WYŻSZA SZKOŁA
INFORMATYKI i ZARZĄDZANIA
z siedzibą w Rzeszowie

KOLEGIUM INFORMATYKI STOSOWANEJ

Kierunek: INFORMATYKA

Specjalność: Programowanie

Mateusz Jaworski
w67237

Elektroniczny system oceniania

Prowadzący: mgr inż. Ewa Żesławska

Praca projektowa programowanie obiekotwe C#

Rzeszów 2024

Spis treści

Wstęp

W dzisiejszym dynamicznym środowisku edukacyjnym napotykamy na liczne wyzwania związane z procesem oceniania oraz zarządzaniem wynikami uczniów. Często zdarza się, że tradycyjne metody oceniania stają się nieefektywne lub niepraktyczne w obszarze nowoczesnych technologii. Nasz elektroniczny system oceniania nie tylko integruje nauczycieli, uczniów i rodziców w procesie monitorowania postępów edukacyjnych, ale także dostarcza im nowoczesne narzędzie do skutecznego śledzenia i analizy wyników nauki. Dzięki temu rozwiązaniu każdy uczestnik procesu edukacyjnego ma szansę aktywnie uczestniczyć w doskonaleniu procesu nauczania i uczenia się, jednocześnie czerpiąc korzyści z wykorzystania nowoczesnych technologii. Projekt naszego zespołu nie tylko zapewnia efektywną platformę do oceniania i monitorowania postępów uczniów, ale także dostarcza narzędzia do zarządzania uczniami, nauczycielami, ocenami oraz innymi danymi potrzenymi do stworzenia dobrego systemu. Dzięki temu unikalnemu podejściu, każdy uczestnik procesu edukacyjnego może przyczynić się do stworzenia bardziej dynamicznego i skutecznego systemu nauczania, jednocześnie rozwijając umiejętności i motywację do osiągania coraz lepszych rezultatów.

Rozdział 1

Opis założeń projektu

1.1 Cele projetu

Mój projekt ma na celu stworzenie nowoczesnego systemu elektronicznego oceniania, który będzie wspierał proces nauczania i uczenia się oraz ułatwiał zarządzanie wynikami uczniów. Głównym problemem, który będzie rozwiązywany, jest konieczność usprawnienia tradycyjnych metod oceniania i zarządzania wynikami edukacyjnymi, w dzisiejszym środowisku edukacyjnym, które coraz bardziej staje się uzależnione od nowoczesnych technologii. Podstawowym źródłem tego problemu jest brak efektywnych narzędzi do monitorowania postępów uczniów oraz analizy wyników nauki, co utrudnia efektywne zarządzanie procesem nauczania. Problem ten jest ważny z kilku powodów. Po pierwsze, tradycyjne metody oceniania często nie są wystarczająco elastyczne ani efektywne w dzisiejszym zróżnicowanym środowisku edukacyjnym, co prowadzi do niesprawiedliwych ocen i frustracji uczniów oraz nauczycieli. Po drugie, coraz większe wymagania stawiane przed systemem edukacji oraz rosnąca konkurencja na rynku pracy sprawiają, że istnieje pilna potrzeba skutecznego monitorowania postępów uczniów i wspierania ich rozwoju edukacyjnego. Ten projekt zakłada stworzenie aplikacji internetowej, która umożliwi nauczycielom wprowadzanie ocen, prowadzenie dziennika lekcyjnego oraz wprowadzanie i aktualizację danych. Efektem projektu będzie system elektronicznego oceniania, który usprawni proces nauczania i uczenia się oraz poprawi jakość edukacji.

1.2 Wymagania funkcjonale i niefunkcjonalne

Wymagania funkcjonalne

- Dodawanie ocen: Nauczyciele mają możliwość dodawania ocen dla uczniów z poszczególnych przedmiotów.
- Przeglądanie ocen: Możliwość przeglądania ocen i śledzenia postępów edukacyjnych ucznia.
- Zarządzanie przedmiotami: Nauczyciele mają możliwość zarządzania listą przedmiotów.
- Zarządzanie uczniami: Możiwość dodawania, usuwania bądź aktualizowania listy uczniów.
- Zarządzanie nauczycielami: Możliwość dodawania, usuwania bądź aktualizowania listy nauczycieli.
- Wyświetlanie ocen końcowych: Możliwość wyświetlania oceny końcowej z danego przedmioty na podstawie ocen.
- Zarządzanie opiekunami danego ucznia: Możliwość dodawania, usuwania bądź aktualizowania danych o poszczególnym rodzicu danego ucznia.
- Aktualizowanie oraz usuwanie ocen cząstkowych.

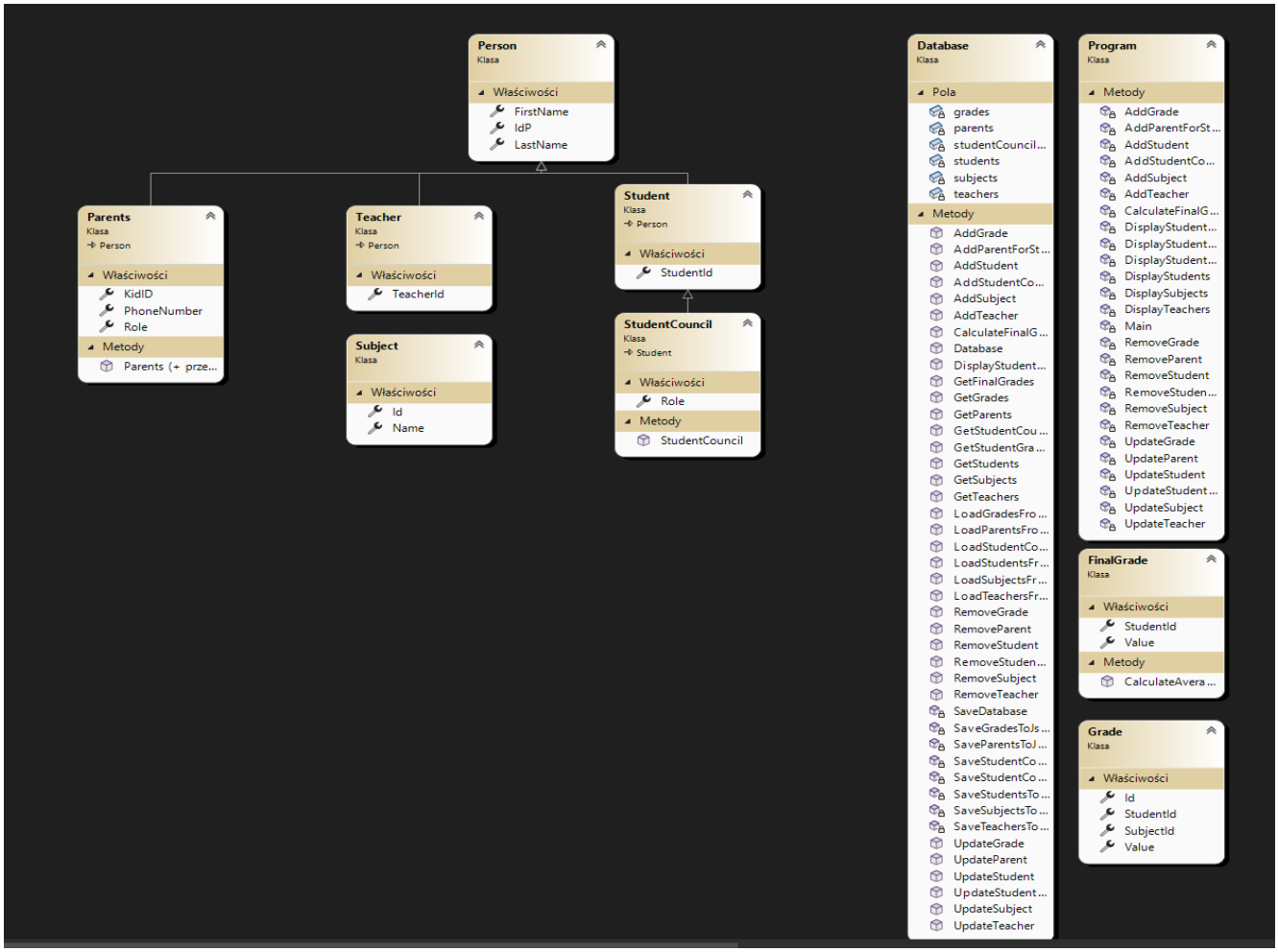
Wymagania niefunkcjonalne

- Optymalizacja kosztów: Koszty wdrożenia i utrzymania systemu są zoptymalizowane.
- Dostępność: System powinien jest dostępny 24/7, zapewniając użytkownikom możliwość korzystania z niego w dowolnym momencie.
- Intuicyjność interfejsu: Interfejs użytkownika jest intuicyjny i łatwy w obsłudze, nawet dla mniej zaawansowanych technologicznie użytkowników.
- Skalowalność: System powinien jest łatwo skalowalny, umożliwiając dostosowanie się do zmieniającej się liczby użytkowników i potrzeb biznesowych.
- Wydajność: System działa sprawnie nawet przy dużej liczbie wprowadzonych użytkowników oraz zapewnia szybkie odpowiedzi na żądania.

