

Dokumentacja projektu GradebookApp

autor: Norbert Fuk, 131431

Spis treści:

Przegląd Techniczny GradeBookApp

- Cel i Główne Funkcjonalności
- Architektura Ról Użytkowników
- Przegląd Architektury Systemu
- Model Podstawowych Encji
- Stos Technologiczny i Konfiguracja
- Funkcje Środowiska Deweloperskiego

Architektura Aplikacji i Uruchomienie

- Przegląd architektury aplikacji
- Proces konfiguracji uruchomienia
- Rejestracja usług i wstrzykiwanie zależności
- Usługi logiki biznesowej**
- Konfiguracja bazy danych i wzorzec fabryki
- Konfiguracja uwierzytelniania i tożsamości
- Konfiguracja potoku middleware

Interfejsy użytkownika

- Przegląd architektury UI
- Główne komponenty i ich trasy
- Przepływ nawigacji komponentów
- Interfejs ucznia (Student Flow)
- Interfejs nauczyciela (Teacher Flow)
- Interfejs Admina (Admin Flow)

Komponenty interfejsu nauczyciela

- Główny dashboard nauczyciela (TeacherDashboard.razor)

Komponenty interfejsu ucznia

- Dashboard ucznia (StudentDashboard.razor)
- Widok frekwencji (StudentAttendance.razor)
- Dashboard administracyjny (AdminDashboard.razor)
- Wzorzec integracji z usługami
- Zarządzanie stanem komponentów

Backend Services & APIs

- Architektura warstwy usług
- Podstawowe komponenty serwisów
- Struktura kontrolerów API
- Wzorce punktów końcowych kontrolerów
- Wzorce obsługi błędów
- Wzorce autoryzacji i bezpieczeństwa
- Kontrola dostępu oparta na rolach

Warstwa serwisów

- Cel i zakres
- Zależności serwisów
- UserService
- SubjectService
- StudentClassService
- TeacherSubjectService
- Service Layer Data Flow
- Użycie kontekstu bazy danych
- Transformacja DTO
- Obsługa błędów

Kontrolery API

| |
|--|
| Cel i architektura |
| Request Flow Architecture |
| Autoryzacja oparta na rolach |
| Macierz autoryzacji |
| Podstawowe kontrolery API |
| StudentClassesController |
| ClassesController |
| SubjectsController |
| TeacherSubjectsController |
| Wzorce projektowe API |
| Standardowe wzorce odpowiedzi HTTP |
| Wzorzec użycia DTO |
| Strategia obsługi błędów |
| Integracja z warstwą serwisów |
| Wzorzec wstrzykiwania serwisów (Service Injection Pattern) |
| Architektura kontekstu bazy danych |
| Modele encji i relacje |
| Konfiguracja relacji encji |
| Obiekty transferu danych (DTO) |
| Struktura UserDto |
| Konfiguracja kontekstu bazy danych |
| Właściwości DbSet |
| Konfiguracja klucza złożonego |
| Seeding bazy danych |
| Struktura zasianych danych |
| Integracja z warstwą usług |
| Zarządzanie bazą danych i wysoka dostępność |
| Przegląd architektury |
| Zarządzanie konfiguracją |
| Monitorowanie stanu bazy danych |
| Mechanizm przełączania bazy danych |
| Przebieg procesu przełączania |
| Proces przełączania bazy danych |
| Synchronizacja bazy danych |
| Logika synchronizacji |
| Proces synchronizacji |
| Interfejs administracyjny |
| Komponenty interfejsu |
| Zarządzanie stanem |
| Obsługa błędów i logowanie |

Przegląd Techniczny GradeBookApp

Niniejszy dokument stanowi kompleksowy przegląd techniczny aplikacji GradeBookApp – internetowego systemu do zarządzania ocenami edukacyjnymi, stworzonego w oparciu o ASP.NET Core oraz Blazor Server. Aplikacja umożliwia, w zależności od roli, zarządzanie ocenami uczniów, śledzenie frekwencji oraz administrowanie danymi akademickimi w ramach wielu klas i przedmiotów.

Dla szczegółowych informacji na temat implementacji mechanizmów uwierzytelniania i autoryzacji zobacz sekcję **Authentication & Authorization**. Aby uzyskać dokumentację dotyczącą komponentów interfejsu użytkownika i przepływów pracy związanych z UI, przejdź do **User Interfaces**. W kwestii architektury serwisów backendowych oraz projektowania API, odsyłamy do **Backend Services & APIs**.

Cel i Główne Funkcjonalności

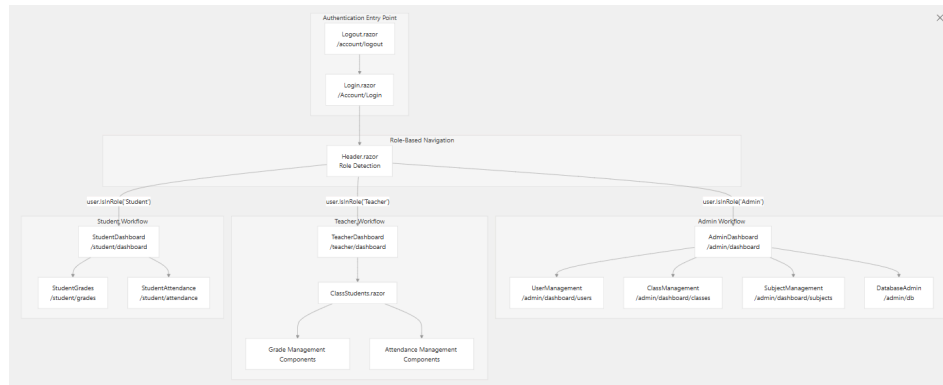
GradeBookApp pełni rolę scentralizowanej platformy dla placówek edukacyjnych, umożliwiając zarządzanie danymi akademickimi z wykorzystaniem trzech odmiennych ról użytkowników:

- **Administratorzy** – zarządzają użytkownikami, klasami, przedmiotami oraz operacjami na bazie danych
- **Nauczyciele** – wprowadzają oceny, śledzą obecności i przeglądają listy uczniów w klasach
- **Uczniowie** – mają dostęp w trybie tylko do odczytu do swoich ocen oraz zapisów frekwencji

System zapewnia wysoką dostępność dzięki mechanizmowi przełączania między bazą główną a zapasową, a także zawiera rozbudowany proces „data seeding” dla środowisk deweloperskich i testowych.

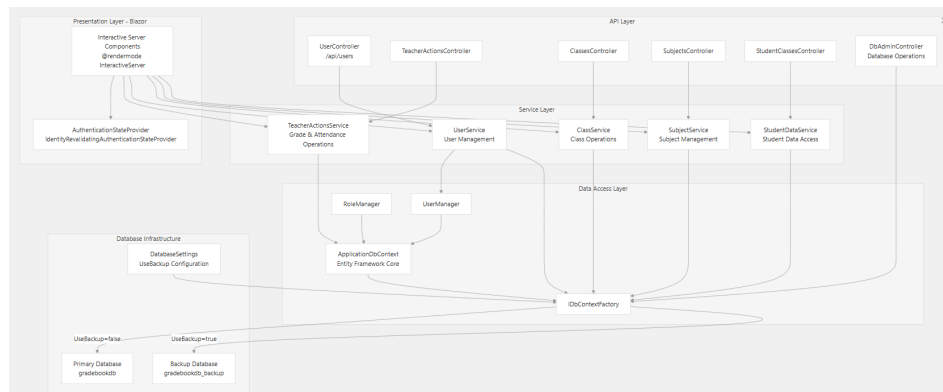
Architektura Ról Użytkowników

Aplikacja stosuje rygorystyczne podejście oparte na kontroli dostępu według ról (role-based access control), definiując odrębne ścieżki (workflows) oraz uprawnienia dla każdego typu użytkownika.



Przegląd Architektury Systemu

GradeBookApp wykorzystuje wielowarstwowy (layered) wzorec architektoniczny, wyraźnie oddzielając warstwę prezentacji, logiki biznesowej oraz dostępu do danych. Dzięki temu kod pozostaje modularny, łatwy w utrzymaniu oraz rozszerzalny.



Model Podstawowych Encji

Aplikacja zarządza danymi akademickimi w oparciu o dobrze zdefiniowany model encji, z jasnymi relacjami pomiędzy nimi:

| Encja | Przeznaczenie | Kluczowe Relacje |
|-------|---------------|------------------|
|-------|---------------|------------------|

| | | |
|------------------------|---|---|
| ApplicationUser | Przechowuje konta użytkowników wszystkich ról | Należy do klasy Class (jako uczeń), posiada wiele obiektów Grade |
| Class | Reprezentuje klasy lub grupy uczniów | Posiada wielu uczniów, należy do nauczyciela (wychowawcy) |
| Subject | Reprezentuje przedmioty (np. Matematyka, Język Angielski) | Posiada wiele obiektów Grade oraz Attendance |
| Grade | Reprezentuje poszczególne oceny | Łączy encje Student, Teacher oraz Subject |
| Attendance | Śledzi obecności uczniów w danym dniu | Łączy Student i Subject na podstawie daty |
| TeacherSubject | Przyporządkowuje nauczycieli do przedmiotów i klas | Stanowi połączenie wiele-do-wielu – bridge table pomiędzy nauczycielem, przedmiotem i klasą |

Taki model danych wspiera złożone scenariusze akademickie, w których nauczyciele mogą prowadzić wiele przedmiotów w różnych klasach, a uczniowie otrzymują oceny od różnych nauczycieli w ramach różnych przedmiotów.

Stos Technologiczny i Konfiguracja

Aplikacja została zbudowana w oparciu o następujące technologie:

- **Framework:** ASP.NET Core 8.0 wraz z Blazor Server
- **Baza Danych:** PostgreSQL w połączeniu z Entity Framework Core
- **Uwierzytelnianie:** ASP.NET Core Identity oparte na ciasteczkach (cookie-based authentication)
- **Interfejs UI:** Bootstrap 5 z interaktywnym renderowaniem po stronie serwera
- **API:** Kontrolery RESTful ze wbudowaną autoryzacją opartą na rolach

Do kluczowych elementów konfiguracji należą:

- Wzorzec Database Factory: IDbContextFactory<ApplicationDbContext> służący do zarządzania połączeniami z bazą danych
- Wysoka Dostępność: Mechanizm przełączania między bazą główną a zapasową realizowany przez właściwość DatabaseSettings.UseBackup
- Bezpieczeństwo Oparte na Rolach: Niestandardowa klasa CustomUserClaimsPrincipalFactory oraz filtry RequireRole zabezpieczające dostęp
- Data Seeding: Rozbudowany generator danych testowych zaimplementowany w metodzie DbSeeder.SeedAsync

W pliku Program.cs rejestrowane są wszystkie niezbędne serwisy, konfigurowane jest Identity, tworzone są połączenia z bazą danych, a także ustalona zostaje kolejność środków pośrednich (middleware) odpowiedzialnych za uwierzytelnianie i autoryzację.

Funkcje Środowiska Deweloperskiego

Aplikacja zawiera szereg funkcjonalności ułatwiających pracę programistom:

- **Automatyczne Migracje Bazy Danych:** Wykonywane przy starcie aplikacji z obsługą warunkowego seeding'u
- **Rozbudowane Dane Testowe:** Ponad 80 wstępnie załadowanych użytkowników we wszystkich rolach, z realistycznymi danymi akademickimi
- **UI Administracji Bazy Danych:** Możliwość przełączania między bazą główną a zapasową w czasie rzeczywistym, wraz z funkcją synchronizacji
- **Szczegółowa Obsługa Błędów:** Tryb „circuit” w Blazor Server, wyświetlający pełne szczegóły błędów w środowisku deweloperskim

- **Cookie-Aware HTTP Client:** Skonfigurowany klient HTTP do wykonywania uwierzytelnionych wywołań API

Proces seeding'u tworzy kompletne środowisko akademickie składające się z:

- 4 klas
- 15 przedmiotów
- 15 wyspecjalizowanych nauczycieli
- 80 uczniów

z realistycznymi danymi dotyczącymi ocen oraz obecności, obejmującymi bieżący okres akademicki.

Architektura Aplikacji i Uruchomienie

Istotne pliki źródłowe

Skupia się na konfiguracji hosta

ASP.NET Core, ustawieniu wstrzykiwania zależności, wzorcu fabryki bazy danych oraz procesie inicjalizacji aplikacji.

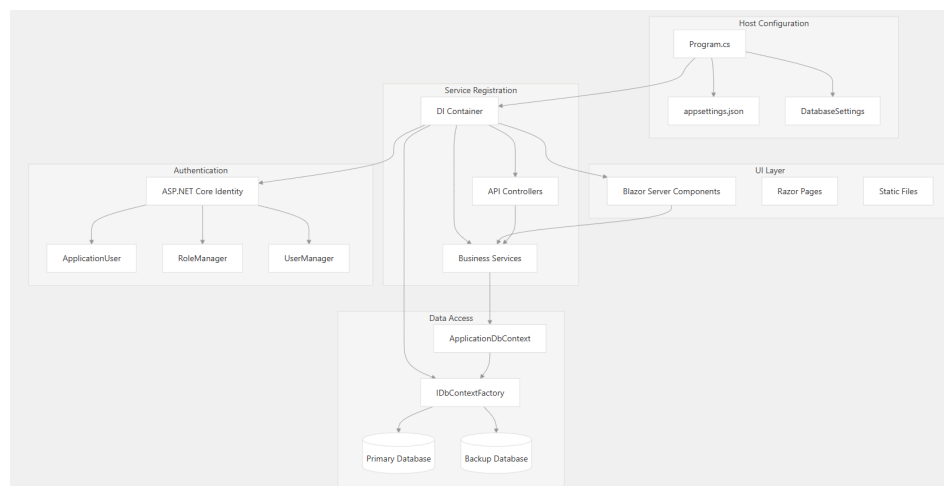
- Szczegóły dotyczące przepływu uwierzytelniania: Authentication & Authorization
- Szczegóły implementacji warstwy serwisów: Backend Services & APIs
- Konfiguracja encji w warstwie danych: Data Layer & Models

Źródła:

- GradeBookApp/Program.cs (linie 1–187)
- GradeBookApp/Data/ApplicationDbContext.cs (linie 1–87)

Przegląd architektury aplikacji

GradeBookApp wykorzystuje wielowarstwową architekturę ASP.NET Core z Blazor Server w warstwie UI, warstwą serwisową dla logiki biznesowej oraz Entity Framework Core do dostępu do danych. W aplikacji zastosowano wzorzec fabryki bazy danych, umożliwiający przełączanie między bazą główną a zapasową w czasie działania.



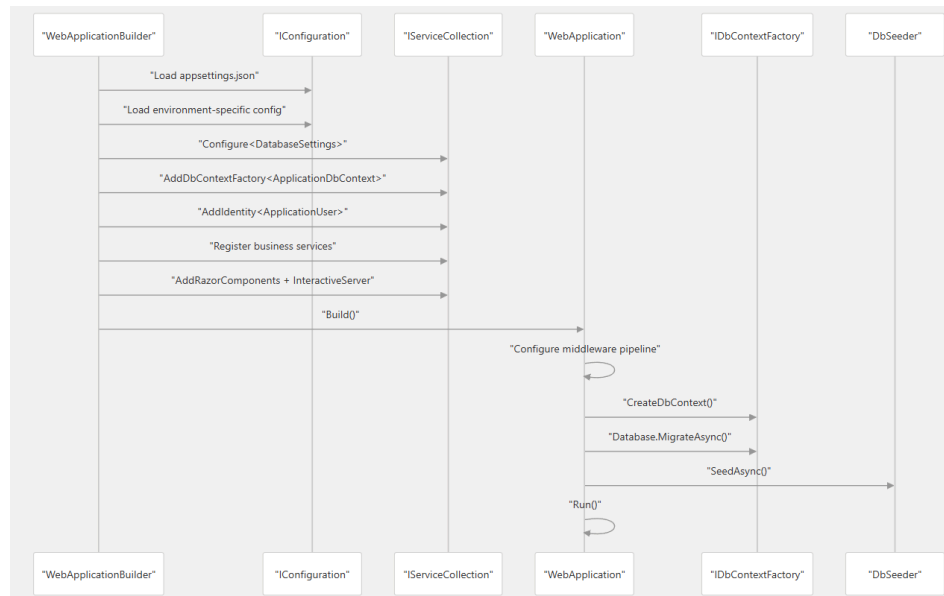
Źródła:

- GradeBookApp/Program.cs (linie 1–187)

- GradeBookApp/Data/ApplicationDbContext.cs (linie 1–87)

Proces konfiguracji uruchomienia

Rozruch aplikacji odbywa się według zdefiniowanego w Program.cs schematu: ładowanie konfiguracji, rejestracja usług, konfiguracja potoku middleware oraz inicjalizacja bazy danych.

