

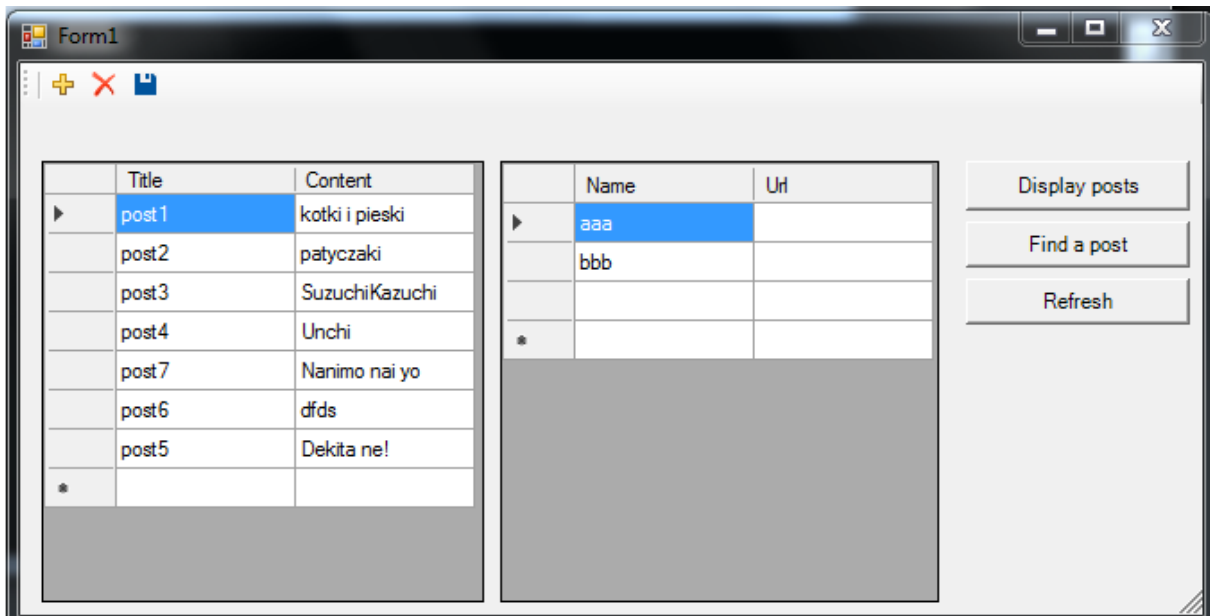
# **Sprawozdanie z zadania domowego**

Bazy danych, laboratorium 2

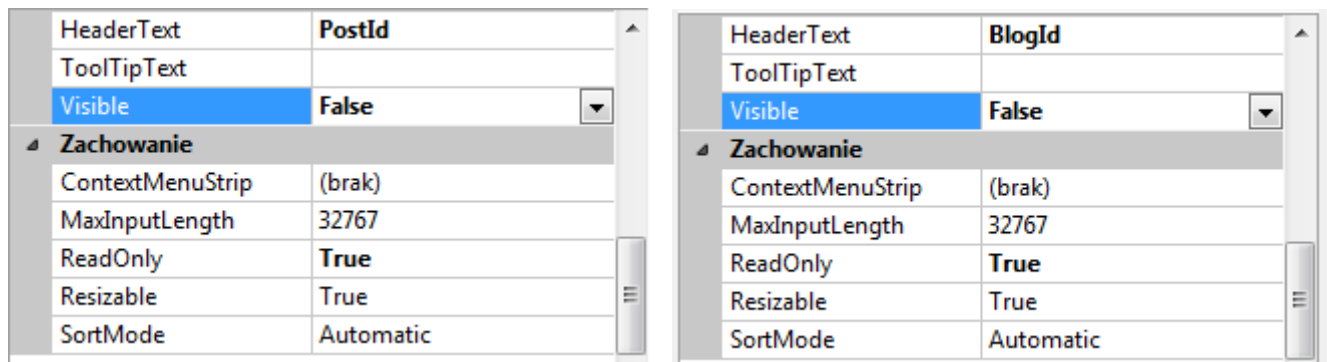
Agata Bogacz

Zadanie polegało na zaimplementowaniu lub rozszerzeniu bloga wg własnego pomysłu w Entity Framework. Zdecydowałam się rozszerzyć funkcjonalność rozpoczętego na zajęciach projektu jako aplikację WindowsFormową.