Rozdział 2

Opis struktury projektu

2.1 Diagram klas



Rysunek 2.1: Diagram klas

Diagram klas tej aplikacji przedstawia strukturę klas oraz ich wzajemne relacje. Zawiera klasy reprezentujące różne elementy systemu, takie jak osoba, uczniowie, nauczyciele, przedmioty, oceny, członkowie samorządu, rodzice. Relacje między klasami są określone za pomocą dziedziczenia, gdzie w przypadku tej aplikacji klasy rodzice, nauczyciel oraz uczeń dziedziczą po klasie osoba. Dodatkowo klasa samorząd uczniowski dziedzicy po klasie uczeń. Diagram klas pozwala na zrozumienie struktury systemu oraz relacji między poszczególnymi elementami.

2.2 Język programowania:

Projekt został zaimplementowany w języku C Sharp.

2.3 Narzędzia:

- Środowisko programistyczne: Visual Studio
- Framework: .NET Framework
- Biblioteki: Newtonsoft.Json do obsługi operacji na plikach JSON

2.4 Minimalne wymagania sprzętowe:

- Procesor: 1 GHz
- Pamięć RAM: 1 GB dla systemu 32-bitowego, 2 GB dla systemu 64-bitowego
- Wolne miejsce na dysku: 4 GB
- System operacyjny: Windows 7 lub nowszy

2.5 Zarządzanie danymi i baza danych:

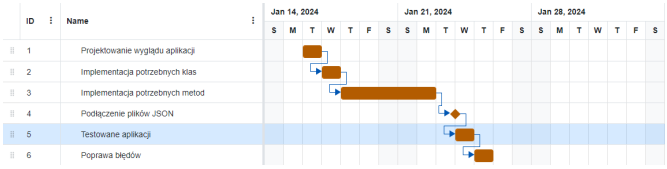
Do przechowywania danych wykorzystywane są pliki w formacie JSON. Każda klasa danych (uczniowie, nauczyciele, przedmioty, oceny, rodzice) ma odpowiadającą mu strukturę w plikach JSON, co umożliwia łatwe odczytywanie i zapisywanie danych.

Rozdział 3

Harmonogram realizacji projektu

W rozdziale tym nalezy umieścić harmonogram realizacji projektu – diagram Ganta. Rysunek oraz krótki opis do niego. Można napisać jakie problemy trudności wystąpiły w trakcie realizacji projektu. Rozdział ten musi zawierać informacje o repozytorium i systemie kontroli wersji.

3.1 Diagram Gantta



Rysunek 3.1: Diagram Gantta

3.2 Repozytorium

Repozytorium: https://github.com/MateuszJaworski7/ProjektPO

3.3 System kontroli wersji

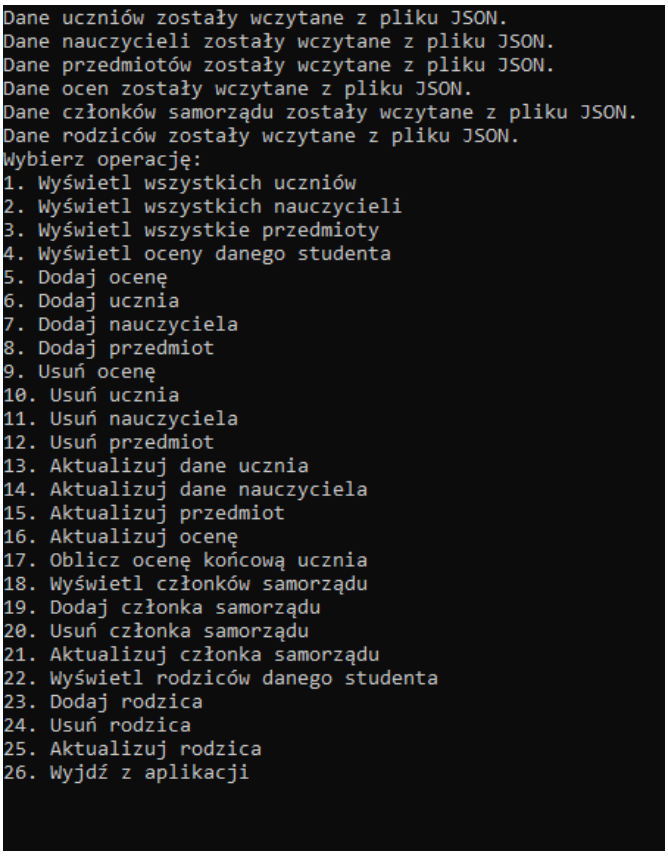
Systemem kontroli wersji tej aplikacji to Git. Git jest bardzo popularnym systemem kontroli wersji, który jest szeroko stosowany w projektach programistycznych ze względu na swoją wydajność, elastyczność i możliwość współpracy zespołowej.

Rozdział 4

Prezentacja warstwy użytkowej projektu

4.1 Wyświetlanie danych

4.1.1 Pierwsze uruchomienie programu

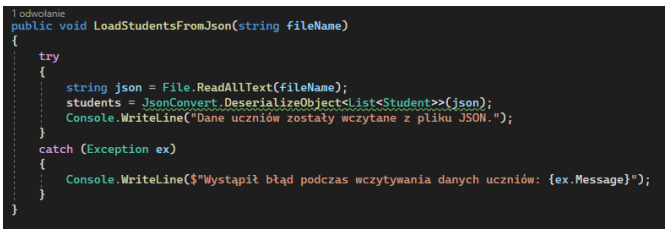


Rysunek 4.1: Wynik konsoli po uruchomieniu programu

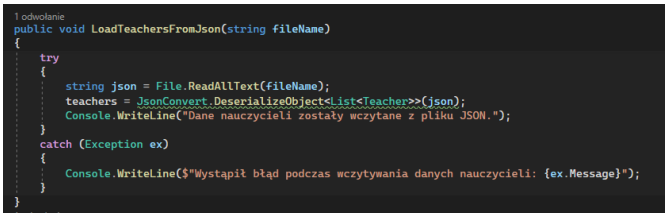
Powyższy rysunek przedstawia wynik konsoli po uruchomieniu aplikacji. Informuje nas o poprawnym wczytaniu danych do naszej aplikacji oraz prezentuje możliwe opcje programu, które możemy wybrać.

4.1.2 Metody wczytywania danych oraz zapisywania danych.

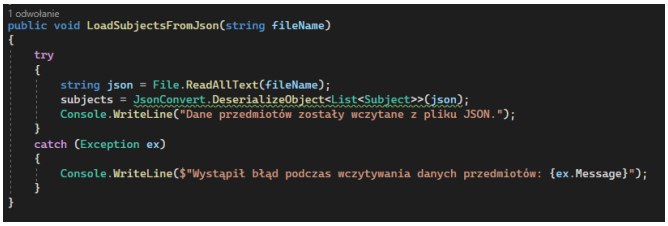
Metody te używane są do zapisywania danych w plikach json, bądź odczytywania danych z tych plików.



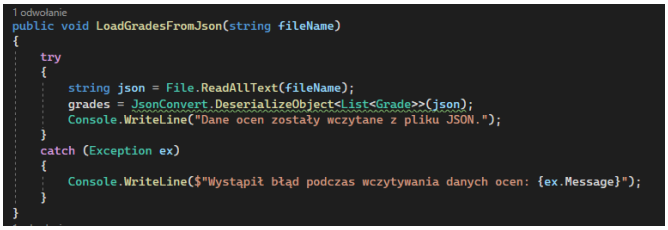
Rysunek 4.2: Metoda wczytywania danych uczniów.



Rysunek 4.3: Metoda wczytywania danych nauczycieli.

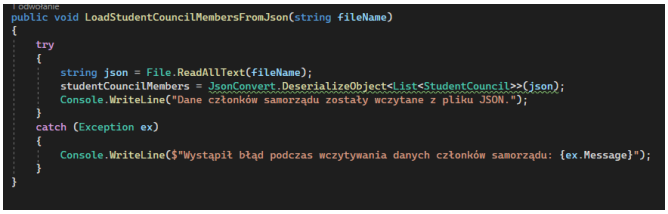


Rysunek 4.4: Metoda wczytywania danych przedmiotów.



