

Projekt Baz Danych

Dziennik nauczyciela.

Students Book

Projektowanie systemów baz danych.

Prowadzący: dr Aleksander Kłosov

Autorzy:

Łukasz Kuczma, Konrad Szatanek
Informatyka, rok II, semestr IV

Specjalność: Programowanie aplikacji mobilnych i internetowych
 Państwowa Wyższa Szkoła Zawodowa im. Witelona w Legnicy

Wydział Nauk technicznych i ekonomicznych

Spis treści

1. Założenia i koncepcja.....	3
1.1 Cel bazy danych.....	3
1.2 Opis dziedziny przedmiotowej.....	3
1.2.1 Użytkownicy obsługiwani przez aplikację.....	3
1.2.2 Funkcjonalność względem dostępności użytkownika.....	3
1.3 Założenia Wstępne.....	4
2.Specyfikacja wymagań systemu.....	5
2.1 Przypadki użycia systemu.....	5
2.2 Diagram czynności.....	6
2.2.1 Diagram – Dodawanie ocen.....	6
2.2.2 Diagram – dodawanie nowego użytkownika.....	7
2.2.3 Model konceptualny danych ERD.....	8
3. Model logiczny systemu.....	9
Diagram klas UML.....	9
4. Interfejs użytkownika.....	10
4.1 Projekt graficzny.....	10
4.1.2 Panel główny.....	10
4.1.3 Strefa nauczyciela – dodawanie uczniów i ocen.....	11
4.2 Opis działania aplikacji.....	12
4.2.1 Połączenie z bazą danych – fragment kodu.....	12
4.2.2 Dodawanie osoby – fragment kodu.....	13
4.2.3 Usuwanie osób – fragment kodu.....	14
4.2.4 Ustawienia ogólne - fragment kodu.....	15
4.2.5 Edycja ocen – fragment kodu.....	16
5. Podsumowanie.....	17

Założenia i koncepcja

1.1 Cel bazy danych

Wstępna koncepcja projektu opierała się na stworzeniu prostej aplikacji wspierającej pracę nauczycieli, na przykładowym stopniu nauczania aplikacja ma przechowywać z możliwością edycji dane uczniów i ich osiągnięcia w postaci ocen. Finalnie aplikacja obsługuje 2 rodzaje użytkowników z różnymi dostępami. Celem naszej bazy danych jest głównie przejrzysty wgląd do postępów uczniów i ewentualna korekta osiągniętych wyników.

1.2 Opis dziedziny przedmiotowej

1.2.1 Użytkownicy obsługiwani przez aplikację:

- Nauczyciel,
- Uczeń/Rodzic.

1.2.2 Funkcjonalność względem dostępności użytkownika

a. Nauczyciel

1. Strefa podopiecznych:
 - Dodawanie oraz usuwanie uczniów
2. Strefa ocen:
 - Dodawanie oraz edycja ocen
 - Podgląd ocen

