



**WYŻSZA SZKOŁA
INFORMATYKI i ZARZĄDZANIA**
z siedzibą w Rzeszowie

KOLEGIUM INFORMATYKI STOSOWANEJ

Kierunek: INFORMATYKA

Specjalność: Inżynieria Danych (ID)

Karol Grabowski
Nr albumu w69781

System biura podróży

Prowadzący: mgr inż. Ewa Żesławska

Praca projektowa programowanie obiektowe C#

Rzeszów 2025

Spis treści

Wstęp	4
1 Opis założeń projektu	5
1.1 Cele projektu	5
1.2 Wymagania funkcjonalne i нефункционалне	5
2 Opis struktury projektu	7
2.1 Modele Danych	8
2.2 Klasy Zarządzające (Service/Manager Classes)	9
2.3 Warstwa Interfejsu Użytkownika	10
2.4 Klasa Główna	11
3 Harmonogram realizacji projektu	12
3.1 Harmonogram realizacji projektu	12
3.1.1 Opis Diagramu Gantta	12
3.1.2 Napotkane problemy i trudności podczas realizacji projektu	12
3.2 Informacje o repozytorium	13
4 Prezentacja warstwy użytkowej projektu	14
4.0.1 Główne Menu Systemu	14
4.0.2 Podmenu Modułów	15
4.0.3 Moduł rezerwacji	15
4.0.4 Moduł rezerwacji - dodawanie rezerwacji	16
4.0.5 Moduł Płatności	16
4.0.6 Moduł Użytkowników	16
4.1 Interakcja i Walidacja	17
4.1.1 Zalety Interfejsu Konsolowego	17
4.2 Podsumowanie warstwy użytkowej	17
4.3 Poszczególne opcje programu	17
5 Podsumowanie	19
Bibliografia	20
Spis rysunków	21
Spis tablic	22

Wstęp

Współczesny rynek turystyczny charakteryzuje się nieustanną dynamiką, a klienci oczekują coraz bardziej zróżnicowanych i dostosowanych do ich potrzeb usług. W związku z tym biura podróży muszą wykorzystywać nowoczesne technologie, które pozwolą im utrzymać konkurencyjność i spełniać wymagania rynku. Wspomaganie codziennej działalności biura podróży za pomocą systemów informatycznych staje się kluczowe w procesach zarządzania ofertami, rezerwacjami, a także w obiegu informacji dotyczących płatności i rozliczeń. Tradycyjne, papierowe metody obsługi mogą prowadzić do błędów, opóźnień i utrudnionego dostępu do danych, co negatywnie wpływa na efektywność pracy oraz satysfakcję klientów. W odpowiedzi na te problemy, niniejszy projekt ma na celu stworzenie systemu informatycznego, który będzie kompleksowo wspierać zarządzanie biurem podróży.

System ten pozwoli na łatwe tworzenie, edytowanie oraz usuwanie ofert turystycznych, umożliwiając szybką reakcję na zmieniające się potrzeby rynku oraz na wprowadzanie nowych produktów do oferty. Dodatkowo, aplikacja będzie umożliwiała dokonywanie rezerwacji biletów na dostępne oferty, co zapewni wygodę klientom oraz pozwoli na pełną kontrolę nad rezerwacjami dla pracowników biura. Ważnym aspektem systemu będzie również moduł umożliwiający obsługę rozliczeń finansowych, który zautomatyzuje procesy związane z płatnościami za usługi oraz umożliwi szybkie i przejrzyste rozliczanie się z klientami.

Projekt zostanie opracowany w języku C#, co pozwoli na stworzenie wydajnej, bezpiecznej i skalowalnej aplikacji. Wykorzystanie tego języka umożliwi budowę systemu, który będzie łatwy do rozbudowy, dostosowania oraz utrzymania w przyszłości, co stanowi istotny element w kontekście rozwijających się potrzeb biura podróży. Projektowana aplikacja będzie zawierała intuicyjny interfejs użytkownika, który ułatwi pracownikom biura szybkie wykonywanie niezbędnych operacji oraz pozwoli klientom na komfortowe korzystanie z dostępnych usług.

Rozdział 1

Opis założeń projektu

1.1 Cele projektu

Celem niniejszego projektu jest opracowanie zaawansowanego systemu informatycznego, którego głównym zadaniem będzie wsparcie procesów związanych z zarządzaniem ofertami turystycznymi, rezerwacjami oraz rozliczeniami w biurze podróży. Projekt zakłada stworzenie elastycznego narzędzia umożliwiającego bieżącą modyfikację i aktualizację oferty turystycznej poprzez funkcjonalności pozwalające na tworzenie, edytowanie oraz usuwanie poszczególnych pozycji oferty, co umożliwi dynamiczne dostosowywanie się do zmieniających się warunków rynkowych. System zostanie wyposażony w moduł rezerwacyjny, który umożliwi klientom samodzielne dokonywanie rezerwacji biletów, a jednocześnie pozwoli pracownikom biura na efektywne zarządzanie procesem rezerwacyjnym. Wdrożenie funkcjonalności rozliczeniowych stanowi kolejny kluczowy element projektu, który ma na celu usprawnienie kontroli płatności oraz rozliczeń finansowych, co przyczyni się do zwiększenia przejrzystości oraz efektywności zarządzania finansami w biurze podróży.