Rozpoczęłam od uporządkowania kolumn w głównym oknie:




Ustawiłam pola BlogId i PostId na niewidoczne:

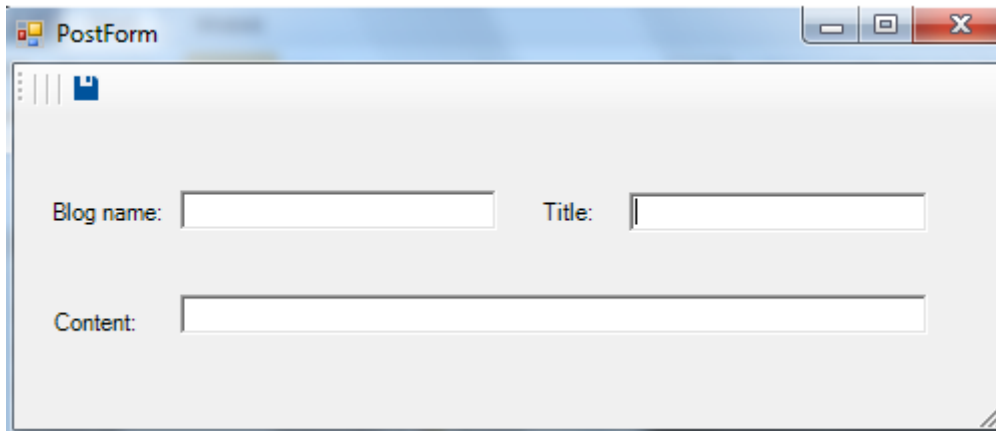


Usunęłam też zbędne przyciski. Następnie dodałam funkcjonalność do ikonki 

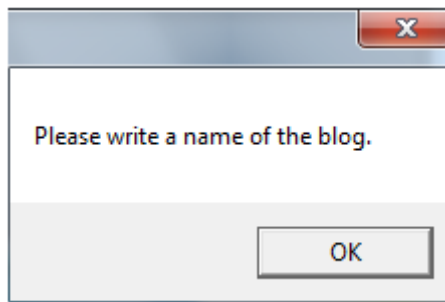
Po jej naciśnięciu pojawia się okienko dodawania postów do bazy, zgodnie z kodem:

```
private void bindingNavigatorAddNewPost_Click(object sender, EventArgs e)
{
    PostForm post_form = new PostForm();
    post_form.ShowDialog();
}
```

Po naciśnięciu  sprawdzane jest czy istnieje blog, do którego chcemy dodać post. Jeśli jakieś pola są puste lub danego bloga nie ma w bazie – wyświetlane są odpowiednie komunikaty.

A screenshot of a Windows application window titled "PostForm". The window has a standard Windows XP-style title bar with minimize, maximize, and close buttons. Inside the window, there is a toolbar with a save icon. Below the toolbar, there are three text input fields: "Blog name:" followed by a text box, "Title:" followed by a text box, and "Content:" followed by a larger text box.

Przykładowy komunikat:



