Dokumentacja
ProperTax

Filip Borecki

1. Opis działania projektu

Aplikacja służy do inwentarzu nieruchomości oraz obliczania podatku za ich powierzchnie. Dzięki temu można przeglądać aktualnie posiadane nieruchomości oraz te które zostały już sprzedane.

2. Technologia

Aplikacja internetowa została napisana w frameworku asp.net 8.0
Architektura: Model-View-Controller
Baza danych: Microsoft sql server
Użyto Entity framework do mapowania obiektowo-relacyjnego.

Korzysta z rozwiązań:
Microsoft.AspNetCore.Identity.EntityFrameworkCore
Microsoft.AspNetCore.Identity.UI
Microsoft.EntityFrameworkCore.SqlServer
Microsoft.EntityFrameworkCore.Tools

3. Pierwsze uruchomienie projektu

Otworzyć przy pomocy Visual Studio.
W konsoli nuget wpisać update-database
Skompilować i uruchomić program.

4. Opis struktury projektu

Aplikacja jest w architekturze MVC. W folderze Controllers są 4 kontrolery: HomeController, NieruchomosciController, StawkiPodatkowController, SumyPowierzchniController. W models 4 modele: ErrorViewModel, Nieruchomosc, StawkaPodatku, SumaPowierzchni. W views Home, Nieruchomosci, Shared, StawkiPodatkow, SumyPowierzchni, _ViewImports, _ViewStart. W Data są migracje oraz ApplicationDbContext. W pliku appsettings.json jest łańcuch połączenia do bazy danych. Oraz Program.cs, który tworzy aplikacje, role użytkowników, użytkownika admina oraz ustawia DateTime na format yyyy-mm-dd.

Aby móc cokolwiek zrobić trzeba dodać nowy rok w “Stawki podatków”. Tam dodajemy rok oraz stawki podatków na różne kategorie powierzchni. Podczas dodawania w “Sumy powierzchni” również doda się 12 miesięcy tego roku. Tam obliczane są sumy powierzchni wszystkich nieruchomości posiadanych w danym miesiącu. Następnie można zacząć dodawać nieruchomości w “Nieruchomości”.

Można dodać tylko wtedy kiedy rok daty zakupu został dodany w “Stawki podatków”. Aby dodać wymagany jest adres, numer księgi i data zakupu, ponieważ nie zawsze od razu wiemy wszystkie dane, które chcemy dodać - resztę można uzupełnić później. Jeśli dany rok w “Stawki podatków” zostanie usunięty to odpowiednie miesiące w “Sumy powierzchni” również oraz wszystkie nieruchomości, które były posiadane w danym roku. Dlatego usuwanie w “Stawki podatków” zarezerwowane jest dla admina.

5. Modele

ErrorViewModel.cs

```
namespace ProperTax.Models
{
    public class ErrorViewModel
    {
        public string? RequestId { get; set; }

        public bool ShowRequestId => !string.IsNullOrEmpty(RequestId);
    }
}
```

Nieruchomosc.cs
Id jest kluczem głównym i automatycznie się inkrementuje. Jest tutaj wiele zmiennych ale większość z nich nie jest ważna dla samego działania programu. Najważniejsze to powierzchnie i daty kupienia i sprzedania.

```
using System.ComponentModel.DataAnnotations;

namespace ProperTax.Models
{
    public class Nieruchomosc
    {
        public int Id { get; set; }
        [Display(Name = "Nr Księgi wieczystej")]
        public required string NrKsiegiWieczystej { get; set; }
        [Display(Name = "Adres")]
        public required string Adres { get; set; }
        [Range(0, 2147483647, ErrorMessage = "Liczba nie może być ujemna.")]
        [Display(Name = "Nr Obrębu")]
        public int? NrObrebu { get; set; }
        [Range(0, 2147483647, ErrorMessage = "Liczba nie może być ujemna.")]
        [Display(Name = "Identyfikator działki")]
        public int? IdDzialki { get; set; }
```

```
        [Display(Name = "Udział [100m]")]
        public string? Udzial100m { get; set; }
        [Range(0, 2147483647, ErrorMessage = "Powierzchnia nie może być ujemna i
ma 2 miejsca po przecinku.")]
        [Display(Name = "Powierzchnia użytkowa budynku [m^2]")]
        public double? PowierzchniaUzytkowaBudynku { get; set; }
        [Range(0, 2147483647, ErrorMessage = "Powierzchnia nie może być ujemna i
ma 2 miejsca po przecinku.")]
        [Display(Name = "[Grunty] Powierzchnia działki mieszkalnej [m^2]")]
        public double? KategoriaGruntyPowierzchniaDzialkiMieszkalnej { get; set; }
        [Range(0, 2147483647, ErrorMessage = "Powierzchnia nie może być ujemna i
ma 2 miejsca po przecinku.")]
        [Display(Name = "[Grunty] Powierzchnia działki NIEmieszkalnej [m^2]")]
        public double? KategoriaGruntyPowierzchniaDzialkiNiemieszkalnej { get; set; }
        [Range(0, 2147483647, ErrorMessage = "Powierzchnia nie może być ujemna i
ma 2 miejsca po przecinku.")]
        [Display(Name = "[Budynki] Powierzchnia użytkowa mieszkalna [m^2]")]
        public double? KategoriaBudynkiPowierzchniaUzytkowaMieszkalna { get; set; }
        [Range(0, 2147483647, ErrorMessage = "Powierzchnia nie może być ujemna i
ma 2 miejsca po przecinku.")]
        [Display(Name = "[Budynki] Powierzchnia użytkowa NIEmieszkalna [m^2]")]
        public double? KategoriaBudynkiPowierzchniaUzytkowaNiemieszkalna { get;
set; }
        [Range(0, 2147483647, ErrorMessage = "Wartość nie może być ujemna i ma 2
miejsca po przecinku.")]
        [Display(Name = "[Budowle] Wartość budowli [zł]")]
        public double? KategoriaWartoscBudowli { get; set; }
        [Display(Name = "Forma władania")]
        public string? FormaWladania { get; set; }
        [Display(Name = "Data zakupienia")]
        public required DateTime DataKupienia { get; set; }
        [Display(Name = "Data sprzedaży")]
        public DateTime? DataSprzedania { get; set; }
        public string? Komentarz { get; set; }
    }
}
```

