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

- 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

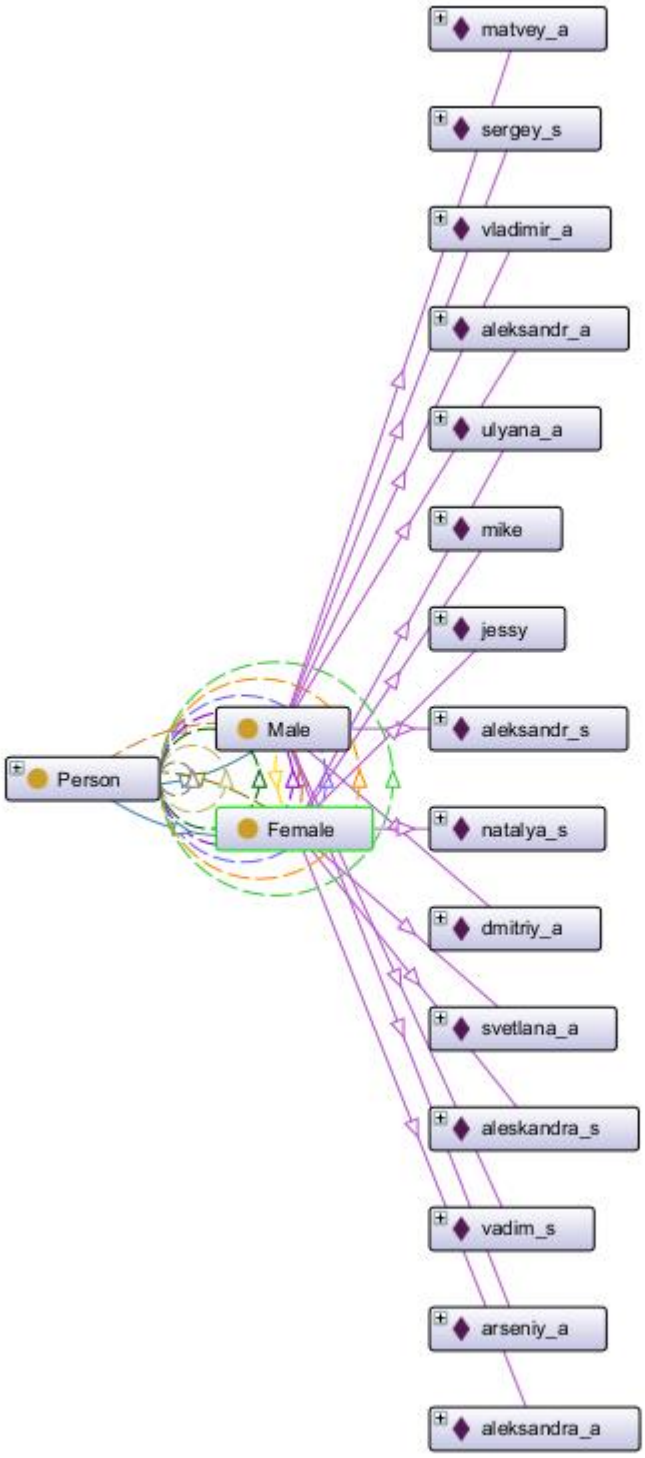

Рисунок 1 – Граф Protege.

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.