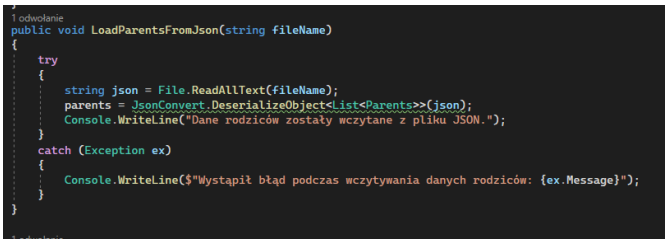
```
1 odwołanie
public void LoadGradesFromJson(string fileName)
{
    try
    {
        string json = File.ReadAllText(fileName);
        grades = JsonConvert.DeserializeObject<List<Grade>>(json);
        Console.WriteLine("Dane ocen zostały wczytane z pliku JSON.");
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Wystąpił błąd podczas wczytywania danych ocen: {ex.Message}");
    }
}
```

Rysunek 4.5: Metoda wczytywania dancyh ocen.



```
public void LoadStudentCouncilMembersFromJson(string fileName)
{
    try
    {
        string json = File.ReadAllText(fileName);
        studentCouncilMembers = JsonConvert.DeserializeObject<List<StudentCouncil>>(json);
        Console.WriteLine("Dane członków samorządu zostały wczytane z pliku JSON.");
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Wystąpił błąd podczas wczytywania danych członków samorządu: {ex.Message}");
    }
}
```

Rysunek 4.6: Metoda wczytywania danych członków samorządu.



```
1 odwołanie
public void LoadParentsFromJson(string fileName)
{
    try
    {
        string json = File.ReadAllText(fileName);
        parents = JsonConvert.DeserializeObject<List<Parents>>(json);
        Console.WriteLine("Dane rodziców zostały wczytane z pliku JSON.");
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Wystąpił błąd podczas wczytywania danych rodziców: {ex.Message}");
    }
}
```

Rysunek 4.7: Metoda wczytywania danych rodziców.

```
Odwołania: 3
private void SaveSubjectsToJson(string fileName)
{
    string json = JsonConvert.SerializeObject(subjects);
    File.WriteAllText(fileName, json);
}


Odwołania: 3
private void SaveStudentsToJson(string fileName)
{
    string json = JsonConvert.SerializeObject(students);
    File.WriteAllText(fileName, json);
}

Odwołania: 3
private void SaveTeachersToJson(string fileName)
{
    string json = JsonConvert.SerializeObject(teachers);
    File.WriteAllText(fileName, json);
}

Odwołania: 3
private void SaveGradesToJson(string fileName)
{
    string json = JsonConvert.SerializeObject(grades);
    File.WriteAllText(fileName, json);
}
1 odwołanie
private void SaveStudentCouncilToJson(string fileName)
{
    string json = JsonConvert.SerializeObject(studentCouncilMembers);
    File.WriteAllText(fileName, json);
}
Odwołania: 3
private void SaveParentsToJson(string fileName)
{
    string json = JsonConvert.SerializeObject(parents, Newtonsoft.Json.Formatting.Indented);
    File.WriteAllText(fileName, json);
}
```

Rysunek 4.8: Metody zapisywania danych w klasie database.

4.1.3 Wyświetlanie uczniów

Metoda ta pozwala wyświetlić wszystkich uczniów ich immiona oraz nazwiska, a także ich numer albumu(ID).

```
1 odwołanie
static void DisplayStudents(Database database)
{
    Console.WriteLine("Uczniowie:");
    List<Student> students = database.GetStudents();
    foreach (var student in students)
    {
        Console.WriteLine($"ID: {student.IdP}, FirstName: {student.FirstName}, LastName: {student.LastName}, Numer albumu: {student.StudentId}");
    }
}
```

Rysunek 4.9: Kod odpowiedzialny za wyświetlenie uczniów.

```
1 odwołanie
public List<Student> GetStudents()
{
    return students;
}
```

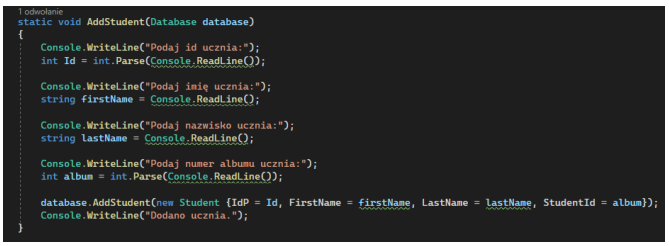
Rysunek 4.10: Tworzenie listy studentów.

```
1
Uczniowie:
ID: 101, FirstName: Andrzej, LastName: Yamal, Numer albumu: 101
ID: 102, FirstName: Michael, LastName: Johnson, Numer albumu: 102
ID: 103, FirstName: David, LastName: Brown, Numer albumu: 103
ID: 104, FirstName: Matthew, LastName: Garcia, Numer albumu: 104
ID: 105, FirstName: Daniel, LastName: Hernandez, Numer albumu: 105
```

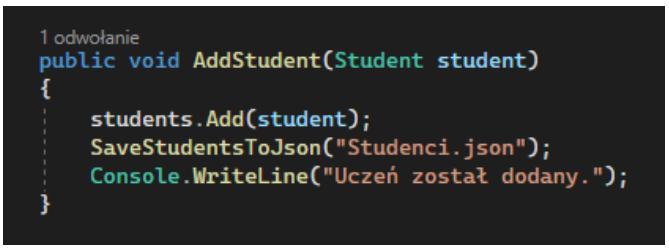
Rysunek 4.11: Wynik operacji wyświetl wszystkich uczniów.

4.1.4 Dodawanie ucznia.

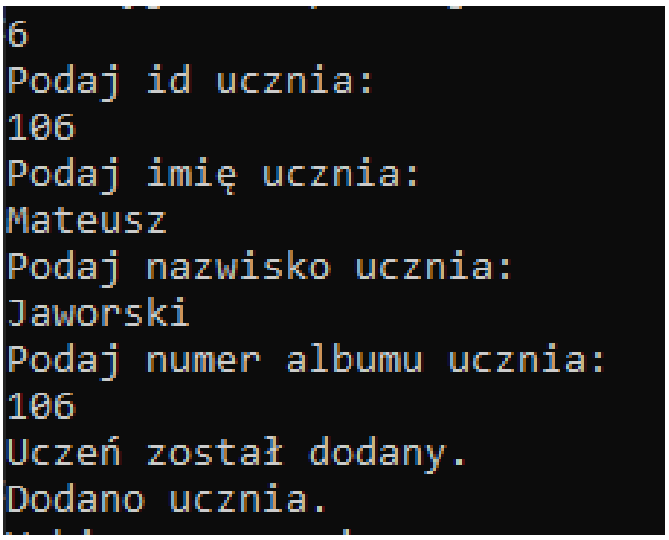
Metoda ta pozwala dodać nam ucznia pobierając od uzytkownika imię, nazwisko oraz numer albumu. Dane te są później zapisywane w pliku tekstowym json.



Rysunek 4.12: Metoda wywoływana w klasie głównej aplikacji.



Rysunek 4.13: Dodawanie ucznia w klasie database



Rysunek 4.14: Wynik operacji dodawania ucznia w konsoli.

4.1.5 Usuwanie ucznia

Metoda ta pozwala usunąć ucznia z pliku json pobierając od użytkownika numer albumu ucznia.

```
1 odwołanie
public void RemoveStudent(int studentId)
{
    Student studentToRemove = students.Find(s => s.IdP == studentId);
    if (studentToRemove != null)
    {
        students.Remove(studentToRemove);
        SaveStudentsToJson("Studenci.json");
        Console.WriteLine("Uczeń został usunięty.");
    }
    else
    {
        Console.WriteLine("Nie znaleziono ucznia o podanym identyfikatorze.");
    }
}
```

Rysunek 4.15: Metoda usuwania ucznia w klasie database.

```
1 odwołanie
static void RemoveStudent(Database database)
{
    Console.WriteLine("Podaj ID ucznia:");
    int studentId = int.Parse(Console.ReadLine());

    database.RemoveStudent(studentId);
    Console.WriteLine("Usunięto ucznia.");
}
```

Rysunek 4.16: Metoda wywoływana w klasie głównej.

```
10
Podaj ID ucznia:
106
Uczeń został usunięty.
Usunięto ucznia.
```

Rysunek 4.17: Wynik operacji usuwania ucznia.

