Министерство образования Республики Беларусь

Учреждение образования

БелорусскиЙ государственный университет

информатики и радиоэлектроники

Факультет компьютерных систем и сетей

Кафедра информатики

|  |
| --- |
| *К защите  допустить*: |
| Заведующая кафедрой информатики |
| \_\_\_\_\_\_\_\_\_\_\_Н. А. Волорова |

Пояснительная записка

к дипломному проекту

на тему

**КОМПОНЕНТЫ СКА ИИ «СОЗДАНИЕ ТЕМАТИЧЕСКИХ ГРАФОВЫХ БД» И «ИНТЕГРАЦИЯ ДОСТУПА К КОМПОНЕНТАМ СИСТЕМЫ»**

БГУИР ДП 1-40 04 01 055 ПЗ

|  |  |  |
| --- | --- | --- |
| Студент |  | А. О. Кулевич |
| Руководитель |  | И. И. Пилецкий |
| Консультанты: |  |  |
| от кафедры информатики |  | И. И. Пилецкий |
| по экономической части |  | Т. А. Рыковская |
|  |  |  |
| Нормоконтролер |  | Н.Н. Бабенко |
|  |  |  |
| Рецензент |  | И. П. Иванов |

Минск 2023

Министерство образования Республики Беларусь

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Учреждение образования  БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | |
| Факультет | | КСиС | | | | | Кафедра | | | | | информатики | | | | | | | | | | | |
| Специальность | | 1-40 04 01 | | | | | Специализация | | | | | | | 00 | | | | | | | | | |
| УТВЕРЖДАЮ | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | |  | | | | | Н.А.Волорова | | | | | |
| « | | | | | | | | | | | | |  | | » | |  | | | | 20 | | г. | |
| **ЗАДАНИЕ** | | | | | | | | | | | | | | | | | | | | | | | |
| **по дипломному проекту студента** | | | | | | | | | | | | | | | | | | | | | | | |
| Кулевича Алексея Олеговича | | | | | | | | | | | | | | | | | | | | | | | |
| (фамилия, имя, отчество) | | | | | | | | | | | | | | | | | | | | | | | |
| 1 Тема проекта: | | | | Компоненты СКА ИИ «Создание тематических графовых бд» и «Интеграция доступа к компонентам системы» | | | | | | | | | | | | | | | | | | | |
| и «Интеграция доступа к компонентам системы» | | | | | | | | | | | | | | | | | | | | | | | |
| утверждена приказом по университету от | | | | | | | « | 17 | | » | марта | | | | | 2023 г. | | | № | 673-c | |
| 2 Срок сдачи студентом законченной работы | | | | | | | | | 01 июня 2023 года | | | | | | | | | | | | | | |
| 3 Исходные данные к проекту: | | | | | Тип операционной системы – ОС Windows; язык | | | | | | | | | | | | | | | | | | |
| программирования – Python; база данных – Neo4j; интегрированная среда | | | | | | | | | | | | | | | | | | | | | | | |
| Разработки – PyCharm. | | | | | | | | | | | | | | | | | | | | | | | |
| Назначение разработки: создать многоцелевой, модифицируемый кластера для | | | | | | | | | | | | | | | | | | | | | | | |
| глубокого анализа данных интернет источников, а именно, для анализа научных | | | | | | | | | | | | | | | | | | | | | | | |
| публикаций. | | | | | | | | | | | | | | | | | | | | | | | |
| 4 Содержание пояснительной записки (перечень подлежащих разработке вопросов) | | | | | | | | | | | | | | | | | | | | | | | |
| Введение | | | | | | | | | | | | | | | | | | | | | | | |
| 1 Анализ литературных источников, прототипов и формирование требований к | | | | | | | | | | | | | | | | | | | | | | | |
| проектируемому программному средству | | | | | | | | | | | | | | | | | | | | | | | |
| 2 Обзор используемых технологий | | | | | | | | | | | | | | | | | | | | | | | |
| 3 Проектирование и реализация приложения | | | | | | | | | | | | | | | | | | | | | | | |
| 4 Программная реализация | | | | | | | | | | | | | | | | | | | | | | | |
| 5 Тестирование программного средства | | | | | | | | | | | | | | | | | | | | | | | |
| 6 Технико-экономическое обоснование | | | | | | | | | | | | | | | | | | | | | | | |
| Заключение | | | | | | | | | | | | | | | | | | | | | | | |
| Список использованных источников | | | | | | | | | | | | | | | | | | | | | | | |
| Приложение А Текст программы | | | | | | | | | | | | | | | | | | | | | | | |
| Приложение Б Иллюстрации работы программы | | | | | | | | | | | | | | | | | | | | | | | |
| 5 Перечень графического материала (с точным указанием наименования) и обозначения вида и типа материала): | | | | | | | | | | | | | | | | | | | | | | |
| Схема базы данных – Плакат – формат А4, лист 1. | | | | | | | | | | | | | | | | | | | | | | |
| Диаграмма вариантов использования – Плакат – формат А4, лист 1. | | | | | | | | | | | | | | | | | | | | | | |
| Взаимодействие компонентов приложения – Плакат – формат А4, лист 1. | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | |
| 6 Содержание задания по технико-экономическому обоснованию | | | | | | | | | | | | | | | | | | | | | | |
| 1 Расчёт затрат на разработку компонентов СКА ИИ «Создание тематических | | | | | | | | | | | | | | | | | | | | | | |
| графовых БД» и «Интеграция доступа к компонентам системы» | | | | | | | | | | | | | | | | | | | | | | |
| 2 Оценка экономической эффективности проекта | | | | | | | | | | | | | | | | | | | | | | |
| Задание выдал | |  | | | / Т. А. Рыковская / | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | |

КАЛЕНДАРНЫЙ ПЛАН

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование этапов дипломного проекта (работы) | Объём этапа в % | Срок выполнения этапа | Примечание |
| Анализ предметной области, разработка технического задания | 15 – 20 | 23.03 – 01.04 |  |
| Разработка функциональных требований, проектирование архитектуры программы | 15 – 20 | 02.04 – 08.04 |  |
| Разработка схемы программы, алгоритмов, схемы данных | 15 – 20 | 09.04 – 15.04 |  |
| Разработка программного средства | 15 – 20 | 16.04 – 29.04 |  |
| Тестирование и отладка | 10 | 30.04 – 13.05 |  |
| Оформление пояснительной записки и графического материала | 20 | 14.05 – 31.05 |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Дата выдачи задания | 23 марта 2023 г. | | Руководитель | |  | / И. И. Пилецкий / | |
| Задание принял к исполнению | |  | | / А. О. Кулевич / | | |

**РЕФЕРАТ**

КОМПОНЕНТЫ СКА ИИ «СОЗДАНИЕ ТЕМАТИЧЕСКИХ ГРАФОВЫХ БД» И «ИНТЕГРАЦИЯ ДОСТУПА К КОМПОНЕНТАМ СИСТЕМЫ»: дипломный проект / А. О. Кулевич – Минск : БГУИР, 2023, – п.з. – 90 с., схем/плакатов – 6 л. формата А4.

Ключевые слова: WEB-САЙТ, ИНТЕРНЕТ-ИСТОЧНИКИ АНАЛИЗ, NEO4J, ГРАФОВАЯ БАЗА ДАННЫХ, RDF, PYTHON, PYCHARM

Целью проекта является проектирование и разработка компонента системы комплексного анализа данных интернет-источников (СКАД ИИ), который позволяет по содержанию сайта построить графовую БД, получить граф знаний для анализа данных конкретного тематического сайта, а также на практике продемонстрировать его работу.

При разработке приложения использовался следующий набор технологий: NEO4J, NEOSEMANTICS(N10S), APOC, PYTHON, DJANGO, HTML, CSS,FIGMA

В первом разделе данной работы осуществляется анализ востребованности разрабатываемого приложения, анализ существующих аналогов и формирование требований к приложению.

Второй раздел содержит информацию о технологиях, используемых для разработки приложения.

В третьем разделе приведён процесс проектирования архитектуры приложения, его бэкенда, проектирования базы данных.

Четвертый раздел содержит описание процесса разработки функционала приложения и бэкенда.

Пятый раздел содержит описание процесса тестирования основной функциональности приложения.

В шестом разделе приведено технико-экономическое обоснование эффективности разработки приложения.

В разделе выводы содержатся итоги проделанной работы по данному дипломному проекту.

Дипломный проект является завершенным, поставленная задача решена в полной мере, присутствует возможность дальнейшего развития приложения и наращивания его функционала.

**АННОТАЦИЯ**

на дипломный проект «Компоненты СКА ИИ «Создание

тематических графовых бд» и «Интеграция доступа к компонентам

системы»» студента учреждения образования «Белорусский государственный университет информатики и радиоэлектроники» Кулевича А. О. (ВЫРАВНЯТЬ!!!)

Темой дипломного проекта является компоненты СКА ИИ «Создание тематических графовых бд» и «Интеграция доступа к компонентам системы».

Во введении определена актуальность темы дипломного проекта, описана область применения.

В первом разделе данной работы осуществляется анализ востребованности разрабатываемого приложения, анализ существующих аналогов и формирование требований к приложению.

Второй раздел содержит информацию о технологиях, используемых для разработки приложения.

В третьем разделе приведён процесс проектирования архитектуры приложения, его бэкенда, проектирования базы данных.

Четвертый раздел содержит описание процесса разработки функционала приложения и бэкенда.

Пятый раздел содержит описание процесса тестирования основной функциональности приложения.

В шестом разделе приведено технико-экономическое обоснование эффективности разработки приложения.

В разделе выводы содержатся итоги проделанной работы по данному дипломному проекту.

АНТИПЛАГИАТ КАРТИНКА

**ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ, СИМВОЛОВ И ТЕРМИНОВ**

В настоящей пояснительной записке используются следующие определения и сокращения.

БД – база данных.

СУБД – система управления базами данных.

API – Application Programming Interface – описание способов (набор классов, процедур, функций, структур или констант), которыми одна компьютерная программа может взаимодействовать с другой программой.

Плагин – независимо компилируемый программный модуль, динамически подключаемый к основной программе и предназначенный для расширения и/или использования её возможностей. Плагины обычно выполняются в виде библиотек общего пользования.

RDF –

**СОДЕРЖАНИЕ**

Оглавление

[Пояснительная записка 1](#_Toc135082572)

[**ЗАДАНИЕ** 2](#_Toc135082573)

[**по дипломному проекту студента** 2](#_Toc135082574)

[1. АНАЛИЗ АНАЛОГОВ, ПРЕДМЕТНОЙ ОБЛАСТИ И ФОРМИРОВАНИЕ ТРЕБОВАНИЙ К ПРОЕКТУ 10](#_Toc135082575)

[1.1 Постановка задачи 10](#_Toc135082576)

[1.2 Анализ предметной области 12](#_Toc135082577)

[1.3 Анализ аналогов программного продукта 16](#_Toc135082582)

[1.4 Формирование требований 19](#_Toc135082587)

[2. ОБЗОР ИСПОЛЬЗУЕМЫХ ТЕХНОЛОГИЙ 22](#_Toc135082594)

[2.1 REST API 22](#_Toc135082596)

[2.2 Django 22](#_Toc135082597)

[2.3 HTML 23](#_Toc135082598)

[2.4 CSS 23](#_Toc135082599)

[2.5 Neo4j 24](#_Toc135082600)

[2.6 Figma 25](#_Toc135082603)

[3. ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ 27](#_Toc135082604)

[3.1 Методология применения построения прототипа компонента СКА ИИ создание графовой БД 27](#_Toc135082606)

[3.2 Функциональная модель программного средства 44](#_Toc135082610)

[3.3 Прототип пользовательского интерфейса 47](#_Toc135082613)

[4. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ 56](#_Toc135082614)

[4.1 Создание и конфигурация базы данных 56](#_Toc135082616)

[4.2 Настройка Neo4j для использования данных RDF 60](#_Toc135082617)

[4.3 Заполнение графовой БД тестовыми данными 61](#_Toc135082618)

[4.4 Cypher запросы для манипулирования данными, используемые на сайте 63](#_Toc135082619)

[5. ТЕСТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА 65](#_Toc135082620)

[6. ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ КОМПОНЕНТОВ СКА ИИ «СОЗДАНИЕ ТЕМАТИЧЕСКИХ ГРАФОВЫХ БД» И «ИНТЕГРАЦИЯ ДОСТУПА К КОМПОНЕНТАМ СИСТЕМЫ» 76](#_Toc135082621)

[**6.1** **Краткая характеристика программного продукта** 76](#_Toc135082622)

[**6.2 Расчет затрат на разработку и цена программного обеспечения, созданного по индивидуальному заказу** 77](#_Toc135082623)

[**6.3 Оценка результата от продажи программного обеспечения** 80](#_Toc135082631)

[**6.4 Расчет показателей эффективности инвестиций в разработку программного обеспечения** 81](#_Toc135082632)

**ВВЕДЕНИЕ**

Одна из больших технических задач — это получение данных из конкретного сайта. Но, еще более сложная проблема — это анализ данных размещенных на данном сайте. Что приводит к появлению ряда новых возможностей моделирования структуры сайта, одна из которых - возможность предложить простой способ представления свойств отношений в виде графовой модели. Что позволяет анализировать свойства сайта с помощью графовых моделей. Многие сайты формально поддерживает тройки RDF, в которых тройка имеет вид «субъект — предикат — объект» и называется триплетом (тройкой). Множество RDF-утверждений образует ориентированный граф, в котором вершинами являются субъекты и объекты, а рёбра отображают отношения.

С помощью графовых технологий можно преобразовать представление сайта в графовую БД со свойствами. Что позволяет анализировать свойства сайта. Последние данные Gartner трендов в области ИТ «Gartner Identifies Top 10 Data and Analytics Technology Trends for 2021» показывают возрастающую роль графовых технологий, так к 2025 году графовые технологии будут использоваться в 80% инноваций в области данных и аналитики по сравнению с 10% в 2021 году, что будет способствовать быстрому принятию решений в организации .

Графы — один из самых мощных и гибких способов представления данных. Они обладают большой выразительной силой, что позволяет их применять для моделирования различных физических систем (например, дороги, трубопроводы, электросети, воздушные авиалинии, логистику и т.д.) и социальные сетей (например, LinkedIn, MySpace, Facebook, ВКонтакте, Академические и т.д.). Граф — это универсальная и выразительная структура, позволяющая моделировать всевозможные структуры данных, например, молекулярные взаимодействия и биологические процессы, фармакология, клинические, медицинские и научные публикации. Графы позволяют моделировать сценарии постройки сложных объектов (например, самолеты, ракеты, корабли и т.д.). В настоящее время графовые модели и графовы БД широко применяются во всех областях моделирования искусственного интеллекта.

Графованые базы по сравнению с другими NoSQL БД обладает свойствами OLTP и OLAP. Графовые технологии обеспечивают организации транзакционных графовых хранилищ, интеллектуальный анализ данных и аналитическую обработку данных в реальном времени, поддерживают транзакции ACID, что не обеспечивает ни одна NoSQL БД. Графовые технологии являются основой для построения интеллектуальных приложений, для применения алгоритмов искусственного интеллекта. Важным отличие графовых баз данных является то, что они явно описывают зависимости между узлами данных, в то время как другие базы данных связывают данные неявными связями. Для NoSQL вместо свойств ACID ввели свойства BASE, которые значительно слабее гарантий ACID, и между ними нет прямого соответствия.

В сложных реляционных базах данных, где количество сущностей может состоять из нескольких сот (например, 300 или 400 сущностей), связность сущностей приводит к увеличению соединений, которые снижают производительность и затрудняют внесение в базу данных новых обновлений. И как правило, выполнение сложных запросов будет медленными, а сама БД со временем может полностью деградировать.

Совместное применение графовых технологий, методов и алгоритмов машинного обучения позволяет получать скрытые зависимости и выполнять предиктивный анализ информации, получать ответы в режиме реального времени, реализовывать алгоритмы искусственного интеллекта.

Графовые базы данных позволяют хранить сущности и отношения между ними. Сущности моделируются узлами (nodes), которые имеют свойства. Узел интерпретируется как экземпляр объекта в приложении. Отношения моделируются ребрами (relationships или edges), которые могут иметь свойства. Neo4j БД отдельно хранит в специальном жестком формате структуру узлов и отношений, а данные (свойства) определяются как пара ключзначение. Такое решение в базе Neo4j позволяет извлекать данные без обхода графа, с определенного узла (смещение).

1. АНАЛИЗ АНАЛОГОВ, ПРЕДМЕТНОЙ ОБЛАСТИ И ФОРМИРОВАНИЕ ТРЕБОВАНИЙ К ПРОЕКТУ
   1. **Постановка задачи**

Получение данных из конкретного сайта и их анализ являются важными задачами в современном информационном мире. Это приводит к появлению ряда новых возможностей моделирования структуры сайта, одна из которых — возможность предложить простой способ представления свойств отношений в виде графовой модели, которая позволяет анализировать свойства сайта, а именно: найти наиболее популярные публикации и экспертов определённой предметной области.

**Целью** дипломного проекта является разработка прототипа компонента системы комплексного анализа данных интернет-источников (СКАД ИИ), который позволяет по содержанию сайта построить графовую БД, получить граф знаний для анализа данных конкретного тематического сайта, а также на практике продемонстрировать его работу. Т.к. приложение должно предоставлять конечному пользователю интерфейс для взаимодействия, необходимо разработать веб-сайт для демонстрации применимости вышеописанного алгоритма.

