Dokumentacja projektu c#

1. Tytuł projektu

Inp0st_org to aplikacja do zarządzania paczkami, użytkownikami oraz dostawcami w firmie dostawczej.

2. Opis projektu

Cel projektu:

Nasza aplikacja ma za cel stworzenie przejrzystego, funkcjonalnego i łatwego w obsłudze systemu obsługującego firmy pocztowe i dostawcze. Będzie ułatwiać zarządzanie i obsługę takich firm.

Co robi aplikacja:

Umożliwia logowanie na konto z odpowiednimi rolami, dodawanie, usuwanie i aktualizowanie oraz wyszukiwanie i wyświetlanie kont użytkowników. Pozwala także na zarządzanie paczkami: zamawianie ich, zmienianie ich statusu, usuwanie, wyszukiwanie i wyświetlanie ich.

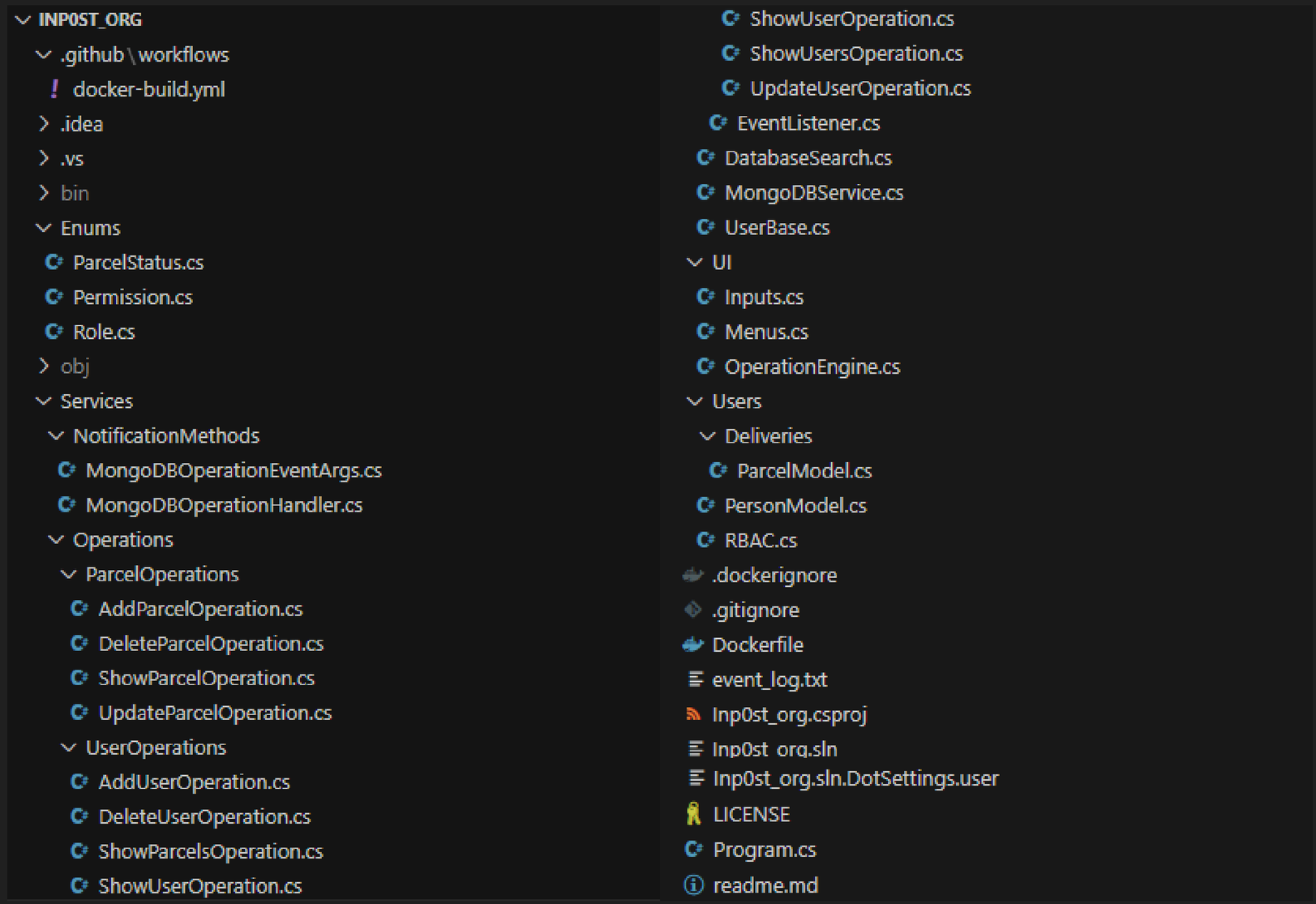
Dla kogo jest przeznaczona:

-Firmy zajmujące się dostarczaniem paczek(poczty, paczkomaty oraz inne placówki)
-Sklepy internetowe, które zawierają funkcję dostarczania
-Osoby, które natrafią na projekt na GitHubie i będą chciały się czegoś nauczyć

3. Technologie

Język programowania: C#
Środowisko: .NET 9.0
IDE: Rider
Inne biblioteki/narzędzia: Docker, MongoDB biblioteka: MongoDB.Driver, zmienne środowiskowe biblioteka: dotenv.net

4. Struktura katalogów


INP0ST_ORG
.github\workflows
docker-build.yml
.idea
.vs
bin
Enums
ParcelStatus.cs
Permission.cs
Role.cs
obj
Services
NotificationMethods
MongoDBOperationEventArgs.cs
MongoDBOperationHandler.cs
Operations
ParcelOperations
AddParcelOperation.cs
DeleteParcelOperation.cs
ShowParcelOperation.cs
UpdateParcelOperation.cs
UserOperations
AddUserOperation.cs
DeleteUserOperation.cs
ShowParcelsOperation.cs
ShowUserOperation.cs
ShowUserOperation.cs
ShowUsersOperation.cs
UpdateUserOperation.cs
EventListener.cs
DatabaseSearch.cs
MongoDBService.cs
UserBase.cs
UI
Inputs.cs
Menus.cs
OperationEngine.cs
Users
Deliveries
ParcelModel.cs
PersonModel.cs
RBAC.cs
.dockerignore
.gitignore
Dockerfile
event_log.txt
Inp0st_org.csproj
Inp0st_org.sln
Inp0st_org.sln.DotSettings.user
LICENSE
Program.cs
readme.md

.github/workflows - jest to folder, który zawiera w sobie operacje, które dzieją się po jakimś danym wydazeniu w repozytorium

docker-build.yml - za każdym spushowaniem na maina buduje się zaktualiowany obraz Dockera

Enums - folder zawierający w sobie pliki przechowujące enumy:
Role.cs, Permission.cs, ParcelStatus.cs

Services - zawarte są w nim klasy odpowiedzialne za inicjację połączenia z bazą danych, klasy odpowiedzialne za operację na tej bazie danych oraz klasy odpowiedzialne za powiadomienia

NotificationMethods - folder zawierający w sobie delegaty i argumenty dla eventów

MongoDBOperationEventArgs - dziedziczy po EventArgs zawiera właściwości powiązane z powiadomieniami w celu skrócenia i ułatwienia przekazywania danych w eventach

MongoDBOperationHandeler - dwa delegaty dla powiadomień związanych z paczkami i z użytkownikami

Operations - wszystkie operacje podzielone ze względu na 2 kolekcje w bazie danych

ParcelOperations - zawiera w sobie pliki odpowiadające za operacje na paczkach: AddParcelOperation.cs, DeleteParcelOperation.cs, ShowParcelOperation.cs, UpdateParcelOperation.cs

UserOperations - zawiera w sobie pliki odpowiadające na operacje na użytkownikach: AddUserOperation.cs, DeleteUserOperation.cs, ShowUserOperation.cs, ShowUsersOperation.cs, ShowParcelsOperation.cs

EventListener.cs - posiada 2 metody które zostają wykonane po wykonaniu operacji na bazie danych oraz system zapisu do pliku .txt

MongoDBServive.cs - klasa, której instancja jest połączeniem z bazą danych

UserBase.cs - zawiera 2 klasy bazowe dla operacji na paczkach oraz użytkownikach

UI - folder który zawiera w sobie pliki odpowiadające za: wyświetlanie różnyh menu(Menus.cs), zarządzanie różnymi inputami podawanymi w czasie działania aplikacji(Inputs.cs) oraz wszelkimi operacjami jakie się dzieją(OperationEngine.cs)

Users - zawiera modele obiektów z bazy oraz klasę RBAC

Program.cs - plik inicjujący pracę programu

event_log.txt - plik przechowujący w sobie logi z operacji

5. Instrukcja instalacji i uruchomienia

*Aplikacja wymaga połączenia internetowego by funkcjonować, ponieważ operacje wykonywane są na klastrze MongoDB Atlas.

Wszystkie instrukcje znajdują się na githubie oraz w pliku readme.

Uruchomienie aplikacji z .zip

Wypakowujemy aplikację i w konsoli w głównym jej katalogu wykonujemy **dotnet build** by zbudować aplikację i komendę **dotnet run** by ją uruchomić, lub otwieramy plik .sln w edytorze (zalecamy ridera). Plik .env jest już zawarty w skompresowanym projekcie.

