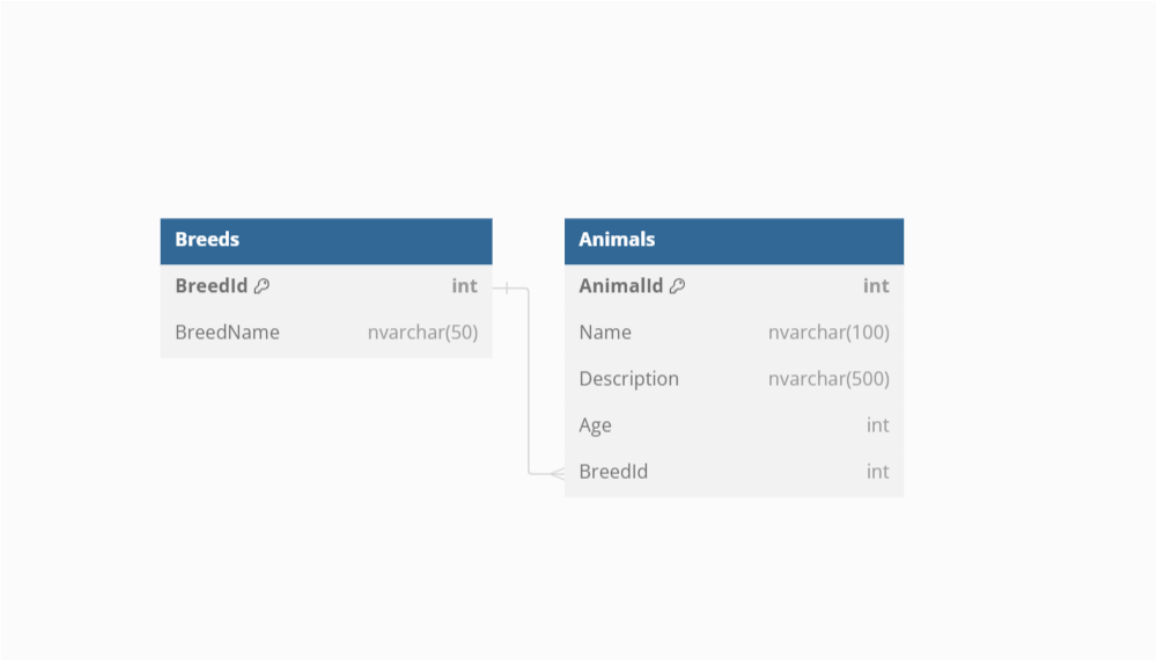
Курсов проект по Програмиране - Задача №6 на Марин Мирев, клас 11а, №13

1. Описание на БД

- **1. Описание на БД**



- **Таблица: Animals**
- **Номер (AnimalId)**: Цяло число, първичен ключ
- **Име на животното (Name)**: Текст, до 100 символа
- **Описание (Description)**: Текст, до 500 символа
- **Възраст (Age)**: Цяло число
- **Порода (BreedId)**: Цяло число, външен ключ към таблица Breeds

```
public class Animal
{
   public int Id { get; set; }
   public string Name { get; set; }
   public string Description { get; set; }
   public int Age { get; set; }
   public int BreedId { get; set; }
   public Breed Breed { get; set; }
}
```

- **Таблица: Breeds**
- **Номер на вида продукт (BreedId)**: Цяло число, първичен ключ
- **Име на типа продукт (BreedName)**: Текст, до 50 символа

```
public class Breed
{
   public int Id { get; set; }
   public string Name { get; set; }
   public ICollection<Animal> Animals { get; set; }
}
```

- **2. Функционално описание**

  Екран за добавяне на нови животни

  - Като за начало проверяваме дали има въведени данни
  - След като мине проверката създаваме нов обект от тип **Animal** и заместваме новите данни с тези от полетата за въвеждане и използваме контролерът ни да създаде новия запис и да запази данните в таблицата

    - Полета за въвеждане на име, описание, възраст и порода.
    - Падащ списък за избор на порода от предварително въведени породи.

```
private void btnAdd_Click(object sender, EventArgs e)
{
   if (string.IsNullOrEmpty(txtBoxName.Text) &&
string.IsNullOrEmpty(txtBoxAge.Text))
   {
      MessageBox.Show("Въведи данни!");
      return;
   }
   Animal animal = new Animal();
   animal.Age = int.Parse(txtBoxId.Text);
   animal.Name = txtBoxName.Text;
   animal.Description = txtBoxDescription.Text;
   animal.BreedId = (int)cmbBoxBreeds.SelectedValue;
   animalController.Create(animal);
   MessageBox.Show("Записът е успешно добавен!");
   LoadComboBox();
}
```