**Назначением** дипломного проекта является создание многоцелевого, модифицируемого кластера для глубокого анализа данных интернет источников, а именно, для анализа научных публикаций.

Для реализации проекта необходимо решить **следующие задачи:**

а) изучить возможности графовой системы управления базами данных Neo4j и плагины APOC и Neosemantics (n10s);

б) изучить построение графа свойств в Neo4j на основе данных RDF, Turtle и данных, встроенные в веб-страницы (JSON);

в) продумать структуру базы данных;

г) подготовить данные из открытых интернет-источников и привести их к подходящему виду;

д) определить функциональные требования к приложению;

е) продумать схему интерфейса веб части проекта;

ж) определить API приложения;

з) продумать способ связи веб-части проекта и приложения, содержащего алгоритм определения популярности;

и) разработать технологию создания тематических графовых баз данных и графов знаний;

й) разработать IT платформу для создания СКА ИИ, которая позволит объединить в себе создание тематических графовых баз данные и их анализ;

к) разработать платформу, позволяющую устанавливать соединение с графовой базой данных, отображать статьи и авторов по степени их популярности, производить быстрый поиск данных;

л) произвести тестирование проекта.

Приложение разрабатывается при помощи фреймворка Django Rest Framework с использованием языка Python, интерфейс пользователя реализуется с помощью языка разметки HTML и каскадных таблиц стилей CSS. В качестве графовой системы управления базами данных используется Neo4j. Загрузка и манипуляция данными будут реализованы за счёт предоставляемых Neo4j расширений, таких как APOC (Awesome Procedures on Cypher) и Neosemantics(n10s), а также вспомогательных функций, написанных на языке Python.

* 1. Анализ предметной области

Одним из сложных современных направлений является представление знаний с помощью специальных глобальных словарей предметных областей, мета-описаний и специальных языков, и методологий их применения.

Многие сайты используют специальную технику описания ресурса Resource Description Framework (RDF, «среда описания ресурса»). RDF представляет собой абстрактную модель, обеспечивающую способ разбиения знаний на дискретные части и позволяет обмениваться информацией. Импорт и экспорт RDF может быть в нескольких форматах (Turtle, N-Triples, JSON-LD, RDF / XML, TriG и N-Quads, TriG \*).

Ресурсом в RDF может быть любая сущность — как информационная (например, веб-сайт или изображение), так и неинформационная (например, человек, город или некое абстрактное понятие). Утверждение, высказываемое о ресурсе, имеет вид «субъект — предикат — объект» и называется триплетом (тройкой). Например, утверждение «ель зеленого цвета» в RDF-терминологии можно представить следующим образом: субъект — «ель», предикат — «имеет цвет», объект — «зеленый». Для обозначения субъектов, отношений и объектов в RDF используются URI (универсальный идентификатор ресурса), на рисунке 1.1 приведена схема триплета RDF.

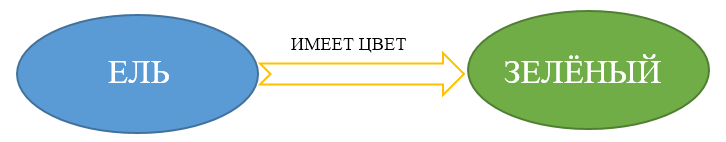


Рисунок 1.1 – Триплет RDF

Множество RDF-утверждений образует ориентированный граф, в котором вершинами являются субъекты и объекты, а рёбра отображают отношения.

Что же касается веб-сайтов, которые хранят данные в JSON-LD, то в коде страницы можно найти нужный нам скрипт JSON-LD. Если он присутствует, то данный сайт можно сериализовать в графовую базу данных способом, описанным в данной работе, если же такого скрипта нет, то понадобится искать другой способ для сериализации данных веб-сайта.

Кроме RDF и JSON, описываемых в курсовой работе, можно сериализовать и другие типы данных. Например, данные на общедоступном популярном сайте Wikipedia хранятся в формате Turtle и тоже могут быть сериализованы в графовую базу данных, для сайта Wikidata информация о данных хранится на самом сайте.

* + 1. Словари предметных областей

Представление знаний с помощью специальных глобальных словарей предметных областей позволяет с помощью специального плагина Neosemantics для графовой БД Neo4j использовать RDF и связанные с ним словари, такие как OWL, RDFS, SKOS и другие для интеграции с компонентами, генерирующими / потребляющими RDF.

RDF словари: FOAF (Friend-of-a-Friend), SKOS (Simple Knowledge Organization System), DC (Dublin Core), BIBO (Bibliographic Ontology), SIOC (Semantically-Interlinked Online Communities), DOAP (Description of a Project), Music Ontology.

RDF словари и онтологии: OWL (Web Ontology Language) – язык представления веб-онтологий, основан на RDF и RDFS.

SKOS (Simple Knowledge Organization System).

Так пример представления знаний c некоторого сайта, см. ниже, приведен на рисунке 1.2.

ex:psycholing rdf:type skos:Concept;<br />

skos:prefLabel "Психолингвистика”<br />

skos:altLabel "Психология языка»<br />

<skos:narrower rdf:resource=

"http://id.loc.gov/authorities/subjects/sh85125403"/>

<skos:exactMatch rdf:resource=

"http://ex.tw/auth/ps654678"/>

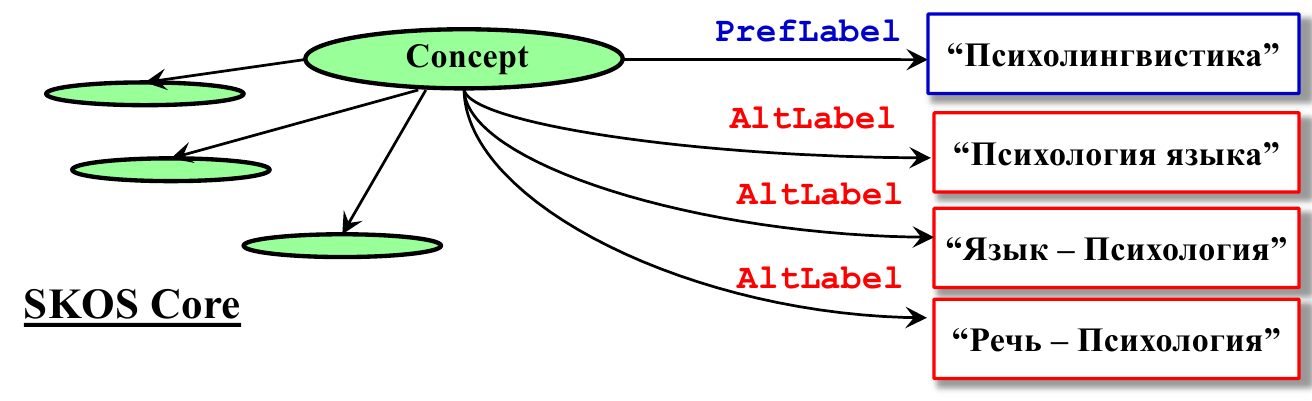


Рисунок 1.2 – Графическое представление данных сайта

* + 1. Правила преобразования троек RDF в графовую БД

Граф RDF - это набор триплетов или операторов (субъект, предикат, объект), где и субъект, и предикат являются ресурсами, а объект может быть либо другим ресурсом, либо литералом. Единственная особенность литералов заключается в том, что они не могут быть предметом других утверждений. В древовидной структуре мы бы назвали их листовыми узлами. Также следует помнить, что ресурсы однозначно идентифицируются URI.

Существует три основных правила сериализации троек RDF («субъект — предикат — объект») в графовую БД.

Первое правило. Субъекты троек отображаются на узлы в Neo4j.

Узел в Neo4j, представляющий ресурс RDF, будет помечен :Resource и будет иметь свойство uri с URI ресурса.

(S,P,O) => (:Resource {uri:S})...

Второе правило. Предикаты троек отображаются в свойствах узла в Neo4j, если объект тройки является литералом.

(S,P,O) && isLiteral(O) => (:Resource {uri:S, P:O})

Третье правило. Предикаты троек отображаются на отношения в Neo4j, если объект тройки является ресурсом.

(S,P,O) && !isLiteral(O) => (:Resource {uri:S})-[:P]->(:Resource {uri:O})

С помощью модулей библиотеки Neo4j можно множество RDF-утверждений преобразовать в графовую БД со свойствами, что позволяет построить прототип двух компонент системы анализа данных Интернет источников, компонента скачивания данных и компонента построения графовой базы данных и графа знаний.

Для доступа к данным таких сайтов можно использовать специально разработанный язык SPARQL Protocol and RDF Query Language. К примеру, с помощью следующего запроса, представленного в листинге 1.1, можно достать данные об архитектурных строениях разных стилей.

WITH ' PREFIX sch: <http://schema.org/>

CONSTRUCT{ ?item a sch:Monument;

sch:name ?monumentName;

sch:location ?location;

sch:img ?imageAsStr;

sch:ARCHITECTURE ?architecture.

?architecture a sch:Architecture;

sch:name ?architectureName }

WHERE { ?item wdt:P31 wd:Q4989906 .

?item wdt:P17 wd:Q29 .

?item rdfs:label ?monumentName .

filter(lang(?monumentName) = "en")

?item wdt:P625 ?location .

?item wdt:P149 ?architecture .

?architecture rdfs:label ?architectureName .

filter(lang(?architectureName) = "en")

?item wdt:P18 ?image .

bind(str(?image) as ?imageAsStr) } limit 100 ' AS sparql CALL n10s.rdf.preview.fetch(

"https://query.wikidata.org/sparql?query=" +

apoc.text.urlencode(sparql),"JSON-LD",

{ headerParams: { Accept: "application/ld+json"} ,

handleVocabUris: "IGNORE"})

YIELD nodes, relationships

RETURN nodes, relationships

Листинг 1.1 – Запрос для сериализации данных об архитектурных строениях и стилях

Результат выполнения запроса представлен на рисунке 1.3.

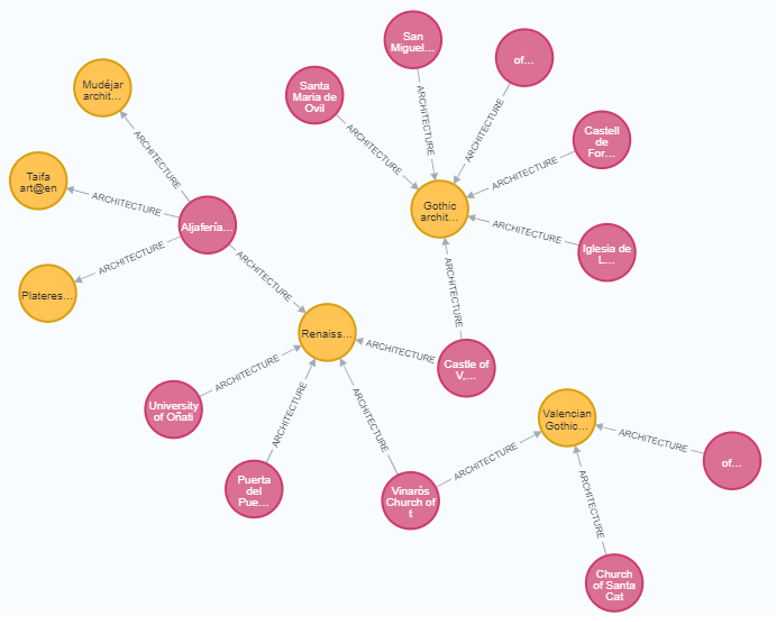


Рисунок 1.3 – Результат выполнения запроса

Розовыми узлами показаны различные архитектурные строения, а жёлтыми узлами являются архитектурные стили. Можно заметить, что некоторые архитектурные строения имеют несколько стилей.

* + 1. Система комплексного анализа данных интернет-источников

«Система комплексного анализа данных интернет-источников» (СКАД ИИ) – инструмент, который позволяет анализировать большие объемы данных из интернет-источников в области научных исследований и предназначен для сбора информации о публикациях, в том числе и о научных, построения графа знаний, что дает возможность определять экспертов предметной области, тематики их работ, их взаимосвязи, а также определять передовые научные направления.

Система позволяет находить экспертов (авторитетов) в предметной области и выдавать оценку их рейтинга влиятельности.

Основными компонентами построения системы анализа данных Интернет источников являются: компонент получения данных с интернет источников; компонент графовая БД и граф знаний; а также компонент извлечения свойств из графовой БД и их анализ с помощью алгоритмов ML. Компонент извлечения свойств из графовой БД обладает дополнительной функциональностью, может использовать технологию включений (эмбеддинга), что позволяет построить векторы свойств меньшей размерности для более глубокого анализа данных. Графовые включения – это методология представления свойств сущностей (узлов) и свойств отношений в графе как вектор свойств.

Результаты могут быть использованы как для прикладных целей, например, определения наиболее перспективных и актуальных научных направлений с учетом конкретной специфики (для конкретной организации или страны, с учетом имеющегося задела и т. Д.), определения экспертов в заданных предметных областях (для приглашения специалистов или формирования команды исследователей), так и для научно-образовательной цели – подготовки специалистов по анализу больших объемов неструктурированных данных (Data Scientist) на основе разработки и оптимизации аналитических ML-алгоритмов, применяемых в системе.

* + 1. Веб-сайт

Веб-сайт – одна или несколько логически связанных между собой веб-страниц, содержащих контент сервера. Сайт в Интернете представляет собой массив связанных данных, имеющий уникальный адрес и воспринимаемый пользователями как единое целое. Веб-сайты называются так, потому что доступ к ним происходит по протоколу HTTP.

Веб-сайт как система электронных документов (файлов данных и кода) может принадлежать частному лицу или организации и быть доступным в компьютерной сети под общим доменным именем и IP-адресом или локально на одном компьютере.

Задача сайта – предоставить материалы в полном объеме по одному определенному предмету. Чтобы вовлечь посетителя в чтение таким образом и заставить как можно дольше находился на них.

В данном дипломном проекте веб-сайт является интегрирующим компонентом для других компонент: «Создание тематических графовых БД», «Интеграция доступа к компонентам системы» и «Интеграция доступа к данным системы и выдачи репортов».

* 1. Анализ аналогов программного продукта

Проведем анализ аналогов, чтобы более точно сформулировать требования к разрабатываемому программному средству. Существует множество аналогов, но для рассмотрения возьмём самые популярные.

* + 1. КиберЛенинка

КиберЛенинка – это научная электронная библиотека открытого доступа (Open Access), основными задачами которой является популяризация науки и научной деятельности, общественный контроль качества научных публикаций, развитие междисциплинарных исследований, современного института научной рецензии, повышение цитируемости российской науки и построение инфраструктуры знаний. КиберЛенинка строится на основе парадигмы открытой науки (Open Science) (рисунок 1.4).



Рисунок 1.4 – Главная страница КиберЛенинки

* + 1. Semantic Scholar

Semantic Scholar — поисковая интернет-платформа, разработанная в Институте искусственного интеллекта Аллена. Проект был запущен в 2015 году. Поиск научных публикаций производится с поддержкой искусственного интеллекта для статей в научных журналах. Поисковый сервис комбинирует машинное обучение, обработку естественного языка и машинного зрения, чтобы добавить слой семантического анализа к традиционным методам анализа цитирования. Semantic Scholar выделяет наиболее важные статьи, а также связи между ними. Сайт включает в себя более 200 миллионов статей на различную тематику (рисунок 1.5).