1.2 Wymagania funkcjonalne i нефункционалне

Wymagania funkcjonalne

- Zarządzanie ofertami: Użytkownicy systemu (pracownicy biura) mogą dodawać nowe oferty, edytować ich parametry oraz usuwać je z bazy.
- Prezentacja oferty: Klienci mają możliwość przeglądania pełnych informacji o dostępnych ofertach, takich jak nazwa, destynacja, cena i liczba dostępnych miejsc.
- Rezerwacje: System umożliwia klientom dokonywanie rezerwacji wybranych ofert, zapisując te dane w wewnętrznej bazie.
- Modyfikacja rezerwacji: Pracownicy biura mogą wprowadzać zmiany w rezerwacjach lub je anulować.
- Obsługa płatności: System rejestruje płatności, generuje proste faktury oraz umożliwia przegląd historii transakcji.
- Zarządzanie użytkownikami: Możliwość tworzenia kont, przypisywania im określonych ról oraz ograniczania dostępu do niektórych funkcji systemu.

Wymagania нефункционалне

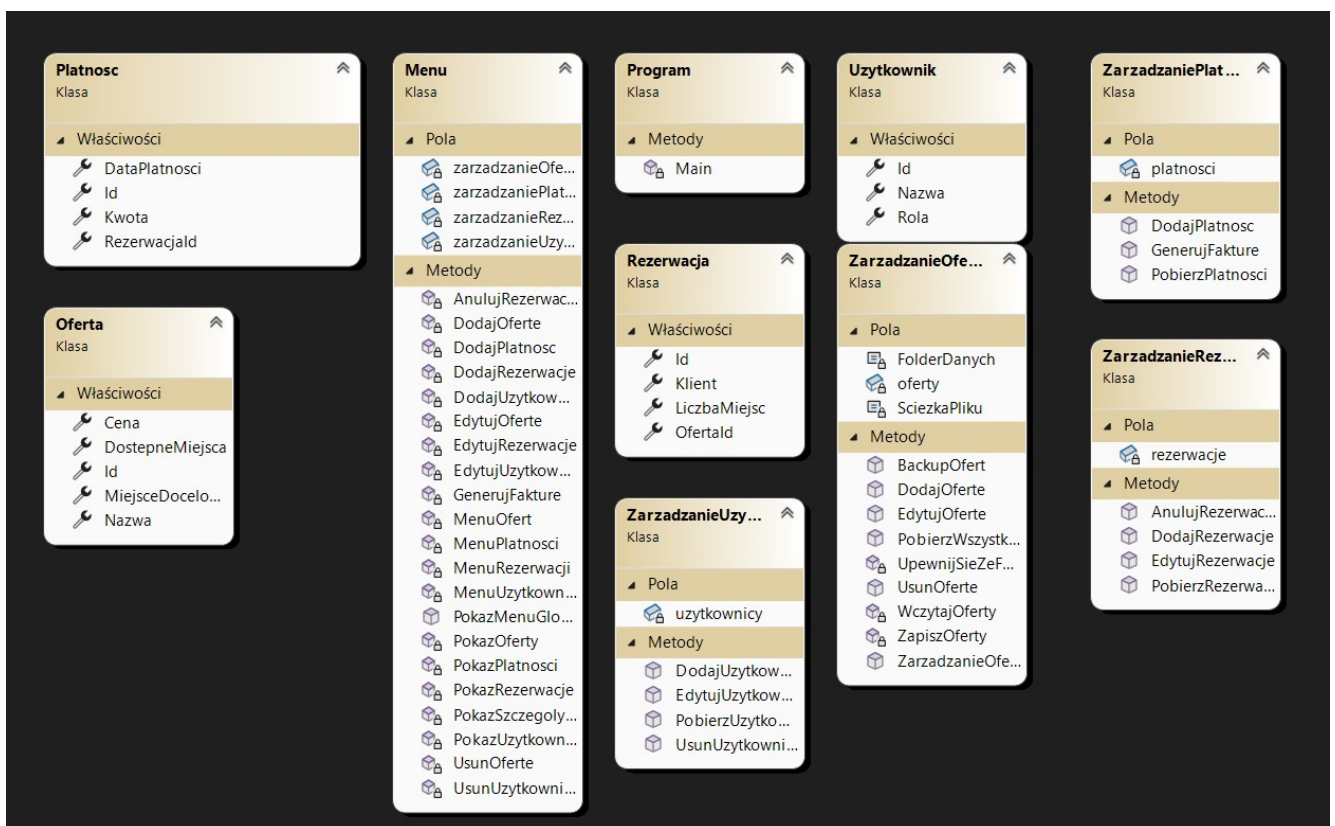
- Wydajność: System powinien działać sprawnie nawet przy dużej liczbie ofert i rezerwacji.
- Skalowalność: Aplikacja musi umożliwiać łatwe dodawanie nowych modułów i funkcjonalności.

