

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЇВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мультимедійних систем факультету інформатики

Реалізація бази знань за допомогою системи PROTEGE
Текстова частина до курсової роботи
за спеціальністю „Інженерія програмного забезпечення” 121

Керівник курсової роботи

к.т.н., доц. Жежерун О.П.

(підпис)

“ ____ ” _____ 2020 р.

Виконала студентка БП ІПЗ-3

Гальченко М.О.

“ ____ ” _____ 2020 р.

Київ 2020

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мультимедійних систем факультету інформатики

ЗАТВЕРДЖУЮ

Зав. кафедри мультимедійних систем,
доцент, к.т.н. _____ Жежерун О.П.
(підпис)

« ____ » _____ 2019р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студентці Гальченко Марії Одиссеївні

Факультету інформатики 3 р.н. бакалаврської програми

ТЕМА: Реалізація бази знань за допомогою системи PROTEGE

Зміст текстової частини до курсової роботи:

Індивідуальне завдання

Вступ

Огляд теоретичного матеріалу і здійснення дослідження

Висновки

Список літератури

Додатки (за необхідністю)

Список посилань

Дата видачі « ____ » _____ 2019 р.

Керівник _____
(підпис)

Завдання отримав _____
(підпис)

Тема: Реалізація бази знань за допомогою системи PROTEGE

| № п/п | Назва етапу курсової роботи | Термін виконання етапу | Примітка |
|----------|---|---------------------------|----------|
| 1. | Отримання завдання на курсову роботу. | 10.11.2019 | |
| 2. | Огляд літератури за темою роботи. | 05.02.2020 | |
| 3. | Проведення досліджень | 20.02.2020 | |
| 4. | Опис результатів дослідження | 01.03.2020 | |
| 5. | Аналіз отриманих результатів з керівником | 20.03.2020 | |
| 6. | Корегування роботи | 25.03.2020 | |
| 7. | Створення презентації та написання доповіді. | 10.04.2020 | |
| 8. | Остаточне оформлення пояснювальної роботи та слайдів. | 23.04.2020 | |
| 9. | Здача курсової роботи на перевірку | 11.05.2020 | |

Студентка Гальченко М.О.

Керівник Жежерун О.П.

“ ”

Зміст

| | |
|---|-----------|
| Анотація..... | 5 |
| Вступ | 6 |
| Розділ 1.Аналіз задачі та вивчення предметної області. | 8 |
| 1.1 Постановка завдання..... | 8 |
| 1.2 Предметна область Фільми | 8 |
| Розділ 2. Поняття онтології та її реалізація за допомогою Protégé. | 10 |
| 2.1 Поняття онтології..... | 10 |
| 2.2 Огляд існуючих засобів побудови онтології. | 11 |
| 2.4 Опис онтології Фільми. | 13 |
| 2.5 Редактор онтологій Protégé..... | 13 |
| 2.6 Онтологія Фільми в редакторі Protégé. | 16 |
| Розділ 3. Мова запитів SPARQL. | 20 |
| 3.1 Мова запитів SPARQL..... | 20 |
| 3.2 Створення запитів. | 20 |
| Розділ 4. Розробка програмного застосунку на основі створеної онтології. | 23 |
| Висновки..... | 38 |
| Список використаної літератури..... | 39 |

Анотація

У своїй роботі на основі існуючих підходів та систем створення онтологій представлено технологію побудови онтології засобами Protégé-OWL. Розроблена онтологія орієнтована на пошук та аналіз інформації про фільми. Також описується процес проектування та розробки бази знань, використання SPARQL запитів до побудованої онтології та створення веб-застосунку з її використанням.

Вступ

Предметна область - фільми, яку я обрала, є досить об'ємною. До її складу входить досить велика кількість різноманітної інформації, завдяки чому проводити аналіз є складною і довгою працею. Сьогодні в багатьох сферах постає надзвичайно важливе питання створення інтелектуальних систем, що могли б полегшити пошук та доступ до потрібної інформації у величезних сховищах даних. Саме тому, в основу архітектури сучасних ІС поклали онтології, що формуються для заданої предметної області.

Онтологія – система, що складається з набору понять та набору тверджень про ці поняття. На їх основі будуються класи, об'єкти, відношення, функції та теорії. Для доступу до даних онтологій рекомендують використовувати мову запитів SPARQL. В зв'язку з цим актуальністю проекту стає можливість використання продукту на ПК та аналіз даних будь-яким користувачем. Актуальність роботи полягає в новому підході до аналізу фільмів та візуалізації результатів за допомогою SPARQL запитів.

Метою даного проекту є створення бази знань та розробка веб застосунку, який спростить роботу для користувача та при цьому не буде вимагати специфічних знань в сфері програмування. Об'єктом дослідження є процес створення бази знань фільмів, яка б відповідала поставленим задачам. Дана робота реалізована за допомогою використання програмного редактора Protégé. Для проектування бази знань та заповнення її даними використовується мова запитів SPARQL. Для відображення результатів пошуків створено веб-застосунок за допомогою засобів HTML, CSS, JS.

Робота містить такі розділи:

Перший розділ описує аналіз поставленої задачі, вивченню предметної області та основних понять.

У другому розділі описується поняття онтології, її побудова в системі Protegé та сама розробка.

У третьому розділі описується обрана мова запитів SPARQL. Надається розроблена база знань та виконання запитів за допомогою SPARQL.

У четвертому розділі описується розробка веб-застосунка з використанням створеної онтології.

Розділ 1. Аналіз задачі та вивчення предметної області.

1.1 Постановка завдання

Метою даної курсової роботи є організація, класифікація і керування даними великого обсягу, в даному випадку фільмами. А також розробка веб-сайту, з використанням запитів мовою SPARQL. Задля полегшення задачі курсової роботи використаний так званий «онтологічний підхід». Структура створеної онтології, містить основні класи з фільмами, їх жанрами, акторами та режисерами, їх взаємозв'язки та типи даних. Правильно створена структура онтології зменшить час пошуку інформації та спростить аналіз. Задля зручності керування онтологією людьми без знань та навичок програмування, був розроблений онлайн-ресурс у вигляді веб-сайту.

Постановка завдання:

- аналіз предметної області;
- огляд існуючих засобів побудови онтологій;
- створення онтології фільмів;
- написання запитів SPARQL;
- розробка зручного інтерфейсу;
- підключення онтології та запитів до створеної програми.

1.2 Предметна область Фільми

Кінофільм, також **фільм**, **кіно** - аудіовізуальний твір кінематографії, що складається з епізодів, поєднаних між собою творчим задумом і зображувальними засобами, та який є результатом спільної діяльності його авторів, виконавців і виробників [1]. За принципом, метою створення та за призначенням слід розрізняти художні, документальні, публіцистичні, навчальні та наукові фільми. За своєю тематикою та засобами вираження фільми поділяють на значну кількість жанрів. Також окрім інформації про

самий жанр та інших характеристик фільму (повнометражний, короткометражний), існує інформація і про знімальну групу кінофільму, з її керівним складом, сценаристами, акторами, режисерами та іншими.

Сьогодні бази знань(Freebase, Dbpedia и Linkedmdb) вражають своїми масивами даних на тему кінематографа. Ми можемо легко подивитись, не лише які актори знімались в кіно, а й де вони народились, їх сімейне становище та будь-що окрім зйомок.

В автоматизованих системах вся інформація про фільми традиційно зберігається в реляційних базах даних (БД). Такі системи дозволяють шукати різноманітні фільми та інформацію про них, сортувати по жанрам, року, тривалості, рейтингу, тощо. Але фільми – складна структура з великою кількістю структурно-логічних залежностей, які часто випускають при описі структури даних в реляційних БД. Тому в даній роботі була використана база знань – онтологія.

Розділ 2. Поняття онтології та її реалізація за допомогою Protégé.

2.1 Поняття онтології.