StawkaPodatku.cs
Rok jest kluczem głównym (jest zabezpieczone aby był unikalny). Pozostałe zmienne to stawki podatków na ten rok. Nie można edytować Rok.

```
using System.ComponentModel.DataAnnotations.Schema;
using System.ComponentModel.DataAnnotations;
```

```
namespace ProperTax.Models
{
    public class StawkaPodatku
    {
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.None)]   //Ten atrybut ustawia ze Rok nie jest automatycznie inkrementowany w bazie danych tylko ustawiany przez uzytkownika
        [Range(2000, 2100, ErrorMessage = "Czy to dobry rok? 🤔")]
        public int Rok { get; set; }
        [Range(0, 2147483647, ErrorMessage = "Stawka nie może być ujemna i ma 2 miejsca po przecinku.")]
        [Display(Name = "[Grunty] Stauwka od powierzchni działki mieszkalnej [zł/m^2]")]
        public double StawkaKategoriiGruntyPowierzchniaDzialkiMieszkalnej { get; set; }
        [Range(0, 2147483647, ErrorMessage = "Stawka nie może być ujemna i ma 2 miejsca po przecinku.")]
        [Display(Name = "[Grunty] Stawka od powierzchni działki NIEmieszkalnej [zł/m^2]")]
        public double StawkaKategoriiGruntyPowierzchniaDzialkiNiemieszkalnej { get; set; }
        [Range(0, 2147483647, ErrorMessage = "Stawka nie może być ujemna i ma 2 miejsca po przecinku.")]
        [Display(Name = "[Budynki] Stawka od powierzchni użytkowej mieszkalnej [zł/m^2]")]
        public double StawkaKategoriiBudynkiPowierzchniaUzytkowaMieszkalna { get; set; }
        [Range(0, 2147483647, ErrorMessage = "Stawka nie może być ujemna i ma 2 miejsca po przecinku.")]
        [Display(Name = "[Budynki] Stawka od powierzchni użytkowej NIEmieszkalnej [zł/m^2]")]
        public double StawkaKategoriiBudynkiPowierzchniaUzytkowaNiemieszkalna { get; set; }
        [Range(0, 2147483647, ErrorMessage = "Stawka nie może być ujemna i ma 2 miejsca po przecinku.")]
        [Display(Name = "[Budowle] Stawka od wartości budowli [% wartości]")]
        public double StawkaKategoriiWartoscBudowli { get; set; }
        public string? Komentarz { get; set; }
    }
}
```

SumaPowierzchni.cs

Rok-miesiac jest kluczem głównym (jest zabezpieczone aby był unikalny). Reprezentuje jaka jest suma powierzchni wszystkich nieruchomości posiadanych w danym miesiącu.

```
using System.ComponentModel.DataAnnotations.Schema;
using System.ComponentModel.DataAnnotations;

namespace ProperTax.Models
{
    public class SumaPowierzchni
    {
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.None)]   //Ten atrybut ustawia
ze RokMiesiac nie jest automatycznie inkrementowany w bazie danych tylko
ustawiany przez uzytkownika
        [Display(Name = "Rok-Miesiąc")]
        public DateTime RokMiesiac { get; set; }
        [Display(Name = "[Grunty] Suma powierzchni działki mieszkalnej [m^2]")]
        public double
SumaPowierzchniKategoriaGruntyPowierzchniaDzialkiMieszkalnej { get; set; }
        [Display(Name = "[Grunty] Suma powierzchni działki NIEmieszkalnej [m^2]")]
        public double
SumaPowierzchniKategoriaGruntyPowierzchniaDzialkiNiemieszkalnej { get; set; }
        [Display(Name = "[Budynki] Suma powierzchni użytkowa mieszkalna [m^2]")]
        public double
SumaPowierzchniKategoriaBudynkiPowierzchniaUzytkowaMieszkalna { get; set; }
        [Display(Name = "[Budynki] Suma powierzchni użytkowa NIEmieszkalna
[m^2]")]
        public double
SumaPowierzchniKategoriaBudynkiPowierzchniaUzytkowaNiemieszkalna { get; set;
}
        [Display(Name = "[Budowle] Suma wartości budowli [zł]")]
        public double SumaPowierzchniKategoriaWartoscBudowli { get; set; }
    }
}
```

