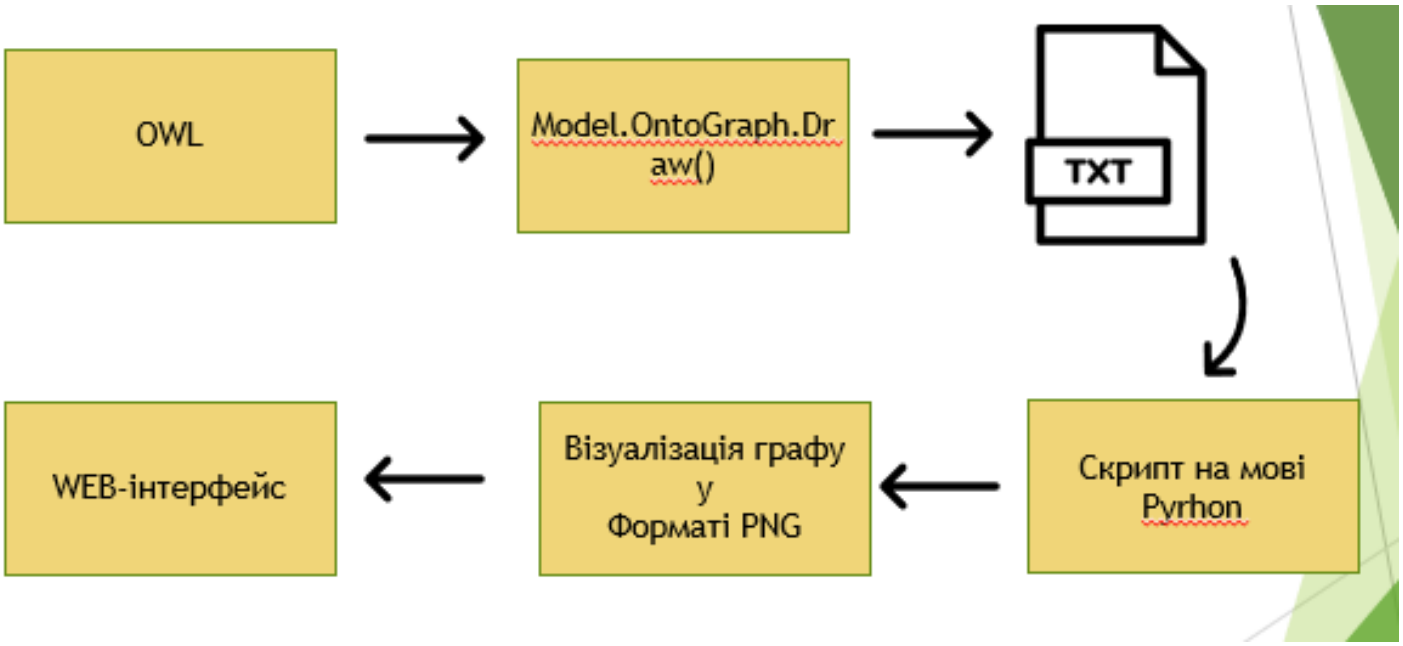

Рисунок 4.4 – Алгоритм побудови графу засобами Python.

Безпосередня візуалізація автоматично побудованої структури виконується засобами бібліотеки matplotlib. Результат зберігаються у файлі graph.png, який відображається на сторінці web-інтерфейсу. Отже, динамічна побудова візуалізації онтологічного графу контенту системи відбувається кожен раз при переході кінцевого користувача на відповідну сторінку.

4.5 Висновки до розділу

Отже, в даному розділі було описано основні принципи побудови програмної системи та конкретні приклади створення її елементів, акцентуючи увагу на шаблонах проектування.

5. Робота користувача з програмною системою

5.1. Системні вимоги та інсталяція

Для запуску системи на сервері необхідно переконатися у виконанні наступних вимог:

— мінімум 2 гігабайти оперативної пам’яті;

— наявність встановленої бібліотеки .NET версії 4.0 або новіше.

Розробка і тестування програмного комплексу відбувалися у операційній системі Windows, однак створена система може бути розгорнута також на системі Linux та інших Unix-орієнтованих операційних системах за умови наявності встановленої бібліотеки .NET.

Локальне розгортання системи для відлагодження можливе також за допомогою вбудованого сервера середовища розробки Visual Studio.

