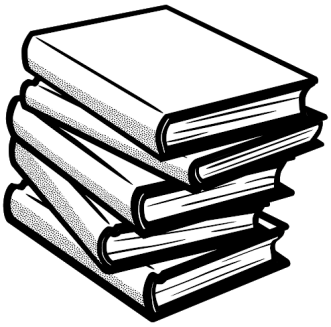
Projektowanie i programowanie obiektowe

Projektowanie Systemów Baz Danych



Students Book

Projekt - dziennik nauczyciela

Łukasz Kuczma, Konrad Szatanek
Informatyka, rok II, semestr IV, n2PAM

czerwiec 2020

Spis treści

Rozdział 1

Założenia i koncepcja

1.1 Cel projektu aplikacji bazy danych

Wstępna koncepcja projektu opierała się na stworzeniu aplikacji wspierającej pracę nauczycieli szkół podstawowych. Aplikacja ta ma charakter dziennika ocen oraz bazy danych zawierającej informacje o uczniach, nauczycielach oraz opiekunach. Aplikacja umożliwia wystawianie ocen oraz śledzenie postępów w nauce.

Stworzony przez nas program jest tak zwanym systemem internetowym działającym pod postacią strony internetowej, co czyni dostęp do niego niebywale łatwym. Użytkownik nie musi instalować żadnego dodatkowego oprogramowania aby obsługiwać system. Wystarczy przeglądarka internetowa z dostępem do sieci internet.

1.2 Założenia projektu

Finalnie aplikacja obsługuje cztery rodzaje użytkowników z różnymi poziomami dostępu do funkcjonalności. W bazie danych funkcjonuje tabela z danymi personalnymi wszystkich osób. Osoby te przypisane mają odpowiednie role, funkcjonujące w systemie jako profesje: administrator, nauczyciel, opiekun oraz uczeń.

Osobą z najszerszą gamą uprawnień jest **administrator** systemu, który czuwa nad strukturą danych oraz jest odpowiedzialny za rejestracje i nadawanie roli użytkownikom.

Użytkownik o profesji **nauczyciel** ma przede wszystkim możliwość wystawiania ocen oraz posiada wgląd w dane klas, nauczanych przedmiotów, uczniów i ich opiekunów.

Uczeń może posiadać wielu **opiekunów**, którzy maja wgląd w oceny swoich podopiecznych. Dzięki temu mogą kontrolować postępy w nauce. **Uczeń** to użytkownik, który ma dostęp (podgląd) do dzienniczka swoich ocen oraz swoich danych personalnych.

1.3 Technologia

Aplikacja działa w oparciu o technologie *ASP.NET Core 2.1*, a projekt został zrealizowany w *MS Visual Studio 2017* przy użyciu framework'u *Entity Framework 6.0* do mapowania relacyjno-obiektowego. Zastosowany wzorzec projektowy to MVC (model, widok, kontroler)
Aplikacja będzie się łączyć z lokalną bazą danych - *MS SQLServer*. Interfejs użytkownika to przede wszystkim *HTML5, CSS* oraz framework *Bootstrap 3.4*.

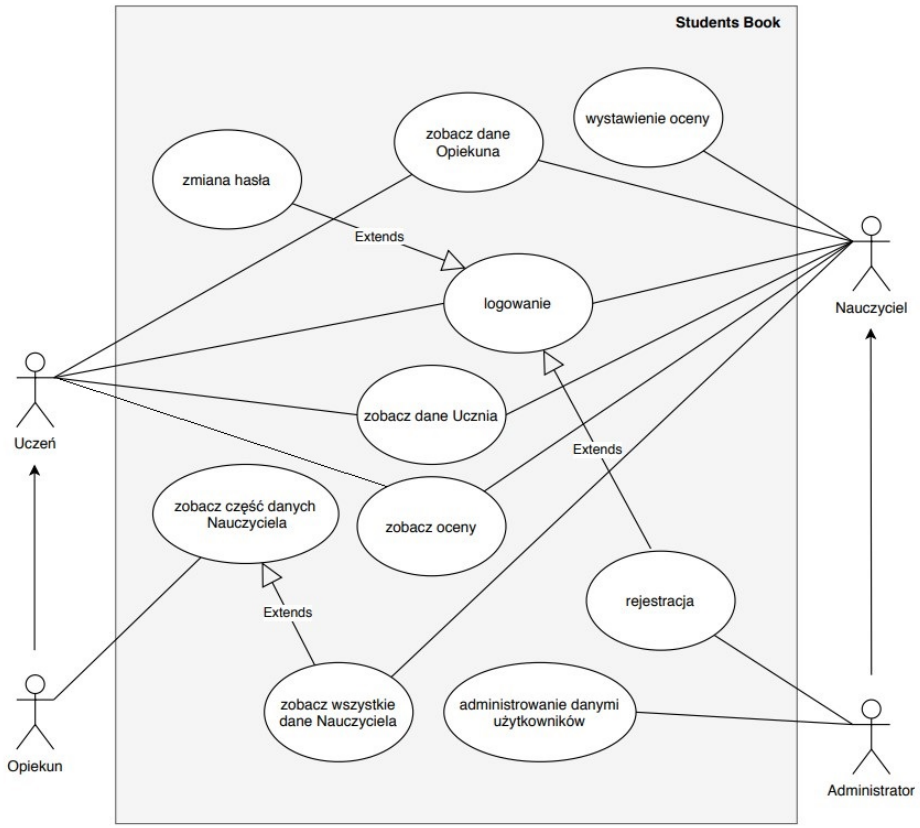
1.4 Podsumowanie

Celem naszego projektu jest stworzenie aplikacji ułatwiającej pracę pedagogów szkolnych poprzez prosty mechanizm wystawiania ocen, przejrzysty wgląd w proces postępów ucznia oraz ewentualna korekta osiągniętych wyników. Dodatkową funkcjonalnością jest także dostęp do danych opiekunów oraz środowiska nauczycieli. Aplikacja powinna być łatwa w obsłudze.
Dostępność zapewnia przeglądarka www z dostępem do internetu.