Ściągnięcie aplikacji z repozytorium Dockera (zalecane)

W konsoli (bez znaczenia na lokalizację) wpisujemy: **docker pull guc10/inp0st_org,** operacja ściągnie gotowy obraz na komputer, który uruchomimy za pomocą komendy: **docker run -it --rm --name Inp0st_org_container --env-file .env guc10/inp0st_org.** Flaga -it oznacza responsywność w konsoli z możliwością podawania danych --rm usuwa kontener po zakończeniu pracy, --name nazywa kontener, a --env-file .env wskazuje, że zostanie użyty plik ze zmiennymi środowiskowymi. **Docker run musi zostać wykonany** w lokalizacji, gzie znajduje się ów plik .env ze zmiennymi: **DATABASE_USER=** i **DATABASE_PASSWORD=,** dane do logowania są podane w komentarzu na moodlu.

*Wymagana jest aplikacja Dockera

Ściągnięcie aplikacji z githuba

W konsoli git bash ściagamy repozytorium: **git clone https://github.com/ZSK-octoTeam/Inp0st_org.git**, następnie w konsoli w głównym jej katalogu wykonujemy **dotnet build** by zbudować aplikację i komendę **dotnet run** by ją uruchomić. Aplikacja wymaga utworzenia pliku .env w głównym katalogu aplikacji ze zmiennymi: **DATABASE_USER=** oraz **DATABASE_PASSWORD=**, których wartości są podane w komentarzu na moodlu.

Logowanie do aplikacji

Program po udanym zpingowaniu bazy danych, spyta się o użytkownika i hasło, w bazie jest 4 użytkowników, ale adminem jest: **admin**, o haśle: **pass**.

Reszta użytkowników to:
Paweł - pass
Janusz - pass
Mohamed - pass

6. Opis działania aplikacji

1) **Logowanie** – pozwala na zalogowanie się do danego utworzonego konta z przydzielonymi rolami, jeżeli podane dane są poprawne przenosi nas do odpowiedniego menu w przeciwnym przypadku ponownie prosi o podanie danych.

Dane:

-Nazwa użytkownika

-Hasło

2) **Wybierz z menu, podmenu** - użytkownik podaje liczbę z zakresu podanego menu, która potem przeniesie do odpowiedniego podmenu lub funkcji.
Dane:
-Liczba wyboru

3) **Dodaj klienta/dostawcę** - pozwala dodać nowego klienta/dostawcę lub dodać rolę jeżeli taki użytkownik istnieje ale nie ma takiej roli
Dane:
-Nazwa użytkownika
-Hasło

4) **Dodaj użytkownika** - funkcja pozwalająca dodać użytkownika do systemu
Dane:
-Nazwa użytkownika
-Rola
-Hasło

5) **Usuń klienta/dostawcę/użytkownika** - pozwala to na usunięcie użytkownika z systemu lub usunięcia mu roli w przypadku gdy wybraliśmy opcję usuń klientaa/dostawcę, a użytkownik ten ma więcej niż 1 rolę
Dane:
-Nazwa użytkownika

6) **Pokaż klientów/dostawców/użytkowników** - wyświetla w konsoli wszystkich użytkowników/klientów/dostawców ułożonych jeden pod drugim

7) **Zaktualizuj klienta/dostawcę/użytkownika** - pozwala na aktualizację hasła podanego użytkownika, nie można zmienić nowego hasła na takie same jakie było poprzednio
Dane:
-Nazwa użytkownika
-Nowe hasło

8) **Wyszukaj klientów/dostawców/użytkowników** - wyświetla informacje o podanym użytkowniku
Dane:
-Nazwa użytkownika

9) **Usuń paczkę** - pozwala na usunięcie podanej paczki z systemu, w przypadku gdy nie jesteśmy zalogowani jako admin paczka musi należeć do nas
Dane:
-Nazwa paczki

10) **Dodaj paczkę** - umożliwia dodanie nowej paczki do systemu oraz przypisanie dostawcy i odbiorcy, trzeba podać użytkowników, którzy mają uprawnienia do obierania, dostarczania paczek
Dane:
-Nazwa odbiorcy
-Nazwa paczki
-Nazwa dostawcy

11) **Zaktualizuj paczkę** - pozwala na zmianę dostawcy lub statusu podanej paczki
Dane:
-Nazwa paczki
-Nazwa nowego dostawcy
-Nowy status

12) **Pokaż paczki/pokaż moje paczki** – pokazuje wszystkie paczki w systemie albo takie paczki w których zalogowany użytkownik jest odbiorcą albo dostawcą.