5.2. Сценарій роботи користувача з системою

Сценарій роботи користувача з системою доволі простий. Непотрібно знати засобів обробки інформації. Потрібно просто запустити програму. Виконавчий файл проекту має назву WebOnto.sln. Користувач запускає цей файл . Після запуску цього файлу користувач побачить головну форму додатку (Рисунок 5.1):

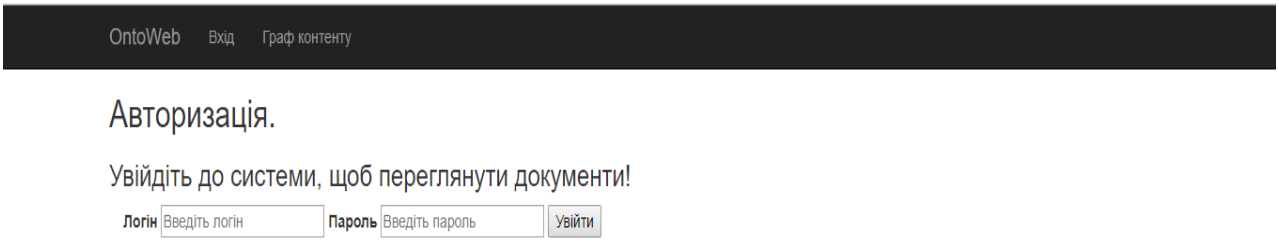

Рисунок 5.1 — Сторінка авторизації користувача

В меню авторизації, користувач повинен вести логін та пароль та клікнути на кнопку “Увійти”. У системі є три ролі: адміністратор (методист), вчитель та учень. В залежності від типу користувача є різні права доступу. Наприклад, для учня прихований розділ “Inner Documents”, для вчителя — “Studying Programs”, для адміністратора доступні всі розділи.

Після вводу даних користувач автоматично попадає на список категорій документів (Рисунок 5.2):

Рисунок 5.2 — Сторінка списку каталогів

В кожному каталозі по принципу ієрархії можуть бути розміщенні інші каталоги.

Далі користувач може переглядати усі категорії документів, які доступні для нього. У кожному каталозі може буде вкладений інший каталог по правилам вкладення. На прикладі “Studying Results” можна побачити файли, які зберігаються в даному розділі. Користувачу надана можливість додавання файлів до відповідного каталогу (Рисунок 4.3):

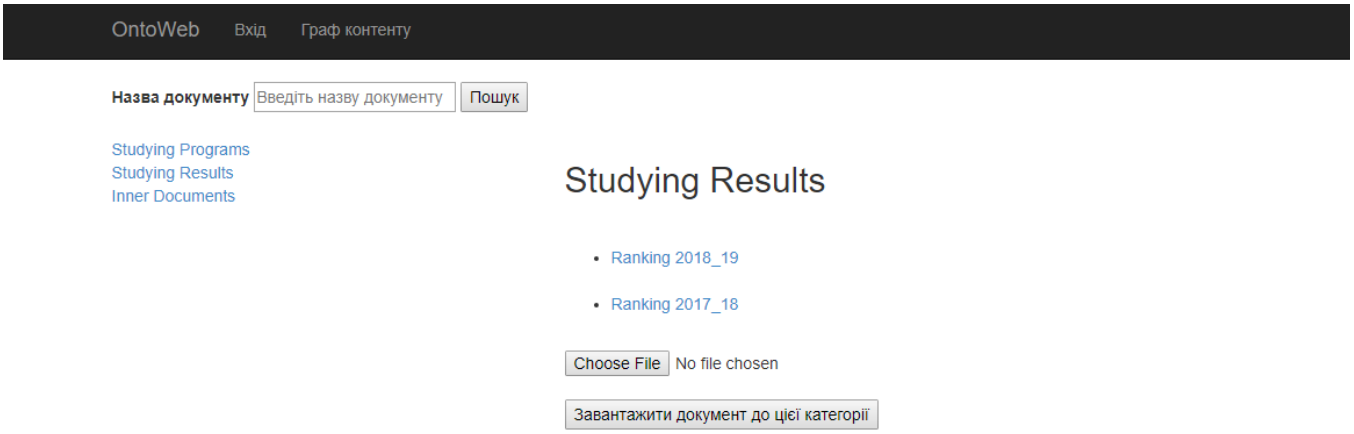

Рисунок 5.3 — Сторінка перегляду вмісту каталогу

Також користувач має можливість переглядати та завантажувати файл на локальний диск. Потрібно просто відкрити відповідний документ та натиснути на кнопку “Натиснути тут, щоб завантажити документ”. Документ завантажується на локальний диск у форматі txt (Рисунок 5.4):

