WYŻSZA SZKOŁA
INFORMATYKI i ZARZĄDZANIA
z siedzibą w Rzeszowie

KOLEGIUM INFORMATYKI STOSOWANEJ

Kierunek: INFORMATYKA

Specjalność: Programowanie

Victoria Polovyy
w67964

Zarządzanie zadaniami "Planer"

Promotor: mgr inż. Ewa Żeszławska

Rzeszów 2023/2024

Spis treści

Wstęp

Stworzyłam program do nagrywania codziennych zadań. Ponieważ każdy planuje coś na swój dzień, wygodne będzie dla niego skorzystanie z aplikacji, w której w prosty i intuicyjny sposób będzie mógł rejestrować, śledzić, zmieniać i usuwać swoje plany.

Pomyślałam, że dobrze byłoby stworzyć takiego „Planera". Uważam, że Planner jest bardzo wygodny w użyciu, ponieważ sortuje wszystko w kategorie.Dzięki temu użytkownik ma pełną kontrolę nad swoimi codziennymi planami.

Daże do tego , aby moja aplikacja konsolowa była szybka i intuicyjna w planowaniu, a jednocześnie zapewniała łatwy dostęp do informacji o zadaniach.

Rozdział 1

Opis założeń projektu

1.1 Cele projektu:

Celem projektu jest stworzenie prostego systemu do zarządzania zadaniami, kategoriami i użytkownikami przy użyciu programowania obiektowego. System ten ma umożliwiać użytkownikowi dodawanie, odczytywanie, aktualizowanie i usuwanie zadań, tworzenie nowych kategorii oraz dodawanie nowych użytkowników. Dane o zadaniach są przechowywane trwale w pliku tekstowym, co pozwala na utrzymanie informacji między kolejnymi uruchomieniami programu. Projekt zakłada również zastosowanie zasad obiektowości, takich jak dziedziczenie, abstrakcja i hermetyzacja, w celu zorganizowania kodu i ułatwienia zarządzania danymi. Dodatkowo, program ma być łatwy w obsłudze poprzez interaktywne menu w konsoli, umożliwiające użytkownikowi wybór różnych opcji w intuicyjny sposób.

1.2 Wymagania funkcjonalne

- System powinien umożliwiać tworzenie nowych zadań, kategorii i użytkowników.
- Użytkownik powinien mieć możliwość odczytywania szczegółów istniejących zadań.
- System powinien wspierać aktualizację istniejących zadań, umożliwiając zmianę tytułu, opisu, kategorii oraz przypisanego użytkownika.
- Użytkownik powinien mieć możliwość usuwania zadań.
- System powinien umożliwiać tworzenie nowych kategorii, przechowując nazwy kategorii.
- System powinien umożliwiać tworzenie nowych użytkowników, przechowując ich nazwy.
- Dane o zadaniach powinny być przechowywane trwale w pliku tekstowym .
- System powinien obsługiwać odczyt i zapis danych do pliku.

1.3 Wymagania niefunkcjonalne

- **Persistencja Danych:** Dane o zadaniach powinny być trwale przechowywane w pliku tekstowym na dysku, aby umożliwić ich utrzymanie między kolejnymi uruchomieniami programu.
- **Interaktywne Menu:** Program powinien dostarczać interaktywne menu w konsoli, ułatwiające użytkownikowi korzystanie z różnych funkcji systemu.
- **Walidacja Danych:** Podczas odczytu danych z pliku, program powinien przeprowadzać walidację, aby uniknąć błędów związanych z nieprawidłowymi danymi.

- **Łatwość Użycia:** Interfejs użytkownika powinien być intuicyjny, umożliwiając łatwe korzystanie z funkcji programu.
- **Obsługa Błędów:** Program powinien obsługiwać sytuacje błędne i informować użytkownika o ewentualnych problemach podczas interakcji z systemem.