- **Екран за редактиране на съществуващи животни**

- Като за начало проверяваме дали е въведено Id а след това дали то е валидно
- След това създаваме и заменяме данните на намерения запис и запазваме промените

```
private void btnEdit_Click(object sender, EventArgs e)
    {
      int foundId = 0;
      if (string.IsNullOrEmpty(txtBoxId.Text) || !txtBoxId.Text.All(char.IsDigit))
      {
        MessageBox.Show("Въведете Id за редактиране!");
        txtBoxId.Focus();
        return;
      }
      else
      {
        foundId = int.Parse(txtBoxId.Text);
      }
      if (string.IsNullOrEmpty(txtBoxName.Text))
      {
        Animal foundAnimal = animalController.Get(foundId);
        if (foundAnimal == null)
        {
          MessageBox.Show("Няма такъв запис въведи ново Id");
          txtBoxId.Focus();
          return;
        }
        LoadAnimal(foundAnimal);
      }
      else
      {
        Animal newAnimal = new Animal
        {
          Name = txtBoxName.Text,
          Age = int.Parse(txtBoxAge.Text),
          BreedId = (int)cmbBoxBreeds.SelectedValue
        };
        animalController.Update(foundId, newAnimal);
        MessageBox.Show($"Успешно е редактиран записът с Id {foundId}");
        LoadComboBox();
}
```

Екран за изтриване на животни

- Започваме отново с проверка дали е въведено Id
- След което дали то е валидно ако е така от нас е нужно потвърждение
- Защото изтриването в базите данни е опасно!

- Премахва записа след нашето потвърждение и запазва промените.

```
private void btnDelete_Click(object sender, EventArgs e)
    {
      int foundId = 0;
      if (string.IsNullOrEmpty(txtBoxId.Text) ||
!txtBoxId.Text.All(char.IsDigit))
      {
        MessageBox.Show("Въведете Id за редактиране!");
        txtBoxId.Focus();
        return;
      }
      else
      {
        foundId = int.Parse(txtBoxId.Text);
      }
      if (string.IsNullOrEmpty(txtBoxName.Text))
      {
        Animal foundAnimal = animalController.Get(foundId);
        if (foundAnimal == null)
        {
          MessageBox.Show("Няма такъв запис въведи ново Id");
          txtBoxId.Focus();
          return;
        }
        LoadAnimal(foundAnimal);
      }
      DialogResult result = MessageBox.Show($"Наистина ли искате
да изтриете запис No {foundId}?", "PROMPT",
MessageBoxButtons.YesNo, MessageBoxIcon.Question);
      if (result == DialogResult.Yes)
      {
        animalController.Delete(foundId);
        MessageBox.Show($"Успешно изтрихте запис с Id {foundId}");
      }
      LoadComboBox();
    }
```

- **3. Анализ на задачата**

- За връзка с базата данни е използвана технологията ORM /Code First/ в MS SQL Server.
- **Подход за избор на контроли и интерфейс:**
- За въвеждане и редактиране на данни са използвани текстови полета (TextBox).
- За избор на порода е използван падащ списък (ComboBox).

- За визуализация на списък с всички животни е използвана таблица (ListBox).
  Реализирана е MVP структура на приложението за избрания интерфейс използваме WindowsFormsApplication .NetFramework 6.1
- **Използване на Миграции**
- **Context скрипт**

  Използваме контекста за реализиране на миграции
  DbSet е ключова дума за назоваване на Таблица в база от данни
  Съответно контекста ни позволява използването на таблиците на по преден план в проекта и разширяването му

```
public class AnimalsContext:DbContext
{
  public DbSet<Animal> Animals { get; set; }
  public DbSet<Breed> Breeds { get; set; }
}
```

- **“Бизнес логиката”**
  Започваме с логиката за **Animal** класа
  Правим си обект от контекст класа за използването на таблците

```
public class AnimalLogic
{
  private AnimalsContext animalsContext = new AnimalsContext();
}
```