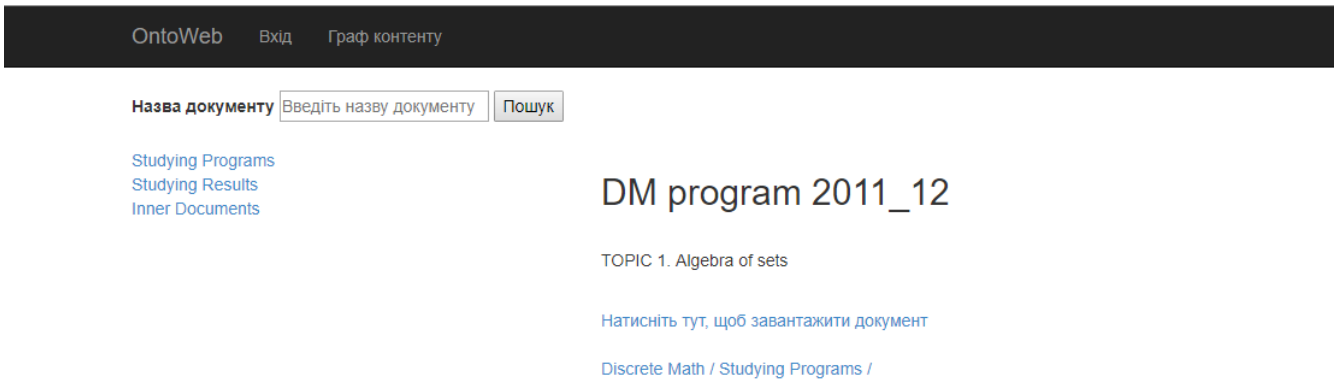

Рисунок 5.4 — Сторінка перегляду та завантаження документу

Вибравши вкладку “Граф контенту”, відображається граф ієрархії онтології, в залежності від доступних для користувача категорій. На вузлах розташовані окремі каталоги, у які вкладені інші каталоги. На кінцях графу розташовані безпосередньо самі файли (Рисунок 5.5):

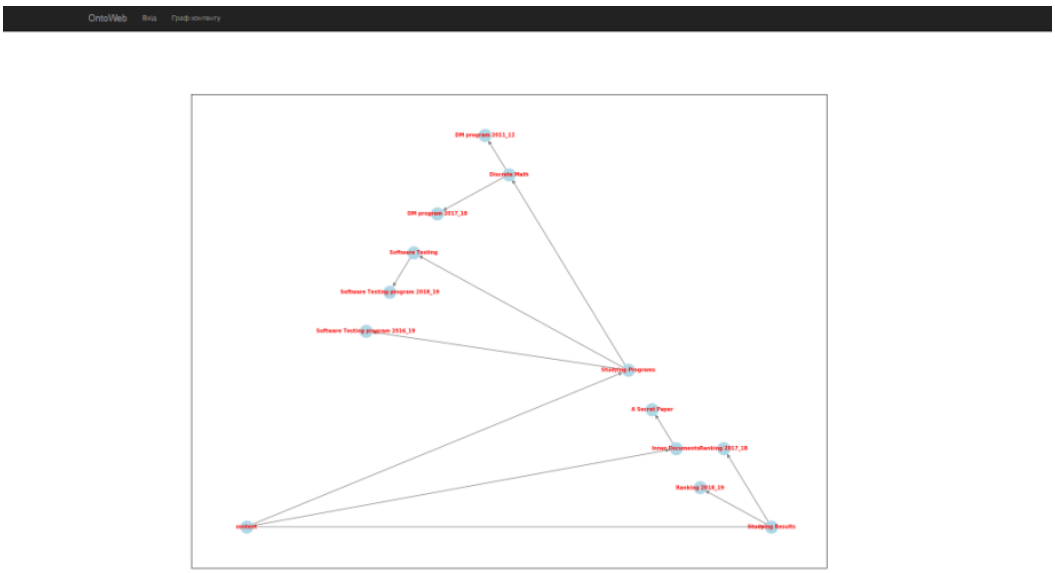

Рисунок 5.5 — Сторінка зображення графу онтології користувача

В залежності від типу доступу користувача, наповненість графу буде різною.