Rozdział 2

Struktura projektu

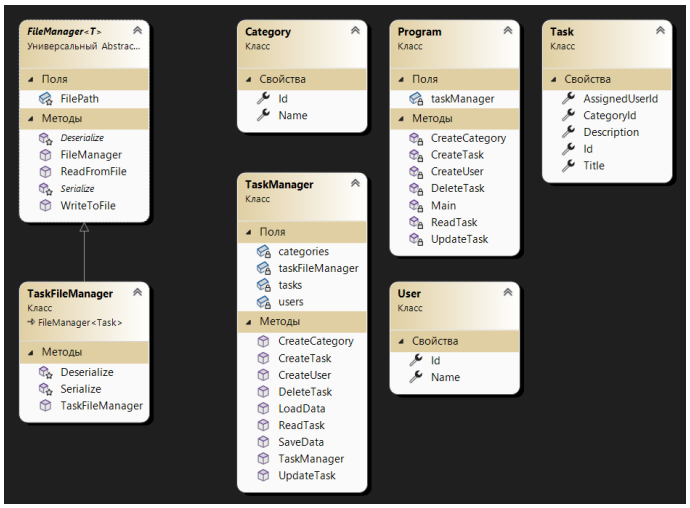
Program został zaprojektowany z myślą o prostocie i łatwości zarządzania zadaniami. Klasy **‘Task‘, ‘Category‘, i ‘User‘** reprezentują główne encje w systemie, a każda z nich przechowuje istotne informacje o zadaniach, kategoriach i użytkownikach. W celu zapewnienia trwałego przechowywania danych, klasa abstrakcyjna **‘FileManager<T>‘** została wprowadzona, aby obsługiwać operacje odczytu i zapisu do pliku dla różnych typów obiektów.

Klasa **‘TaskFileManager‘** specjalizuje operacje plikowe dla klas zadań, implementując metody serializacji i deserializacji. To podejście umożliwia łatwą rozbudowę systemu o kolejne klasy, gdyby były potrzebne.

Klasa **‘TaskManager‘** pełni kluczową rolę w zarządzaniu operacjami na zadaniach, kategoriach i użytkownikach. Wykorzystuje **‘TaskFileManager‘** do obsługi operacji plikowych, zapewniając jednocześnie funkcje takie jak tworzenie, odczytywanie, aktualizowanie, usuwanie, a także ładowanie i zapisywanie danych.

Główna logika interakcji z użytkownikiem została umieszczona w klasie **‘Program‘**. W nieskończonej pętli użytkownik ma dostęp do tekstowego menu, gdzie może wybierać opcje związane z tworzeniem, odczytywaniem, aktualizowaniem i usuwaniem zadań, kategorii i użytkowników. To interaktywne podejście sprawia, że program jest przyjazny dla użytkownika i prosty w obsłudze.

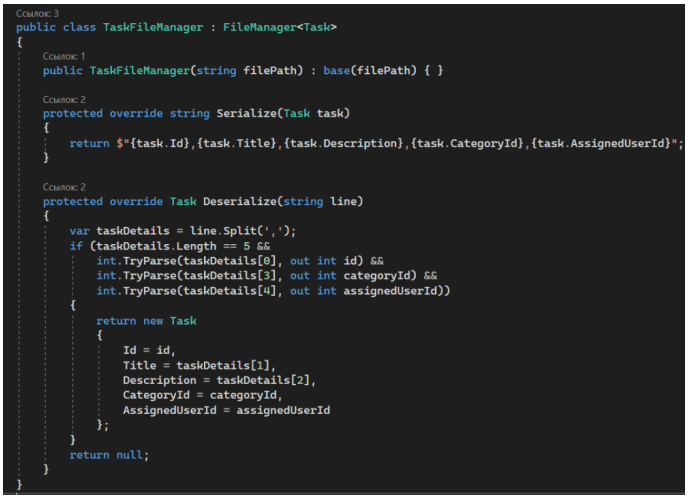
2.1 Diagram klas



Rysunek 2.1: Przedstawiono diagram klas

2.2 Zarządzanie danymi i baza danych

W tym projekcie, baza danych jest reprezentowana przez plik tekstowy, w którym przechowywane są dane o zadaniach. Zastosowano prosty mechanizm serializacji i deserializacji danych zadań w formacie tekstowym. Klasa TaskFileManager odpowiada za odczyt i zapis danych do pliku. Metody Serialize i Deserialize w klasie TaskFileManager są odpowiedzialne za konwersję obiektów Task na ciągi i odwrotnie.



```
Ссылок: 3
public class TaskFileManager : FileManager<Task>
{
    Ссылок: 1
    public TaskFileManager(string filePath) : base(filePath) { }

    Ссылок: 2
    protected override string Serialize(Task task)
    {
        return $"{task.Id},{task.Title},{task.Description},{task.CategoryId},{task.AssignedUserId}";
    }

    Ссылок: 2
    protected override Task Deserialize(string line)
    {
        var taskDetails = line.Split(',');
        if (taskDetails.Length == 5 &&
            int.TryParse(taskDetails[0], out int id) &&
            int.TryParse(taskDetails[3], out int categoryId) &&
            int.TryParse(taskDetails[4], out int assignedUserId))
        {
            return new Task
            {
                Id = id,
                Title = taskDetails[1],
                Description = taskDetails[2],
                CategoryId = categoryId,
                AssignedUserId = assignedUserId
            };
        }
        return null;
    }
}
```