13) **Wyszukaj paczkę** - pozwala na wyświetlenie informacji o podanej paczce, w przypadku gdy zalogowany użytkownik nie jest adminem można wyszukiwać tylko swoje paczki
Dane:
-Nazwa paczki

14) **Przyjmij/dostarcz paczkę** - aktualizuje podaną paczkę tak, że zmienia status na W Transporcie/Dostarczona oraz dostawcę na użytkownika, który wykonuje operację
Dane:
-Nazwa paczki

15) **Zamów paczkę** - umożliwia zamawianie paczki przez użytkownika, dodaje podaną paczkę do systemu i ustawia odbiorcę jako osobę która wykonywała operację
Dane:
-Nazwa paczki

7. Zrzuty ekranu

```
=== LOG IN ===
Enter your username:
gustaw
Enter your password:
*****
```

1) Logowanie

```
=== MENU ===
1. Menage clients
2. Menage deliverers
3. Menage packages
4. Menage users
5. Log out
6. Exit
Enter your choice:
```

2) Menu główne

```
=== CLIENTS MENU ===
1. Add client
2. Show all clients
3. Delete client
4. Update client
5. Search client
6. Back
7. Exit
Enter your choice:
```

3) Menu zarządzania klientami

```
=== DELIVERERS MENU ===
1. Add deliverer
2. Show all deliverers
3. Delete deliverer
4. Update deliverer
5. Search deliverer
6. Back
7. Exit
Enter your choice:
```

4) Menu zarządzania dostawcami

```
=== USERS MENU ===
1. Add user
2. Show all users
3. Update user
4. Delete user
5. Search user
6. Back
7. Exit
Enter your choice:
```

5) Menu zarządzania użytkownikami

```
=== PACKAGES MENU ===
1. Add package
2. Show all packages
3. Delete package
4. Update package
5. Search package
6. Back
7. Exit
Enter your choice:
```

6) Menu zarządzania paczkami

```
=== MENU ===
1. Menage packages
2. Log out
3. Exit
Enter your choice:
```

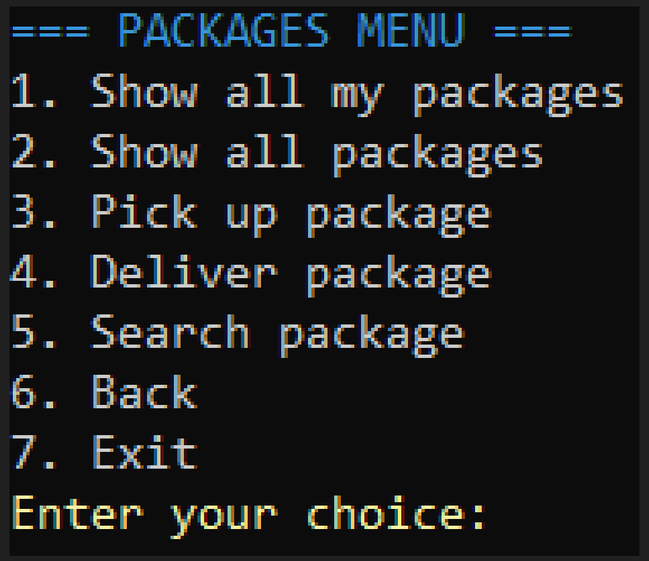
7) Menu główne klienta/dostawcy

```
=== Customer and Deliverer Menu ===
1. Menage packages(deliverer)
2. Menage packages(client)
3. Log out
4. Exit
Enter your choice:
```

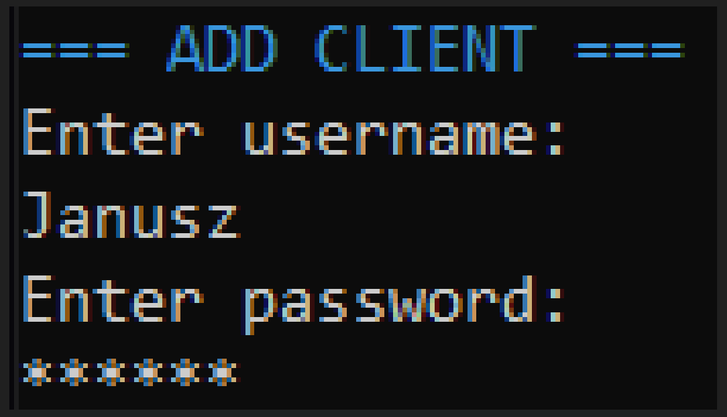
8) Menu główne klienta i dostawcy

```
=== PACKAGES MENU ===
1. Order package
2. Show all my packages
3. Cancel package
4. Search package
5. Back
6. Exit
Enter your choice:
```