Вперше, термін «онтологія» з'явився в роботі Томаса Грубера, в якій розглядались різноманітні аспекти взаємодії інтелектуальних систем між собою та з людиною [2]. За визначенням Тома Грубера, що застосував це поняття в області інформаційних технологій, онтологія - це «специфікація концептуалізації» [3]. Існує безліч об'єктів, що підпадають під визначення онтології, наприклад ієрархія класів в ООП, концептуальні карти і так далі.

Інтелектуальні системи – програми, що можуть моделювати деякі види інтелектуальної діяльності людини. Саме для цього і створювали комп'ютери, адже комп'ютерна система та програми, що виконують будь-яку діяльність, звільняє людину від її виконання. Ця діяльність може бути як простою, наприклад звичайні обрахунки, так і дуже складною, проте вона завжди є однотипною. Комп'ютерна система, що була створена з метою обробки відео, не може бути використана для перекладу іноземних мов. Для виконання своєї роботи, в кожному системі закладають певну інформацію, яка зазвичай не змінюється. Інтелектуальні системи є більш гнучкими та універсальними в цьому плані, адже інформація про те, що вона має робити не закладають раз і назавжди, вона може змінюватись. Саме тому виникла потреба у передачі інформації у вигляді даних та її опису. Опис має бути зрозумілим будь-якій системі та іншій людині. Для цього Грубер запропонував два методи опису знань:

- В традиційній формі мовою логіки предикатів (наприклад Prolog);
- В формі онтології.

Онтологія в сучасному розумінні – інструмент розуміння навколишнього світу через побудову концептуальної моделі. При розробці онтології завжди намагаються досягти незалежності від інформаційних

потреб і груп користувачів, тому онтологія включає в себе більш загальні знання.

Грубер розділяв специфікації знань на дві форми (канонічну та онтологію). Звісно, такий розподіл не є зручним, адже тоді доводилось би описувати одні й ті самі знання двічі. Сучасні онтології мають перевагу в суміщенні обох форм специфікацій в одне ціле.

Онтологія - будь-який опис декларативних знань, що має бути представлений у вигляді класів з відношенням ієрархії між ними. У звичайному вигляді онтологія - набір елементів чотирьох типів:

- поняття (класи);
- екземпляри (індивіди);
- відносини (предикати);
- аксіоми.

Поняття (класи) – абстрактні групи або набори об'єктів. До їх складу можуть входити інші класи, екземпляри або поєднання обох.

Екземпляри(індивіди) – основі, нижньорівневі компоненти. Одна з основних цілей онтології є правильна та точна класифікація екземплярів.

Відносини(предикати) - зв'язки між об'єктами.

Атрибути – інформація про об'єкти.

Аксіоми – моделювання тверджень.

2.2 Огляд існуючих засобів побудови онтології.

Онтології широко використовуються у всіх областях. У зв'язку з їх використанням у різноманітних застосунках виникла необхідність створити способи їх представлення. Тоді почався розвиток мов, які б могли використовуватись у всіх системах, найвідомішими стали RDF і OWL. Також виникла велика кількість редакторів для створення та редагування онтологій. Кожен із редакторів направлений на роботу з конкретним форматом даних і має свої властивості.

Для створення онтологій та їх редагування розроблено безліч редакторів, середовищ розробки, парсерів та інших засобів. Найбільш ефективними з яких є: KAON [4], OilEd [5], OntoEdit [6], Ontosaurus [7], OpenCyc [8], Protégé [9]. Також розроблено безліч мов специфікації онтологій, насамперед: CycL, Ontolinngua, мови, засновані на логіці дескрипту (LOOM), мови, засновані на фреймах (OKBC, OCML, Flogic), мови, засновані на Web-стандартах (XOL, SHOE, UPML). для обміну онтологіями через Web були створені RDF (S), DAML, OIL, OWL [10].

2.3 Переваги та недоліки онтологічної моделі.

Онтологічна модель в деякому сенсі являє собою базу даних, проте онтологія має значну кількість відмінностей, що визначають її переваги та недоліки, серед них можна виділити:

- Інформація, якої немає в онтології визначається як unknown, в той час, як в моделі баз даних відсутність інформації визначається помилкою (false).
- Онтологія дає можливість визначити неявні вказані залежності між сутностями, коли база даних дозволяє визначати лише однозначні виведення, що в неї внесли.
- Екземпляри онтології можуть мати більше одного імені, коли в базі даних кожна сутність повинна мати унікальне ім'я.
- Онтологія написана по загальним знанням певної області, що дає можливість уникнути відмінностей в термінології, проблеми синонімії та багатомовності.
- Гнучкість онтології дає можливість легко змінювати та розширювати модель.
- Можливість повторного використання раніше створених онтологій. Ця перевага дає змогу користуватися готовими рішеннями та не створювати схему даних з нуля.

- Можливість об'єднувати різні організації, прикладаючи мінімальні зусилля.

Саме тому, онтологічна модель буде виводити більш точну та правдоподібну інформацію на сформовані запити. Наприклад, в тому випадку, коли декілька об'єктів (у нашому випадку: фільми) матимуть схожі характеристики (атрибути), такі як однаковий режисер, актори, тривалість, жанр, онтологія, завдяки своїм властивостям зможе з легкістю знайти правильну відповідь за допомогою прописаним відношенням між сутностями.

Серед недоліків онтологічної моделі можна визначити наступні:

- Для визначення сутності потребується виконання логічних правил.
- Зв'язки між об'єктами онтологічної моделі являються досить складними в порівнянні з такою ж моделлю в базі даних. Досить часто через розмаїття зв'язків онтологія виходить досить масштабною.

2.4 Опис онтології Фільми.

Предметна область онтології – фільми. Онтологія буде використовуватись для пошуку за різними характеристиками фільмів та знаходити фільми, що підходять по критеріям пошуку користувача.

Для формування онтології, що описує фільми, введемо наступні поняття. У якості класів виступають фільми та знімальна група. Фільми має містити підкласи за метою створення та за призначенням. Знімальна група має містити підкласи свого складу (актори, режисери, сценаристи і так далі). В якості екземплярів виступають конкретні фільми та конкретні люди, що брали участь у створенні фільму.

2.5 Редактор онтологій Protégé.

Protégé – програмний застосунок з відкритим безкоштовним кодом для редагування онтологій і систем управління базами знань [11]. Даний редактор був розроблений у Стенфордському університеті. Protégé

підтримується значною кількістю користувачів, які використовують його для вирішення задач пов'язаних із базами знань у різноманітних сферах сучасного життя. Для створення власної онтології було обрано редактор Protégé, так як він має ряд переваг та зручностей у використанні, такі як:

- Постійні оновлення та нові версії.
- Вільна інтеграція в інші проекти (крос-платформеність).
- Велика кількість доповнень (плагінів).
- Сумісність з OWL.
- Зрозуміле та зручне представлення онтології у вигляді графу за допомогою плагіну OntoGraph.
- Підтримує правила, засновані на мові SWRL.
- Підтримує будь-яку базу даних з драйвером JDBC 1.0, що дає можливість використовувати більшість реляційних баз даних (Access, SQL Server, Oracle, MySQL).
- Зручний та зрозумілий користувачеві інтерфейс.

Редактор Protégé має два типи властивостей(properties):

- Data Properties – властивості типів даних.
- Object Properties – властивості об'єктів.

Властивостями об'єкта являються відношення між двома екземплярами.

У Protégé існують наступні типи Object Properties:

1. Функціональні (Functional)

Якщо властивість є функціональною, то для одного екземпляра може існувати не більше одного екземпляра, який має відношення до першого через цю властивість.

2. Обернено-функціональні (Inverse functional).

Якщо властивість є обернено-функціональною, то дана властивість є оберненою до функціональної властивості.

3. Транзитивні (Transitive).

Якщо властивість є транзитивною, то є умова транзитивності: якщо екземпляр a зв'язаний з b , а b зв'язаний з c , то можемо зробити висновок, що a зв'язаний з c через транзитивну властивість.

4. Симетричні (Symmetric).

Якщо властивість x є симетричною, і екземпляр a зв'язаний з b через таку властивість, то можемо зробити висновок, що b також зв'язаний з a через властивість x .