Rysunek 2.2: Przedstawiono kod serializacji i deserializacji danych zadań w formacie tekstowym

2.3 Wymagania sprzętowe i programowe

Minimalne wymagania sprzętowe dla uruchomienia tego programu są dość niskie, ponieważ jest to prosty program konsolowy. Wymagania te obejmują:

- Procesor o częstotliwości taktowania 1Ghz lub więcej
- Kilku megabajtów na dysku twardym
- System operacyjny: Windows
- System 64-bit

Rozdział 3

Harmonogram realizacji projektu

Projekt rozpoczęłam od wymyślenia tematu, zamierzającego stworzyć prosty system zarządzania zadaniami oparty na programowaniu obiektowym. Następnie przygotowałam dokumentację projektową zawierającą opis wybranego tematu oraz wymagania funkcjonalne i niefunkcjonalne.

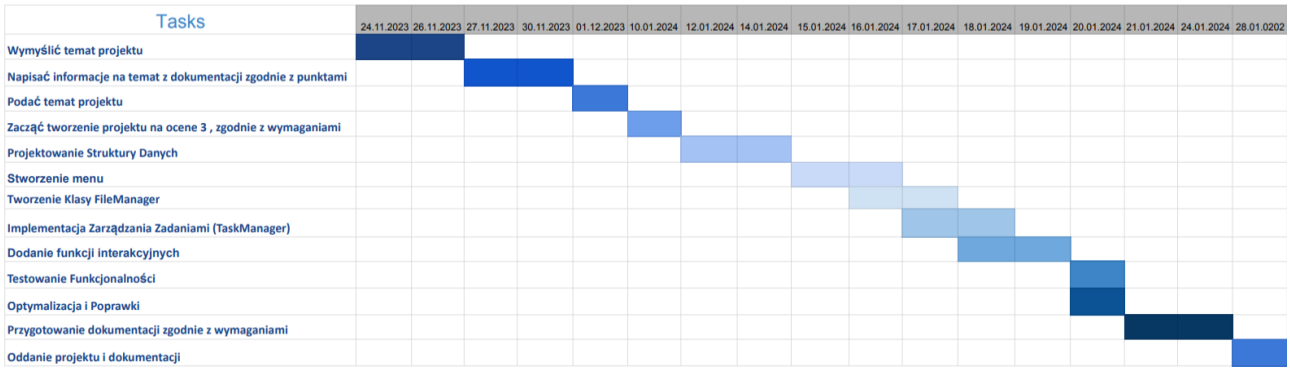
Zglosiłam temat projektu jako „System zarządzania zadaniami 'Planer'". Zaimplementowałam podstawową strukturę programu zgodnie z wymaganiami oceny 3, włączając utworzenie klas Task, Category, User i abstrakcyjnej klasy FileManager<T>.

Następnie zaprojektowałam strukturę danych i stworzyłam interaktywne menu w konsoli, które pozwala użytkownikowi łatwo korzystać z różnych funkcji programu. Zaimplementowałam klasy takie jak TaskFileManager do obsługi operacji na plikach oraz TaskManager, które zarządzały operacjami na zadaniach, kategoriach i użytkownikach.

Testowałam program, poprawiłam błędy i wprowadziłam poprawki.

Ostatecznie przygotowałam kompletną dokumentację projektową opisującą mój projekt.

3.1 Diagram Ganta:



Rysunek 3.1: Przedstawiono diagram Ganta

Rozdział 4

Repozytorium i system kontroli wersji.

4.1 Mój projekt znajduje się w repozytorium pod linkiem:

https://github.com/humster2612/C-programming/tree/main/projectOOP_3.0_Vika

Rozdział 5

Prezentacja warstwy użytkowej projektu

5.1 Menu główne:

Stworzyłam menu w programie, aby użytkownik mógł wejść w interakcję z programem. Głównym założeniem przy tworzeniu menu było uproszczenie nawigacji, aby użytkownik mógł w łatwy sposób wybrać interesującą go operację. Każda opcja ma przejrzysty opis. W efekcie tworzenie menu jest kluczowym elementem zwiększającym użyteczność aplikacji, pozwalającym użytkownikowi na wygodne korzystanie z funkcji programu bez konieczności znajomości szczegółów implementacji czy składni.

```
while (true)
{
    Console.WriteLine("Choose an option:");
    Console.WriteLine("1. Create Task");
    Console.WriteLine("2. Read Task");
    Console.WriteLine("3. Update Task");
    Console.WriteLine("4. Delete Task");
    Console.WriteLine("5. Create Category");
    Console.WriteLine("6. Create User");
    Console.WriteLine("0. Exit");

    var choice = Console.ReadLine();

    switch (choice)
    {
        case "1":
            CreateTask();
            break;

        case "2":
            ReadTask();
            break;

        case "3":
            UpdateTask();
            break;

        case "4":
```