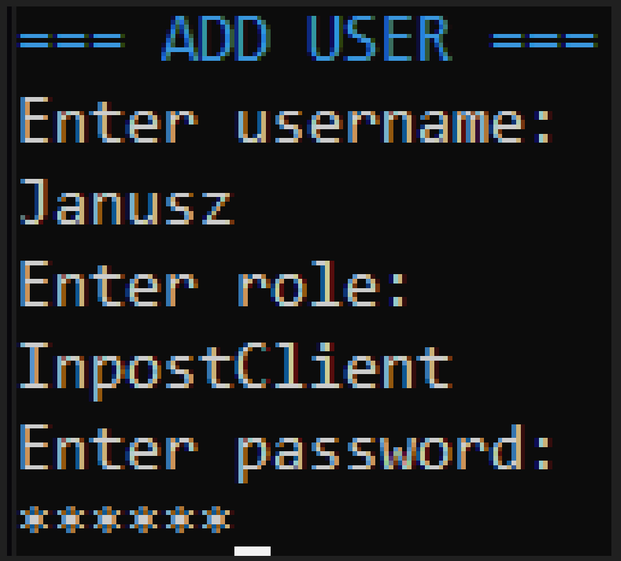
9) Menu zarządzania paczkami(klient)



10) Menu zarządzania paczkami(dostawca)



11) Dodaj klienta/dostawcę



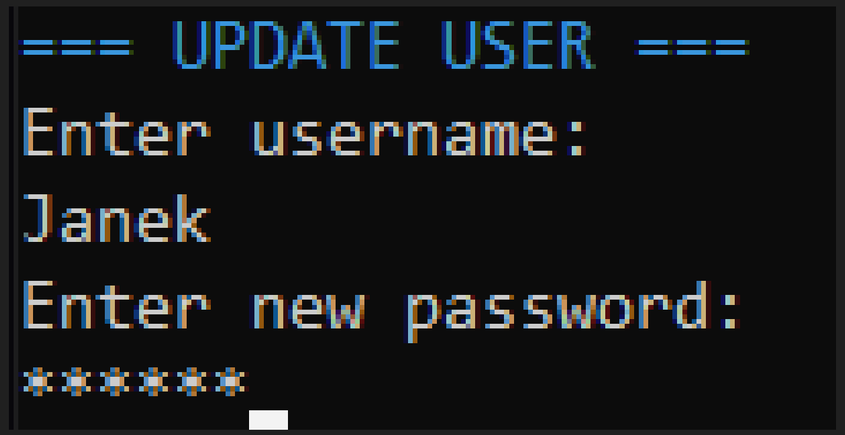
12) Dodaj użytkownika

```
=== DELETE CLIENT ===
Enter username:
Halina
```

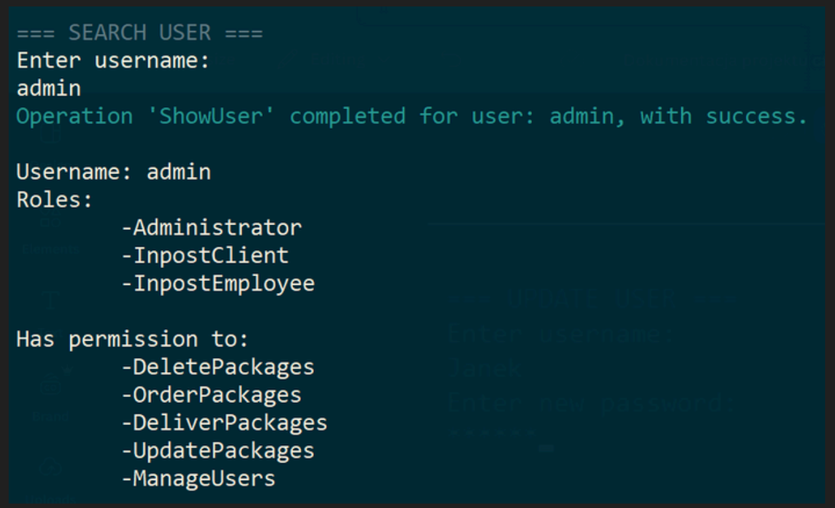
13) Usuń klienta/dostawcę/użytkownika

```
Operation 'Show users' completed for user: gustaw, with success.
Showing users:

- gustaw - roles:
        'Administrator'
        'InpostClient'
        'InpostEmployee'

- Mohamed - roles:
        'InpostClient'
        'InpostEmployee'

- Kaśka - roles:
        'InpostEmployee'

- Janusz - roles:
        'InpostClient'
```