Kod definiujący zachowanie się aplikacji przy próbie zapisu zmian:

```
private void postsBindingNavigatorSaveItem_Click(object sender, EventArgs e)
{
    string blogName = nameTextBox.Text;
    string postTitle = titleTextBox.Text;
    string content = contentTextBox.Text;
    int findBlogId;

    if(blogName == "")
    {
        MessageBox.Show("Enter the name of the blog where you would like to put a post.");
    }
    if (postTitle == "")
    {
        MessageBox.Show("Enter a title of a new post.");
    }
    if (content == "")
    {
        MessageBox.Show("Please write something as a content of the post.");
    }
}
```

```

    }
    using (var bloggingContext = new BloggingContext())
    {
        if (blogName == null)
        {
            MessageBox.Show("Enter a name of the blog.");
        }
        else
        {
            findBlogId = (from b in bloggingContext.Blogs
                          where b.Name == blogName
                          select b.BlogId).FirstOrDefault();

            if (findBlogId == 0)
            {
                MessageBox.Show("There is no such blog.");
            }
            else
            {
                var postToAdd = new Post();
                postToAdd.Title = postTitle;
                postToAdd.BlogId = findBlogId;
                postToAdd.Content = content;
                bloggingContext.Posts.Add(postToAdd);
                bloggingContext.SaveChanges();
                MessageBox.Show("Dodano post.");
            }
        }
    }

    this.Validate();
    this.postsBindingSource.EndEdit();

    this.tableAdapterManager.UpdateAll(this._CodeFirstNewDatabaseSample_BloggingContext
    DataSet);

    }
}

```

Sprawdzanie, czy dany blog istnieje odbywa się z wykorzystaniem **query syntax**. **Deferred execution** sprawia, że zapytanie odbywa się dopiero przy *if (findBlogId == 0)*.