В системі також реалізовано пошук файлів в системі. Потрібно перейти на вкладку “OntoWeb”. В текстове поле “Назва документу ввести назву” шуканого документу та клікнути на кнопку “Пошук”. Якщо документ є в системі і він доступний користувачу, документ виводиться на екран (Рисунок 5.6):

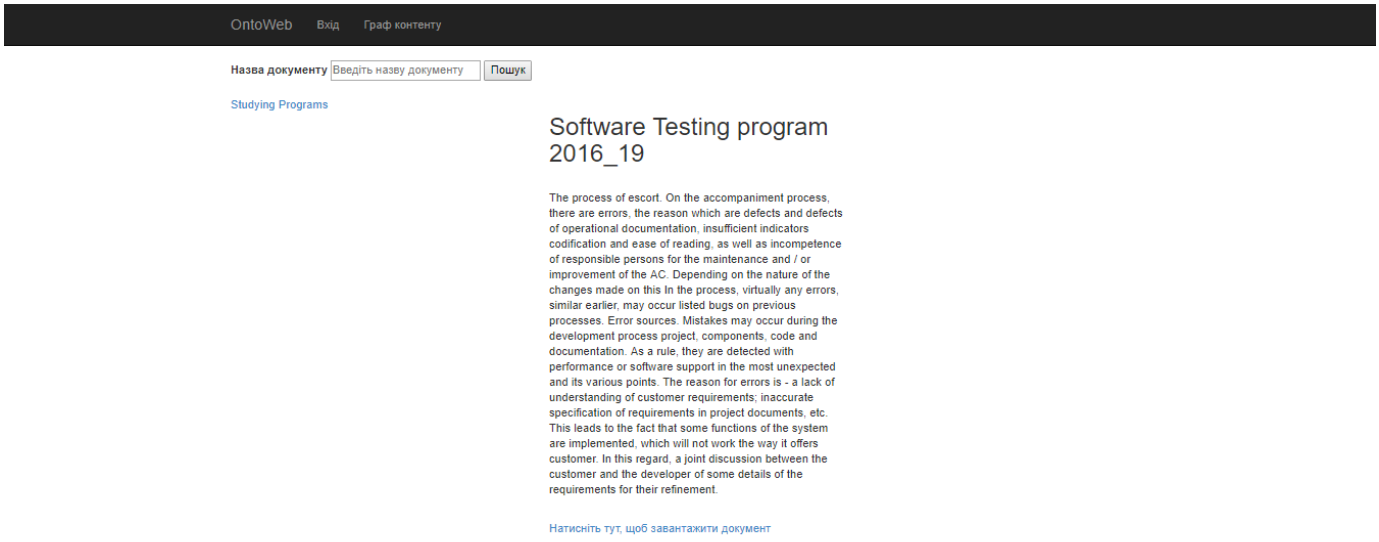

Рисунок 5.6 — Сторінка пошуку документа в системі

Програма є зручною для користування. Кожен користувач може з легкістю проводити операції, такі як

- звантажувати будь-який OWL-файл
- проводити навігацію по дереву онтології
- підвантажувати та скачувати документи, які прописані в онтології

будувати граф.

5.3 Висновки до розділу

У данному розділі було проведено опис методики роботи користувача з програмної системою. Для цього не має потреби в спеціальному обладнанні, а достатньо звичайного персонального комп’ютера. Існують різні типи доступу (наприклад, “учень”, “вчитель” та “адміністратор”), в залежності від яких будуть різні типи категорій та документів. Також буде різне наповнення графу, але структура буде однаковою для всіх користувачів. В будь-яку категорію можна добавляти свої текстові файли. Було описано системні вимоги для запуску програмної системи, загальні інструкції по роботі з нею для кінцевого користувача. За допомогою скріншотів програми користувач зможе з легкістю користуватись програмним продуктом.

ВИСНОВКИ

В рамках дипломної роботи було зроблено огляд середовищ веб візуалізації онтологій. Огляд показав,що такі ресурси поділяються на редактори онтологій та вузькоспеціалізоване програмне забезпечення, призначене для роботи з окремими задачами. Було прийняте рішення про створення системи.

Було запропоновано створити систему, яка крім можливості графічного відображення онтології надає можливість рухатись по дереву онтології, та працювати з документами, прив’язаними до вузлів дерева.

Було розроблено онтологічну інформаційно-довідкову систему школи, яка включає онтологію та інформаційну систему роботи з інформаційними ресурсами.