4.1.6 Aktualizacja danych ucznia.

Metoda ta służy do aktualizacji danych ucznia takich jak imię, nazwisko pobierając od uzytkownika numer albumu.

```
1 odwołanie
static void UpdateStudent(Database database)
{
    Console.WriteLine("Podaj numer albumu ucznia do zaktualizowania:");
    int studentId = int.Parse(Console.ReadLine());

    Console.WriteLine("Podaj nowe imię ucznia:");
    string firstName = Console.ReadLine();

    Console.WriteLine("Podaj nowe nazwisko ucznia:");
    string lastName = Console.ReadLine();

    Student updatedStudent = new Student { StudentId = studentId, FirstName = firstName, LastName = lastName };

    database.UpdateStudent(updatedStudent);
}
```

Rysunek 4.18: Kod w klasie głównej.

```
1 odwołanie
public void UpdateStudent(Student updatedStudent)
{
    Student existingStudent = students.FirstOrDefault(s => s.StudentId == updatedStudent.StudentId);
    if (existingStudent != null)
    {
        existingStudent.FirstName = updatedStudent.FirstName;
        existingStudent.LastName = updatedStudent.LastName;
        SaveStudentsToJson("Studenci.json");
        Console.WriteLine("Dane ucznia zostały zaktualizowane.");
    }
    else
    {
        Console.WriteLine("Nie znaleziono ucznia o podanym numerze albumu.");
    }
}
```

Rysunek 4.19: Metoda w klasie database.

```
13
Podaj numer albumu ucznia do zaktualizowania:
101
Podaj nowe imię ucznia:
Andrzej
Podaj nowe nazwisko ucznia:
Yamal
Dane ucznia zostały zaktualizowane.
```

Rysunek 4.20: Wynik operacji aktualizowania danych ucznia.

4.1.7 Wyświetlanie nauczycieli

Metoda pozwalająca wyświetlić nam dane nauczycieli znajdujących sie w pliku tekstowym.

```
1 odwołanie
static void DisplayStudents(Database database)
{
    Console.WriteLine("Uczniowie:");
    List<Student> students = database.GetStudents();
    foreach (var student in students)
    {
        Console.WriteLine($"ID: {student.IdP}, FirstName: {student.FirstName}, LastName: {student.LastName}, Numer albumu: {student.StudentId}");
    }
}
```

Rysunek 4.21: Kod w klasie głównej.

```
1 odwołanie
public List<Teacher> GetTeachers()
{
    return teachers;
}
```

Rysunek 4.22: Kod w klasie database.

```
2
Nauczyciele:
ID: 201, FirstName: Edyta, LastName: Górniak, Numer nauczyciela: 201
ID: 202, FirstName: Emily, LastName: Williams, Numer nauczyciela: 202
ID: 203, FirstName: Sarah, LastName: Jones, Numer nauczyciela: 203
ID: 204, FirstName: Jessica, LastName: Martinez, Numer nauczyciela: 204
ID: 205, FirstName: Amanda, LastName: Lopez, Numer nauczyciela: 205
```

Rysunek 4.23: wynik operacji wyświetlania nauczycieli.

4.1.8 Dodawanie nauczyciela

Metoda ta pozwala nam na dodanie oraz zapisanie w pliku tekstowym json nowego nauczyciela pobierając od uzytkownika imię, nazwisko oraz numer nauczyciela.

```
1 odwołanie
public void AddTeacher(Teacher teacher)
{
    teachers.Add(teacher);
    SaveTeachersToJson("Nauczyciele.json");
    Console.WriteLine("Nauczyciel został dodany.");
}
```

Rysunek 4.24: Dodawanie ucznia w klasie database.

```
1 odwołanie
static void AddTeacher(Database database)
{
    Console.WriteLine("Podaj id nauczyciela:");
    int Id = int.Parse(Console.ReadLine());

    Console.WriteLine("Podaj imię nauczyciela:");
    string firstName = Console.ReadLine();

    Console.WriteLine("Podaj nazwisko nauczyciela:");
    string lastName = Console.ReadLine();

    Console.WriteLine("Podaj numer nauczyciela:");
    int numer = int.Parse(Console.ReadLine());

    database.AddTeacher(new Teacher {IdP = Id,  FirstName = firstName, LastName = lastName, TeacherId = numer });
    Console.WriteLine("Dodano nauczyciela.");
}
```

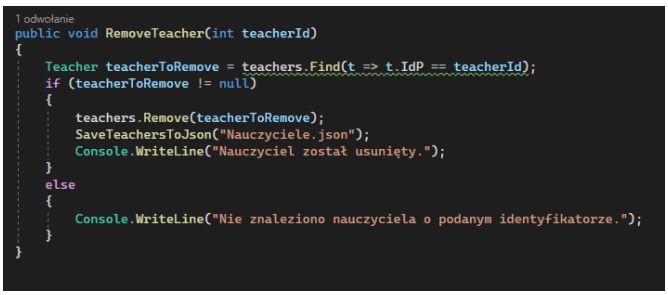
Rysunek 4.25: Kod metody w klasie głównej.

```
7
Podaj id nauczyciela:
207
Podaj imię nauczyciela:
Krystian
Podaj nazwisko nauczyciela:
Kowalski
Podaj numer nauczyciela:
207
Nauczyciel został dodany.
Dodano nauczyciela.
```

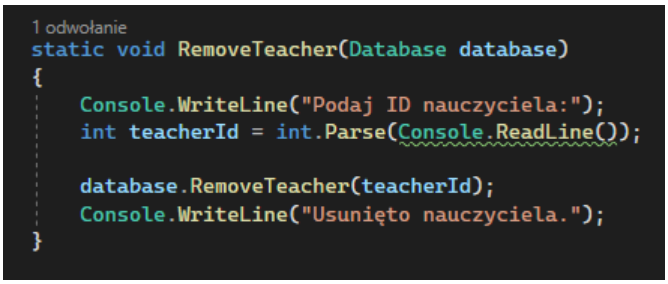
Rysunek 4.26: Wynik operacji dodawania nauczyciela.

4.1.9 Usuwanie nauczyciela

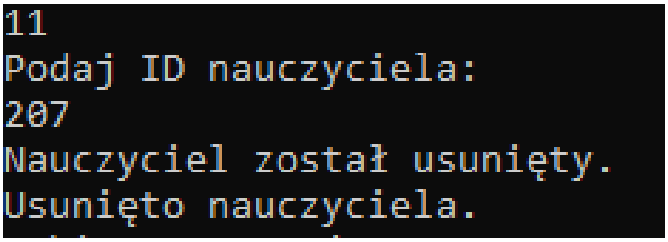
Metoda ta pozwala nam na usunięcie istniejącego nauczyciela z bazy poprzez podanie jego numeru nauczyciela.



Rysunek 4.27: Kod w klasie database.



Rysunek 4.28: Metoda usuwająca nauczyciela w klasie głównej.



Rysunek 4.29: wynik operacji usuwania nauczyciela.

4.1.10 Aktualizowanie danych nauczyciela

Metoda ta pozwala nam zaktualizować dane istniejącego już nauczyciela takie jak imię, nazwisko pobierając od uzytkownika numer nauczyciela.

```
1 odwołanie
public void UpdateTeacher(Teacher updatedTeacher)
{
    Teacher existingTeacher = teachers.Find(t => t.TeacherId == updatedTeacher.TeacherId);

    if (existingTeacher != null)
    {
        existingTeacher.FirstName = updatedTeacher.FirstName;
        existingTeacher.LastName = updatedTeacher.LastName;
        SaveTeachersToJson("Nauczyciele.json");
        Console.WriteLine("Dane nauczyciela zostały zaktualizowane.");
    }
    else
    {
        Console.WriteLine("Nie znaleziono nauczyciela o podanym identyfikatorze.");
    }
}
```

Rysunek 4.30: Kod metody w klasie database.

```
1 odwołanie
static void UpdateTeacher(Database database)
{
    Console.WriteLine("Podaj ID nauczyciela do zaktualizowania:");
    int teacherId = int.Parse(Console.ReadLine());

    Console.WriteLine("Podaj nowe imię nauczyciela:");
    string firstName = Console.ReadLine();

    Console.WriteLine("Podaj nowe nazwisko nauczyciela:");
    string lastName = Console.ReadLine();

    Teacher updatedTeacher = new Teacher { TeacherId = teacherId, FirstName = firstName, LastName = lastName };

    database.UpdateTeacher(updatedTeacher);
}
```

Rysunek 4.31: Metoda aktualizowania danych nauczyciela w klasie głównej.

```
14
Podaj ID nauczyciela do zaktualizowania:
201
Podaj nowe imię nauczyciela:
Mateusz
Podaj nowe nazwisko nauczyciela:
Jaworski
Dane nauczyciela zostały zaktualizowane.
```

Rysunek 4.32: Wynik metody aktualizowania danych nauczyciela.