6. Kontrolery

HomeController.cs

```
using System.Diagnostics;
using Microsoft.AspNetCore.Mvc;
using ProperTax.Models;
```

```
namespace ProperTax.Controllers
{
    public class HomeController : Controller
    {
        private readonly ILogger<HomeController> _logger;

        public HomeController(ILogger<HomeController> logger)
        {
            _logger = logger;
        }

        public IActionResult Index()
        {
            return View();
        }

        public IActionResult Privacy()
        {
            return View();
        }

        [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None,
NoStore = true)]
        public IActionResult Error()
        {
            return View(new ErrorViewModel { RequestId = Activity.Current?.Id ??
HttpContext.TraceIdentifier });
        }
    }
}
```

NieruchomosciController.cs
Metoda Create przypisuje domyślne wartości gdy użytkownik pozostawi puste pola. Waliduje czy data kupienia jest przed sprzedania. Sprawdza czy data kupienia i sprzedania ma swój rok w StawkaPodatku. Wywołuje AktualizujWszystkieSumyPowierzchni. Tworzy Nieruchomosci.

Metoda Edit przypisuje domyślne wartości gdy użytkownik pozostawi puste pola. Waliduje czy data kupienia jest przed sprzedania. Sprawdza czy data kupienia i sprzedania ma swój rok w StawkaPodatku. Wywołuje AktualizujWszystkieSumyPowierzchni. Edytuje Nieruchomosci.

Metoda DeleteConfirmed wywołuje AktualizujWszystkieSumyPowierzchni. Usuwa Nieruchomosc.

Metoda AktualizujWszystkieSumyPowierzchni liczy sumę powierzchni nieruchomości dla SumaPowierzchni.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using ProperTax.Data;
using ProperTax.Models;

namespace ProperTax.Controllers
{
    [Authorize]
    public class NieruchomosciController : Controller
    {
        private readonly ApplicationDbContext _context;

        public NieruchomosciController(ApplicationDbContext context)
        {
            _context = context;
        }

        // GET: Nieruchomosci
        public async Task<IActionResult> Index()
        {
            return View(await _context.Nieruchomosci.ToListAsync());
        }

        // GET: Nieruchomosci/Details/5
        public async Task<IActionResult> Details(int? id)
        {
            if (id == null)
            {
                return NotFound();
            }

            var nieruchomosc = await _context.Nieruchomosci
                .FirstOrDefaultAsync(m => m.Id == id);
            if (nieruchomosc == null)
```

```
            {
                return NotFound();
            }

            return View(nieruchomosc);
        }

        // GET: Nieruchomosci/Create
        public IActionResult Create()
        {
            return View();
        }

        // POST: Nieruchomosci/Create
        // To protect from overposting attacks, enable the specific properties you want to
bind to.
        // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult>
Create([Bind("Id,NrKsiegiWieczystej,Adres,NrObrebu,IdDzialki,Udzial100m,PowierzchniaUzytkowaBudynku,KategoriaGruntyPowierzchniaDzialkiMieszkalnej,KategoriaGruntyPowierzchniaDzialkiNiemieszkalnej,KategoriaBudynkiPowierzchniaUzytkowaMieszkalna,KategoriaBudynkiPowierzchniaUzytkowaNiemieszkalna,KategoriaWartoscBudowli,FormaWladania,DataKupienia,DataSprzedania,Komentarz")] Nieruchomosc
nieruchomosc)
        {
            //Dzieki tej ifologii w formularzu zostaną przesłane domyślne wartości gdy
użytkownik zostawi te pola puste
            //Probowalem ustawic domyslne wartosci w klasie Nieruchomosc ale nie
dzialalo i do bazy przesylane byly wartosci null
            if (!nieruchomosc.NrObrebu.HasValue)
            {
                nieruchomosc.NrObrebu = 0; // Ustaw domyślną wartość
            }
            if (!nieruchomosc.IdDzialki.HasValue)
            {
                nieruchomosc.IdDzialki = 0; // Ustaw domyślną wartość
            }
            if (!nieruchomosc.PowierzchniaUzytkowaBudynku.HasValue)
            {
                nieruchomosc.PowierzchniaUzytkowaBudynku = 0; // Ustaw domyślną
wartość
            }
```

