Учреждение образования

 «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Кафедра интеллектуальных информационных технологий

**Отчет по лабораторной работе №3**

**по курсу «Проектирование баз знаний»**

**на тему:**

**“Приложение для работы с онтологиями”**

Выполнила студентка

группы 021703:                   Склема Е.Д.

Проверила:       Василевская А. П.

Минск

2023

1. **Обоснование выбора языка и средств реализации**

В качестве языка программирования для реализации был выбран **Python**.

Это язык программирования общего назначения, нацеленный в первую очередь на повышение продуктивности самого программиста, нежели кода, который он пишет. Говоря простым человеческим языком, на Python можно написать практически что угодно (веб-/настольные приложения, игры, скрипты по автоматизации, комплексные системы расчёта, системы управления жизнеобеспечением и многое другое) без ощутимых проблем. Более того, порог вхождения низкий, а код во многом лаконичный и понятный даже тому, кто никогда на нём не писал. За счёт простоты кода, дальнейшее сопровождение программ, написанных на Python, становится легче и приятнее по сравнению с Java или C++.

Для работы с онтологиями была выбрана библиотека [**RDFLib**](https://docs.google.com/document/d/12584QKdSAxiq1inXA0Z0BS85jXfwGz6Y9G69sTEQIzw/edit#heading=h.6ojq6edwg9r8) . Это графовая база данных, которая поддерживает RDF и SPARQL и предоставляет множество инструментов для работы с онтологиями.

Для пользовательского интерфейса была использована библиотека **TKinter**. Это пакет для Python, предназначенный для работы с библиотекой Tk. Библиотека Tk содержит компоненты графического интерфейса пользователя.

Под графическим интерфейсом пользователя подразумеваются окна, кнопки, текстовые поля и поля ввода, списки и др., которые мы видим на экране, открывая приложения. Через них вы взаимодействуете с программой и управляете ею.

Далее рассмотрим их преимущества для работы с онтологиями:

1. **Простота**. Python является простым и гибким языком программирования, который позволяет легко и быстро создавать код для работы с онтологиями. RDFLib также предоставляет удобный и гибкий интерфейс для работы с графовыми данными.
2. **Поддержка различных форматов данных**. RDFLib поддерживает различные форматы данных, такие как RDF/XML, Turtle, N-Triples, JSON-LD и другие, что позволяет работать с графовыми данными в различных форматах.

Таким образом это упрощает процесс работы с онтологиями и позволяет быстро находить решения для различных задач.

**2. Структура онтологии “Породы кошек”**

2.1. Онтология “ Породы кошек ”

Назначение онтологии " Породы кошек " состоит в том, чтобы исследовать породы кошек и иметь возможность описать их характеристики: отношения между той и иной породой, стоимость, место их возниконовения, день рождения и возраст. Онтология помогает пользователю лучше понять какая порода образовалась раньше, а какая позже и как они взаимосвязаны.

В данной лабораторной работе рассматривались такие понятия как виды кошек, возраст, когда они появились, стоимость той или иной породы.

2.2 Иерархия понятий классов

Для разработки онтологии требуется специальная программа, выбор которой пал на Protege.

Protege - это мощный инструмент для разработки онтологий и экспертных систем. Плюсами которой являются:

1. Бесплатное и открытое программное обеспечение: доступен бесплатно и его исходный код можно свободно скачать и изменять в соответствии с нуждами пользователя.
2. Интуитивно понятный пользовательский интерфейс: имеет интуитивно понятный пользовательский интерфейс, который позволяет пользователям легко создавать и редактировать онтологии.
3. Мощные функции редактирования онтологий: Protege предоставляет множество мощных функций для создания, редактирования и визуализации онтологий, таких как поддержка нескольких форматов онтологий, возможность импортирования и экспорта онтологий, автоматическое создание классов, свойств и ограничений и многое другое
4. Поддержка широкого спектра языков и форматов: поддерживает множество языков и форматов, таких как RDF, OWL, XML, RDFS, JSON, CSV и другие, что позволяет пользователям использовать Protege с различными системами и приложениями.
5. Широкие возможности интеграции: интегрируется с множеством других инструментов и приложений, таких как статистические пакеты R и Python, что позволяет пользователям создавать сложные экспертные системы и аналитические приложения.
6. Масштабируемость: Protege позволяет пользователям работать с множеством онтологий одновременно, что делает его удобным инструментом для разработки крупных экспертных систем и баз знаний.

