|  |
| --- |
| Uniwersytet WSB Merito |
| Programowanie Zaawansowane II |
| Ćwiczenia 3 - zadania |

|  |
| --- |
| Autor: Lesław Pawlaczyk  2023/05/18 |

Spis treści

[Rozdział 1 – Zadania do zrealizowania 2](#_Toc135299228)

[Zadanie Nr 1 2](#_Toc135299229)

[Zadanie nr 2 3](#_Toc135299230)

[Zadanie nr 3 4](#_Toc135299231)

[Zadanie nr 4 5](#_Toc135299232)

[Zadanie nr 5 6](#_Toc135299233)

[Zadanie nr 6 7](#_Toc135299234)

[Zadanie nr 7 8](#_Toc135299235)

[Rozdział 2 – Ocenianie 10](#_Toc135299236)

[Bibliografia 12](#_Toc135299237)

# Rozdział 1 – Zadania do zrealizowania

Napisać jedno z poniższych zadań w technologii WPF.

Założenia:

1. Zadanie powinno być bazowane na rozwiązaniu z ćwiczeń nr 2.
2. GUI stworzone w WPF.
3. Dane powinny być zapisywane i odczytywane z pliku JSON, lub bazy danych.
4. Należy używać wstrzykiwania zależności.
5. Należy używać architektury serwisowej.
6. Używać MVVM.
7. Używać walidacji danych.

## Zadanie Nr 1

Zadanie: Aplikacja do zarządzania rezerwacją biletów do kina

Wymagania:

1. Stwórz aplikację okienkową w środowisku Windows Presentation Foundation (WPF) przy użyciu języka C#.
2. Wykorzystaj bazę danych SQL Server lub SQLite, albo JSON do przechowywania danych.
3. Aplikacja powinna umożliwiać zarządzanie rezerwacją biletów do kina.
4. Interfejs użytkownika powinien zawierać: a. Listę dostępnych filmów wraz z informacjami takimi jak tytuł, opis, długość trwania itp. b. Możliwość wyboru filmu i określenia liczby biletów do zarezerwowania. c. Formularz rezerwacji, w którym użytkownik wprowadza dane osobowe (imię, nazwisko, adres e-mail itp.). d. Podgląd i potwierdzenie rezerwacji przed dokonaniem płatności.
5. Aplikacja powinna automatycznie kontrolować dostępność miejsc na dany seans i uniemożliwiać rezerwację, jeśli nie ma wystarczającej liczby wolnych miejsc.
6. Po dokonaniu rezerwacji, informacje o rezerwacji (film, liczba biletów, dane osobowe itp.) powinny być zapisane w bazie danych.
7. Dodatkowo, aplikacja powinna umożliwiać przeglądanie i zarządzanie istniejącymi rezerwacjami, takie jak anulowanie rezerwacji czy wyświetlanie historii rezerwacji dla danego użytkownika.
8. Zapewnij walidację danych, aby upewnić się, że wprowadzane informacje są poprawne (np. sprawdź, czy adres e-mail jest w prawidłowym formacie).
9. Skonfiguruj bazę danych z odpowiednimi tabelami i relacjami, takimi jak tabele "Filmy", "Rezerwacje" i "Użytkownicy".
10. Po zakończeniu pracy nad projektem umieść go w repozytorium Git.

Dodatkowe wskazówki:

* Upewnij się, że baza danych jest poprawnie skonfigurowana i dostępna z poziomu aplikacji.
* Wykorzystaj odpowiednie kontrolek w interfejsie użytkownika, takie jak ListBox, TextBox, DataGrid itp., aby prezentować dane i umożliwiać użytkownikowi interakcję.
* Zadbaj o obsługę błędów i wyjątków, oraz dostarczaj użytkownikowi odpowiednie komunikaty zwrotne.
* Podziel kod na odpowiednie klasy i metody, aby zapewnić czytelność i separację odpowiedzialności.