b. Uczeń

1. Strefa podopiecznych:

-Brak dostępu

2. Strefa ocen:

-Brak dostępu

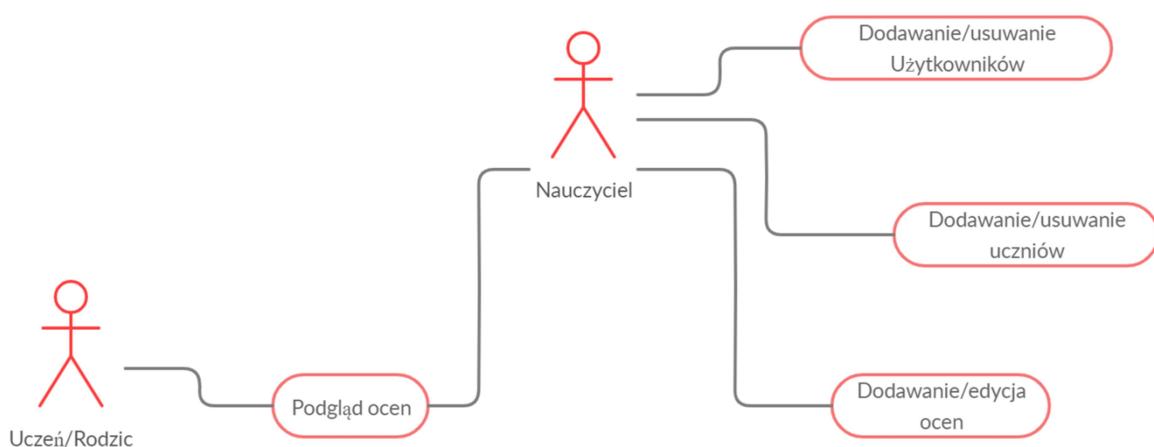
-Podgląd ocen

1.3 Założenia wstępne

Użytkownicy, zgodnie ze swoimi uprawnieniami, mogła pobierać dane z bazy danych oraz je dodawać lub edytować. Administrator - w naszym przypadku Nauczyciel ma możliwość usuwania oraz edycji rekordów oraz użytkowników.