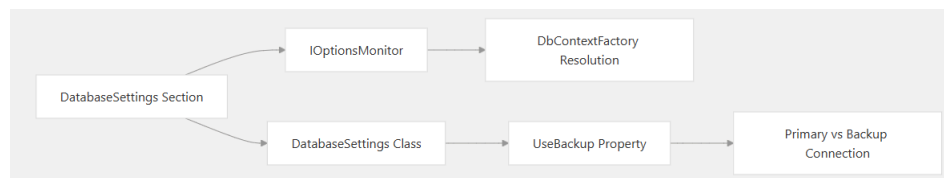


Źródła:

- GradeBookApp/Program.cs (linie 14–181)

Rejestracja usług i wstrzykiwanie zależności

W Program.cs usługi rejestrowane są w kilku grupach: konfiguracja, dostęp do danych, uwierzytelnianie, logika biznesowa oraz komponenty UI.



Usługi dostępu do danych

- Wzorzec fabryki kontekstu bazy danych pozwala na dynamiczne przełączanie połączeń:

Źródła:

- GradeBookApp/Program.cs (linie 28–41)
- GradeBookApp/Data/ApplicationDbContext.cs (linie 9–15)

Usługi logiki biznesowej

- Główne serwisy warstwy usług zarejestrowane jako zależności o zasięgu Scoped:

| Klasa usługi | Przeznaczenie | Odniesienie do linii |
|-----------------------|---|----------------------|
| ClassService | Operacje związane z zarządzaniem klasami | 69 |
| SubjectService | Operacje związane z zarządzaniem przedmiotami | 70 |
| TeacherSubjectService | Przypisywanie nauczycieli do przedmiotów i klas | 71 |
| StudentClassService | Przypisywanie uczniów do klas | 72 |
| UserService | Operacje zarządzania użytkownikami | 73 |
| StudentDataService | Operacje na danych uczniów | 74 |
| TeacherActionsService | Działania nauczyciela (oceny, frekwencja) | 75 |

Źródła:

- GradeBookApp/Program.cs (linie 68–76)
- GradeBookApp/Services/TeacherActionsService.cs (linie 9–16)
- GradeBookApp/Services/ClassService.cs (linie 9–16)

Konfiguracja bazy danych i wzorzec fabryki

Zaimplementowany wzorzec fabryki bazy danych pozwala na przełączanie między bazą główną a zapasową w czasie działania, w oparciu o ustawienia z appsettings.json.

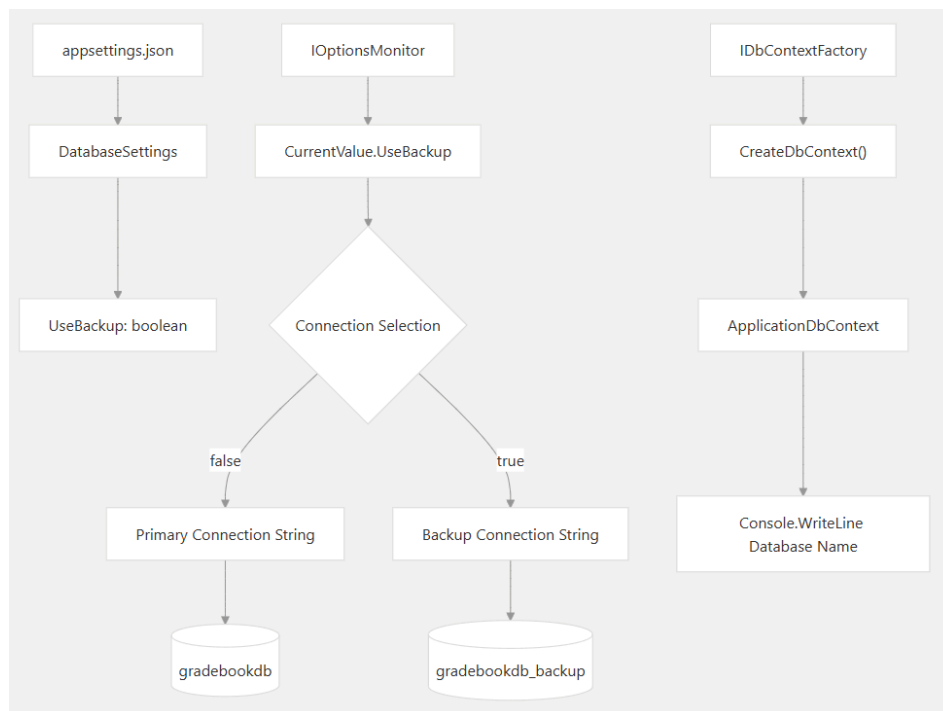
```
"UseBackup": false // wybór bazy głównej
"UseBackup": true  // wybór bazy zapasowej
```

Implementacja wzorca fabryki (linia 28–41 w Program.cs):

- Wykorzystuje IOptionsMonitor<DatabaseSettings> do pobrania aktualnych ustawień.
- Na podstawie flagi UseBackup wybiera odpowiedni ciąg połączenia.
- Loguje (Console.WriteLine) wybraną bazę w celach debugowania.
- Konfiguruje PostgreSQL za pomocą UseNpgsql().

Konstruktor ApplicationDbContext (linie 9–15 w ApplicationDbContext.cs):

Loguje z kolei nazwę podłączonej bazy w celu weryfikacji poprawności wyboru.



Źródła:

- GradeBookApp/Program.cs (linie 28–41, 183–186)
- GradeBookApp/Data/ApplicationDbContext.cs (linie 9–15)

Konfiguracja uwierzytelniania i tożsamości

Aplikacja korzysta z ASP.NET Core Identity z niestandardową encją ApplicationUser oraz autoryzacją opartą na rolach.



Szczegóły konfiguracji Identity (linie 48–86 w Program.cs):

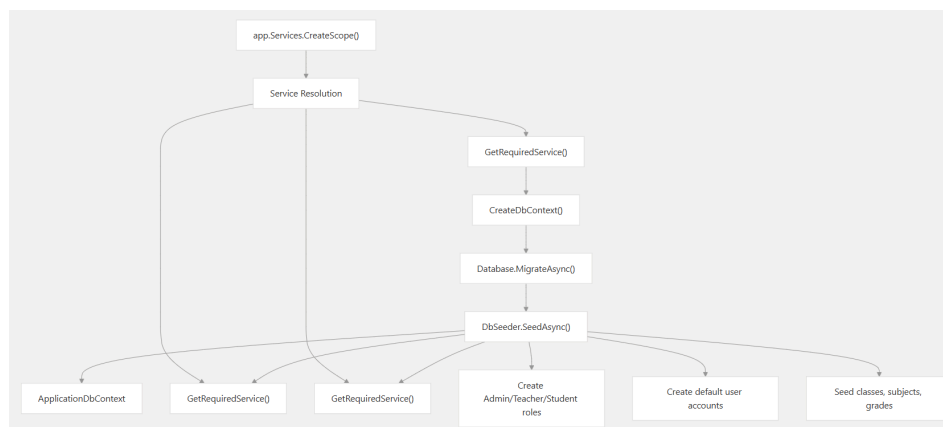
- Polityka hasel: wyłączone wymogi dotyczące znaków dużych, cyfr i znaków specjalnych (linie 53–55).
- Weryfikacja adresu e-mail: wyłączona (SignIn.RequireConfirmedAccount = false, linia 50).
- Polityka blokowania kont: wyłączona dla nowych użytkowników (Lockout.AllowedForNewUsers = false, linia 51).
- Ścieżki ciasteczek: skonfigurowane w ConfigureApplicationCookie (linie 61–66).

Źródła:

- GradeBookApp/Program.cs (linie 48–86)

Konfiguracja potoku middleware

Aplikacja konfiguruje wieloelementowy potok middleware, uwzględniając różne zachowania zależnie od środowiska.



Mapowanie punktów końcowych (linie 136–138 w Program.cs):

- MapControllers() – rejestruje punkty końcowe kontrolerów API.
- MapRazorComponents<App>() – rejestruje komponenty Blazor Server.
- MapAdditionalIdentityEndpoints() – rejestruje dodatkowe punkty końcowe dla UI Identity.

Źródła:

- GradeBookApp/Program.cs (linie 118–138)

Interfejsy użytkownika

Istotne pliki źródłowe

- GradeBookApp/Components/Pages/Admin/AdminDashboard.razor
- GradeBookApp/Components/Pages/Student/StudentAttendance.razor
- GradeBookApp/Components/Pages/Student/StudentAttendance.razor.css
- GradeBookApp/Components/Pages/Student/StudentDashboard.razor
- GradeBookApp/Components/Pages/Teacher/ClassStudents.razor
- GradeBookApp/Components/Pages/Teacher/TeacherDashboard.razor
- GradeBookApp/Controllers/TeacherActionsController.cs

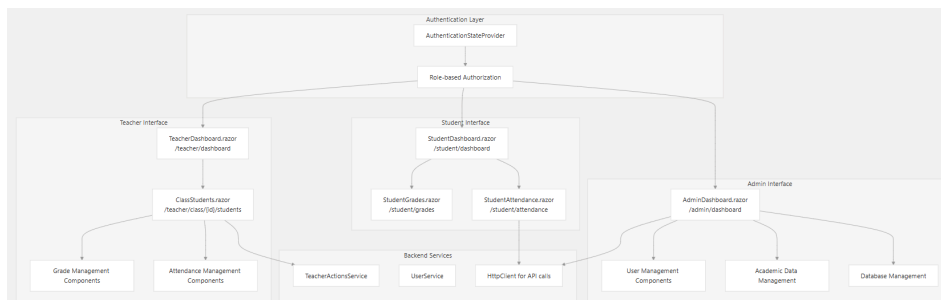
Każdy interfejs oferuje funkcje dedykowane konkretnej roli, wykorzystując komponenty Blazor Server z trybem **@rendermode InteractiveServer** dla interaktywnego renderowania po stronie serwera.

- Interfejsy implementują kontrolę dostępu opartą na rolach (role-based access control) oraz zapewniają odrębne workflowy do zarządzania ocenami i frekwencją.
- Szczegóły mechanizmów uwierzytelniania i autoryzacji: zobacz **Authentication & Authorization**.
- Szczegóły integracji z warstwą serwisów backendowych: zobacz **Backend Services & APIs**.

Przegląd architektury UI

- Aplikacja wykorzystuje komponenty Blazor Server z atrybutem **@rendermode="InteractiveServer"**, co umożliwia dwukierunkową komunikację między front-endem a back-endem w czasie rzeczywistym.

- Każda kategoria interfejsu (Admin, Nauczyciel, Uczeń) posiada własne komponenty oraz nawigację dostosowaną do uprawnień roli.
- Komponenty UI komunikują się z backendem poprzez wstrzykiwane serwisy (TeacherActionsService, UserService) lub bezpośrednio za pomocą HttpClient.



Główne komponenty i ich trasy

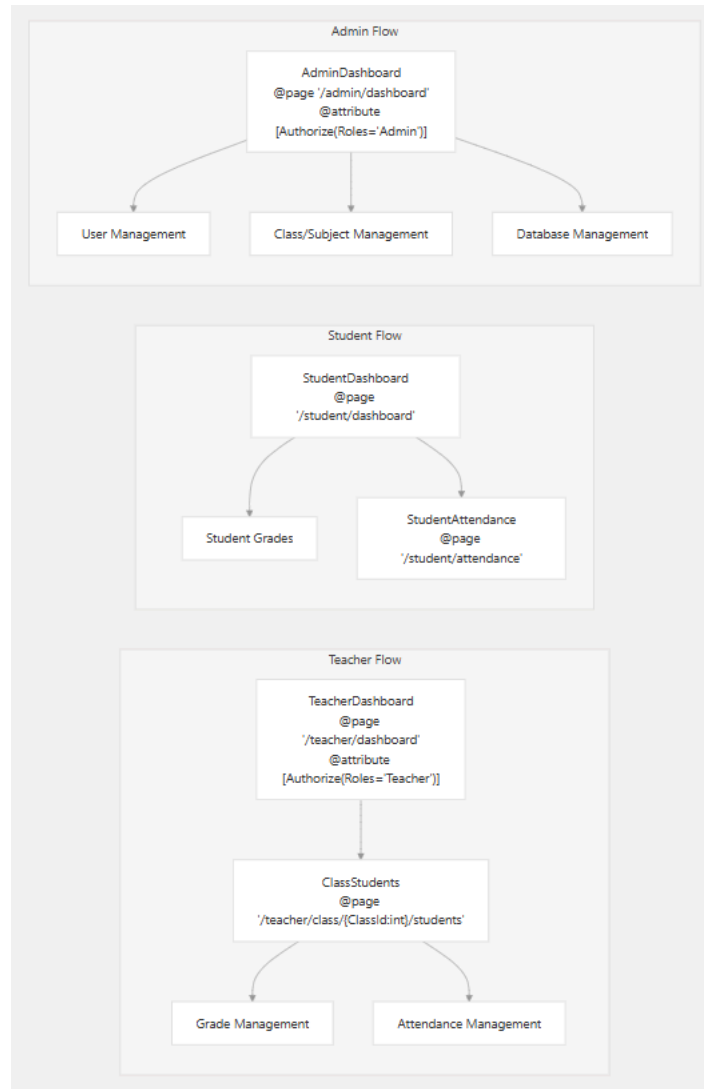
| Kategoria | Komponent | Ścieżka routingu | Opis |
|-----------------------|-----------------------------|---------------------------------------|---|
| Interfejs nauczyciela | TeacherDashboard.razor | /teacher/dashboard | Dashboard nauczyciela z listą przypisanych klas |
| | /teacher/dashboard | /teacher/class/{ClassId:int}/students | Zarządzanie uczniami w klasie (oceny, frekwencja) |
| Interfejs ucznia | StudentDashboard.razor | /student/dashboard | Prosta strona główna dla ucznia |
| | StudentGrades.razor | /student/grades | Widok ocen ucznia |
| | StudentAttendance.razor | /student/attendance | Widok frekwencji ucznia |
| | StudentAttendance.razor.css | | Arkusz stylów do wizualizacji statusu frekwencji |
| Interfejs admina | AdminDashboard.razor | /admin/dashboard | Dashboard administracyjny z danymi statystycznymi i zarządzaniem bazą |
| Kontroler API | TeacherActionsController.cs | /api/teacheractions | Endpointy do operacji nauczyciela (oceny, frekwencja) |

Źródła:

- GradeBookApp/Components/Pages/Teacher/TeacherDashboard.razor (linie 1–82)
- GradeBookApp/Components/Pages/Teacher/ClassStudents.razor (linie 1–202)
- GradeBookApp/Components/Pages/Admin/AdminDashboard.razor (linie 1–271)
- GradeBookApp/Components/Pages/Student/StudentDashboard.razor (linie 1–38)

Przepływ nawigacji komponentów

System routingu w GradeBookApp wykorzystuje parametryzowane trasy oraz atrybuty [Authorize(Roles=...)] w celu ograniczenia dostępu do właściwych komponentów. Poniżej przedstawiono hierarchię nawigacji dla każdej roli.



