

Липецкий государственный технический университет

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

ЛАБОРАТОРНАЯ РАБОТА №1

по дисциплине «Системы искусственного интеллекта»

Разработка экспертной системы

Студент

Быкова А.В.

Группа АС-19-1

Руководитель

Кургасов В.В.

Липецк 2022 г.

Цель работы

Получение навыков проектирования и разработки экспертной системы
на всех этапах ее создания

Задание кафедры

Отработать этапы разработки экспертной системы для решения задачи (проблемы) выбора. Осуществить программную реализацию экспертной системы на любом языке программирования.

Разрабатываемая экспертная система относится к классу поверхностных демонстрационных (учебных) систем. Поверхностные ЭС представляют знания в виде правил (условие – действие).

Создание экспертной системы в рамках данного занятия проекта позволяет изучить и реализовать все этапы разработки ЭС:

1. идентификация,
2. концептуализация,
3. формализация,
4. выполнение,
5. тестирование,
6. опытная эксплуатация.

Особенностью работы является то, что выполняет функционал всех членов коллектива разработчиков ЭС – эксперта, инженера по знаниям, программиста и пользователя.

Вариант 5.

Разработка экспертной системы «Выбор домашнего животного».

Ход работы

1 Сформулированная проблема (задача)

Прежде всего, поставим задачу, для решения которой будет разрабатываться экспертная система. Подходящей задачей, при решении которой можно использовать обратную цепочку рассуждений, может быть задача, вытекающая из следующей ситуации: человек пришел в зоомагазин и хочет определиться с выбором домашнего животного.

На первый взгляд задача не очень сложная, но на решение влияет много факторов, таких как размер жилплощади, количество членов семьи и др.

Поскольку в задаче надо выбрать один из нескольких возможных вариантов, для её решения можно воспользоваться обратной цепочкой рассуждений. В действительности ответ уже существует. Продавцу необходимо задать посетителю такие вопросы, ответы на которые дадут возможность сделать правильный выбор.

Итак, задача поставлена. Теперь нужно наглядно ее представить. Для описания подобных задач обычно используются диаграммы, которые называются деревьями решений. Деревья решений дают необходимую наглядность и позволяют проследить ход рассуждений.

2 Дерево решений для выбранной проблемы (задачи)

Дерево решений – это ориентированный граф, вершинами которого являются условия и выводы, а дугами результат выполнения (проверки) условий.

Диаграммы называются деревьями решений потому, что, подобно настоящему дереву, имеют ветви. Ветви деревьев решений заканчиваются логическими выводами. Для рассматриваемого примера вывод заключается в том, предложит ли директор должность поступающему на работу, и если да, то какую. Многие задачи сложны, и их непросто представить (или для их решения не собираются использовать экспертную систему). Дерево решений помогает преодолеть эти трудности.

В приложении А показано дерево решений для задачи с выбором домашнего животного. Видно, что диаграмма состоит из кружков и прямоугольников, которые называются вершинами. Каждой вершине присваивается номер. На вершины можно ссылаться по этим номерам. Номера вершин можно выбрать произвольно, т. к. они и служат только для удобства идентификации. Линии, соединяющие вершины, называются дугами. Совокупность вершин и дуг называется ветвями.

Кружки, содержащие вопросы, называются вершинами условий. Прямоугольники содержат логические выводы. Линии (стрелки) показывают направление диаграммы. Подписи возле линий — это ответы на вопрос, содержащийся в вершине условия. Вершины условий могут иметь сразу по несколько выходящих линий (стрелок), связывающих их с другими вершинами. В этом случае каждая линия (стрелка) должна быть четко определена. Не может быть две линии, у которых подписи одинаковые, например, подпись «Да». Выбор выходящей из вершины ветви определяется проверкой условия (вопроса), содержащегося в вершине. Можно сказать, что вершины содержат переменные, а пути - это условия, в соответствии с которыми переменным присваиваются значения.

После того как для проблемной области сформулированы правила, эти условия становятся условными частями (ЕСЛИ) правила. Прямоугольники содержат выводы

В дереве решений могут быть локальные (частные) выводы или цели. Локальный вывод - это также составляющая условной части (ЕСЛИ) правила.