- Bezpieczeństwo: Dane osobowe klientów i informacje finansowe muszą być odpowiednio chronione, co w przyszłości można rozszerzyć o mechanizmy szyfrowania i autoryzacji.
- Przyjazny interfejs: Nawet jeśli obecnie jest to interfejs konsolowy, system powinien być intuicyjny i łatwy w obsłudze.
- Kompatybilność: System oparty na technologii .NET musi działać na różnych systemach operacyjnych oraz być zgodny z aktualnymi standardami.
- Niezawodność: System powinien minimalizować ryzyko awarii oraz umożliwiać tworzenie kopii zapasowych danych.

Rozdział 2

Opis struktury projektu

Projekt został zbudowany w oparciu o modułową architekturę, dzięki której łatwo można zarządzać poszczególnymi częściami systemu. Główne komponenty to:



Rysunek 2.1: Diagram klas użytych w programie.

Modele Danych:

- **Oferta** – reprezentuje ofertę turystyczną, przechowując dane takie jak nazwa, destynacja, cena i dostępność miejsc.
- **Rezerwacja** – zawiera informacje o rezerwacjach, w tym identyfikator oferty, dane klienta oraz liczbę zarezerwowanych miejsc.
- **Płatność** – przechowuje dane dotyczące płatności, w tym kwotę, datę oraz powiązanie z konkretną rezerwacją.
- **Użytkownik** – model użytkownika systemu, który pozwala na zarządzanie kontami i uprawnieniami.

Logika Biznesowa: Każdy z modułów zarządzających (oferty, rezerwacje, płatności, użytkownicy) zawiera operacje CRUD, czyli funkcje umożliwiające tworzenie, odczyt, modyfikację i usuwanie danych. Moduły te korzystają z wbudowanych bibliotek .NET, takich jak `System.IO` i `System.Text.Json`, do operacji na plikach oraz serializacji danych.

Warstwa Prezentacji: Interfejs użytkownika opiera się na prostym menu konsolowym, które umożliwia nawigację między poszczególnymi modułami systemu. Użytkownik może wybrać odpowiednią opcję, aby wykonać operacje związane z zarządzaniem ofertami, rezerwacjami, płatnościami i kontami.

Punkt Wejścia: Program rozpoczyna działanie od metody `Main()`, która inicjuje obiekt odpowiedzialny za wyświetlenie menu głównego.

2.1 Modele Danych

Klasa Oferta

- **Atrybuty:**
 - `int Id` – unikalny identyfikator oferty.
 - `string Nazwa` – nazwa oferty (np. "Wakacje w Grecji").
 - `string MiejsceDocelowe` – miejsce docelowe oferty.
 - `decimal Cena` – cena oferty; typ `decimal` zapewnia dokładność operacji finansowych.
 - `int DostepneMiejsca` – liczba dostępnych miejsc w ofercie.
- **Metody:** Klasa wykorzystuje autogenerowane właściwości (`get`; `set`), pełniąc funkcję przechowywania danych bez dodatkowej logiki.

Klasa Rezerwacja

- **Atrybuty:**
 - `int Id` – unikalny identyfikator rezerwacji.
 - `int OfertaId` – identyfikator powiązanej oferty.
 - `string Klient` – dane klienta (np. imię i nazwisko) dokonującego rezerwacji.
 - `int LiczbaMiejsc` – liczba miejsc zarezerwowanych przez klienta.
- **Metody:** Podobnie jak w klasie **Oferta**, właściwości przechowują dane bez dodatkowej logiki.

Klasa Platnosc

- **Atrybuty:**
 - `int Id` – unikalny identyfikator płatności.
 - `int RezerwacjaId` – identyfikator rezerwacji, do której odnosi się płatność.
 - `decimal Kwota` – kwota płatności.
 - `string DataPlatnosci` – data dokonania płatności (jako ciąg znaków).
- **Metody:** Klasa służy do przechowywania danych dotyczących płatności przy użyciu autogenerowanych właściwości.

Klasa Uzytkownik

- **Atrybuty:**
 - `int Id` – unikalny identyfikator użytkownika.
 - `string Nazwa` – nazwa lub login użytkownika.
 - `string Rola` – rola przypisana użytkownikowi (np. administrator, pracownik, klient).
- **Metody:** Podstawową funkcją klasy jest przechowywanie informacji o użytkowniku.

2.2 Klasy Zarządzające (Service/Manager Classes)

Klasa ZarządzanieOfertami

- **Atrybuty:**
 - `const string FolderDanych` – nazwa folderu (np. "Dane") do przechowywania pliku.
 - `const string SciezkaPliku` – pełna ścieżka do pliku JSON (np. "Dane/oferty.json").
 - `List<Oferta> oferty` – lista ofert wczytanych z pliku.
- **Kluczowe Metody:**
 - `ZarzadzanieOfertami()` – Konstruktor; sprawdza istnienie folderu (metoda `UpewnijSieZeFolderIstnieje()` tworzy plik JSON, jeżeli nie istnieje, oraz wczytuje oferty (metoda `WczytajOferty()`).
 - `UpewnijSieZeFolderIstnieje()` – Sprawdza, czy folder `FolderDanych` istnieje; w razie potrzeby tworzy go.
 - `WczytajOferty()` – Odczytuje zawartość pliku JSON, deserializuje dane do listy obiektów `Oferta` i zwraca tę listę.
 - `ZapiszOferty()` – Serializuje listę ofert do formatu JSON i zapisuje ją w pliku.
 - `PobierzWszystkieOferty()` – Zwraca listę wszystkich ofert.
 - `DodajOferte(Oferta oferta)` – Przed dodaniem nowej oferty przypisuje jej unikalne `Id` i zapisuje ofertę.
 - `EdytujOferte(int id, Oferta nowaOferta)` – Wyszukuje ofertę po `Id` i aktualizuje jej dane (`Nazwa`, `MiejsceDocelowe`, `Cena`, `DostepneMiejsca`), a następnie zapisuje zmiany.
 - `UsunOferte(int id)` – Usuwa ofertę o podanym `Id` z listy i zapisuje zmiany.
 - `BackupOfert()` – Tworzy kopię zapasową pliku JSON z ofertami.