Źródła:

- GradeBookApp/Components/Pages/Admin/AdminDashboard.razor (linie 1–3)

Interfejs ucznia (Student Flow)

```
@page "/student/dashboard"
```



Panel ucznia

Witaj w swoim panelu. Wybierz, co chcesz zobaczyć:

 Moje oceny

 Moja obecność

Strona główna ucznia z przyciskami nawigacji do ocen i frekwencji.

```
private void GoToGrades() ⇒ Nav.NavigateTo("/student/grades");  
private void GoToAttendance() ⇒ Nav.NavigateTo("/student/attendance");
```

/student/grades:

Moje oceny

| |
|----------|
| POL |
| 546535 |
| MATH |
| 636323 |
| ENG |
| 3436 |
| INFO |
| 55264452 |
| TECH |
| 522556 |
| GEOG |
| 52643662 |
| BIO |
| 626543 |
| CHEM |
| 546 |
| PHYS |
| 363 |
| HIST |
| 2445 |
| WOS |
| 32345 |
| MUS |
| 4462 |
| ART |
| 526 |
| PE |
| 242532 |
| EDB |
| 6342 |

/student/attendance:

| | | |
|-------------------------------|--|------------------------------|
| GradeBookApp Uczeń | | w: 1488 px h: 708 px Wyloguj |
| Moja obecność | | |
| wtorek, 03 czerwca 2025 | | |
| ART | | Obecny |
| poniedziałek, 02 czerwca 2025 | | |
| CHEM | | Obecny |
| piątek, 30 maja 2025 | | |
| INFO | | Obecny |
| czwartek, 29 maja 2025 | | |
| HIST | | Obecny |
| WOS | | Obecny |
| środa, 28 maja 2025 | | |
| POL | | Nieobecny |
| TECH | | Obecny |
| PE | | Obecny |
| wtorek, 27 maja 2025 | | |
| ENG | | Obecny |
| PHYS | | Obecny |
| EDB | | Obecny |
| poniedziałek, 26 maja 2025 | | |
| MATH | | Obecny |
| niedziela, 25 maja 2025 | | |
| GEOG | | Obecny |
| BIO | | Obecny |
| czwartek, 22 maja 2025 | | |
| MUS | | Obecny |

Źródła:

- GradeBookApp/Components/Pages/Student/StudentDashboard.razor (linie 28–36)

1. /student/grades

- Komponent StudentGrades.razor wyświetla listę ocen ucznia (czytelny widok tylko do odczytu).

2. /student/attendance

- Komponent StudentAttendance.razor pokazuje historię frekwencji pogrupowaną według daty, z ikonami statusu:
 - status-Obecny
 - status-Nieobecny
 - status-Spóźniony

Źródła:

GradeBookApp/Components/Pages/Student/StudentAttendance.razor (linie 53–72)

GradeBookApp/Components/Pages/Student/StudentAttendance.razor.css (linie 39–52)

Interfejs nauczyciela (Teacher Flow)

```
@page "/teacher/dashboard"
@attribute [Authorize(Roles="Teacher")]
```

Główny panel nauczyciela wyświetlający przypisane klasy.

| GradeBookApp Nauczyciel | | | | w: 1488 px h: 708 px |
|--|-------------|------|--------------|----------------------|
| Panel Nauczyciela | | | | |
| Witaj, Nauczycielu! Poniżej znajdują się Twoje przypisane klasy. | | | | |
| # | Nazwa klasy | Rok | Akcja | |
| 1 | 4A | 2024 | Zobacz klasę | |
| 2 | 5B | 2024 | Zobacz klasę | |
| 3 | 6A | 2024 | Zobacz klasę | |
| 4 | 8B | 2024 | Zobacz klasę | |

Pobieranie danych nauczyciela bazuje na stanie uwierzytelnienia:

```
var teacherId = user.FindFirst("sub")?.Value
    ?? user.FindFirst("http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier")?.Value;
classes = await TeacherService.GetClassesForTeacherAsync(teacherId);
```

Źródła:

- GradeBookApp/Components/Pages/Teacher/TeacherDashboard.razor (linie 62–81)

/teacher/attendance/{ClassId:int}/{SubjectId}

| GradeBookApp Nauczyciel | | | | w: 1488 px h: 708 px |
|---------------------------|-------------------|--------------------------------------|-------------------------------------|----------------------|
| Sprawdź obecność | | | | |
| Klasa: 1 Przedmiot: POL | | | | |
| # | Imię i nazwisko | ID ucznia | Obecny | |
| 1 | Uczeń2 Klasowy4A | 1d19583a-7367-44b2-83ba-b5b5a3561ec | <input checked="" type="checkbox"/> | |
| 2 | Uczeń3 Klasowy4A | 78ac750b-b973-46d3-8d62-b94380be1766 | <input checked="" type="checkbox"/> | |
| 3 | Uczeń4 Klasowy4A | 476e12ae-ccd8-4ea0-aac5-f518e280b118 | <input checked="" type="checkbox"/> | |
| 4 | Uczeń5 Klasowy4A | 823f69ad-3ad7-41ac-9ed2-569b322cb7b4 | <input checked="" type="checkbox"/> | |
| 5 | Uczeń6 Klasowy4A | ecc09d52-9110-4f50-9dfe-e51694a645ab | <input checked="" type="checkbox"/> | |
| 6 | Uczeń7 Klasowy4A | 89dee5d3-35a8-4507-b1f8-140c65e68c63 | <input checked="" type="checkbox"/> | |
| 7 | Uczeń8 Klasowy4A | 16811669-816d-48cd-8409-31e38a0bfd21 | <input checked="" type="checkbox"/> | |
| 8 | Uczeń9 Klasowy4A | 7f11a10a-c0dc-4f74-a4bc-5bcefbb2ad77 | <input checked="" type="checkbox"/> | |
| 9 | Uczeń10 Klasowy4A | 977177ac-bd79-49a8-bbb1-7408b17fcdd | <input checked="" type="checkbox"/> | |
| 10 | Uczeń11 Klasowy4A | 368bda1f-6375-4bea-8f57-949de3b172c5 | <input checked="" type="checkbox"/> | |
| 11 | Uczeń12 Klasowy4A | 8623863d-aa9a-42f3-aa84-4742d0257210 | <input checked="" type="checkbox"/> | |
| 12 | Uczeń13 Klasowy4A | 16647880-1978-4107-97a1-c2ebe63f307a | <input checked="" type="checkbox"/> | |
| 13 | Uczeń14 Klasowy4A | 7109a1f1-dfcf-4d30-a12c-952da052cfbb | <input checked="" type="checkbox"/> | |
| 14 | Uczeń15 Klasowy4A | 019dccc6-e975-4811-b398-dce3a0b15afc | <input checked="" type="checkbox"/> | |
| 15 | Uczeń16 Klasowy4A | 86fa1fa-3e39-4559-9f86-fb5b2bf7fd1d | <input checked="" type="checkbox"/> | |
| 16 | Uczeń17 Klasowy4A | 46fe1c3e-ed00-4b58-bf1c-e0ddf489bfc3 | <input checked="" type="checkbox"/> | |
| 17 | Uczeń18 Klasowy4A | df79fd86-c613-4b71-a94f-6d1adc49112e | <input checked="" type="checkbox"/> | |
| 18 | Uczeń19 Klasowy4A | 5ea7d12a-4b1d-481b-8cad-72f0f08c8bc3 | <input checked="" type="checkbox"/> | |
| 19 | Uczeń20 Klasowy4A | 5f5e85bc-21f6-4708-a89e-076cbe714bef | <input checked="" type="checkbox"/> | |
| Zapisz obecność. | | | | |

/teacher/attendance/class/{ClassId:int}/{SubjectId}

GradeBookApp

Nauczyciel

w: 1488 pxh: 708 pxWyjdź

Sprawdź obecność

Klasa: 1 | Przedmiot: POL

| # | Imię i nazwisko | ID ucznia | Obecny |
|----|-------------------|--------------------------------------|-------------------------------------|
| 1 | Uczeń2 Klasowy4A | 1d19503a-7367-44b2-85ba-b5b5a3561ec | <input checked="" type="checkbox"/> |
| 2 | Uczeń3 Klasowy4A | 70ac750b-b973-46d3-8d62-b94308be1766 | <input checked="" type="checkbox"/> |
| 3 | Uczeń4 Klasowy4A | 476e12ae-ccd8-4ea0-aac5-f518e286b118 | <input checked="" type="checkbox"/> |
| 4 | Uczeń5 Klasowy4A | 823f69ad-3ad7-41ac-9ed2-569b322cb7b4 | <input checked="" type="checkbox"/> |
| 5 | Uczeń6 Klasowy4A | ecc09d52-9110-4f50-9dfe-e51694a645ab | <input checked="" type="checkbox"/> |
| 6 | Uczeń7 Klasowy4A | 89dee5d3-35a8-4507-b1f8-140c65e68c63 | <input checked="" type="checkbox"/> |
| 7 | Uczeń8 Klasowy4A | 16811669-816d-48cd-8409-31e3840bf2d1 | <input checked="" type="checkbox"/> |
| 8 | Uczeń9 Klasowy4A | 7f11a10a-c0dc-4f74-a4bc-5bcefbb2ad77 | <input checked="" type="checkbox"/> |
| 9 | Uczeń10 Klasowy4A | 977177ac-bd79-49a8-b8b1-7408b17fcddd | <input checked="" type="checkbox"/> |
| 10 | Uczeń11 Klasowy4A | 368bda1f-6375-4bea-8f57-949de3b172c5 | <input checked="" type="checkbox"/> |
| 11 | Uczeń12 Klasowy4A | 8623083d-aa9a-42f3-aa84-4742d0257210 | <input checked="" type="checkbox"/> |
| 12 | Uczeń13 Klasowy4A | 16647080-1978-4107-97a1-c2ebe63f307a | <input checked="" type="checkbox"/> |
| 13 | Uczeń14 Klasowy4A | 7109a1f1-dfcf-4d30-a12c-952da052cfbb | <input checked="" type="checkbox"/> |
| 14 | Uczeń15 Klasowy4A | 019dccc6-e975-4811-b398-dce3a9b15afc | <input checked="" type="checkbox"/> |
| 15 | Uczeń16 Klasowy4A | 86fa41fa-3e39-4559-9f86-fb5b28f7fd1d | <input checked="" type="checkbox"/> |
| 16 | Uczeń17 Klasowy4A | 46fe1c3e-ed00-4b58-bf1c-e0ddfa89bfc3 | <input checked="" type="checkbox"/> |
| 17 | Uczeń18 Klasowy4A | d79fd86-c613-4b71-a94f-6d1ad49112e | <input checked="" type="checkbox"/> |
| 18 | Uczeń19 Klasowy4A | 5ea7d12a-4b1d-481b-8cad-72f9f00c8bc3 | <input checked="" type="checkbox"/> |
| 19 | Uczeń20 Klasowy4A | 5f5e85bc-21f6-4708-a89e-076cbe714bef | <input checked="" type="checkbox"/> |

Zapisz obecność

/teacher/attendance/edit/{ClassId:int}/{SubjectId}/{Date:datetime}

GradeBookApp

Nauczyciel

w: 1488 pxh: 708 pxWyjdź

Edytuj obecność

Klasa: 1 | Przedmiot: POL | Data: 2025-05-28

| # | Imię i nazwisko | ID ucznia | Obecny |
|----|-------------------|--------------------------------------|-------------------------------------|
| 1 | Uczeń2 Klasowy4A | 1d19503a-7367-44b2-85ba-b5b5a3561ec | <input checked="" type="checkbox"/> |
| 2 | Uczeń3 Klasowy4A | 70ac750b-b973-46d3-8d62-b94308be1766 | <input checked="" type="checkbox"/> |
| 3 | Uczeń4 Klasowy4A | 476e12ae-ccd8-4ea0-aac5-f518e286b118 | <input type="checkbox"/> |
| 4 | Uczeń5 Klasowy4A | 823f69ad-3ad7-41ac-9ed2-569b322cb7b4 | <input checked="" type="checkbox"/> |
| 5 | Uczeń6 Klasowy4A | ecc09d52-9110-4f50-9dfe-e51694a645ab | <input type="checkbox"/> |
| 6 | Uczeń7 Klasowy4A | 89dee5d3-35a8-4507-b1f8-140c65e68c63 | <input checked="" type="checkbox"/> |
| 7 | Uczeń8 Klasowy4A | 16811669-816d-48cd-8409-31e3840bf2d1 | <input checked="" type="checkbox"/> |
| 8 | Uczeń9 Klasowy4A | 7f11a10a-c0dc-4f74-a4bc-5bcefbb2ad77 | <input checked="" type="checkbox"/> |
| 9 | Uczeń10 Klasowy4A | 977177ac-bd79-49a8-b8b1-7408b17fcddd | <input checked="" type="checkbox"/> |
| 10 | Uczeń11 Klasowy4A | 368bda1f-6375-4bea-8f57-949de3b172c5 | <input checked="" type="checkbox"/> |
| 11 | Uczeń12 Klasowy4A | 8623083d-aa9a-42f3-aa84-4742d0257210 | <input checked="" type="checkbox"/> |
| 12 | Uczeń13 Klasowy4A | 16647080-1978-4107-97a1-c2ebe63f307a | <input checked="" type="checkbox"/> |
| 13 | Uczeń14 Klasowy4A | 7109a1f1-dfcf-4d30-a12c-952da052cfbb | <input checked="" type="checkbox"/> |
| 14 | Uczeń15 Klasowy4A | 019dccc6-e975-4811-b398-dce3a9b15afc | <input checked="" type="checkbox"/> |
| 15 | Uczeń16 Klasowy4A | 86fa41fa-3e39-4559-9f86-fb5b28f7fd1d | <input checked="" type="checkbox"/> |
| 16 | Uczeń17 Klasowy4A | 46fe1c3e-ed00-4b58-bf1c-e0ddfa89bfc3 | <input checked="" type="checkbox"/> |
| 17 | Uczeń18 Klasowy4A | d79fd86-c613-4b71-a94f-6d1ad49112e | <input checked="" type="checkbox"/> |
| 18 | Uczeń19 Klasowy4A | 5ea7d12a-4b1d-481b-8cad-72f9f00c8bc3 | <input type="checkbox"/> |
| 19 | Uczeń20 Klasowy4A | 5f5e85bc-21f6-4708-a89e-076cbe714bef | <input checked="" type="checkbox"/> |