3 Таблица переменных

Для каждой вершины логического вывода определяется путь и записывается правило. Длинную фразу можно заменить переменной, принимающей значения “да” или “нет”. В действительности все вершины содержат переменные, имеющие уникальные имена. Список имен переменных, текст, который они заменяют, и номера вершин пути сводят в таблицу, таблица 1. Использование переменных вместо полного текста упрощает формирование и запись правил.

Таблица 1 - Таблица имён переменных

| Имя переменной | Условия |
|----------------|----------------------------------------------------------------|
| FLAT | У Вас большая квартира? |
| TIME | У Вас много свободного времени? |
| EXOTIC | Вы любите экзотику? |
| RODENT | Вам нравятся грызуны? |
| WALK | У Вас есть возможность много гулять? |
| KOMAND | Вы часто уезжаете в командировки? |
| CHILD | У Вас есть дети? |
| NOISE | Вы спокойно относитесь к шуму? |
| COLOR | Любите яркие цвета? |
| SHOW | Вы планируете участвовать в выставках? |
| HAIR | Какой тип шерсти Вы предпочитаете? |
| ACTIVE | Вы хотели бы активное животное? |
| WORK | В какое время вы обычно работаете? |
| SPEAK | Вы бы хотели иметь приятного собеседника? |
| FAMILY | У Вас большая семья? |
| BIG | Вам нравятся крупные животные? |
| DECOR | Вам нравятся декоративные собаки? |
| CLEAN | Вы готовы часто делать уборку? |
| ALLERGY | Есть ли у Вас склонность к аллергии? |
| PRICE | Вы можете себе позволить дорогое животное? |
| DORMANT | Как вы относитесь к тому, что животное будет впадать в спячку? |
| PROTECT | Животное должно уметь защитить Вас? |

| | |
|-----------|-------------------------------------------------------------------------------------------|
| CHARACTER | Какое качество Вам более важно в животном? |
| STYLE | Для Вас животное это атрибут стиля? |
| RENT | Где вы будете оставлять животное в свое отсутствие? |
| LIKE | Вам нравится шерсть? |
| SMALL | У вас есть маленькие родственники? |
| UNUSUAL | Вы любите необычных животных? |
| TIMEWORK | В какое время вы обычно работаете? днём (да)/ ночью (нет) |
| PLACE | Где вы будете оставлять животное в свое отсутствие? гостинница (да)/родственники (нет) |

4 База знаний (правила)

В таблице 2 приведены все правила для дерева решений. Правила соответствуют всем путям, ведущим к возможным целям дерева решений. Правила желательно пронумеровать. Совокупность правил является формализованными знаниями и представляет собой базу знаний.

