Изображение выглядит как эмблема, символ, герб, нашивка

Автоматически созданное описание

|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **"МИРЭА - Российский технологический университет"**  **РТУ МИРЭА** |

**Институт** Информационных Технологий

**Кафедра** Вычислительной Техники

**ПРАКТИЧЕСКАЯ РАБОТА №1**

**по дисциплине**

**«Системный анализ данных в системах поддержки принятия**

**решений»**

**Онтология**

Студент группы: ИКБО-15-22 \_\_Оганнисян Г.А.\_ *(Ф. И.О. студента)*

Преподаватель \_\_Железняк Л.М.\_\_

*(Ф.И.О. преподавателя)*

|  |  |  |
| --- | --- | --- |
|  |  |  |

Москва 2024

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc177917504)

[1 ОНТОЛОГИЯ 5](#_Toc177917505)

[1.1 Постановка задачи 5](#_Toc177917506)

[1.2 Описание предметной области 5](#_Toc177917507)

[1.3 Иерархия классов 5](#_Toc177917508)

[1.4 Реализация в Protégé 6](#_Toc177917509)

[1.5 Результаты работы программы 9](#_Toc177917510)

[ЗАКЛЮЧЕНИЕ 13](#_Toc177917511)

[СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ 14](#_Toc177917512)

[ПРИЛОЖЕНИЯ 15](#_Toc177917513)

ВВЕДЕНИЕ

Возникновение онтологий и их стремительное развитие связано с проявлением в нашей реальности следующих новых факторов:

* колоссальный рост объемов информации, предъявляемых для обработки (анализа, использования) специалистам самых различных областей деятельности;
* чрезвычайная зашумленность этих потоков (повторы, противоречивость, разноуровневость);
* острая необходимость в использовании одних и тех же знаний разными специалистами в разных целях;
* всеобщая интернетизация нашей жизни и острая необходимость в структуризации информации для её представления пользователям и более эффективного поиска;
* необходимость сокращения времени на поиск нужной информации и повышения качества информационных услуг в Интернете.

Появление онтологий стало ответом ряда наук, связанных с информационными технологиями и системами искусственного интеллекта на перечисленные проблемы. Именно они обеспечили возможность их перехода на новый качественный уровень обработки и поиска информации. Наиболее распространенными стали следующие определения.

Онтология – это точная спецификация концептуализации.

Онтология – это формальная точная спецификация совместно используемой концептуализации.

Онтологии – это базы знаний специального типа, которые могут читаться и пониматься, отчуждаться от разработчика и/или физически разделяться их пользователями.

Под концептуализацией понимается абстрактная модель явлений (процессов) в мире, составленная посредством определения существенных для описания данных явлений понятий, т.е. концептов. Точность подразумевает, что типы используемых понятий и ограничения на область применения данных понятий явно определены. Формальность означает, что онтология должна быть ориентирована на компьютерное представление, что исключает использование естественных языков в полной мере, в связи с их неоднозначностью и сложностью. Совместное использование отражает понятие того, что онтология описывает всеобщие знания, т.е. не персональные знания одного человека, а знания, принятые в группе, сообществе.

1 онтология

1.1 Постановка задачи

Выбрать предметную область по личному интересу и изучить её. Составить модель онтологии по предметной области. Реализовать модель созданной онтологии в программе Protégé. Написать программный продукт на выбранном языке программирования, реализующий построенную модель онтологии.

1.2 Описание предметной области

В качестве предметной области выбрана «Курсы по программированию». Актуальность выбранной предметной области заключается в том, что изучение программирования становится всё более популярным и востребованным навыком в современном мире. Онтология курсов по программированию — это мощный инструмент, который может значительно повысить эффективность обучения, улучшить качество информации и стимулировать развитие инновационных образовательных сервисов. Основными понятиями для рассматриваемой предметной области являются курсы, модули, преподаватели и студенты. Модель онтологии позволяет найти взаимоотношения между объектами, например, узнать, какие курсы преподаёт определённый преподаватель или какие модули включены в тот или иной курс. Яркий пример сервиса, использующего схожую онтологию – «Coursera». В этом сервисе онтология используется для организации контента и предоставления персонализированных рекомендаций.