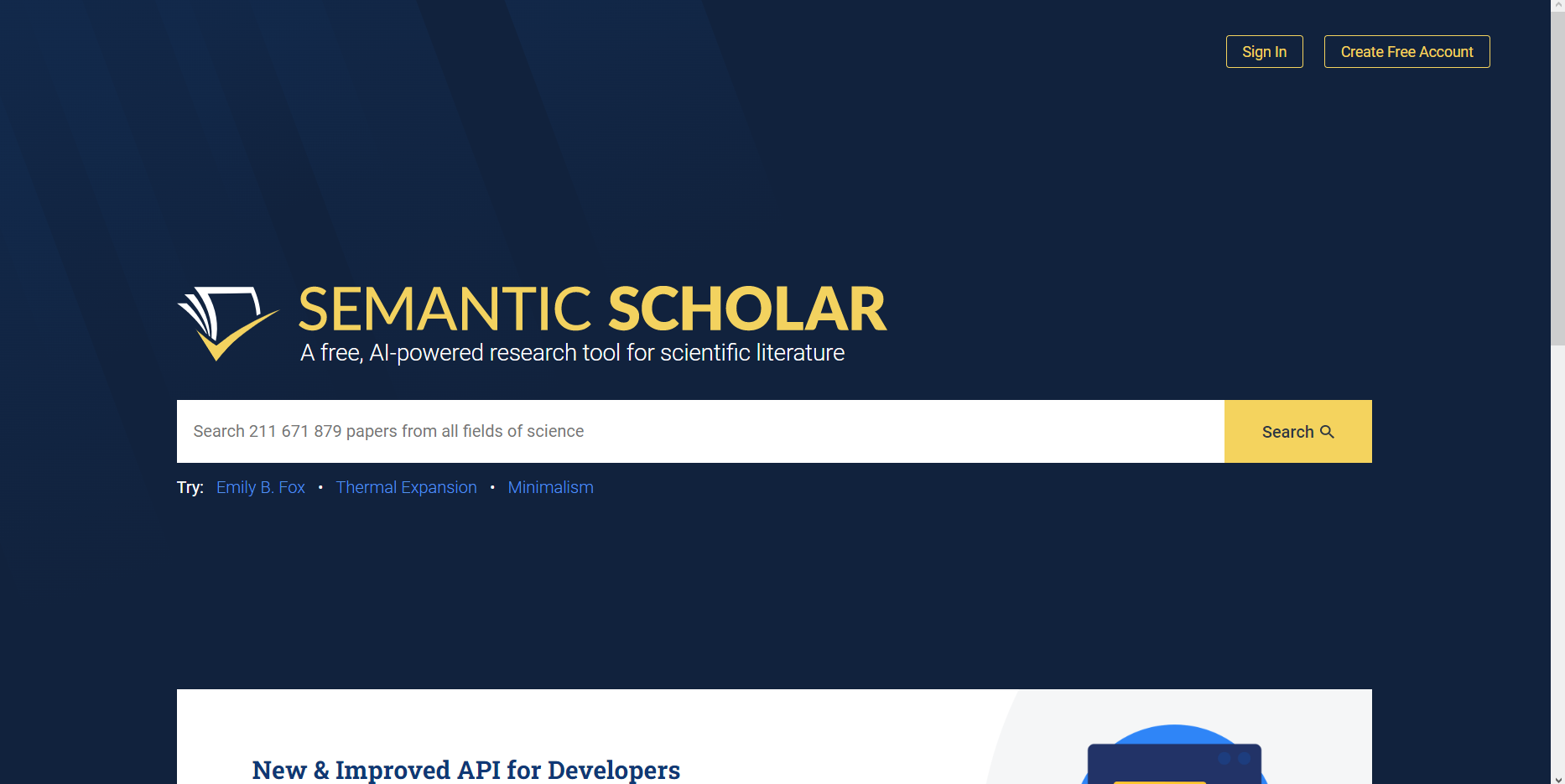


Рисунок 1.5 – Главная страница сайта Semantic Scholar

* + 1. SpringerOpen

SpringerOpen — сайт, содержащий коллекцию электронных журналов по различным областям знаний. Журналы SpringerOpen доступны сразу после публикации и открыты для всех желающих. Издательство рассматривает это как важную коммуникативную составляющую научных исследований. Публикации, представленные в SpringerOpen, подвергаются самой высокой и строгой экспертной оценке. Многие журналы SpringerOpen активно цитируются и имеют Импакт-фактор. За большинством журналов SpringerOpen следит компания Thomson Reuters и они учтены, соответственно, в базе данных Journals Citation Reports (рисунок 1.6).

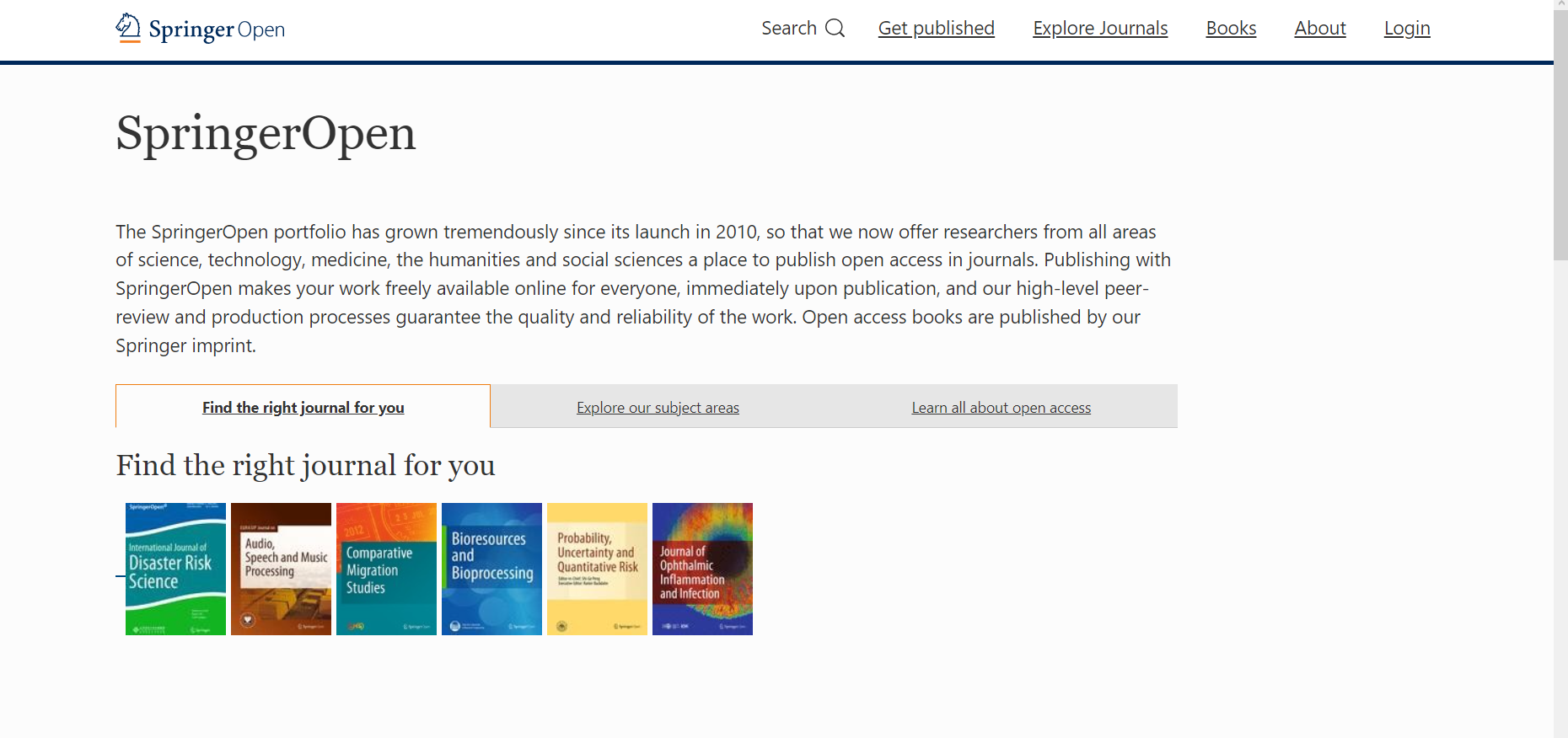


Рисунок 1.6 – Главная страница сайта SpringerOpen

* + 1. Medium

Medium — платформа для социальной журналистики. Сервис запущен в августе 2012 года сооснователями Twitter – Эваном Уильямсом и Бизом Стоуном. В Медиуме 96 разделов по интересам. Авторы пишут о веб-дизайне, маркетинге, программировании, управлении проектами, есть статьи об искусстве, кино, образовании, деньгах, людях и животных. На площадке можно вести авторскую колонку, создать блог издания или компании. Ежемесячно ресурс посещают около двухсот миллионов человек (по данным Similarweb), большинство из них живут в США. Платформа англоязычная, отдельного русского раздела нет. Содержит миллионы статей и публикаций на различную тематику (рисунок 1.7).

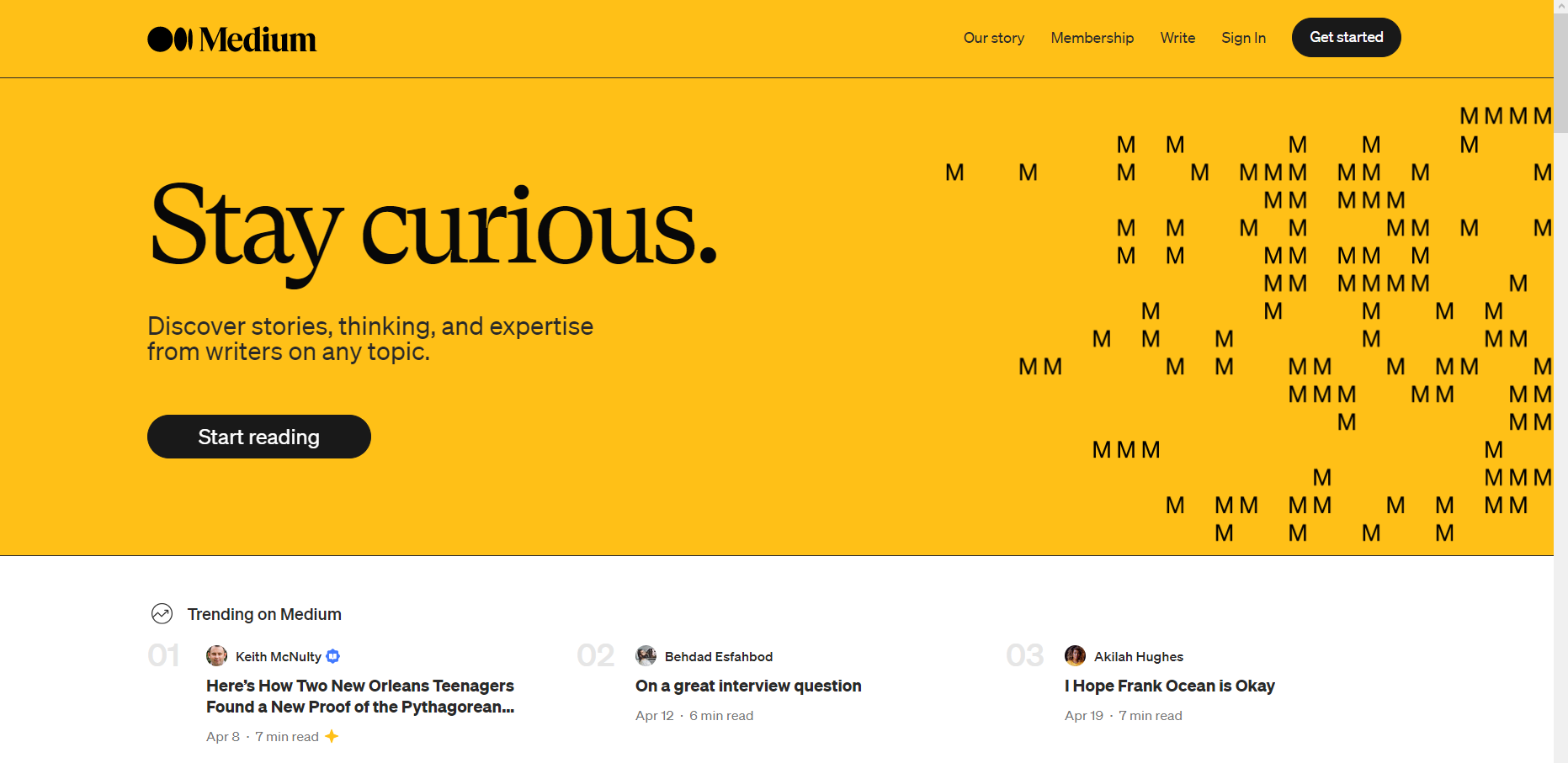


Рисунок 1.7 – Главная страница сайта Medium

Был проведёт анализ аналогов длятого, чтобы выявить их недостатки и преимущества и учесть их при разработке веб-сайта.

Таким образом, будет разработано мультиплатформенное решение для различных предметных областей, с помощью которого можно будет быстро брать данные и информацию о предметной области с крупных систем и на базе этого строить предметную область для дальнешего её анализа.

* 1. Формирование требований

После анализа предметной области, а также обзора уже существующих аналогов необходимо сформулировать требования к проектируемому программному средству.

* + 1. Назначение проекта

Назначением проекта является создание многоцелевого, модифицируемого кластера для глубокого анализа данных интернет источников, а именно, для анализа научных публикаций.

* + 1. Функциональные требования

Программное средство должно поддерживать следующие основные функции:

* регистрация, аутентификация, авторизация;
* поддержка системы ролей;
* просмотр информации о публикациях различных предметных областей и авторах;
* просмотр публикаций различных предметных областей;
* просмотр параметров некоторой предметной области;
* поиск публикации по названию и автора по имени;
* выполнение произвольного запроса к графу знаний;
* загрузка данных с тематических сайтов в базу данных разными способами.
* управление анализом данных до нескольких сот тысяч документов;
  + 1. Требования к входным данным

Входные данные для программного средства должны быть представлены в виде вводимого пользователем с клавиатуры текста, документов соответствующего формата и содержания и опций, предоставляемых пользовательским интерфейсом приложения.

В бизнес-логике программного средства должны быть реализованы проверки входных данных на корректность, и в случае их невалидности, пользователь должен получать соответствующее уведомление с просьбой ввести данные необходимого формата.

* + 1. Требования к выходным данным

Выходные данные должны быть представлены посредством отображения информации при помощи различных элементов реализованного и доступного пользовательского интерфейса.

* + 1. Требования к надежности

Для надежного функционирования программного средства от заказчика, а также команды поддержки приложения необходимо выполнение следующего комплекса административных и организационно-технических мероприятий:

* выполнение рекомендаций Министерства труда и социальной защиты РБ, изложенных в Постановлении от 23 марта 2011 г. «Об утверждении Норм времени на работы по обслуживанию персональных электронно-вычислительных машин, организационной техники и офисного оборудования»;
* выполнение требований ГОСТ 31078-2002 «Защита информации. Испытания программных средств на наличие компьютерных вирусов»;
* организация бесперебойного питания.

Время на восстановление работоспособности системы после отказа, вызванного фатальным сбоем операционной системы или техническими неисправностями используемых аппаратных средств, не должно превышать время, необходимое для устранения текущих неисправностей или переустановку операционной системы с последующим перезапуском программного средства.

Время на восстановление работоспособности системы после отказа, вызванного нефатальным сбоем операционной системы, не должно превышать время, необходимое для перезагрузки операционной системы и запуска программного средства.

Поскольку одна из основных причин отказа приложения – это некорректное взаимодействие пользователя с операционной системой, необходимо обеспечить возможность эксплуатации программного средства конечным пользователем без предоставления ему административных привилегий.

Также для надежного функционирования программного средства обязательно наличие необходимого уровня опыта и квалификации у конечного потребителя.

* + 1. Требования к аппаратному обеспечению клиентской части

Клиентская часть программного средства должна функционировать на ЭВМ со следующими минимальными характеристиками:

* процессор Intel Сore с тактовой частотой 2 ГГц и более;
* оперативная память 4 Гб и более;
* возможность выхода в сеть Интернет;

Для корректной работы программного средства необходим один из следующих браузеров с соответствующей минимальной версией:

* Google Chrome 70;
* Opera 58;
* Mozilla Firefox 66;
* Apple Safari 12.0;
* Microsoft Edge 44.

Таким образом, в данной главе был проанализирована предметная область и сделаны выводы об актуальности разрабатываемого приложения; выполнен анализ аналогов веб-сайта, в результате которого были выявлены их преимущества и недостатки. На основании полученных данных были описаны варианты реализации этапов работы приложения и требования к программному средству. Результаты данного анализа позволили заложить основу для определения технологий, которые необходимо использовать для данной разработки.

1. ОБЗОР ИСПОЛЬЗУЕМЫХ ТЕХНОЛОГИЙ
2. 1. REST API

REST – это способ получить доступ к ресурсам, которые находятся в определённой среде. Например, у вас может быть сервер с важными документами или фотографиями. Всё это – ресурсы. Если клиенту, скажем, веб-браузеру, нужны какие-то из этих ресурсов, ему необходимо отправить 35 запрос на сервер для получения доступа к ним. REST определяет, как может осуществляться доступ к этим ресурсам.

* 1. Django

Django — это высокоуровневая веб-инфраструктура Python, которая способствует быстрой разработке и чистому, прагматичному дизайну. Созданный опытными разработчиками, он берет на себя большую часть хлопот веб-разработки, поэтому вы можете сосредоточиться на написании своего приложения, не изобретая велосипед. Это бесплатно и с открытым исходным кодом.

Сайт на Django строится из одного или нескольких приложений, которые рекомендуется делать отчуждаемыми и подключаемыми. Это одно из существенных архитектурных отличий этого фреймворка от некоторых других (например, Ruby on Rails). Один из основных принципов фреймворка — DRY (англ. Don't repeat yourself).

Также, в отличие от других фреймворков, обработчики URL в Django конфигурируются явно при помощи регулярных выражений.

Для работы с базой данных Django использует собственный ORM, в котором модель данных описывается классами Python, и по ней генерируется схема базы данных.

Архитектура Django похожа на «Модель-Представление-Контроллер» (MVC). Контроллер классической модели MVC примерно соответствует уровню, который в Django называется Представление (англ. View), а презентационная логика Представления реализуется в Django уровнем Шаблонов (англ. Template). Из-за этого уровневую архитектуру Django часто называют «Модель-Шаблон-Представление» (MTV).

Первоначальная разработка Django как средства для работы новостных ресурсов достаточно сильно отразилась на его архитектуре: он предоставляет ряд средств, которые помогают в быстрой разработке веб-сайтов информационного характера. Так, например, разработчику не требуется создавать контроллеры и страницы для административной части сайта, в Django есть встроенное приложение для управления содержимым, которое можно включить в любой сайт, сделанный на Django, и которое может управлять сразу несколькими сайтами на одном сервере. Административное приложение позволяет создавать, изменять и удалять любые объекты наполнения сайта, протоколируя все совершённые действия, и предоставляет интерфейс для управления пользователями и группами (с пообъектным назначением прав).

В дистрибутив Django также включены приложения для системы комментариев, синдикации RSS и Atom, «статических страниц» (которыми можно управлять без необходимости писать контроллеры и представления), перенаправления URL и другое.