5. Асиметричні (Asymmetric).

Якщо властивість x є асиметричною, і екземпляр a зв'язаний з b через таку властивість, то b не може бути зв'язаний з a через властивість x .

6. Рефлексивні (Reflexive).

Властивість x є рефлексивною, якщо екземпляр a зв'язаний сам із собою.

7. Іррефлексивні (Irreflexive).

Якщо властивість x є іррефлексивною, то вона зв'язує екземпляр a і b через таку властивість, проте екземпляри a та b обов'язково мають бути різними.

- ☒ **Functional**
- ☐ **Inverse functional**
- ☐ **Transitive**
- ☐ **Symmetric**
- ☐ **Asymmetric**
- ☐ **Reflexive**
- ☐ **Irreflexive**

Мал. 2.1 Види властивостей об'єктів

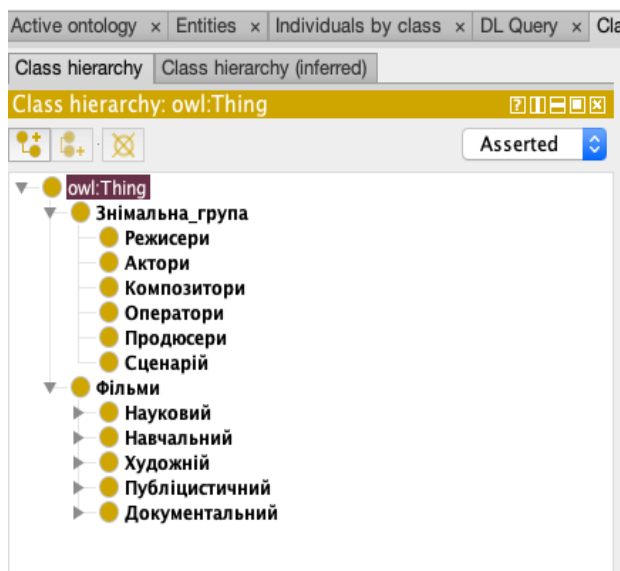
Властивості типів даних створюються для характеристики екземплярів.

2.6 Онтологія Фільми в редакторі Protégé.

В результаті вибору написання онтології в редакторі Protégé були визначені наступні кроки:

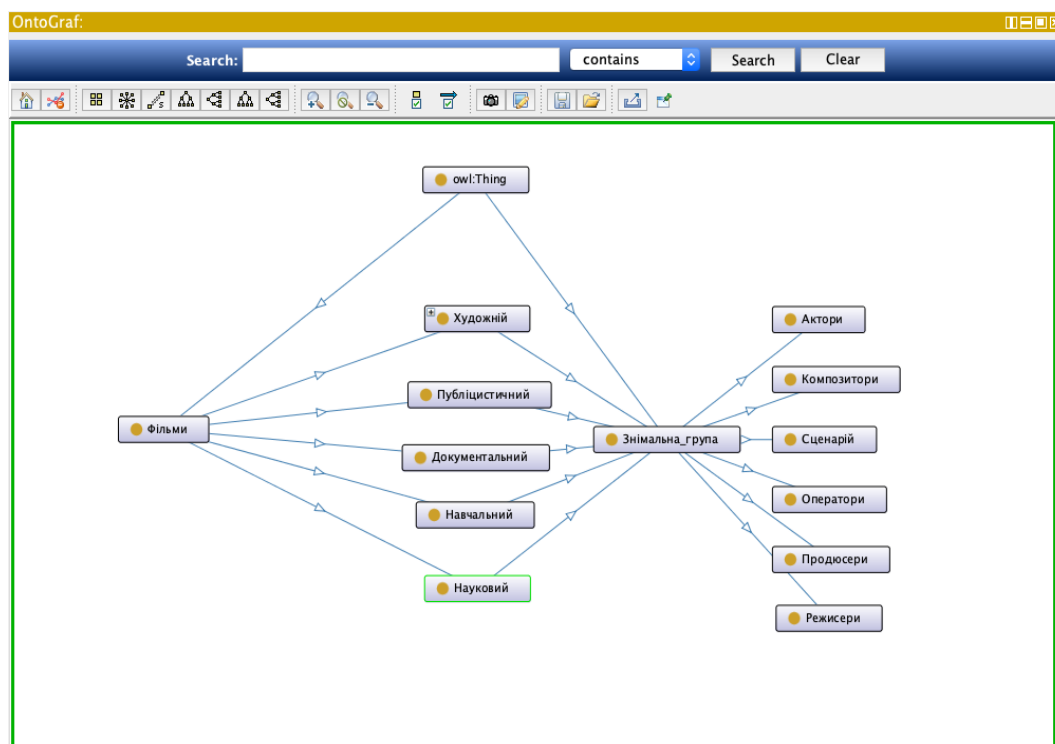
1. Введення класів:

Класи в Protégé – інтерпретація множин, елементами яких є екземпляри. Вони можуть бути сформовані в ієрархію. Підкласи -підмножини свого суперкласу. Початково пуста онтологія має лише один клас під іменем Thing. Даний клас – набір, що містить в собі всі об'єкти предметної області. Враховуючи, що обрана предметна область являється «Фільми», були визначені наступні класи та підкласи:



Мал. 2.2 Класи предметної області «Фільми»

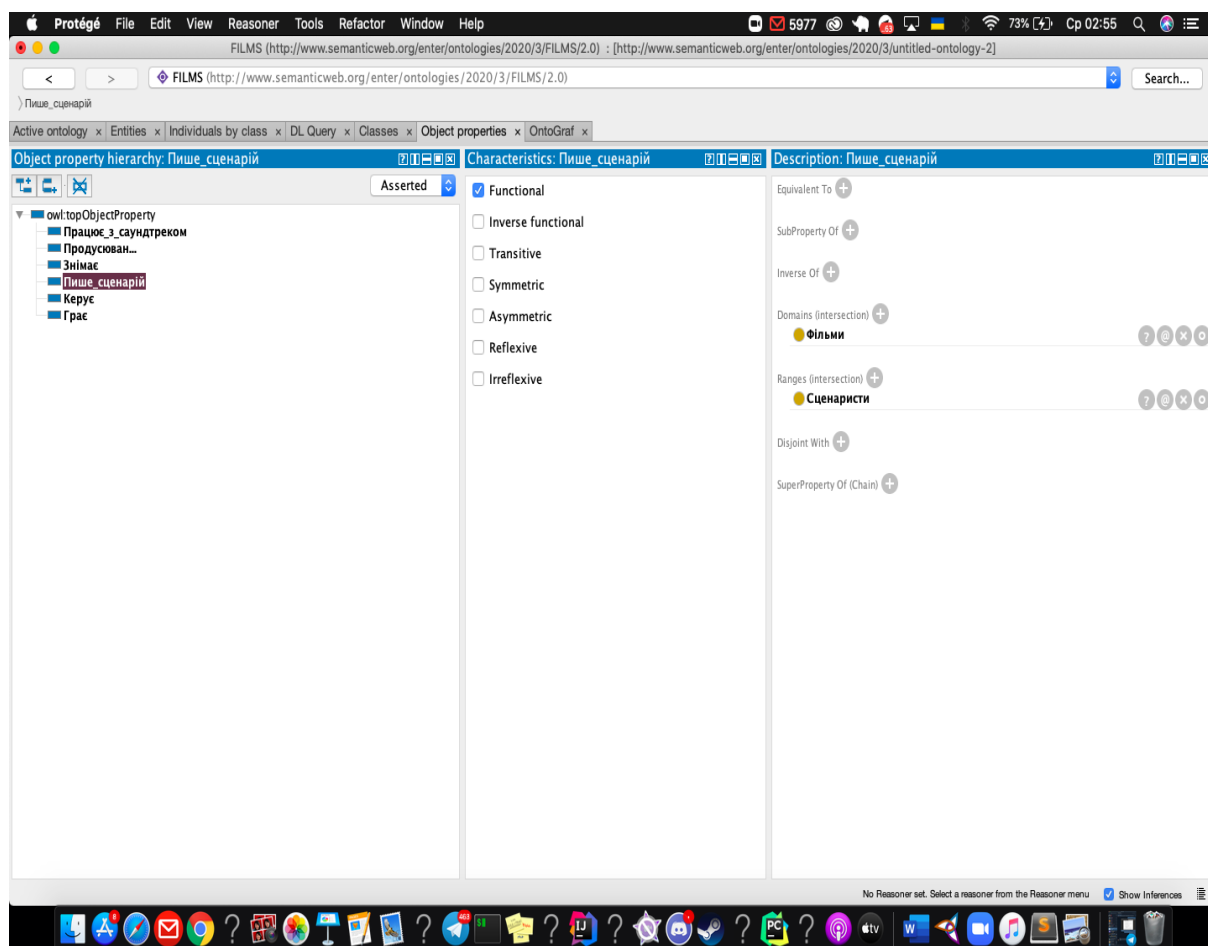
Також отриманий результат можна вивести у вигляді графіку за допомогою плагіну OntoGraph:



Мал. 2.3 Граф створених класів

2. У даній роботі були створені наступні властивості об'єктів: всі підкласи класу «Знімальна група» відносяться до підкласів класу «Фільми».
- Об'єктна властивість Грає функціонально зв'язує підклас Актори з класом Фільми, де доменом (Domains) є клас Фільми, а діапазоном (Ranges) є клас Актори.
 - Об'єктна властивість Керує функціонально зв'язує підклас Режисери з класом Фільми, де доменом (Domains) є клас Фільми, а діапазоном (Ranges) є клас Режисери.
 - Об'єктна властивість Пише_сценарій функціонально зв'язує підклас Сценаристи з класом Фільми, де доменом (Domains) є клас Фільми, а діапазоном (Ranges) є клас Сценаристи.
 - Об'єктна властивість Пише_сценарій функціонально зв'язує підклас Сценаристи з класом Фільми, де доменом (Domains) є клас Фільми, а діапазоном (Ranges) є клас Сценаристи.

- Об'єктна властивість Знімає функціонально зв'язує підклас Оператори з класом Фільми, де доменом (Domains) є клас Фільми, а діапазоном (Ranges) є клас Оператори.
- Об'єктна властивість Продюсування функціонально зв'язує підклас Продюсери з класом Фільми, де доменом (Domains) є клас Фільми, а діапазоном (Ranges) є клас Продюсери.
- Об'єктна властивість Працює_з_саундтреком функціонально зв'язує підклас Композитори з класом Фільми, де доменом (Domains) є клас Фільми, а діапазоном (Ranges) є клас Композитори.



Мал. 2.4 Об'єктна властивість на прикладі сценаристів

3. Створення властивостей даних:

| Назва властивості | Опис | Тип даних |
|-------------------|--|-----------|
| Жанр | Жанр конкретного фільму (комедія, бойовик, тощо) | string |
| Назва | Назва фільму («Леон», «Таксі», «Люсі») | string |
| Реліз | Дата випуску фільму | datetime |
| Тривалість | Час фільму у хвилинах | integer |
| Ім'я | Ім'я | string |
| Прізвище | Прізвище | string |
| Громадянство | Громадянство | string |
| Кількість_фільмів | Кількість фільмів, у яких брала участь людина | integer |
| Зріст | Зріст | integer |
| Дата_народження | Дата народження | datetime |

Таб. 2.1 Властивості типів даних

Отже, в даній роботі була створена онтологія по предметній області «Фільми». До її складу входить: 14 класів і підкласів, 6 властивостей об'єктів, 10 атрибутів класів і 240 екземплярів. Загальна кількість аксіом онтології – 1563.

Розділ 3. Мова запитів SPARQL.

3.1 Мова запитів SPARQL.

У якості мови запитів до створеної онтології було використано мову SPARQL. Дана мова високотехнологічна та має хороший потенціал.

- У 2008 році SPARQL отримав статус офіційної рекомендації консорціуму W3C2.
- Мова SPARQL не прив'язана до жодного програмного комплексу.
- Для SPARQL існує велика кількість програмних реалізацій та застосунків.
- SPAQRL дає можливість користувачам писати глобально однозначні запити.
- У SPARQL не накладаються обмеження на формат даних, що дає можливість взаємодіяти ресурсам різних типів.

3.2 Створення запитів.

Загальна схема SPARQL-запиту має вигляд:

```
PREFIX foo: <http://example.com/resources/>
# префіксні оголошення
FROM ...
# джерела запиту
SELECT ...
# пункт результату
WHERE {...}
# критерії запиту
ORDER BY ...
# модифікатори запиту
```

Мал. 3.1 Приклад запиту SPARQL

- Префіксні оголошення (Prefix) – скорочення URI.
- Джерела запиту (FROM) – визначають RDF графи.
- Пункт результату(SELECT) – повертає вибірку, що задовольняє критерії запиту.
- Модифікатори(ORDER BY) – упорядкування, сортування, перетворення результатів запиту.

Запит, що виводить всі фільми жанру фантастика:

```
SELECT DISTINCT ?x ?y WHERE
{ ?x Фільми:жанр ?y .
  FILTER regex(?y, 'фантастика')
}
```

| x | y |
|--|--------------|
| 1 Фільми:Валеріан і місто тисячі планет | "фантастика" |
| 2 Фільми:П'ятий елемент | "фантастика" |
| 3 Фільми:Голодні ігри | "фантастика" |
| 4 Фільми:Гаррі Поттер та Філософський камінь | "фантастика" |
| 5 Фільми:Фантастичні звірі та де їх шукати | "фантастика" |
| 6 Фільми:Зоряні війни | "фантастика" |
| 7 Фільми:Вартові Галактики | "фантастика" |
| 8 Фільми:Загін самогубців | "фантастика" |

Мал. 3.2 Фільми жанру фантастика

Запит, що виводить інформацію про фільми, режисер яких є Люк Бессон:

```
SELECT ?x ?y ?z WHERE{
?x Фільми:назва ?y .
?x Фільми:жанр ?z .
?x Фільми:тривалість ?k .
?x Знімальна група: режисери ?k .
  FILTER regex(?k, 'Люк Бессон') .
}
```

| x | y | z |
|----------------------------------|--------------|-----|
| 1 Валеріан і місто тисячі планет | "фантастика" | 137 |
| 2 П'ятий елемент | "фантастика" | 127 |
| 3 Леон | "бойовик" | 110 |
| 4 Люсі | "бойовик" | 90 |
| 5 Анна | "бойовик" | 100 |
| 6 Таксі | "комедія" | 89 |
| 7 Васабі | "бойовик" | 95 |
| 8 13-ий район | "бойовик" | 84 |

Showing 1 to 8 of 8 entries

Мал. 3.3 Фільми, що знімав Люк Бессон

Запит, що виводить перші 5 назв документальних фільмів:

```
SELECT DISTINCT ?x ?y WHERE{
?x Документальні:назва ?y .
}
LIMIT 5|
```

Showing 1 to 5 of 5 entries

x

-
- 1 Океани
 - 2 Незалежна гра
 - 3 Створюючи вбивцю
 - 4 Кімната 237
 - 5 Генезис 2.0
-

Мал. 3.4 Документальні фільми

Розділ 4. Розробка програмного застосунку на основі створеної онтології.

При розробці веб-сайту використовувались наступні технології: Bootstrap, HTML5, CSS3, JQuery.