## Zadanie nr 2

Zadanie: Aplikacja do zarządzania listą zadań

Wymagania:

1. Stwórz aplikację okienkową w środowisku Windows Presentation Foundation (WPF) przy użyciu języka C#.
2. Wykorzystaj bazę danych SQL Server lub SQLite, albo JSON do przechowywania danych.
3. Aplikacja powinna umożliwiać zarządzanie listą zadań.
4. Interfejs użytkownika powinien zawierać: a. Listę zadań z możliwością ich dodawania, edycji i usuwania. b. Każde zadanie powinno mieć pola takie jak tytuł, opis, data rozpoczęcia, data zakończenia, priorytet itp. c. Przyciski do oznaczania zadań jako zakończone lub do zmiany ich priorytetu. d. Możliwość sortowania i filtrowania zadań według różnych kryteriów.
5. Aplikacja powinna umożliwiać przypisanie zadań do konkretnych użytkowników lub kategorii.
6. Zaimplementuj relacje między encjami, takie jak wiele-do-jednego (np. zadanie może być przypisane do jednego użytkownika, ale użytkownik może mieć wiele zadań).
7. Użyj odpowiednich kontrolek w interfejsie użytkownika, takich jak ListBox, DataGrid, TextBox itp., aby prezentować dane i umożliwiać użytkownikowi interakcję.
8. Zapewnij walidację danych, aby upewnić się, że wprowadzane informacje są poprawne (np. sprawdź, czy daty są w odpowiednim formacie).
9. Skonfiguruj bazę danych z odpowiednimi tabelami i relacjami, takimi jak tabele "Zadania", "Użytkownicy" i "Kategorie".
10. Po zakończeniu pracy nad projektem umieść go w repozytorium Git.

Dodatkowe wskazówki:

* Upewnij się, że baza danych jest poprawnie skonfigurowana i dostępna z poziomu aplikacji.
* Wykorzystaj mechanizmy filtrowania, sortowania i przeszukiwania danych w interfejsie użytkownika, aby ułatwić użytkownikowi zarządzanie zadaniami.
* Zadbaj o obsługę błędów i wyjątków, oraz dostarczaj użytkownikowi odpowiednie komunikaty zwrotne.
* Podziel kod na odpowiednie klasy i metody, aby zapewnić czytelność i separację odpowiedzialności.

## Zadanie nr 3

Zadanie: Aplikacja do zarządzania książkami w bibliotece

Wymagania:

1. Stwórz aplikację okienkową w środowisku Windows Presentation Foundation (WPF) przy użyciu języka C#.
2. Wykorzystaj bazę danych SQL Server lub SQLite, albo JSON do przechowywania danych.
3. Aplikacja powinna umożliwiać zarządzanie książkami w bibliotece.
4. Interfejs użytkownika powinien zawierać: a. Listę książek wraz z informacjami takimi jak tytuł, autor, gatunek, rok wydania itp. b. Możliwość dodawania nowych książek do biblioteki. c. Edycję istniejących danych książek. d. Usuwanie książek z biblioteki.
5. Aplikacja powinna umożliwiać wyszukiwanie książek na podstawie różnych kryteriów, takich jak tytuł, autor czy gatunek.
6. Dodatkowo, aplikacja powinna umożliwiać rezerwację i wypożyczenie książek przez użytkowników biblioteki.
7. Użyj odpowiednich kontrolek w interfejsie użytkownika, takich jak ListBox, DataGrid, TextBox itp., aby prezentować dane i umożliwiać użytkownikowi interakcję.
8. Skonfiguruj bazę danych z odpowiednimi tabelami i relacjami, takimi jak tabele "Książki", "Użytkownicy" i "Wypożyczenia".
9. Zapewnij walidację danych, aby upewnić się, że wprowadzane informacje są poprawne (np. sprawdź, czy rok wydania jest w prawidłowym formacie).
10. Po zakończeniu pracy nad projektem umieść go w repozytorium Git.

Dodatkowe wskazówki:

* Upewnij się, że baza danych jest poprawnie skonfigurowana i dostępna z poziomu aplikacji.
* Wykorzystaj mechanizmy wyszukiwania i filtrowania danych w interfejsie użytkownika, aby ułatwić użytkownikowi odnalezienie konkretnych książek.
* Zadbaj o obsługę błędów i wyjątków, oraz dostarczaj użytkownikowi odpowiednie komunikaty zwrotne.
* Podziel kod na odpowiednie klasy i metody, aby zapewnić czytelność i separację odpowiedzialności.