4.1.11 Wyświetlanie ocen danego studenta

Metoda ta pobiera od użytkownika numer albumu danego studenta, a następnie wyświetla oceny tego ucznia.

```
1 odwołanie
static void DisplayStudentGrades(Database database)
{
    Console.WriteLine("Podaj numer albumu studenta:");
    int studentId = int.Parse(Console.ReadLine());

    List<Grade> studentGrades = database.GetStudentGrades(studentId);
    if (studentGrades.Count > 0)
    {
        Console.WriteLine($"Oceny studenta o numerze albumu {studentId}:");
        foreach (var grade in studentGrades)
        {
            Console.WriteLine($"Subject ID: {grade.SubjectId}, ID oceny: {grade.Id}, Value: {grade.Value}");
        }
    }
    else
    {
        Console.WriteLine($"Brak ocen dla studenta o ID {studentId}.");
    }
}
```

Rysunek 4.33: Kod metody w klasie głównej.

```
1 odwołanie
public List<Grade> GetStudentGrades(int studentId)
{
    List<Grade> studentGrades = grades.Where(g => g.StudentId == studentId).ToList();
    return studentGrades;
}
```

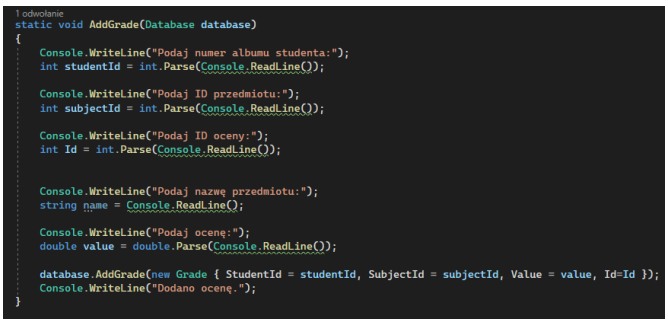
Rysunek 4.34: Metoda pozwalajaca pozyskać oceny w klasie database.

```
4
Podaj numer albumu studenta:
101
Oceny studenta o numerze albumu 101:
Subject ID: 1, ID oceny: 11013, Value: 4
Subject ID: 1, ID oceny: 11014, Value: 4
Subject ID: 1, ID oceny: 11015, Value: 3
Subject ID: 1, ID oceny: 11016, Value: 3,5
```

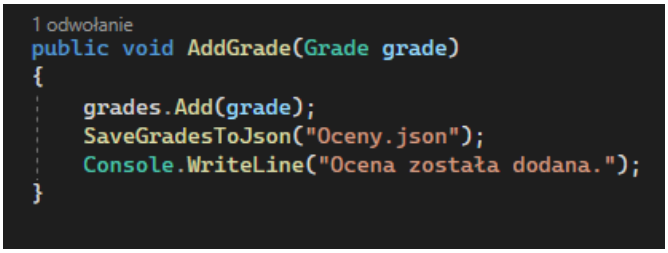
Rysunek 4.35: Wynik metody wyświetlenia ocen danego studenta.

4.1.12 Dodawanie oceny

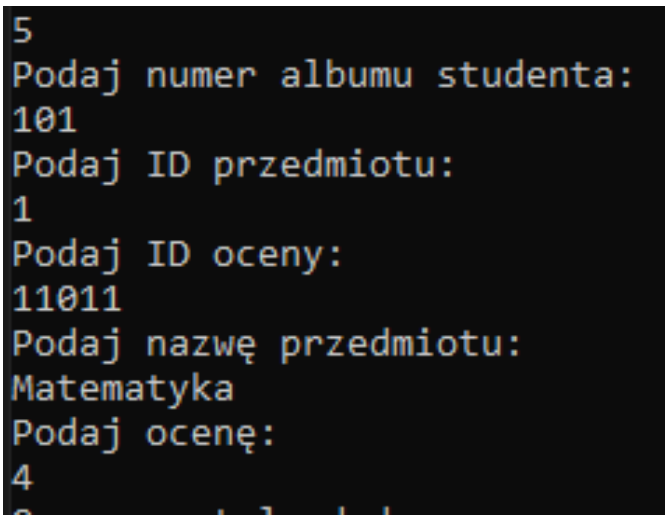
Metoda ta pozwala nam dodać ocenę danemu uczniowi poprzez podanie numeru albumu studenta, id przedmiotu, id oceny, nazwy przedmiotu oraz wartości oceny.



Rysunek 4.36: Metoda dodawania oceny w klasie głównej.



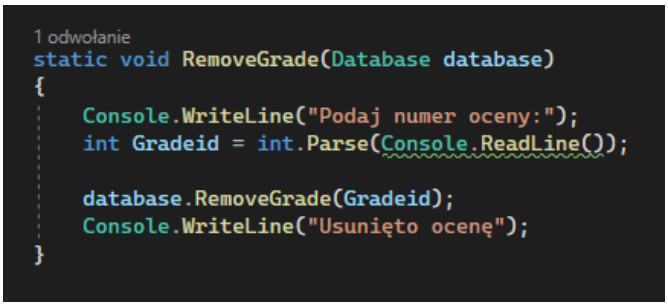
Rysunek 4.37: Kod metody w klasie database.



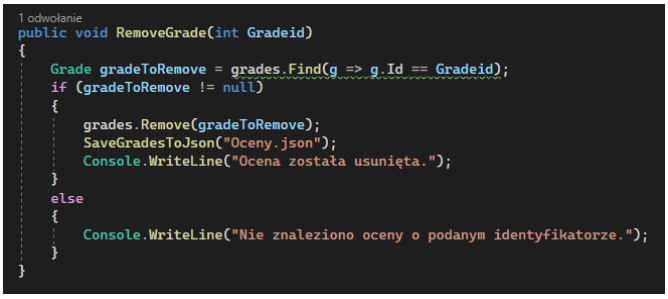
Rysunek 4.38: Wynik operacji dodawania oceny.

4.1.13 Usuwanie oceny

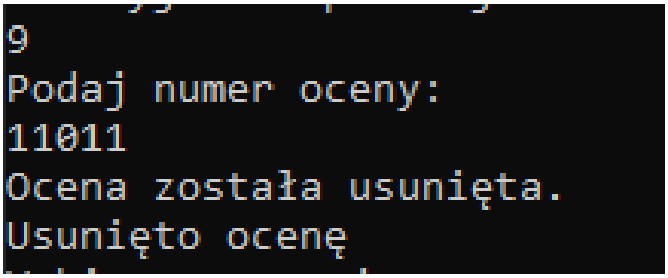
Metoda ta usuwa ocenę pobierając od uzytkownika id oceny.



Rysunek 4.39: Metoda usuwania oceny w klasie głównej.



Rysunek 4.40: Metoda usuwania oceny w klasie database.



Rysunek 4.41: Wynik operacji usuwania oceny.

4.1.14 Aktualizowanie oceny

Metoda ta pozwala nam zaktualizować istniejącą juz ocenę poprzez pobranie id oceny. Następnie użytkownik podaje nowe dane takie jak ID studenta, ID przedmiotu oraz wartość oceny.

```
1 odwołanie
static void UpdateGrade(Database database)
{
    Console.WriteLine("Podaj ID oceny do aktualizacji:");
    int gradeId = int.Parse(Console.ReadLine());

    Console.WriteLine("Podaj nowe ID studenta:");
    int studentId = int.Parse(Console.ReadLine());

    Console.WriteLine("Podaj nowe ID przedmiotu:");
    int subjectId = int.Parse(Console.ReadLine());

    Console.WriteLine("Podaj nową wartość oceny:");
    double value = double.Parse(Console.ReadLine());

    Grade updatedGrade = new Grade
    {
        Id = gradeId,
        StudentId = studentId,
        SubjectId = subjectId,
        Value = value
    };

    database.UpdateGrade(updatedGrade);
}
```

Rysunek 4.42: Kod metody aktualizowania oceny w klasie głównej.

```
1 odwołanie
public void UpdateGrade(Grade updatedGrade)
{
    Grade gradeToUpdate = grades.Find(g => g.Id == updatedGrade.Id);

    if (gradeToUpdate != null)
    {
        gradeToUpdate.StudentId = updatedGrade.StudentId;
        gradeToUpdate.SubjectId = updatedGrade.SubjectId;
        gradeToUpdate.Value = updatedGrade.Value;

        SaveGradesToJson("Oceny.json");
        Console.WriteLine("Ocena została zaktualizowana.");
    }
    else
    {
        Console.WriteLine("Nie znaleziono oceny o podanym identyfikatorze.");
    }
}
```