* 1. HTML

Интерфейс веб-сайта реализуется с помощью HTML. HTML – стандартизированный язык разметки документов для просмотра веб-страниц в браузере. Веб-браузеры получают HTML документ от сервера по протоколам HTTP/HTTPS или открывают с локального диска, далее интерпретируют код в интерфейс, который будет отображаться на экране монитора.

Элементы HTML являются строительными блоками HTML страниц. С помощью HTML разные конструкции, изображения и другие объекты, такие как интерактивная веб-форма, могут быть встроены в отображаемую страницу веб-сайта. HTML предоставляет средства для создания заголовков, абзацев, списков, ссылок, цитат и других элементов. Элементы HTML выделяются тегами, записанными с использованием угловых скобок. Такие теги, как <img /> и <input />, напрямую вводят контент на страницу. Другие теги, такие как <p>, окружают и оформляют текст внутри себя и могут включать другие теги в качестве подэлементов. Браузеры не отображают HTML-теги, но используют их для интерпретации содержимого страницы.

Язык XHTML является более строгим вариантом HTML, он следует синтаксису XML и является приложением языка XML в области разметки гипертекста.

В HTML можно встроить программный код на языке программирования JavaScript, для управления поведением и содержанием веб-страниц. Также включение CSS в HTML описывает внешний вид и макет страницы.

Текстовые документы, содержащие разметку на языке HTML (такие документы традиционно имеют расширение .html или .htm), обрабатываются специальными приложениями, которые отображают документ в его форматированном виде. Такие приложения, называемые «браузерами» или «интернет-обозревателями», обычно предоставляют пользователю удобный интерфейс для запроса веб-страниц, их просмотра (и вывода на иные внешние устройства) и, при необходимости, отправки введённых пользователем данных на сервер. Наиболее популярными на сегодняшний день браузерами являются Google Chrome, Mozilla Firefox, Opera, Internet Explorer и Safari.

* 1. CSS

Для добавления стилей и изменения внешнего вида элементов на веб-сайте используется CSS. CSS – нормальный язык описания внешнего вида документа (веб-страницы), написанного с использованием языка разметки (чаще всего HTML или XHTML). Также может применяться к любым XML-документам, например, к SVG или XUL.

CSS используется создателями веб-страниц для задания цветов, шрифтов, стилей, расположения отдельных блоков и других аспектов представления внешнего вида этих веб-страниц. Основной целью разработки CSS является ограждение и отделение описания логической структуры веб-страницы (которое производится с помощью HTML или других языков разметки) от описания внешнего вида этой веб-страницы (которое теперь производится с помощью формального языка CSS). Такое разделение может увеличить доступность документа, предоставить большую гибкость и возможность управления его представлением, а также уменьшить сложность и повторяемость в структурном содержимом.

Кроме того, CSS позволяет представлять один и тот же документ в различных стилях или методах вывода, таких как экранное представление, печатное представление, чтение голосом (специальным голосовым браузером или программой чтения с экрана) или при выводе устройствами, использующими шрифт Брайля.

* 1. Neo4j

Neo4j – графовая система управления базами данных с открытым исходным кодом, реализованная на Java. По состоянию на 2015 год считается самой распространённой графовой СУБД.

Данные хранит в собственном формате, специализированно приспособленном для представления графовой информации, такой подход в сравнении с моделированием графовой базы данных средствами реляционной СУБД позволяет применять дополнительную оптимизацию в случае данных с более сложной структурой. Также утверждается о наличии специальных оптимизаций для SSD-накопителей, при этом для обработки графа не требуется его помещение целиком в оперативную память вычислительного узла, таким образом, возможна обработка достаточно больших графов.

Основные транзакционные возможности – поддержка ACID и соответствие спецификациям JTA, JTS и XA. Интерфейс программирования приложений для СУБД реализован для многих языков программирования, включая Java, Python, Clojure, Ruby, PHP, также реализовано API в стиле REST. Расширить программный интерфейс можно как с помощью серверных плагинов, так и с помощью неуправляемых расширений (unmanaged extensions); плагины могут добавлять новые ресурсы к REST-интерфейсу для конечных пользователей, а расширения позволяют получить полный контроль над программным интерфейсом, и могут содержать произвольный код.

В СУБД используется собственный язык запросов – Cypher. Cypher является не только языком запросов, но и языком манипулирования данными, так как предоставляет функции CRUD для графового хранилища. Запросы можно делать и другими способами, например, напрямую через Neomodel Query API. Каждая модель данных WEB-сайта (унаследованные от StructureNode или DjangoNode) имеет некоторые свойства и методы, которые выражают запросы к базе данных Neo4j.

Для построения графовой БД Web-сайта используются специальные плагины Neosemantics (n10s) и APOC, которые нужно заранее установить.

* + 1. Плагин Neosemantics (n10s)

Neosemantics (n10s) – это плагин, который позволяет использовать RDF и связанные с ним словари, такие как OWL, RDFS, SKOS и другие в Neo4j. Данный плагин используется для интеграции с компонентами, генерирующими / потребляющими RDF. Neosemantics работает как расширение базы данных Neo4j. Основными функциями данного плагина являются:

* импорт и экспорт RDF в нескольких форматах (Turtle, N-Triples, JSON-LD, RDF / XML, TriG и N-Quads, TriG \*);
* отображение модели при импорте и экспорте;
* импорт и экспорт онтологий и таксономий в различных словарях (OWL, SKOS, RDFS);
* проверка графа на основе ограничений SHACL;
* базовый вывод.
  + 1. Плагин APOC

APOC (Awesome Procedures on Cypher) – вспомогательный плагин для базы данных. Библиотека APOC считается самой большой и наиболее широко используемой библиотекой расширений для Neo4j. Она включает более 450 стандартных процедур, обеспечивающих функциональные возможности для утилит, преобразований, обновлений графов и многого другого. Они хорошо поддерживаются, и их очень легко запускать как отдельные функции или включать в запросы языка Cypher.

* 1. Figma

Figma - это веб-приложение, которое используется для создания макетов, прототипов и интерфейсов пользовательских приложений. Она позволяет дизайнерам и разработчикам работать в одном пространстве, делая процесс проектирования более эффективным и коллаборативным.

Одним из основных преимуществ Figma является возможность работы в режиме реального времени. Это означает, что несколько пользователей могут работать над проектом одновременно, делая коммуникацию и совместное редактирование более простым и быстрым.

Figma также предлагает множество функций для ускорения процесса дизайна. Эти функции включают в себя возможность создания макетов и элементов интерфейса с помощью векторных форм, использование библиотек и шаблонов для ускорения работы над проектами, а также интеграцию с другими инструментами и сервисами, такими как Trello, Slack и GitHub.

Одно из главных преимуществ Figma заключается в его доступности. Он работает на любой операционной системе и доступен через веб-браузер, что позволяет пользователям работать над проектом в любом месте и на любом устройстве. Кроме того, Figma предлагает бесплатный план, который позволяет пользователям создавать и редактировать неограниченное количество проектов, делая его доступным для малых команд и начинающих дизайнеров.

В целом, использование Figma имеет множество преимуществ для дизайнеров и команд разработки. Он позволяет ускорить процесс проектирования и сделать его более эффективным и коллаборативным, что может привести к более высокому качеству конечного продукта.

1. ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ
2. 1. Методология применения построения прототипа компонента СКА ИИ создание графовой БД

В данном разделе приведены методы организации работы компонента получения данных с интернет источников и быстрого построения графовой БД Web-сайта и анализа его свойств.

* + 1. Построение графа свойств в Neo4j на основе данных RDF

На сайте Open Food Facts имеется общедоступный набор данных с 6,2 миллионами троек по пищевым продуктам, их ингредиентам, аллергенам, фактам питания и многому другому. Импортируя тройки RDF с помощью запроса, представленного ниже, получается тематическая графовая БД (рисунок 3.1).

CALL n10s.rdf.import.fetch(”http://fr.openfoodfacts.org/data/ fr.openfoodfacts.org.products.rdf” , ”RDF/XML”)

A screenshot of a computer

Description automatically generated with medium confidence

Рисунок 3.1 – Импорт данных RDF с сайта Open Food Facts

После импорта данных RDF общее представление полученной БД представлено на рисунке 3.2.

CALL db.schema.visualization()

A screenshot of a computer

Description automatically generated with medium confidence

Рисунок 3.2 – Общее представление схемы графовой базы данных

В данной БД основными типами узлов являются:

* *FoodProduct* – продукты;
* *Food* – ингредиенты, которые могут содержаться в продуктах;
* *Ingredient* – промежуточные узлы, соединяющие узлы *FoodProduct* и *Food*, в которых указывается, сколько ингредиента содержится в продукте.

Основными типами связей являются:

* *containsIngredient* – соединяет продукт *FoodProduct* и промежуточный узел *Ingredient*;
* *food* – соединяет промежуточный узел *Ingredient* и ингредиент *Food*.

В полученной базе данных можно искать различную информацию о продуктах и составе их ингредиент. К примеру, с помощью запроса, представленного ниже, можно узнать, какой набор общих ингредиентов у двух продуктов. Результат запроса показан на рисунке 3.3.

MATCH (prod1:Resource { uri: 'http://world-fr.openfoodfacts.org/produit/5053827196642/coco-pops-kellogg-s'})

MATCH (prod2:Resource { uri: 'http://world-fr.openfoodfacts.org/produit/5010358227290/2-snack-pork-pies-spar'})

MATCH (prod1)-[:ns1\_\_containsIngredient]->(x1)-[:ns1\_\_food]->(sharedIngredient)<-[:ns1\_\_food]-(x2)<-[:ns1\_\_containsIngredient]-(prod2)

RETURN prod1, prod2, x1, x2, sharedIngredient

A picture containing screenshot, diagram, text, circle

Description automatically generated

Рисунок 3.3 – Вывод общих ингредиентов у двух продуктов

В данных RDF хранятся все данные и характеристики продуктов и ингредиентов. На рисунке 3.4 видно, что каждый узел продукта FoodProduct имеет набор свойств, таких как идентификационный номер узла, название продукта, содержание различных микроэлементов в продукте и т.д., каждый узел ингредиента имеет идентификационный номер, название ингредиента и uri, а промежуточный узел между продуктом и ингредиентом Ingredient имеет идентификационный номер, свойство, указывающее на то, сколько ингредиента содержится в продукте и uri.

A screenshot of a computer

Description automatically generated with medium confidence

Рисунок 3.4 – Свойства узлов графовой БД с продуктами питания

Таким образом, мы получаем универсальный инструмент для быстрого поиска информации о нужном нам продукте, сортировке и выборке продуктов по определённому свойству и др.

К примеру, вы можем найти все продукты, в которых содержится пастеризованное молоко и в которых на 100 г продукта содержится меньше 100 ккал. Для этого требуется выполнить следующий запрос. Результат запроса представлен на рисунке 3.5.

MATCH (f:ns1\_\_FoodProduct)

WHERE ((any(ingridient in split(f.ns1\_\_IngredientListAsText,',') WHERE ingridient = 'Pasteurized milk')) and (toFloat(f.ns1\_\_energyKcalPer100g) < 100))

RETURN f LIMIT 100

A picture containing text, screenshot

Description automatically generated

Рисунок 3.5 – Продукты, содержащие пастеризованное молоко и содержащие меньше 100 ккал на 100 г

Таким образом, в нужную выборку продуктов попало 29 продуктов питания.

Норма соли для человека в день не более 5 грамм. А пальмовое масло для человека не очень полезное. С помощью следующего не простого запроса можем найти все продукты, которые не содержат пальмовое масло и содержания соли которых на 100 г меньше 0,5 г. Результат запроса представлен на рисунке 3.6.

MATCH (f:ns1\_\_FoodProduct)

WHERE ((any(ingridient in split(f.ns1\_\_IngredientListAsText,',') WHERE ingridient <> 'palm oil')) and (toFloat(f.ns1\_\_saltPer100g) < 0.5))

RETURN f LIMIT 100

A picture containing text, screenshot

Description automatically generated

Рисунок 3.6 – Продукты, не содержащие пальмового масла и содержащие меньше 0,5 г соли на 100 г

* + 1. Построение тематической графовой БД на основе данных Turtle

В качестве примера технологии построения графовой БД будут использоваться данные на общедоступном популярном сайте Wikipedia. Эти данные хранятся в формате **Turtle** и могут быть сериализованы в графовую базу данных с помощью комбинирования запросов SPARQL и возможностей Neo4j (информация о данных хранится на сайте Wikidata).

В зависимости от конфигурации базы данных, плагин Neosemantics будет обрабатывать многозначные свойства по-разному (многозначные свойства распространены в многоязычных наборах данных, таких как Wikidata, где находятся несколько значений для имен, по одному для каждого доступного языка: Соединенное Королевство, United Kingdom, Regno Unito, Royaume Uni и так далее). По умолчанию Neosemantics предполагает, что свойства имеют одно значение, и каждое новое значение того же свойства на том же узле перезаписывает предыдущее значение, но в некоторых случаях надо сохранить все значения. Для этого надо поменять конфигурацию базы данных с помощью следующего запроса на рисунке 3.7.

CALL n10s.graphconfig.init({

  handleVocabUris: 'MAP',

  handleMultival: 'ARRAY',

  keepLangTag: true,

  keepCustomDataTypes: true,

  applyNeo4jNaming: true

})

A screenshot of a computer program

Description automatically generated with low confidence

Рисунок 3.7 – Изменение конфигурации базы данных

Данный запрос состоит из следующих строк:

1 – установка в параметре handleVocabUris значения MAP указывает Neosemantics применять сопоставления к элементам схемы по мере их добавления в граф с помощью процедур n10s.nsprefixes.add и n10s.mapping.add.

2 – данный параметр гарантирует, что несколько значений будут храниться в Neo4j в виде массива.

3 – сохранение тега языка будет означать, что каждое переведенное свойство будет иметь суффикс, например: United Kingdom@en или Regno Unito@it.

4 – данный параметр гарантирует, что любые пользовательские (определяемые пользователем, отличные от XML-схемы) типы данных также сохраняются в Neo4j в виде строки, за которой следует их тип данных URI.

5 – означает, что к элементам графа будут применятся рекомендуемые Neo4j имена – все заглавные буквы для типов отношений, Upper Camel Case для меток и т. д.

С помощью следующего запроса, представленного на рисунке 3.8, сериализуем данные об архитектурных строениях в Испании и стилях, к которым они относятся.

A screenshot of a computer code

Description automatically generated with low confidence

Рисунок 3.8 – Запрос для сериализации данных об архитектурных строениях и стилях

Перевод данных в RDF осуществляется запросом SPARQL CONSTRUCT, в результате выполнения которого поток троек субъекта, предиката и объекта, которые вместе представляют граф RDF.

SPARQL CONSTRUCT состоит из следующий строк:

1 – данный оператор определяет пространство имен neo4j://. Для того, чтобы определить новые термины (Страна, имя континента и т. д.), необходимо дать им полное имя в RDF.

2 – раздел CONSTRUCT запроса определяет триплеты, которые в итоге будут получены.

3 – a sch:Monument заменяется на wdt:P31 (instance of, т.е. «является») wd:Q4989906 (monument – архитектурное строение). Neosemantics переведет это утверждение в метку на узлах архитектурного строения.

4 – узел архитектурного строения будет иметь свойство sch\_\_name (название строения), взятое из monumentName.

5 – узел архитектурного строения будет иметь свойство sch\_\_location (название строения), взятое из location. Заменяется на wdt:P17 (страна) wd:Q29 (Испания).

6 – Создание связи sch\_ARCHITECTURE между архитектурным строением и архитектурным стилем.

7 – a sch:Architecture заменяется на wdt:P149 (architectural style – архитектурный стиль). Neosemantics переведет это утверждение в метку на узлах архитектурного стиля.

8 – узел архитектурного стиля будет иметь свойство sch\_\_name (название стиля), взятое из architectureName.

Результат выполнения запроса представлен на рисунке 3.9.

A picture containing circle

Description automatically generated

Рисунок 3.9 – Результат выполнения запроса

В результаты жёлтые узлы – это архитектурные стили, а розовые узлы – это архитектурные строения.

Таким образом, с помощью плагина Neosemantics можно сериализовать данные в формате «Turtle».

* + 1. Построение графа свойств на основе данных, встроенных в веб-страницы (JSON)

При сериализации данных, наиболее распространённым типом данных является JSON. Он часто используется на сайтах со статьями и публикациями. Рассмотрим сайт со статьями на научные темы Semantic Scholar. Как правило, все элементы, нужные для сериализации и построения графовой БД находятся в коде страницы сайта в script элементе с типом *application/ld+json*.

Здесь в специальном формате представлены необходимые данные об узлах, их свойствах и связях между ними будущей графовой базы данных. Для извлечения информации с веб-страницы можно использовать процедуру APOC **apoc.load.html**, представленную ниже. Для этого требуется URL-адрес страницы и CSS-подобный селектор, чтобы выбрать конкретный элемент, который вам нужен. Результат запроса представлен на рисунке 3.10.

CALL apoc.load.html("https://www.semanticscholar.org/paper/Big-data%3A-The-next-frontier-for-innovation%2C-and-Manyika/91b63db746becca15090963a8990dfe2b5103799" , { jsonld: 'head script[type="application/ld+json"]'}) YIELD value

A close-up of a document

Description automatically generated with low confidence

Рисунок 3.10 – Извлечение информации с веб-страницы с помощью APOC

С помощью запроса выше получили скрипт со страницы с типом *application/ld+json*.