Zapisz zmiany

/teacher/grades/class/{ClassId:int}/add/{SubjectId}

Dodaj oceny dla klasy

ID klasy: 1 | Przedmiot: POL

Waga

Opis

1

| # | Imię i nazwisko | ID użytkownika | Ocena |
|----|-------------------|--------------------------------------|-------|
| 1 | Uczeń2 Klasowy4A | 1d19503a-7367-44b2-85ba-b5b5a3561ec | 0 |
| 2 | Uczeń3 Klasowy4A | 70ac750b-b973-46d3-8d62-b94380be1766 | 0 |
| 3 | Uczeń4 Klasowy4A | 476e12ae-ccd8-4ea0-aac5-f518e286b118 | 0 |
| 4 | Uczeń5 Klasowy4A | 823f69ad-3ad7-41ac-9ed2-569b322cb7b4 | 0 |
| 5 | Uczeń6 Klasowy4A | ecc9d52-9110-4f50-9dfe-e51694a645ab | 0 |
| 6 | Uczeń7 Klasowy4A | 89dee5d3-35a8-4507-b1f8-140c6e68c63 | 0 |
| 7 | Uczeń8 Klasowy4A | 16811669-816d-48cd-8409-31e3848bfd21 | 0 |
| 8 | Uczeń9 Klasowy4A | 7f11a10a-c0dc-4f74-a4bc-5bcefb2ad77 | 0 |
| 9 | Uczeń10 Klasowy4A | 977177ac-bd79-49a8-b0b1-7408b17fcbdd | 0 |
| 10 | Uczeń11 Klasowy4A | 368dda1f-6375-4bea-8f57-949de3b172c5 | 0 |
| 11 | Uczeń12 Klasowy4A | 8623083d-aa9a-42f3-aa84-474280257210 | 0 |
| 12 | Uczeń13 Klasowy4A | 16647b80-1978-4107-97a1-c2ebe63f307a | 0 |
| 13 | Uczeń14 Klasowy4A | 7109a1f1-dfcf-4d30-a12c-952dab52cfbb | 0 |
| 14 | Uczeń15 Klasowy4A | 019dccc6-e975-4811-b398-dce3a9b15afc | 0 |
| 15 | Uczeń16 Klasowy4A | 86fa41fa-3e39-4559-9f86-fb5b28f7fd1d | 0 |
| 16 | Uczeń17 Klasowy4A | 46fe1c3e-ede0-4b58-bf1c-e0ddfa89bfc3 | 0 |
| 17 | Uczeń18 Klasowy4A | df79fd86-c613-4b71-a94f-6d1ad49112e | 0 |
| 18 | Uczeń19 Klasowy4A | 5ea7d12a-4b1d-481b-8cad-72f0f00c0bc3 | 0 |
| 19 | Uczeń20 Klasowy4A | 5f5e85bc-21f6-4708-a89e-076cbe714b6f | 0 |

Zatwierdź oceny

/teacher/grades/add/{StudentId}/{SubjectId}

Dodaj ocenę

Uczeń: Uczeń2 Klasowy4A

Wartość oceny

0

Waga

0

Opis

Zatwierdź ocenę

/teacher/grades/edit/{GradeId:int}

GradeBookApp Nauczyciel W: 1488 px H: 708 px Wyloguj

Edytuj ocenę

Wartość oceny

Waga

Opis

Data wystawienia

Zapisz zmiany Anuluj

/teacher/class/{ClassId:int}/students

GradeBookApp Nauczyciel W: 1488 px H: 708 px Wyloguj

Uczniowie w klasie 4A

[Dodaj oceny całej klasie](#)
[Sprawdź obecność](#)
[Zobacz obecności](#)

| # | Imię i nazwisko | E-mail | Oceny | Średnia | Akcje |
|----|-------------------|------------------------|-----------------|---------|-----------------------------|
| 1 | Uczeń2 Klasowy4A | student2@school.local | 6 4 6 3 4 2 | 4,2 | Dodaj ocenę |
| 2 | Uczeń3 Klasowy4A | student3@school.local | 6 4 2 6 2 | 4,44 | Dodaj ocenę |
| 3 | Uczeń4 Klasowy4A | student4@school.local | 5 4 6 3 3 5 | 4,83 | Dodaj ocenę |
| 4 | Uczeń5 Klasowy4A | student5@school.local | 2 4 3 3 2 5 3 4 | 3,47 | Dodaj ocenę |
| 5 | Uczeń6 Klasowy4A | student6@school.local | 2 4 4 3 2 5 5 2 | 3,2 | Dodaj ocenę |
| 6 | Uczeń7 Klasowy4A | student7@school.local | 2 4 4 6 4 3 4 | 4,12 | Dodaj ocenę |
| 7 | Uczeń8 Klasowy4A | student8@school.local | 4 3 2 5 | 4,4 | Dodaj ocenę |
| 8 | Uczeń9 Klasowy4A | student9@school.local | 2 4 5 | 4,17 | Dodaj ocenę |
| 9 | Uczeń10 Klasowy4A | student10@school.local | 5 3 2 6 2 1 | 3,5 | Dodaj ocenę |
| 10 | Uczeń11 Klasowy4A | student11@school.local | 2 5 4 5 5 | 3,91 | Dodaj ocenę |
| 11 | Uczeń12 Klasowy4A | student12@school.local | 4 3 4 2 4 5 5 | 3,9 | Dodaj ocenę |
| 12 | Uczeń13 Klasowy4A | student13@school.local | 5 6 6 4 | 5,22 | Dodaj ocenę |
| 13 | Uczeń14 Klasowy4A | student14@school.local | 4 6 6 3 2 4 5 | 5,08 | Dodaj ocenę |
| 14 | Uczeń15 Klasowy4A | student15@school.local | 3 3 6 2 4 | 3,5 | Dodaj ocenę |
| 15 | Uczeń16 Klasowy4A | student16@school.local | 3 5 5 3 4 4 4 5 | 4,44 | Dodaj ocenę |
| 16 | Uczeń17 Klasowy4A | student17@school.local | 5 5 5 | 5 | Dodaj ocenę |
| 17 | Uczeń18 Klasowy4A | student18@school.local | 3 5 5 5 2 | 3,43 | Dodaj ocenę |
| 18 | Uczeń19 Klasowy4A | student19@school.local | 5 5 1 | 4,43 | Dodaj ocenę |
| 19 | Uczeń20 Klasowy4A | student20@school.local | 5 6 4 6 2 2 | 3,92 | Dodaj ocenę |

```
@page "/teacher/class/{ClassId:int}/students"
@attribute [Authorize(Roles="Teacher")]
```

Komponent ClassStudents.razor — główne miejsce pracy nauczyciela z listą uczniów w wybranej klasie.

Funkcje:

- Wyświetlanie listy uczniów z ich średnimi ocenami.
- Modalne okna do przeglądu, edycji i usuwania pojedynczych ocen.
- Przejście do zarządzania frekwencją.

- Bulk operations (masowe dodawanie/edycja ocen).
- Integracja z TeacherActionsService do wywołań backendowych.

Źródła:

- GradeBookApp/Components/Pages/Teacher/ClassStudents.razor (linie 9–125, 139–201)

Interfejs Admina (Admin Flow)

/admin/dashboard:

GradeBookAppAdmin

w: 1488 pxh: 708 pxWyloguj

Panel Administracyjny

Jesteś zalogowany jako: admin@school.local
Role: Admin, Admin

Aktywna baza
Backup
(gradebookdb_backup)

Użytkownicy
79
Razem zarejestrowanych

Klasy
4
Aktywne klasy

Przedmioty
15
Zdefiniowane przedmioty

Przełącz bazę

Synchronizuj

Odsiewz dane

Wyloguj

/admin/dashboard/classes

GradeBookAppAdmin

w: 1488 pxh: 708 pxWyloguj



Lista klas

Dodaj klasę

| Nazwa | Rok | Wychowawca | Akcje |
|-------|------|---------------------|-----------------------|
| 4A | 2024 | Maria Matematyczna | <div>EdytujUsuń</div> |
| 5B | 2024 | Bartek Biologiczny | <div>EdytujUsuń</div> |
| 6A | 2024 | Helena Historyczna | <div>EdytujUsuń</div> |
| 8B | 2024 | Paweł Polonistyczny | <div>EdytujUsuń</div> |



/admin/dashboard/classes/edit/{ClassId:int}

Edytuj klasę

| | |
|---|---|
| Nazwa klasy | <input type="text" value="4A"/> |
| Rok rozpoczęcia | <input type="text" value="2024"/> |
| Wychowawca | <input type="text" value="Maria Matematyczna (math@school.local)"/> |
| <div> Zapisz zmiany  Anuluj</div> | |

/admin/dashboard/classes/add

Dodaj nową klasę

| | |
|--|---|
| Nazwa klasy | <input type="text"/> |
| Rok rozpoczęcia | <input type="text" value="0"/> |
| Wychowawca | <input type="text" value="-- Wybierz wychowawcę --"/> |
| <div> Zapisz  Anuluj</div> | |

/admin/dashboard/subjects

GradeBookApp Admin 1488 px 708 px Wyloguj

Lista przedmiotów

Dodaj przedmiot

| Nazwa | Skrót | Akcje |
|-----------------------------|-------|--------------|
| Plastyka | ART | Edytuj Usuń |
| Biologia | BIO | Edytuj Usuń |
| Chemia | CHEM | Edytuj Usuń |
| Edukacja dla bezpieczeństwa | EDB | Edytuj Usuń |
| Język angielski | ENG | Edytuj Usuń |
| Geografia | GEOG | Edytuj Usuń |
| Historia | HIST | Edytuj Usuń |
| Informatyka | INFO | Edytuj Usuń |
| Matematyka | MATH | Edytuj Usuń |
| Muzyka | MUS | Edytuj Usuń |
| Wychowanie fizyczne | PE | Edytuj Usuń |
| Fizyka | PHYS | Edytuj Usuń |
| Język polski | POL | Edytuj Usuń |
| Technika | TECH | Edytuj Usuń |
| Wiedza o społeczeństwie | WOS | Edytuj Usuń |

/admin/dashboard/subjects/edit/{Id}

GradeBookApp Admin 1488 px 708 px Wyloguj

Edytuj przedmiot

Nazwa przedmiotu

Plastyka

Skrót

ART

Wybierz nauczyciela

Andrzej Artystyczny

Zapisz zmiany Anuluj

/admin/dashboard/subjects/add

+ Dodaj nowy przedmiot

Nazwa przedmiotu

Np. Matematyka

Skrót

Np. MATH

Wybierz nauczyciela

-- wybierz nauczyciela --

Zapisz Anuluj

/admin/dashboard/users

👤 Zarządzanie użytkownikami

+ Dodaj użytkownika

Szukaj użytkownika

Imię, nazwisko, e-mail lub login

Klasa

4A

Rola

Student

Szukaj

| Login | E-mail | Imię | Nazwisko | Klasa | Rola | Akcje |
|------------------------|------------------------|---------|-----------|-------|---------|-------------|
| student2@school.local | student2@school.local | Uczeń2 | Klasowy4A | 4A | Student | Edytuj Usun |
| student3@school.local | student3@school.local | Uczeń3 | Klasowy4A | 4A | Student | Edytuj Usun |
| student4@school.local | student4@school.local | Uczeń4 | Klasowy4A | 4A | Student | Edytuj Usun |
| student5@school.local | student5@school.local | Uczeń5 | Klasowy4A | 4A | Student | Edytuj Usun |
| student6@school.local | student6@school.local | Uczeń6 | Klasowy4A | 4A | Student | Edytuj Usun |
| student7@school.local | student7@school.local | Uczeń7 | Klasowy4A | 4A | Student | Edytuj Usun |
| student8@school.local | student8@school.local | Uczeń8 | Klasowy4A | 4A | Student | Edytuj Usun |
| student9@school.local | student9@school.local | Uczeń9 | Klasowy4A | 4A | Student | Edytuj Usun |
| student10@school.local | student10@school.local | Uczeń10 | Klasowy4A | 4A | Student | Edytuj Usun |
| student11@school.local | student11@school.local | Uczeń11 | Klasowy4A | 4A | Student | Edytuj Usun |
| student12@school.local | student12@school.local | Uczeń12 | Klasowy4A | 4A | Student | Edytuj Usun |
| student13@school.local | student13@school.local | Uczeń13 | Klasowy4A | 4A | Student | Edytuj Usun |
| student14@school.local | student14@school.local | Uczeń14 | Klasowy4A | 4A | Student | Edytuj Usun |
| student15@school.local | student15@school.local | Uczeń15 | Klasowy4A | 4A | Student | Edytuj Usun |
| student16@school.local | student16@school.local | Uczeń16 | Klasowy4A | 4A | Student | Edytuj Usun |
| student17@school.local | student17@school.local | Uczeń17 | Klasowy4A | 4A | Student | Edytuj Usun |
| student18@school.local | student18@school.local | Uczeń18 | Klasowy4A | 4A | Student | Edytuj Usun |
| student19@school.local | student19@school.local | Uczeń19 | Klasowy4A | 4A | Student | Edytuj Usun |
| student20@school.local | student20@school.local | Uczeń20 | Klasowy4A | 4A | Student | Edytuj Usun |

/admin/dashboard/users/edit/{UserId}

GradeBookApp Admin w: 1488 px h: 708 px Wyloguj

Edytuj użytkownika

Login

student2@school.local

Imię

Uczeń2

Data urodzenia

01.01.2014

Adres

PESEL

Klasa

4A (2024)

Email

student2@school.local

Nazwisko

Klasowy4A

Płeć

Kobieta

Data rozpoczęcia nauki

01.09.2022

Zapisz zmiany Anuluj

/admin/dashboard/users/add

GradeBookApp Admin w: 1488 px h: 708 px Wyloguj

+ Dodaj użytkownika

Imię

Nazwisko

Data urodzenia

dd.mm.rrrr

Płeć

Kobieta

Adres

PESEL

Czy nauczyciel?

☐

Data rozpoczęcia nauki

dd.mm.rrrr

Przypisz do klasy

-- Wybierz klasę --

Rola

Student

Hasło

Szkola123!

Dodaj Anuluj

/admin/db

GradeBookApp Admin w: 1488 px h: 708 px Wyloguj

Administrowanie bazą danych

Aktualnie: Backup

Wybierz bazę: Backup Zapisz Synchronizuj

Komponenty interfejsu nauczyciela

Główny dashboard nauczyciela (TeacherDashboard.razor)

- **Opis:**
 - Wyświetla krótki podgląd wszystkich klas, które dany nauczyciel ma przypisane.
 - Pozwala na nawigację do konkretnej klasy w celu zarządzania ocenami i frekwencją.
- **Kluczowe elementy w kodzie (TeacherDashboard.razors, linie 1–82):**
 - Iniekcja serwisu do operacji nauczyciela i dostępu do stanu uwierzytelnienia:

```
@inject TeacherActionsService TeacherService
@Inject AuthenticationStateProvider AuthStateProvider
```

Metoda OnInitializedAsync() pobierająca identyfikator nauczyciela z tokenów i wywołująca GetClassesForTeacherAsync(teacherId).

Interfejs zarządzania klasą (ClassStudents.razor)

Opis:

- Główne narzędzie nauczyciela do zarządzania uczniami w konkretnej klasie.
- Wyświetla tabelę uczniów razem z ich średnimi ocenami.
- Umożliwia:
 - Przeglądanie szczegółów ocen w modalnych oknach.
 - Edycję i usuwanie ocen za pomocą przycisków w interfejsie.
 - Dostęp do funkcji masowych operacji (bulk operations) i przejście do widoku frekwencji.
- Kluczowe fragmenty (ClassStudents.razor):
 - Iniekcja serwisu oraz AuthenticationStateProvider w liniach 5–7:

```
@inject TeacherActionsService TeacherService
@Inject AuthenticationStateProvider AuthStateProvider
```

- Bloki logiczne obsługi stanu ładowania i błędów (linie 139–169).
- Definicja tabeli z uczniami i średnimi ocenami (linie 9–125).
- Obsługa modalnych dialogów z formularzem edycji/usuwania oceny (linie 139–201).