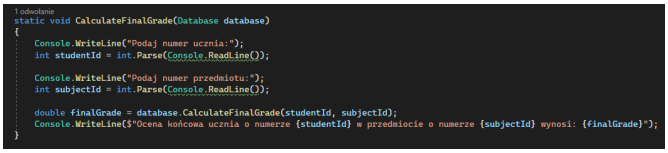
Rysunek 4.43: Metoda aktualizowania oceny w klasie databse.

```
16
Podaj ID oceny do aktualizacji:
11014
Podaj nowe ID studenta:
102
Podaj nowe ID przedmiotu:
2
Podaj nową wartość oceny:
5
Ocena została zaktualizowana.
```

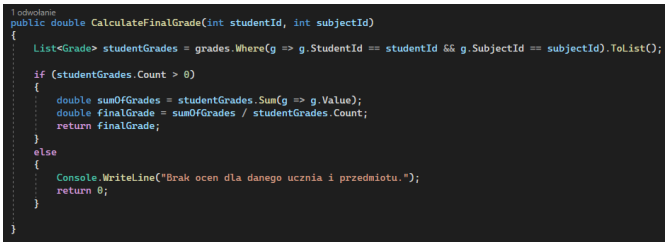
Rysunek 4.44: Wynik operacji aktualizowania oceny.

4.1.15 Obliczanie oceny końcowej

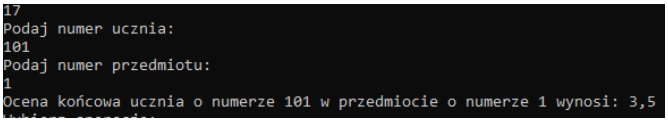
Metoda ta pozwala obliczyć ocenę końcową na podstawie wszystkich ocen z danego przedmiotu. Pobiera ona od uzytkownika ID ucznia oraz nazwę przedmiotu.



Rysunek 4.45: Metoda obliczania oceny końcowej w klasie głównej.



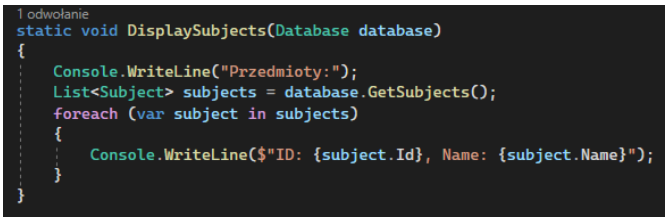
Rysunek 4.46: Metoda obliczania oceny końcowej w klasie database.



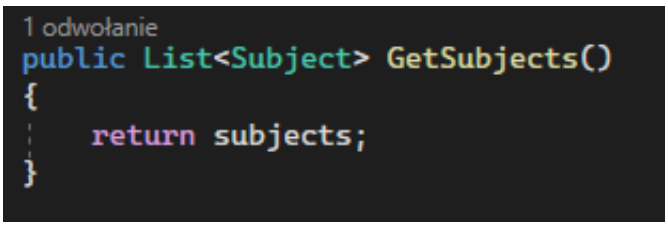
Rysunek 4.47: Wynik operacji obliczania oceny końcowej.

4.1.16 Wyświetlanie przedmiotów

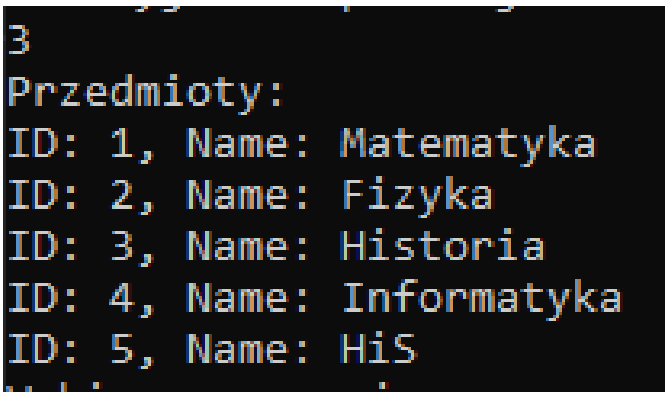
Metoda pozwalająca wyświetlić wszystkie przedmioty znajdujące się w pliku tekstowym json.



Rysunek 4.48: Metoda wyświetlania przedmiotów w klasie głównej.



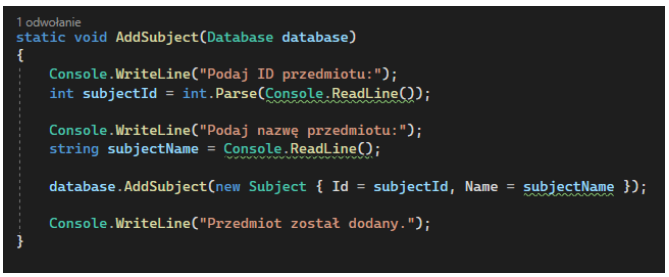
Rysunek 4.49: Metoda wyświetlania przedmiotów w klasie database.



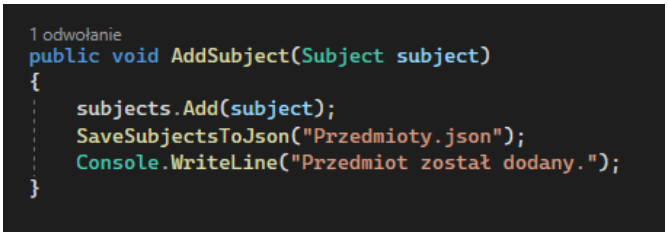
Rysunek 4.50: Wynik operacji wyświetlania przedmiotów.

4.1.17 Dodawanie przedmiotu

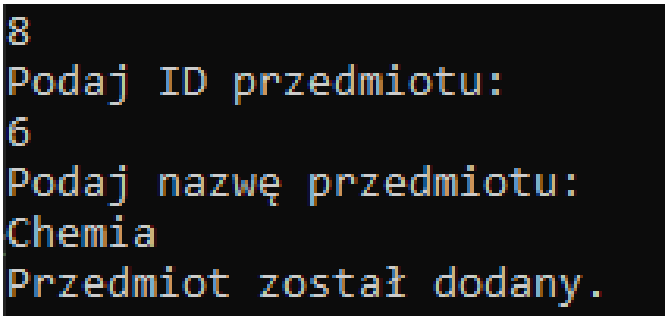
Metoda pozwalająca dodać przedmiot oraz zapisać go w pliku tekstowym. Pobiera ona od użytkownika ID przedmiotu oraz jego nazwę.



Rysunek 4.51: Metoda dodawania przedmiotu w klasie głównej.



Rysunek 4.52: Kod metody dodawania przedmiotu w klasie database.



Rysunek 4.53: Wynik operacji dodawania przedmiotu.

4.1.18 Usuwanie przedmiotu

Metoda pozwalająca usunąć przedmiot z bazy danych pobierając od użytkownika ID przedmiotu.

```
1 odwołanie
static void RemoveSubject(Database database)
{
    Console.WriteLine("Podaj ID przedmiotu do usunięcia:");
    int subjectId = int.Parse(Console.ReadLine());

    database.RemoveSubject(subjectId);

    Console.WriteLine("Przedmiot został usunięty.");
}
```

Rysunek 4.54: Kod metody usuwania przedmiotu w klasie głównej.

```
1 odwołanie
public void RemoveSubject(int subjectId)
{
    Subject subjectToRemove = subjects.Find(s => s.Id == subjectId);
    if (subjectToRemove != null)
    {
        subjects.Remove(subjectToRemove);
        SaveSubjectsToJson("Przedmioty.json");
        Console.WriteLine("Przedmiot został usunięty.");
    }
    else
    {
        Console.WriteLine("Nie znaleziono przedmiotu o podanym identyfikatorze.");
    }
}
```

Rysunek 4.55: Metoda usuwania przedmiotu w klasie database.

```
12
Podaj ID przedmiotu do usunięcia:
6
Przedmiot został usunięty.
```

Rysunek 4.56: Wynik operacji usuwania przedmiotu.