Для визуализации и анализа RDF используется плагин Neosemantics. С помощью комбинированного запроса, представленного ниже, можно сериализовать данные четырёх статей сайта по тематике Big Data в RDF. Результат сериализации представлен на рисунке 3.11.

UNWIND ["https://www.semanticscholar.org/paper/Big-data%3A-The-next-frontier-for-innovation%2C-and-Manyika/91b63db746becca15090963a8990dfe2b5103799", "https://www.semanticscholar.org/paper/Business-Intelligence-and-Analytics%3A-From-Big-Data-Chen-Chiang/f117c6f12d067bd66dad40996b3931c069daa2da", " https://www.semanticscholar.org/paper/Digital-twin-driven-product-design%2C-manufacturing-Tao-Cheng/1597449a7f64b6bd24639b4deab96c8a8c184177 ", " https://www.semanticscholar.org/paper/Big-Data-Storage-Strohbach-Daubert/3c30882e0ecb769dc3bbe170d0f662ea521452dd "] as page

CALL apoc.load.html(page, { jsonld: 'head script[type="application/ld+json"]'}) YIELD value

CALL n10s.rdf.import.inline(value.jsonld[0].data, "JSON-LD")  yield terminationStatus, triplesLoaded, triplesParsed, extraInfo

RETURN page, terminationStatus, triplesLoaded, triplesParsed, extraInfo

A picture containing text, line, receipt, screenshot

Description automatically generated

Рисунок 3.11 – Сериализация четырёх статей сайта Semantic Scholar

На рисунке 3.12, приведено общее представление графовой базы данных, полученной из сайта с помощью технологии быстрого построения тематической графовой БД.

CALL db.schema.visualization()

A picture containing circle, child art, drawing, art

Description automatically generated

Рисунок 3.12 – Общее представление графовой БД сайта Semantic Scholar

На рисунке 3.13 представлены все полученные типы узлов и связей между ними.

A picture containing text, screenshot, font

Description automatically generated

Рисунок 3.13 – Типы узлов и связей графовой БД сайта Semantic Scholar

После сериализации образовались узлы 9 видов и связи 9 видов. Фрагмент JSON-LD использует Schema.org словарь, откуда берутся все элементы в схеме графа. Каждый элемент имеет определённый тип и свойства, характерные только ему.

В случае получившегося запроса имеются следующие виды узлов:

* *Organization* – организации, на которые могут ссылаться статьи, (к примеру, школа, неправительственная организация, корпорация, университет)
* *Article –* тип узлов, обозначающий статью или публикацию любого характера (к примеру, научного, публицистического, обзорного, аналитического и т.д.)
* *ScholarlyArticle* – научная статья или публикация;
* *Person –* человек, в случае со статьями, автор или издатель статьи;
* *ListItem* – типа узла, указывающий на то, что узел является элементом какого-то списка;
* *BreadcrumbList* – это ItemList, состоящий из цепочки связанных веб-страниц, обычно описываемых с использованием их URL-адреса и имени и обычно заканчивающихся текущей страницей.;
* *ImageObject* – изображение чего-то (графический файл);
* *\_GraphConfig* – узел, содержащий в себе все настройки и конфигурацию плагина Neosemantics;
* *Resource –* все остальные узлы, которые не попали под какую-либо из категорий.

Имеются следующие виды связей:

* *sameAs –* связь, которая указывает на идентичность элементов;
* *image –* изображение предмета. Это может быть URL-адрес или полностью описанный ImageObject;
* *author* – автор какой-либо публикации или статьи;
* *publisher –* издатель творческой работы;
* *mainEntityOfPage* –указывает на страницу (или другую творческую работу), для которой эта вещь является основным описываемым объектом.
* *item* –какая-то сущность, данные
* *itemListElement* –какая-то сущность, являющаяся ListItem
* *url –* URL-адрес элемента;
* *contentUrl* *–* URL-адрес на медиа объект (фото или видео)

В полученной БД можно искать информацию с помощью запросов. К примеру, если известно название статьи и нужно узнать, кто написал данную статью, то это можно сделать с помощью следующего запроса. Результат выполнения запроса представлен на рисунке 3.14.

MATCH (ar:sch\_\_Article)-[:sch\_\_author]->(ath:sch\_\_Person)

WHERE ar.sch\_\_name = 'Big Data Storage'

RETURN ar, ath

A screenshot of a diagram

Description automatically generated with low confidence

Рисунок 3.14 – Поиск авторов по названию статьи

Статьи имеют авторское право и могут принадлежать какой-то организации. С помощью следующего запроса можно узнать, какой организации принадлежат статьи. Результат запроса представлен на рисунке 3.15.

MATCH (ar:sch\_\_Article)-[:sch\_\_publisher]->(org:sch\_\_Organization)

RETURN ar, org

A screenshot of a diagram

Description automatically generated with low confidence

Рисунок 3.15 – Статьи и организации, которым они принадлежат

Созданная ИТ среда Neo4j Desktop позволяет анализировать какие веб-страницы и элементы связаны друг с другом и какой связью, а также объединять несколько похожих графовых баз данных (к примеру, различные сайты со статьями) в одну графовую базу данных. При этом повторяющиеся узлы, которые будут присутствовать в нескольких баз данных, при объединении так же будут объединяться.

Сериализуем данные с ещё одного сайта с журналами и статьями SpringerOpen. Для этого выполним запрос, указанный ниже. На рисунке 3.16 результат запроса.

UNWIND [

"https://journalofbigdata.springeropen.com/articles/10.1186/s40537-022-00664-6 ", " https://journalofbigdata.springeropen.com/articles/10.1186/s40537-022-00654-8", " https://journalofbigdata.springeropen.com/articles/10.1186/s40537-022-00652-w", " https://journalofbigdata.springeropen.com/articles/10.1186/s40537-022-00650-y "] as page

CALL apoc.load.html(page, { jsonld: 'head script[type="application/ld+json"]'}) YIELD value

CALL n10s.rdf.import.inline(value.jsonld[0].data, "JSON-LD")  yield terminationStatus,

triplesLoaded, triplesParsed, extraInfo

RETURN page, terminationStatus, triplesLoaded, triplesParsed, extraInfo

A picture containing text, receipt, line, number

Description automatically generated

Рисунок 3.16 – Сериализация четырёх статей по теме Big Data сайта SpringerOpen

На рисунке 3.17, приведено общее представление графовой базы данных, полученной из сайта SpringerOpen.

CALL db.schema.visualization()

A picture containing circle, drawing, child art, sketch

Description automatically generated

Рисункок 3.17 – Общее представление графовой БД сайта SpringerOpen

На рисунке 3.18 представлены все полученные типы узлов и связей между ними.

A picture containing text, screenshot, font

Description automatically generated

Рисунок 3.18 – Типы узлов и связей графовой БД сайта Semantic Scholar

В полученной БД также можно искать информацию с помощью запросов. К примеру, с помощью следующего запроса можно узнать, какие авторы написали статьи, которые мы сериализовали. Результат выполнения запроса представлен на рисунке 3.19.

MATCH (ar:sch\_\_ScholarlyArticle)-[:sch\_\_author]->(ath:sch\_\_Person)

RETURN ar, ath

A screenshot of a computer

Description automatically generated with low confidence

Рисунок 3.19 – Статьи и их авторы

Тип узла sch\_\_Periodical указывает на то, что статьи, которые связаны с данным узлом, выпущены последовательно в каком-то журнале, т.е. являются статьями по одной теме. И с помощью следующего запроса можно определить, связаны ли статьи или нет. Результат запроса показан на рисунке 3.20.

MATCH (ar:sch\_\_ScholarlyArticle)-[:sch\_\_isPartOf]->(per:sch\_\_Periodical)

RETURN ar, per

A screenshot of a computer

Description automatically generated with low confidence

Рисунок 3.20 – Связанные статьи

Из полученного результата видно, что статьи являются продолжением или дополнением друг друга в журнале «Journal of Big Data».

С помощью таких запросов можно сериализовать данные с разных в одну графовую БД. На рисунке 3.21 представлено общее представление графовой БД сайтов Semantic Scholar и SpringerOpen. На данном представлении видно, что узлы (Person, ImageObject, ScholarlyArticle, Organization, NsPreDef, Resource, GraphConfig), которые были и в первой, и во второй графовой базах данных, объединились, как и повторяющиеся типы связей (sameAs, image, url, publisher, author).

A picture containing drawing, circle, child art, sketch

Description automatically generated

Рисунок 3.21 – Общее представление графовой БД сайтов Semantic Scholar и SpringerOpen

На рисунке 3.22 представлены все полученные типы узлов и связей между ними.

A picture containing text, screenshot, font, design

Description automatically generated

Рисунок 3.22 – Типы узлов и связей графовой БД сайтов Semantic Scholar и SpringerOpen

Таким образом можно объединять разные базы данных в одну тематическую базу данных. В данном случае объдинены 4 статьи по тематике Big Data с сайта Semantic Scholar и 4 сайта по тематике Big Data с сайта SpringerOpen.

С помощью следующего запроса можно вывести все статьи и их авторов. Результат запроса представлен на рисунке 3.23.

MATCH (a:sch\_\_ScholarlyArticle)-[f:sch\_\_author]->(ath:sch\_\_Person)

RETURN a, ath

A picture containing circle, design, illustration

Description automatically generated

Рисунок 3.23 – Все статьи и их авторы

Таким образом, с помощью запросов к графовой БД, можно находить различную информацию об статьях и авторов.

* 1. Функциональная модель программного средства

Функциональная модель программного средства представлена в виде диаграммы вариантов использования и информационной модели предметной области. Варианты использования отражают функциональность системы в ответ на внешние воздействия с точки зрения получения значимого результата для пользователей.

* + 1. Варианты использования программного средства

Исходя из выдвинутых требований, проектируемое программное средство предполагает поддержку системы ролей, т.е. разделение пользователей на различные роли, а следовательно, и выдача различных прав нескольким категориям пользователей: гостю, зарегистрированному пользователю, администратору.

Возможности обычного пользователя и обслуживающего систему персонала представлены на рисунке 3.24 в виде диаграммы вариантов использования.

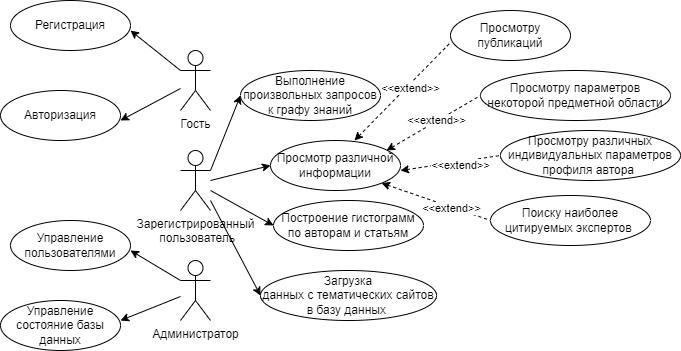


Рисунок 3.24 – Диаграмма вариантов использования

Рассмотрим подробно представленные на рисунке прецеденты. Регистрация и авторизация – функции, которые доступны для роли «Гость» (пользователь, не зарегистрированный в системе). После регистрации пользователь получает роль «Зарегистрированный пользователь» и имеет доступ к:

* Просмотру публикаций различных предметных областей.
* Поиску наиболее цитируемых экспертов предметной области.
* Просмотру параметров некоторой предметной области.
* Просмотру различных индивидуальных параметров профиля автора публикации. Пользователь может видеть имя, количество публикаци и сами публикации автора.
* Выполнение произвольных запросов к графу знаний. Пользователь может задать конкретные условия и свойства, по которым будут искаться авторы и статьи.
* Построение гистограмм по авторам и статьям. Гистограммы по статьям строятся с помощью алгоритма PageRank, а по авторам в зависимости от того, сколько статей написал автор. При этом пользователь сам выбирает, какая гистограмма ему нужна и какое количество статей или авторов должно в неё входить.
* Загрузка данных с тематических сайтов в базу данных разными способами. Пользователь может загрузить дополнительные статьи в базу данных, вставя ссылку на статью в специальное поле или загрузя txt файл, в котором находятся ссылки на статьи.

Кроме того, в системе есть еще один вид пользователей – администраторы. Они не имеют доступа к личным аккаунтам пользователей, однако обладают возможностью наделять пользователей различными правами. Помимо этого, администраторы обладают функциями управления системой:

* Управление пользователями. Администратор может добавлять, удалять пользователей.
* Управление состояние базы данных. Администратор может менять струтуру БД, содержимое БД и делать замену одной БД на сайте на другую.
  + 1. Разработка инфологической модели базы данных

Исходя из необходимости использования в проектируемом приложении базы данных, разработаем ее инфологическую модель. Для создания данной модели возьмем за основу предметную область проекта. Предметная область разрабатываемого программного средства включает в себя следующие сущности и их атрибуты:

* User (sch\_\_Users):
  + name – имя пользователя:
  + hashed password – захешированный пароль;
  + email – почта пользователя;
  + created date – дата регистрации пользователя;
  + role – роль пользователя;
* Article (sch\_\_Article):
  + title – название статьи;
  + year – год выпуска статьи;
  + venue – награды, которые статья получила;
  + n\_citation – цитирование статьи;
  + abstract – краткое описание статьи;
  + theme – тематика статьи;
  + references – ссылки, на которые статья указывает;
  + pagerank – рейтинг статьи по pagerank;
* Author (sch\_\_Person):
  + name – имя и фамилия автора;
  + articles – статьи, которые принадлежат автору;
  + popularity – популярность автора по количеству статей;
  + pagerank – рейтинг автора по pagerank ;

Схема базы данных представлена на рисунке 3.25:

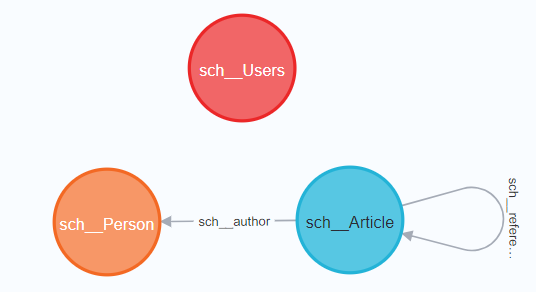


Рисунок 3.25 – Схема базы даных

Таким образом, в графовой БД есть три типа узлов: пользователи, статьи и авторы. Начальная БД имеет следующие свойства:

* sch\_\_author – свойство, связывающее статью и автора, который написал её
* sch\_\_references – свойства, связывающее первую и вторую статью, на которую ссылается первая статья
  1. Прототип пользовательского интерфейса

Для демонстрации работы компонента СКА ИИ был разработан сайт. В данном разделе рассматривается прототип пользовательского интерфейса, созданный в графическом онлайн-редакторе Figma, который в дальнейшем был использован при написании фронтенда сайта.

При входе на сайт пользователь с ролью гость попадает на стартовую страницу, которая содержит надпись BSUIR Science Work, и в хедере страницы находятся кнопки перехода на страницу регистрации и авторизации (рисунок 3.25):



Рисунок 3.25 – Стартовая страница пользователя с ролью «гость»

При нажатии на кнопку SIGN UP пользователь попадает на страницу регистрации (рисунок 3.26). Для регистрации пользователь должен в нужные поля ввести имя пользователя, почту, пароль, в поле Confirm password ввести ещё раз пароль и нажать на кнопку ниже SIGN UP.

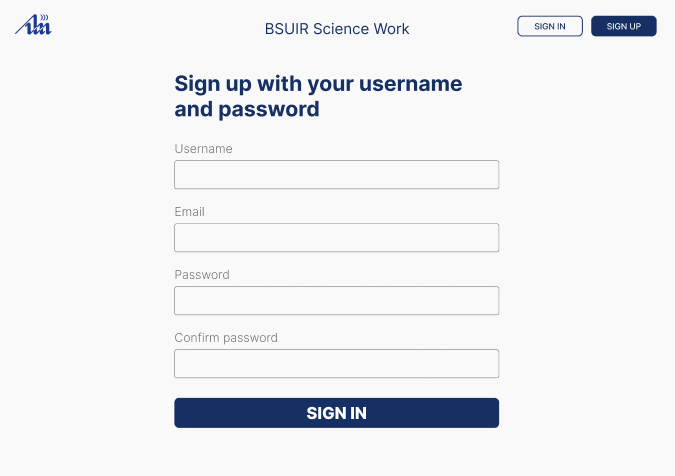


Рисунок 3.26 – Страница регистрации

Если пользователь ранее был зарегистрирован на сайте, он может нажать на кнопку SIGN IN и попасть на страницу авторизации (рисунок 3.27). Для хода в система пользователю надо ввести имя и пароль, указанные при регистрации, и нажать на кнопку ниже SIGN IN.