Komponenty interfejsu ucznia

Dashboard ucznia (StudentDashboard.razor)

- **Opis:**
 - Prosta strona główna umożliwiająca nawigację do widoku ocen lub frekwencji ucznia.
 - Zawiera przyciski wywołujące metody GoToGrades() i GoToAttendance().
- **Kluczowe fragmenty (StudentDashboard.razor, linie 1–38):**


```

@page "/student/dashboard"
@inject NavigationManager Nav

<button @onclick="GoToGrades">Moje oceny</button>
<button @onclick="GoToAttendance">Moja frekwencja</button>

@code {
    private void GoToGrades() => Nav.NavigateTo("/student/grades");
    private void GoToAttendance() => Nav.NavigateTo("/student/attendance");
}

```

Widok frekwencji (StudentAttendance.razor)

- **Opis:**

- Wyświetla historię frekwencji ucznia pogrupowaną według daty.
- Używa HttpClient do pobrania danych z endpointa /api/studentclasses/{userId}/attendance.
- Grupy frekwencji tworzone są na bazie obiektu DateOnly (klucz mapowania).
- Kolumny tabeli lub kafelki wyróżniają statusy obecności za pomocą klas CSS:
 - `.status-Obecny`
 - `.status-Nieobecny`

- **Kluczowe fragmenty kodu:**

- Iniekcja zależności (StudentAttendance.razor, linie 4–5):

```

@inject HttpClient Http
@inject AuthenticationStateProvider AuthStateProvider

```

Logika grupowania i renderowania w OnInitializedAsync() (linie 53–72).

Definicje stylów w StudentAttendance.razor.css (linie 39–52):

```

.status-Obecny { background-color: #d4edda; }
.status-Nieobecny { background-color: #f8d7da; }
.status-Spóźniony { background-color: #fff3cd; }

```

Źródła:

- GradeBookApp/Components/Pages/Student/StudentAttendance.razor (linie 53–72)
- GradeBookApp/Components/Pages/Student/StudentAttendance.razor.css (linie 39–52)

Dashboard administracyjny (AdminDashboard.razor)

- **Opis:**

- Główne miejsce kontrolne dla administratora, prezentujące kluczowe statystyki i umożliwiające zarządzanie bazą danych.
- Wyświetla między innymi:
 - Status aktywnej bazy (flaga useBackup)

- Liczbę użytkowników, klas oraz przedmiotów (pobrane z endpointów API)
- Ostatnie wpisy w dzienniku logów systemowych
- Pozwala na:
 - Przełączanie bazy (nav /admin/db)
 - Synchronizację danych poprzez wywołanie POST /api/db/sync
 - Odświeżanie danych systemowych (ręczne wywołanie funkcji odświeżenia)
- **Kluczowe fragmenty (AdminDashboard.razor):**
 - Iniekcja zależności (AdminDashboard.razor, linie 9–12):

```
@inject HttpClient Http
@inject AuthenticationStateProvider AuthStateProvider
```

- Ładowanie statystyk w OnInitializedAsync() (linie 49–121).
- Sekcja odpowiadająca za operacje na bazie (linie 166–246), w tym przyciski do przełączania i synchronizacji.

Źródła:

- GradeBookApp/Components/Pages/Admin/AdminDashboard.razor (linie 49–121)
- GradeBookApp/Components/Pages/Admin/AdminDashboard.razor (linie 166–246)

Wzorzec integracji z usługami

Wszystkie komponenty UI korzystają ze spójnego wzorca integracji z backendem:

- **Komponenty nauczyciela:**
 - Iniekcja serwisu TeacherActionsService oraz AuthenticationStateProvider.
 - Wywołania do API realizowane przez metody serwisu (GetClassesForTeacherAsync, GetStudentsInClassAsync, AddGradeAsync, itp.).

Przykład iniekcji (ClassStudents.razor):

```
@inject TeacherActionsService TeacherService
@inject AuthenticationStateProvider AuthStateProvider
```

Źródła:

- GradeBookApp/Components/Pages/Teacher/ClassStudents.razor (linie 5–7)
- **Komponenty ucznia i administratora:**
 - Iniekcja HttpClient oraz AuthenticationStateProvider.
 - Bezpośrednie wywołania endpointów RESTowych (np. /api/studentclasses/{userId}/attendance, /api/admin/users/count).

Przykład iniekcji (StudentAttendance.razor):

```
@inject HttpClient Http
@inject AuthenticationStateProvider AuthStateProvider
```

Źródła:

- GradeBookApp/Components/Pages/Student/StudentAttendance.razor (linie 4–5)
- GradeBookApp/Components/Pages/Admin/AdminDashboard.razor (linie 9–12)

Zarządzanie stanem komponentów

Komponenty korzystają ze standardowych cykli życia Blazor Server, definiując flagi stanu ładowania i obsługę błędów:

- **Metoda `OnInitializedAsync()`**

- Używana do asynchronicznego pobierania danych potrzebnych do wyświetlenia komponentu (lista klas, frekwencja, statystyki).
- Przykładowy fragment obsługi stanu ładowania:

```
private bool isLoading = true;
private string? errorMessage;

protected override async Task OnInitializedAsync()
{
    try
    {
        var data = await TeacherService.GetClassesForTeacherAsync(teacherId);
        classes = data;
    }
    catch (Exception ex)
    {
        errorMessage = "Wystąpił błąd: " + ex.Message;
    }
    finally
    {
        isLoading = false;
    }
}
```

Źródła:

- `GradeBookApp/Components/Pages/Teacher/ClassStudents.razor` (linie 139–169)
- `GradeBookApp/Components/Pages/Admin/AdminDashboard.razor` (linie 152–200)
- `GradeBookApp/Components/Pages/Student/StudentAttendance.razor` (linie 53–71)
- Flagi stanu (`loading`, `isLoading`)
 - Wykorzystywane do warunkowego renderowania spinnerów lub komunikatów o ładowaniu.
 - Przykład:

```
@if (isLoading)
{
    <div>Ładowanie danych...</div>
}
else if (!string.IsNullOrEmpty(errorMessage))
{
    <div class="alert alert-danger">@errorMessage</div>
}
else
{
    <!-- Renderowanie właściwych danych -->
}
```

- **Zarządzanie modalnymi dialogami**

- Kluczowe dla komponentów nauczyciela (edycja/usuń oceny) – używane flagi typu bool `isModalOpen`, metody `OpenModal()`, `CloseModal()`, a następnie warunkowe wyświetlenie modalnego okna.

Backend Services & APIs

Istotne pliki źródłowe:

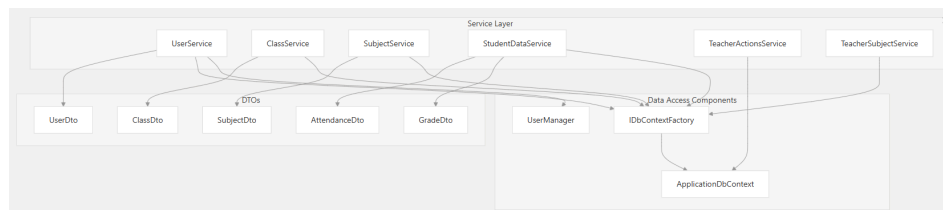
- `GradeBookApp/Controllers/ClassesController.cs`
- `GradeBookApp/Controllers/StudentClassesController.cs`
- `GradeBookApp/Controllers/SubjectsController.cs`
- `GradeBookApp/Controllers/TeacherSubjectsController.cs`
- `GradeBookApp/Services/UserService.cs`

Backend jest zorganizowany w dwie główne warstwy: warstwę usług, która enkapsuluje logikę biznesową i operacje dostępu do danych, oraz warstwę kontrolerów API, która udostępnia punkty końcowe RESTful do interakcji z klientem.

Dla szczegółowych informacji o warstwie danych i konfiguracji Entity Framework, zobacz [Data Layer & Models](#). Dla szczegółów dotyczących implementacji interfejsu użytkownika, zobacz [User Interfaces](#).

Architektura warstwy usług

Warstwa usług zapewnia przejrzystą abstrakcję nad operacjami dostępu do danych i logiką biznesową, stosując wzorzec repozytorium. Usługi obsługują operacje specyficzne dla domeny i koordynują współpracę z kontekstami Entity Framework Core oraz komponentami [ASP.NET Core Identity](#).



Podstawowe komponenty serwisów

| Serwis | Zakres odpowiedzialności | Kluczowe zależności |
|------------------------------|---|---|
| UserService | Zarządzanie kontami użytkowników, przypisywanie ról | UserManager<ApplicationUser>, IDbContextFactory |
| ClassService | Operacje CRUD na klasach | IDbContextFactory<ApplicationDbContext> |
| SubjectService | Operacje CRUD na przedmiotach | IDbContextFactory<ApplicationDbContext> |
| StudentDataService | Pobieranie danych o ocenach i frekwencji uczniów | IDbContextFactory<ApplicationDbContext> |
| TeacherActionsService | Zarządzanie ocenami i frekwencją | ApplicationDbContext |
| TeacherSubjectService | Przypisywanie nauczycieli do przedmiotów i klas | IDbContextFactory<ApplicationDbContext> |

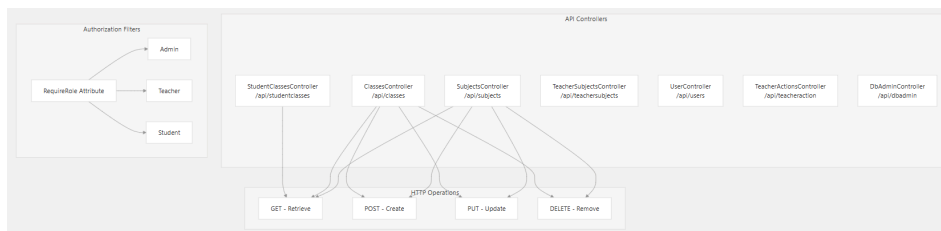
UserService demonstruje wzorzec warstwy serwisów, oferując kompleksowe operacje zarządzania użytkownikami. Obsługuje tworzenie kont z automatycznym generowaniem nazwy użytkownika, zarządzanie rolami oraz funkcje wyszukiwania.

Źródło:

- GradeBookApp/Services/UserService.cs (linie 1–275)

Struktura kontrolerów API

Warstwa API udostępnia punkty końcowe RESTful, zorganizowane według encji domenowych. Kontrolery stosują spójne wzorce operacji CRUD oraz zawierają autoryzację opartą na rolach realizowaną za pomocą niestandardowych filtrów.



Wzorce punktów końcowych kontrolerów

Kontrolery implementują standardowe wzorce RESTful ze spójną strukturą tras:

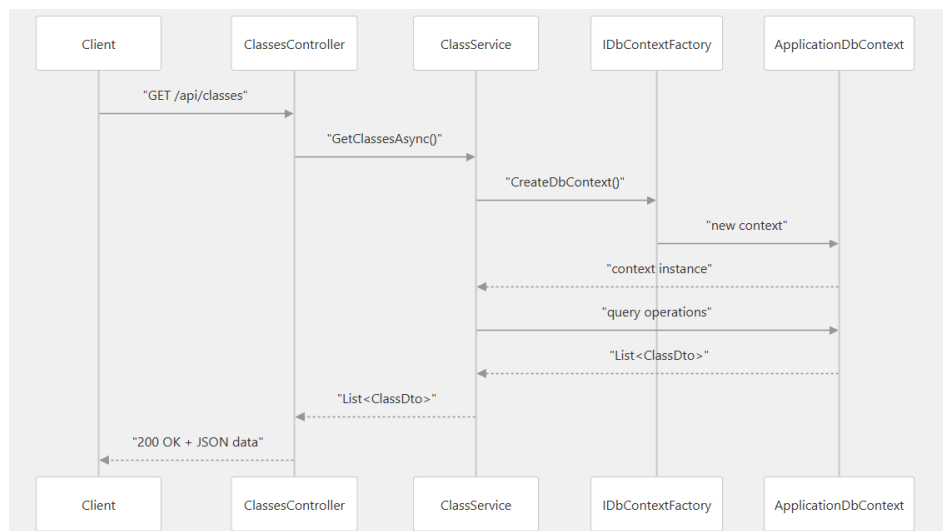
| Kontroler | Trasa bazowa | Obsługiwane operacje | Autoryzacja |
|---------------------------|----------------------|-------------------------------------|---------------------------|
| StudentClassesController | /api/studentclasses | GET uczniów, ocen i frekwencji | Administrator, Uczeń |
| ClassesController | /api/classes | Pełne CRUD, zliczanie, prosty widok | Administrator, Nauczyciel |
| SubjectsController | /api/subjects | Pełne CRUD, zliczanie | Administrator, Nauczyciel |
| TeacherSubjectsController | /api/teachersubjects | GET, POST, DELETE przypisać | Administrator, Nauczyciel |

Źródła:

- GradeBookApp/Controllers/StudentClassesController.cs (10–13)
- GradeBookApp/Controllers/ClassesController.cs (12–14)
- GradeBookApp/Controllers/SubjectsController.cs (13–15)
- GradeBookApp/Controllers/TeacherSubjectsController.cs (8–10)

Wzorce interakcji serwis–kontroler

Kontrolery stosują spójny wzorzec wstrzykiwania zależności i delegowania logiki do serwisów. Obsługują kwestie specyficzne dla HTTP, delegując logikę biznesową do warstwy serwisów.



Wzorce obsługi błędów

Kontrolery stosują spójne podejście do obsługi błędów, zwracając odpowiednie kody statusu HTTP:

- **NotFound()** – gdy nie znaleziono zasobu
- **BadRequest()** – dla błędów walidacji
- **Conflict()** – w przypadku naruszenia reguł biznesowych
- **CreatedAtAction()** – przy pomyślnym utworzeniu zasobu

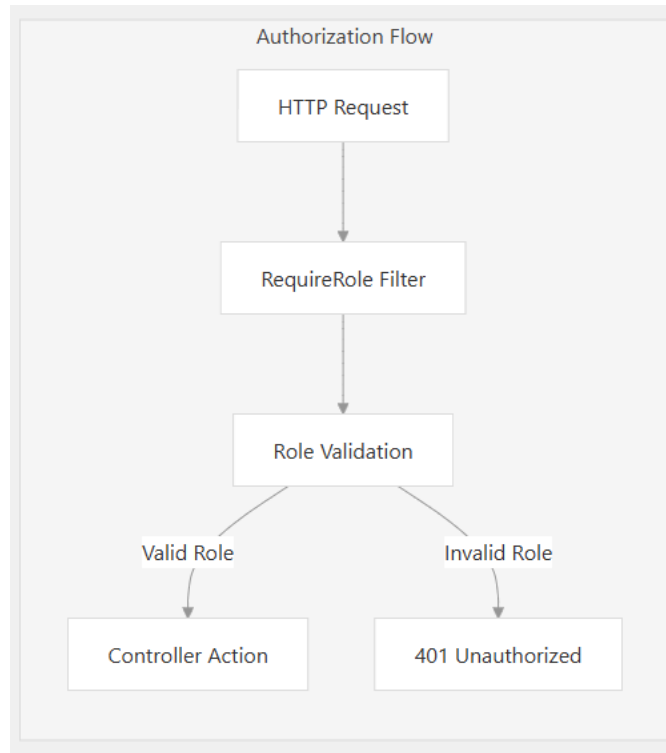
Kontroler SubjectsController pokazuje kompleksową obsługę błędów, korzystając z konkretnych typów wyjątków oraz lokalizowanych komunikatów o błędach.

Źródła:

- GradeBookApp/Controllers/SubjectsController.cs (linie 53–68)
- GradeBookApp/Controllers/ClassesController.cs (linie 52–63)

Wzorce autoryzacji i bezpieczeństwa

Wszystkie kontrolery stosują autoryzację opartą na rolach za pomocą filtra RequireRole. Dzięki temu punkty końcowe są dostępne wyłącznie dla użytkowników posiadających odpowiednie uprawnienia.