1.3 Иерархия классов

В рассматриваемой онтологии предполагается иерархия классов, в которой родительским классом является непосредственно абстрактный класс «Образовательные курсы». Образовательные курсы включают в себя курсы, студентов и преподавателей. В свою очередь, курсы подразделяются на очно-заочные курсы и онлайн курсы. Курсы также содержат такие элементы, как занятия, которые, в свою очередь, включают практические задания и тесты.. Иерархия классов показана на рисунке 1.3.1.

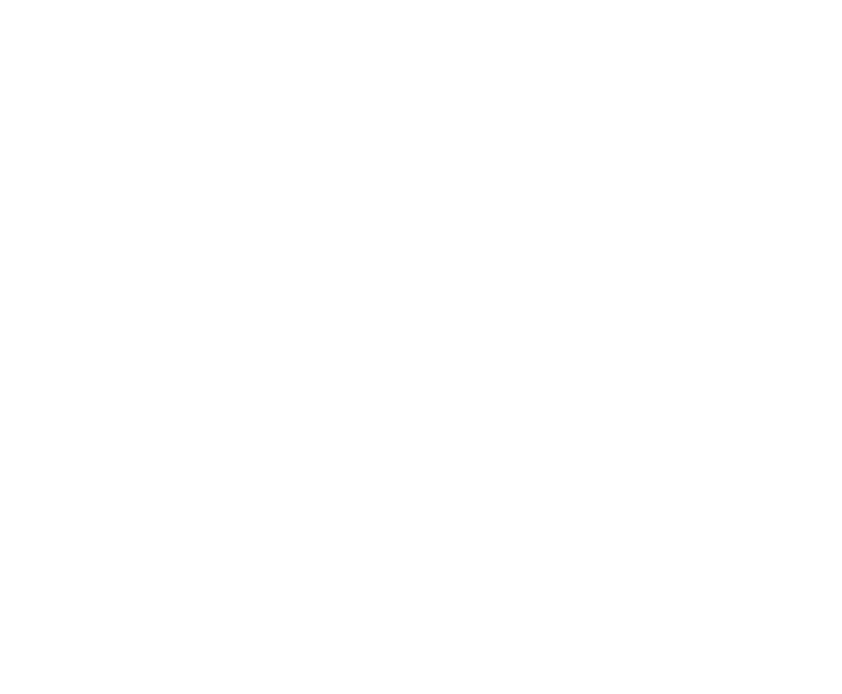


Рисунок 1.3.1 – Иерархия классов онтологии

1.4 Реализация в Protégé

Иерархия онтологии была перенесена в Protégé (Рисунок 1.4.1).

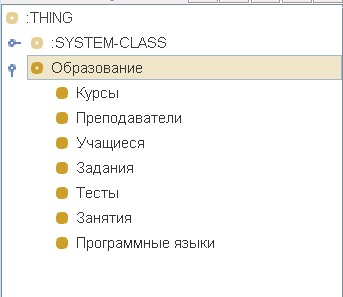


Рисунок 1.4.1 – Иерархия классов в Protégé

Класс «Образование» является абстрактным. Созданы слоты для класса «Программные языки» (Рисунок 1.4.2).

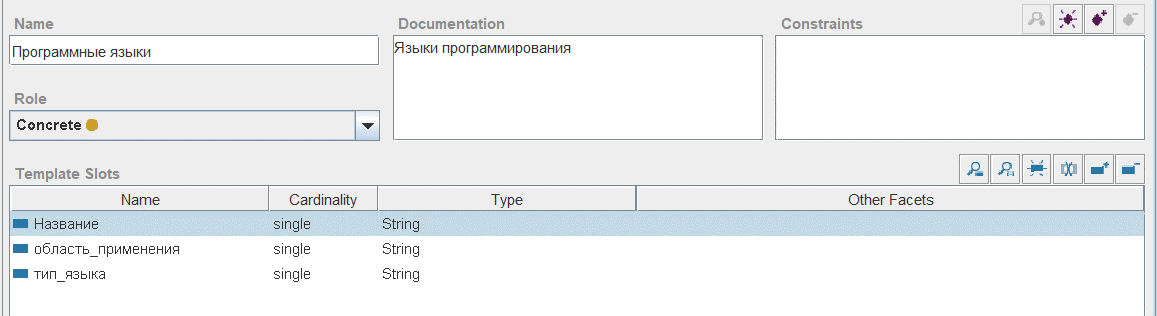


Рисунок 1.4.2 – Слоты для класса «Фильм»

Созданы слоты для класса «Задания» (Рисунок 1.4.3).