Онтология **“Породы кошек”** включает в себя **11** классов, из которых **8** являются подклассами), **22** объекта классов и **4** отношения типа **DataProperty.** Каждый класс, в том числе подкласс, имеет не менее 2-ух объектов, для каждого объекта существует не менее 2-ух отношений

Для начала рассмотрим классы на Рис 1:

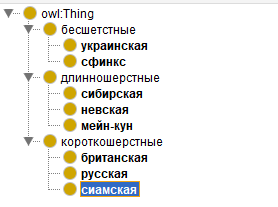


Рис 1. Классы онтологии

Каждый класс соответствует определенному типу кошек.

Подклассы раскрывают и конкретизируют соответствующие классы породы

кошек.

Рассмотри объекты классов (Рис 2).



Рис 2. Объекты классов

В данной онтологии есть 8 отношений, 4 из которых DataProperties и 4 ОbjectProperties, которые связывают классы и объекты(Рис 3, 4).

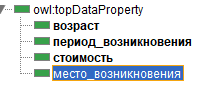


Рис 3. DataProperties

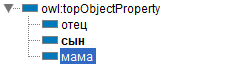


Рис 4. ОbjectProperties

Таким образом, данная онтология позволяет изучать породы кошек. И дает возможность расширять эту онтологию для большего погружения.

**3. Описание разработанного приложения**

3.1 Общая структура приложения

Структура данного приложения.

1. Запуск программы.

После запуска программы мы видим окно графического пользовательского интерфейса, где есть кнопки: Запустить, Добавить, Изменить, Обновить, Удалить, SPARQL запросы, Сохранить.

* 1. Кнопка «Загрузить» позволяет нам включить имеющуюся онтологию в программу и вывести все столбцы.
  2. Кнопка «Добавить» дает возможность добавить Класс, Объект, Подкласс и т.д.
  3. «Изменить» помогает добавить изменения в уже существующие классы, подклассы или объекты.
  4. Кнопка «Обновить» дает возможность обновить список и просмотреть новые изменения.
  5. Так же мы можем удалить классы, подклассы и т.д. с помощью кнопки «Удалить».
  6. Для создания SPARQL запросов нужно нажать кнопку «, SPARQL запросы» и в появившемся окошке написать наш запрос, программа в том же окошке выдаст нам ответ на запрос.
  7. Все изменения можно сохранить в новый файл на компьютере в любом удобном для вас месте с помощью кнопки «Сохранить»

1. Когда мы выполнили загрузку, на экране появляется вся нужная нам информация из выбранной онтологии.

2.1) На страничке «Classes» добавляются все созданные нами классы в нашей онтологии.

2.2) На страничке «Property» видны все созданные отношения между индивидами.

2.3) Страница «Data Properties» выводит Субъект-Отношение-Объект.

2.4) Страничка «Individuals» показывает все созданные нами объекты классов.

3.2 Описание RDF-хранилища

Для поддержки хранения в RDFLib используются графы. Каждый субъект добавляется в граф в виде тройки «Субъект-Отношение-Объект».

3.4 Примеры работы программы

3.4.1 При запуске программы первым делом мы видим пустое окно, в котором нет никаких онтологий и информации по ним. (Рис 5.)