Таблица 2 - База знаний

| № правила | Правило |
|-----------|--------------------------------------------------------------------------------------------------------|
| 10 | ЕСЛИ FLAT=нет И EXOTIC=нет И RODENT=да И CHILD=да И ACTIVE=да И CLEAN=да ТО PET=мышь |
| 20 | ЕСЛИ FLAT=нет И EXOTIC=нет И RODENT=да И CHILD=да И ACTIVE=да И CLEAN=нет ТО PET=песчанка |
| 30 | ЕСЛИ FLAT=нет И EXOTIC=нет И RODENT=да И CHILD=да И ACTIVE=нет ТО PET=крыса |
| 40 | ЕСЛИ FLAT=нет И EXOTIC=нет И RODENT=да И CHILD=нет И WORK=днем И ALLERGY=да ТО PET=шиншилла |
| 50 | ЕСЛИ FLAT=нет И EXOTIC=нет И RODENT=да И CHILD=нет И WORK=днем И ALLERGY=нет ТО PET=кролик |
| 60 | ЕСЛИ FLAT=нет И EXOTIC=нет И RODENT=да И CHILD=нет И WORK=ночью ТО PET=хомяк |
| 70 | ЕСЛИ FLAT=нет И EXOTIC=нет И RODENT=нет И NOISE= да И SPEAK=нет ТО PET=канарейка |
| 80 | ЕСЛИ FLAT=нет И EXOTIC=нет И RODENT=нет И NOISE= да И SPEAK=да И PRICE=да ТО PET=попугай жако |
| 90 | ЕСЛИ FLAT=нет И EXOTIC=нет И RODENT=нет И NOISE= да И SPEAK=да И PRICE=нет ТО PET=волнистый попугай |
| 100 | ЕСЛИ FLAT=нет И EXOTIC=нет И RODENT=нет И NOISE= нет И DORMANT=да ТО PET=черепаха |
| 110 | ЕСЛИ FLAT=нет И EXOTIC=нет И RODENT=нет И NOISE= нет И DORMANT=нет ТО PET=рыбки |
| 120 | ЕСЛИ FLAT=нет И EXOTIC=да И WALK=да ТО PET=хорек |
| 130 | ЕСЛИ FLAT=нет И EXOTIC=да И WALK=нет И COLOR=нет ТО PET=сцинк |
| 140 | ЕСЛИ FLAT=нет И EXOTIC=да И WALK=нет И COLOR=да ТО PET=хамелеон |
| 150 | ЕСЛИ FLAT=да И TIME=да И EXOTIC=нет И SHOW=нет И FAMILY=нет И PROTECT=да ТО PET=овчарка |
| 160 | ЕСЛИ FLAT=да И TIME=да И EXOTIC=нет И SHOW=нет И FAMILY=нет И PROTECT=нет ТО PET=болонка французская |
| 170 | ЕСЛИ FLAT=да И TIME=да И EXOTIC=нет И SHOW=нет И FAMILY=да И CHARACTER=преданность ТО PET=собака хаски |

| | |
|-----|---------------------------------------------------------------------------------------------------------|
| 180 | ЕСЛИ FLAT=да И TIME=да И EXOTIC=нет И SHOW=нет И FAMILY=да И CHARACTER=ласка ТО PET=персидская кошка |
| 190 | ЕСЛИ FLAT=да И TIME=да И EXOTIC=нет И SHOW=нет И FAMILY=да И CHARACTER=внимание ТО PET= сибирская кошка |
| 200 | ЕСЛИ FLAT=да И TIME=да И EXOTIC=нет И SHOW=да ТО PET=скоттиш-фолд |
| 210 | ЕСЛИ FLAT=да И TIME=да И EXOTIC=да И HAIR=длинный и BIG=нет ТО PET=гималайская кошка |
| 220 | ЕСЛИ FLAT=да И TIME=да И EXOTIC=да И HAIR=длинный и BIG=да ТО PET=кошка мейн кун |
| 230 | ЕСЛИ FLAT=да И TIME=да И EXOTIC=да И HAIR=короткий PET=сфинкс |
| 240 | ЕСЛИ FLAT=да И TIME=да И EXOTIC=да И HAIR=короткий ТО PET=абиссинская кошка |
| 250 | ЕСЛИ FLAT=да И TIME=нет И KOMAND=нет И CHILD=нет И DECORATE=да И STYLE=да ТО PET=той терьер |
| 260 | ЕСЛИ FLAT=да И TIME=нет И KOMAND=нет И CHILD=нет И DECORATE=да И STYLE=нет ТО PET=шпиц |
| 270 | ЕСЛИ FLAT=да И TIME=нет И KOMAND=нет И CHILD=нет И DECORATE=нет ТО PET=колли |
| 280 | ЕСЛИ FLAT=да И TIME=нет И KOMAND=нет И CHILD=да ТО PET=ньюфаундленд |
| 290 | ЕСЛИ FLAT=да И TIME=нет И KOMAND=нет И RENT=гостиница ТО PET=такса |
| 300 | ЕСЛИ FLAT=да И TIME=нет И KOMAND=нет И RENT=родственники ТО PET=морская свинка |

Таким образом, дерево решений позволяет просто и наглядно представить ход рассуждений эксперта при решении задачи и формировать правила для базы знаний, а без базы знаний экспертную систему не построить.