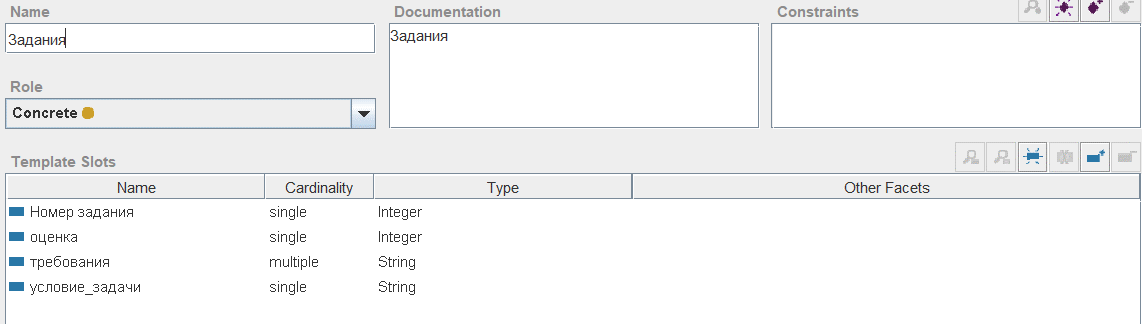


Рисунок 1.4.3 - Слоты для класса «Задания»

Созданы слоты для класса «Тесты» (Рисунок 1.4.4).

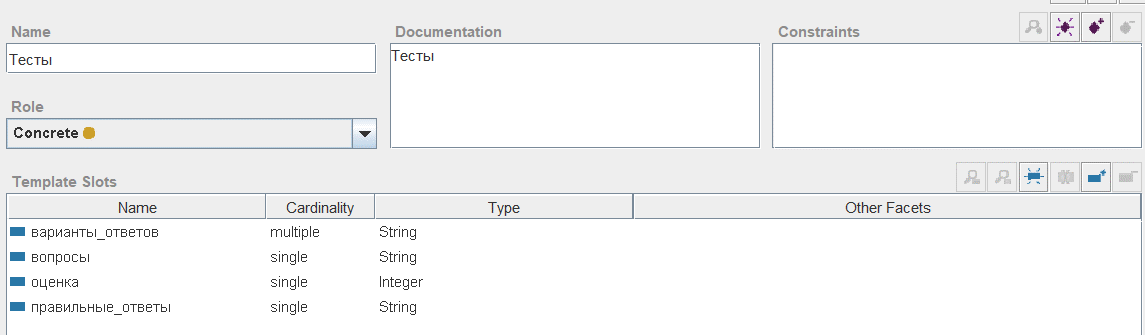


Рисунок 1.4.4 - Слоты для класса «Тесты»

Созданы слоты для класса «Занятия» (Рисунок 1.4.5).

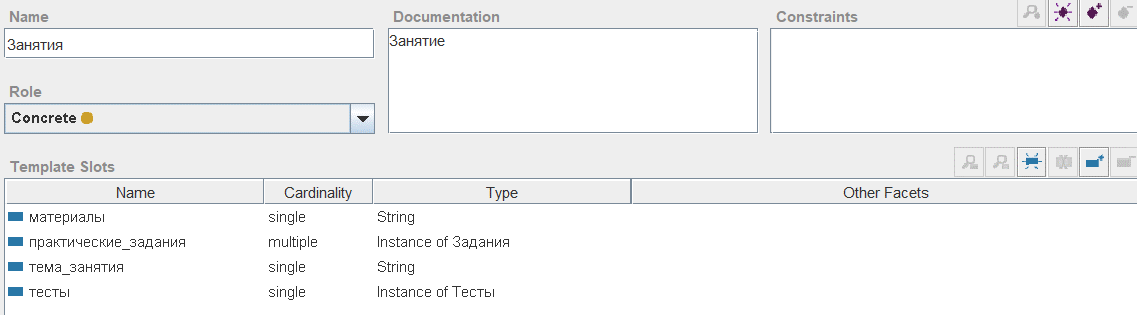


Рисунок 1.4.5 – Слоты для класса «Занятия»

Класс «Занятия» содержит слот «практические задания», который ссылается на класс «Задания». Так же содержит слот «тесты», который ссылается на класс «Тесты». Созданы слоты для класса «Преподаватель» (Рисунок 1.4.6).