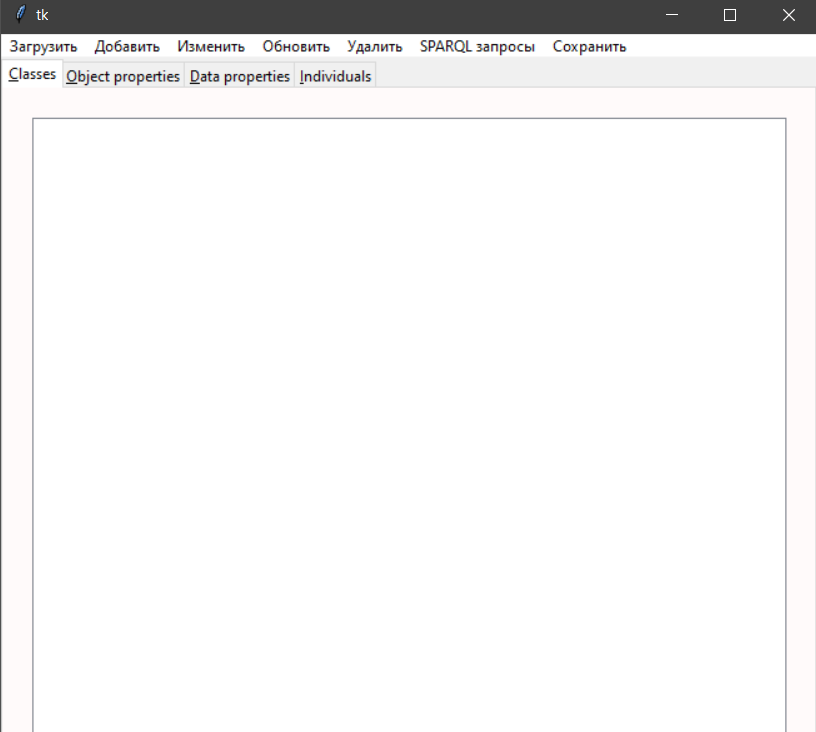


Рис 5. Начальное окно

Для загрузки онтологии нам нужно нажать кнопку «Загрузить».

После загрузки сохраненной онтологии в формате «\*.owl» мы увидим всю информацию (Рис 6,7,8,9)