Результатом виконання дипломного проекту стала розробка веб-візуалізації онтології для сиситеми “Школа”. Програмний продукт розроблений на мові програмування C#, платформи .NET Framefork 4.0, середовища розробки програмного забезпечення Visual Studio 2017 та середовища Python.exe. .

Розроблений додаток надає користувачу можливість переглядати онтологію.

Необхідними можливостями, які має забезпечувати модуль, є:

— завантаження файлу онтології з комп’ютера користувача;

— можливість перегляду існуючої онтології;

— можливість додавати онтологію (документи) до системи;

— можливість завантажувати документи на локальний диск;

— можливість перегляду онтології у вигляді графу.

— можливість перекладу інтерфейсу українською мовою.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Gruber T. R. Towards principles for the design of ontologies used for knowledge sharing / T. R. Gruber. // Formal Ontology in Conceptual Analysis and Knowledge Representation. – 1993.
2. Guarino N. What Is an Ontology? / N. Guarino, D. Oberle, S. Staab. // International Handbooks on Information Systems. – 2009.
3. Jean S. Domain Ontologies: A Database-Oriented Analysis / S. Jean, J. Pierra, Y. Ait-Ameur. // Web Information Systems and Technologies, International Conferences, WEBIST 2005 and WEBIST 2006. Revised Selected Papers. – 2007. – С. 238–254.
4. Semantic Web - W3C [Електронний ресурс] – Режим доступу до ресурсу: https://www.w3.org/standards/semanticweb/.
5. Фримен А. Asp.Net MVC 5 с примерами на C# 5.0 / Адам Фримен. – М: Вильямс, 2016. – 736 с.
6. Рихтер Д. CLR via C# Программирование на платформе Microsoft .NET Framework 2.0 на языке C# / Джеффри Рихтер. – Киев: Питер, 2008. – 656 с. – (2).
7. Троелсен Э. Язык программирования C# и платформа .NET 4.5 / Эндрю Троелсен. – Киев: вильямс, 2014. – 1312 с. – (6).
8. Бодягін І. Model-View-Controller в .Net / Іван Бодягін. // RSDN Magazine. – 2016. – №2.
9. Приёмы объектно-ориентированного проектирования. Паттерны проектирования / Э.Гамма, Р. Хелм, Р. Джонсон, Д. Влиссидес. – Харьков: Питер, 2001. – 368 с.

ДОДАТОК 1

Інструментальний засіб формування запитів SPARQL до онтологій

Специфікація

УКР.НТУУ”КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТВ51146_19Б

Аркушів 2

Київ – 2019

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ” КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС _ТВ51153_19Б	Записка.docx	Пояснювальна записка
Компоненти		
УКР.НТУУ” КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС _ТВ51153_19Б 12-1	/Controllers/*.cs /Models/*.cs /Services/*.cs	Модулі серверної частини програмного продукту
УКР.НТУУ” КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС _ТВ51153_19Б 12-2	modules/workspace modules/subject modules/property modules.modals modules/result modules/config modules/i18n	Модулі клієнської частини програмного продукту
УКР.НТУУ” КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС _ТВ51153_19Б 13-1	Опис.docx	Опис серверної частини програми