Rozdział 2

Przypadki użycia - diagram

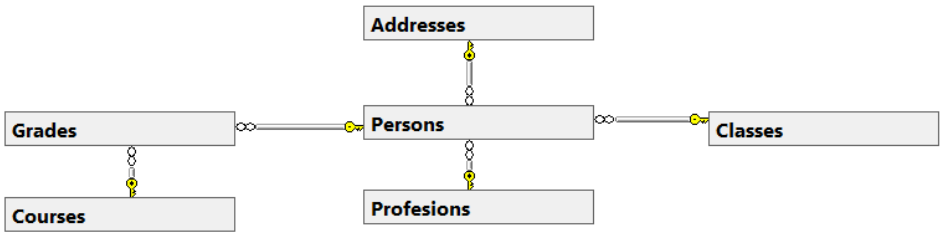
Poniższy diagram prezentuję przypadki użycia systemu wszystkich użytkowników. Opiekun dziedziczy funkcjonalności po Uczniu, dodatkowo ma wgląd w częściowe dane Nauczyciela. Administrator dziedziczy funkcjonalności po Nauczycielu, ma również możliwość rejestracji użytkowników oraz administrowanie ich danymi.



Rozdział 3

Model danych

3.1 Konceptualny model danych - ERD



3.2 Mapowanie obiektowo-relacyjne (ORM). Model danych

Modele stworzone w postaci klas C# są przenoszone w procesie mapowania do bazy danych. Wcześnie konfigurowana jest klasa *DbContext* - ustalone są również nazwy instancji w bazie. Proces mapowania odbywa się za pośrednictwem migracji*(ASP.NET Core 2.1 - Entity Framework 6.0)*. Poniżej widoczna jest klasa *AppDbContext* i odwołania do tabeli w bazie danych. Następnie zaimplementowane są modele: Adres, Przedmiot, Ocena, Osoba, Profesja, Klasa.

3.2.1 Plik kontekstu - konfiguracja

Konstruktor klasy *AppDbContext* (dziedziczącej po *DbContext*) i odwołania do bazy danych.

```
1
2  public AppDbContext(DbContextOptions<AppDbContext> options) :
       base(options)
3          {
4          }
5          public DbSet<PersonModel> Persons { get; set; }
6          public DbSet<AddressModel> Addresses { get; set; }
7          public DbSet<ProfessionModel> Professions { get; set;
               }
8
9          public DbSet<SchoolClassModel> Classes { get; set; }
10         public DbSet<GradeModel> Grades { get; set; }
11         public DbSet<CourseModel> Courses { get; set; }
12
13         protected override void OnModelCreating(ModelBuilder
               modelBuilder)
14         {
15             modelBuilder.Entity<PersonModel>().ToTable("
                   Persons");
16             modelBuilder.Entity<AddressModel>().ToTable("
                   Addresses");
17             modelBuilder.Entity<ProfessionModel>().ToTable("
                   Profesions");
18             modelBuilder.Entity<SchoolClassModel>().ToTable("
                   Classes");
19             modelBuilder.Entity<GradeModel>().ToTable("Grades"
                   );
20             modelBuilder.Entity<CourseModel>().ToTable("
                   Courses");
21         }
```

3.2.2 Adres

Model zawierający dane adresowe uzytkownika.

```
1
2  public class AddressModel
3      {
4          [Key]
5          public int AddressId { get; set; }
6
7          [Required(ErrorMessage = "To pole jest wymagane")]
8          [DisplayName("Ulica")]
```

```
9           public string Street { get; set; }
10
11          [Required(ErrorMessage = "To pole jest wymagane")]
12          [DisplayName("Nr domu")]
13          public string HomeNr { get; set; }
14
15          [Required(ErrorMessage = "To pole jest wymagane")]
16          [DisplayName("Miejscowość")]
17          public string City { get; set; }
18
19          [Required(ErrorMessage = "To pole jest wymagane")]
20          [DisplayName("Kod Pocztowy")]
21          public string ZipCode { get; set; }
22
23          [Required(ErrorMessage = "To pole jest wymagane")]
24          [DisplayName("Kraj")]
25          public string Country { get; set; }
26
27          [Display(Name = "Pełny adres")]
28          public string FullAddress
29          {
30              get { return Street +" "+ HomeNr +" "+ ZipCode +"
                    "+ City;}
31          }
32
33      }
```

3.2.3 Przedmiot

Dziedzina, która zostaje poddana ocenie.

```
1
2   public class CourseModel
3       {
4           [Key]
5           public int CourseId { get; set; }
6
7           [Display(Name = "Nazwa przedmiotu")]
8           public string CourseName { get; set; }
9
10          public virtual ICollection<GradeModel> Grades { get;
                set; }
11
12      }
```

3.2.4 Ocena

Model pełni rolę dziennika ocen.

```
1
2  public class GradeModel
3      {
4          [Key]
5          public int GradeId { get; set; }
6
7          [Required(ErrorMessage = "To pole jest wymagane")]
8          [DisplayName("Data wystawienia")]
9          public DateTime GradeDate { get; set; }
10
11         [Required(ErrorMessage = "To pole jest wymagane")]
12         [DisplayName("Ocena")]
13         public int GradeValue { get; set; }
14
15         public int CourseId { get; set; }
16
17         public virtual CourseModel Course { get; set; }
18
19         public int PersonId { get; set; }
20
21         public virtual PersonModel Student { get; set; }
22
23
24     }
```