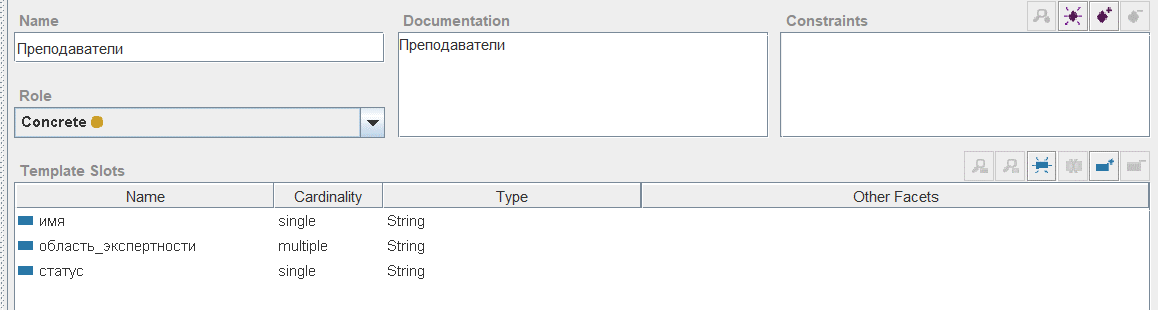
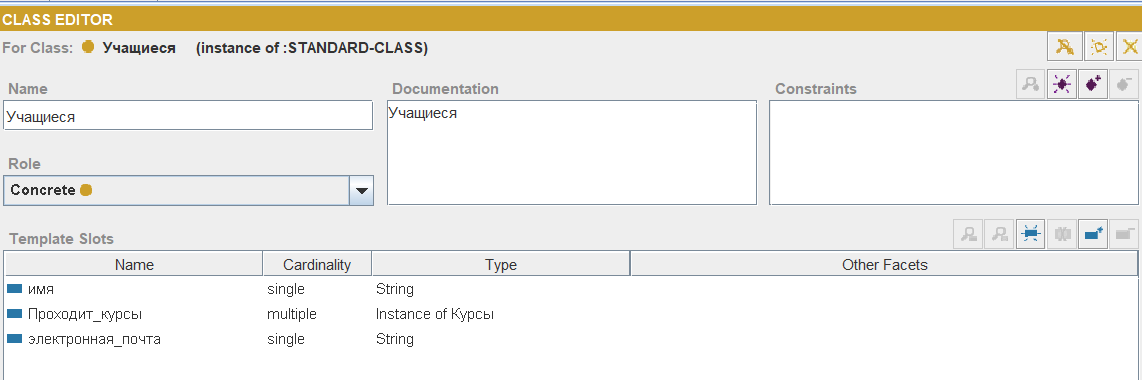


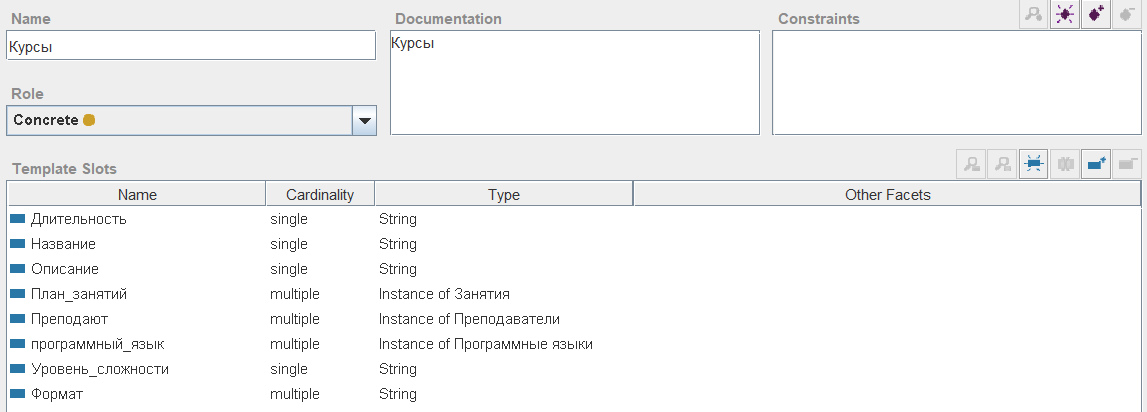
Рисунок 1.4.6 – Слоты для класса «Преподаватель»

Созданы слоты для класса «Учащиеся» (Рисунок 1.4.7).