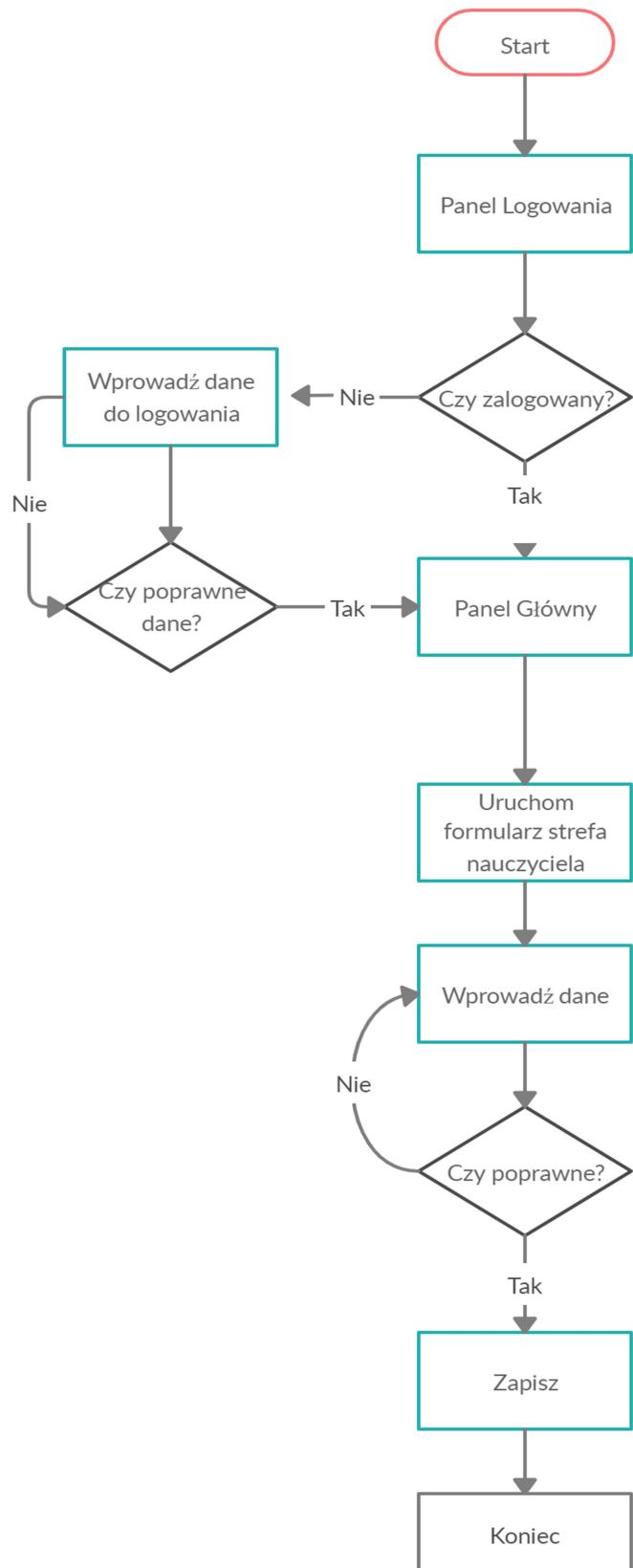
2. Specyfikacja wymagań systemu

2.1 Przypadki użycia systemu

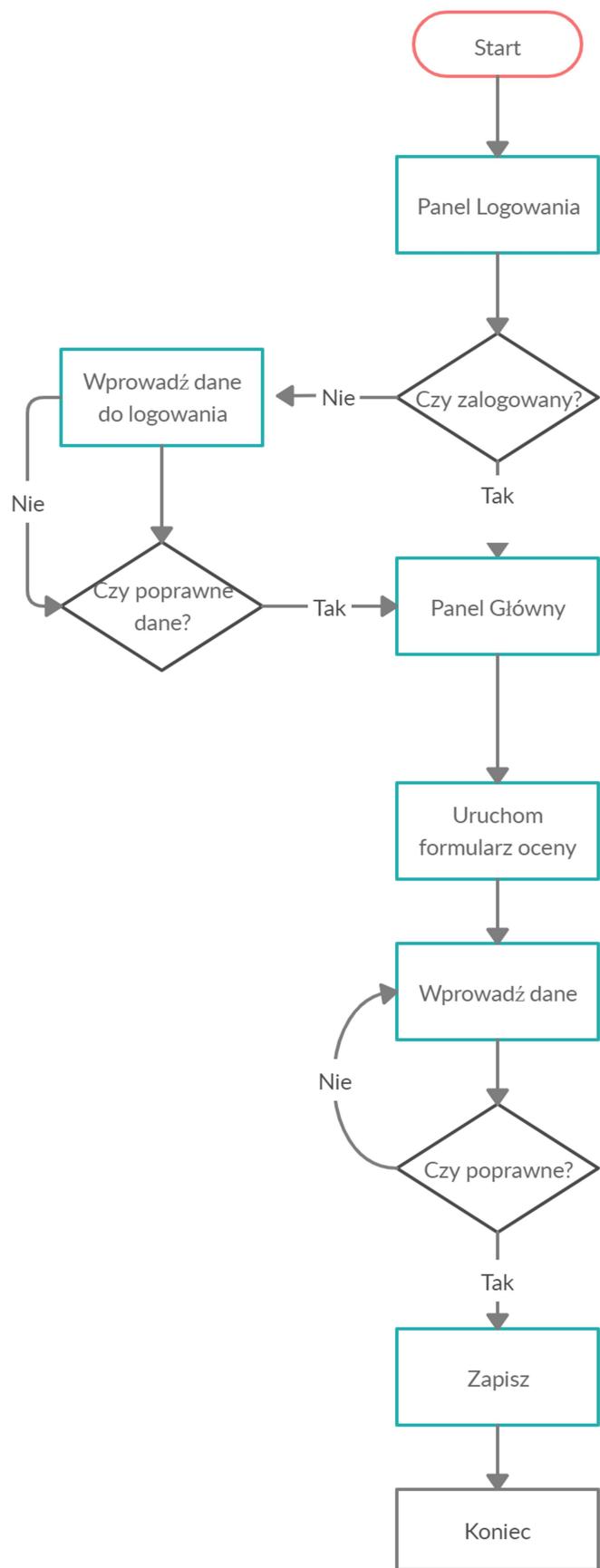


2.2 Diagram czynności

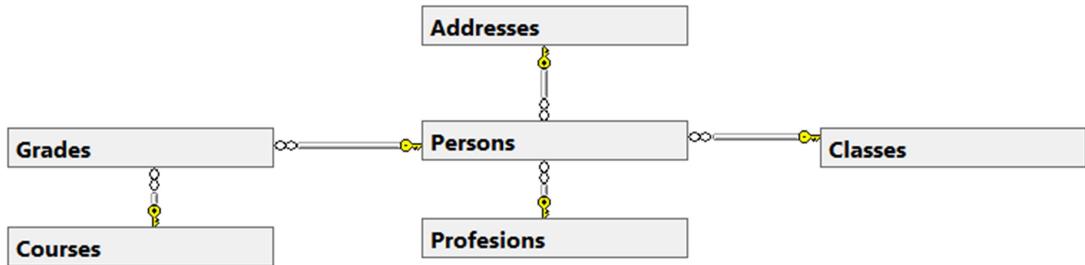
2.2.1 Diagram – dodawania nowego ucznia.



