Wolfie Airlines

Wygenerowano przez Doxygen 1.9.8

Rozdział 1

Indeks hierarchiczny

1.1 Hierarchia klas

Ta lista dziedziczenia posortowana jest z grubsza, choć nie całkowicie, alfabetycznie:

Rozdział 2

Indeks klas

2.1 Lista klas

Tutaj znajdują się klasy, struktury, unie i interfejsy wraz z ich krótkimi opisami:

Rozdział 3

Indeks plików

3.1 Lista plików

Tutaj znajduje się lista wszystkich udokumentowanych plików z ich krótkimi opisami:

Rozdział 4

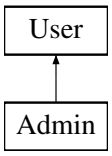
Dokumentacja klas

4.1 Dokumentacja klasy Admin

Ta klasa reprezentuje użytkownika admina.

`#include <admin.h>`

Diagram dziedziczenia dla Admin



Metody publiczne

- Admin (const std::string &username, const std::string &email, double discount, const std::string &discount↩ _type, const std::string &premium_card, const std::string &payment_method, mongocxx::client &client, const std::string &profession, const std::string &registration_date, double money_spent, double money_saved, int ticket_bought, const std::vector< bsoncxx::document::value > &user_flights, bool is_admin, std::string hashed_admin_password)

  Konstruuje nowy obiekt Admin.
- Admin (const User &user)

  Konstruuje nowy obiekt Admin z obiektu User.
- void AddFlight (User &user)

  Dodaje lot.
- void AddVerificationQuestion (User &user)

  Dodaje pytanie weryfikacyjne.
- void ManageUsers (User &user)

  Zarządza użytkownikami.
- void AddLuggageItem (User &user)

  Dodaje przedmiot bagażu.

Metody publiczne dziedziczone z User

- User (mongocxx::client &client)

  Konstruuje nowy obiekt User.
- **User** (std::string username, std::string email, double discount, std::string discount_type, std::string premium↩ _card, std::string payment_method, mongocxx::client &client, std::string profession, std::string registration↩ _date, double money_spent, double money_saved, int ticket_bought, std::vector< bsoncxx::document::value > user_flights, bool is_admin)

  Konstruuje nowy obiekt User z określonymi parametrami.
- void **Reset** ()

  Resetuje użytkownika.
- mongocxx::collection & **GetCollection** ()

  Zwraca kolekcję.
- mongocxx::collection **GetSpecificCollection** (const std::string &collection_name)

  Zwraca określoną kolekcję.
- std::string **GetPassword** ()

  Zwraca hasło.
- void **SetPassword** (const std::string &password)

  Ustawia hasło.
- void **SetPremiumCard** (User &user, const std::string &card)

  Ustawia kartę premium.
- void **SetBlik** (const std::string &payment_method)

  Ustawia metodę płatności Blik.
- void **SetVisa** (const std::string &card_number, const std::string &card_cvv)

  Ustawia metodę płatności Visa.
- void **ChangeUsername** (const std::string &username)

  Zmienia nazwę użytkownika.
- void **ChangeEmail** (const std::string &email)

  Zmienia email.
- void **ChangePassword** (const std::string &password)

  Zmienia hasło.
- void **SetDiscount** (double discount, const std::string &discount_type)

  Ustawia zniżkę.
- double **GetDiscount** () const

  Zwraca zniżkę.
- std::string **RecognizeDiscount** () const

  Rozpoznaje zniżkę.
- void **AddTicketToUser** (const std::vector< int > &seats, const FlightConnection &flight_connection)

  Dodaje bilet do użytkownika.
- void **UpdateMoneySaved** (double normal_price, double discount_price)

  Aktualizuje zaoszczędzone pieniądze.
- Admin * **LoginAsAdmin** ()

  Loguje jako admin.
- bool **CheckIfAdmin** () const

  Sprawdza, czy użytkownik jest administratorem.
- void **SetIsAdmin** (bool is_administrator)

  Ustawia, czy użytkownik jest administratorem.
- void **LuggageCheckin** (int flight_number)

  Odprawia bagaż.
- mongocxx::cursor **FindUserInDatabase** ()

  Znajduje użytkownika w bazie danych.
- template<typename T >
  void UpdateUserInDatabase (const std::string &value_in_database, const T &value_to_set)

  Aktualizuje użytkownika w bazie danych.