Kontrola dostępu oparta na rolach

| Kategoria punktów końcowych | Wymagane role | Wzorzec dostępu |
|-----------------------------|---------------------------|---|
| Dostęp do danych uczniów | Administrator, Uczeń | Tryb tylko do odczytu dla własnych danych |
| Zarządzanie klasami | Administrator, Nauczyciel | Pełne operacje CRUD |
| Zarządzanie przedmiotami | Administrator, Nauczyciel | Pełne operacje CRUD |
| Zarządzanie użytkownikami | Administrator | Pełny dostęp administracyjny |

Kontroler `StudentClassesController` ogranicza dostęp do ról Administrator i Uczeń, zapewniając, że uczniowie mogą uzyskiwać dostęp jedynie do swoich własnych danych akademickich, podczas gdy administratorzy mają pełny dostęp do systemu.

Źródła:

- `GradeBookApp/Controllers/StudentClassesController.cs`, linia 10
- `GradeBookApp/Controllers/ClassesController.cs`, linia 12
- `GradeBookApp/Controllers/SubjectsController.cs`, linia 13

Warstwa serwisów

Istotne pliki źródłowe:

- `GradeBookApp/Services/StudentClassService.cs`
- `GradeBookApp/Services/SubjectService.cs`

- GradeBookApp/Services/TeacherSubjectService.cs
- GradeBookApp/Services/UserService.cs

Cel i zakres

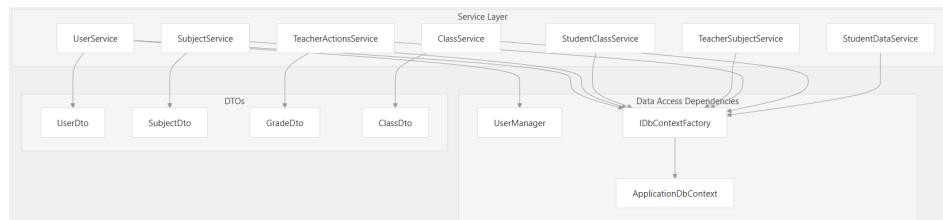
Warstwa serwisów zawiera usługi logiki biznesowej, które obsługują podstawowe operacje domenowe i oddzielają interakcje z bazą danych od komponentów UI oraz kontrolerów API. Usługi te zarządzają transformacją danych między encjami a obiektami DTO, implementują reguły biznesowe oraz koordynują pracę z Entity Framework Core i [ASP.NET Core Identity](#).

Dla informacji o punktach końcowych API korzystających z tych usług zobacz API Controllers. Dla szczegółów dotyczących modeli danych i encji, z którymi pracują te usługi, zobacz Data Layer & Models.

Przegląd architektury serwisów

Warstwa serwisów opiera się na podejściu Domain-Driven Design, gdzie każdy serwis odpowiada za konkretną część domeny biznesowej. Serwisy korzystają z wstrzykiwania zależności (dependency injection) w celu uzyskania dostępu do kontekstów bazodanowych oraz komponentów zarządzania tożsamością.

Zależności serwisów



Źródła:

- GradeBookApp/Services/UserService.cs (linia 13–21)
- GradeBookApp/Services/SubjectService.cs (linia 10–15)
- GradeBookApp/Services/StudentClassService.cs (linia 9–14)
- GradeBookApp/Services/TeacherSubjectService.cs (linia 9–14)

UserService

UserService obsługuje kompleksowe operacje zarządzania użytkownikami, w tym operacje CRUD, zarządzanie rolami oraz funkcje wyszukiwania użytkowników. Łączy mechanizmy ASP.NET Core Identity z logiką domenową aplikacji.

| Metoda | Cel | Typ zwracany |
|---|--|----------------------|
| GetUsersAsync() | Pobiera wszystkich użytkowników z załadowanymi rolami | Task<List<UserDto>> |
| GetUserByIdAsync(string id) | Pobiera pojedynczego użytkownika po identyfikatorze | Task<UserDto?> |
| CreateUserAsync(UserDto, string, string) | Tworzy nowego użytkownika z wygenerowanymi danymi uwierzytelnienia | Task<IdentityResult> |
| UpdateUserAsync(string, UserDto, string?) | Aktualizuje dane użytkownika i opcjonalnie zmienia rolę | Task<IdentityResult> |

| | | |
|--------------------------------|--|----------------------|
| DeleteUserAsync(string id) | Usuwa użytkownika z systemu | Task<IdentityResult> |
| SearchUsersAsync(string query) | Wyszukuje użytkowników po różnych kryteriach | Task<List<UserDto>> |
| GetTeachersAsync() | Pobiera użytkowników z rolą „Teacher” | Task<List<UserDto>> |

Kluczowe funkcje:

- Automatyczne generowanie nazwy użytkownika/adresu e-mail na podstawie imienia i nazwiska
(GradeBookApp/Services/UserService.cs, linie 97–102)
- Obsługa duplikatów nazw użytkowników poprzez dodawanie sufiksów liczbowych
(GradeBookApp/Services/UserService.cs, linie 255–274)
- Filtrowanie i zarządzanie użytkownikami w oparciu o role
(GradeBookApp/Services/UserService.cs, linie 231–250)

Źródło:

GradeBookApp/Services/UserService.cs (linie 11–275)

SubjectService

SubjectService zarządza encjami przedmiotów akademickich, zapewniając walidację unikalności nazw i skrótów, aby uniknąć duplikacji.

| Metoda | Cel | Typ zwracany |
|------------------------------------|--|------------------------|
| GetSubjectsAsync() | Pobiera wszystkie przedmioty | Task<List<SubjectDto>> |
| GetSubjectByIdAsync(string id) | Pobiera przedmiot po identyfikatorze | Task<SubjectDto?> |
| CreateSubjectAsync(SubjectDto dto) | Tworzy nowy przedmiot z walidacją unikalności | Task<SubjectDto> |
| UpdateSubjectAsync(SubjectDto dto) | Aktualizuje przedmiot z kontrolą konfliktu (sprawdza unikalność) | Task<bool> |
| DeleteSubjectAsync(string id) | Usuwa przedmiot | Task<bool> |

Reguły biznesowe:

- Nazwy przedmiotów i ich skróty muszą być unikalne (niezależnie od wielkości liter)
(GradeBookApp/Services/SubjectService.cs, linie 50–57)
- Automatyczna generacja GUID dla nowych przedmiotów
(GradeBookApp/Services/SubjectService.cs, linia 61)

Źródło:

GradeBookApp/Services/SubjectService.cs (linie 8–106)

StudentClassService

StudentClassService zarządza przypisaniem uczniów do klas, przez operacje na właściwości ClassId w encjach ApplicationUser.

| Metoda | Cel | Typ zwracany |
|---|--|-----------------------------|
| GetStudentsInClassAsync(int classId) | Pobiera uczniów przypisanych do konkretnej klasy | Task<List<ApplicationUser>> |
| AssignStudentsToClass(int classId, List<string> studentIds) | Przypisuje uczniów do klasy, usuwając wcześniejsze przypisania | Task<bool> |

Logika przypisywania:

- Usuwa istniejące przypisania uczniów do klas przed utworzeniem nowych

(GradeBookApp/Services/StudentClassService.cs, linie 37–44)

- Wykonuje operację zbiorczą (bulk) dla efektywności

(GradeBookApp/Services/StudentClassService.cs, linie 46–50)

Źródło:

GradeBookApp/Services/StudentClassService.cs (linie 7–55)

TeacherSubjectService

TeacherSubjectService zarządza relacjami wiele-do-wielu między nauczycielami, przedmiotami i klasami za pośrednictwem encji TeacherSubject.

| Metoda | Cel | Typ zwracany |
|---|--|----------------------------|
| GetTeacherSubjectsAsync() | Pobiera wszystkie przypisania nauczycieli do przedmiotów i klas (z wykorzystaniem Include) | Task<List<TeacherSubject>> |
| AssignTeacherToSubjectAndClass(string teacherId, string subjectId, int classId) | Tworzy nowe przypisanie, jeśli jeszcze nie istnieje | Task<bool> |
| RemoveTeacherSubjectAssignment(string id) | Usuwa istniejące przypisanie | Task<bool> |

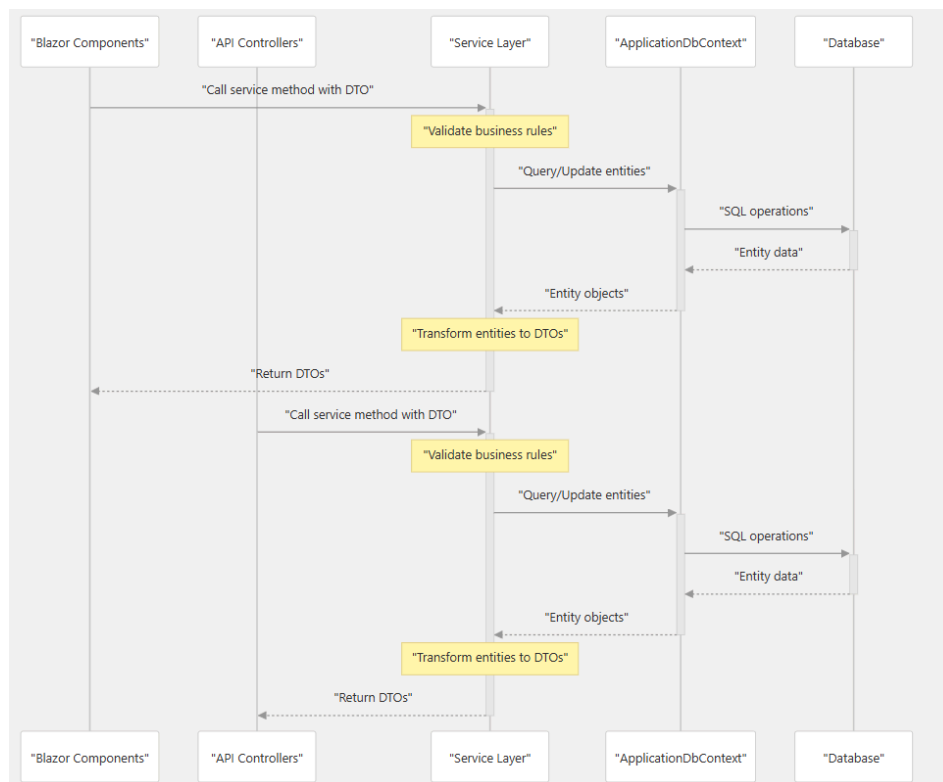
Zarządzanie relacjami:

- Uwzględnia powiązane encje (Include), aby uzyskać pełne dane
(GradeBookApp/Services/TeacherSubjectService.cs, linie 20–24)
- Zapobiega duplikacji przypisań
(GradeBookApp/Services/TeacherSubjectService.cs, linie 31–33)

Źródło:

GradeBookApp/Services/TeacherSubjectService.cs (linie 7–58)

Service Layer Data Flow



Użycie kontekstu bazy danych

Wszystkie serwisy stosują wzorec using z `IDbContextFactory<ApplicationDbContext>`, aby zapewnić prawidłowe zamykanie zasobów:

```
using var context = _contextFactory.CreateDbContext();
// Operacje na bazie danych
await context.SaveChangesAsync();
```

Ten wzorec występuje w:

- `GradeBookApp/Services/SubjectService.cs`, linia 19
- `GradeBookApp/Services/StudentClassService.cs`, linia 19
- `GradeBookApp/Services/TeacherSubjectService.cs`, linia 19

Transformacja DTO

Serwisy konsekwentnie przekształcają encje Entity Framework w obiekty DTO w celu transferu danych:

- **Mapowanie encja → DTO w projekcjach zapytań**
(`GradeBookApp/Services/SubjectService.cs`, linie 21–27)
- **Ręczne wypełnianie DTO dla złożonych scenariuszy**
(`GradeBookApp/Services/UserService.cs`, linie 36–52)

Obsługa błędów

Serwisy stosują różne strategie obsługi błędów:

- **Zwracanie null w przypadku niezlokalizowania zasobu**
(GradeBookApp/Services/SubjectService.cs, linia 34)
- **Zwracanie false w przypadku niepowodzenia operacji**
(GradeBookApp/Services/SubjectService.cs, linia 77)
- **Rzucanie wyjątków przy naruszeniu reguł biznesowych**
(GradeBookApp/Services/SubjectService.cs, linia 57)
- **Zwracanie IdentityResult dla operacji związanych z tożsamością**
(GradeBookApp/Services/UserService.cs, linie 119–129)

Źródła:

- GradeBookApp/Services/UserService.cs (linie 87–95)
- GradeBookApp/Services/SubjectService.cs (linie 50–57)
- GradeBookApp/Services/StudentClassService.cs (linia 34)
- GradeBookApp/Services/TeacherSubjectService.cs (linie 31–33)

Kontrolery API

Istotne pliki źródłowe

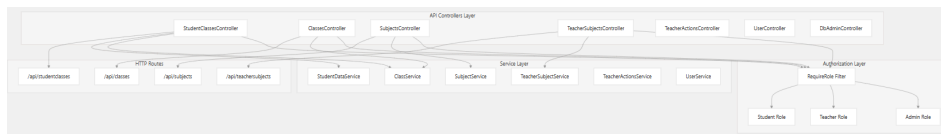
- GradeBookApp/Controllers/ClassesController.cs
- GradeBookApp/Controllers/StudentClassesController.cs
- GradeBookApp/Controllers/SubjectsController.cs
- GradeBookApp/Controllers/TeacherSubjectsController.cs

Te kontrolery pełnią rolę warstwy interfejsu HTTP, obsługując przychodzące żądania i koordynując współpracę z warstwą usług w celu realizacji operacji biznesowych.

Aby uzyskać informacje na temat leżących u podstaw usług logiki biznesowej, zobacz **Warstwa serwisów**. Aby poznać szczegóły modeli danych i DTO używanych przez te kontrolery, zobacz **Warstwa danych i modele**.

Cel i architektura

Kontrolery API w GradeBookApp implementują punkty końcowe RESTful, zorganizowane według funkcjonalności domenowych. Każdy kontroler koncentruje się na określonej domenie biznesowej (klasy, przedmioty, uczniowie, nauczyciele) i udostępnia standardowe operacje CRUD z autoryzacją opartą na rolach.

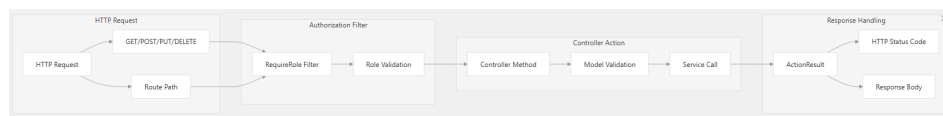


Źródła:

- GradeBookApp/Controllers/StudentClassesController.cs (10-13)

- GradeBookApp/Controllers/ClassesController.cs (12-15)
- GradeBookApp/Controllers/SubjectsController.cs (13-16)
- GradeBookApp/Controllers/TeacherSubjectsController.cs (8-11)

Request Flow Architecture



Źródła:

- GradeBookApp/Controllers/ClassesController.cs (50-63)
- GradeBookApp/Controllers/SubjectsController.cs (51-68)

Autoryzacja oparta na rolach

Wszystkie kontrolery API stosują autoryzację opartą na rolach, wykorzystując atrybut filtra RequireRole. Zapewnia to dostęp do konkretnych punktów końcowych wyłącznie użytkownikom z odpowiednimi rolami.

Macierz autoryzacji

| Kontroler | Dostęp Administratora | Dostęp Nauczyciela | Dostęp Ucznia |
|---------------------------|-----------------------|--------------------|---------------|
| StudentClassesController | ✓ | × | ✓ |
| ClassesController | ✓ | ✓ | × |
| SubjectsController | ✓ | ✓ | × |
| TeacherSubjectsController | ✓ | ✓ | × |

Źródła:

- GradeBookApp/Controllers/StudentClassesController.cs, linia 10
- GradeBookApp/Controllers/ClassesController.cs, linia 12
- GradeBookApp/Controllers/SubjectsController.cs, linia 13
- GradeBookApp/Controllers/TeacherSubjectsController.cs, linia 8

Podstawowe kontrolery API

StudentClassesController

StudentClassesController udostępnia punkty końcowe do pobierania danych uczniów, w tym ocen i frekwencji. Kontroler jest dostępny dla ról Administrator i Uczeń.