**Рисунок 1.4.7 – Слоты для класса «Учащиеся»**

Класс «Учащиеся» содержит слот «Проходит\_курсы», который указывается на курсы, которые проходит учащийся. Созданы слоты для класса «Курсы» (Рисунок 1.4.8).



**Рисунок 1.4.8 – Слоты для класса «Курсы»**

Класс «Курсы» имеет слоты, которые ссылаются на классы «Занятия», «Преподаватели», «Программные языки».

Создано несколько объектов классов (Рисунок 1.4.9).

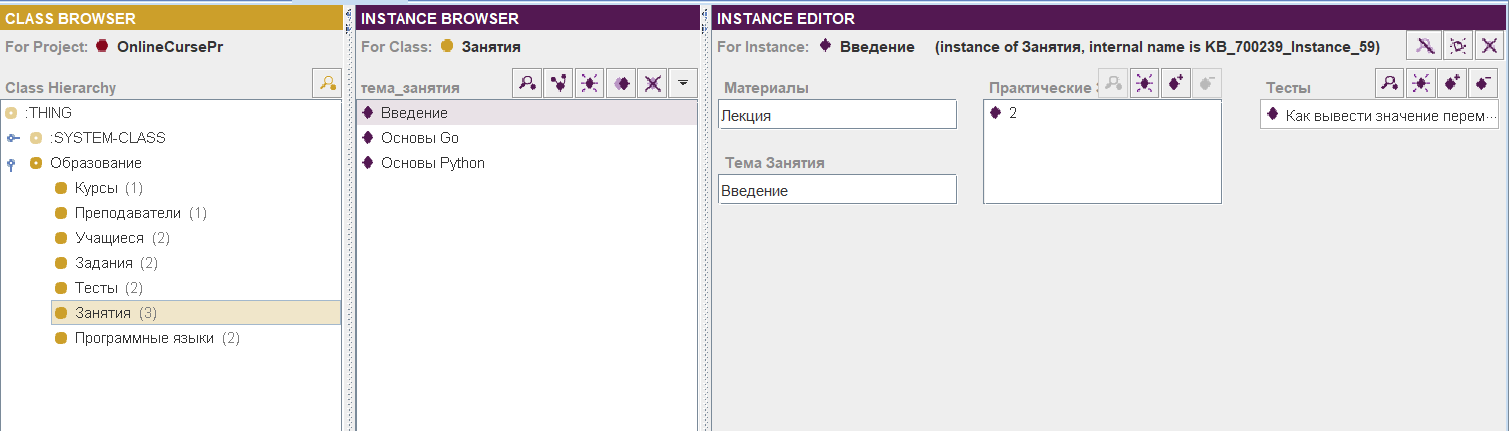


Рисунок 1.4.9 – Созданные объекты классов

Произведено несколько запросов (Рисунок 1.4.9 – 1.4.11).