Bootstrap – безкоштовний HTML, CSS, JS фреймворк, що використовується багатьма веб-розробниками для швидкого створення веб-сайтів та веб-застосунків. Даний фреймворк використовують у всьому світі. Його популярність можна пояснити тим, що на створення сайту з використанням Bootstrap витрачається у два рази менше часу, ніж без нього. Фреймворк являється набором CSS та JavaScript файлів. Для їх використання достатньо їх підключення до сторінки. Після цього стануть доступні всі інструменти: сітка Bootstrap, класи та компоненти. Наприклад, для створення кнопки на сторінці за допомогою Bootstrap, достатньо до посилання або кнопки button додати декілька класів.

```
<!--Щоб створити посилання у вигляді кнопки, додамо 2 класи: btn и btn-  
success -->  
<a href="#" class="btn btn-success"> Посилання, оформлене у вигляді кнопки  
</a>
```

Мал. 4.1 Приклад використання Bootstrap

Переваги та недоліки Bootstrap:

- Швидкість та легкість створення якісних, адаптивних сайтів.
- Кросбраузерність і кросплатформеність.
- Велика кількість якісно продуманих, протестованих, готових компонентів.
- Добре продуманий дизайн.

- Велика кількість матеріалів та гайдів для роботи.
- Розмір CSS та JS файлів є значно більшим, ніж при прописуванні їх на чистому CSS, JS.

Для роботи зі створеною онтологією було використано один із модулів Bootstrap – BootOx. Даний модуль можна використовувати для завантаження та змін онтології. Для того, щоб завантажити онтологію потрібно ввести наступний код:

```
new ontology = ontologyFilms;
ontologyFilms get_ontology ('file://Desktop/films.owl')
ontologyFilms.load();
```

Лістинг 4.1 Під'єднання онтології за допомогою BootOx

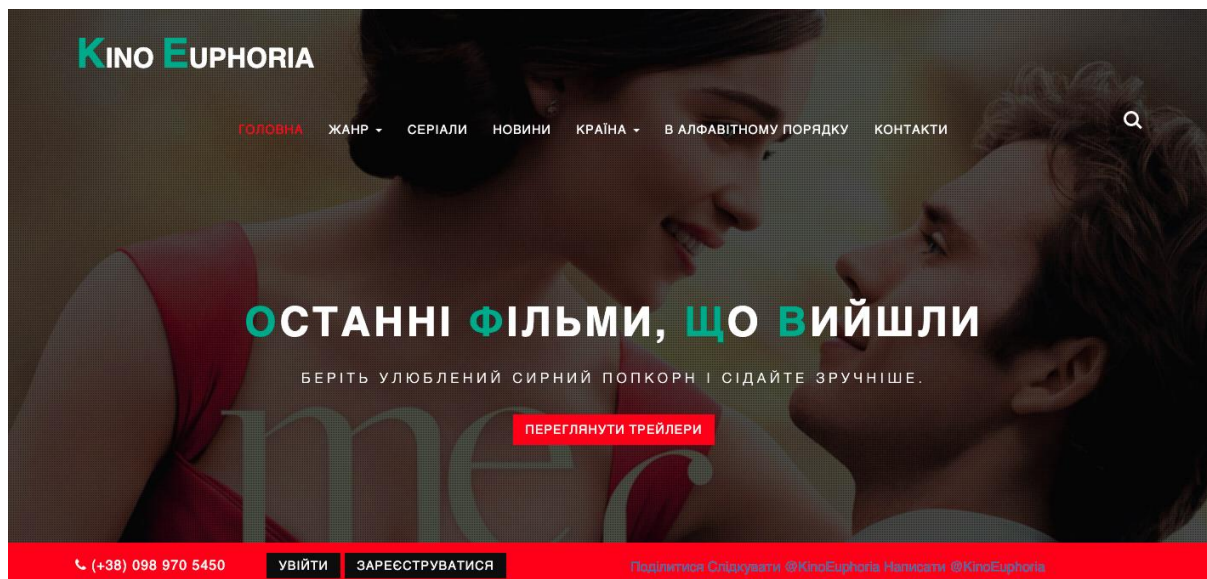
Створений веб-застосунок призначений для користувачів, що шукають інформацію про фільми або ж самі фільми для перегляду по заданим критеріям. Відповіді, що отримуються у результаті тесту обробляються та передаються у запит для знаходження співпадінь.

```
<p> Фільми з жанром фантастика
<input type="genre" name="fantasy" value="yes"> True
<input type="genre " name="fantasy " value="no"> False
</p>
```

Лістинг 4.2 Пошук співпадінь в базі знань

Рівень представлення виконаний за допомогою HTML5 та CSS3.

Головна сторінка містить вкладки з різними видами сортування та кнопку трейлерів, де наявне відео трейлерів фільм, що вийшли останнім часом. Також наявні кнопки реєстрації та входу.



Мал. 4.2 Головна сторінка

Частини HTML, CSS та JS коду, що демонструють скріншоти:

```
<div class="baner-info">
<h3><span>О</span>станні <span>Ф</span>ільми, <span>щ</span>о
<span>в</span>ийшли</h3>
<h4>Беріть улюблений сирний попкорн і сідайте зручніше.</h4>
<a class="w3_play_icon1" href="#small-dialog1">Переглянути трейлери</a>
</div>
<ul>
<li><i class="fa fa-phone" aria-hidden="true"></i> (+38) 098 970 5450</li>
<li><a href="#" class="login" data-toggle="modal" data-target="#myModal4">
Увійти </a></li>
```

```

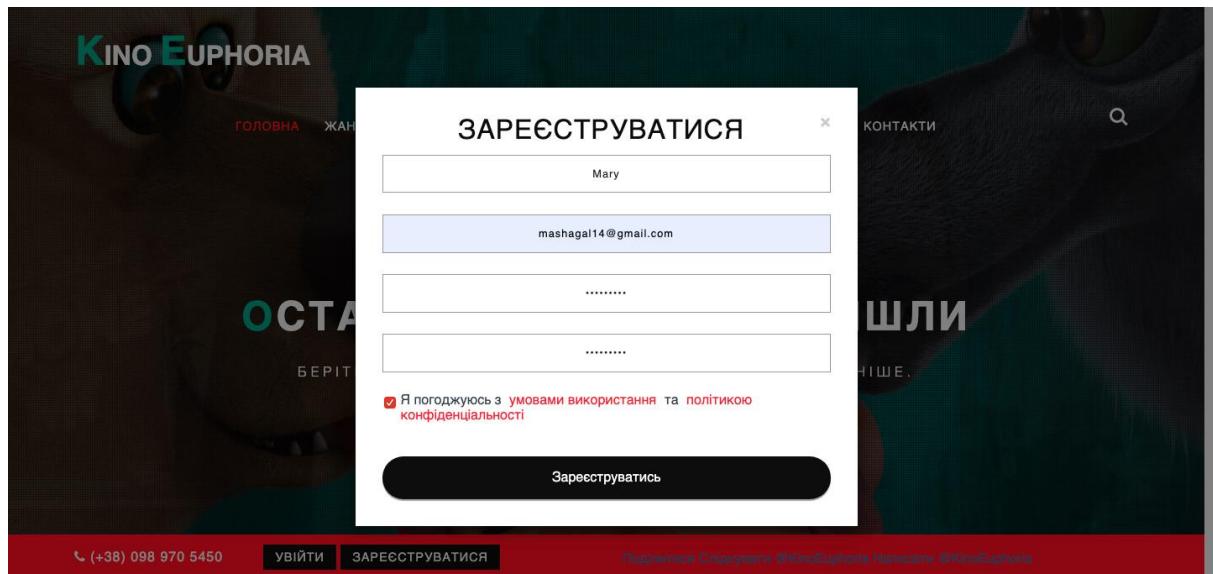
<li><a href="#" class="login reg" data-toggle="modal" data-
target="#myModal5">Зареєструватися</a></li>
</ul>
<ul>
<li>
<div class="fb-like" data-href="https://www.facebook.com/w3layouts" data-
layout="button_count" data-action="like" data-size="small" data-show-
faces="false" data-share="false"></div>
<script>(function(d, s, id) { var js, fjs = d.getElementsByTagName(s)[0];
if (d.getElementById(id))
return;
js = d.createElement(s); js.id = id;                                js.src =
"//connect.facebook.net/en_GB/sdk.js#xfbml=1&version=v2.7";
fjs.parentNode.insertBefore(js, fjs);}
(document, 'script', 'facebook-jssdk'));
</script>
</li>
<li>
<div class="fb-share-button" data-href="https://www.facebook.com/w3layouts"
data-layout="button_count" data-size="small" data-mobile-iframe="true"> <a
class="fb-xfbml-parse-ignore" target="_blank"
href="https://www.facebook.com/sharer/sharer.php?u=https%3A%2F%2Fwww.fa
cebook.com%2Fw3layouts&src=sdprepares">Поділитися</a></div>
</li>
<li class="news-twitter">
<a href="https://twitter.com/w3layouts" class="twitter-follow-button" data-show-
count="false">Слідкувати @KinoEuphoria</a><script async
src="//platform.twitter.com/widgets.js" charset="utf-8"></script>
</li>
</script>