5 Таблицы структур данных

При создании экспертной системы для упрощения ответа на вопросы и решения поставленной задачи в систему включается ряд полезных таблиц или структур данных. Структуры данных нужны для работы с базой знаний. После определения метода решения выбранного круга задач можно приступить к разработке системы.

Список логических выводов - это структура данных, содержащая упорядоченный список возможных логических выводов.

Список состоит из номера правила, логического вывода, связанного с этим правилом, и условий, которые формируют вывод. На каждое правило базы знаний в списке приходится одна запись. Список логических выводов используется исключительно для поиска вывода по номеру правила. Когда условия части ЕСЛИ истинны, вызывается часть ТО правила, ей присваивается значение.

Первоначально предполагается, что переменным значения еще не присвоены и признак инициализации null.

Например, переменная FLAT находится в первой строке списка, но ей еще не присвоено значение, значит, обратиться к условной части правила нельзя. Для того чтобы присвоить значение переменной FLAT, нужно узнать длину волос. Ответ и будет значением переменной и признак NI заменится на признак I., а переменной FLAT присваивается конкретное значение.

В любом случае после присвоения переменной FLAT значения, для нее в соответствующей строке списка переменных признак инициализации NI заменяется на I. С этого момента, в каком бы правиле в условной части не встретилась переменная FLAT, она будет считаться проинициализированной, имеющей какое-либо значение и ее можно использовать для работы с любыми правилами.

Таким образом, до того, как правило включается в работу, все переменные, входящие в его условную часть, должны быть

проинициализированы. Определить, присвоено ли переменной условия значение, можно, просмотрев список переменных.

Список переменных условия – это перечень всех переменных для всех условных частей всех правил базы знаний.

Условная часть правила (ЕСЛИ) может содержать несколько переменных. Под каждое правило выделяется одинаковое число позиций в списке переменных условия. Минимальное число позиций равно числу переменных условия самого «длинного» правила.

Теперь рассмотрим, каким образом три описанные структуры данных соотносятся с мыслительной деятельностью человека в процессе обратной цепочки рассуждений. Прежде всего, человек просматривает все возможные пути, способные привести к решению задачи. Затем он выделяет условия, составляющие эти пути.

6 Блок схема алгоритма программной реализации

Блок схема алгоритма реализованной программы изображен на рисунке



Рисунок 1 – Блок схема

7 Программная реализация ЭС

```
using System;
using System.Collections.Generic;
using System.Data.SQLite;
using ExpertSystem.Models;
using System.Runtime.Serialization.Formatters.Binary;
using System.IO;
namespace ExpertSystem.Services
{
    public class Database
    {
        private readonly SQLiteConnection _connection;

        public Database()
        {
            _connection = new SQLiteConnection("Data Source=expert_system.db;
Version=3");
            _connection.Open();
            InitializeDatabase();
        }

        public void CreateData(string sql)
        {
            var cmd = new SQLiteCommand(sql, _connection);
            cmd.ExecuteNonQuery();
        }

        public DetailQuestion GetDetailQuestion(int id)
        {
            var yesAnswer = GetAnswerAfterQuestion(id, true);
            var noAnswer = GetAnswerAfterQuestion(id, false);
            var yesQuestion = GetNextQuestion(id, true);
            var noQuestion = GetNextQuestion(id, false);
        }
    }
}
```

```

return new DetailQuestion
{
    NoAnswer = noAnswer,
    YesQuestion = yesQuestion,
    YesAnswer = yesAnswer,
    NoQuestions = noQuestion
};
}

```

```

public void DeleteAnswer(int id)
{
    var sql = "DELETE FROM answer WHERE answer_id = @answer_id";
    var cmd = new SQLiteCommand(sql, _connection);
    cmd.Parameters.AddWithValue("answer_id", id);
    cmd.ExecuteNonQuery();
}