ДОДАТОК 2

Побудова графу онтології предметної області

Лістинг програмного модулю

УКР.НТУУ”КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТВ51153_19Б 12-1

Аркушів 4

Київ – 2019

```
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
//Використання зовнішніх модулів
namespace WebOnto.Models
{
   public class Category
   {//Клас опису категорії
      public Category() //Конструктор класу
      {
         Supercategories = new List<Category>();
         Subcategories = new List<Category>();
         Documents = new List<Document>();
      }
//Властивості структури
      public string ID { get; set; }
      public string Name { get; set; }
      public List<Category> Supercategories { get; set; }
      public List<Category> Subcategories { get; set; }
      public List<Document> Documents { get; set; }
   }
}
//Підключення зовнішніх модулів
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace WebOnto.Models
{ //Клас опису документу
   public class Document
   {
      public Document()
      {
      }
//Властивості класу
      public string ID { get; set; }
      public string Name { get; set; }
      public string Content { get; set; }
      public Category Category { get; set; }
   }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace WebOnto.Models
{// Клас побудови онтологічного графу контенту
    public class OntoGraph
    {
        public String Draw(List<Category> topCategories)
        {
            String graph = "";
            foreach(Category category in topCategories)
            {
                graph += "content\t" + category.Name + "\r\n";
                graph += DrawRecursively(category);
            }
            return graph;
        }
//Рекурсивний метод
        private String DrawRecursively(Category category)
        {
            String graph = "";
            foreach(Category child in category.Subcategories)
            {

                graph += category.Name + "\t" + child.Name + "\r\n" + DrawRecursively(child);
            }
            foreach (Document doc in category.Documents)
            {

                graph += category.Name + "\t" + doc.Name + "\r\n";
            }
            return graph;
        }
    }
} //Скрипт для побудови візуалізації графу
import networkx as nx
import numpy as np
import matplotlib.pyplot as plt
// Завантаження інформації з текстового файлу
f = open("C:\\Users\\Max\\Documents\\Visual Studio
2017\\Projects\\WebOnto\\WebOnto\\graphinfo.txt", "r")
edges = []
```

```
for line in f:
    nodes = line.replace("\n", "").split("\t")
    edges.append((nodes[0], nodes[1]))
//Сортування ребер графу
edges.sort()
edges.reverse()
G = nx.DiGraph()
G.add_edges_from(edges)
//Побудова конфігурації візуалізації
pos = nx.planar_layout(G)
nx.draw_networkx_nodes(G, pos, cmap = plt.get_cmap('jet'), node_color='lightblue',
node_shape='o')

//Побудова зображення
nx.draw_networkx_labels(G, pos, font_size=7, font_color='red', font_weight="bold")
nx.draw_networkx_edges(G, pos, edge_color = 'grey', style = 'dotted')
figure = plt.gcf()
figure.set_size_inches(16, 12)
plt.savefig("graph.png", dpi = 100)
plt.savefig("C:\\Users\\Max\\Documents\\Visual Studio
2017\\Projects\\WebOnto\\WebOnto\Content/graph.png", dpi = 100)
plt.show()
```

ДОДАТОК 3

Побудова графу онтології предметної області

Опис програмного модулю

УКР.НТУУ”КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТВ51153_19Б 13-1

Аркушів 8

Київ – 2019

АНОТАЦІЯ

Додаток надає інформацію про структуру та функції програмного модулю побудови графу онтології предметної області.

Розроблене програмне забезпечення дозволяє перетворювати дані про онтологію з внутрішнього представлення системи у зручну візуалізацію у формі графа контенту.

Модуль було створено з використанням інтегрованого середовища розробки Microsoft Visual Studio та декількох бібліотек мови програмування Python, в тому числі бібліотеки networkx.

ЗМІСТ

ЗАГАЛЬНІ ВІДОМОСТІ

Програмний модуль є внутрішньою скадовою системи та викликається при переході користувача системи на сторінку відображення онтологічного графу контенту.

Програма була написана мовами програмування C# та Python. При цьому використовувалися бібліотеки numpy, matplotlib та networkx.

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Створена програма формує зображення онтологічного графу контенту відповідно до поточної онтології та оновлює відповідне зображення, що відображається на сторінці web-інтерфейсу.

Окремі компоненти модулю відповідають за реалізацію наступних функцій:

— запису інформації про ребра графу до проміжного подання у вигляді текстового файлу у відповідності до поточного стану системи;

— запуску на виконання скриптового файлу;

— зчитування та обробки інформації з текстового файлу відповідно до потреб візуалізації;

— побудови графу у вигляді зображення, що може бути відображене у web-сторінці.

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Структура модулю складається з двох головних складових (обробки списку категорій та побудови графу), що можуть додатково поділені на менші компоненти:

1) опис сутностей категорій та документів у класах Category та Document;
2) ініціатор відклику на дії користувача у вигляді HomeController;
3) реалізація створення списку ребер графу у класі OntoGraph;
4) скриптовий файл graph.py.

ТЕХНІЧНІ ЗАСОБИ, ЩО ВИКОРИСТОВУВАЛИСЯ

Для створення системи були використані засоби мов програмування C# та Python, фреймворк ASP.NET MVC та бібліотеки numpy, matplotlib, networkx.

Задачі web-дизайну сторінок інтерфейсу користувача були вирішені за допомогою мов розміток CSS та HTML без використання скриптової мови JavaScript.

ВХІДНІ І ВИХІДНІ ДАНІ

Вхідними даними є:

— онтологія предметної області у внутрішньому форматі системи;

— інформація про місце розташування графічного зображення графу, який треба оновити.

Вихідними даними є:

— проміжний текстовий файл зі списком ребер графу;

— графічне зображення графу контенту для поточної онтології.