- **Trasa bazowa:** /api/studentclasses

Kluczowe punkty końcowe:

- `GET /{classId}/students` – pobiera identyfikatory uczniów dla określonej klasy
- `GET /{studentId}/grades` – pobiera wszystkie oceny dla danego ucznia
- `GET /{studentId}/attendance` – pobiera rekordy frekwencji dla danego ucznia

Zależności serwisu:

- ClassService – do pobierania danych o klasach
- StudentDataService – do pobierania ocen i frekwencji uczniów

Źródła:

- GradeBookApp/Controllers/StudentClassesController.cs, linie 13–22
- GradeBookApp/Controllers/StudentClassesController.cs, linie 24–48

ClassesController

ClassesController obsługuje pełne operacje CRUD związane z zarządzaniem klasami. Dostęp przyznany jest rołom Administrator i Nauczyciel.

- Trasa bazowa: /api/classes

Kluczowe punkty końcowe:

- `GET /` – pobiera listę wszystkich klas
- `GET /count` – zwraca liczbę klas
- `GET /{id}` – pobiera szczegóły konkretnej klasy
- `POST /` – tworzy nową klasę
- `PUT /{id}` – aktualizuje istniejącą klasę
- `DELETE /{id}` – usuwa klasę
- `GET /simple` – zwraca uproszczone dane klas (tylko Id i Name)

Zależności serwisu:

- ClassService – obsługuje logikę biznesową związaną z klasami

Źródła:

- GradeBookApp/Controllers/ClassesController.cs, linie 15–22
- GradeBookApp/Controllers/ClassesController.cs, linie 24–96

SubjectsController

SubjectsController zarządza operacjami CRUD na przedmiotach, z dostępem dla ról Administrator i Nauczyciel.

- Trasa bazowa: /api/subjects

Kluczowe punkty końcowe:

- `GET /` – pobiera wszystkie przedmioty
- `GET /count` – zwraca liczbę przedmiotów
- `GET /{id}` – pobiera szczegóły konkretnego przedmiotu
- `POST /` – tworzy nowy przedmiot
- `PUT /{id}` – aktualizuje istniejący przedmiot
- `DELETE /{id}` – usuwa przedmiot

Zależności serwisu:

- SubjectService – obsługuje logikę biznesową związaną z przedmiotami

Obsługa błędów:

Zawiera specyficzną obsługę DbUpdateException z lokalizowanymi komunikatami błędów w języku polskim.

Źródła:

- GradeBookApp/Controllers/SubjectsController.cs, linie 16–23
- GradeBookApp/Controllers/SubjectsController.cs, linie 25–98

TeacherSubjectsController

TeacherSubjectsController zarządza relacjami przypisań pomiędzy nauczycielami, przedmiotami i klasami.

- Trasa bazowa: /api/teachersubjects

Kluczowe punkty końcowe:

- **GET /** – pobiera wszystkie przypisania nauczyciel–przedmiot–klasa
- **POST /** – tworzy nowe przypisanie (nauczyciel, przedmiot, klasa)
- **DELETE /{id}** – usuwa istniejące przypisanie

DTO:

- AssignTeacherSubjectDto – zawiera TeacherId, SubjectId i ClassId dla przypisań

Zależności serwisu:

- TeacherSubjectService – obsługuje logikę przypisań

Źródła:

- GradeBookApp/Controllers/TeacherSubjectsController.cs, linie 11–18
- GradeBookApp/Controllers/TeacherSubjectsController.cs, linie 20–53

Wzorce projektowe API

Standardowe wzorce odpowiedzi HTTP

Kontrolery stosują spójne wzorce odpowiedzi HTTP:

| Operacja | Odpowiedź sukcesu | Odpowiedź błędu |
|------------------|-----------------------------------|-------------------------------------|
| GET (pojedynczy) | 200 OK z encją | 404 NotFound |
| GET (kolekcja) | 200 OK z listą | Pusta lista, jeśli brak zasobów |
| POST | 201 Created z nagłówkiem Location | 400 BadRequest dla błędów walidacji |
| PUT | 204 NoContent | 400 BadRequest lub 404 NotFound |
| DELETE | 204 NoContent | 404 NotFound |

Wzorzec użycia DTO

Wszystkie kontrolery operują na obiektach DTO (Data Transfer Objects), zamiast bezpośrednio na modelach encji:

- **ClassDto** – do operacji na klasach
- **SubjectDto** – do operacji na przedmiotach
- **GradeDto** – do danych o ocenach
- **AttendanceDto** – do danych o frekwencji

Strategia obsługi błędów

Kontrolery wdrażają spójne podejście do obsługi błędów:

- **Błędy walidacji:** zwracają `400 BadRequest` z opisowymi komunikatami
- **Nie znaleziono zasobu:** zwracają `404 NotFound`
- **Konflikty:** zwracają `409 Conflict` w przypadku duplikacji lub naruszenia unikalności
- **Błędy bazy danych:** łapią `DbUpdateException` i zwracają odpowiednie komunikaty błędów

Źródła:

- GradeBookApp/Controllers/ClassesController.cs, linie 59–62
- GradeBookApp/Controllers/SubjectsController.cs, linie 64–67
- GradeBookApp/Controllers/TeacherSubjectsController.cs, linie 32–34

Integracja z warstwą serwisów

Kontrolery API działają jako cienka warstwa koordynacyjna, delegując logikę biznesową do odpowiednich klas serwisowych. To podejście umożliwia:

- **Testowalność:** Logika biznesowa może być testowana niezależnie od kontekstu HTTP
- **Ponowne użycie:** Serwisy mogą być wykorzystywane zarówno przez kontrolery API, jak i komponenty Blazor
- **Łatwość utrzymania:** Routing HTTP jest oddzielony od logiki biznesowej

Wzorzec wstrzykiwania serwisów (Service Injection Pattern)

Wszystkie kontrolery korzystają z wstrzykiwania zależności w konstruktorze, aby otrzymać wymagane usługi, zgodnie ze standardowym wzorcem ASP.NET Core.

Źródła:

- GradeBookApp/Controllers/StudentClassesController.cs, linie 18–22
- GradeBookApp/Controllers/ClassesController.cs, linie 19–22
- GradeBookApp/Controllers/SubjectsController.cs, linie 20–23

Warstwa danych i modele

Istotne pliki źródłowe

- GradeBookApp/Controllers/UserController.cs
- GradeBookApp/Data/ApplicationDbContext.cs
- GradeBookApp/Data/Seed/DbSeeder.cs
- GradeBookApp/Services/ClassService.cs
- GradeBookApp/Shared/UserDto.cs

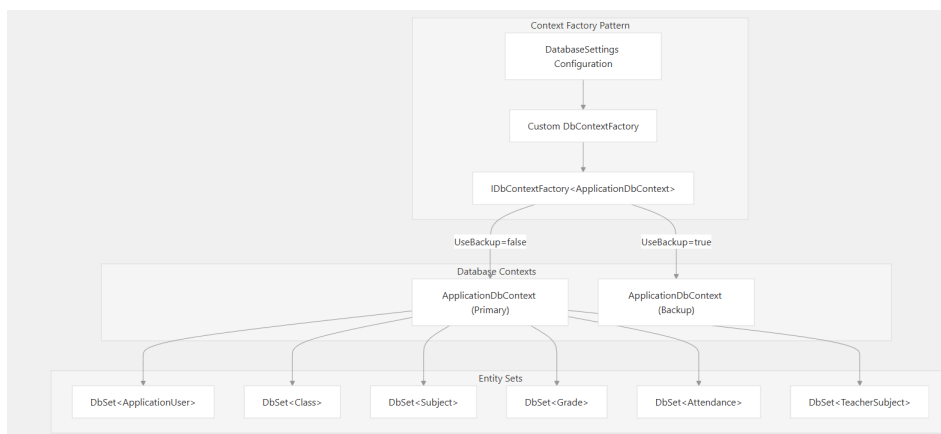
Warstwa danych stanowi fundament wszystkich operacji dziennika ocen, w tym zarządzania użytkownikami, rekordami akademickimi oraz relacjami edukacyjnymi.

Dla informacji o administrowaniu bazą danych i funkcjach wysokiej dostępności zobacz Database Management & High Availability. Aby poznać szczegóły dotyczące sposobu, w jaki usługi korzystają z tej warstwy danych, zobacz Service Layer.

Architektura Entity Framework Core

Aplikacja używa Entity Framework Core z PostgreSQL jako głównym dostawcą bazy danych, skonfigurowanym za pomocą wzorca fabryki, aby wspierać wiele instancji bazy danych.

Architektura kontekstu bazy danych



Źródła:

- GradeBookApp/Data/ApplicationDbContext.cs (linie 7–22)
- GradeBookApp/Services/ClassService.cs (linie 11–16)

Modele encji i relacje

Aplikacja definiuje rozbudowany zestaw modeli encji odzwierciedlających domenę edukacyjną, z precyzyjnie skonfigurowanymi relacjami w celu zachowania integralności danych.

Podstawowy diagram relacji encji

Źródła:

- GradeBookApp/Data/ApplicationDbContext.cs (linie 17–21)
- GradeBookApp/Data/ApplicationDbContext.cs (linie 23–85)

Konfiguracja relacji encji

Klasa ApplicationDbContext konfiguruje zachowania relacji za pomocą metody OnModelCreating:

| Relacja | Zachowanie usuwania | Cel |
|-----------------------|---------------------|--|
| ApplicationUser.Class | SetNull | Uczniowie mogą istnieć bez przypisania do klasy |
| Class.Teacher | Restrict | Zapobiega usunięciu nauczyciela nadzorującego klasę |
| Grade.Student | Cascade | Usuwa oceny, gdy usunięty zostanie uczeń |
| Grade.Teacher | Restrict | Zachowuje oceny, gdy konto nauczyciela zostanie zmienione lub usunięte |
| Attendance.Student | Cascade | Usuwa rekordy frekwencji, gdy usunięty zostanie uczeń |

Źródło:

- GradeBookApp/Data/ApplicationDbContext.cs (linie 28–85)

Obiekty transferu danych (DTO)

Aplikacja używa DTO, aby oddzielić warstwę prezentacji od modeli encji i zapewnić kontrolowaną serializację danych.

Struktura UserDto

Klasa `UserDto` zawiera właściwości obliczane wykorzystywane w celu prezentacji:

- `RolesDisplay` – łączy wszystkie role użytkownika w jeden ciąg lub zwraca „(brak)”, jeśli użytkownik nie ma przypisanych ról
- `FullName` – łączy `FirstName` i `LastName` w pełne imię i nazwisko

Źródło:

- `GradeBookApp/Shared/UserDto.cs` (linie 9–41)

Konfiguracja kontekstu bazy danych

`ApplicationDbContext` dziedziczy po `IdentityDbContext<ApplicationUser>`, co pozwala na integrację ASP.NET Core Identity z własnymi encjami.

Właściwości DbSet

| Właściwość DbSet | Typ encji | Cel |
|------------------------------|-----------------------------|--|
| <code>Attendances</code> | <code>Attendance</code> | Rekordy frekwencji uczniów |
| <code>Classes</code> | <code>Class</code> | Definicje klas akademickich |
| <code>Grades</code> | <code>Grade</code> | Rekordy ocen uczniów |
| <code>Subjects</code> | <code>Subject</code> | Definicje przedmiotów |
| <code>TeacherSubjects</code> | <code>TeacherSubject</code> | Przypisania nauczyciel–przedmiot–klasa |

Konfiguracja klucza złożonego

Encja `TeacherSubject` używa złożonego klucza głównego, łączącego pola `TeacherId`, `SubjectId` i `ClassId` w celu zapewnienia unikalności przypisań nauczyciela do przedmiotu i klasy:

```
builder.Entity<TeacherSubject>()  
    .HasKey(ts => new { ts.TeacherId, ts.SubjectId, ts.ClassId });
```

Źródła:

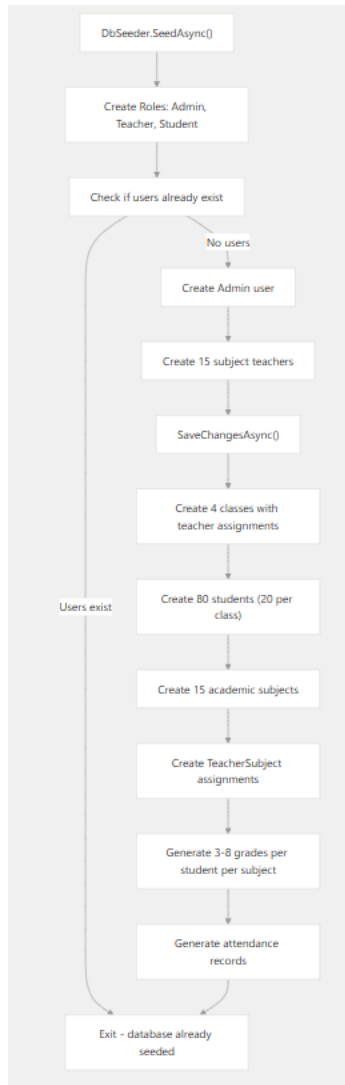
- `GradeBookApp/Data/ApplicationDbContext.cs` (linie 17–21)
- `GradeBookApp/Data/ApplicationDbContext.cs` (linie 40–41)

Seeding bazy danych

Klasa `DbSeeder` zapewnia kompleksowe generowanie danych testowych do celów deweloperskich i demonstracyjnych.

Przebieg procesu seedingowania

Źródło:



- GradeBookApp/Data/Seed/DbSeeder.cs

Struktura zasianych danych

| Rodzaj danych | Ilość | Szczegóły |
|-------------------------|--------|---|
| Użytkownicy Admin | 1 | admin@school.local |
| Użytkownicy Nauczyciele | 15 | Nauczyciele przypisani do konkretnych przedmiotów (np. math@school.local) |
| Użytkownicy Studenci | 80 | 20 uczniów w każdej klasie |
| Klasy | 4 | 4A, 5B, 6A, 8B z przypisanymi wychowawcami |
| Przedmioty | 15 | Podstawowe przedmioty akademickie z polskimi nazwami |
| Oceny | ~9 600 | 3-8 losowych ocen na ucznia w każdym przedmiocie |
| Frekwencja | ~1 120 | Najnowsze wpisy frekwencji z 85 % obecności |

Źródło:

GradeBookApp/Data/Seed/DbSeeder.cs (linie 14-264)

Integracja z warstwą usług

Warstwa danych integruje się z warstwą usług za pomocą wzorca IDbContextFactory<ApplicationDbContext>, co umożliwia prawidłowe zarządzanie połączeniami oraz przełączanie między bazami danych.

Integracja serwisów z warstwą danych



Wzorzec użycia fabryki kontekstu

Usługi stosują spójny schemat dostępu do bazy danych:

- Wstrzykiwanie IDbContextFactory<ApplicationDbContext>
- Tworzenie kontekstu:

```
await using var context = _contextFactory.CreateDbContext();
```

- Logowanie szczegółów połączenia z bazą w celach debugowania
- Wykonywanie operacji na encjach z użyciem odpowiednich wywołań Include()
- Kontekst jest automatycznie zwalniany dzięki zastosowaniu await using

Źródła:

- GradeBookApp/Services/ClassService.cs (linie 11–24)
- GradeBookApp/Services/ClassService.cs (linie 49–53)

Zarządzanie bazą danych i wysoka dostępność

Istotne pliki źródłowe

- GradeBookApp/Components/Pages/Admin/Database/DatabaseAdmin.razor
- GradeBookApp/Controllers/DbAdminController.cs
- GradeBookApp/appsettings.json

System realizuje architekturę dual-bazy z możliwością przełączania w czasie rzeczywistym i operacjami synchronizacji.