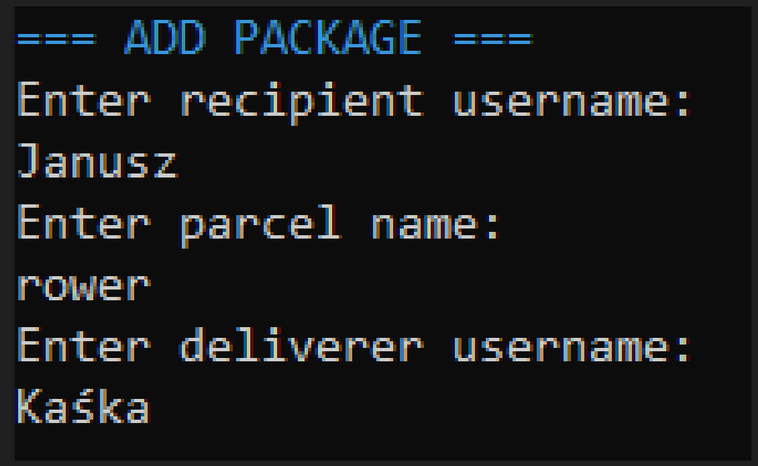
14) Pokaż klientów/dostawców/użytkowników



15) Zaktualizuj klienta/dostawcę/użytkownika



16) Wyszukaj klienta/dostawcę /użytkownika



17) Dodaj paczkę

```
=== DELETE PACKAGE ===
Enter parcel name:
wór ziemniaków
```

18) Usuń paczkę

```
=== UPDATE PACKAGE ===
Enter parcel name:
wór ziemniaków
Enter new deliverer username:
Kaśka
Enter new status:
InTransport
```

19) Zaktualizuj paczkę

```
=== PICK UP PACKAGE ===
Enter parcel name:
telewizor
```

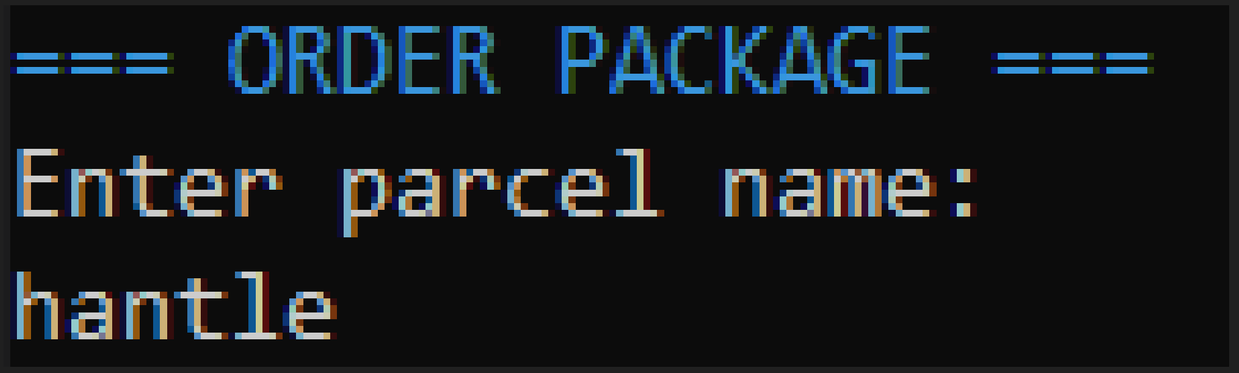
20) Przygarnij/dostarcz paczkę

```
=== PACKAGES ===
Operation 'Show parcels' completed for user: gustaw, with success.
Showing parcels:
Parcel name: hantle, parcel status: InWarehouse, parcel sender: gustaw, parcel reciever: gustaw
Parcel name: rower, parcel status: InWarehouse, parcel sender: Kaśka, parcel reciever: Janusz
```

21) Pokaż paczki/pokaż moje paczki

```
=== SEARCH PACKAGE ===
Enter parcel name:
hantle
Operation 'ShowParcel' completed for parcel: hantle, with status success.
Parcel name: hantle
Parcel status: InWarehouse
Parcel owner: gustaw
Parcel deliverer: gustaw
```

22) Wyszukaj paczkę



23) Zamów paczkę

8. Przykłady użycia

1) Logowanie do aplikacji:

-Uruchamiamy aplikację

-Podajemy dane do logowania(nazwa, hasło)

-Zostajemy przeniesieni na odpowiednie menu

2) Obsługa menu:

2.1 Menu/podmenu klienta/dostawcy/klienta i dostawcy:

- Wybieramy jedną z podanych opcji
- Jeżeli aplikacja tego wymaga podajemy potrzebne dane(nazwa paczki)
- Podana funkcja zostanie wykonana lub zostajemy przeniesieni do odpowiedniego podmenu
- Powrócimy do ostatniego menu/podmenu w przypadku gdy podaliśmy funkcję