```

```
</li>
</ul>
```

Лістинг 4.3 Клас index.html

Вікно реєстрації:



Мал. 4.3 Вікно реєстрації

```
<button type="button" class="close" data-dismiss="modal">&times;</button>
<h4>Зареєструватися</h4>
<div class="login-form">
  <form action="#" method="post">
    <input type="text" name="name" placeholder="Name" required="">
    <input type="email" name="email" placeholder="E-mail" required="">
    <input type="password" name="password" placeholder="Password" required="">
    <input type="password" name="confirm password" placeholder="Confirm
    Password" required="">
    <div class="signin-rit">
      <span class="agree-checkbox">
        <label class="checkbox"><input type="checkbox" name="checkbox">Я
        погоджуюсь з <a class="w3layouts-t" href="#" target="_blank">умовами
```

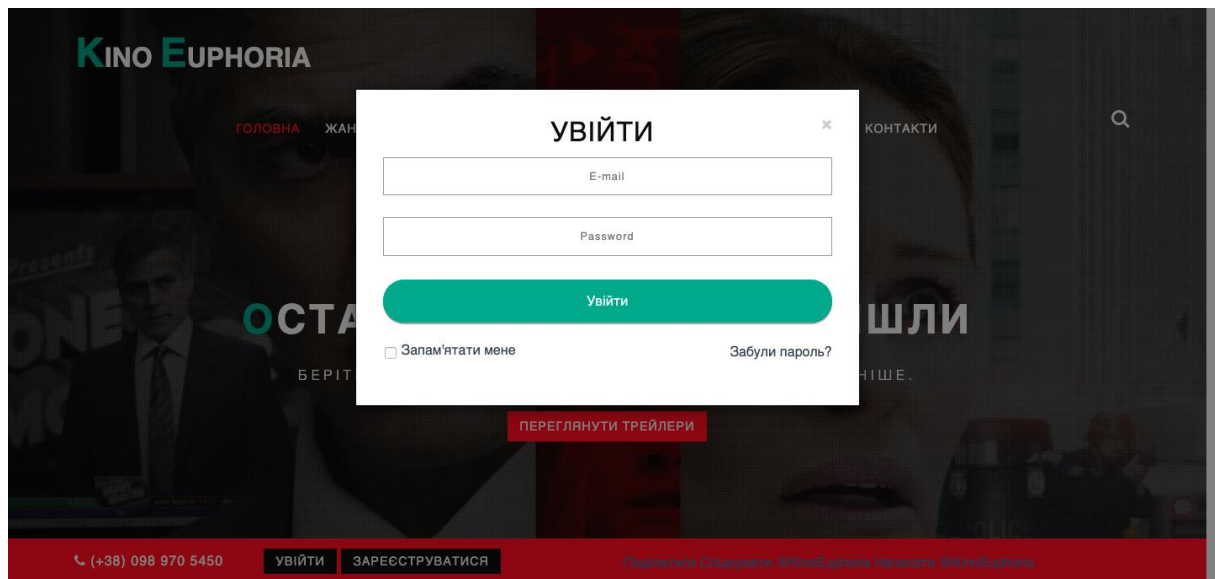
```

використання</a> та <a class="w3layouts-t" href="#"
target="_blank">політикою конфіденціальності</a></label>
</span>
</div>

```

Лістинг 4.4 Створення вікна реєстрації

Вікно входу:



Мал. 4.4 Вікно входу

```

<button type="button" class="close" data-dismiss="modal">&times;</button>
<h4>Увійти</h4>
<div class="login-form">
  <form action="#" method="post">
    <input type="email" name="email" placeholder="E-mail" required="">
    <input type="password" name="password" placeholder="Password" required="">
    <div class="tp">
      <div class="forgot-grid">
        <div class="log-check">
          <label class="checkbox"><input type="checkbox"
name="checkbox">Запам'ятати мене</label>

```

```

</div>
<div class="forgot">
<a href="#" data-toggle="modal" data-target="#myModal2">Забули пароль?</a>
</div>
<div class="clearfix"></div>
</div>

```

Лістинг 4.5 Створення вікна входу

//перевірка введених даних

```

    if(mail.value == "" || pass.value == "" || surname.value == "" || name.value
== "" || c_pass.value == ""){
mail.style.border = "1px solid red";
pass.style.border = "1px solid red";
surname.style.border = "1px solid red";
name.style.border = "1px solid red";
c_pass.style.border = "1px solid red";

    return;}

mail.style.border = "none";
pass.style.border = "none";
surname.style.border = "none";
name.style.border = "none";
c_pass.style.border = "none";

    if(c_pass.value != pass.value){
        alert("Невірний пароль");
        pass.style.border = "1px solid red";
        c_pass.style.border = "1px solid red";
        return;
    }

    var dog = false;

```

```

for (var i = 0; i < mail.value.length; i++) {
    if(mail.value.charAt(i) == '@'){
        dog = true;
        break;
    }
}
if(!dog){
    mail.style.border = "1px solid red";
    alert("email має містити @");
    return;
}