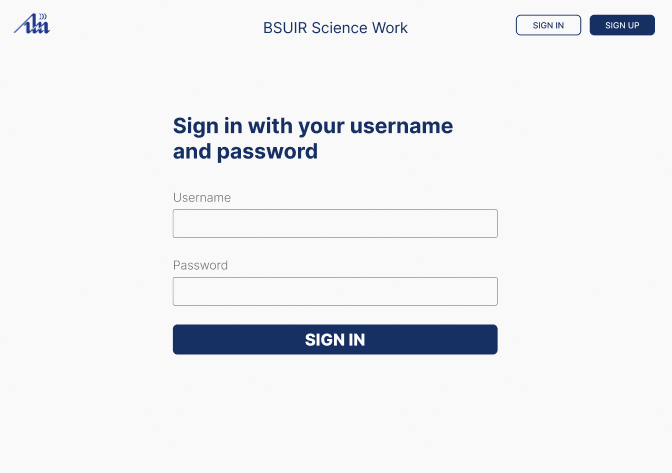


Рисунок 3.27 – Страница авторизации

После регистрации/авторизации вид главной страницы пользователя выглядит так, как показано на рисунке 3.28. В хедере страницы появились кнопки DATA, STATISTICS и DATA MANIPULATION.

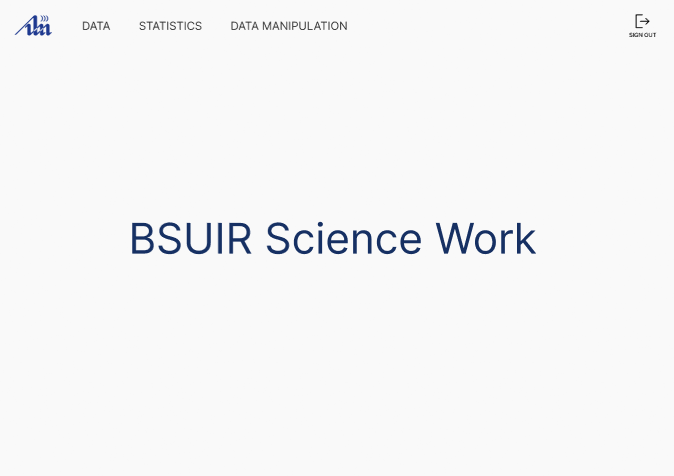


Рисунок 3.28 – Главная страница пользователя после регистрации/авторизации

При нажатии кнопки DATA пользователь переходит на страницу со всеми статьями и авторами, которые находятся в графовой базе данных (рисунок 3.29). При этом изначально на странице не отображаются ни статьи, ни авторы.

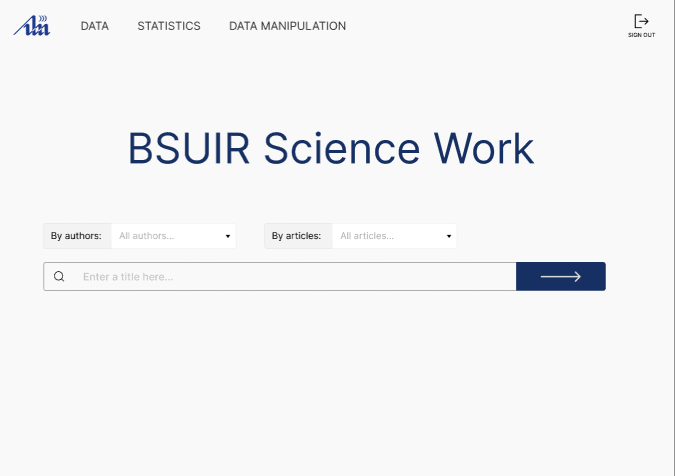


Рисунок 3.29 – Страница со статьями и авторами

На странице присутствует фильтрация по авторам и статьям, а также пользователь может искать статью по её названию, введи нужный текст в поле search. При нажатии на значок лупы появляется таблица, в которой отображены данные по статьям (рисунок 3.30).

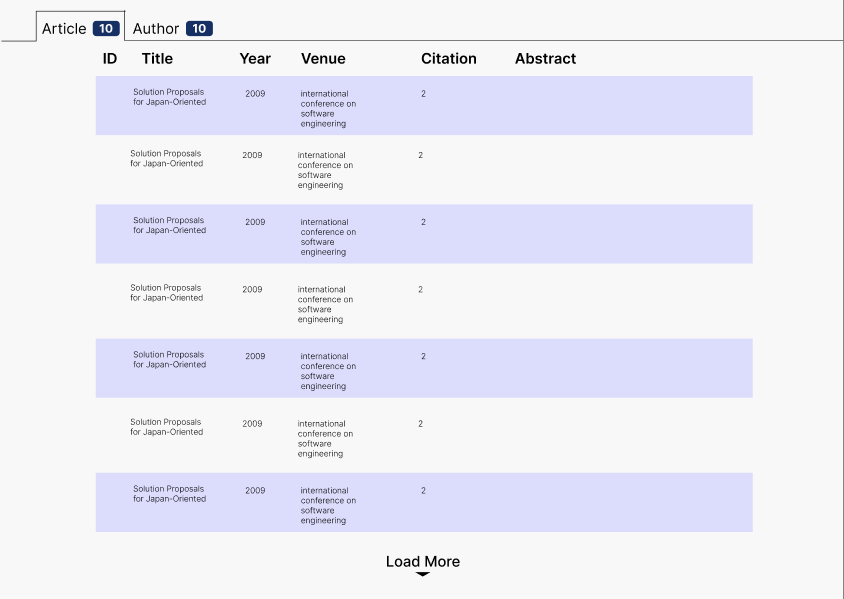


Рисунок 3.30 – Вывод фильтрованных данных по статьям

В таблице показана такая информция по статьям, как заголовок статьи, год издания, награды, если статья их получала, цитировании и краткое описании статьи. Изначально показывается 10 статей. Если пользователь хочет увидеть ещё статьи, он должен нажать на кнопку Load More под таблицей. При нажатии на какую-либо статью, в сплывающем окне выводится информация об ней (рисунок 3.31).

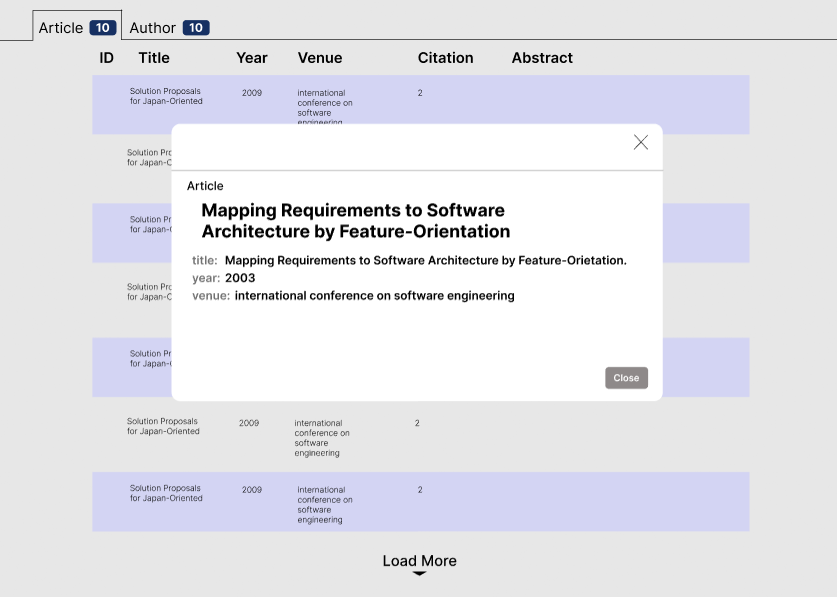


Рисунок 3.31 – Вывод инфорации о выбранной статье

Чтобы убрать всплывающее окно с данными о статье, пользователь может нажать на кнопку Close или в любое место затемнённой области вокруг окна. Для того, чтобы просмотреть авторов, пользователь должен нажать на вкладку Author (рисунок 3.32). Изначально показывается 10 авторов. Если пользователь хочет увидеть ещё авторов, он должен нажать на кнопку Load More под таблицей.



Рисунок 3.32 – Вывод фильтрованных данных по авторам

При нажатии на какого-либо автора, в сплывающем окне выводится информация о нём (рисунок 3.33).

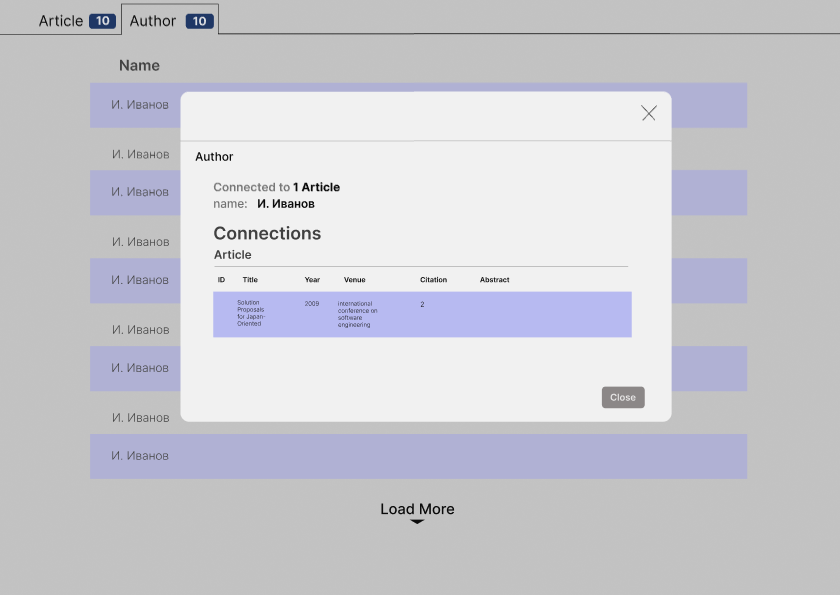


Рисунок 3.33 – Вывод фильтрованных данных по авторам

Здесь указано, сколько статей из тех, которые имеются на сайте, написал именно этот автор и ниже выведены эти статьи. Чтобы убрать всплывающее окно с данными о статье, пользователь может нажать на кнопку Close или в любое место затемнённой области вокруг окна.

Нажав на кнопку STATISTICS пользователь попадает на страницу для сбора статистики по имеющимся данным на сайте (рисунок 3.34). На данной странице пользователь может строить различные гистограммы. Для этого во всплывающем списке Node type он должен выбрать, по каким данным будет строиться гистограмма (по статьям или по авторам), и ниже в поле Number of columns ввести количество статей или авторов, которые должны быть в гистограмме. После выбора и ввода данных и нажатия на кнопку SUBMIT ниже выводиться гистограмма. Статьи в ней отсортированы по алгоритму PageRank, а авторы по количеству написанных статей.

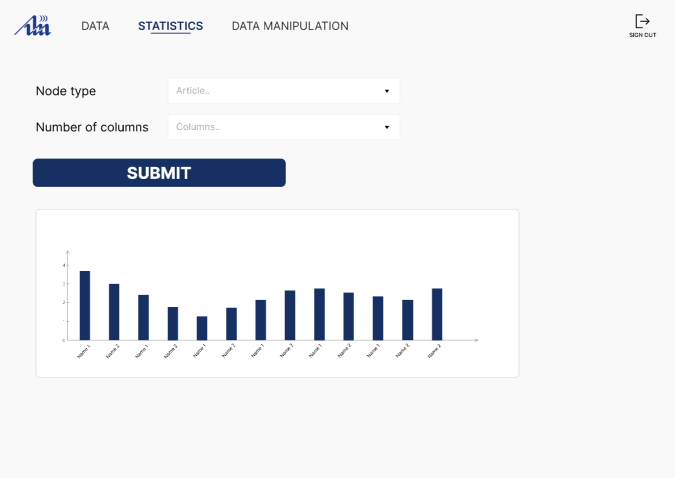


Рисунок 3.34 – Страница построения гистограмм по имеющимся данным

В хедере нажав на кнопку DATA MANIPULATION пользователь попадает на страницу для загрузки данных с тематических сайтов в графовую базу данных проекта (рисунок 3.35).

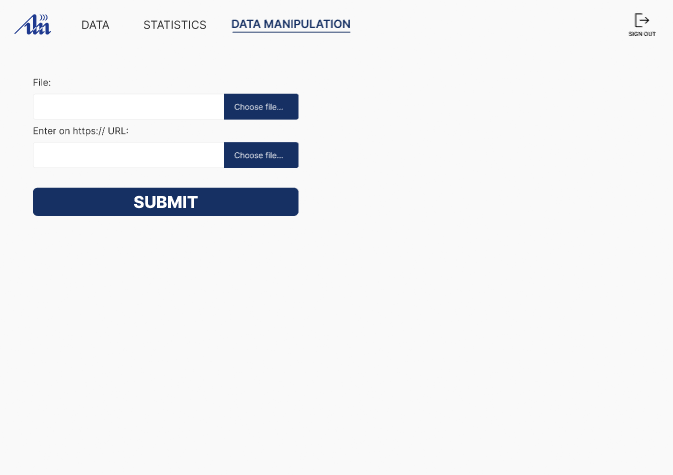


Рисунок 3.35 – Страница загрузки данных с тематических сайтов в

базу данных Neo4j

Загружать данные пользователь может двумя способами. Первый – нажать на кнопку Choose file…, выбрать нужный файл с расширением txt, в котором находятся ссылки на статьи, которые пользователь хочет добавить в графовую БД проекта и нажать на кнопку SUBMIT. Второй способ – во второе поле вставить ссылку на статью, котор которые пользователь хочет добавить в графовую БД проекта и нажать на кнопку SUBMIT.

1. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ
2. 1. Создание и конфигурация базы данных

Разработчиком Neo4j предоставляется удобное десктоп приложение Neo4j Desktop, которое предоставляет удобный интерфейс для управления базами данных. Создадим базу данных приложения с помощью Neo4j Desktop (рисунок 4.1):

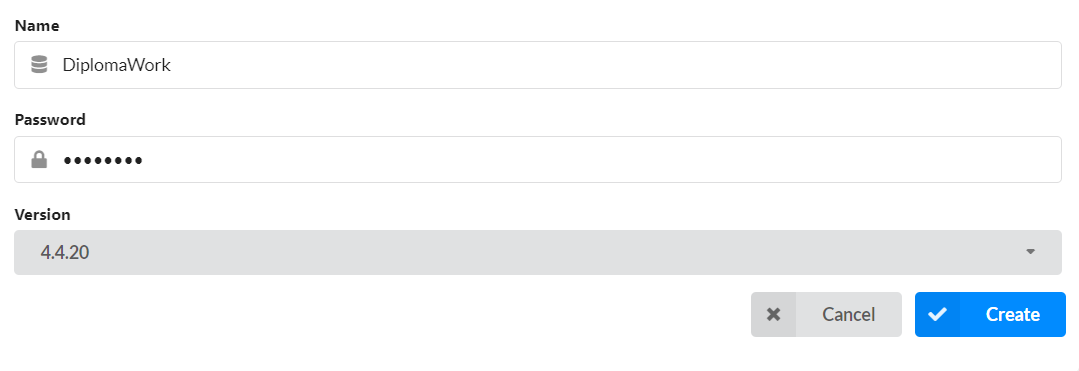


Рисунок 4.1 – Создание новой базы данных через Neo4j Desktop

Перед запуском необходимо сконфигурировать виртуальную машину Java (JVM), обеспечивающую работу СУБД. В конфигурационном файле содержится множество настроек по умолчанию, однако необходимо увеличить доступную память для JVM. Для этого установим значения определенных свойств как показано на рисунке 4.2:

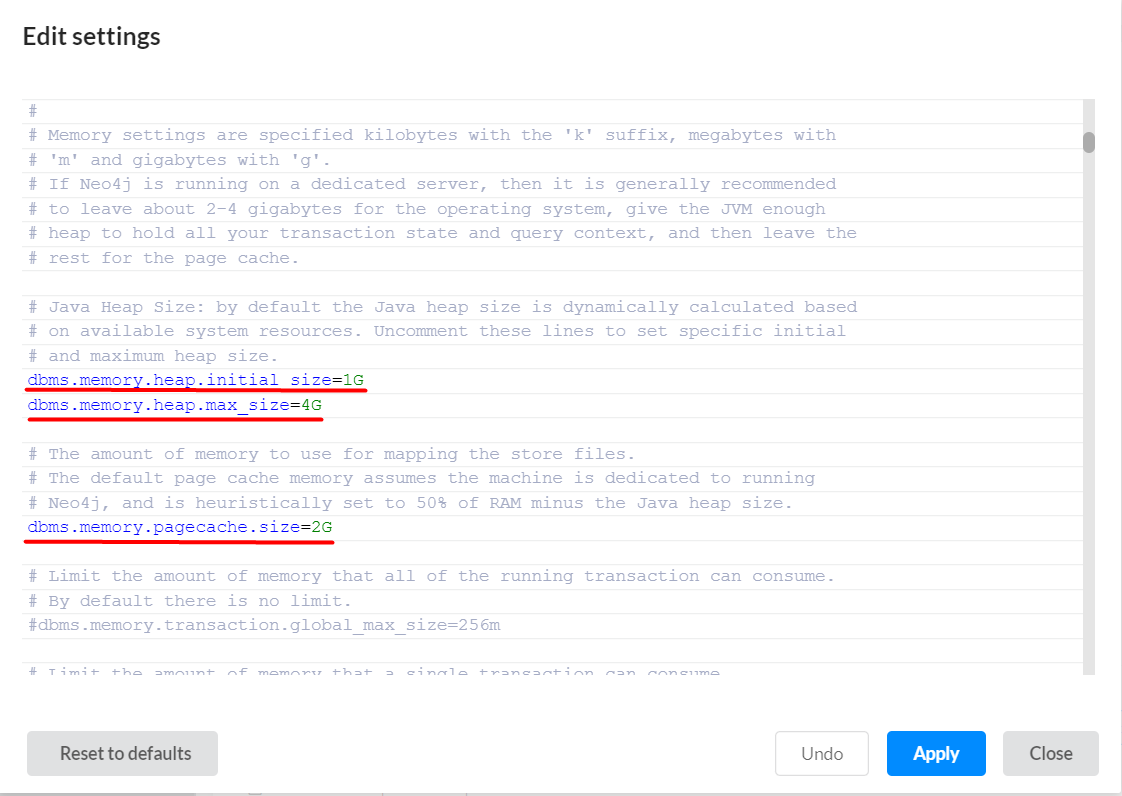


Рисунок 4.2 – Настраиваемые свойства JVM

Далее необходимо установить плагины *APOC* и *Neosemantics (n10s)*,которые используются для построения графовой БД Web-сайта.