```
            if (!nieruchomosc.KategoriaGruntyPowierzchniaDzialkiMieszkalnej.HasValue)
            {
                nieruchomosc.KategoriaGruntyPowierzchniaDzialkiMieszkalnej = 0; //
Ustaw domyślną wartość
            }
            if
(!nieruchomosc.KategoriaGruntyPowierzchniaDzialkiNiemieszkalnej.HasValue)
            {
                nieruchomosc.KategoriaGruntyPowierzchniaDzialkiNiemieszkalnej = 0; //
Ustaw domyślną wartość
            }
            if
(!nieruchomosc.KategoriaBudynkiPowierzchniaUzytkowaMieszkalna.HasValue)
            {
                nieruchomosc.KategoriaBudynkiPowierzchniaUzytkowaMieszkalna = 0; //
Ustaw domyślną wartość
            }
            if
(!nieruchomosc.KategoriaBudynkiPowierzchniaUzytkowaNiemieszkalna.HasValue)
            {
                nieruchomosc.KategoriaBudynkiPowierzchniaUzytkowaNiemieszkalna = 0;
// Ustaw domyślną wartość
            }
            if (!nieruchomosc.KategoriaWartoscBudowli.HasValue)
            {
                nieruchomosc.KategoriaWartoscBudowli = 0; // Ustaw domyślną wartość
            }
            if (string.IsNullOrWhiteSpace(nieruchomosc.Komentarz))
            {
                nieruchomosc.Komentarz = ""; // Ustaw domyślną wartość
            }
            if (string.IsNullOrWhiteSpace(nieruchomosc.FormaWladania))
            {
                nieruchomosc.FormaWladania = "własność"; // Ustaw domyślną wartość
            }
            nieruchomosc.Udzial100m =
string.Concat((Math.Max(nieruchomosc.KategoriaBudynkiPowierzchniaUzytkowaMie
szkalna ?? 0, nieruchomosc.KategoriaBudynkiPowierzchniaUzytkowaNiemieszkalna
?? 0) * 100).ToString(), "/", (nieruchomosc.PowierzchniaUzytkowaBudynku *
100).ToString());

            //Sprawdza czy data kupienia jest mniejsza niz sprzedania
            if (nieruchomosc.DataSprzedania.HasValue)
            {
```

```
            if (nieruchomosc.DataSprzedania < nieruchomosc.DataKupienia)
            {
                ModelState.AddModelError("DataSprzedania", "Data sprzedaży nie
może być wcześniejsza niż data kupienia.");
            }
        }

        // Sprawdzenie, czy rok zakupu istnieje w tabeli StawkiPodatkow
        bool rokZakupuIstnieje = _context.StawkiPodatkow.Any(s => s.Rok ==
nieruchomosc.DataKupienia.Year);
        if (!rokZakupuIstnieje)
        {
            ModelState.AddModelError("DataKupienia", "Przed wpisaniem daty z tym
rokiem musisz dodać go w Stawki podatków.");
        }

        // Sprawdzenie, czy rok sprzedaży istnieje w tabeli StawkiPodatkow
        if (nieruchomosc.DataSprzedania.HasValue)
        {
            bool rokSprzedazyIstnieje = _context.StawkiPodatkow.Any(s => s.Rok ==
nieruchomosc.DataSprzedania.Value.Year);
            if (!rokSprzedazyIstnieje)
            {
                ModelState.AddModelError("DataSprzedania", "Przed wpisaniem daty z
tym rokiem musisz dodać go w Stawki podatków.");
            }
        }

        if (ModelState.IsValid)
        {
            _context.Add(nieruchomosc);
            await AktualizujWszystkieSumyPowierzchni();
            await _context.SaveChangesAsync();
            return RedirectToAction(nameof(Index));
        }
        return View(nieruchomosc);
    }

    // GET: Nieruchomosci/Edit/5
    public async Task<IActionResult> Edit(int? id)
    {
        if (id == null)
        {
            return NotFound();
```

```
            }

            var nieruchomosc = await _context.Nieruchomosci.FindAsync(id);
            if (nieruchomosc == null)
            {
                return NotFound();
            }
            return View(nieruchomosc);
        }

        // POST: Nieruchomosci/Edit/5
        // To protect from overposting attacks, enable the specific properties you want to
bind to.
        // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Edit(int id,
[Bind("Id,NrKsiegiWieczystej,Adres,NrObrebu,IdDzialki,Udzial100m,PowierzchniaUz
ytkowaBudynku,KategoriaGruntyPowierzchniaDzialkiMieszkalnej,KategoriaGruntyPo
wierzchniaDzialkiNiemieszkalnej,KategoriaBudynkiPowierzchniaUzytkowaMieszkaln
a,KategoriaBudynkiPowierzchniaUzytkowaNiemieszkalna,KategoriaWartoscBudowli,
FormaWladania,DataKupienia,DataSprzedania,Komentarz")] Nieruchomosc
nieruchomosc)
        {
            if (id != nieruchomosc.Id)
            {
                return NotFound();
            }

            //Dzieki tej ifologii w formularzu zostaną przesłane domyślne wartości gdy
użytkownik zostawi te pola puste
            //Probowalem ustawic domyslne wartosci w klasie Nieruchomosc ale nie
dzialalo i do bazy przesylane byly wartosci null
            if (!nieruchomosc.NrObrebu.HasValue)
            {
                nieruchomosc.NrObrebu = 0; // Ustaw domyślną wartość
            }
            if (!nieruchomosc.IdDzialki.HasValue)
            {
                nieruchomosc.IdDzialki = 0; // Ustaw domyślną wartość
            }
            if (!nieruchomosc.PowierzchniaUzytkowaBudynku.HasValue)
            {
```