```

```

public Answer UpdateAnswer(Answer answer)
{
    var sql = "UPDATE answer SET answer_text = @answer_text, question_id = @question_id, answer_true = @answer_true WHERE answer_id = @answer_id";
    var cmd = new SQLiteCommand(sql, _connection);
    cmd.Parameters.AddWithValue("answer_text", answer.Text);
    if (answer.QuestionID == 0)
    {
        cmd.Parameters.AddWithValue("question_id", DBNull.Value);
    }
    else
    {
        cmd.Parameters.AddWithValue("question_id", answer.QuestionID);
    }
    cmd.Parameters.AddWithValue("answer_true", answer.IsTrue);
    if (answer.ID == 0)

```

```

    {
        cmd.Parameters.AddWithValue("answer_id", DBNull.Value);
    }
    else
    {
        cmd.Parameters.AddWithValue("answer_id", answer.ID);
    }
    cmd.ExecuteNonQuery();
    return answer;
}

```

```

public Answer CreateAnswer(string text, byte[] image)
{
    var sql = "INSERT INTO answer (answer_text, answer_image) VALUES (@text,
@image) RETURNING answer_id, answer_text, question_id, answer_true, answer_image";
    var cmd = _connection.CreateCommand();
    cmd.CommandText = sql;
    cmd.Parameters.AddWithValue("text", text);
    if (image == null)
    {
        cmd.Parameters.AddWithValue("image", DBNull.Value);
    }
    else
    {
        cmd.Parameters.AddWithValue("image", Convert.ToBase64String(image));
    }
    var response = cmd.ExecuteReader();
    if (response.HasRows)
    {
        while (response.Read())
        {
            var answer = new Answer
            {

```



```

        ID = Convert.ToInt32(response["answer_id"]),
        isTrue = response["answer_true"] != DBNull.Value ?
Convert.ToBoolean(response["answer_true"]) : false,
        Text = Convert.ToString(response["answer_text"]),
        QuestionID = response["question_id"] == DBNull.Value ? 0 :
Convert.ToInt32(response["question_id"]),
        Image = ObjectToByteArray(response["answer_image"] == DBNull.Value ?
null : response["answer_image"])
    };
    return answer;
}
}
return null;
}

```

```

public Answer GetAnswerAfterQuestion(int questionId, bool isTrue)
{
    var sql = "SELECT answer_id, answer_text, question_id, answer_true,
answer_image FROM answer WHERE question_id = @question_id AND answer_true =
@answer_true";

    var cmd = _connection.CreateCommand();
    cmd.CommandText = sql;
    cmd.Parameters.AddWithValue("question_id", questionId);
    cmd.Parameters.AddWithValue("answer_true", isTrue);
    var response = cmd.ExecuteReader();
    if (response.HasRows)
    {
        while (response.Read())
        {
            var answer = new Answer
            {
                ID = Convert.ToInt32(response["answer_id"]),

```

```

        isTrue = response["answer_true"] != DBNull.Value ?
Convert.ToBoolean(response["answer_true"]) : false,
        Text = Convert.ToString(response["answer_text"]),
        QuestionID = response["question_id"] == DBNull.Value ? 0 :
Convert.ToInt32(response["question_id"]),
        Image = ObjectToByteArray(response["answer_image"] == DBNull.Value ?
null : response["answer_image"]);
    };
    return answer;
}
}
return null;
}

```

```

public Question GetStartQuestion()
{
    var sql = "SELECT question_id, question_text, parent_id, question_true FROM
question WHERE parent_id IS NULL";
    var cmd = _connection.CreateCommand();
    cmd.CommandText = sql;
    var response = cmd.ExecuteReader();
    if (response.HasRows)
    {
        while (response.Read())
        {
            var question = new Question
            {
                ID = Convert.ToInt32(response["question_id"]),
                isTrue = response["question_true"] == DBNull.Value ? false :
Convert.ToBoolean(response["question_true"]),
                Text = Convert.ToString(response["question_text"]),
                ParentID = response["parent_id"] == DBNull.Value ? 0 :
Convert.ToInt32(response["parent_id"])
            };
            return question;
        }
    }
    return null;
}