Klasa ZarządzanieRezerwacjami

- **Atrybuty:**
 - `List<Rezerwacja> rezerwacje` – lista rezerwacji przechowywana w pamięci.
- **Kluczowe Metody:**
 - `DodajRezerwacje(Rezerwacja rezerwacja)` – Dodaje nową rezerwację, przypisując jej unikalne `Id`.
 - `EdytujRezerwacje(int id, Rezerwacja nowaRezerwacja)` – Wyszukuje rezerwację po `Id` i aktualizuje jej dane (`OfertaId`, `Klient`, `LiczbaMiejsc`).
 - `AnulujRezerwacje(int id)` – Usuwa rezerwację o określonym `Id` z listy.
 - `PobierzRezerwacje()` – Zwraca aktualną listę rezerwacji.

Klasa ZarządzaniePłatnościami

- **Atrybuty:**

- `List<Platnosc> platnosci` – lista płatności.

- **Kluczowe Metody:**

- `DodajPlatnosc(Platnosc platnosc)` – Dodaje nową płatność, przypisując jej unikalne Id.
- `PobierzPlatnosci()` – Zwraca listę wszystkich płatności.
- `GenerujFakture(int platnoscId)` – Wyszukuje płatność po Id i zwraca sformatowaną fakturę zawierającą dane płatności; w przypadku braku płatności, zwraca komunikat o błędzie.

Klasa ZarządzanieUzytkownikami

- **Atrybuty:**

- `List<Uzytkownik> uzytkownicy` – lista użytkowników.

- **Kluczowe Metody:**

- `DodajUzytkownika(Uzytkownik uzytkownik)` – Dodaje nowego użytkownika, nadając mu unikalne Id.
- `EdytujUzytkownika(int id, Uzytkownik nowyUzytkownik)` – Znajduje użytkownika o danym Id i aktualizuje jego dane (Nazwa, Rola).
- `UsunUzytkownika(int id)` – Usuwa użytkownika o podanym Id z listy.
- `PobierzUzytkownikow()` – Zwraca listę wszystkich użytkowników.

2.3 Warstwa Interfejsu Użytkownika

Klasa Menu

- **Atrybuty:**

- `ZarządzanieOfertami zarzadzanieOfertami` – obiekt do zarządzania ofertami.
- `ZarządzanieRezerwacjami zarzadzanieRezerwacjami` – obiekt do zarządzania rezerwacjami.
- `ZarządzaniePłatnościami zarzadzaniePlatnosciami` – obiekt do obsługi płatności.
- `ZarządzanieUżytkownikami zarzadzanieUzytkownikami` – obiekt do zarządzania użytkownikami.

- **Kluczowe Metody:**

- `PokazMenuGlowne()` – Uruchamia główną pętlę interakcji, wyświetlając menu główne i kierując użytkownika do odpowiednich podmenu.
- **Podmenu ofert:**
 - * `MenuOfert()` – Wyświetla opcje zarządzania ofertami.
 - * `PokazOferty()` – Prezentuje listę wszystkich ofert.
 - * `PokazSzczegolyOferty()` – Po podaniu ID, wyświetla szczegóły wybranej oferty.

- * `DodajOferte()` – Pobiera dane od użytkownika, tworzy nową ofertę i wywołuje metodę dodawania w module ofert.
 - * `EdytujOferte()` – Umożliwia edycję istniejącej oferty.
 - * `UsunOferte()` – Umożliwia usunięcie oferty.
- **Podmenu rezerwacji:**
- * `MenuRezerwacji()` – Wyświetla opcje związane z rezerwacjami.
 - * `DodajRezerwacje()` – Pobiera dane wejściowe i dodaje nową rezerwację.
 - * `EdytujRezerwacje()` – Umożliwia edycję wybranej rezerwacji.
 - * `AnulujRezerwacje()` – Pozwala na anulowanie rezerwacji.
 - * `PokazRezerwacje()` – Wyświetla listę rezerwacji.
- **Podmenu płatności:**
- * `MenuPlatnosci()` – Wyświetla opcje związane z płatnościami.
 - * `DodajPlatnosc()` – Dodaje nową płatność na podstawie danych wejściowych.
 - * `PokazPlatnosci()` – Prezentuje listę płatności.
 - * `GenerujFakture()` – Generuje i wyświetla fakturę dla wybranej płatności.
- **Podmenu użytkowników:**
- * `MenuUzytkownikow()` – Wyświetla opcje zarządzania użytkownikami.
 - * `DodajUzytkownika()` – Dodaje nowego użytkownika.
 - * `EdytujUzytkownika()` – Umożliwia edycję danych użytkownika.
 - * `UsunUzytkownika()` – Pozwala na usunięcie użytkownika.
 - * `PokazUzytkownikow()` – Wyświetla listę użytkowników.