```

Лістинг 4.6 Перевірка введених даних

```

//створення нового користувача

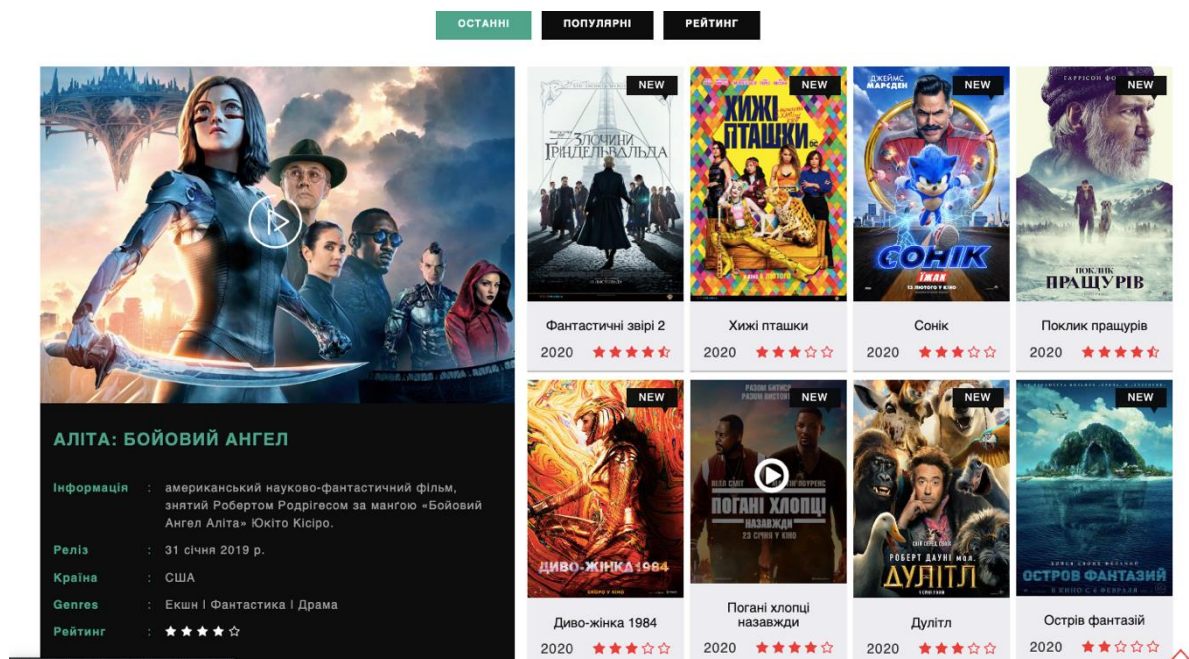
var person = {email: mail.value, password: pass.value, name: name.value,
surname: surname.value};

user = person;
people = newArray(person);
document.location.href = "main.html";
return false;
}

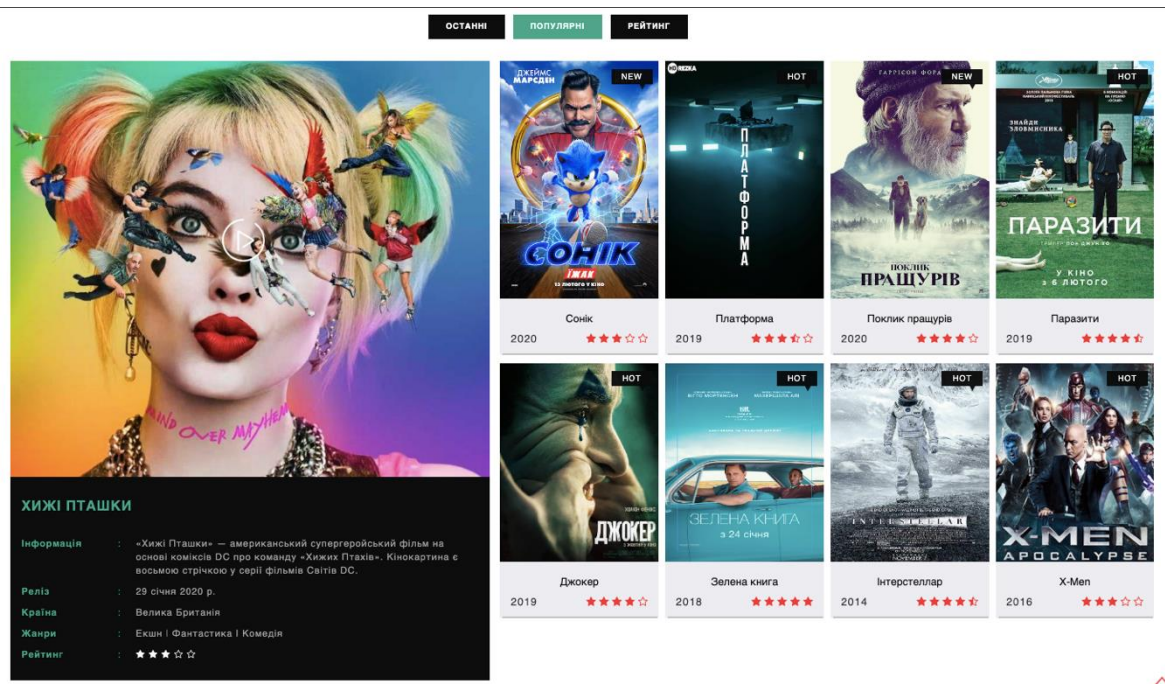
```

Лістинг 4.7 Створення нового користувача

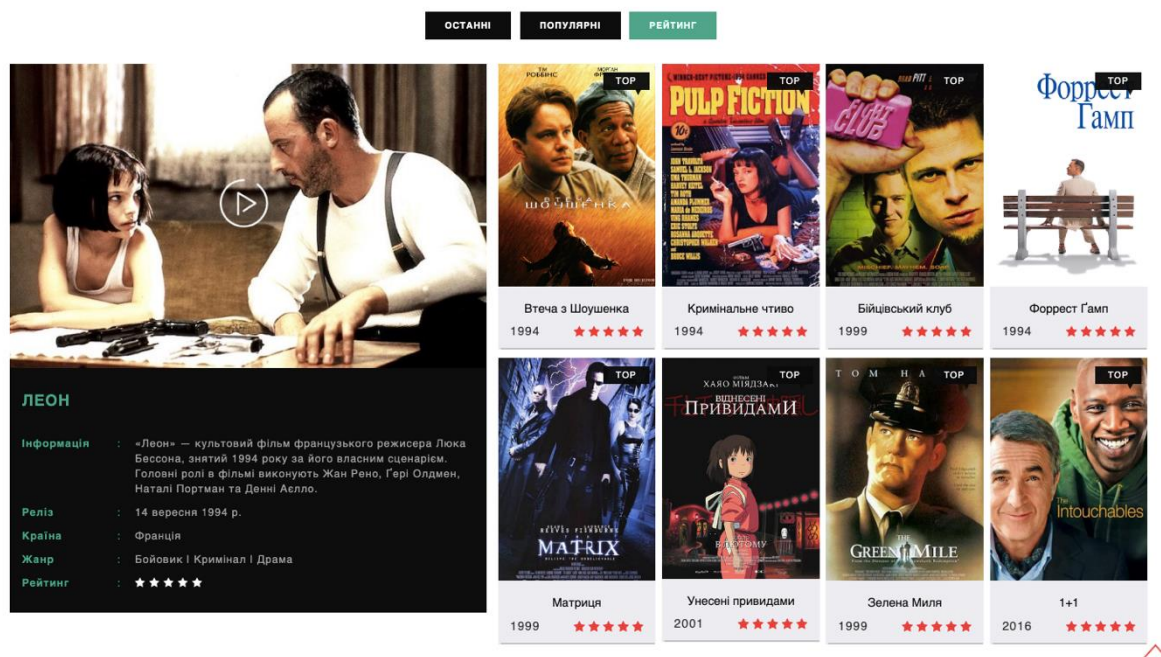
Якщо прокрутити сайт нижче, можемо побачити 3 вкладки з фільмами з сортуванням по року, популярності та рейтингу, згідно написаним запитам SPARQL та інформації екземплярів з створеної онтології.



Мал. 4.5 Останні фільми



Мал. 4.6 Популярні фільми



Мал. 4.7 Фільми за рейтингом

```

☐

```

Лістинг 4.8 Створення кнопок на головній сторінці

```
//фільтрація по останнім рокам
```

```

function filtrbyYearMax(element){
    var elements, year;
    if(element == "last"){
        elements = animation;
        year = document.getElementById('year_max_a');
    }
    else if(element == "film"){

```



```

        elements = film;
        year = document.getElementById('year_max_f');
    }

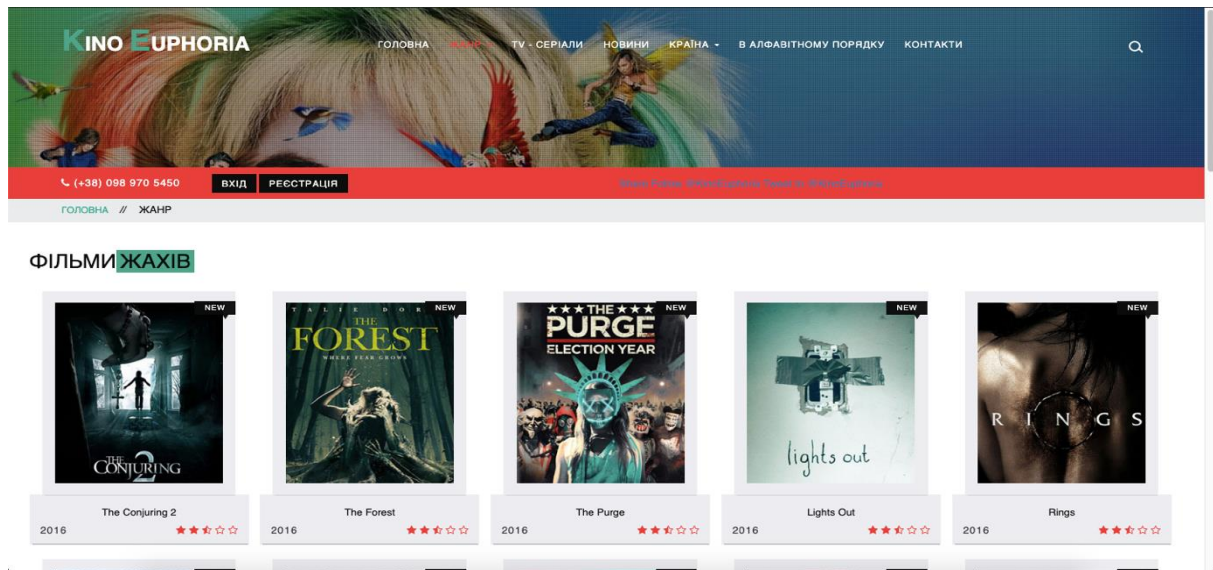
    if(year.value == ""){//перевірка значення
        year.style.border = "1px solid red";
        return;
    }
    else{
        var fields = document.getElementsByClassName("field");
        for (var i = 0; i < fields.length; i++) //зробити блоки не видимими
            fields[i].style.border = "none";
        year = year.value.toLowerCase();
    }

    for (var i = 0; i < elements.length; i++) { //зробити блоки не видимими
        if(elements[i].year < year)
            document.getElementById(elements[i].id).style.display = "none";
    }
    return false;
}