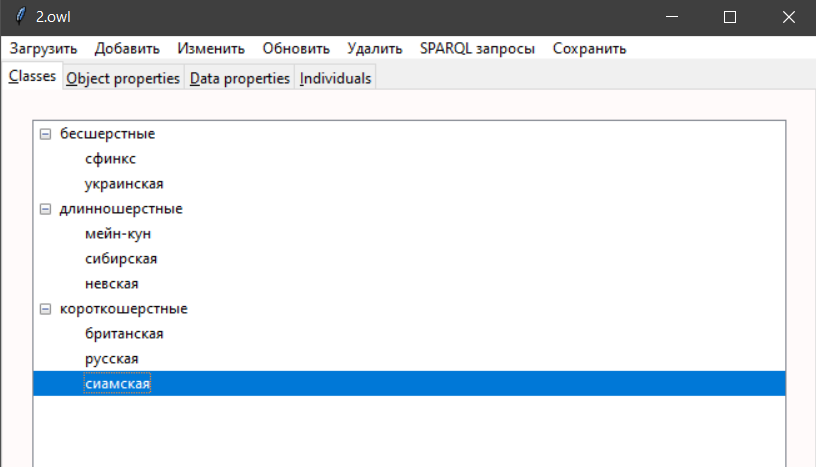


Рис 6. Classes

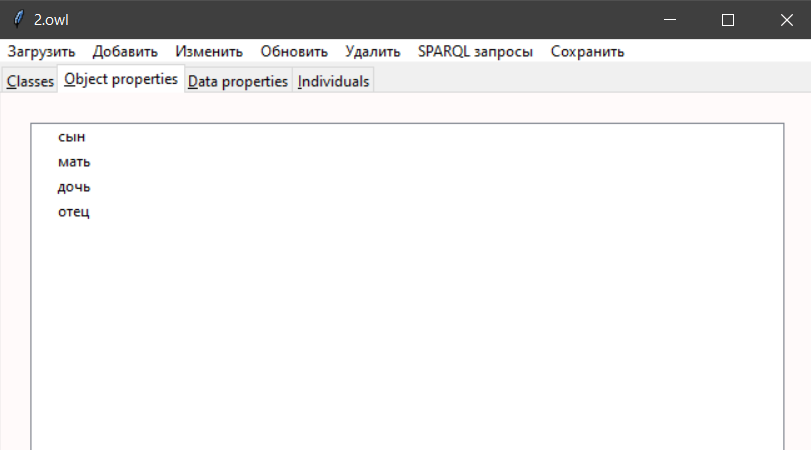


Рис 7. Object properties

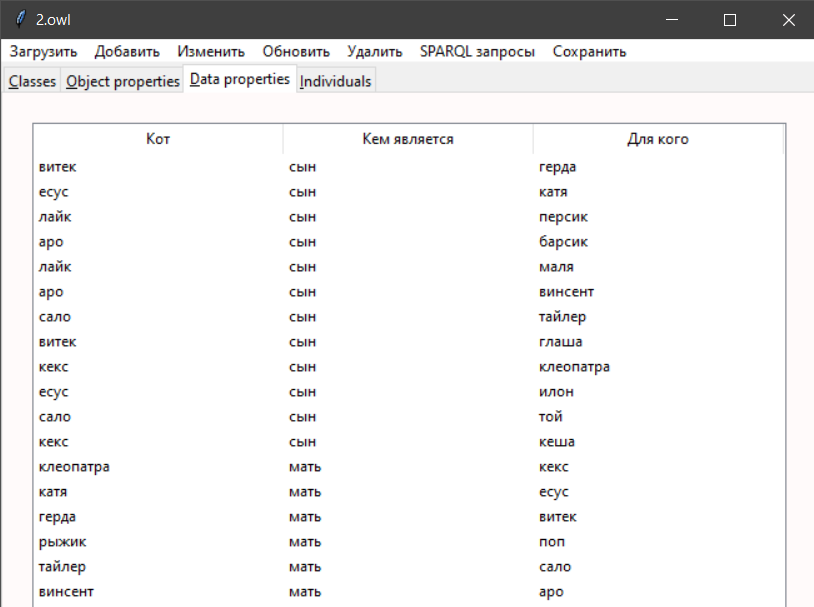
****

Рис 8. «Субъект-Отношение-Объект»

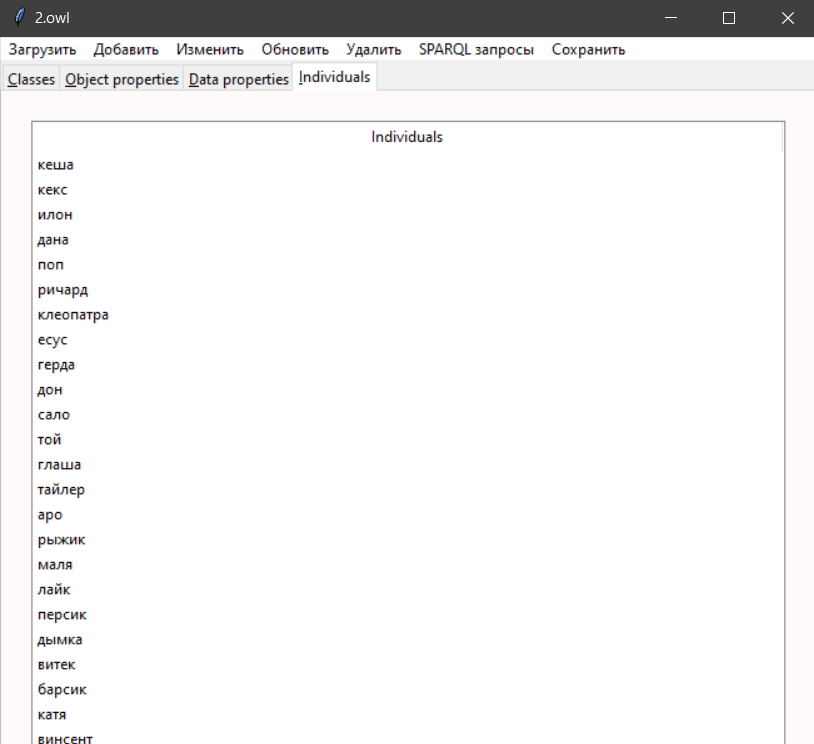


Рис 9. Individuals

В данной программе мы можем добавить, изменить и удалить новый class, properties и individual.

Добавление объекта представлено на рис.10, 11,12.