2.2.2 Diagram – dodawania ocen.

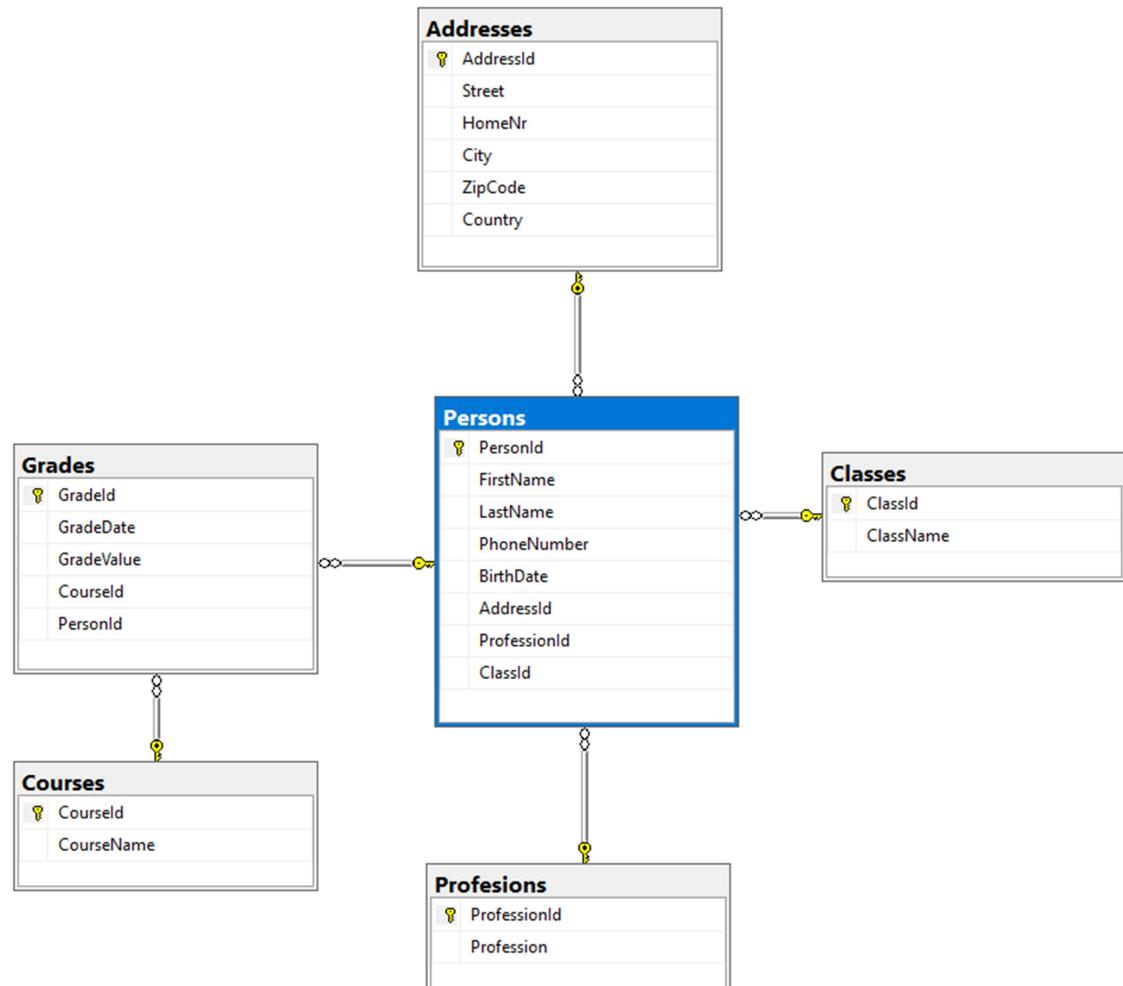


2.2.3 Model konceptualny danych ERD



3. Model logiczny systemu.

Diagram klas UML



4. Projekt graficzny

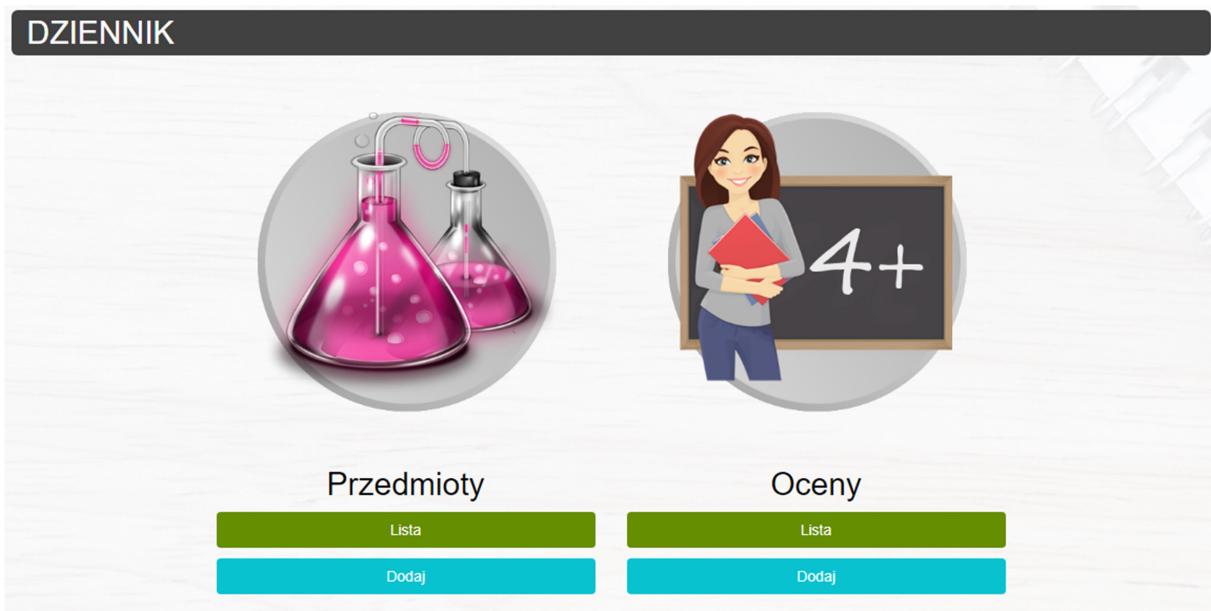
4.1.2 Panel główny.

The screenshot shows the main interface of the 'Students Book' application. At the top, there's a navigation bar with links: 'Students Book', 'Adresy', 'Osoby', 'Klasy', 'Przedmioty', 'Profesje', and 'Oceny'. On the right side of the top bar are 'O projekcie' and 'Kontakt'. Below the navigation is a header with the text 'Students Book' and 'Szkołna elektroniczna baza danych'. The main content area is divided into sections: 'DANE OSOBOWE' (with icons for Adresy, Klasy, Osoby, and Profesje), 'DZIENNIK' (with icons for laboratory glassware and a teacher holding a red book), and a sidebar featuring a notepad and pen. The bottom left corner contains copyright information: '© 2020 - Students Book - n2PAMIV2 • Łukasz Kuczma • Konrad Szatanek'.

4.1.3 Strefa Dane osobowe – Adresy, Klasy, Osoby, Profesje.

This screenshot shows the 'DANE OSOBOWE' section of the application. It features four circular icons representing 'Adresy' (location pin), 'Klasy' (classroom), 'Osoby' (group of people), and 'Profesje' (profession). Below each icon is a label and two buttons: 'Lista' (green) and 'Dodaj' (blue). The background has a light gray grid pattern.