```

```

        };
        return question;
    }
}
return null;
}

public Question GetNextQuestion(int id, bool isTrue)
{
    var sql = "SELECT question_id, question_text, parent_id, question_true FROM
question WHERE parent_id = @parent_id AND question_true = @question_true";
    var cmd = _connection.CreateCommand();
    cmd.CommandText = sql;
    cmd.Parameters.AddWithValue("parent_id", id);
    cmd.Parameters.AddWithValue("question_true", isTrue);
    var response = cmd.ExecuteReader();
    if (response.HasRows)
    {
        while (response.Read())
        {
            var question = new Question
            {
                ID = Convert.ToInt32(response["question_id"]),
                isTrue = response["question_true"] == DBNull.Value ? false :
Convert.ToBoolean(response["question_true"]),
                Text = Convert.ToString(response["question_text"]),
                ParentID = response["parent_id"] == DBNull.Value ? 0 :
Convert.ToInt32(response["parent_id"])
            };
            return question;
        }
    }
    return null;
}

```

```
}
```

```
public List<Question> ResetQuestions()
```

```
{
```

```
    var sql = "UPDATE question SET parent_id = NULL, question_true = NULL;";
```

```
    var cmd = _connection.CreateCommand();
```

```
    cmd.CommandText = sql;
```

```
    cmd.ExecuteNonQuery();
```

```
    return GetQuestions();
```

```
}
```

```
public void DeleteQuestion(int id)
```

```
{
```

```
    var sql = "DELETE FROM question WHERE question_id = @question_id";
```

```
    var cmd = _connection.CreateCommand();
```

```
    cmd.CommandText = sql;
```

```
    cmd.Parameters.AddWithValue("question_id", id);
```

```
    cmd.ExecuteNonQuery();
```

```
}
```

```
public Question UpdateQuestion(Question question)
```

```
{
```

```
    var sql = "UPDATE question SET question_text = @text, parent_id = @parent_id,  
question_true = @question_true WHERE question_id = @question_id;";
```

```
    var cmd = _connection.CreateCommand();
```

```
    cmd.CommandText = sql;
```

```
    cmd.Parameters.AddWithValue("text", question.Text);
```

```
    if (question.ParentID == 0)
```

```
    {
```

```
        cmd.Parameters.AddWithValue("parent_id", DBNull.Value);
```

```
    }
```

```
    else
```

```
    {
```

```

        cmd.Parameters.AddWithValue("parent_id", question.ParentID);
    }
    cmd.Parameters.AddWithValue("question_true", question.isTrue);
    if (question.ID == 0)
    {
        cmd.Parameters.AddWithValue("question_id", DBNull.Value);
    }
    else
    {
        cmd.Parameters.AddWithValue("question_id", question.ID);
    }
    cmd.ExecuteNonQuery();
    return question;
}

public Question CreateQuestion(string text)
{
    var sql = "INSERT INTO question (question_text) VALUES (@text) RETURNING
question_id, question_text, parent_id, question_true;";
    var cmd = _connection.CreateCommand();
    cmd.CommandText = sql;
    cmd.Parameters.AddWithValue("text", text);
    var response = cmd.ExecuteReader();
    if (response.HasRows)
    {
        while (response.Read())
        {
            var question = new Question
            {
                ID = Convert.ToInt32(response["question_id"]),
                isTrue = response["question_true"] == DBNull.Value ? false :
Convert.ToBoolean(response["question_true"]),
                Text = Convert.ToString(response["question_text"]),
            }
        }
    }
}

```

```

        ParentID = response["parent_id"] == DBNull.Value ? 0 :
Convert.ToInt32(response["parent_id"])
    };
    return question;
}
}
return null;
}

public List<Answer> GetAnswers()
{
    var answerList = new List<Answer>();
    var answerSql = @"SELECT answer_id, answer_text, question_id, answer_true,
answer_image
        FROM answer
        ORDER BY answer_id;";
    var answerCmd = new SQLiteCommand(answerSql, _connection);
    var response = answerCmd.ExecuteReader();
    if (response.HasRows)
    {
        while (response.Read())
        {
            var answer = new Answer
            {
                ID = Convert.ToInt32(response["answer_id"]),
                isTrue = response["answer_true"] != DBNull.Value ?
Convert.ToBoolean(response["answer_true"]) : false,
                Text = Convert.ToString(response["answer_text"]),
                QuestionID = response["question_id"] == DBNull.Value ? 0 :
Convert.ToInt32(response["question_id"]),
                Image = ObjectToByteArray(response["answer_image"] == DBNull.Value ?
null : response["answer_image"])
            };