Główne okno aplikacji po zmianach można odświeżyć za pomocą przycisku

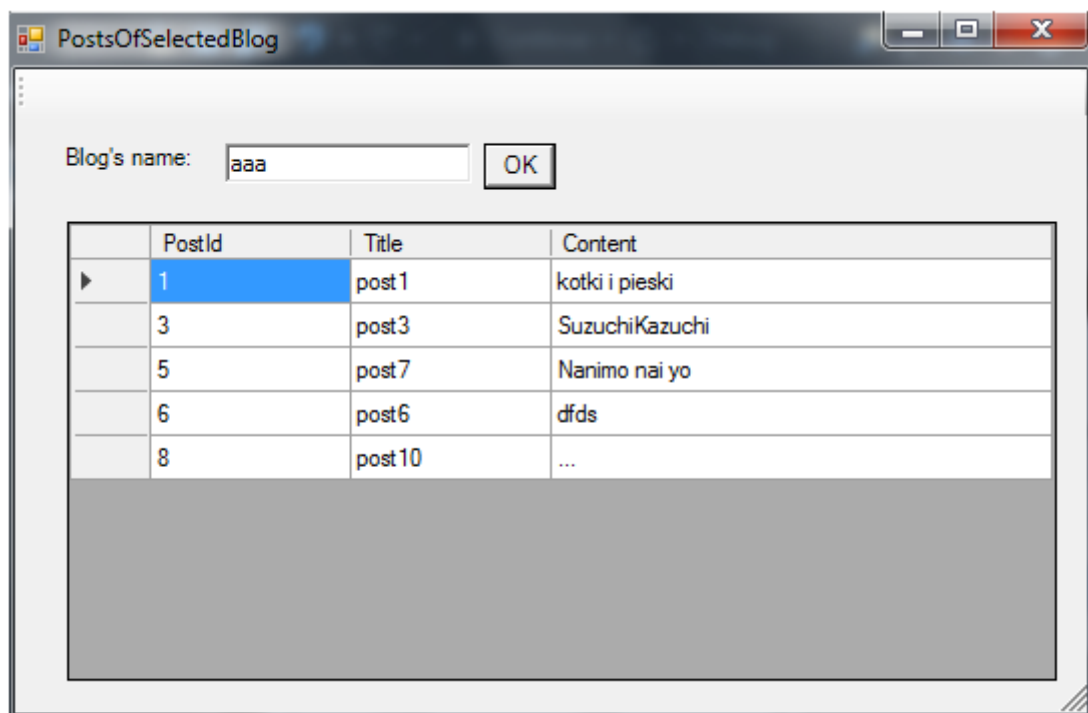
Refresh

zdefiniowanego następująco:

```
private void refbutton_Click(object sender, EventArgs e)
{
    this.usersTableAdapter.Fill(this._CodeFirstNewDatabaseSample_BloggingContextDataSet.Users);
    this.postsTableAdapter.Fill(this._CodeFirstNewDatabaseSample_BloggingContextDataSet.Posts);
    this.blogsTableAdapter.Fill(this._CodeFirstNewDatabaseSample_BloggingContextDataSet.Blogs);
    bContext = new BloggingContext();
    bContext.Blogs.Load();
    blogsBindingSource.DataSource = bContext.Blogs.Local.ToBindingList();

    bContext.Posts.Load();
    postsBindingSource.DataSource = bContext.Posts.Local.ToBindingList();
}
```

Kolejną dodaną przeze mnie funkcjonalnością było stworzenie okienka do wyświetlania postów o konkretnym tytule.



```
private void button1_Click(object sender, EventArgs e)
{
    string blogName = nameTextBox.Text;
    int findBlogId;
    if (blogName == "")
    {
        MessageBox.Show("Please write a name of the blog.");
    }
    else
```

```

{
    using (var bloggingContext = new BloggingContext())
    {
        //Wykorzystanie method syntax i Eager loading - metoda Include
        findBlogId = (bloggingContext.Blogs
            .Include("Posts")
            .Where(b => b.Name == blogName)
            .FirstOrDefault<Blog>()).BlogId;

        //brak takiego blogu
        if (findBlogId == 0)
        {
            MessageBox.Show("There is no such blog.");
        }
        else //jesli jest to wyswietlamy posty
        {
            List<Post> postList = new List<Post>();

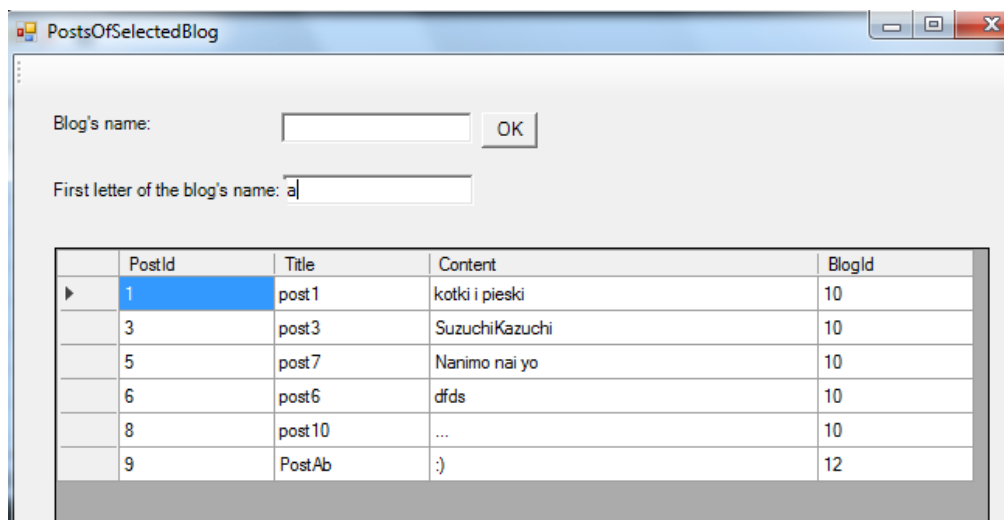
            postList = (from blogs in bloggingContext.Blogs
                where blogs.BlogId == findBlogId
                select blogs.Posts).
                FirstOrDefault();

            dataGridPosts.DataSource = postList;
            dataGridPosts.Update();
        }
    }
}

```

Wykorzystałam tu **method syntax** oraz metodę `Include()`, dzięki czemu skorzystałam również z **eager loadingu**.

Następnie dodałam do tego możliwość wyszukiwania przedrostku tytułu.