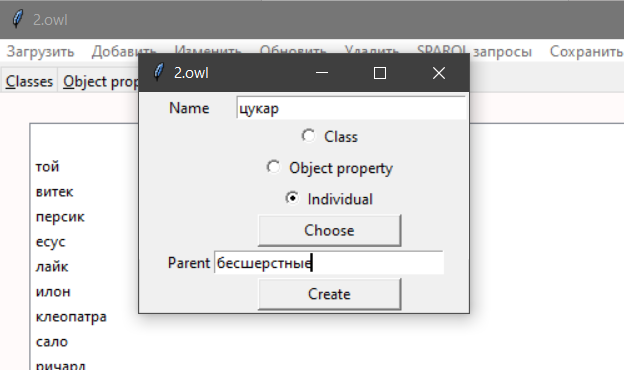


Рис 10. Добавление

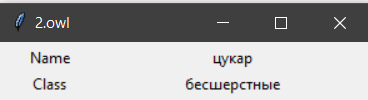


Рис 10. Просмотр общей информации

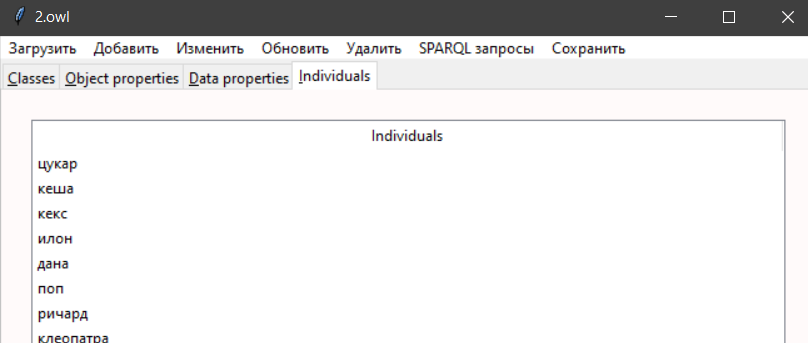


Рис 12. Результат добавления

Добавление класса и подкласса представлено на рис.13, 14.

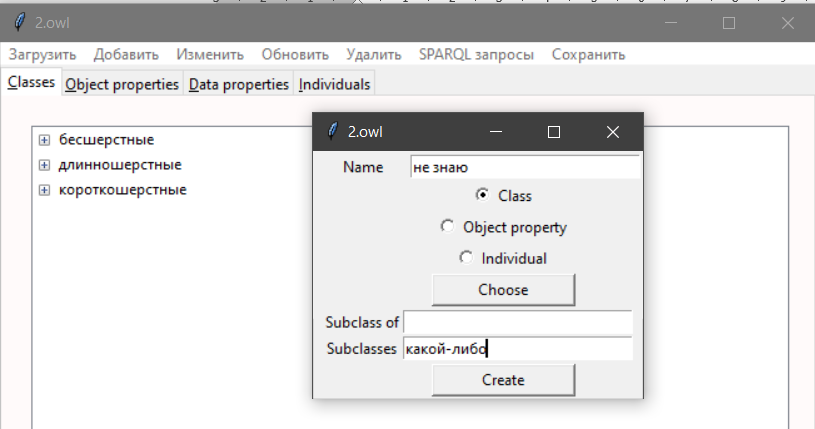


Рис 13. Добавление

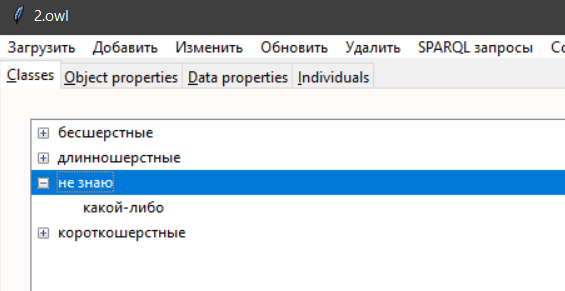


Рис 14. Результат добавления

Так же можно удалить класс или подкласс. Да и в принципе любую информацию. Рис 15, 16.