```

```

        answerList.Add(answer);
    }
}
return answerList;
}

public List<Question> GetQuestions()
{
    var questionList = new List<Question>();
    var questionSql = @"SELECT question_id, question_text, parent_id, question_true
                        FROM question
                        ORDER BY question_id;";
    var questionCmd = new SQLiteCommand(questionSql, _connection);
    var response = questionCmd.ExecuteReader();
    if (response.HasRows)
    {
        while (response.Read())
        {
            var question = new Question
            {
                ID = Convert.ToInt32(response["question_id"]),
                isTrue = response["question_true"] == DBNull.Value ? false :
Convert.ToBoolean(response["question_true"]),
                Text = Convert.ToString(response["question_text"]),
                ParentID = response["parent_id"] == DBNull.Value ? 0 :
Convert.ToInt32(response["parent_id"])
            };
            questionList.Add(question);
        }
    }
    return questionList;
}

```

```

private byte[] ObjectToByteArray(object obj)
{
    if (obj == null)
        return null;
    try
    {
        return Convert.FromBase64String(obj.ToString());
    }
    catch (Exception)
    {
        BinaryFormatter bf = new BinaryFormatter();
        using (MemoryStream ms = new MemoryStream())
        {
            bf.Serialize(ms, obj);
            return ms.ToArray();
        }
    }
}

```

```

private void InitializeDatabase()
{
    var questionSql = @"CREATE TABLE IF NOT EXISTS question (
        question_id INTEGER PRIMARY KEY,
        question_text TEXT NOT NULL,
        parent_id INTEGER NULL,
        question_true BOOLEAN NULL
    ); ";

    var answerSql = @"CREATE TABLE IF NOT EXISTS answer (
        answer_id INTEGER PRIMARY KEY,
        answer_text TEXT NOT NULL,
        question_id INTEGER NULL,
        answer_true BOOLEAN NULL,
        answer_image TEXT NULL,
    ); ";
}

```



```

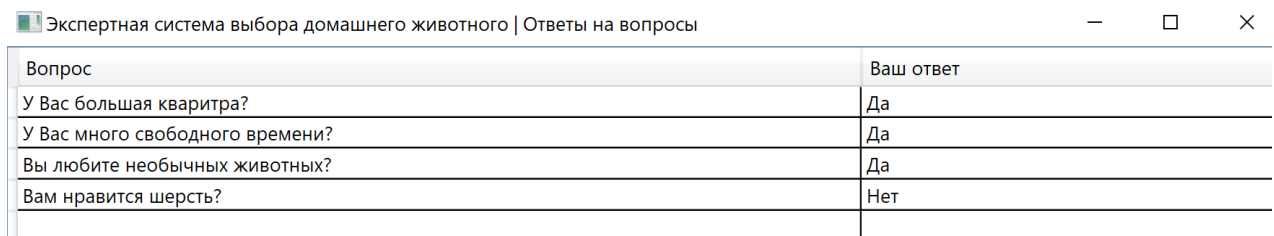
        FOREIGN KEY (question_id) REFERENCES question(question_id) ON
        DELETE SET NULL

    );";

    var questionCommand = new SQLiteCommand(questionSql, _connection);
    questionCommand.ExecuteNonQuery();

    var answerCommand = new SQLiteCommand(answerSql, _connection);
    answerCommand.ExecuteNonQuery();
}
}
}

```



| Вопрос | Ваш ответ |
|---------------------------------|-----------|
| У Вас большая квартира? | Да |
| У Вас много свободного времени? | Да |
| Вы любите необычных животных? | Да |
| Вам нравится шерсть? | Нет |
| | |

Рисунок 2 – История ответов

Вывод

В ходе выполнения лабораторной работы получила навыки проектирования и разработки экспертной системы на всех этапах ее создания.

Приложение А Дерево решений