4.1.19 Aktualizowanie przedmiotu

Metoda pozwalająca zaktualizować oraz nadpisać istniejący już przedmiot poprzez pobranie ID przedmiotu od użytkownika oraz nowej nazwy przedmiotu.

```
1 odwołanie
static void UpdateSubject(Database database)
{
    Console.WriteLine("Podaj ID przedmiotu do aktualizacji:");
    int subjectId = int.Parse(Console.ReadLine());

    Console.WriteLine("Podaj nową nazwę przedmiotu:");
    string newName = Console.ReadLine();

    Subject updatedSubject = new Subject { Id = subjectId, Name = newName };

    database.UpdateSubject(updatedSubject);
}
```

Rysunek 4.57: Metoda aktualizowania przedmiotu w klasie głównej.

```
1 odwołanie
public void UpdateSubject(Subject updatedSubject)
{
    Subject existingSubject = subjects.Find(s => s.Id == updatedSubject.Id);

    if (existingSubject != null)
    {
        existingSubject.Name = updatedSubject.Name;
        SaveSubjectsToJson("Przedmioty.json");
        Console.WriteLine("Dane przedmiotu zostały zaktualizowane.");
    }
    else
    {
        Console.WriteLine("Nie znaleziono przedmiotu o podanym identyfikatorze.");
    }
}
```

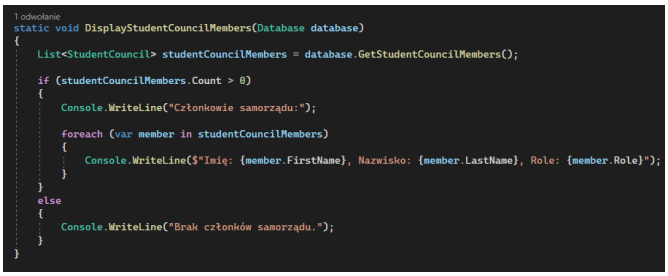
Rysunek 4.58: Kod metody aktualizowania przedmiotu w klasie database.

```
15
Podaj ID przedmiotu do aktualizacji:
5
Podaj nową nazwę przedmiotu:
Biologia
Dane przedmiotu zostały zaktualizowane.
```

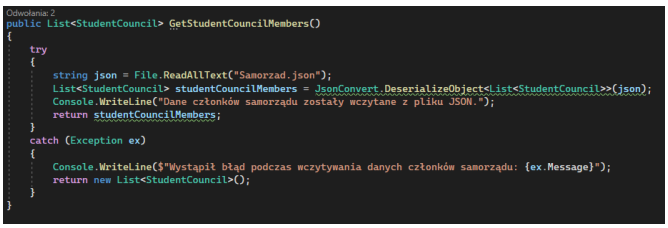
Rysunek 4.59: Wynik operacji aktualizowania przedmiotu.

4.1.20 Wyświetlanie członków samorządu uczniowskiego

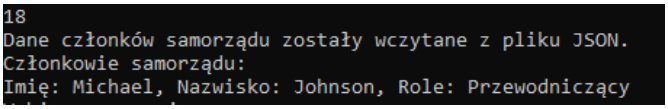
Metoda pozwalająca wyświetlić członków samorządu znajdujących się w pliku tekstowym.



Rysunek 4.60: Kod metody wyświetlania członków samorządu uczniowskiego w klasie głównej.



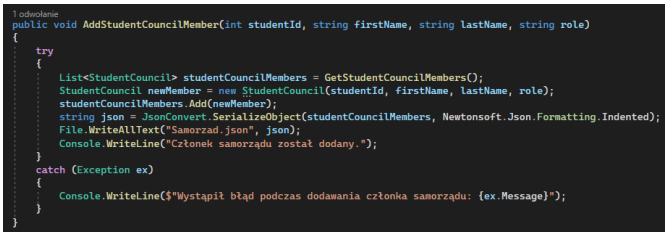
Rysunek 4.61: Metoda wyświetlania członków samorządu w klasie database.



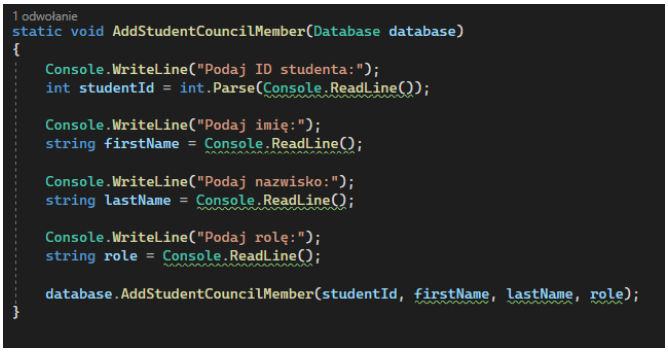
Rysunek 4.62: Wynik operacji wyświetlania członków samorządu.

4.1.21 Dodawanie członka samorządu

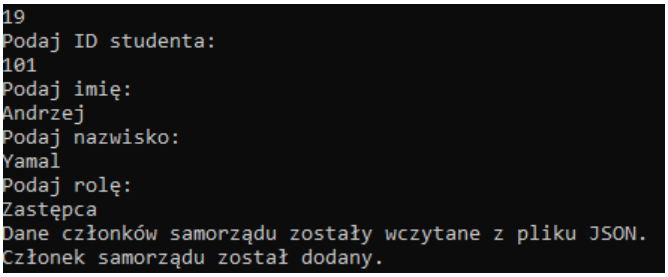
Metoda pozwalająca dodać członka samorządu oraz zapisać go w pliku tekstowym. Pobiera ona od uzytkownika ID ucznia, imię, nazwisko a także rolę jaką dana osoba ma pełnić w samorządzie uczniowskim.



Rysunek 4.63: Metoda dodawania członka samorządu w klasie database.



Rysunek 4.64: Metoda dodawania członka samorządu w klasie głównej.



Rysunek 4.65: Wynik operacji dodawania członka samorządu.

4.1.22 Usuwanie członka samorządu

Metoda pozwalająca usunąć wybranego członka z samorządu uczniowskiego poprzez pobranie od uzytkownika ID ucznia.

```
1 odwołanie
static void RemoveStudentCouncilMember(Database database)
{
    Console.WriteLine("Podaj ID członka samorządu do usunięcia:");
    int studentId = int.Parse(Console.ReadLine());

    database.RemoveStudentCouncilMember(studentId);
}
```

Rysunek 4.66: Metoda usuwania członka samorządu w klasie głównej.

```
1 odwołanie
public void RemoveStudentCouncilMember(int studentId)
{
    StudentCouncil memberToRemove = studentCouncilMembers.FirstOrDefault(m => m.StudentId == studentId);
    if (memberToRemove != null)
    {
        studentCouncilMembers.Remove(memberToRemove);
        SaveStudentCouncilToJson("Samorzad.json");
        Console.WriteLine("Członek samorządu został usunięty.");
    }
    else
    {
        Console.WriteLine("Nie znaleziono członka samorządu o podanym identyfikatorze studenta.");
    }
}
```

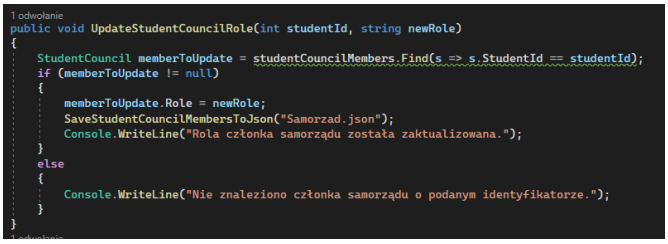
Rysunek 4.67: Kod metody usuwania członka samorządu w klasie database.

```
20
Podaj ID członka samorządu do usunięcia:
102
Członek samorządu został usunięty.
```

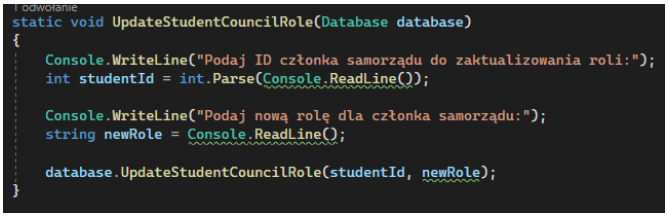
Rysunek 4.68: wynik operacji usuwania członka samorządu.

4.1.23 Aktualizowanie członka samorządu

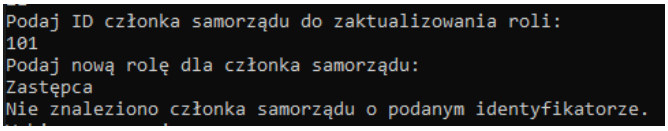
Metoda pozwalająca zaktualizować rolę danego członka samorządu. Pobiera ona od użytkownika ID ucznia, a następnie pobiera nową rolę studenta.



Rysunek 4.69: Metoda aktualizowania danych członka samorządu.



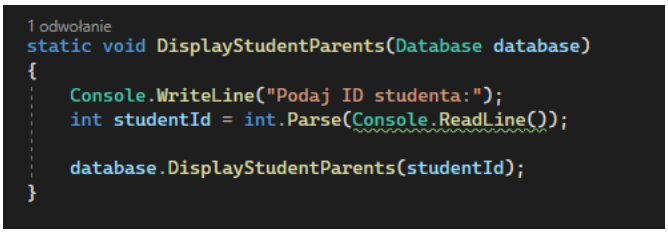
Rysunek 4.70: Kod metody aktualizowania członka samorządu w klasie głównej.