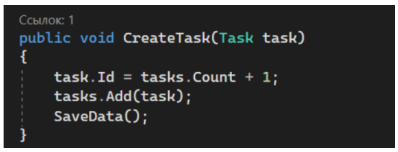
```
Choose an option:
1. Create Task
2. Read Task
3. Update Task
4. Delete Task
5. Create Category
6. Create User
0. Exit
```

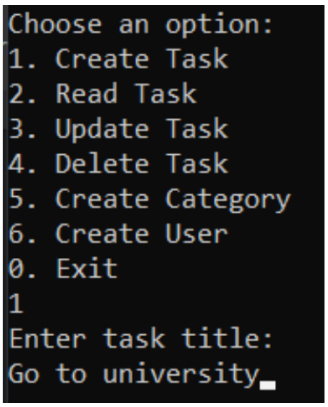
Rysunek 5.1: Przedstawiono menu główne

5.2 Operacje na Zadaniach:

5.2.1 Tworzenie Zadania (CreateTask):

Użytkownik może wprowadzić tytuł, opis, oraz przypisać kategorię i użytkownika do nowego zadania. Po utworzeniu, zadanie zostaje zapisane do pliku.

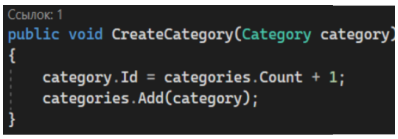


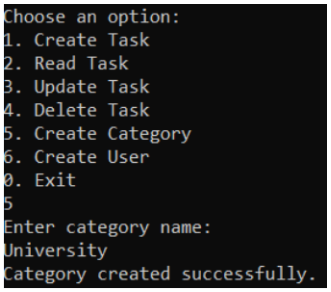


Rysunek 5.2: Metoda tworząca nowe zadanie i dodająca je do listy zadań

5.2.2 Tworzenie Kategorii (CreateCategory):

Kategorie są używane do grupowania zadań w logiczny sposób. Użytkownik, korzystając z tej funkcji, może dynamicznie rozszerzać system o nowe kategorie.

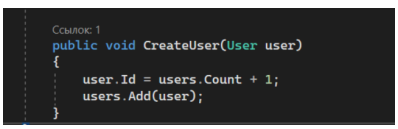


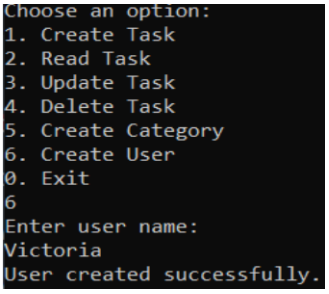


Rysunek 5.3: Metoda tworząca nową kategorię i dodająca ją do listy kategorii

5.2.3 Tworzenie Użytkownika (CreateUser):

Pozwala na dodanie nowego użytkownika do bazy. Użytkownicy są istotni, gdy chcemy przypisać zadania do konkretnych osób.





Rysunek 5.4: Metoda tworząca nowego użytkownika i dodająca go do listy użytkowników

5.2.4 Odczytywanie Zadania (ReadTask):

Umożliwia użytkownikowi podanie identyfikatora zadania, a następnie wyświetla szczegóły zadania, jeżeli istnieje.

```
Ссылок: 2
public Task ReadTask(int id)
{
    return tasks.Find(t => t.Id == id);
}
```

Rysunek 5.5: Metoda do odczytywania zadania na podstawie identyfikatora

5.2.5 Aktualizacja Zadania (UpdateTask):

Pozwala na modyfikację istniejącego zadania. Użytkownik może wprowadzić nowy tytuł, opis, kategorię i użytkownika przypisanego do zadania.

```
Ссылок: 1
public void UpdateTask(Task updatedTask)
{
    var existingTask = tasks.Find(t => t.Id == updatedTask.Id);
    if (existingTask != null)
    {
        existingTask.Title = updatedTask.Title;
        existingTask.Description = updatedTask.Description;
        existingTask.CategoryId = updatedTask.CategoryId;
        existingTask.AssignedUserId = updatedTask.AssignedUserId;
        SaveData();
    }
}
```