Установить плагин Neosemantics можно несколькими способами. Можно либо загрузить готовый jar из области релизов**,** либо собрать его из исходников. Рассмотрим второй вариант.

Инструкция по установке n10s:

1. Устанавливаем сам плагин Neosemantics для базы данных (рисунок 4.3).

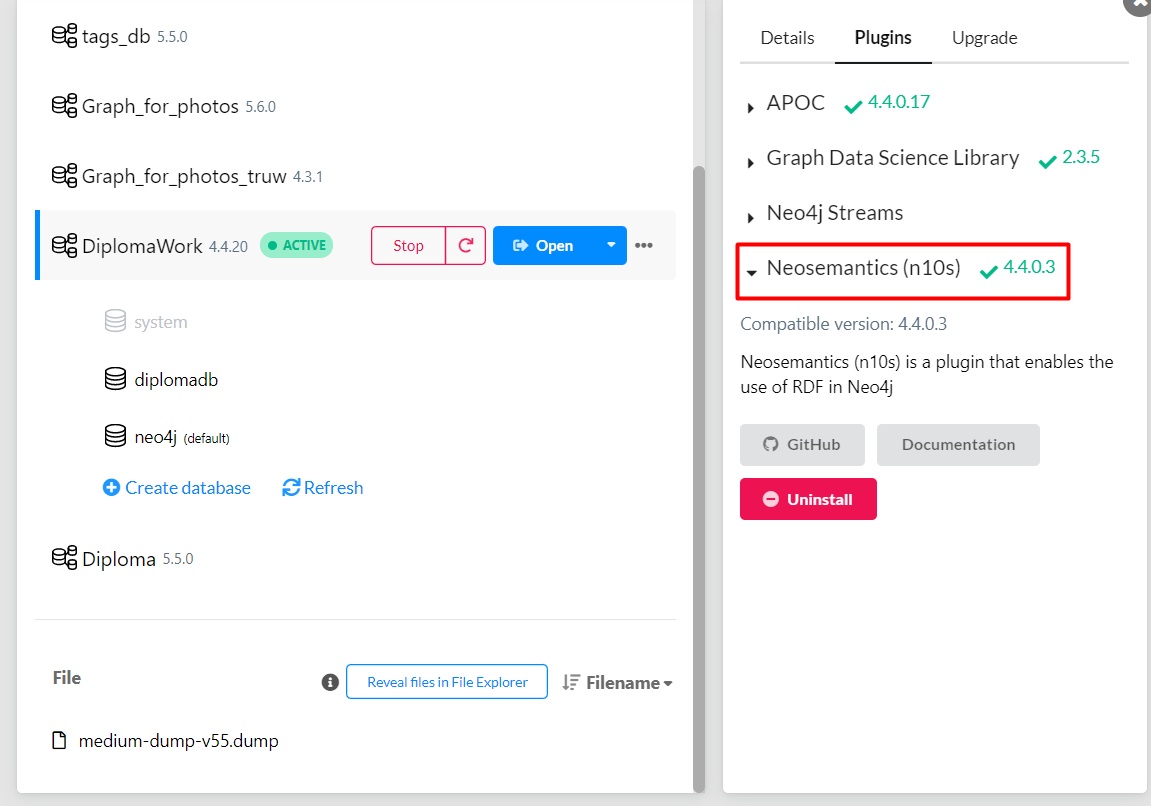


Рисунок 4.3 – Установка плагина Neosemantics

1. Добавляем следующую строку в ваш <NEO\_HOME> /conf/neo4j.conf и нажмите Apply (рисунок 4.4). Данная строка в настройках укажет БД, что в работе будет использоваться n10s.

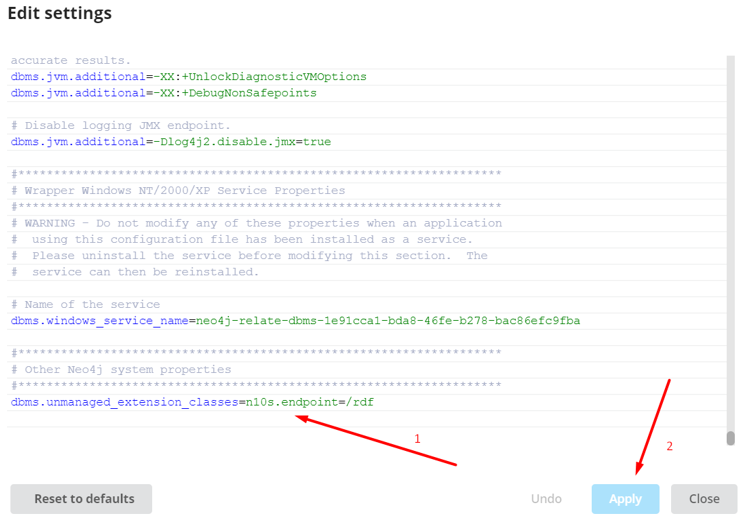


Рисунок 4.4 – Конфигурация базы данных

1. Чтобы проверить, что neosemantics установлен правильно, в Neo4j Browser вводим команду, представленную ниже. И если всё правильно установлено, то должен вывестись список команд n10s (рисунок 4.5).

call dbms.procedures()

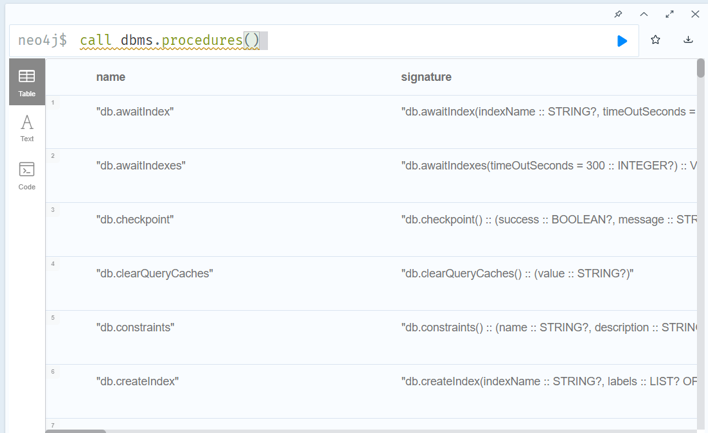


Рисунок 4.5 – Список команд n10s

1. Для проверки того, что расширение n10s установлено надо ввести команду, представленную на рисуноке 4.6*.* И если всё установлено правильно, то вы должны увидеть следующий вывод.

:GET http:*//localhost*[*:*](http://localhost:(имя)*(имя хоста)/rdf/ping*



Рисунок 4.6 – Проверка на правильность установки n10s

Если после выполнения запроса вывелось *{“ping”: “here!”}*, то плагин установлен правильно.

Чтобы установить плагин APOC потребуется войти в нужную базу данных, справа выбрать раздел Plugins, затем плагин APOC и нажать на кнопку Install (рисунок 4.7).



Рисунок 4.7 – Установка плагина APOC

После этого действия плагин будет установлен и может использоваться в запросах.

Также неообходимо установить плагины *Graph Data Science Library*, *Graphaware Server*, *Graphaware NLP*, *StanfordNLP* и *Stanford CoreNLP* с моделями. Все они поставляются в виде JAR-пакетов, и автоматически определяются, и подхватываются во время работы JVM. Для некоторых плагинов необходимо установить следующие настройки.

dbms.unmanaged\_extension\_classes=com.graphaware.server=/graphaware

com.graphaware.runtime.enabled=true

com.graphaware.module.NLP.1=com.graphaware.nlp.module.NLPBootstrapper

dbms.security.procedures.whitelist=ga.nlp.\*, apoc.\*, gds.\*

apoc.import.file.enabled=true

Листинг 4.1 – Настройки плагинов Neo4j

После всех этих действий можно запускать JVM. Это можно сделать из приложения Neo4j Desktop. Там же можно открыть Neo4j Browser, который предоставляет оболочку для написания Cypher запросов (рисунок 4.8).

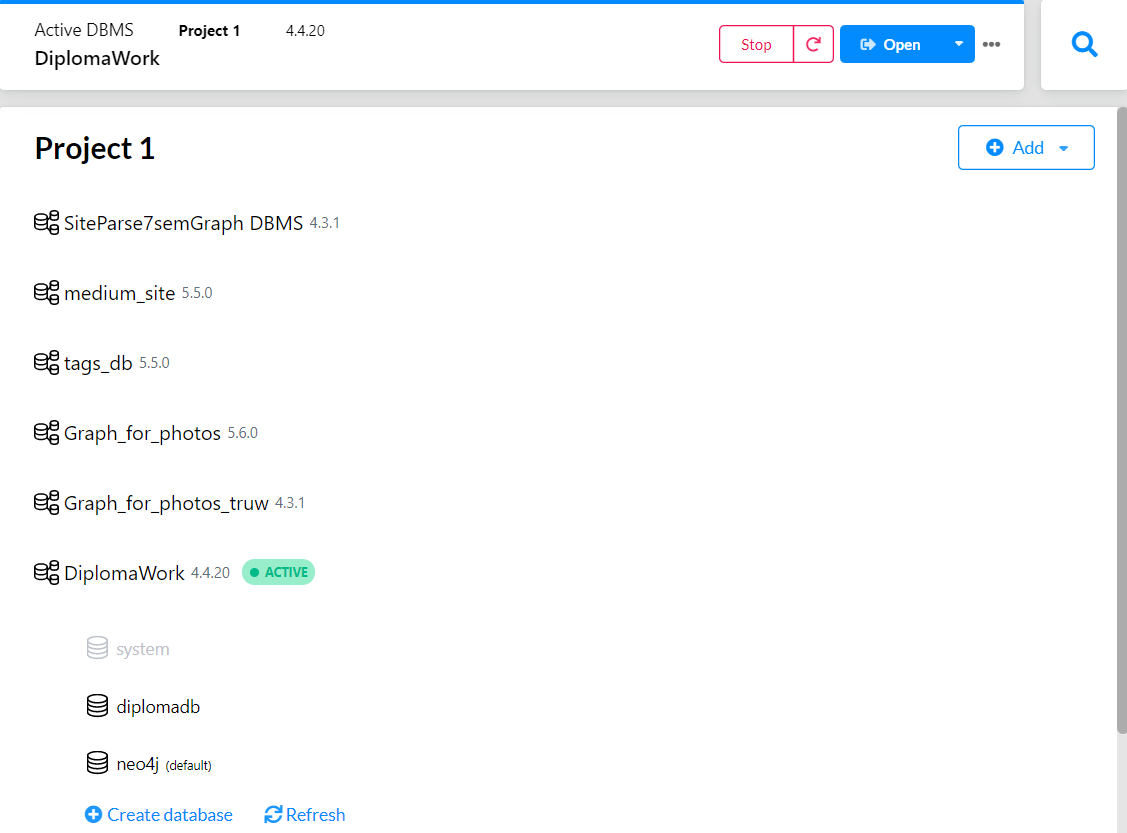


Рисунок 4.8 – Отображение статуса базы данных и панель управления

Для дальнейшей работы с данными формата RDF необходимо настроить Neo4j.

* 1. Настройка Neo4j для использования данных RDF

Перед импортом данных RDF в Neo4j и работы с ними, необходимо определить, как эти данные будут обрабатываться в графой БД. Это можно сделать с помощью Graph Config.

Все настройки, определенные в Graph Config, являются глобальными и остаются действительными в течение всего времени существования графа и будут определять поведение функций и процедур в n10s.

В дополнение к глобальным настройкам в Graph Config существуют другие параметры конфигурации, зависящие от метода, такие как, например, ограничение на количество троек, которые будет принимать конкретная процедура, языковой фильтр или запрос SPARQL для отправки конечной точке, производящей RDF.

Есть предварительное условие перед эффективным выполнением любого импорта данных в Neo4j с помощью Neosemantics: нужно определить характеристики графа, который мы будем создавать. Они хранятся в GraphConfig. Метод **n10s.graphconfig.init** помогает в этой настройке. Вызов процедуры без параметров установит все значения по умолчанию (рисунок 4.9).

CALL n10s.graphconfig.init();

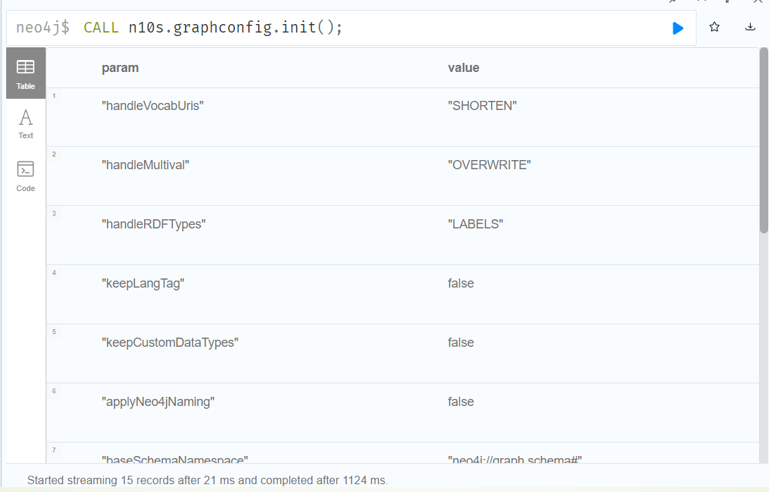


Рисунок 4.9 – Установка значений по умолчанию

Все методы, сохраняющие данные в Neo4j, имеют предварительное условие: это наличие ограничения уникальности для свойства uri узлов с меткой Resource. Если ограничение не присутствует, нужно запустить следующую команду в своей БД, иначе все процедуры импорта RDF выдадут сообщение об ошибке (рисунок 4.10).

CREATE CONSTRAINT n10s\_unique\_uri ON (r:Resource)

ASSERT r.uri IS UNIQUE;

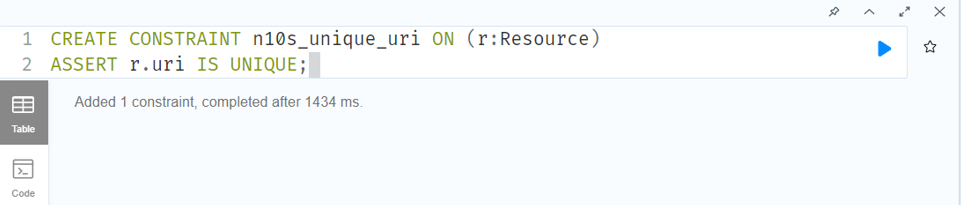


Рисунок 4.10 – Добавление уникальности для свойства uri

После данных манипуляций Neo4j готово к работе с RDF.

* 1. Заполнение графовой БД тестовыми данными

Для разработки понадобятся тестовые данные. Данные статей извлекаются из КиберЛенинки (российская научная электронная библиотека), Semantic Scholar, Medium и других источников. Наборы данных взяты из открытого источника.

Ниже приведены скрипты для подготовки базы данных и ее заполнения. В листинге 4.2 приведен скрипт добавления уникальных ограничений для нод с ярлыками «Article» и «Author» на атрибуты «index» и «name» соответственно:

CALL apoc.schema.assert(

{},

{Article: ['index'], Author: ['name']}

)

Листинг 4.2 – Добавление уникальных ограничений

Следующий скрипт заполняет таблицы данными из вышеупомянутых наборов (листинг 4.3):

CALL apoc.periodic.iterate(

'UNWIND ["dblp-ref-0.json","dblp-ref-1.json","dblp-ref-2.json",

"dblp-ref-3.json"] AS file

CALL apoc.load.json(file)

YIELD value RETURN value',

'MERGE (a:Article{index:value.id})

ON CREATE SET a +=

apoc.map.clean(value, ["id", "authors", "references"], [0])

WITH a, value.authors AS authors

UNWIND authors AS author

MERGE (b:Author {name: author})

MERGE (b)-[:AUTHOR]->(a)',

{batchSize: 10000, iterateList: true}

)

Листинг 4.3 – Заполнение БД данными

Далее необходимо добавить связи между сущностями данную операцию выполняет скрипт, представленный в листинге 4.4:

CALL apoc.periodic.iterate(

'UNWIND ["dblp-ref-0.json","dblp-ref-1.json","dblp-ref-2.json",

"dblp-ref-3.json"] AS file

CALL apoc.load.json(file)

YIELD value RETURN value',

'MERGE (a:Article {index: value.id})

WITH a, value.references AS references

UNWIND references AS reference

MERGE (b:Article {index: reference})

MERGE (a)-[:REFERENCES]->(b)',

{batchSize: 10000, iterateList: true}

)

Листинг 4.4 – Добавление связей между сущностями

На этом первичная настройка и заполнение БД заканчивается.