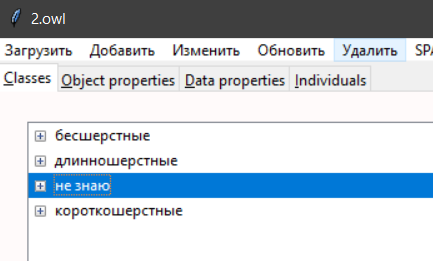


Рис 15. Удаление

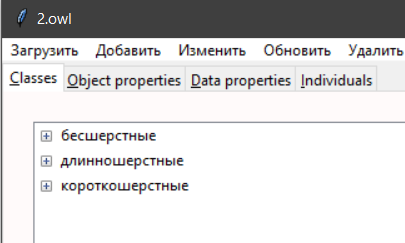


Рис 15. Результат

3.5 SPARQL-запросы

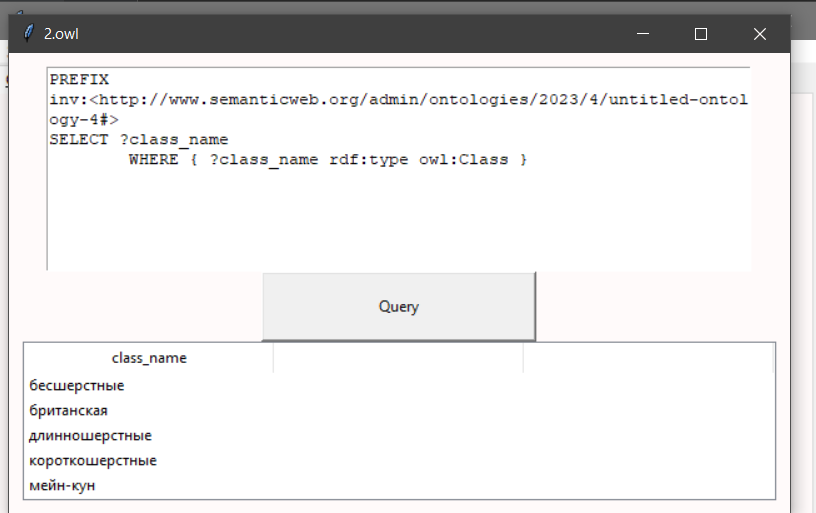


Рис 16. Вывод классов

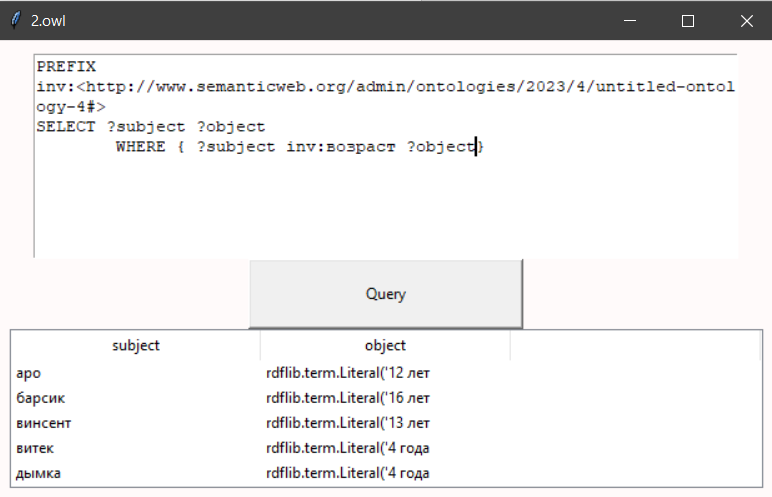


Рис 17. Вывод объектов и их возраст

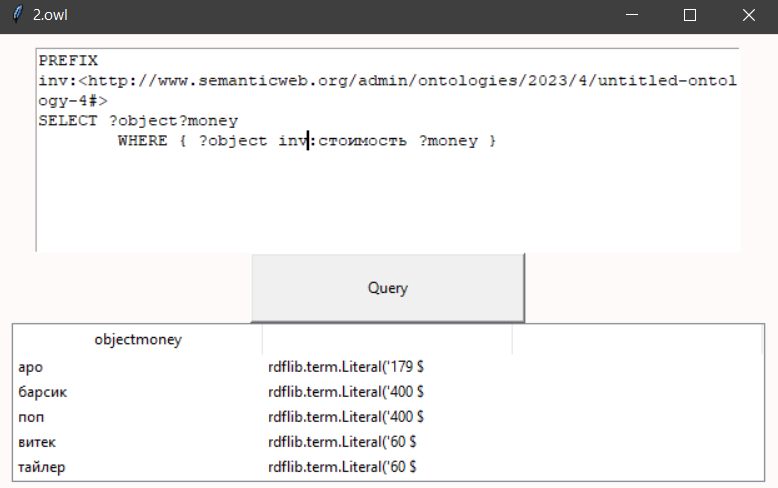


Рис 18. Вывод объектов и их стоимость

ВЫВОД

В ходе лабораторной работы были выполнены следующие задачи:

1. Разработана онтология содержащая как иерархический набор понятий, так и экземпляры понятий и связи между ними;
2. Полученная онтология погружена в RDF-хранилище RDFLib развернутое локально;
3. Разработано графическое приложение, позволяющее  просматривать информацию из погруженной в хранилище онтологии, создавать в  ней новые экземпляры и, при необходимости, классы, редактировать имеющиеся  экземпляры и классы;
4. Реализованы поисковые запросы по нескольким  параметрам.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[] Luciano, Ramalho. Fluent Python / Ramalho Luciano. — 2022.

[] Wikipedia [Электронный ресурс]. <https://ru.wikipedia.org/wiki/>

[] Д.И. Муромцев. Онтологический инжиниринг знаний в системе Protégé. – СПб: СПб ГУ ИТМО, 2007. – 62 с.

[] Сайт по работе с GraphDB - <https://graphdb.ontotext.com/>

[] Работа с platuml - <https://plantuml.com/ru/>