```
            nieruchomosc.PowierzchniaUzytkowaBudynku = 0; // Ustaw domyślną
wartość
        }
        if (!nieruchomosc.KategoriaGruntyPowierzchniaDzialkiMieszkalnej.HasValue)
        {
            nieruchomosc.KategoriaGruntyPowierzchniaDzialkiMieszkalnej = 0; //
Ustaw domyślną wartość
        }
        if
(!nieruchomosc.KategoriaGruntyPowierzchniaDzialkiNiemieszkalnej.HasValue)
        {
            nieruchomosc.KategoriaGruntyPowierzchniaDzialkiNiemieszkalnej = 0; //
Ustaw domyślną wartość
        }
        if
(!nieruchomosc.KategoriaBudynkiPowierzchniaUzytkowaMieszkalna.HasValue)
        {
            nieruchomosc.KategoriaBudynkiPowierzchniaUzytkowaMieszkalna = 0; //
Ustaw domyślną wartość
        }
        if
(!nieruchomosc.KategoriaBudynkiPowierzchniaUzytkowaNiemieszkalna.HasValue)
        {
            nieruchomosc.KategoriaBudynkiPowierzchniaUzytkowaNiemieszkalna = 0;
// Ustaw domyślną wartość
        }
        if (!nieruchomosc.KategoriaWartoscBudowli.HasValue)
        {
            nieruchomosc.KategoriaWartoscBudowli = 0; // Ustaw domyślną wartość
        }
        if (string.IsNullOrWhiteSpace(nieruchomosc.Komentarz))
        {
            nieruchomosc.Komentarz = ""; // Ustaw domyślną wartość
        }
        if (string.IsNullOrWhiteSpace(nieruchomosc.FormaWladania))
        {
            nieruchomosc.FormaWladania = "własność"; // Ustaw domyślną wartość
        }
        nieruchomosc.Udzial100m =
string.Concat((Math.Max(nieruchomosc.KategoriaBudynkiPowierzchniaUzytkowaMie
szkalna ?? 0, nieruchomosc.KategoriaBudynkiPowierzchniaUzytkowaNiemieszkalna
?? 0) * 100).ToString(), "/", (nieruchomosc.PowierzchniaUzytkowaBudynku *
100).ToString());
```

```
            //Sprawdza czy data kupienia jest mniejsza niz sprzedania
            if (nieruchomosc.DataSprzedania.HasValue)
            {
                if (nieruchomosc.DataSprzedania < nieruchomosc.DataKupienia)
                {
                    ModelState.AddModelError("DataSprzedania", "Data sprzedaży nie
może być wcześniejsza niż data kupienia.");
                }
            }

            // Sprawdzenie, czy rok zakupu istnieje w tabeli StawkiPodatkow
            bool rokZakupuIstnieje = _context.StawkiPodatkow.Any(s => s.Rok ==
nieruchomosc.DataKupienia.Year);
            if (!rokZakupuIstnieje)
            {
                ModelState.AddModelError("DataKupienia", "Przed wpisaniem daty z tym
rokiem musisz dodać go w Stawki podatków.");
            }

            // Sprawdzenie, czy rok sprzedaży istnieje w tabeli StawkiPodatkow
            if (nieruchomosc.DataSprzedania.HasValue)
            {
                bool rokSprzedazyIstnieje = _context.StawkiPodatkow.Any(s => s.Rok ==
nieruchomosc.DataSprzedania.Value.Year);
                if (!rokSprzedazyIstnieje)
                {
                    ModelState.AddModelError("DataSprzedania", "Przed wpisaniem daty z
tym rokiem musisz dodać go w Stawki podatków.");
                }
            }

            if (ModelState.IsValid)
            {
                try
                {
                    _context.Update(nieruchomosc);
                    await _context.SaveChangesAsync();
                }
                catch (DbUpdateConcurrencyException)
                {
                    if (!NieruchomoscExists(nieruchomosc.Id))
                    {
                        return NotFound();
                    }
```

```
            else
            {
                throw;
            }
        }
        return RedirectToAction(nameof(Index));
    }
    return View(nieruchomosc);
}

// GET: Nieruchomosci/Delete/5
public async Task<IActionResult> Delete(int? id)
{
    if (id == null)
    {
        return NotFound();
    }

    var nieruchomosc = await _context.Nieruchomosci
        .FirstOrDefaultAsync(m => m.Id == id);
    if (nieruchomosc == null)
    {
        return NotFound();
    }

    return View(nieruchomosc);
}

// POST: Nieruchomosci/Delete/5
[HttpPost, ActionName("Delete")]
[ValidateAntiForgeryToken]
public async Task<IActionResult> DeleteConfirmed(int id)
{
    var nieruchomosc = await _context.Nieruchomosci.FindAsync(id);
    if (nieruchomosc != null)
    {
        _context.Nieruchomosci.Remove(nieruchomosc);
    }