  По нататък прилагаме CRUD (Create,Read,Update,Delete)Като се стремим да имаме 2 Read операции за извличане данните от таблиците по специфичен критерий и по-лесно четим код

```
public Animal Get(int id)
{
  Animal foundAnimal = animalsContext.Animals.Find(id);
  if (foundAnimal != null)
  {
    animalsContext.Entry(foundAnimal).Reference(n =>
n.Breed).Load();
  }
  return foundAnimal;
}
public List<Animal> GetAll()
{
  return animalsContext.Animals.Include("Breed").ToList();
}
```

След това имаме Create логиката и тя се състои от Добавяне на нов запис в таблицата от контекс класа

```
public void Create(Animal animal)
{
   animalsContext.Animals.Add(animal);
   animalsContext.SaveChanges();
}
```

След това идва и най-опасната логика Delete логиката тя изтрива запис по даден критерий в случая Id но първо намира записа

```
public void Delete(int id)
{
   Animal foundAnimal = animalsContext.Animals.Find(id);
   animalsContext.Animals.Remove(foundAnimal);
   animalsContext.SaveChanges();
}
```

Идва и Update Логиката която е най-сложната от всички и изисква въвеждането на Id и заменянето на данните от записа интересното е че тя приема 2 параметъра данните на новия запис и Id на старото

```
public void Update(int id,Animal animal)
{
   Animal foundAnimal = animalsContext.Animals.Find(id);
   if(foundAnimal == null)
   {
      return;
   }
   foundAnimal.Name = animal.Name;
   foundAnimal.Age = animal.Age;
   foundAnimal.BreedId = animal.BreedId;
   animalsContext.SaveChanges();
}
```

И така приключва бизнес логиката за Главната таблица а именно **Animal.**

Следващата логика е за **Breed** таблицата тя реализира само Read операциите в нашия случай но може и да се използват и всички CRUD операции ако редактираме записите в нея
Започваме отново със създаването на обект от контекст класа

```
public class BreedLogic
{
  private AnimalsContext animalsContext = new AnimalsContext();
}
```

И имаме отново 2 Read операции както следват

```
  public List<Breed> GetAllBreeds()
  {
    return animalsContext.Breeds.ToList();
  }
  public string GetBreedById(int id)
  {
    return animalsContext.Breeds.Find(id).Name;
  }
```

- **Interface логика или в MVP View логика**

В този проект тя се случва чрез вградения интерфейс на MS WindowsFormsApplication
Като за начало се правят обекти за всеки един от нашите контролери

```
  public AnimalLogic animalController = new AnimalLogic();
  public BreedLogic breedController = new BreedLogic();
```

Да започнем с метод за зареждане на изходните данни в обозначените за това контроли (Listbox/ComboBox)
Тук използваме DataSource свойството им за да черпят информация от нашите контролери и да използват декларираните там методи за четене

```
  private void LoadComboBox()
  {
    cmbBoxBreeds.DataSource = breedController.GetAllBreeds();
    cmbBoxBreeds.ValueMember = "Id";
    cmbBoxBreeds.DisplayMember = "Name";
```

```
            cmbBoxDogSelect.ValueMember = "Id";
            cmbBoxDogSelect.DisplayMember = "Name";
            PopulateListBox();
        }
```

PopulateListBox() метода е за зареждане на предварително подготвената от нас информация в него и за визуализацията им.

```
        private void PopulateListBox()
        {
            listBoxOutput.DataSource = animalController.GetAll();
        }
```

След това идва и метод за зареждане на запис и неговите данни взима 1 параметър и този метод най често се използва при подмяната на избора ни в ComboBox или ListBox

```
        private void LoadAnimal(Animal selectedAnimal)
        {
            txtBoxName.Text = selectedAnimal.Name;
            txtBoxDescription.Text = selectedAnimal.Description;
            txtBoxAge.Text = selectedAnimal.Age.ToString();
            cmbBoxBreeds.SelectedIndex = selectedAnimal.BreedId - 1;
            txtBoxId.Text = selectedAnimal.Id.ToString();
        }
```

Примери:

```
    private void cmbBoxDogSelect_SelectedIndexChanged(object sender,
    EventArgs e)
        {
```

```
            LoadAnimal(cmbBoxDogSelect.SelectedItem as Animal);
        }
```

```
    private void listBoxOutput_SelectedIndexChanged(object sender,
    EventArgs e)
        {
            LoadAnimal(listBoxOutput.SelectedItem as Animal);
        }
```