Rysunek 5.6: Metoda do aktualizacji istniejącego zadania

```
    var updatedTask = new Task
    {
        Id = taskId,
        Title = updatedTitle,
        Description = updatedDescription,
        CategoryId = updatedCategoryId,
        AssignedUserId = updatedUserId
    };
```

Rysunek 5.7: Przedstawiono napisany kod

5.2.6 Usuwanie Zadania (DeleteTask):

Usuwa wybrane zadanie na podstawie podanego identyfikatora.

```
Ссылок: 1
public void DeleteTask(int id)
{
    tasks.RemoveAll(t => t.Id == id);
    SaveData();
}
```

Rysunek 5.8: Metoda do usuwania zadania na podstawie identyfikatora

```
Ссылок: 1
static void DeleteTask()
{
    Console.WriteLine("Enter task ID to delete:");
    if (int.TryParse(Console.ReadLine(), out var taskId))
    {
        taskManager.DeleteTask(taskId);
        Console.WriteLine("Task deleted successfully.");
    }
    else
    {
        Console.WriteLine("Invalid task ID.");
    }
}
```

Rysunek 5.9: Przedstawiono napisany kod

```
Retrieved Task: Zadanie na dzisiaj, Assigned to: 1
Choose an option:
1. Create Task
2. Read Task
3. Update Task
4. Delete Task
5. Create Category
6. Create User
0. Exit
4
Enter task ID to delete:
1
Task deleted successfully.
```

Rysunek 5.10: Przedstawiono menu DeleteTask

Rozdział 6

Podsumowanie

6.1 Planowane dalsze prace rozwojowe:

Tworząc ten projekt zrozumiałam, że chciałabym go rozwinąć, wprowadzić pewne zmiany, ulepszyć i podnieść jakość.Wybrałam ten temat, ponieważ często spotykam się z poszukiwaniem normalnej aplikacji do nagrywania moich planów, myślę, że tak wszystkich ludzi, dlatego chciałam stworzyć własną aplikację konsolową na ten temat, która byłaby dla mnie odpowiednia.

Pierwszą rzeczą, którą chciałabym dodać, jest interfejs graficzny (GUI): chciałabym na przykład stworzyć elementy graficzne ekranu (ikony, stylizować menu, dodawać przyciski itp.)

Naprzykład interfejs konsoli CLI, jest używany w wierszu poleceń. Nie ma przycisków ani okien, a aby sterować programem, należy pisać polecenia do konsoli. Jest to wygodne tylko dla programistów. A programy i aplikacje do spraw osobistych są łatwiejsze i wygodniejsze w obsłudze za pośrednictwem GUI, ponieważ od razu patrząc na program, osoba powinna zrozumieć, co oznacza każdy element programu.

Również dla mnie główne 2 aspekty:

- Aby interfejs nie zwalniał ani nie zawieszał się;
- Powinien także umożliwiać szybkie dotarcie do żądanej funkcji.

Następnie dodałabym dodatkowe funkcje takie jak powiadomienia o zadaniach, generowanie raportów, aby zapewnić użytkownikowi jeszcze bardziej wszechstronne narzędzie.

Wprowadźić także zmiany przenosząc hurtownię danych na bardziej zaawansowany system zarządzania bazami danych, który umożliwi sprawne zarządzanie danymi na większą skalę.

Dodałabym funkcję dla współpracy użytkowników umożliwia współpracę poprzez wymianę zadań, co jest przydatne w pracy zespołowej.

Bibliografia

[1] https://www.w3schools.com/cs/cs-methods.php

[2] https://www.overleaf.com/learn/latex/Bibliography-management-with-bibtex

[3] https://app.ganttpro.com//project/1705772967266/gantt

[4] https://learn.microsoft.com/pl-pl/dotnet/csharp/

Spis rysunków

Spis tabel

Wyższa Szkoła Informatyki i Zarządzania z siedzibą w Rzeszowie
Kolegium Informatyki Stosowanej

Projekt na temat
Zarządzanie zadaniami "Planer"

Autor: Victoria Polovyy
Promotor: mgr inż. Ewa Żeszławska
Słowa kluczowe: tutaj umieść słowa kluczowe

Treść streszczenia, czyli kilka zdań dotyczących treści pracy dyplomowej w języku polskim.

The University of Information Technology and Management in Rzeszow
Faculty of Applied Information Technology

Thesis Summary
Task management

Author: Victoria Polovyy
Supervisor: mgr inż. Ewa Żeszławska
Key words: tutaj umieść słowa kluczowe

Treść streszczenia, czyli kilka zdań dotyczących treści pracy dyplomowej w języku angielskim - tłumaczenie tekstu z języka polskiego.