    await _context.SaveChangesAsync();
    return RedirectToAction(nameof(Index));
}

private bool NieruchomoscExists(int id)
```

```
        {
            return _context.Nieruchomosci.Any(e => e.Id == id);
        }

        public async Task AktualizujWszystkieSumyPowierzchni()
        {
            // Pobierz wszystkie nieruchomości
            var wszystkieMiesiace = await _context.SumyPowierzchni.ToListAsync();

            foreach (var miesiac in wszystkieMiesiace)
            {
                DateTime aktualnyRokMiesiacPoczatek = new
DateTime(miesiac.RokMiesiac.Year, miesiac.RokMiesiac.Month, 1);
                DateTime aktualnyRokMiesiacKoniec =
aktualnyRokMiesiacPoczatek.AddMonths(1).AddDays(-1);

                // Przechwytujemy wszystkie nieruchomości, które były posiadane w
danym miesiącu
                /*var nieruchomosci = await _context.Nieruchomosci.Where(n =>
                    n.DataKupienia <= aktualnyRokMiesiacPoczatek &&
                    (n.DataSprzedania == null || n.DataSprzedania >=
aktualnyRokMiesiacKoniec))
                    .ToListAsync();*/

                var nieruchomosci = await _context.Nieruchomosci
                    .FromSqlInterpolated($@"
                        SELECT *
                        FROM Nieruchomosci
                        WHERE
                            DATEFROMPARTS(YEAR(DataKupienia), MONTH(DataKupienia),
1) <= {aktualnyRokMiesiacPoczatek}
                            AND
                            (DataSprzedania IS NULL OR
                             DataSprzedania >= {aktualnyRokMiesiacKoniec})")
                    .ToListAsync();



                Console.WriteLine($"Nieruchomości w miesiącu {miesiac.RokMiesiac}:");
                foreach (var n in nieruchomosci)
                {
                    Console.WriteLine($"Id: {n.Id}, DataKupienia: {n.DataKupienia},
DataSprzedania: {n.DataSprzedania}");
                }
```

```
                // Sumowanie wartości
                double SumaPowierzchniKategoriaGruntyPowierzchniaDzialkiMieszkalnej
= nieruchomosci.Sum(n => n.KategoriaGruntyPowierzchniaDzialkiMieszkalnej ?? 0);
                double
SumaPowierzchniKategoriaGruntyPowierzchniaDzialkiNiemieszkalnej =
nieruchomosci.Sum(n => n.KategoriaGruntyPowierzchniaDzialkiNiemieszkalnej ??
0);
                double
SumaPowierzchniKategoriaBudynkiPowierzchniaUzytkowaMieszkalna =
nieruchomosci.Sum(n => n.KategoriaBudynkiPowierzchniaUzytkowaMieszkalna ??
0);
                double
SumaPowierzchniKategoriaBudynkiPowierzchniaUzytkowaNiemieszkalna =
nieruchomosci.Sum(n => n.KategoriaBudynkiPowierzchniaUzytkowaNiemieszkalna
?? 0);
                double SumaPowierzchniKategoriaWartoscBudowli =
nieruchomosci.Sum(n => n.KategoriaWartoscBudowli ?? 0);

                // Przypisanie sum powierzchni w aktualnym miesiacu
                miesiac.SumaPowierzchniKategoriaGruntyPowierzchniaDzialkiMieszkalnej
= SumaPowierzchniKategoriaGruntyPowierzchniaDzialkiMieszkalnej;

miesiac.SumaPowierzchniKategoriaGruntyPowierzchniaDzialkiNiemieszkalnej =
SumaPowierzchniKategoriaGruntyPowierzchniaDzialkiNiemieszkalnej;

miesiac.SumaPowierzchniKategoriaBudynkiPowierzchniaUzytkowaMieszkalna =
SumaPowierzchniKategoriaBudynkiPowierzchniaUzytkowaMieszkalna;

miesiac.SumaPowierzchniKategoriaBudynkiPowierzchniaUzytkowaNiemieszkalna =
SumaPowierzchniKategoriaBudynkiPowierzchniaUzytkowaNiemieszkalna;
                miesiac.SumaPowierzchniKategoriaWartoscBudowli =
SumaPowierzchniKategoriaWartoscBudowli;
            }