* 1. Cypher запросы для манипулирования данными, используемые на сайте

Определим следующие модели в соответствии со схемой базы данных, предоставленной в предыдущем разделе:

1. статья (Article),
2. автор (Author),
3. пользователь (User).

На сайте на странице DATA для представления данных используются специальные cypher запросы.

Для вывода всех статей, отсортированных по алгоритму PageRank, используется следующий скрипт, представленный в листинге 4.5:

MATCH (a:Article)

WHERE NOT a.title=''

RETURN DISTINCT a, a.pagerank, a.title, a.pk, a.id

ORDER BY a.pagerank DESC LIMIT 1000

Листинг 4.5 – Вывод статей по PageRank

Для вывода всех авторов, отсортированных по популярностью, используется следующий скрипт, представленный в листинге 4.6:

MATCH (a:Author)-[]-(at:Article)

WHERE NOT a.name=''

RETURN DISTINCT a.name as name, a.popularity, a.id

ORDER BY a.popularity DESC LIMIT 1000

Листинг 4.6 – Вывод авторов по популярности

На сайте на странице DATA MANIPULATION можно загружать данные в графовую базу данных. Для этого используется следующий скрипт, представленный в листинге 4.7:

UNWIND ['{link}'] as page  
CALL apoc.load.html (

page, {jsonld: 'head script[type="application/ld+json"]'})

YIELD value

CALL n10s.rdf.import.inline(value.jsonld[0].data, "JSON-LD")  
YIELD terminationStatus, triplesLoaded, triplesParsed, extraInfo

RETURN page, terminationStatus, triplesLoaded, triplesParsed, extraInfo

Листинг 4.7 – Загрузка данных через ссылку на статью

1. ТЕСТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

Тестирование программного обеспечения – процесс анализа программного средства и сопутствующей документации с целью выявления дефектов и повышения качества продукта . Вот уже несколько десятков лет его стабильно включают в планы разработки как одна из основных работ, причем выполняемая практически на всех этапах проектов. Важность своевременного выявления дефектов подчеркивается выявленной эмпирически зависимостью между временем допущения ошибки и стоимостью ее исправления: график данной функции круто возрастает.

В данной главе будет произведено тестирование программого средства, чтобы получить его полное представление. Будут рассмотрены основные сценарии использования и их результаты:

* регистрация пользователя;
* авторизация пользователя;
* вывод статей без фильтрации;
* поиск статей по их названию;
* вывод информации о статье;
* вывод всех авторов без фильтрации;
* вывод автора по фильтрации;
* вывод информации об авторе;
* вывод гистограммы статей по PageRank;
* вывод гистограммы авторов по популярности;
* загрузка данный с помощью файла со статьями;
* загрузка данный через ссылку на статью.

Тестирование и описание перечисленных процессов показано в таблицах 5.1 – 5.12.

Таблица 5.1 – Регистрация пользователя

|  |  |
| --- | --- |
| Наименование процесса | Описание процесса |
| Тестовый сценарий | 1 Перейти на страницу SIGN UP  2 Ввести имя пользователя, почту, пароль и подтверждённый пароль  3 Нажать на кнопку SIGN UP |
| Исходный набор данных |  |
| Ожидаемый результат | Произойдёт регистрация и пользователь попадёт на главную страницу |
| Фактический результат |  |

Таблица 5.2 – Авторизация пользователя

|  |  |
| --- | --- |
| Наименование процесса | Описание процесса |
| Тестовый сценарий | 1 Нажать на кнопку SIGN IN  2 Ввести имя пользователя и пароль  3 Нажать на кнопку SIGN IN |
| Исходный набор данных |  |
| Ожидаемый результат | Произойдёт авторизация и пользователь попадёт на главную страницу |
| Фактический результат |  |

Таблица 5.3 – Вывод статей без фильтрации

|  |  |
| --- | --- |
| Наименование процесса | Описание процесса |
| Тестовый сценарий | 1 Перейти на страницу DATA  2 Нажать на значок лупы |
| Исходный набор данных |  |
| Ожидаемый результат | Выведется таблица со всеми статьями |
| Фактический результат |  |

Таблица 5.4 – Поиск статей по их названию

|  |  |
| --- | --- |
| Наименование процесса | Описание процесса |
| Тестовый сценарий | 1 Перейти на страницу DATA  2 В поле search ввести «big Data»  3 Нажать на значок лупы |
| Исходный набор данных |  |
| Ожидаемый результат | Выведется таблица со всеми статьями, в названии которых есть словосочетание «big data» |
| Фактический результат |  |

Таблица 5.5 – Вывод информации о статье

|  |  |
| --- | --- |
| Наименование процесса | Описание процесса |
| Тестовый сценарий | 1 Перейти на страницу DATA  2 Нажать на значок лупы  3 Выбрать одну из статей и нажать на неё |
| Исходный набор данных |  |
| Ожидаемый результат | Во всплывающем окне выведется информация о выбранной статье |
| Фактический результат |  |

Таблица 5.6 – Вывод всех авторов без фильтрации

|  |  |
| --- | --- |
| Наименование процесса | Описание процесса |
| Тестовый сценарий | 1 Перейти на страницу DATA  2 Нажать на значок лупы  3 Нажать на вкладку Author |
| Исходный набор данных |  |
| Ожидаемый результат | Выведется таблица со всеми авторами |
| Фактический результат |  |

Таблица 5.7 – Вывод автора по фильтрации

|  |  |
| --- | --- |
| Наименование процесса | Описание процесса |
| Тестовый сценарий | 1 Перейти на страницу DATA  2 Во всплывающем списке By authors выбрать автора  3 Нажать на значок лупы |
| Исходный набор данных |  |
| Ожидаемый результат | Выведется выбранный автор |
| Фактический результат |  |

Таблица 5.8 – Вывод информации об авторе

|  |  |
| --- | --- |
| Наименование процесса | Описание процесса |
| Тестовый сценарий | 1 Перейти на страницу DATA  2 Нажать на значок лупы  3 Нажать на вкладку Author  4 Выбрать автора и нажать на него |
| Исходный набор данных |  |
| Ожидаемый результат | Во всплывающем окне выведется информация о выбранном авторе |
| Фактический результат |  |

Таблица 5.9 – Вывод гистограммы статей по PageRank

|  |  |
| --- | --- |
| Наименование процесса | Описание процесса |
| Тестовый сценарий | 1 Перейти на страницу STATISTICS  2 В поле Number of columns ввести число количество статей, которое должны быть в гистограмме  3 Нажать на кнопку SUBMIT |
| Исходный набор данных |  |
| Ожидаемый результат | Вывод гистограммы с 1000 статьями, отсортированными по PageRank |
| Фактический результат |  |

Таблица 5.10 – Вывод гистограммы авторов по популярности

|  |  |
| --- | --- |
| Наименование процесса | Описание процесса |
| Тестовый сценарий | 1 Перейти на страницу STATISTICS  2 В выпадающем списке Node type выбрать Aurthor  3 В поле Number of columns ввести число количество авторов, которое должны быть в гистограмме  4 Нажать на кнопку SUBMIT |
| Исходный набор данных |  |
| Ожидаемый результат | Вывод гистограммы с 1000 авторами, отсортированными по популярности |
| Фактический результат |  |

Таблица 5.11 – Загрузка данный с помощью файла со статьями

|  |  |
| --- | --- |
| Наименование процесса | Описание процесса |
| Тестовый сценарий | 1 Перейти на страницу DATA MANIPULATION  2 Нажать на кнопку Choose file…  3 Выбрать файл с ссылками на статьи  4 Нажать на кнопку SUBMIT |
| Исходный набор данных |  |
| Ожидаемый результат | Все статьи, которые были в выбранном файле, загрузятся в графовую БД и будут отображены на сайте |
| Фактический результат |  |

Таблица 5.12 – Загрузка данный через ссылку на статью

|  |  |
| --- | --- |
| Наименование процесса | Описание процесса |
| Тестовый сценарий | 1 Перейти на страницу DATA MANIPULATION  2 В поле Enter an https:// URL: ввести ссылку на нужную статью  3 Нажать на кнопку SUBMIT |
| Исходный набор данных |  |
| Ожидаемый результат | Все статья, ссылку на которую пользователь ввёл, загрузится в графовуб БД проекта и будет отображена на сайте |
| Фактический результат |  |

В результате итогового тестирования серьезных дефектов выявлено не

было. Все осноные сценарии были пройдены успешно. Фактический результат

совпал с ожидаемым результатом в каждом из сценариев.

1. ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ КОМПОНЕНТОВ СКА ИИ «СОЗДАНИЕ ТЕМАТИЧЕСКИХ ГРАФОВЫХ БД» И «ИНТЕГРАЦИЯ ДОСТУПА К КОМПОНЕНТАМ СИСТЕМЫ»

## **Краткая характеристика программного продукта**

Разрабатываемое в дипломном проекте программное обеспечение системы комплексного анализа интернет источников позволяет определить наиболее важные публикации в некоторой области (например, в космических исследованиях, здравоохранении, в социальной сфере), провести тематический анализ этих публикаций и выявить лидера научного направления.

Основными задачами данного дипломного проекта является разработка программного обеспечения системы комплексного анализа интернет источников, поиск архитектурных решений быстрого построения компонента тематического прототипа графовой БД и внедрение компонента в программное обеспечение для дальнейшего анализа данных.

Разрабатываемый продукт, предназначенный для решения вышеперечисленных задач, имеет следующие основные функции:

* регистрация, аутентификация и авторизация пользователей;
* поддержка системы ролей (администраторы, пользователи с расширенными правами и обычные пользователи);
* просмотр информации о публикациях различных предметных областей и авторах;
* просмотр параметров некоторой предметной области;
* поиск публикации по названию и автора по имени;
* выполнение произвольного запроса к графу знаний;
* управление анализом данных до нескольких сот тысяч документов;
* загрузка данных с тематических сайтов в базу данных разными способами.

Таким образом, данный программный продукт может быть использован как для прикладных целей, например, определения наиболее перспективных и актуальных научных направлений с учетом конкретной специфики (для конкретной организации или страны, с учетом имеющегося задела и т. д.), определения экспертов в заданных предметных областях (для приглашения специалистов или формирования команды исследователей), так и для создания интеллектуальной системы на основе совместного применения графовых моделей некоторой предметной области и аналитических методов ML-алгоритмов для глубокого интеллектуального анализа путем применения методов искусственного интеллекта на основе преобразований семантических данных из графовой модели в различные матричные представления для их анализа в ML.

Организацией-заказчиком данного программного продукта является Белорусский государственный университет информатики и радиоэлектроники.

Продукт разрабатывается по заказу, что подразумевает, что пользоваться продуктом будут сотрудники организации-заказчика.

## **6.2 Расчет затрат на разработку и цена программного обеспечения, созданного по индивидуальному заказу**

### **6.2.1** Затраты на основную заработную плату разработчиков

Расчёт затрат на основную заработную плату команды, занимающейся его разработкой и поддержкой, определяется по следующей формуле:

(6.1)

где n – количество исполнителей, занятых разработкой конкретного ПО;

– коэффициент премий (составляет 1,5);

Зчi – часовая заработная плата i-го исполнителя (р.);

ti – трудоемкость работ, выполняемых i-м исполнителем (ч).

Расчётная норма рабочего времени составляет 168 часов. Срок разработки данного приложения – шесть месяцев (1028 рабочих часов).

Проанализировав требования, принято решение, что для разработки данного проекта необходимо привлечь следующих специалистов:

* Проектный менеджер (привлекается на начальных этапах разработки проекта, часовой оклад – 20,82 рубля).
* Бизнес-аналитик (привлекается периодически по мере разработки проекта, часовой оклад – 16,19 рублей)
* Инженер-программист, Frontend разработчик (привлекается на половину периода разработки проекта, часовой оклад – 15,29 рубля).
* Инженер-программист, Backend разработчик (привлекается на весь период разработки проекта, часовой оклад – 15,91 рубль).
* Тестировщик (привлекается на последнюю треть периода разработки, часовой оклад – 12,03 рубля).

Размеры должностных окладов специалистов указаны по состоянию на 31.03.2023. Расчёты затрат на основную заработную плату команды разработчиков с учётом их трудозатрат по проекту представлены в таблице 6.1.

Таблица 6.1 – Расчет затрат на основную заработную плату команды разработчиков.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Категория исполнителя | Месячный оклад, р. | Часовой оклад, р. | Трудоём-кость работ, ч. | Итого, р. |
| Проектный менеджер | 3 497,76 | 20,82 | 40 | 832,8 |
| Бизнес-аналитик | 2 719,92 | 16,19 | 64 | 1 036,16 |
| Frontend разработчик | 2 568,72 | 15,29 | 252 | 3 853,08 |
| Backend разработчик | 2 672,88 | 15,91 | 504 | 8 018,64 |
| Тестировщик | 2 021,04 | 12,03 | 168 | 2 021,04 |
| *Итого* | | | | 15 761,72 |
| Премия(50%) | | | | 7 880,86 |
| Итого затраты на основную заработную плату разработчиков | | | | 23 642,68 |

**6.2.2** Затраты на дополнительную заработную плату разработчиков

Затраты на дополнительную заработную плату команды разработчиков определяется по формуле:

(6.2)

где – затраты на основную заработную плату, (р.);

– норматив дополнительной заработной платы (15%).  
Рассчитаем затраты на дополнительную заработную плату:

**6.2.3** Отчисления на социальные нужды

Отчисления на социальные нужды (в фонд социальной защиты населения и на обязательное страхование) определяются в соответствии с действующими законодательными актами Республики Беларусь по формуле:

(6.3)

где ‒ норматив отчислений на социальные нужды, %, = 34,6

Сумма отчислений на социальные нужды, согласно формуле (6.3), составит:

**6.2.4** Прочие затраты

Расходы по данной статье определяются по формуле:

, (6.4)

где – норматив прочих в целом по организации.

Приняв значение равным 15% и подставив в формулу (6.4) оставшиеся значения, произведем расчет прочих затрат:

**6.2.5** Полная сумма затрат

Общая сумма затрат на разработку программного обеспечения системы комплексного анализа интернет источников находится путем суммирования всех рассчитанных статей затрат. Расчет общей суммы затрат представлен в таблице 6.2.

Таблица 6.2 — Общая сумма затрат на разработку программного обеспечения.

|  |  |
| --- | --- |
| Статья затрат | Сумма, р. |
| Основная заработная плата команды разработчиков | 23 642,68 |
| Дополнительная заработная плата команды разработчиков | 3 546,39 |
| Отчисления на социальные нужды | 9 407,42 |
| Прочие расходы | 3 546,4 |
| Общая сумма затрат на разработку | 40 142,89 |

Полная себестоимость разрабатываемого программного обеспечения системы комплексного анализа интернет источников составит 40 142,89 рублей.

**6.2.6** Плановая прибыль

Расчеты по данной статье исходят из нужд на развитие производства и затрат на заработную плату команде.

, (6.5)

где Рпс – рентабельность затрат на разработку программного средства.

Приняв значение Рпс равным 30% и подставив в формулу (6.5) оставшиеся значения, произведём расчёт плановой прибыли:

**6.2.7** Отпускная цена программного средства

Расчеты по данной статье исходят из затрат на заработную плату и плановой прибыли.

, (6.6)

Произведя подстановку выше полученных значений в формулу (6.6), получим отпускную цену программного средства:

Цена программного обеспечения системы комплексного анализа интернет источников для организации-заказчика является приемлемой, так как цена продукта не превышает уровень цен других схожих программных средств и составит 52 185,76 р.

**6.3 Оценка результата от продажи программного обеспечения**

Экономический эффект для организации-разработчика программного обеспечения системы комплексного анализа интернет источников заключается в получении прибыли от его реализации, а также его поддержки в будущем, по согласованию с заказчиком.

В результате расчет инвестиций в разработку программного средства выявлено, что стоимость данного продукта составит 52 185,76 рублей.

Так как программное средство реализовывается организацией- разработчиком по отпускной цене, сформированной на основе затрат на разработку, то экономический эффект, полученный организацией-разработчиком, в виде прироста чистой прибыли от его разработки, определяется по формуле:

, (6.7)

где Ппс ‒ прибыль, включаемая в цену программного средства, р.

Таким образом, экономический эффект, полученный организацией- разработчиком, в виде прироста чистой прибыли от его разработки и продажи заказчику составит 9 634,3 рубля.

## **6.4 Расчет показателей эффективности инвестиций в разработку программного обеспечения**

Для разработчика программного средства оценка экономической эффективности разработки осуществляется с помощью расчета простой нормы прибыли. Такая рентабельность затрат на разработку программного средства рассчитывается по формуле:

(6.8)

где ∆Пч ‒ прирост чистой прибыли, полученной от разработки программного средства организацией-разработчиком по индивидуальному заказу, р.;

Зр ‒ затраты на разработку программного средства организацией- разработчиком, р.

В соответствии с формулой (6.8), рентабельность инвестиций составит:

Рассчитанный показатель показывает, что разработка имеет положительный экономический эффект.

В ходе технико-экономического обоснования разрабатываемого программного продукта произведён расчёт затрат на разработку программного обеспечения, выполнена оценка экономического эффекта от его продажи, произведён расчёт эффективности инвестиций в разработку, а также показана его экономическая целесообразность.

Чистая прибыль от реализации программного продукта составит 10485,43 рублей.

Исходя из произведённых расчётов и выполненных оценок, можно сделать вывод, что проект является экономически выгодным.