Rysunek 4.71: wynik operacji aktualizowania członka samorządu.

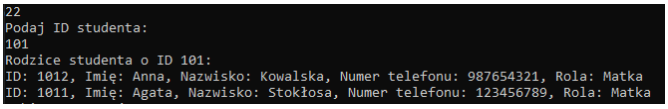
4.1.24 Wyświetlanie rodziców danego ucznia

Metoda pozwalająca wyświetlić nam dane rodziców wybranego studenta takie jak imie, nazwisko, numer telefonu oraz informacje o tym czy dana osoba jest matką czy ojcem wybranego studenta.



Rysunek 4.72: Metoda wyświetlania rodziców danego ucznia.

Rysunek 4.73: Kod metody wyświetlania rodziców danego ucznia w klasie database.

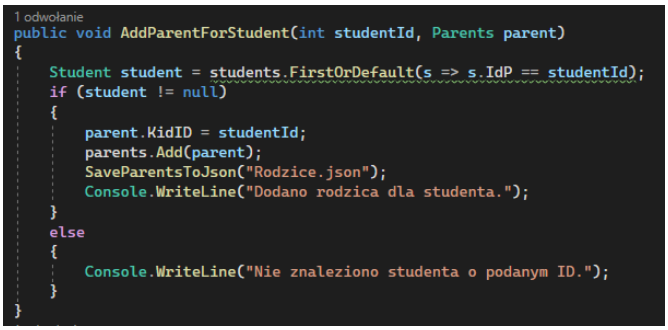


Rysunek 4.74: Wynik operacji wyświetlania rodziców danego ucznia.

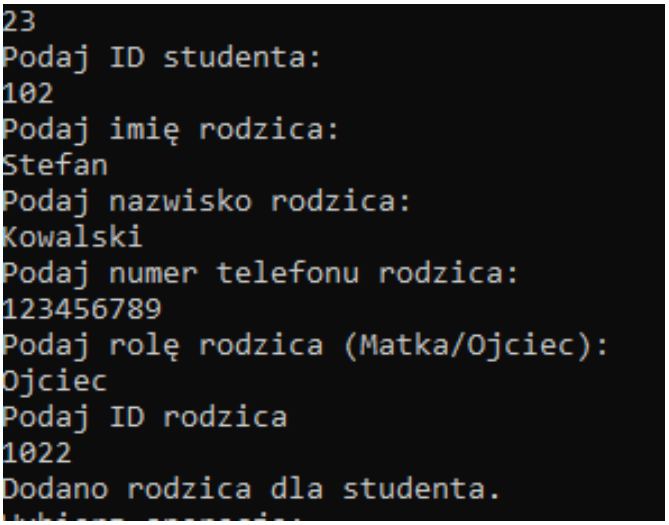
4.1.25 Dodawanie rodzica

Metoda pozwalająca na dodanie rodzica wybranemu studentowi. Pobiera ona ID ucznia od uzytkownika, a nastepnie pobiera imię, nazwisko, numer telefonu, rolę rodzica, a także ID rodzica.

Rysunek 4.75: Kod metody dodawania rodzica w klasie głównej.



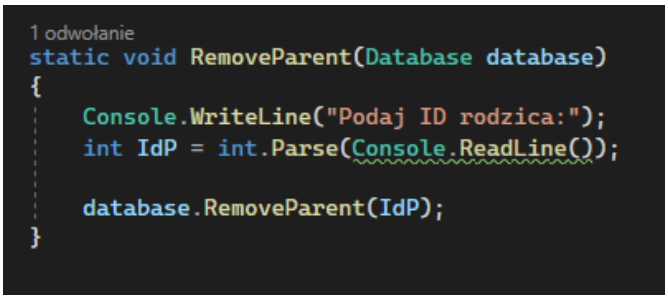
Rysunek 4.76: Metoda dodawania rodzica w klasie database.



Rysunek 4.77: wynik operacji dodawania rodzica.

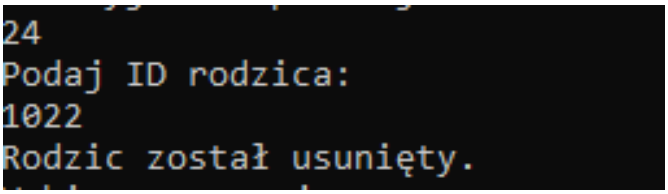
4.1.26 Usuwanie rodzica

Metoda pozwalająca nam na usunięcie istniejącego już rodzica poprzez podanie jego ID.



Rysunek 4.78: Metoda usuwania rodzica w klasie głównej.

Rysunek 4.79: Kod metody usuwania rodzica w klasie database.



Rysunek 4.80: Wynik operacji usuwania rodzica.

4.1.27 Aktualizowanie rodzica

Metoda pozwalająca nam zaktualizować dane wybranego rodzica. Metoda pobiera od użytkownika ID rodzica, a następnie pobiera nowe imię, nazwisko, numer telefonu oraz rolę.

```
1 odwołanie
static void UpdateParent(Database database)
{
    Console.WriteLine("Podaj ID rodzica do zaktualizowania:");
    int parentId = int.Parse(Console.ReadLine());

    Console.WriteLine("Podaj nowe imię rodzica:");
    string parentFirstName = Console.ReadLine();

    Console.WriteLine("Podaj nowe nazwisko rodzica:");
    string parentLastName = Console.ReadLine();

    Console.WriteLine("Podaj nowy numer telefonu rodzica:");
    string parentPhoneNumber = Console.ReadLine();

    Console.WriteLine("Podaj nową rolę rodzica (Matka/Ojciec):");
    string parentRole = Console.ReadLine();

    Parents parent = new Parents
    {
        IdP = parentId,
        FirstName = parentFirstName,
        LastName = parentLastName,
        PhoneNumber = parentPhoneNumber,
        Role = parentRole
    };

    database.UpdateParent(parent);
}
```

Rysunek 4.81: Kod metody aktualizowania danych rodzica w klasie głównej.

```
1 odwołanie
public void UpdateParent(Parents parent)
{
    Parents parentToUpdate = parents.Find(p => p.IdP == parent.IdP);
    if (parentToUpdate != null)
    {
        parentToUpdate.FirstName = parent.FirstName;
        parentToUpdate.LastName = parent.LastName;
        parentToUpdate.PhoneNumber = parent.PhoneNumber;
        parentToUpdate.Role = parent.Role;
        SaveParentsToJson("Rodzice.json");
        Console.WriteLine("Rodzic został zaktualizowany.");
    }
    else
    {
        Console.WriteLine("Nie znaleziono rodzica o podanym identyfikatorze.");
    }
}
```

Rysunek 4.82: Metoda aktualizowania danych rodzica w klasie database.

```
25
Podaj ID rodzica do zaktualizowania:
1011
Podaj nowe imię rodzica:
Antoni
Podaj nowe nazwisko rodzica:
Kowalski
Podaj nowy numer telefonu rodzica:
123456789
Podaj nową rolę rodzica (Matka/Ojciec):
Ojciec
Rodzic został zaktualizowany.
```

Rysunek 4.83: Wynik operacji aktualizowania danych rodzica.

Rozdział 5

Podsumowanie

Celem projektu było stworzenie systemu elektronicznego do zarządzania ocenami, uczniami, nauczycielami oraz członkami samorządu szkolnego. Zbudowana została stabilna platforma, która umożliwia zarządzanie danymi uczniów, nauczycieli oraz członków samorządu, co przyczynia się do lepszego funkcjonowania szkolnego środowiska. Podczas tworzenia aplikacji zostały napotkane różnorodne wyzwania, takie jak zapewnienie odpowiedniej integracji między poszczególnymi modułami aplikacji oraz utrzymanie spójności i poprawności danych. Przeszkody jednak zostały pokanane i dostarczony został produkt spełniający wymagania. Dostarczone rozwiązanie umożliwia sprawne zarządzanie ocenami, uczniami oraz nauczycielami. Interfejs użytkownika, który jest intuicyjny i łatwy w obsłudze, pozwala użytkownikom na szybkie i efektywne korzystanie z systemu.

Bibliografia

[1] https://www.youtube.com/watch?v=Y14gG9IJ230

[2] https://www.youtube.com/watch?v=w6M-Bj-tfv4

[3] https://www.youtube.com/watch?v=2CV-EDbFC-Qt=719s

Spis rysunków