            // Zapisz zmiany w bazie danych
            await _context.SaveChangesAsync();
        }
    }
}
```

StawkiPodatkowController.cs

Metoda Create sprawdza czy Rok jest unikalny. Tworzy 12 instancji miesięcy dla tego roku w SumaPowierzchni. Tworzy StawkiPodatkow.
Metoda Edit sprawdza ile nieruchomości było posiadane w danym roku i pokazuje ich listę. Edytuje StawkiPodatkow.
Metoda DeleteConfirmed usuwa 12 instancji miesięcy dla tego roku w SumaPowierzchni. Usuwa wszystkie nieruchomości posiadane w danym roku. Usuwa StawkiPodatkow.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using ProperTax.Data;
using ProperTax.Models;

namespace ProperTax.Controllers
{
    [Authorize]
    public class StawkiPodatkowController : Controller
    {
        private readonly ApplicationDbContext _context;

        public StawkiPodatkowController(ApplicationDbContext context)
        {
            _context = context;
        }

        // GET: StawkiPodatkow
        public async Task<IActionResult> Index()
        {
            return View(await _context.StawkiPodatkow.ToListAsync());
        }

        // GET: StawkiPodatkow/Details/5
        public async Task<IActionResult> Details(int? id)
        {
            if (id == null)
            {
                return NotFound();
            }
```

```
            var stawkaPodatku = await _context.StawkiPodatkow
                .FirstOrDefaultAsync(m => m.Rok == id);
            if (stawkaPodatku == null)
            {
                return NotFound();
            }

            return View(stawkaPodatku);
        }

        // GET: StawkiPodatkow/Create
        public IActionResult Create()
        {
            return View();
        }

        // POST: StawkiPodatkow/Create
        // To protect from overposting attacks, enable the specific properties you want to
bind to.
        // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult>
Create([Bind("Rok,StawkaKategoriiGruntyPowierzchniaDzialkiMieszkalnej,StawkaKa
tegoriiGruntyPowierzchniaDzialkiNiemieszkalnej,StawkaKategoriiBudynkiPowierzchn
iaUzytkowaMieszkalna,StawkaKategoriiBudynkiPowierzchniaUzytkowaNiemieszkaln
a,StawkaKategoriiWartoscBudowli,Komentarz")] StawkaPodatku stawkaPodatku)
        {
            // Sprawdzenie, czy Rok jest unikalny
            if (_context.StawkiPodatkow.Any(s => s.Rok == stawkaPodatku.Rok))
            {
                // Dodanie komunikatu z linkiem do widoku Index
                ModelState.AddModelError("Rok", "Ten rok został już zapisany. Możesz go
znaleźć i zedytować w Stawki podatków.");
            }

            if (ModelState.IsValid)
            {
                _context.Add(stawkaPodatku);

                //sprawdzenie czy istnieja juz miesiace dla tego roku
                if (_context.SumyPowierzchni.Any(s => s.RokMiesiac.Year ==
stawkaPodatku.Rok))
```

```
            {
                ModelState.AddModelError("", "Miesiące dla tego roku już istnieją w
SumaPowierzchni");
                return View(stawkaPodatku);
            }

            // Tworzenie 12 instancji SumaPowierzchni czyli 12 miesiecy dodawanego
roku
            List<SumaPowierzchni> miesiace = new List<SumaPowierzchni>();
            for (int month = 1; month <= 12; month++)
            {
                var miesiac = new SumaPowierzchni
                {
                    RokMiesiac = new DateTime(stawkaPodatku.Rok, month, 1),
                    SumaPowierzchniKategoriaGruntyPowierzchniaDzialkiMieszkalnej =
0,
                    SumaPowierzchniKategoriaGruntyPowierzchniaDzialkiNiemieszkalnej
= 0,

SumaPowierzchniKategoriaBudynkiPowierzchniaUzytkowaMieszkalna = 0,

SumaPowierzchniKategoriaBudynkiPowierzchniaUzytkowaNiemieszkalna = 0,
                    SumaPowierzchniKategoriaWartoscBudowli = 0
                };

                miesiace.Add(miesiac);
            }

            // Dodanie listy do bazy danych
            _context.SumyPowierzchni.AddRange(miesiace);

            await _context.SaveChangesAsync();
            return RedirectToAction(nameof(Index));
        }
        return View(stawkaPodatku);
    }

    // GET: StawkiPodatkow/Edit/5
    public async Task<IActionResult> Edit(int? id)
    {
        if (id == null)
        {
            return NotFound();
        }
```

```
            var stawkaPodatku = await _context.StawkiPodatkow.FindAsync(id);
            if (stawkaPodatku == null)
            {
                return NotFound();
            }

            // Pobierz powiązane nieruchomości
            var powiazaneNieruchomosci = _context.Nieruchomosci
                .Where(n => n.DataKupienia.Year <= id &&
                            (n.DataSprzedania == null || n.DataSprzedania.Value.Year >= id))
                .ToList();

            // Przekaż informacje do widoku
            ViewBag.PowiazaneNieruchomosci = powiazaneNieruchomosci;
            ViewBag.IloscNieruchomosci = powiazaneNieruchomosci.Count;

            return View(stawkaPodatku);
        }

        // POST: StawkiPodatkow/Edit/5
        // To protect from overposting attacks, enable the specific properties you want to
bind to.
        // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Edit(int id,
[Bind("Rok,StawkaKategoriiGruntyPowierzchniaDzialkiMieszkalnej,StawkaKategorii
GruntyPowierzchniaDzialkiNiemieszkalnej,StawkaKategoriiBudynkiPowierzchniaUzy
tkowaMieszkalna,StawkaKategoriiBudynkiPowierzchniaUzytkowaNiemieszkalna,Sta
wkaKategoriiWartoscBudowli,Komentarz")] StawkaPodatku stawkaPodatku)
        {
            if (id != stawkaPodatku.Rok)
            {
                return NotFound();
            }

            if (ModelState.IsValid)
            {
                try
                {
                    _context.Update(stawkaPodatku);
                    await _context.SaveChangesAsync();
                }
```

```
            catch (DbUpdateConcurrencyException)
            {
                if (!StawkaPodatkuExists(stawkaPodatku.Rok))
                {
                    return NotFound();
                }
                else
                {
                    throw;
                }
            }
            return RedirectToAction(nameof(Index));
        }
        return View(stawkaPodatku);
    }

    // GET: StawkiPodatkow/Delete/5
    [Authorize(Roles = "Admin")]
    public async Task<IActionResult> Delete(int? id)
    {
        if (id == null)
        {
            return NotFound();
        }

        var stawkaPodatku = await _context.StawkiPodatkow
            .FirstOrDefaultAsync(m => m.Rok == id);
        if (stawkaPodatku == null)
        {
            return NotFound();
        }

        // Pobierz powiązane nieruchomości
        var powiazaneNieruchomosci = _context.Nieruchomosci
            .Where(n => stawkaPodatku.Rok >= n.DataKupienia.Year &&
                (n.DataSprzedania == null || stawkaPodatku.Rok <=
n.DataSprzedania.Value.Year)).ToList();

        // Przygotuj widok z modelem
        ViewBag.PowiazaneNieruchomosci = powiazaneNieruchomosci;
        ViewBag.IloscNieruchomosci = powiazaneNieruchomosci.Count;

        return View(stawkaPodatku);
    }
```

```
        // POST: StawkiPodatkow/Delete/5
        [Authorize(Roles = "Admin")]
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> DeleteConfirmed(int id)
        {
            // Pobranie obiektu stawki podatkowej do usunięcia
            var stawkaPodatku = await _context.StawkiPodatkow.FindAsync(id);

            if (stawkaPodatku == null)
            {
                return NotFound();
            }

            // Pobranie powiązanych miesięcy z SumaPowierzchni dla usuwanego roku
            var powiazaneMiesiace = _context.SumyPowierzchni
                .Where(s => s.RokMiesiac.Year == id)
                .ToList();

            // Usunięcie powiązanych miesięcy
            _context.SumyPowierzchni.RemoveRange(powiazaneMiesiace);

            // Pobranie powiązanych nieruchomości, których rok zakupu i rok sprzedaży
(jeśli istnieje)
            // mieszczą się w zakresie roku stawki podatkowej
            var powiazaneNieruchomosci = _context.Nieruchomosci
                .Where(n => n.DataKupienia.Year <= id &&
                            (n.DataSprzedania == null || n.DataSprzedania.Value.Year >= id))
                .ToList();

            // Usunięcie powiązanych nieruchomości
            _context.Nieruchomosci.RemoveRange(powiazaneNieruchomosci);

            // Usunięcie stawki podatkowej
            _context.StawkiPodatkow.Remove(stawkaPodatku);

            // Zapisanie zmian w bazie danych
            await _context.SaveChangesAsync();

            // Przekierowanie do widoku Index po usunięciu
            return RedirectToAction(nameof(Index));
        }
```

```
        private bool StawkaPodatkuExists(int id)
        {
            return _context.StawkiPodatkow.Any(e => e.Rok == id);
        }
    }
}
```