3.2.5 Osoba

Najważniejszy model danych, zawiera dane osobowe użytkownika, oraz odniesienia do innych modeli (tabel).

```
1
2  public class PersonModel
3      {
4          [Key]
5          public int PersonId { get; set; }
6
7          [Required(ErrorMessage = "To pole jest wymagane")]
8          [DisplayName("Imię")]
9          public string FirstName { get; set; }
10
11         [Required(ErrorMessage = "To pole jest wymagane")]
12         [DisplayName("Nazwisko")]
13         public string LastName { get; set; }
```

```
14
15          [Display(Name = "Nazwisko i Imię")]
16          public string FullName
17          {
18              get { return LastName + " " + FirstName; }
19          }
20
21          [DisplayName("Numer telefonu")]
22          public string PhoneNumber { get; set; }
23
24          [DisplayFormat(DataFormatString = "{0:dd-MM-yyyy}"
25                    ,ApplyFormatInEditMode = true)]
26          [Required(ErrorMessage = "To pole jest wymagane")]
27          [Display(Name = "Data urodzenia")]
28          public DateTime BirthDate { get; set; }
29
30          public int AddressId { get; set; }
31
32          public virtual AddressModel Address { get; set; }
33
34          public int ProfessionId { get; set; }
35
36          public virtual ProfessionModel SchoolProfession { get;
                set; }
37
38          [ForeignKey("SchoolClassModel")]
39          public int ClassId { get; set; }
40
41          public virtual SchoolClassModel Class { get; set; }
42
43          public virtual ICollection<GradeModel> Grades { get;
                set; }
44
45      }
```

3.2.6 Profesja

Profesja w szkole oraz rola użytkownika w systemie. Na podstawie tej wartości przydzielane są funkcjonalności w systemie.

```
1
2 public class ProfessionModel
3     {
4         [Key]
5         public int ProfessionId { get; set; }
6
7         [Required(ErrorMessage = "To pole jest wymagane")]
8         [DisplayName("Profesja")]
```

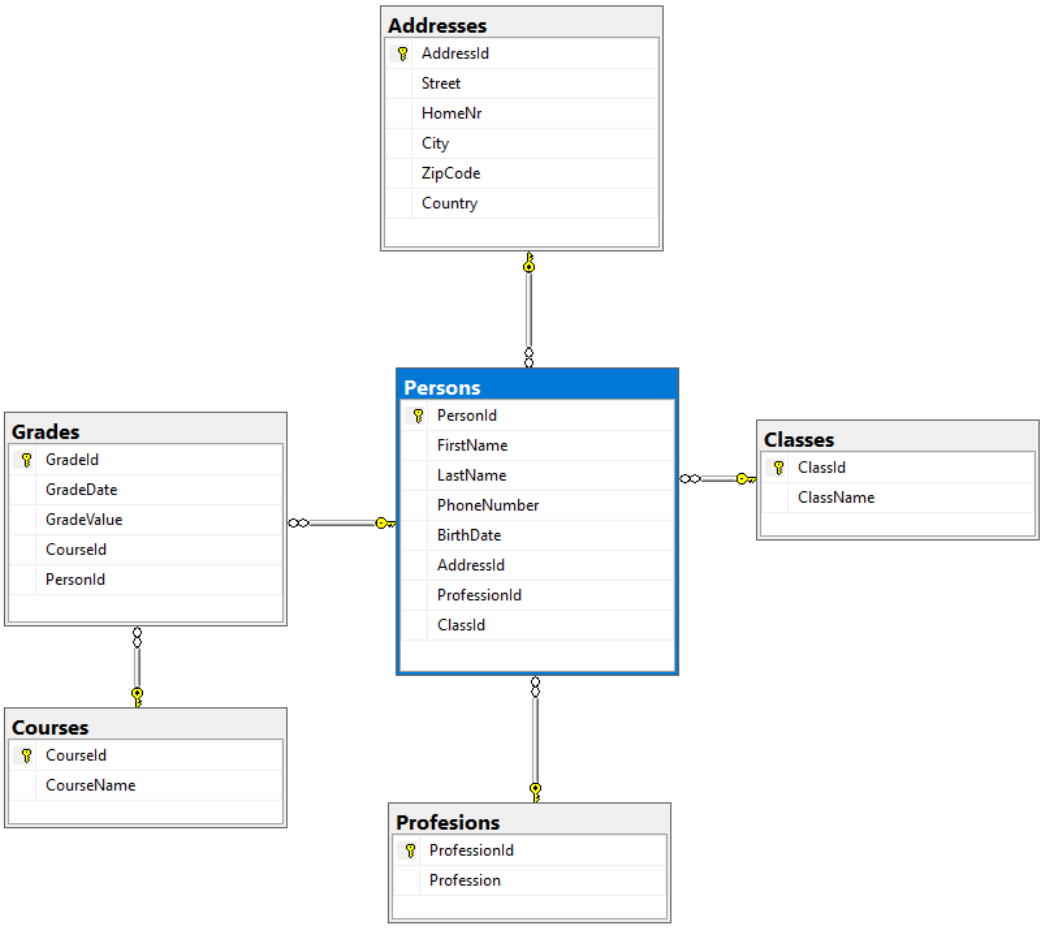
```
9           public string Profession { get; set; }
10
11          public virtual ICollection<PersonModel> Persons { get;
                set; }
12
13      }
```

3.2.7 Klasa

Model klasy szkolnej.

```
1
2   public class SchoolClassModel
3       {
4           [Key]
5           public int ClassId { get; set; }
6
7           [Required(ErrorMessage = "To pole jest wymagane")]
8           [DisplayName("Klasa")]
9           public string ClassName { get; set; }
10
11          public virtual ICollection<PersonModel> ClassStudents
                { get; set; }
12
13      }
```

3.3 Diagram klas



Rozdział 4

Funkcjonalność

4.1 Użytkownicy systemu

Głównymi użytkownikami systemu będą osoby, które podzielić możemy na cztery kategorie, ściśle związane z pełnionymi profesjami:

- Administrator
- Nauczyciel
- Opiekun
- Uczeń

4.2 Funkcjonalności systemu z podziałem na zakresy uprawnień użytkowników.

4.2.1 Administrator

Użytkownik w roli Administrator posiada pełen dostęp do wszystkich funkcjonalności systemu.

1. Rejestracja użytkowników.
2. Logowanie do systemu
3. Zmiana hasła
4. Przeglądanie wszystkich danych systemu.
5. Dodawanie, edycja, usuwanie danych wszystkich użytkowników.