2.4 Klasa Główna

Klasa Program

- **Metody:**

- `Main()` – Główna metoda wejścia do aplikacji. Tworzy instancję klasy **Menu** i wywołuje metodę `PokazMenuGlowne()`, inicjując interakcję z użytkownikiem.

Hierarchia klas w systemie zarządzania biurem podróży została zaprojektowana w sposób modularny, co umożliwia oddzielenie modelu danych (Oferta, Rezerwacja, Platnosc, Uzytkownik) od logiki biznesowej (moduły ZarzadzanieOfertami, ZarzadzanieRezerwacjami, ZarzadzaniePlatnosciami, ZarzadzanieUzytkownikami) oraz interfejsu użytkownika (klasa Menu). Kluczowe metody umożliwiają wykonywanie operacji CRUD na danych oraz zapewniają prostą obsługę aplikacji przez użytkownika. Główny punkt wejścia, metoda `Main()` w klasie Program, inicjuje cały proces, uruchamiając główne menu i umożliwiając interakcję z systemem.

Rozdział 3

Harmonogram realizacji projektu

3.1 Harmonogram realizacji projektu

W tym rozdziale przedstawiono harmonogram projektu w formie diagramu Gantta. Jego celem jest graficzne zobrazowanie planu, etapów projektu oraz wzajemnych zależności czasowych między poszczególnymi zadaniami.

3.1.1 Opis Diagramu Gantta

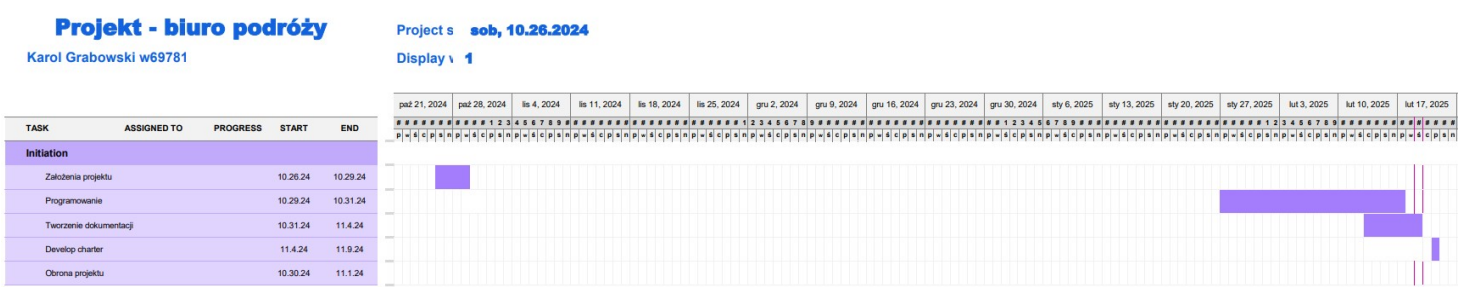
Diagram Gantta prezentuje szczegółowy harmonogram realizacji projektu, w ramach którego wyróżniono następujące etapy:

Tworzenie założeń projektu: Przeprowadzenie analizy wymagań, sporządzenie dokumentacji oraz opracowanie planu realizacji.

Kodowanie: Implementacja i integracja poszczególnych modułów aplikacji.

Tworzenie dokumentacji: Opracowanie kompletnej dokumentacji technicznej oraz użytkowej.

Obrona projektu: Przygotowanie prezentacji oraz obrona projektu przed komisją.



Rysunek 3.1: Diagram Gantta przedstawiający harmonogram realizacji projektu

3.1.2 Napotkane problemy i trudności podczas realizacji projektu

Podczas tworzenia systemu zarządzania biurem podróży napotkałem na kilka kluczowych wyzwań. Pierwszym z nich było ustalenie odpowiedniej architektury systemu. Zależało mi na tym, aby był on modułowy i skalowalny, co oznaczało łatwość w dodawaniu nowych funkcjonalności. W praktyce wymagało to oddzielenia modelu danych od logiki biznesowej oraz interfejsu użytkownika. Trudnością okazało się zaprojektowanie spójnego interfejsu komunikacyjnego między modułami, aby operacje związane z ofertami, rezerwacjami i płatnościami były zintegrowane i działały harmonijnie.

Kolejnym wyzwaniem było zarządzanie danymi. Przypisanie unikalnych identyfikatorów do ofert, rezerwacji, płatności i użytkowników stanowiło problem, zwłaszcza na początku projektu. Zastosowanie funkcji takich jak „Any()” czy „Max()” w LINQ ułatwiło generowanie tych identyfikatorów, ale

wymagało to starannego dopracowania logiki, aby uniknąć konfliktów i duplikacji. Dodatkowo, wdrożenie zapisu i odczytu danych w formacie JSON wymagało precyzyjnego typowania, szczególnie w przypadku wartości pieniężnych, gdzie użycie typu „decimal” było kluczowe dla zachowania dokładności. Na początku napotymano trudności związane z doбором odpowiednich opcji serializacji, co mogło prowadzić do utraty precyzji danych.