4.1.4 Strefa Dziennik – Przedmioty, Oceny.



4.2 Opis działania aplikacji.

Aplikacja jest prosta w obsłudze oraz intuicyjna. Pozwala na tworzenie nowych osób oraz umieszczenia ich w systemie. Nauczyciele mogą dodawać oraz usuwać osoby, przypisywać oceny do uczniów. Mają wgląd do wszystkich ocen przez co mogą weryfikować postępy nauczania.

4.2.1 Połączenie z bazą danych – fragment kodu.

```
namespace StudentsBook
{
    public class Startup
    {
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
        }

        public IConfiguration Configuration { get; }

        // This method gets called by the runtime. Use this method to add services to the container.
        public void ConfigureServices(IServiceCollection services)
        {
            services.Configure<CookiePolicyOptions>(options =>
            {
                // This lambda determines whether user consent for non-essential cookies is needed for a given request.
                options.CheckConsentNeeded = context => true;
                options.MinimumSameSitePolicy = SameSiteMode.None;
            });

            services.AddDbContext<AppDbContext>(options => options
                .UseSqlServer(Configuration.GetConnectionString("DbConnS")));

            services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version_2_1);
        }

        // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
        public void Configure(IApplicationBuilder app, IHostingEnvironment env)
        {
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
            }
            else
        }
    }
}
```

```
{
    "ConnectionStrings": {
        "DbConnS": "Server=(LocalDb)\\MSSQLLocalDB;Database=SchoolBook;Trusted_Connection=True;MultipleActiveResultSets=True"
    },

    "Logging": {
        "LogLevel": {
            "Default": "Warning"
        }
    },
    "AllowedHosts": "*"
}
```

4.2.2 Dodawanie osoby (Metoda POST – dodawanie adresów przez formularz do bazy) – fragment kodu.

```
// GET: Person/Create
public IActionResult Create()
{
    ViewData["AddressId"] = new SelectList(_context.Addresses, "AddressId", "FullAddress");      // **** edycja
    ViewData["ClassId"] = new SelectList(_context.Classes, "ClassId", "ClassName");
    ViewData["ProfessionId"] = new SelectList(_context.Professions, "ProfessionId", "Profession");
    return View();
}

// POST: Person/Create
// To protect from overposting attacks, please 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("PersonId,FirstName,LastName,PhoneNumber,BirthDate,AddressId,ProfessionId,ClassId")] PersonModel personModel)
{
    if (ModelState.IsValid)
    {
        _context.Add(personModel);
        await _context.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
    }
    ViewData["AddressId"] = new SelectList(_context.Addresses, "AddressId", "FullAddress", personModel.AddressId);      // **** edycja
    ViewData["ClassId"] = new SelectList(_context.Classes, "ClassId", "ClassName", personModel.ClassId);
    ViewData["ProfessionId"] = new SelectList(_context.Professions, "ProfessionId", "Profession", personModel.ProfessionId);
    return View(personModel);
}

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

    var personModel = await _context.Persons.FindAsync(id);
    if (personModel == null)
    {
        return NotFound();
    }
    ViewData["AddressId"] = new SelectList(_context.Addresses, "AddressId", "FullAddress", personModel.AddressId);      //edycja City -> FullAddress
    ViewData["ClassId"] = new SelectList(_context.Classes, "ClassId", "ClassName", personModel.ClassId);
    ViewData["ProfessionId"] = new SelectList(_context.Professions, "ProfessionId", "Profession", personModel.ProfessionId);
    return View(personModel);
}

// POST: Person/Edit/5
```

The screenshot shows a web page titled 'Dodaj nową osobę' (Add new person). The page has a header with navigation links: 'Students Book', 'Adresy', 'Osoby', 'Klasy', 'Przedmioty', 'Profesje', and 'Oceny'. The main content area contains the following form fields:

- Imię**: Andrzej
- Nazwisko**: Kowalski
- Numer telefonu**: 605874652
- Data urodzenia**: 12.06.1999
- Pełny adres**: Moniuszki 45 | Legnica | 59-220
- ProfessionId**: Uczeń
- Klasa**: 2B
- Dodaj** button
- Powrót do listy** button