6. Dodawanie, edycja, usuwanie danych adresowych.
7. Dodawanie, edycja, usuwanie danych profesji (ról w systemie).
8. Dodawanie, edycja, usuwanie danych ocen.
9. Dodawanie, edycja, usuwanie danych klas.
10. Dodawanie, edycja, usuwanie danych przedmiotów.

4.2.2 Nauczyciel

Użytkownik o profesji Nauczyciel posiada dostęp do ograniczonej liczby funkcjonalności.

1. Logowanie do systemu
2. Zmiana hasła
3. Przeglądanie i edycja własnych danych osobowych.
4. Przeglądanie danych osobowych użytkowników o profesji Uczeń.
5. Przeglądanie danych osobowych użytkowników o profesji Opiekun.
6. Przeglądanie danych klas.
7. Przeglądanie danych przedmiotów.
8. Przeglądanie, dodawanie, usuwanie danych ocen.

4.2.3 Opiekun

Użytkownik o profesji Opiekun posiada dostęp do ograniczonej liczby funkcjonalności.

1. Logowanie do systemu
2. Zmiana hasła
3. Przeglądanie, edycja własnych danych osobowych.
4. Przeglądanie, edycja danych osobowych przypisanego użytkownika o profesji Uczeń.
5. Przeglądanie danych ocen przypisanego użytkownika o profesji Uczeń.

6. Przeglądanie ograniczonych danych klasy (klasa przypisanego użytkownika Uczeń).

7. Przeglądanie ograniczonych danych osobowych użytkowników o profesji Nauczyciel.

4.2.4 Uczeń

Użytkownik o profesji Uczeń posiada dostęp do ograniczonej liczby funkcjonalności.

1. Logowanie do systemu

2. Zmiana hasła

3. Przeglądanie własnych danych ocen.

4. Przeglądanie własnych danych adresowych.

5. Przeglądanie własnych danych osobowych.

4.3 Opis przypadków użycia

Użytkownicy, zgodnie ze swoimi uprawnieniami, mogła pobierać dane z bazy danych oraz je dodawać lub edytować.

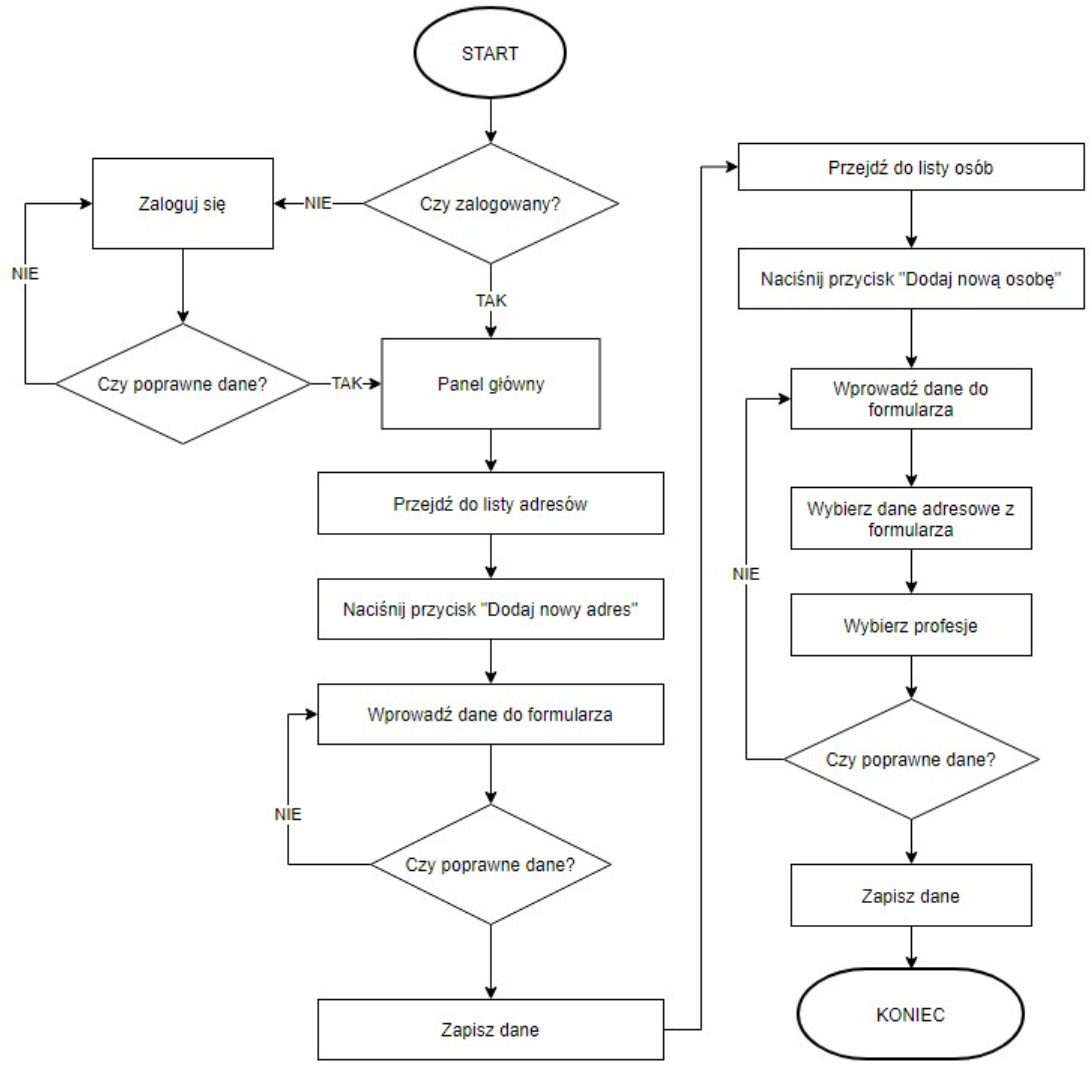
Administrator posiada najwyższe uprawnienia systemowe. Jest osobą czuwającą nad spójnością i poprawnością danych - ma możliwość rejestracji nowych użytkowników systemu oraz edycji i usuwania ich danych. Ponad to dba o to, aby dane w systemie odzwierciedlały rzeczywisty stan strukturalny przedmiotów i klas szkoły. Jest też odpowiedzialny za wprowadzone do bazy dane personalne wszystkich użytkowników.