Obsługa plików oraz mechanizmy backupu stanowiły kolejny obszar wyzwań. System musiał regularnie zapisywać dane do pliku JSON oraz tworzyć kopie zapasowe, aby chronić informacje przed utratą. Problemy pojawiały się przy implementacji mechanizmu backupu – kluczowe było uwzględnienie sytuacji, w których plik docelowy już istniał, lub gdy występowały trudności z dostępem do dysku. W miarę postępu projektu optymalizacja operacji na plikach może stać się istotnym wyzwaniem, zwłaszcza przy większej liczbie rekordów, co może prowadzić do wąskich gardeł podczas zapisu i odczytu. Potencjalne rozwiązania obejmują wdrożenie mechanizmów buforowania lub migrację do bardziej zaawansowanej bazy danych.

Nie mniej ważnym aspektem była walidacja danych oraz zapewnienie intuicyjnego interfejsu użytkownika. Konieczne stało się wdrożenie mechanizmów sprawdzania poprawności danych wprowadzanych przez użytkowników, na przykład weryfikacja formatu cen czy dostępności miejsc. Początkowo pojawiały się błędy związane z nieprawidłowymi danymi, co wymagało dodatkowych korekt. Mimo że interfejs został zaprojektowany jako konsolowy, musiał być przejrzysty i czytelny, co nie zawsze było łatwe do osiągnięcia na etapie projektowania.

3.2 Informacje o repozytorium

Zarządzanie projektem odbywało się za pomocą systemu kontroli wersji Git, co pozwoliło na precyzyjne monitorowanie postępów. Ponadto, projekt był hostowany na platformie GitHub, zapewniającej łatwy dostęp do repozytorium oraz narzędzi wspierających współpracę i integrację z innymi systemami. Repozytorium można znaleźć pod adresem:

```
https://github.com/grabowskikarol/w69781\_Programowanie\_Objektowe/tree/main/Projekt\_w69781/Biuro\_podrozy
```

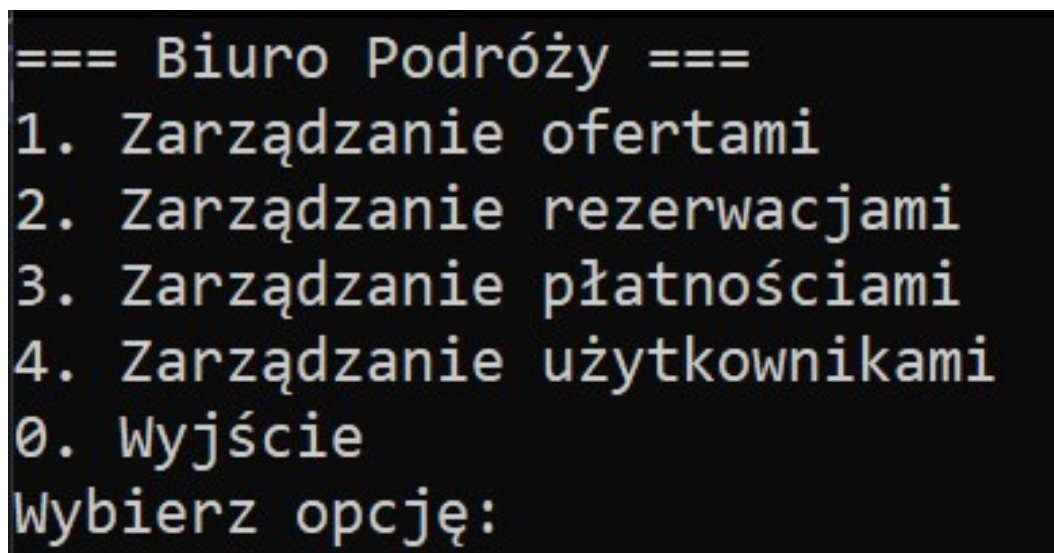
Rozdział 4

Prezentacja warstwy użytkowej projektu

4.0.1 Główne Menu Systemu

Po uruchomieniu aplikacji, użytkownik natychmiast widzi główne menu, które jest punktem wyjścia do wszystkich funkcji systemu. Menu to prezentuje kilka opcji, takich jak:

- Zarządzanie ofertami,
- Zarządzanie rezerwacjami,
- Zarządzanie płatnościami,
- Zarządzanie użytkownikami,
- Wyjście z aplikacji.



```
=== Biuro Podróży ===  
1. Zarządzanie ofertami  
2. Zarządzanie rezerwacjami  
3. Zarządzanie płatnościami  
4. Zarządzanie użytkownikami  
0. Wyjście  
Wybierz opcję:
```

Rysunek 4.1: Menu startowe aplikacji

Dzięki tej strukturze, użytkownik może szybko zorientować się w dostępnych funkcjonalnościach oraz łatwo przechodzić do interesującego go modułu.