4.2.3 Usuwanie osób – fragment kodu.

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

    var personModel = await _context.Persons
        .Include(p => p.Address)
        .Include(p => p.Class)
        .Include(p => p.SchoolProfession)
        .FirstOrDefaultAsync(m => m.PersonId == id);
    if (personModel == null)
    {
        return NotFound();
    }

    return View(personModel);
}

// POST: Person/Delete/5
[HttpPost, ActionName("Delete")]
[ValidateAntiForgeryToken]
public async Task<IActionResult> DeleteConfirmed(int id)
{
    var personModel = await _context.Persons.FindAsync(id);
    _context.Persons.Remove(personModel);
    await _context.SaveChangesAsync();
    return RedirectToAction(nameof(Index));
}
```

4.2.4 Ustawienia ogólne - fragment kodu.

```
// This method gets called by the runtime. Use this method to add services to the container.
public void ConfigureServices(IServiceCollection services)
{
    services.Configure<CookiePolicyOptions>(options =>
    {
        // This lambda determines whether user consent for non-essential cookies is needed for a given request.
        options.CheckConsentNeeded = context => true;
        options.MinimumSameSitePolicy = SameSiteMode.None;
    });

    services.AddDbContext<AppDbContext>(options => options
        .UseSqlServer(Configuration.GetConnectionString("DbConnS")));
}
```

```
public class Startup
{
    public Startup(IConfiguration configuration)
    {
        Configuration = configuration;
    }

    public IConfiguration Configuration { get; }

    public void ConfigureServices(IServiceCollection services)
    {
        services.Configure<CookiePolicyOptions>(options =>
        {

            options.CheckConsentNeeded = context => true;
            options.MinimumSameSitePolicy = SameSiteMode.None;
        });

        services.AddDbContext<AppDbContext>(options => options
            .UseSqlServer(Configuration.GetConnectionString("DbConnS")));
    }

    services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version_2_1);

}

public void Configure(IApplicationBuilder app, IHostingEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }
    else
    {
        app.UseExceptionHandler("/Home/Error");
        app.UseHsts();
    }
    app.AddEfDiagrams<AppDbContext>();
    app.UseHttpsRedirection();
    app.UseStaticFiles();
    app.UseCookiePolicy();

    app.UseMvc(routes =>
    {
        routes.MapRoute(
            name: "default",
            template: "{controller=Home}/{action=Index}/{id?}");
    });
}
```

4.2.5 Edycja ocen – fragment kodu.

```
// POST: Grade/Edit/5
// To protect from overposting attacks, please 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("GradeId,GradeDate,GradeValue,CourseId,PersonId")] GradeModel gradeModel)
{
    if (id != gradeModel.GradeId)
    {
        return NotFound();
    }

    if (ModelState.IsValid)
    {
        try
        {
            _context.Update(gradeModel);
            await _context.SaveChangesAsync();
        }
        catch (DbUpdateConcurrencyException)
        {
            if (!GradeModelExists(gradeModel.GradeId))
            {
                return NotFound();
            }
            else
            {
                throw;
            }
        }
        return RedirectToAction(nameof(Index));
    }
    ViewData["CourseId"] = new SelectList(_context.Courses, "CourseId", "CourseId", gradeModel.CourseId);
    ViewData["PersonId"] = new SelectList(_context.Persons, "PersonId", "FirstName", gradeModel.PersonId);
    return View(gradeModel);
}
```

5. Podsumowanie.

Końcowo nasza aplikacja działa zgodnie ze wstępными założeniami i może pełnić funkcję aplikacji wspierającej pracę nauczycieli zarówno dla małej jak i zespołu szkół. W przyszłości można aplikacji do aplikacji dodać np. możliwości obliczania średnich ważonych oraz rozwijać program aby był bardziej zautomatyzowany. Celem naszego projektu było stworzenie prostej bazy danych połączonej z łatwą w obsłudze aplikacją i uważały, że to założenie zostało w pełni zrealizowane.