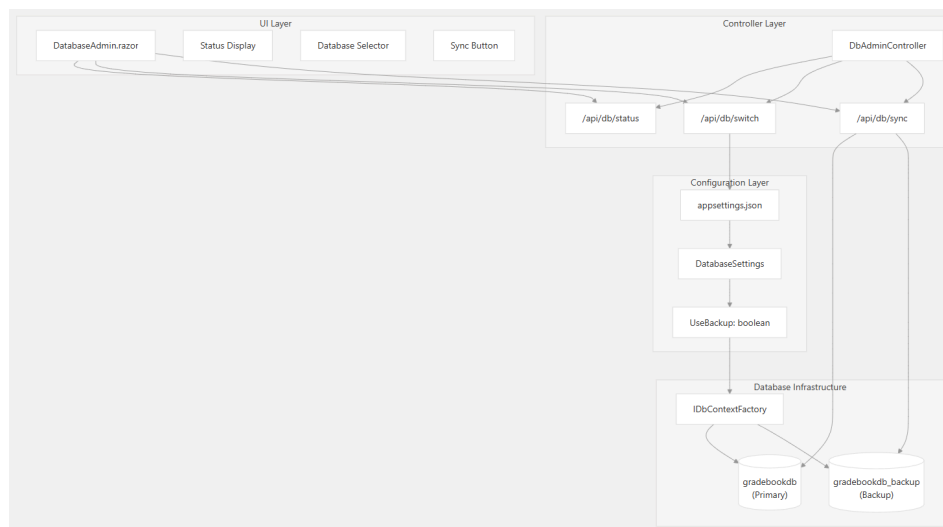
- Aby uzyskać informacje o ogólnej architekturze warstwy danych i konfiguracji Entity Framework, zobacz **Warstwa danych i modele**.
- Aby poznać komponenty interfejsu administratora związane z zarządzaniem, zobacz **Interfejs administratora**.

Przegląd architektury

System zarządzania bazą danych składa się z trzech głównych elementów:

1. **Zarządzanie konfiguracją** – wczytywanie ustawień połączeń do baz głównej i zapasowej z appsettings.json.
2. **Operacje przełączania baz** – mechanizm, który w czasie działania aplikacji pozwala na zmianę aktywnej bazy na podstawie flagi konfiguracyjnej.
3. **Kontrole administracyjne** – interfejsy i punkty końcowe API umożliwiające administratorom ręczne przełączanie oraz synchronizację danych między bazami.

System utrzymuje dwie bazy PostgreSQL i umożliwia przełączanie między nimi bez potrzeby restartu aplikacji.



Zarządzanie konfiguracją

Konfiguracja bazy danych jest zarządzana za pomocą połączenia plików konfiguracyjnych i aktualizacji w czasie działania. System wykorzystuje dwa łańcuchy połączeń oraz flagę przełączającą, aby określić, która baza jest aktywna.

Konfiguracja łańcuchów połączeń

| Ustawienie | Przeznaczenie | Przykładowa wartość |
|----------------------------|------------------------------|--|
| ConnectionStrings:Primary | Połączenie do bazy głównej | Host=localhost;Port=5432;Database=gradebookdb;Username=postgres;Password=1212 |
| ConnectionStrings:Backup | Połączenie do bazy zapasowej | Host=localhost;Port=5432;Database=gradebookdb_backup;Username=postgres;Passwor |
| DatabaseSettings:UseBackup | Flaga wyboru aktywnej bazy | false (główna) lub true (zapasowa) |

Konstruktor DbAdminController odczytuje powyższe łańcuchy połączeń i weryfikuje ich dostępność.

(GradeBookApp/Controllers/DbAdminController.cs, linie 40–47)

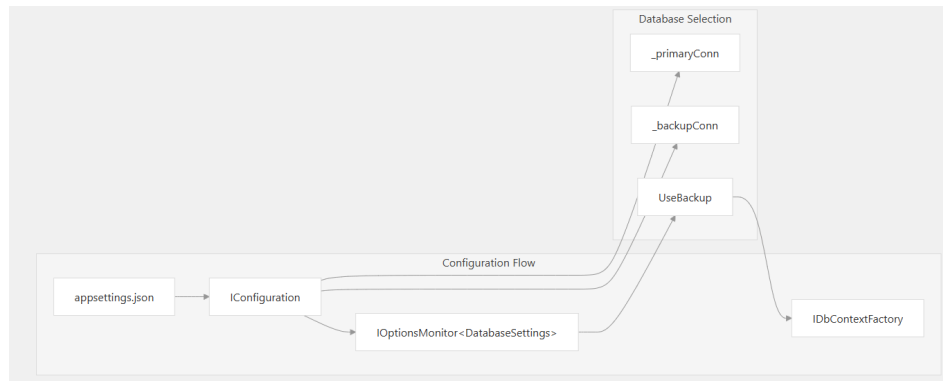
Kontroler wykorzystuje IOptionMonitor<DatabaseSettings> do śledzenia zmian w flagu UseBackup w czasie rzeczywistym.

(GradeBookApp/Controllers/DbAdminController.cs, linia 27)

Źródła:

- GradeBookApp/Controllers/DbAdminController.cs (linie 1–246)

- GradeBookApp/appsettings.json (linie 1–21)



Źródła:

- GradeBookApp/appsettings.json (linie 2–8)
- GradeBookApp/Controllers/DbAdminController.cs (linie 32–52)

Monitorowanie stanu bazy danych

System udostępnia punkt końcowy zwracający informację o aktualnie aktywnej bazie danych. Metoda `GetStatus` w `DbAdminController` pobiera bieżącą wartość `UseBackup` za pomocą wzorca monitorowania opcji:

```
// GET api/db/status endpoint implementation
[HttpGet("status")]
public IActionResult GetStatus()
{
    bool useBackup = _dbSettingsMonitor.CurrentValue.UseBackup;
    return Ok(new { UseBackup = useBackup });
}
```

Powyższy status jest wykorzystywany przez interfejs administracyjny do wyświetlenia aktualnie aktywnej bazy danych oraz do inicjalizacji kontrolki wyboru bazy w widoku:

- GradeBookApp/Components/Pages/Admin/Database/DatabaseAdmin.razor (linie 106–125)

Źródła:

- GradeBookApp/Controllers/DbAdminController.cs (linie 58–64)

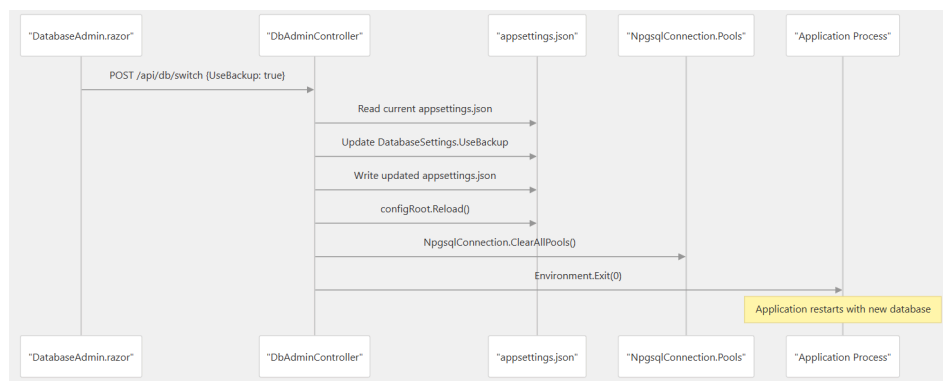
Mechanizm przełączania bazy danych

Funkcja przełączania bazy danych pozwala administratorom na zmianę aktywnej bazy w czasie działania aplikacji. Proces ten obejmuje:

1. Zaktualizowanie pliku konfiguracyjnego
2. Przeładowanie konfiguracji
3. Wyczyszczenie pul połączeń (connection pools)
4. Restart aplikacji

Przebieg procesu przełączania

(Notka: szczegółowy przebieg procesu znajduje się w kodzie i dokumentacji, w tym w DbAdminController oraz w skrypcie konfiguracyjnym.)



Proces przełączania bazy danych

Funkcja przełączania bazy danych jest zaimplementowana w metodzie SwitchDatabase

GradeBookApp/Controllers/DbAdminController.cs (linie 71–136):

- Aktualizacja konfiguracji: Odczytuje plik appsettings.json, aktualizuje wartość DatabaseSettings.UseBackup i zapisuje plik z powrotem
(GradeBookApp/Controllers/DbAdminController.cs, linie 81–108)
- Przeładowanie konfiguracji: Wymusza ponowne wczytanie konfiguracji przy użyciu IConfigurationRoot.Reload()
(GradeBookApp/Controllers/DbAdminController.cs, linie 114–118)
- **Czyszczenie puli połączeń:** Czyści wszystkie pule połączeń Npgsql, aby zapobiec utrzymywaniu nieaktualnych połączeń
(GradeBookApp/Controllers/DbAdminController.cs, linie 121–122)
- Zakończenie procesu: Wywołuje Environment.Exit(0), aby zrestartować aplikację z nowymi ustawieniami
(GradeBookApp/Controllers/DbAdminController.cs, linie 125–126)

Źródła:

GradeBookApp/Controllers/DbAdminController.cs (linie 71–136)

Synchronizacja bazy danych

Funkcja synchronizacji pozwala administratorom na skopiowanie danych z aktualnie aktywnej bazy do bazy nieaktywnej. Dzięki temu obie bazy pozostają zsynchronizowane, co zapewnia mechanizm awaryjnego odzyskiwania danych.

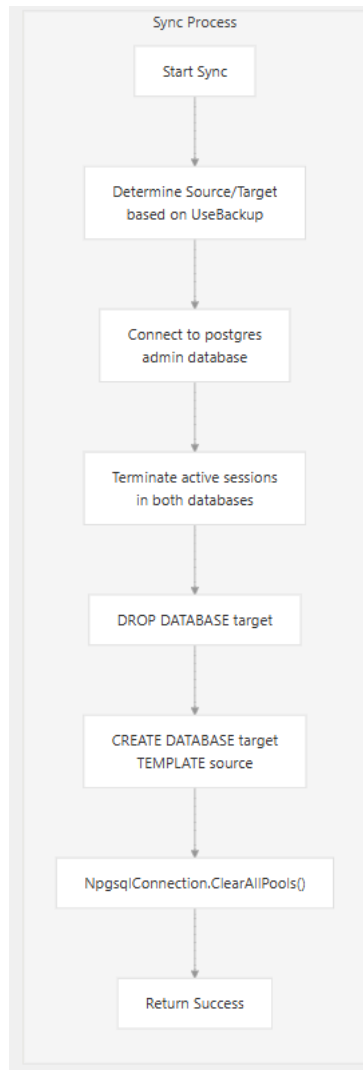
Logika synchronizacji

Metoda SyncBasedOnSetting określa bazę źródłową i docelową w oparciu o aktualną wartość UseBackup

GradeBookApp/Controllers/DbAdminController.cs (linie 150–244):

| Aktualne ustawienie | Baza źródłowa | Baza docelowa | Operacja |
|---------------------|-------------------------------|-------------------------------|-------------------|
| UseBackup = false | Główna (gradebookdb) | Zapasowa (gradebookdb_backup) | Główna → Zapasowa |
| UseBackup = true | Zapasowa (gradebookdb_backup) | Główna (gradebookdb) | Zapasowa → Główna |

Proces synchronizacji



Proces synchronizacji korzysta z funkcji PostgreSQL CREATE DATABASE ... TEMPLATE w celu utworzenia dokładnej kopii bazy źródłowej

(GradeBookApp/Controllers/DbAdminController.cs, linie 219–225):

- **Zamykanie sesji:** Zamyka wszystkie aktywne połączenia do baz źródłowej i docelowej
(GradeBookApp/Controllers/DbAdminController.cs, linie 185–207)
- **Odtwarzanie bazy danych:** Usuwa bazę docelową i odtwarza ją, używając bazy źródłowej jako szablonu
(GradeBookApp/Controllers/DbAdminController.cs, linie 210–225)
- **Czyszczenie puli połączeń:** Czyści pulę połączeń, aby zapewnić świeże połączenia
(GradeBookApp/Controllers/DbAdminController.cs, linia 229)

Źródło:

GradeBookApp/Controllers/DbAdminController.cs (linie 150–244)

Interfejs administracyjny

Interfejs administracyjny do zarządzania bazą danych jest zaimplementowany w pliku DatabaseAdmin.razor i zapewnia przyjazny sposób wykonywania operacji. Komponent jest dostępny wyłącznie dla użytkowników z rolą "Admin"

(GradeBookApp/Components/Pages/Admin/Database/DatabaseAdmin.razor, linia 1)

Komponenty interfejsu

| Komponent | Przeznaczenie | Implementacja |
|--|---|--|
| Wyświetlanie stanu (linia 32–36) | Pokazuje, która baza jest aktualnie aktywna | Odznaka (badge) wyświetlająca "Primary" lub "Backup" |
| Wybór bazy (linia 42–45) | Dropdown umożliwiający wybór bazy docelowej | Element select powiązany z selectedOption |
| Przycisk przełączania (linia 48–60) | Wywołuje operację przełączania bazy | Wywołuje metodę OnSubmitAsync() |
| Przycisk synchronizacji (linia 61–73) | Rozpoczyna proces synchronizacji bazy | Wywołuje metodę OnSyncAsync() |

Zarządzanie stanem

Komponent utrzymuje kilka zmiennych stanu, aby obsługiwać informacje o ładowaniu i komunikaty dla użytkownika:

```
private bool useBackup;      // Bieżący stan bazy (główna/zapasowa)
private bool isLoading = true; // Stan ładowania przy inicjalizacji
private bool isSubmitting = false; // Stan operacji przełączania
private bool isSyncing = false; // Stan operacji synchronizacji
private string? loadError;    // Wiadomość o błędzie przy ładowaniu
private string selectedOption = "Primary"; // Wybrana przez użytkownika opcja
private string? syncMessage;  // Komunikat o wyniku synchronizacji
```

Komponent łączy bieżący stan bazy podczas inicjalizacji i zapewnia informacje zwrotne w czasie rzeczywistym w trakcie wykonywania operacji

(GradeBookApp/Components/Pages/Admin/Database/DatabaseAdmin.razor, linie 99–138)

Źródło:

GradeBookApp/Components/Pages/Admin/Database/DatabaseAdmin.razor (linie 1–214)

Obsługa błędów i logowanie

System zarządzania bazą danych zawiera kompleksową obsługę błędów i logowanie w trakcie wszystkich operacji. DbAdminController korzysta z logowania do konsoli, aby śledzić postęp operacji i diagnozować problemy.

Wzorce logowania:

- **Rozpoczęcie operacji:** Logowanie przy starcie operacji z wartościami przekazanych parametrów
- **Śledzenie postępu:** Logowanie etapów pośrednich, takich jak pobieranie łańcuchów połączeń i operacje na bazie
- **Raportowanie błędów:** Przechwytywanie wyjątków i logowanie szczegółowych informacji o błędach
- **Potwierdzenie sukcesu:** Logowanie pomyślnego zakończenia operacji

Interfejs użytkownika obsługuje błędy API w sposób przyjazny — wyświetla użytkownikowi komunikaty o błędach i przywraca stan UI w przypadku niepowodzenia operacji

(GradeBookApp/Components/Pages/Admin/Database/DatabaseAdmin.razor, linie 152–161)

Źródła:

- GradeBookApp/Controllers/DbAdminController.cs (linie 46–47, 77–78, 131–135)
- GradeBookApp/Components/Pages/Admin/Database/DatabaseAdmin.razor (linie 152–175)