4.0.2 Podmenu Modułów

Każdy z głównych modułów systemu posiada własne podmenu, co pozwala na szczegółową obsługę konkretnej dziedziny:

- **Wyświetlenie wszystkich ofert** – system prezentuje listę ofert, pokazując podstawowe dane, takie jak identyfikator, nazwa, destynacja, cena i liczba dostępnych miejsc.
- **Wyświetlenie szczegółów oferty** – po wprowadzeniu ID konkretnej oferty, system pokazuje wszystkie informacje dotyczące tej oferty.
- **Dodanie nowej oferty** – użytkownik wprowadza dane dotyczące nowej oferty (nazwa, miejsce docelowe, cena, dostępne miejsca), które następnie są zapisywane w systemie.
- **Edycja istniejącej oferty** – umożliwia modyfikację parametrów oferty, na podstawie której system aktualizuje dane w bazie.
- **Usuwanie oferty** – pozwala na usunięcie oferty o podanym ID.
- **Backup ofert** – opcja ta tworzy kopię zapasową pliku zawierającego dane ofert, co zwiększa bezpieczeństwo przechowywanych informacji.

```
=== Zarządzanie ofertami ===
1. Pokaż wszystkie oferty
2. Pokaż szczegóły oferty
3. Dodaj ofertę
4. Edytuj ofertę
5. Usuń ofertę
6. Backup ofert
0. Powrót do głównego menu
Wybierz opcję: 1

=== Lista Ofert ===
ID: 1, Nazwa: teneryfa, Miejsce: cos, Cena: 2, Miejsca: 3
ID: 2, Nazwa: Wakajce w Grecji, Miejsce: Vonitsa, Cena: 2400, Miejsca: 2
ID: 3, Nazwa: Litwa last minute, Miejsce: Wilno, Cena: 1200, Miejsca: 5
ID: 4, Nazwa: Wakacje w Grecji LAST MINUTE, Miejsce: Ateny, Cena: 2500, Miejsca: 10
```

Rysunek 4.2: Zarządzanie ofertami

4.0.3 Moduł rezerwacji

W podmenu rezerwacji użytkownik może:

- **Dodać nową rezerwację** – wprowadzając identyfikator oferty, dane klienta i liczbę miejsc, co pozwala na rejestrację rezerwacji w systemie.
- **Edytować istniejącą rezerwację** – umożliwiając aktualizację danych rezerwacji (np. zmiana liczby miejsc lub korekta danych klienta).
- **Anulować rezerwację** – usuwając rezerwację, gdy zajdzie taka potrzeba.
- **Wyświetlić listę wszystkich rezerwacji** – co pozwala na bieżący przegląd dokonanych rezerwacji.

4.0.4 Moduł rezerwacji - dodawanie rezerwacji

Poniżej przedstawiony sposób możliwości dodawania rezerwacji:

```
=== Zarządzanie rezerwacjami ===
1. Dodaj rezerwację
2. Edytuj rezerwację
3. Anuluj rezerwację
4. Pokaż wszystkie rezerwacje
0. Powrót do głównego menu
Wybierz opcję: 1

Podaj ID oferty: 3
Podaj imię i nazwisko klienta: Karol Grabowski
Podaj liczbę miejsc: 1
Rezerwacja została dodana!

=== Zarządzanie rezerwacjami ===
1. Dodaj rezerwację
2. Edytuj rezerwację
3. Anuluj rezerwację
4. Pokaż wszystkie rezerwacje
0. Powrót do głównego menu
Wybierz opcję: _
```

Rysunek 4.3: Zarządzanie rezerwacjami

4.0.5 Moduł Płatności

Podmenu płatności umożliwia:

- **Dodanie nowej płatności** – na podstawie identyfikatora rezerwacji, kwoty i daty płatności.
- **Wyświetlenie listy płatności** – system prezentuje wszystkie dokonane transakcje.
- **Generowanie faktur** – po podaniu ID płatności, system generuje sformatowaną fakturę, zawierającą kluczowe informacje o transakcji.

4.0.6 Moduł Użytkowników

- **Dodawanie nowych użytkowników** – umożliwiając rejestrację nowych kont wraz z przypisaniem roli (administrator, pracownik, klient).
- **Edycja danych użytkowników** – co pozwala na aktualizację informacji takich jak nazwa czy przypisana rola.

- **Usuwanie użytkowników** – umożliwiając kontrolę nad dostępem do systemu.
- **Wyświetlanie listy użytkowników** – co pozwala na szybki przegląd dostępnych kont.

4.1 Interakcja i Walidacja

Interfejs konsolowy został zaprojektowany tak, aby zapewnić czytelne komunikaty oraz intuicyjne wprowadzanie danych. Każda operacja jest poprzedzona instrukcjami wyświetlanymi na ekranie, a system sprawdza poprawność wprowadzonych danych (np. weryfikacja, czy podana cena ma właściwy format lub czy liczba miejsc jest liczbą całkowitą). Dzięki temu użytkownik otrzymuje natychmiastową informację zwrotną o poprawności swojej operacji, co zmniejsza ryzyko błędów.

4.1.1 Zalety Interfejsu Konsolowego

Mimo że interfejs jest oparty na konsoli, ma on kilka istotnych zalet:

Prostota: System opiera się na przejrzystym menu, co pozwala na szybkie opanowanie obsługi aplikacji nawet przez osoby niezaznajomione z technologiami informatycznymi.

Łatwość Utrzymania: Dzięki modularnej budowie menu oraz oddzieleniu logiki biznesowej, przyszłe modyfikacje lub rozbudowa interfejsu są łatwe do wykonania.

Szybkość Działania: Aplikacja reaguje natychmiastowo na wprowadzane komendy, co zwiększa efektywność pracy użytkownika.