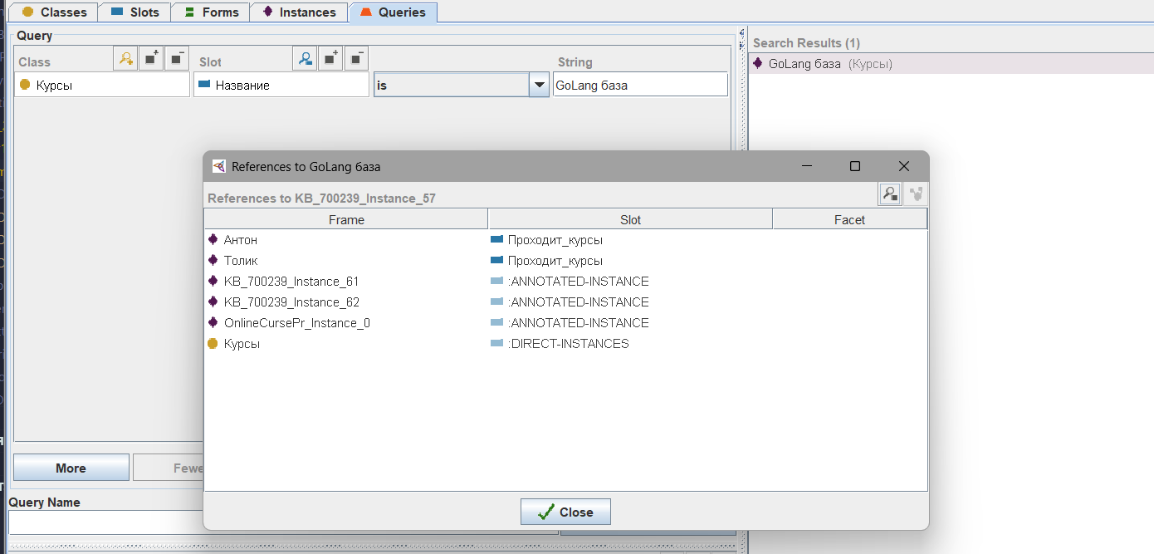


Рисунок 1.4.9 – Запрос от класса «Курсы»

1.5 Результаты работы программы

Разработан программный продукт, описывающий созданную модель онтологии. Для большей наглядности добавлены дополнительные объекты в каждый класс (Рисунок 1.5.1 – 1.5.2).

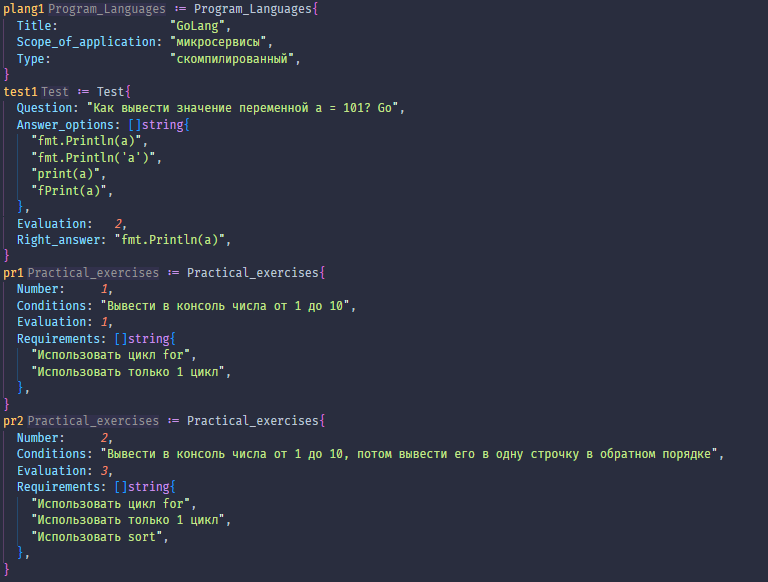


Рисунок 1.5.1 – Созданные объекты классов

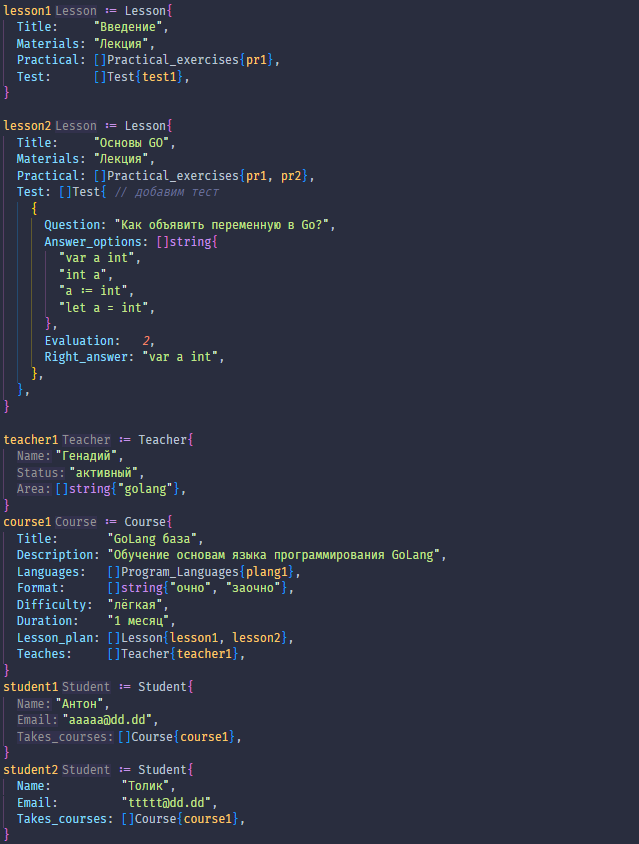


Рисунок 1.5.2 – Созданные объекты классов

Результаты выполнения запросов от каждого классы представлены на рисунке 1.5.2.