## Zadanie nr 4

Zadanie: Aplikacja do monitorowania stanu magazynu

Wymagania:

1. Stwórz aplikację okienkową w środowisku Windows Presentation Foundation (WPF) przy użyciu języka C#.
2. Wykorzystaj bazę danych SQL Server lub SQLite, albo JSON do przechowywania danych.
3. Aplikacja powinna umożliwiać monitorowanie stanu magazynu.
4. Interfejs użytkownika powinien zawierać: a. Listę produktów w magazynie wraz z informacjami takimi jak nazwa produktu, ilość dostępna, cena itp. b. Możliwość dodawania nowych produktów do magazynu. c. Edycję istniejących danych produktów. d. Usuwanie produktów z magazynu.
5. Aplikacja powinna umożliwiać sprawdzanie dostępności produktów na podstawie ich stanu w magazynie.
6. Dodatkowo, aplikacja powinna rejestrować historię ruchu towarów, taką jak przyjęcie dostawy, sprzedaż czy usunięcie z magazynu.
7. Użyj odpowiednich kontrolek w interfejsie użytkownika, takich jak ListBox, DataGrid, TextBox itp., aby prezentować dane i umożliwiać użytkownikowi interakcję.
8. Skonfiguruj bazę danych z odpowiednimi tabelami i relacjami, takimi jak tabele "Produkty", "RuchTowaru" itp.
9. Zapewnij walidację danych, aby upewnić się, że wprowadzane informacje są poprawne (np. sprawdź, czy cena jest w prawidłowym formacie).
10. Po zakończeniu pracy nad projektem umieść go w repozytorium Git.

Dodatkowe wskazówki:

* Upewnij się, że baza danych jest poprawnie skonfigurowana i dostępna z poziomu aplikacji.
* Wykorzystaj mechanizmy filtrowania i sortowania danych w interfejsie użytkownika, aby ułatwić użytkownikowi przeglądanie stanu magazynu.
* Zadbaj o obsługę błędów i wyjątków, oraz dostarczaj użytkownikowi odpowiednie komunikaty zwrotne.
* Podziel kod na odpowiednie klasy i metody, aby zapewnić czytelność i separację odpowiedzialności.

## Zadanie nr 5

Zadanie: Aplikacja do śledzenia wydatków domowych

Wymagania:

1. Stwórz aplikację okienkową w środowisku Windows Presentation Foundation (WPF) przy użyciu języka C#.
2. Wykorzystaj bazę danych SQL Server lub SQLite, albo JSON do przechowywania danych.
3. Aplikacja powinna umożliwiać śledzenie wydatków domowych.
4. Interfejs użytkownika powinien zawierać: a. Formularz do wprowadzania nowych transakcji, z polami takimi jak data, kategoria (np. jedzenie, mieszkanie, rozrywka), kwota itp. b. Możliwość wyświetlania listy transakcji wraz z ich szczegółami. c. Możliwość filtrowania transakcji według kategorii lub określonego okresu czasu. d. Wykresy lub statystyki, które prezentują podsumowanie wydatków w różnych kategoriach.
5. Aplikacja powinna automatycznie sumować i aktualizować całkowitą kwotę wydatków w określonym okresie czasu.
6. Dodatkowo, aplikacja powinna umożliwiać ustawienie celów oszczędnościowych i monitorowanie postępów w osiąganiu tych celów.
7. Użyj odpowiednich kontrolek w interfejsie użytkownika, takich jak TextBox, DataGrid, ComboBox, Chart itp., aby prezentować dane i umożliwiać użytkownikowi interakcję.
8. Skonfiguruj bazę danych z odpowiednimi tabelami i relacjami, takimi jak tabele "Transakcje", "Kategorie" i "CeleOszczędnościowe".
9. Zapewnij walidację danych, aby upewnić się, że wprowadzane informacje są poprawne (np. sprawdź, czy data jest w prawidłowym formacie).
10. Po zakończeniu pracy nad projektem umieść go w repozytorium Git.