4.2 Podsumowanie warstwy użytkowej

Warstwa użytkowa systemu "Biuro Podróży" jest kluczowym elementem, który umożliwia użytkownikom efektywną interakcję z systemem. Dzięki dobrze zaprojektowanemu interfejsowi konsolowemu, użytkownicy mogą łatwo przechodzić między różnymi modułami systemu – od zarządzania ofertami, poprzez rezerwacje i płatności, aż po zarządzanie użytkownikami. Jasno zdefiniowane komunikaty oraz mechanizmy walidacji wprowadzanych danych zapewniają stabilność i przejrzystość działania systemu, co stanowi solidną podstawę do dalszej rozbudowy, na przykład o graficzny interfejs użytkownika lub dodatkowe mechanizmy zabezpieczeń.

4.3 Poszczególne opcje programu

```
=== Zarządzanie użytkownikami ===  
1. Dodaj użytkownika  
2. Edytuj użytkownika  
3. Usuń użytkownika  
4. Pokaż wszystkich użytkowników  
0. Powrót do głównego menu  
Wybierz opcję:
```

Rysunek 4.4: Zarządzanie użytkownikami

```
=== Zarządzanie płatnościami ===  
1. Dodaj płatność  
2. Pokaż wszystkie płatności  
3. Generuj fakturę  
0. Powrót do głównego menu  
Wybierz opcję: 
```

Rysunek 4.5: Zarządzanie płatnościami

Rozdział 5

Podsumowanie

Projekt systemu zarządzania biurem podróży, napisany w języku C#, stanowi przykład rozwiązania informatycznego opartego na modularnej architekturze, w której kluczową rolę odgrywają trzy główne warstwy: modele danych, logika biznesowa oraz interfejs użytkownika. Modele danych, takie jak Oferta, Rezerwacja, Płatność oraz Użytkownik, służą do przechowywania podstawowych informacji niezbędnych dla funkcjonowania systemu. Dedykowane klasy zarządzające, odpowiadające za operacje CRUD oraz dodatkowe funkcjonalności, jak tworzenie kopii zapasowych, generowanie faktur czy kontrola dostępu, umożliwiają sprawną obsługę danych. Warstwa interfejsu, mimo że opiera się na konsolowym menu, zapewnia intuicyjną i przejrzystą interakcję, umożliwiając użytkownikowi łatwą nawigację między poszczególnymi modułami systemu.

Realizacja projektu wiązała się z wieloma wyzwaniami, z których najważniejsze dotyczyły stworzenia spójnej architektury systemu oraz zapewnienia integralności danych przechowywanych w formacie JSON. Przypisywanie unikalnych identyfikatorów oraz wdrożenie mechanizmu tworzenia kopii zapasowych wymagało precyzyjnego podejścia i iteracyjnego testowania. Dodatkowo, walidacja danych wprowadzanych przez użytkowników oraz projektowanie czytelnego interfejsu konsolowego stanowiły istotne aspekty, których celem było zminimalizowanie błędów i zapewnienie stabilności działania systemu.

Podsumowując, projekt umożliwia skuteczne zarządzanie ofertami turystycznymi, rezerwacjami oraz rozliczeniami finansowymi, stanowiąc solidną bazę do dalszych rozwoju. W przyszłości system można udoskonalić poprzez migrację danych do dedykowanej bazy, integrację z zewnętrznymi platformami płatności, wdrożenie graficznego interfejsu użytkownika oraz implementację zaawansowanych mechanizmów bezpieczeństwa, takich jak dwuskładnikowe uwierzytelnianie i szyfrowanie danych. Takie usprawnienia pozwolą na obsługę większych obciążeń oraz podniosą poziom ochrony informacji, co wpłynie na zwiększenie efektywności i atrakcyjności systemu na rynku.

Bibliografia

- [1] Sommerville, I. (2015). *Inżynieria oprogramowania*. Wydawnictwo Helion.
- [2] Martin, R. C. (2009). *Czysty kod. Podręcznik dobrego programisty*. Wydawnictwo Helion.
- [3] Sutherland, J. (2014). *Scrum. Zwinne zarządzanie projektami*. Wydawnictwo Helion.
- [4] Microsoft. (n.d.). *C# Programming Guide*. Dostępne pod adresem: <https://docs.microsoft.com/pl-pl/dotnet/csharp/>.
- [5] Microsoft. (n.d.). *System.Text.Json Namespace*. Dostępne pod adresem: <https://docs.microsoft.com/pl-pl/dotnet/api/system.text.json>.
- [6] JSON.org. (n.d.). *JSON: The JavaScript Object Notation*. Dostępne pod adresem: <https://www.json.org/json-en.html>.
- [7] Agile Manifesto. (2001). *Manifesto for Agile Software Development*. Dostępne pod adresem: <https://agilemanifesto.org>.

Spis rysunków

2.1	Diagram klas użytych w programie.	7
3.1	Diagram Gantta przedstawiający harmonogram realizacji projektu	12
4.1	Menu startowe aplikacji	14
4.2	Zarządzanie ofertami	15
4.3	Zarządzanie rezerwacjami	16
4.4	Zarządzanie użytkownikami	18
4.5	Zarządzanie płatnościami	18

Spis tabel