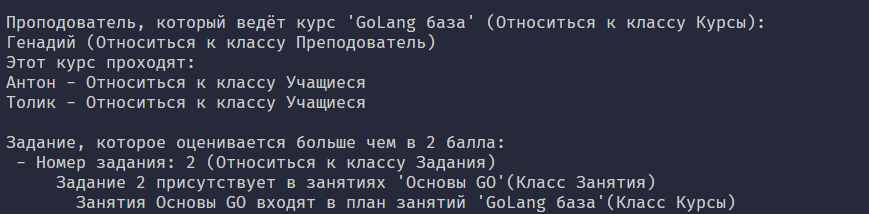


Рисунок 1.5.1 – Запросы

Заключение

Protégé — это мощное программное обеспечение для разработки и управления онтологиями.

К основным плюсам Protégé относятся:

1. Удобный графический интерфейс, который облегчает создание и редактирование онтологий, что делает его доступным для пользователей с разным уровнем опыта.

2. Protégé поддерживает множество форматов (OWL, RDF, XML и др.), что позволяет интегрировать онтологии с другими системами и инструментами.

3. Возможность использования плагинов для добавления новых функций и возможностей, что позволяет адаптировать программу под конкретные нужды пользователей.

4. Обширная документация, что упрощает поиск решений для возникающих проблем и обмен опытом.

Среди минусов Protégé можно выделить:

1. Несмотря на интуитивно понятный интерфейс, начинающие пользователи могут столкнуться с трудностями в понимании семантики и структуры онтологий.

2. При работе с большими онтологиями производительность может значительно снижаться, что требует оптимизации.

3. Основные инструменты визуализации могут быть недостаточно мощными для сложных или больших онтологий, что затрудняет анализ структуры.

СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ

1. Protege URL: <https://protegewiki.stanford.edu/wiki/ProtegeDocs> (Дата обращения: 16.09.2024).
2. Сорокин А. Б. Теория графов в машинном обучении [Электронный ресурс]: учебное пособие / А. Б. Сорокин, Р. Э. Семенов. — М.: РТУ МИРЭА, 2023.
3. Сорокин А. Б. Введение в роевой интеллект: теория, расчет и приложения [Электронный ресурс]: учебно-метод. пособие / А. Б. Сорокин. — М.: РТУ МИРЭА, 2019.
4. Тузовский А. Ф., Чириков С. В., Ямпольский В. З. Системы управления знаниями (методы и технологии). – Томск: Издательство НТЛ, 2020 – 260 c.

ПРИЛОЖЕНИЯ

Приложение А – Код реализации онтологии на языке GoLang.

Приложение А

*Листинг А.1. Реализация онтологии.*

package main

import "fmt"

type Course struct {

Title string

Description string

Languages []Program\_Languages // Привязка: курс содержит программные языки

Format []string // очный, заочный

Difficulty string

Duration string

Lesson\_plan []Lesson

Teaches []Teacher

}

type Program\_Languages struct {

Title string

Scope\_of\_application string // область применения

Type string

}

type Lesson struct {

Title string

Materials string

Practical []Practical\_exercises

Test []Test

}

type Practical\_exercises struct {

Number int

Conditions string

Evaluation int //оценка

Requirements []string //требования

}

type Test struct {

Question string

Answer\_options []string //варианты ответы

Evaluation int

Right\_answer string // правильный ответ

}

type Student struct {

Name string

Email string

Продолжение Листинга А.1.