Dodatkowe wskazówki:

* Upewnij się, że baza danych jest poprawnie skonfigurowana i dostępna z poziomu aplikacji.
* Wykorzystaj mechanizmy filtrowania i sortowania danych w interfejsie użytkownika, aby ułatwić użytkownikowi przeglądanie i analizowanie wydatków.
* Zadbaj o obsługę błędów i wyjątków, oraz dostarczaj użytkownikowi odpowiednie komunikaty zwrotne.
* Podziel kod na odpowiednie klasy i metody, aby zapewnić czytelność i separację odpowiedzialności.

## Zadanie nr 6

Zadanie: Aplikacja do planowania zadań i zarządzania czasem

Wymagania:

1. Stwórz aplikację okienkową w środowisku Windows Presentation Foundation (WPF) przy użyciu języka C#.
2. Wykorzystaj bazę danych SQL Server lub SQLite, albo JSON do przechowywania danych.
3. Aplikacja powinna umożliwiać planowanie zadań i zarządzanie czasem.
4. Interfejs użytkownika powinien zawierać: a. Kalendarz lub widok kalendarza, który pokazuje planowane zadania w różnych dniach. b. Formularz do dodawania nowych zadań, z polami takimi jak tytuł, opis, data rozpoczęcia, czas trwania itp. c. Możliwość oznaczania zadań jako ukończone. d. Wyświetlanie listy zadań wraz z ich szczegółami.
5. Aplikacja powinna umożliwiać ustalanie priorytetów zadań oraz przypisywanie ich do kategorii (np. praca, dom, nauka).
6. Dodatkowo, aplikacja powinna umożliwiać generowanie raportów dotyczących czasu poświęconego na poszczególne kategorie zadań.
7. Użyj odpowiednich kontrolek w interfejsie użytkownika, takich jak Calendar, TextBox, DataGrid, ComboBox itp., aby prezentować dane i umożliwiać użytkownikowi interakcję.
8. Skonfiguruj bazę danych z odpowiednimi tabelami i relacjami, takimi jak tabele "Zadania", "Kategorie" i "Raporty".
9. Zapewnij walidację danych, aby upewnić się, że wprowadzane informacje są poprawne (np. sprawdź, czy data jest w prawidłowym formacie).
10. Po zakończeniu pracy nad projektem umieść go w repozytorium Git.

Dodatkowe wskazówki:

* Upewnij się, że baza danych jest poprawnie skonfigurowana i dostępna z poziomu aplikacji.
* Wykorzystaj mechanizmy filtrowania i sortowania danych w interfejsie użytkownika, aby ułatwić użytkownikowi przeglądanie i zarządzanie zadaniami.
* Zadbaj o obsługę błędów i wyjątków, oraz dostarczaj użytkownikowi odpowiednie komunikaty zwrotne.
* Podziel kod na odpowiednie klasy i metody, aby zapewnić czytelność i separację odpowiedzialności.

To zadanie pozwoli studentom na praktyczne zastosowanie umiejętności programowania w środowisku WPF oraz pracy z bazą danych. Aplikacja do planowania zadań i zarządzania czasem umożliwi im praktykę w tworzeniu interfejsu użytkownika, implementacji funkcji dodawania i filtrowania danych, oraz generowania raportów czasowych.

## Zadanie nr 7

Zadanie: Aplikacja do zarządzania listą kontaktów

Wymagania:

1. Stwórz aplikację okienkową w środowisku Windows Presentation Foundation (WPF) przy użyciu języka C#.
2. Wykorzystaj bazę danych SQL Server lub SQLite, albo JSON do przechowywania danych.
3. Aplikacja powinna umożliwiać zarządzanie listą kontaktów.
4. Interfejs użytkownika powinien zawierać: a. Formularz do dodawania nowych kontaktów, z polami takimi jak imię, nazwisko, adres e-mail, numer telefonu itp. b. Możliwość edycji istniejących danych kontaktu. c. Wyświetlanie listy kontaktów wraz z ich podstawowymi informacjami. d. Możliwość usuwania kontaktów z listy.
5. Aplikacja powinna umożliwiać wyszukiwanie kontaktów na podstawie różnych kryteriów, takich jak imię, nazwisko czy adres e-mail.
6. Dodatkowo, aplikacja powinna umożliwiać grupowanie kontaktów w kategorie (np. Praca, Rodzina, Przyjaciele) i umożliwiać przypisanie kontaktu do jednej lub wielu kategorii.
7. Użyj odpowiednich kontrolek w interfejsie użytkownika, takich jak TextBox, DataGrid, ComboBox itp., aby prezentować dane i umożliwiać użytkownikowi interakcję.
8. Skonfiguruj bazę danych z odpowiednimi tabelami i relacjami, takimi jak tabele "Kontakty" i "Kategorie".
9. Zapewnij walidację danych, aby upewnić się, że wprowadzane informacje są poprawne (np. sprawdź, czy adres e-mail jest w prawidłowym formacie).
10. Po zakończeniu pracy nad projektem umieść go w repozytorium Git.

Dodatkowe wskazówki:

* Upewnij się, że baza danych jest poprawnie skonfigurowana i dostępna z poziomu aplikacji.
* Wykorzystaj mechanizmy filtrowania i sortowania danych w interfejsie użytkownika, aby ułatwić użytkownikowi przeglądanie i zarządzanie kontaktami.
* Zadbaj o obsługę błędów i wyjątków, oraz dostarczaj użytkownikowi odpowiednie komunikaty zwrotne.
* Podziel kod na odpowiednie klasy i metody, aby zapewnić czytelność i separację odpowiedzialności.

# Rozdział 2 – Ocenianie

**Rozwiązania powinny być indywidualne**.

1. Wykonać jedno z siedmiu zadań na ocenę co najmniej dst.
2. Wynik w postaci linku do repozytorium na GitHub oraz sprawozdanie w PDF. Sprawozdanie w PDF powinno się nazywać: ProgramowanieObiektowe\_Zadanie4\_Rozwiazanie\_ImieNazwisko\_XXXXX.pdf, gdzie ImieNazwisko zastąpić swoim imieniem i nazwiskiem a XXXXX numerem albumu.
3. Rozwiązanie powinno być linkiem do repo na GitHub, gdzie będzie pokazane PDFy albo BMP w wysokiej rozdzielczości z diagramami i wskazaniami w nazwie, którego zadania dotyczą.

**Strona tytułowa**:

1. Imię i nazwisko.
2. Adres email.
3. Numer albumu.
4. Data.
5. Link do repo na Github.

**Sposób oceniania**:

Ocena 3 (dostateczny)

* Aplikacja działa i spełnia podstawowe wymagania zadania, mogą się pojawić drobne błędy w funkcjonalności.

Ocena 3.5 (dostateczny plus)

* Spełniono kryteria na ocenę 4.5
* Aplikacja działa bezbłędnie i nie ma ostrzeżeń kompilacji.
* Aplikacja korzysta z serwisów oraz architektury MVVM/

Ocena 4 (dobry)

* Spełniono kryteria na ocenę 4.5
* Aplikacja ma ładne i zrozumiałe GUI.

Ocena 4.5 (dobry plus):

* Spełniono kryteria na ocenę 4.5
* Zastosowano poprawne komentarze, a kod jest czytelny i łatwy w zrozumieniu.

Ocena 5 (bardzo dobry):

* Spełniono kryteria na ocenę 4.5
* Napisane są unit testy dla wszystkich serwisów.

**Czas do wykonania**:

1. Do 2023/09/15.

**Uwaga:**

Przy zauważeniu, że kod powtarza się z kodem innego studenta rozwiązanie zostanie uznane za plagiat i ocenione na ocenę „ndst”.

# Bibliografia

1. <https://github.com/palles77/ProgrammingObjectOriented/raw/main/UmlTutorial.zip>
2. <https://github.com/palles77/ProgrammingObjectOriented/raw/main/UmlSpecification.pdf>
3. <https://uml.org>
4. <https://draw.io>
5. <https://www.microsoft.com/en-ww/microsoft-365/business-insights-ideas/resources/guide-to-uml-diagramming-and-database-modeling>
6. <https://www.modelio.org/index.htm>
7. <https://www.softwaretestinghelp.com/uml-diagram-tutorial/>