**Министерство науки и высшего образования Российской Федерации**

**федеральное государственное автономное образовательное учреждение высшего образования**

**“НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО”**

|  |  |
| --- | --- |
| **Факультет** | **Программной Инженерии и Компьютерной Техники** |
| **Направление подготовки (специальность)** |  |
| **Дисциплина** | **Системы искусственного интеллекта** |

Лабораторная работа 2

ОТЧЕТ

|  |  |
| --- | --- |
| **Выполнил студент:** | ***Алхимовици Арсений(408138)*** |
| **Группа:** | **P3310** |
| **Преподаватель:** | **Болдырева Елена Александровна (157150)** |

г. Санкт-Петербург

2025

Содержание

[***ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ*** 2](#_Toc210149369)

[***ОТЧЕТ О ХОДЕ ВЫПОЛНЕНИЯ*** 3](#_Toc210149370)

[***Онтология в Protégé*** 3](#_Toc210149371)

[***Реализация обращения к онтологии*** 4](#_Toc210149372)

[***Заключение*** **Error! Bookmark not defined.**](#_Toc210149373)

# ***­­­***

# ***ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ***

* Создать программу, которая позволяет пользователю ввести запрос через командную строку. Например, информацию о себе, своих интересах и предпочтениях в контексте выбора видеоигры - на основе фактов из БЗ (из первой лабы)/Онтологии(из второй).
* Использовать введенные пользователем данные, чтобы выполнить логические запросы к БЗ/Онтологии.
* На основе полученных результатов выполнения запросов, система должна предоставить рекомендации или советы, связанные с выбором из БЗ или онтологии.
* Система должна выдавать рекомендации после небольшого диалога с пользователем.

# ***ОТЧЕТ О ХОДЕ ВЫПОЛНЕНИЯ***

## ***Онтология в Protégé***

A screenshot of a computer

AI-generated content may be incorrect.

*Рисунок 1 – Граф Protege.*

***Реализация обращения к онтологии***

|  |
| --- |
| def list\_individuals(self) -> List[str]:          q = f"""          PREFIX onto: <{self.base\_ns}>          SELECT DISTINCT ?s WHERE {{              ?s a onto:Person .          }}          """          res = {str(row[0]).split("#")[-1] for row in self.graph.query(q)}          q2 = f"""          PREFIX onto: <{self.base\_ns}>          SELECT DISTINCT ?s WHERE {{              {{ ?s a onto:Male }} UNION {{ ?s a onto:Female }}          }}          """          res.update(str(row[0]).split("#")[-1] for row in self.graph.query(q2))          return sorted(res)    def is\_male(self, person\_local: str) -> bool:  q = f"""  PREFIX onto: <{self.base\_ns}>  ASK {{ <{self.\_local(person\_local)}> a onto:Male }}  """  return bool(self.graph.query(q).askAnswer)  def \_explicit\_siblings\_of(self, person\_local: str) -> List[str]:        q = f"""        PREFIX onto: <{self.base\_ns}>        SELECT DISTINCT ?s WHERE {{            {{ <{self.\_local(person\_local)}> onto:sibling ?s }}            UNION            {{ ?s onto:sibling <{self.\_local(person\_local)}> }}        }}        """  return [str(row[0]).split("#")[-1] for row in self.graph.query(q)] |

***Пример использования и вывод программы***

Пример 1:

|  |
| --- |
| Доступные команды:  - help: показать помощь  - list: показать всех индивидуалов из онтологии  - exit: выйти  Доступные отношения:  - мать  - отец  - родители  - братья\_сёстры  - братья  - сестры  - тети  - дяди  - кузены  - двоюродные\_братья  - бабушки\_дедушки  - внуки  Пример:  Я mike ищу тети кузены  Введите строку: 'Я: <индивидуал>, ищу: <отношение1>;<отношение2>;...'  Команды: help, list, exit. > я arseniy\_a ищу внуки;родители;дяди;тети  внуки: нет данных  родители: dmitriy\_a, svetlana\_a  дяди: aleksandr\_a  тети: natalya\_s |

Пример 2 (Некорректный ввод):

|  |
| --- |
| > я ыдваод адв  Некорректный ввод. Пример: 'Я mike ищу тети кузены' или 'Я: dmitriy\_a, ищу: тети;кузены'  Введите строку: 'Я: <индивидуал>, ищу: <отношение1>;<отношение2>;...'  Команды: help, list, exit. |

# ***ЗАКЛЮЧЕНИЕ***

Реализована консольная система рекомендаций на Python поверх онтологии родственных связей. Ввод запросов идет через командную строку. Запросы выполняются через rdflib/SPARQL; используются факты из lab1\_full.rdf и вывод (родители, мать/отец, братья/сёстры, тёти/дяди, кузены, бабушки/дедушки, внуки). Добавлены команды help/list, предусмотрены базовые проверки ввода. Система удобна для пользователя.