*Takes\_courses []Course*

*}*

*type* Teacher *struct* {

  Name   string

  Status string

  Area   []string //*область экспертности*

}

*func* main() {

  plang1 := Program\_Languages{

    Title:                "GoLang",

    Scope\_of\_application: "микросервисы",

    Type:                 "скомпилированный",

  }

  test1 := Test{

    Question: "Как вывести значение переменной a = 101? Go",

    Answer\_options: []string{

      "fmt.Println(a)",

      "fmt.Println('a')",

      "print(a)",

      "fPrint(a)",

    },

    Evaluation:   *2*,

    Right\_answer: "fmt.Println(a)",

  }

  pr1 := Practical\_exercises{

    Number:     *1*,

    Conditions: "Вывести в консоль числа от 1 до 10",

    Evaluation: *1*,

    Requirements: []string{

      "Использовать цикл for",

      "Использовать только 1 цикл",

    },

  }

  pr2 := Practical\_exercises{

    Number:     *2*,

    Conditions: "Вывести в консоль числа от 1 до 10, потом вывести его в одну строчку в обратном порядке",

    Evaluation: *3*,

    Requirements: []string{

      "Использовать цикл for",

      "Использовать только 1 цикл",

      "Использовать sort",

    },

  }

  lesson1 := Lesson{

    Title:     "Введение",

    Materials: "Лекция",

    Practical: []Practical\_exercises{pr1},

    Test:      []Test{test1},

  }

Продолжение Листинга А.1.

  lesson2 := Lesson{

    Title:     "Основы GO",

    Materials: "Лекция",

    Practical: []Practical\_exercises{pr1, pr2},

    Test: []Test{ // добавим тест

      {

        Question: "Как объявить переменную в Go?",

        Answer\_options: []string{

          "var a int",

          "int a",

          "a := int",

          "let a = int",

        },

        Evaluation:   2,

        Right\_answer: "var a int",

      },

    },

  }

  teacher1 := Teacher{

    "Генадий",

    "активный",

    []string{"golang"},

  }

  course1 := Course{

    Title:       "GoLang база",

    Description: "Обучение основам языка программирования GoLang",

    Languages:   []Program\_Languages{plang1},

    Format:      []string{"очно", "заочно"},

    Difficulty:  "лёгкая",

    Duration:    "1 месяц",

    Lesson\_plan: []Lesson{lesson1, lesson2},

    Teaches:     []Teacher{teacher1},

  }

  student1 := Student{

    "Антон",

    "aaaaa@dd.dd",

    []Course{course1},

  }

Продолжение Листинга А.1.

  lesson2 := Lesson{

    Title:     "Основы GO",

    Materials: "Лекция",

    Practical: []Practical\_exercises{pr1, pr2},

    Test: []Test{ // добавим тест

      {

        Question: "Как объявить переменную в Go?",

        Answer\_options: []string{

          "var a int",

          "int a",

          "a := int",

          "let a = int",

        },

        Evaluation:   2,

        Right\_answer: "var a int",

      },

    },

  }

  teacher1 := Teacher{

    "Генадий",

    "активный",

    []string{"golang"},

  }

  course1 := Course{

    Title:       "GoLang база",

    Description: "Обучение основам языка программирования GoLang",

    Languages:   []Program\_Languages{plang1},

    Format:      []string{"очно", "заочно"},

    Difficulty:  "лёгкая",

    Duration:    "1 месяц",

    Lesson\_plan: []Lesson{lesson1, lesson2},

    Teaches:     []Teacher{teacher1},

  }

  student1 := Student{

    "Антон",

    "aaaaa@dd.dd",

    []Course{course1},

  }

Продолжение Листинга А.1.

  fmt.Println()

  fmt.Println("Задание, которое оценивается больше чем в 2 балла: ")

  for \_, i := range pracs {

    if i.Evaluation > 2 {

      fmt.Printf(" - Номер задания: %v (Относиться к классу Задания) \n", i.Number)

      for \_, les := range lessons {

        for \_, prac := range les.Practical {

          if prac.Number == i.Number {

            fmt.Printf("     Задание %v присутствует в занятиях '%s'(Класс Занятия)", i.Number, les.Title)

            for \_, c := range courses {

              for \_, l := range c.Lesson\_plan {

                if l.Title == les.Title {

                  fmt.Printf("\n       Занятия %s входят в план занятий '%s'(Класс Курсы)", l.Title, c.Title)

                }

              }

            }

          }

        }

      }

    }

  }

}