2.2 Menu/podmenu administratora:

- Wybieramy jedną z dostępnych opcji
- Jeżeli aplikacja wymaga podajemy potrzebne dane(nazwa użytkownika, hasło, rola, nazwa paczki, status, nazwa dostawcy, nazwa odbiorcy)
- Podana funkcja zostanie wykonana lub zostajemy przeniesieni do odpowiedniego podmenu
- Powrócimy do ostatniego menu/podmenu w przypadku gdy podaliśmy funkcję

9. Struktury danych i klas

Klasy User/ParcelOperations i User/ParcelBase

Wszystkie te klasy dziedziczą po abstrakcyjnej klasie User/ParcelBase, która definiuje wspólne elementy operacji, takie jak zdarzenie Notify i metodę abstrakcyjną Operation. Przykładem jest klasa AddParcelOperation, która implementuje metodę Operation do dodawania paczek do bazy MongoDB, sprawdzając unikalność paczki, istnienie użytkownika oraz uprawnienia odbiorcy. Dziedziczenie zapewnia spójność w obsłudze operacji i zdarzeń.

Person i ParcelModel

Klasa PersonModel reprezentuje użytkownika systemu, przechowując jego dane, takie jak nazwa użytkownika, hasło oraz lista ról (Roles). Umożliwia dodawanie ról za pomocą metody AddRole, zapewniając, że role nie będą się powtarzać. Każdy użytkownik ma unikalny identyfikator (Id) przechowywany jako ObjectId w MongoDB.

Klasa ParcelModel reprezentuje paczkę w systemie, zawierając informacje o nazwie paczki (ParcelName), nadawcy (Sender), odbiorcy (Recipient) oraz statusie (Status). Umożliwia zmianę statusu paczki za pomocą metody ChangeStatus oraz zmianę nadawcy za pomocą metody ChangeSender. Podobnie jak w przypadku użytkownika, każda paczka ma unikalny identyfikator (Id) przechowywany jako ObjectId w MongoDB.

RBAC

Klasa RBAC (Role-Based Access Control) implementuje mechanizm kontroli dostępu oparty na rolach użytkowników. Przechowuje mapowanie ról (Role) na listy uprawnień (Permission) w prywatnym słowniku _rolePermissions. Posiada metoda HasPermission, która sprawdza, czy użytkownik (PersonModel) posiada określone uprawnienie (Permission). Iteruje przez role użytkownika i weryfikuje, czy którakolwiek z nich zawiera wymagane uprawnienie.

Inputs/Menus/OperationEngine

Inputs to klasa, która jest odpowiedzialna za obsługę danych wejściowych w systemie. Obejmuje pobieranie danych od użytkownika takich jak: string, hasło lub int, walidację wprowadzonych danych, koloruje on różne elementy w konsoli oraz zawiera metodę **CheckCredentials**, która sprawdza poprawność podanych danych podczas logowania przez metodę **LogIn.**
Menus to komponent zarządzający interfejsem menu w aplikacji. Zawiera metody, które wyświetlają odpowiednie menu oraz wywołują odpowiednie inicjatory operacji w OperationEngine.
OperationEngine to klasa zarządzająca logiką operacyjną systemu. W jej metodach wywołyuwane są operacje, wyświetlane powiadomienia proszące o podanie danych oraz w niektórych sprawdzenie poprawności danych dla zapobiegnięcia powstania konfliktów w bazie danych.

MongoDBService

Klasa MongoDBService odpowiada za obsługę operacji na bazie danych MongoDB, implementując interfejs IMongoDBService, który definiuje metody Connect i ListCollections. Konstruktor klasy inicjalizuje nazwę bazy danych oraz kolekcji, a także tworzy łańcuch połączenia na podstawie zmiennych środowiskowych załadowanych z pliku .env. Metoda Connect nawiązuje połączenie z bazą danych, inicjalizuje kolekcje użytkowników (collectionUsers) i paczek (collectionParcels), a także wykonuje testowe "pingowanie" bazy w celu weryfikacji poprawności połączenia. W przypadku błędnych danych uwierzytelniających program zostaje zatrzymany. Metoda ListCollections wyświetla listę wszystkich kolekcji w bazie danych. Klasa zapewnia centralny punkt dostępu do bazy MongoDB, umożliwiając zarządzanie kolekcjami i operacjami na danych.

DatabaseSearch

Klasa DatabaseSearch odpowiada za wyszukiwanie danych w bazie MongoDB, korzystając z instancji MongoDBService. Udostępnia statyczne metody do pobierania użytkowników i paczek z bazy danych oraz funkcję do haszowania haseł. Metoda FindUsers zwraca wszystkich użytkowników w postaci słownika, gdzie kluczem jest nazwa użytkownika, a wartością obiekt PersonModel. Podobnie, metoda FindParcels zwraca wszystkie paczki w postaci słownika, gdzie kluczem jest nazwa paczki, a wartością obiekt ParcelModel. Funkcja HashPassword umożliwia haszowanie haseł za pomocą algorytmu SHA-256, zwracając wynik w formacie Base64. Klasa ta pełni rolę pomocniczą, ułatwiając interakcję z danymi przechowywanymi w bazie MongoDB.