```

Лістинг 4.9 сортування за останніми роками

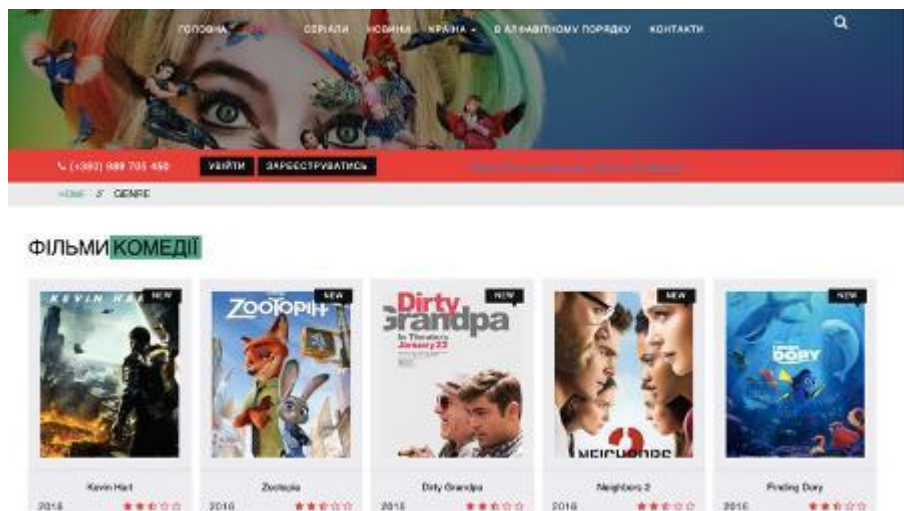
При виборі жанру (наприклад жахи/комедія):



Мал. 4.8 Вікно з жанром жахи



Мал. 4.9 Вибір жанру



Мал. 4.10 Вікно з жанром комедія


```

<h3 class="w3l-inner-h-title">Список Фільмів</h3>
<p class="w3ls_head_para">Натисніть потрібну букву, щоб відсортувати за ім'ям</p>
<div class="bs-example bs-example-tabs" role="tabpanel" data-example-id="togglable-tabs">
<ul id="myTab" class="nav nav-tabs" role="tablist">
<li role="presentation" class="active"><a href="#home" id="home-tab" role="tab" data-toggle="tab" aria-controls="home" aria-expanded="true">Б</a></li>
<li role="presentation"><a href="#a" role="tab" id="a-tab" data-toggle="tab" aria-controls="a">А</a></li>
<li role="presentation"><a href="#b" role="tab" id="b-tab" data-toggle="tab" aria-controls="b">Б</a></li>
<li role="presentation"><a href="#d" role="tab" id="d-tab" data-toggle="tab" aria-controls="d">Д</a></li>

```

Лістинг 4.11 Створення вікна з сортуванням по букві алфавіту

```

function sortByName(array) {
  for (var i = 1; i < array.length; i++) {
    var current = array[i];
    var j = i;
    while (j > 0 && array[j - 1].name > current.name) {
      array[j] = array[j - 1];
      j--;
    }
    array[j] = current;
  }
  return array;
}

```

Лістинг 4.12 Функція сортування по алфавіту

Вікно з контактами:

Мал. 4.12 Вікно з контактами


```

<h3 class="w3l-inner-h-title">Контакти</h3>
<p class="w3ls_head_para">Напишіть нам</p>
<div class="w3_mail_grids">
<form action="#" method="post">
<div class="col-md-6 w3_agile_mail_grid">
<span class="input input--ichiro">
<input class="input__field input__field--ichiro" type="text" id="input-25"
placeholder=" " required="">
<label class="input__label input__label--ichiro" for="input-25">
<span class="input__label-content input__label-content--ichiro">Ваше
ім'я</span></label></span>
<span class="input input--ichiro">
<input class="input__field input__field--ichiro" type="email" id="input-26"
placeholder=" " required="">
<label class="input__label input__label--ichiro" for="input-26">
<span class="input__label-content input__label-content--ichiro">Ваш e-
mail</span></label></span>
<span class="input input--ichiro">
<input class="input__field input__field--ichiro" type="text" id="input-27"
placeholder=" " required="">
<label class="input__label input__label--ichiro" for="input-27">
<span class="input__label-content input__label-content--ichiro">Ваш номер
телефону</span></label></span>

```

Лістинг 4.13 Створення вікна з контактами

Приклад сторінки з інформацією про фільм:



Мал. 4.13 Вікно з обраним фільмом

Висновки

Отже, заплановані цілі були успішно виконані. У ході роботи було проаналізовано існуючі підходи та методи розробки онтологій. Також було проведено аналіз існуючих технологій, середовищ розробки, фреймворків та плагінів. Обрана програма Protege, на мою думку, являється найкращою для створення онтологій. У результаті виконання роботи були розроблені: онтологічна модель – «Фільми», взаємозв'язки між її об'єктами та їх властивості. У дану онтологію увійшло 14 класів і підкласів, 6 властивостей об'єктів, 10 атрибутів класів і 240 екземплярів. Також були створені запити мовою SPARQL, що відображають правильну роботу онтологічної моделі та виводять потрібну інформацію користувачеві. Крім того, у результаті був створений веб-застосунок на основі створеної онтології.

Подальші кроки в розвитку та вдосконаленню продукту: наповнення онтологічної моделі більшою кількістю екземплярів, встановлення залежностей між ними з метою покращення бази знань на тему фільми.

Список використаної літератури

1. Офіційний веб-портал Державної служби інтелектуальної власності України. Архів оригіналу за 16 квітень 2015. Процитовано 29 січень 2015.
2. Gruber T.R. The role of common ontology in achieving sharable, reusable knowledge bases // Principles of Knowledge Representation and Reasoning. Proceedings of the Second International Conference. J.A. Allen, R. Fikes, E. Sandewell – eds. Morgan Kaufmann, 1991, 601-602.
3. Gruber T.R. A translation approach to portable ontologies. Knowledge Acquisition, 5(2): 199-220, 1993.
4. The KAON. – Режим доступу: <http://kaon.semanticweb.org/>
5. The OilEd. – Режим доступу: <http://oiled.man/>
6. The OntoEdit. – Режим доступу: <http://www.ontoprise.de/com/ontoedit.htm>
7. The Ontosaurus. – Режим доступу: <http://www.isi.edu/isd/ontosaurus.html>
8. The OpenСус. – Режим доступу: <http://www.opencyc.org/>
9. The Protege Project. – Режим доступу: <http://protege.stanford.edu>.
10. Briukhov D.O., Shumilov S.S. Ontology Specification and Integration Facilities in a Semantic Interoperation Framework. Proc. of the Second Intern. Workshop ADBIS'95. Moscow, 1995. - P. 195-200.
11. Natalya F. Noy and Deborah L. McGuinness. «Ontology Development 101: A Guide to Creating Your First Ontology». Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880, March 2001.