Nauczyciel jest użytkownikiem, który zaraz po Administratorze posiada dostęp do największej liczby funkcjonalności systemu. Ma wgląd w dane osobowe i adresowe wszystkich uczniów oraz Opiekunów. Nauczyciel ma również dostęp do danych wszystkich klas i przedmiotów.Jego najważniejszą funkcjonalnością jest dodawanie, edytowanie i usuwanie ocen.

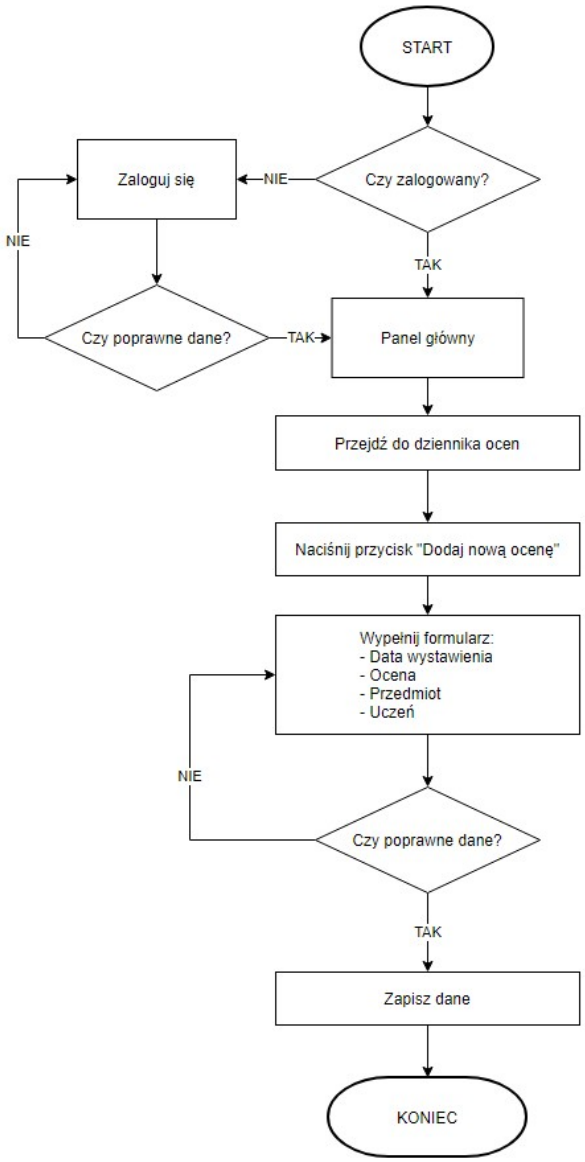
Uczeń posiada najniższe uprawnienia systemowe a zakres jego funkcjonalności sprowadza się do przeglądania własnych ocen oraz danych osobowych i adresowych.