EventListener

Klasa EventListener odpowiada za obsługę zdarzeń związanych z operacjami na użytkownikach i paczkach w systemie, dziedzicząc funkcjonalność z klasy LogManager. Metody OnUserOperation i OnParcelOperation obsługują odpowiednio zdarzenia związane z użytkownikami i paczkami, wyświetlając wyniki operacji w konsoli oraz zapisując szczegóły zdarzeń do pliku logów. W przypadku operacji na użytkownikach metoda OnUserOperation wyświetla komunikat o powodzeniu lub niepowodzeniu operacji, uwzględniając nazwę użytkownika lub wskazując na administratora, jeśli użytkownik jest null. Analogicznie, metoda OnParcelOperation obsługuje zdarzenia związane z paczkami, wyświetlając komunikaty o statusie operacji oraz nazwie paczki. Obie metody korzystają z metody LogToFile z klasy bazowej LogManager, aby zapisać szczegóły zdarzenia do pliku event_log.txt.

10. Obsługa błędów

```
=== MENU ===
1. Menage clients
2. Menage deliverers
3. Menage packages
4. Menage users
5. Log out
6. Exit
Enter your choice:
8
Invalid input. Please enter a correct number:
```

1) Użytkownik poda niewłaściwy numer do menu

```
=== ADD USER ===
Enter username:
Janusz
Enter role:
kierownik
Invalid role. Please enter one of following roles:
-Administrator
-InpostClient
-InpostEmployee
Enter role:
```

2) Użytkownik poda wartość, która nie może się tam znaleźć

```
=== DELETE CLIENT ===
Enter username:
PanDyrektor
Operation 'DeleteUser' completed for user: PanDyrektor, with failure.
User does not exist.
```

3) Próbujemy usunąć, zmodyfikować, znaleźć użytkownika lub paczkę, która nie znajduje się w bazie danych

```
=== LOG IN ===
Enter your username:
uzytkownik
Enter your password:
********
Log in failed. User not found.
```

4) Niepoprawne logowanie

```
=== ADD USER ===
Enter username:
gustaw
Enter role:
Administrator
Enter password:
****
Operation 'AddUser' completed for user: gustaw, with failure.
User already exists.
```

5) Próbujemy dodać użytkownika lub paczkę, który już jest w bazie danych

```
=== UPDATE USER ===
Enter username:
gustaw
Enter new password:
****
Operation 'UpdateUser' completed for user: gustaw, with failure.
New password is the same as the old one.
```

6) Zmieniamy użytkownikowi hasło na takie same jakie posiadał wcześniej

```
=== UPDATE CLIENT ===
Enter username:
Mohamed
Enter new password:
kiribati
Operation 'UpdateUser' completed for user: Mohamed, with failure.
User is not a InpostClient.
```

7) Próbujemy działać na użytkowniku, który nie spełnia wymagań(np. Użycie funkcji Modify Client na użytkowniku, który nie jest klientem)

```
=== SEARCH PACKAGE ===
Enter parcel name:
hantle
Operation 'ShowParcel' completed for parcel: hantle, with status failure.
User: Mohamed does not have a parcel called: hantle
```

8) Działamy na paczce która nie jest nasza

11. Testowanie

Każda możliwa funkcja i przypadki zostały przetestowane ręcznie, a wyniki testów zostały zapisane w pliku event_log,txt

12. Problemy i ograniczenia

- Całość kodu na pewno da się jeszcze bardziej zoptymalizować, usprawnić, przyspieszyć niektóre działania
- W paczkach statusy oraz dostawców można zmieniać dowolnie, niezależnie od jej aktualnego stanu(np. Z w Magazynie na Dostarczona), co mogłoby wprowadzać chaos w rzeczywistej firmie
- Menu może być bardziej przejrzyste i lepiej złączone w przypadku gdy użytkownik jest zarówno dostawcą jak i klientem

13. Plany rozwoju

- Dodanie wsparcia na przyszłe wersje .NET
- Rozbudowanie programu o jeszcze więcej funkcji związanych z zarządzaniem paczkami
- Wprowadzanie zmian i aktualizacji w oparciu na otrzymany feedback

14. Autorzy

Autorzy:
Filip Maciejewski oraz Gustaw Grześkowiak

Kontakt:
filip.maciejewski@uczen.zsk.poznan.pl
gustaw.grzeskowiak@uczen.zsk.poznan.pl