SumyPowierzchniController.cs
Ma tylko Index i Details.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using ProperTax.Data;
using ProperTax.Models;

namespace ProperTax.Controllers
{
    [Authorize]
    public class SumyPowierzchniController : Controller
    {
        private readonly ApplicationDbContext _context;

        public SumyPowierzchniController(ApplicationDbContext context)
        {
            _context = context;
        }

        // GET: SumyPowierzchni
        public async Task<IActionResult> Index()
        {
            return View(await _context.SumyPowierzchni.ToListAsync());
        }

        // GET: SumyPowierzchni/Details/5
        public async Task<IActionResult> Details(DateTime? id)
        {
            if (id == null)
            {
                return NotFound();
```

```
            }

            var sumaPowierzchni = await _context.SumyPowierzchni
                .FirstOrDefaultAsync(m => m.RokMiesiac == id);
            if (sumaPowierzchni == null)
            {
                return NotFound();
            }

            return View(sumaPowierzchni);
        }
    }
}
```

7. Opis systemu użytkowników

Są 2 role użytkowników: Admin i Bookkeeper. Admin ma dostęp do wszystkiego. Bookkeeper ma dostęp do wszystkiego poza usuwaniem StawkaPodatku. Strona Home jest dostępna bez logowania, pozostałe wymagają logowania.
Passy dla admina to:
email: admin@mail.com
hasło: Admin1234!

8. Charakterystyka ciekawej funkcjonalności

Podczas próby usunięcia lub edytowania StawkiPodatków jest ostrzeżenie z listą wszystkich dotkniętych tym nieruchomości i usuwa wszystkie dotknięte tym nieruchomości.
W “Sumy podatków” jest liczona na dany miesiąc suma powierzchni wszystkich posiadanych nieruchomości. (Z jakiegoś powodu nie liczy najstarszej nieruchomości - do naprawy)