4.4 Diagram czynności

4.4.1 Diagram – dodawanie nowego ucznia



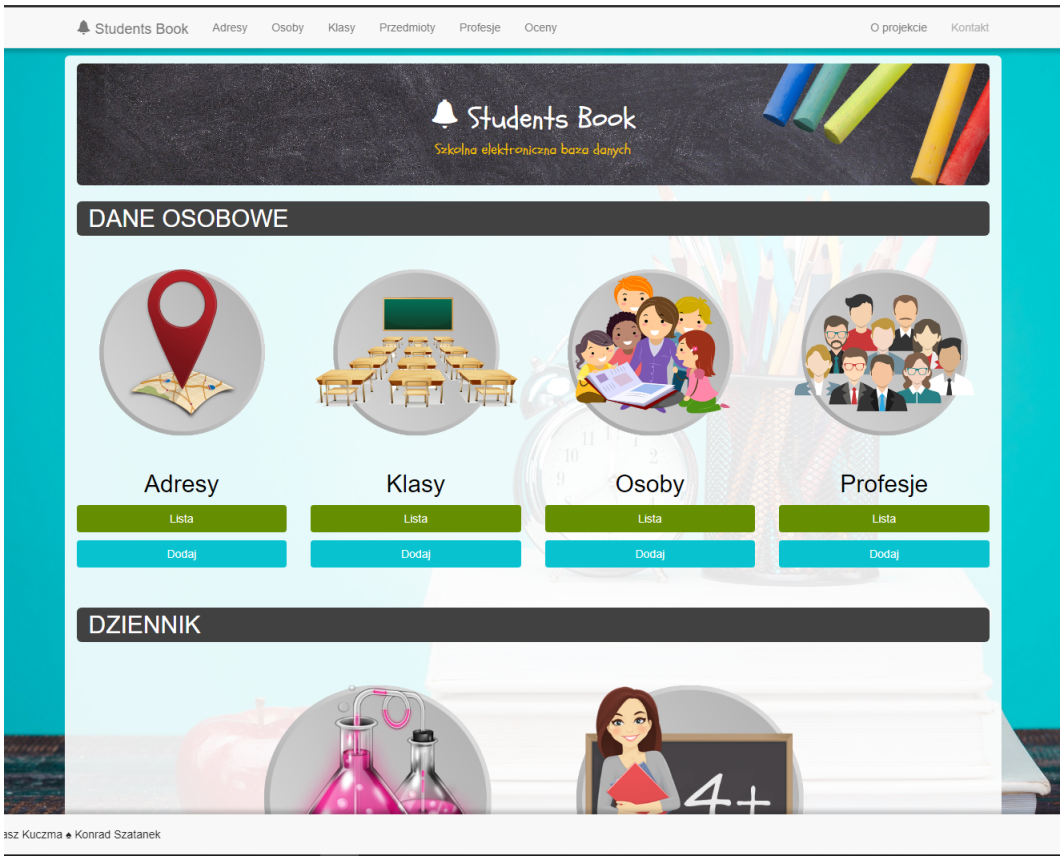
4.4.2 Diagram – wystawienie oceny



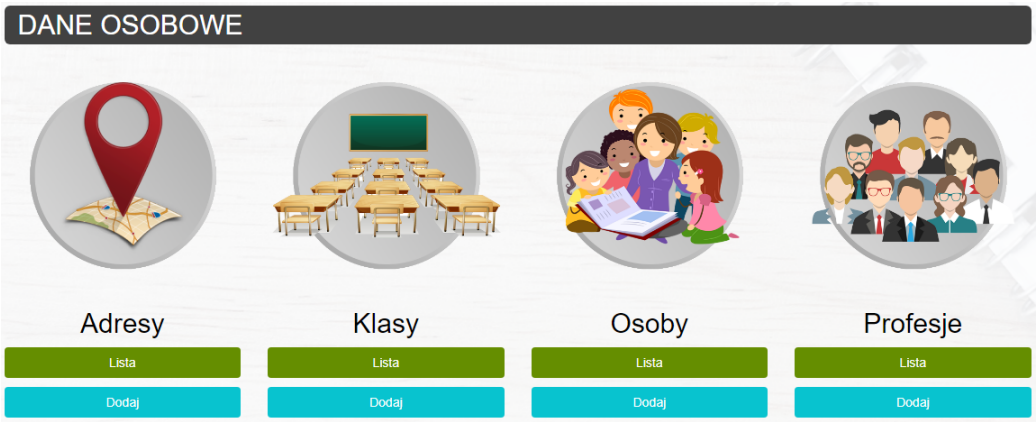
Rozdział 5

Model interfejsu użytkownika

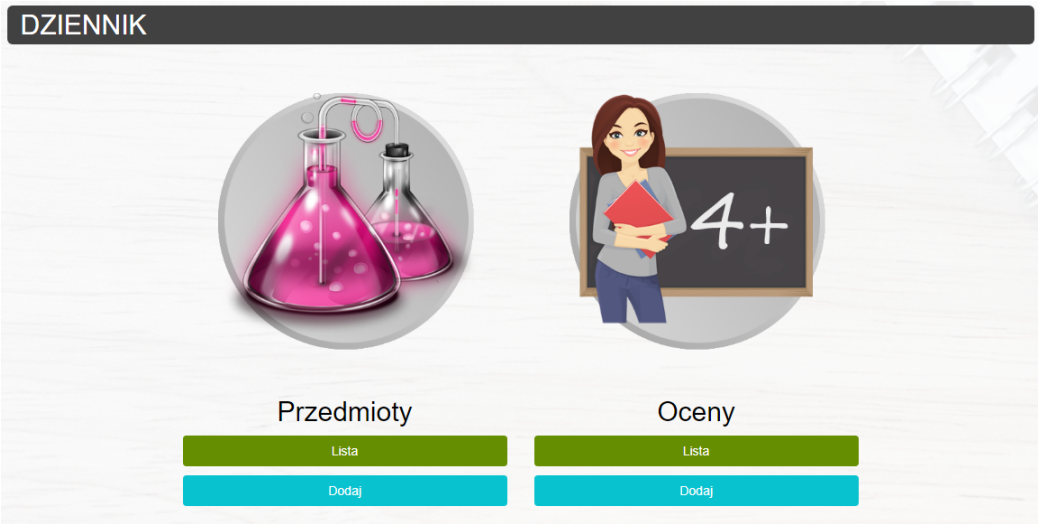
5.1 Panel główny.



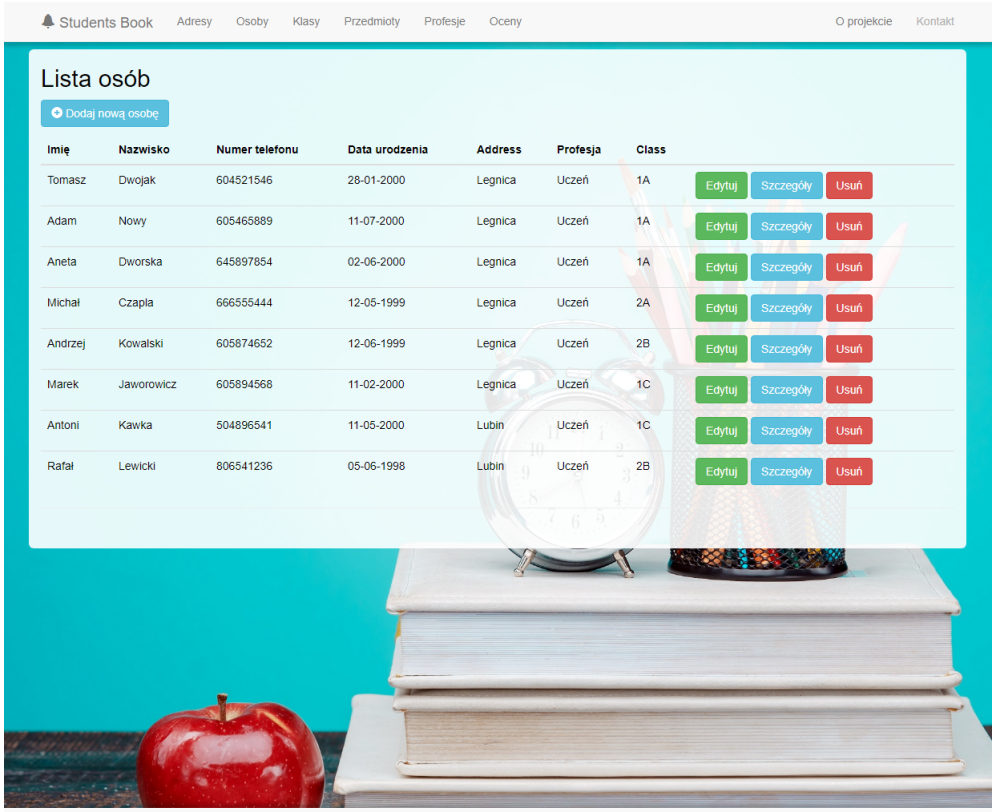
5.2 Dane osobowe.