Blog's name:  OK

First letter of the blog's name: a

PostId	Title	Content	BlogId
1	post1	kotki i pieski	10
3	post3	SuzuchiKazuchi	10
5	post7	Nanimo nai yo	10
6	post6	dřds	10
8	post10	...	10
9	PostAb	:)	12

Zastosowałam tutaj konkatencję list, aby wyświetlić posty ze wszystkich blogów o tym samym przedrostku. Po wprowadzeniu jakiegoś znaku do pola tekstowego wykonywany jest następujący kod:

```
private void textBox1_TextChanged(object sender, EventArgs e)
{
    string letter = textBox1.Text;

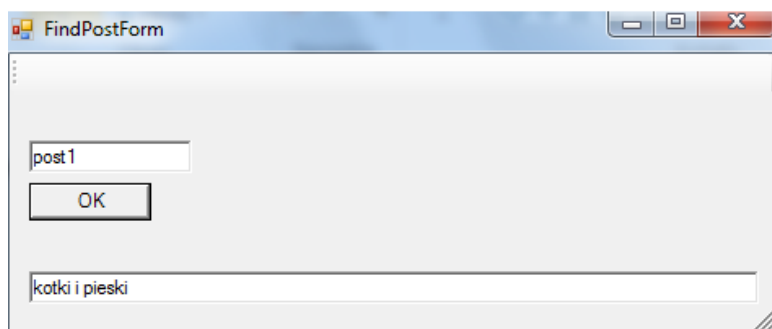
    //navigation property - wyciągnięcie z blogów list postów
    using (var bloggingContext = new BloggingContext())
    {

        IQueryable<Blog> desiredBlogs = bloggingContext.Blogs
            .Where(b => b.Name.StartsWith(letter));

        if (desiredBlogs == null)
        {
            MessageBox.Show("There are no such blogs that start on the letter you typed.");
        }
        else
        {
            List<Post> postList = new List<Post>();
            List<Post> tmp;
            foreach (Blog b in desiredBlogs)
            {
                tmp = postList.Union(b.Posts).ToList();
                postList = tmp;
            }
            dataGridPosts.DataSource = postList;
            dataGridPosts.Update();
        }
    }
}
```

Do wyciągnięcia konkretnych postów użyłam **Navigation Property** – najpierw wyodrębniłam interesujące mnie blogi, a z nich dopiero odpowiednie posty.

Kolejną dodaną przeze mnie funkcjonalnością było wyszukiwanie posta w bazie.



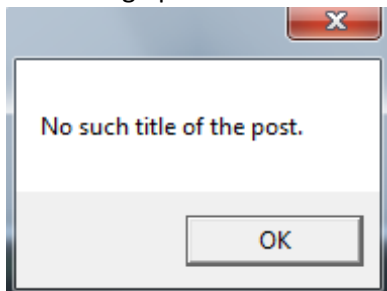
Funkcjonalność po naciśnięciu ok:

```
private void button1_Click(object sender, EventArgs e)
{
    string postTitle = PostToFindBox.Text;
    if (String.IsNullOrEmpty(postTitle) == true)
    {
        MessageBox.Show("Enter a title of the post.");
    }
    else
    {
        using (var bloggingContext = new BloggingContext())
        {
            Post post = (from posts in bloggingContext.Posts
                        where posts.Title == postTitle
                        select posts).FirstOrDefault<Post>();

            if (post != null)
                contentBox.Text = post.Content;

            else
                MessageBox.Show("No such title of the post.");
        }
    }
}
```

Jeśli danego postu nie znaleziono w bazie to wyświetlany jest następujący komunikat:



**Fluent Api** użyłam w celu wyznaczenia maksymalnej długości posta:

```
protected override void OnModelCreating(DbModelBuilder modelBuilder)
{
    modelBuilder.Entity<Post>().Property(t => t.Content).HasMaxLength(100);
}
```

Link do kodu źródłowego okienek:

<https://drive.google.com/file/d/0BybHhgjJ4KzSaDhHZjZNS2NLMDQ/view?usp=sharing>

Cała aplikacja:

<https://drive.google.com/file/d/0BybHhgjJ4KzSTGtfOFVKQjQ0QUE/view>