5.3 Dziennik – przedmioty, oceny.

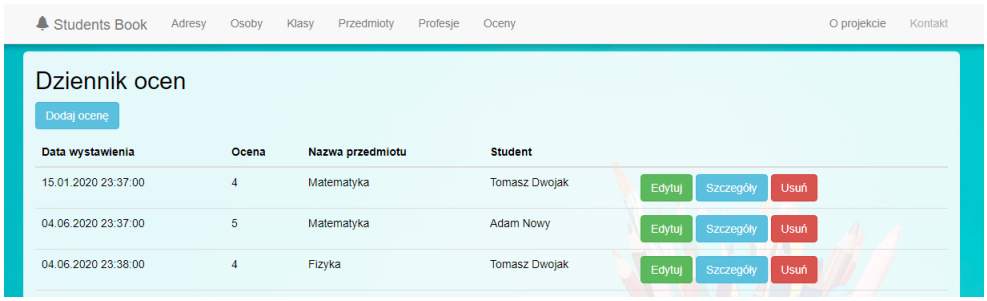


5.4 Lista osób.



ma ♠ Konrad Szatanek

5.5 Formularz wprowadzania nowej osoby.

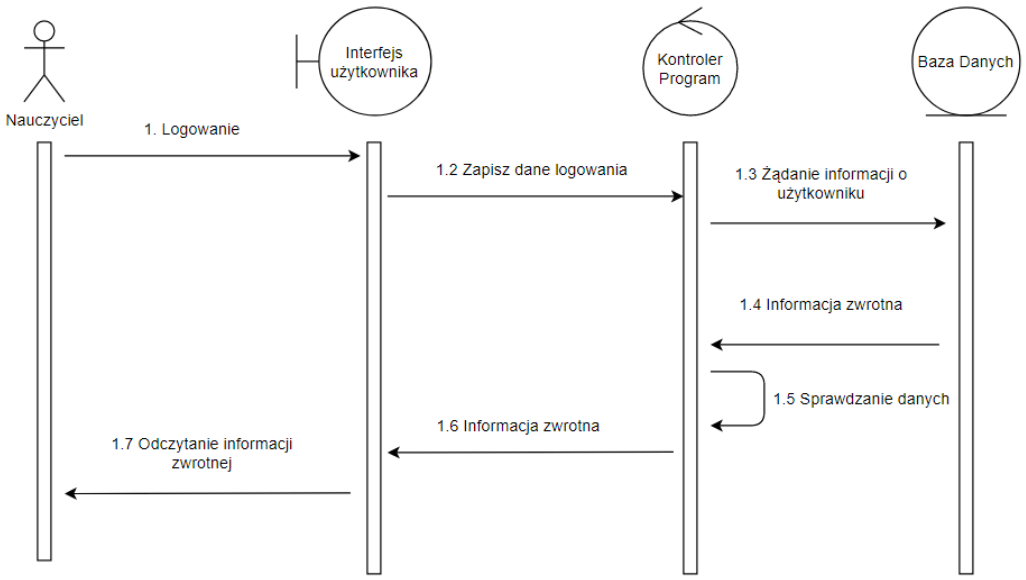


Rozdział 6

Diagram UML sekwencji

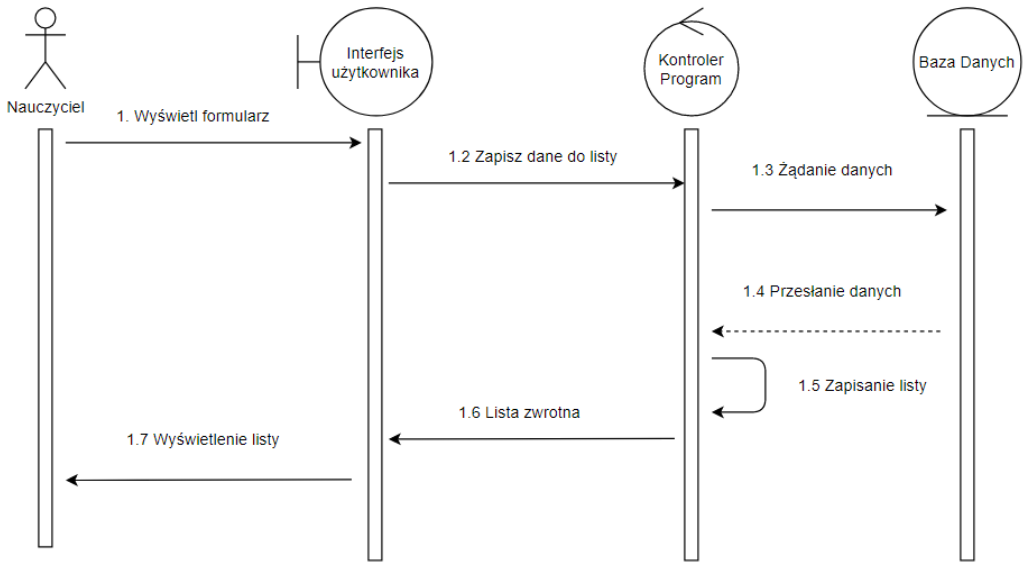
6.1 Logowanie

Poniższy diagram przedstawia proces logowania użytkownika oraz komunikację między obiektami systemu: interfejsem użytkownika, kontrolerem i bazą danych.



6.2 Wyświetlenie listy uczniów

Diagram o podobnej strukturze - prezentuje proces wyświetlania listy uczniów. Zostaje skierowane polecenie wyświetlenia formularza. Interfejs oczekuje na listę. Kontroler kieruje zapytanie do bazy danych, a następnie otrzymane informacje zapisuje w postaci listy i przekazuje do interfejsu użytkownika.



Rozdział 7

Kod źródłowy

7.1 Połączenie z bazą danych

7.1.1 Plik konfiguracyjny *appsettings.json*

W pliku konfigurowany jest *Connection String*, czyli łańcuch informacji niezbędnych do połączenia z serwerem (nazwa serwera, nazwa bazy danych, rodzaj połączenia, login, hasło, itd). W tym przypadku jest to lokalny serwer i baza danych o nazwie *SchoolBook* (MS SQLServer).W pliku może znajdować się wiele łańcuchów połączeń.

```
1  appsettings.json
2
3  {
4    "ConnectionStrings": {
5      "DbConnS": "Server=(LocalDb)\\MSSQLLocalDB;Database=
           SchoolBook;Trusted_Connection=True;
           MultipleActiveResultSets=True"
6    },
7
8
9    "Logging": {
10     "LogLevel": {
11       "Default": "Warning"
12     }
13   },
14   "AllowedHosts": "*"
15 }
```

7.1.2 Dodanie serwisu obsługi bazy danych (*Startup.cs*)

W pliku *Startup.cs* konfigurowana jest aplikacja. Do serwisów aplikacji zostaje dodana klasa *DbContext* odpowiedzialna za komunikowanie się z bazą danych - w tym przypadku MS SQLServer. Połączenie jest możliwe dzięki informacją z łańcucha *Connection String*.

```
1
2   public void ConfigureServices(IServiceCollection services)
3         {
4             services.Configure<CookiePolicyOptions>(options =>
5             {
6                 // This lambda determines whether user consent
                      for non-essential cookies is needed for a
                      given request.
7                 options.CheckConsentNeeded = context => true;
8                 options.MinimumSameSitePolicy = SameSiteMode.
                      None;
9             });
10
11            services.AddDbContext<AppDbContext>(options =>
                  options
12                .UseSqlServer(Configuration.
                      GetConnectionString("DbConnS")));
13
14            services.AddMvc().SetCompatibilityVersion(
                  CompatibilityVersion.Version_2_1);
15
16        }
```

7.2 Zapytania łączone do bazy danych

Zapytania są realizowane za pomącą składni biblioteki *LINQ* i wyrażeń lambda. Użyta jest klasa *DbContex*, która pobiera dane z bazy, a następnie zapisuje je w postaci listy. W tym przypadku do listy pobierane są dane z tabeli *Oceny, Przedmioty, Osoby*. Jest to akcja wyświetlania wszystkich wystawionych ocen.

W wierszu 9 rozpoczyna się akcja podglądu informacji o konkretnej ocenie. Zapytanie jest podobne, jednak zostaje uwarunkowane kluczem id. W efekcie zostaje zwrócony jeden rekord.

```
1
2   public async Task<IActionResult> Index()
3         {
4             var appDbContext = _context.Grades.Include(g => g.
                  Course).Include(g => g.Student);
5             return View(await appDbContext.ToListAsync());
6         }
7
8         // GET: Grade/Details/5
9         public async Task<IActionResult> Details(int? id)
10        {
11            if (id == null)
12            {
13                return NotFound();
14            }
15
16            var gradeModel = await _context.Grades
17                .Include(g => g.Course)
18                .Include(g => g.Student)
19                .FirstOrDefaultAsync(m => m.GradeId == id);
20            if (gradeModel == null)
21            {
22                return NotFound();
23            }
24
25            return View(gradeModel);
26        }
```

7.3 Usuwanie informacji osobowych z bazy danych

Poniżej dwie akcje dotyczące usuwania rekordu z bazy danych. Metoda GET wczytująca dane oraz metoda POST usuwająca rekord.

Tworzony jest kontekst, poprzez zapytanie o osobę z konkretnym numerem id, następnie dane osoby usuwane są z bazy.

```
1
2    // GET: Person/Delete/5
3         public async Task<IActionResult> Delete(int? id)
4         {
5             if (id == null)
6             {
7                 return NotFound();
8             }
```

```
9
10              var personModel = await _context.Persons
11                  .Include(p => p.Address)
12                  .Include(p => p.Class)
13                  .Include(p => p.SchoolProfession)
14                  .FirstOrDefaultAsync(m => m.PersonId == id);
15              if (personModel == null)
16              {
17                  return NotFound();
18              }
19
20              return View(personModel);
21          }
22
23          // POST: Person/Delete/5
24          [HttpPost, ActionName("Delete")]
25          [ValidateAntiForgeryToken]
26          public async Task<IActionResult> DeleteConfirmed(int
                id)
27          {
28              var personModel = await _context.Persons.FindAsync
                    (id);
29              _context.Persons.Remove(personModel);
30              await _context.SaveChangesAsync();
31              return RedirectToAction(nameof(Index));
32          }
```

Rozdział 8

Podsumowanie

Celem naszego projektu było stworzenie prostej bazy danych połączonej z łatwą w obsłudze aplikacją i uważamy, że to założenie zostało w pełni zrealizowane. Aplikacja *Students Book* działa zgodnie ze wstępnymi założeniami i może pełnić funkcję aplikacji wspierającej pracę nauczycieli, zarówno dla małej szkoły jak i zespołu szkół.

W przyszłości można do aplikacji dodać np. możliwości obliczania średnich ważonych oraz usprawnić ją o nowe opcje takie jak mechanizmy filtrowania i sortowania danych oraz moduł wysyłania powiadomień o ocenach. Dodatkowo należałoby rozwijać program aby był bardziej zautomatyzowany.

Podczas tworzenia aplikacji poznaliśmy nową wiedzę, niezwykle istotną w przyszłych pracach projektowych. To doświadczenie pozwoli nam efektywniej zarządzać projektem, co sprawi, że realizacja przebiegnie szybciej, oraz uda nam się uniknąć popełnionych wcześniej błędów. Różnorodność technologi obsługiwanych w środowisku tworzenia projektu, zmuszała nas do ciągłego uzupełniania wiedzy związanej z tematem baz danych, programowania oraz tworzenia stron www.