Atrybuty publiczne

- std::string **hashed_admin_password_**

  Zaszyfrowane hasło admina.

Atrybuty publiczne dziedziczone z User

- std::string **username_**

  Nazwa użytkownika.
- std::string **profession_**

  Zawód użytkownika.
- std::string **email_**

  Email użytkownika.
- std::string **discount_type_**

  Typ zniżki użytkownika.
- double **discount_**

  Wartość zniżki użytkownika.
- std::string **premium_card_**

  Karta premium użytkownika.
- std::string **payment_method_**

  Metoda płatności użytkownika.
- std::string **registration_date_**

  Data rejestracji użytkownika.
- double **money_spent_**

  Całkowita kwota wydana przez użytkownika.
- double **money_saved_**

  Całkowita kwota zaoszczędzona przez użytkownika.
- int **ticket_bought_**

  Całkowita liczba biletów kupionych przez użytkownika.
- std::vector< bsoncxx::document::value > **user_flights_**

  Loty użytkownika.
- bool **is_admin_**

  Czy użytkownik jest administratorem.

Dodatkowe Dziedziczone Składowe

Atrybuty chronione dziedziczone z User

- mongocxx::client & **_client**

  Klient MongoDB.

4.1.1 Opis szczegółowy

Ta klasa reprezentuje użytkownika admina.

Dziedziczy po klasie User i dodaje dodatkowe funkcje specyficzne dla adminów.

4.1.2 Dokumentacja konstruktora i destruktora

4.1.2.1 Admin() [1/2]

```
Admin::Admin (
            const std::string & username,
            const std::string & email,
            double discount,
            const std::string & discount_type,
            const std::string & premium_card,
            const std::string & payment_method,
            mongocxx::client & client,
            const std::string & profession,
            const std::string & registration_date,
            double money_spent,
            double money_saved,
            int ticket_bought,
            const std::vector< bsoncxx::document::value > & user_flights,
            bool is_admin,
            std::string hashed_admin_password )
```

Konstruuje nowy obiekt Admin.

Parametry

username	Nazwa użytkownika admina.
email	Email admina.
discount	Dostępna dla admina zniżka.
discount_type	Typ dostępnej dla admina zniżki.
premium_card	Karta premium admina.
payment_method	Metoda płatności admina.
client	Klient MongoDB.
profession	Zawód admina.
registration_date	Data rejestracji admina.
money_spent	Kwota wydana przez admina.
money_saved	Kwota zaoszczędzona przez admina.
ticket_bought	Liczba biletów kupionych przez admina.
user_flights	Loty admina.
is_admin	Flaga wskazująca, czy użytkownik jest adminem.
hashed_admin_password	Zaszyfrowane hasło admina.

4.1.2.2 Admin() [2/2]

```
Admin::Admin (
            const User & user )  [inline]
```

Konstruuje nowy obiekt Admin z obiektu User.

Parametry

user	Obiekt User.

4.1.3 Dokumentacja funkcji składowych

4.1.3.1 AddFlight()

```
void Admin::AddFlight (
            User & user )
```

Dodaje lot.

Parametry

user	Użytkownik, do którego dodawany jest lot.

4.1.3.2 AddLuggageItem()

```
void Admin::AddLuggageItem (
            User & user )
```

Dodaje przedmiot bagażu.

Parametry

user	Użytkownik, do którego dodawany jest przedmiot bagażu.

4.1.3.3 AddVerificationQuestion()

```
void Admin::AddVerificationQuestion (
            User & user )
```

Dodaje pytanie weryfikacyjne.

Parametry

user	Użytkownik, do którego dodawane jest pytanie weryfikacyjne.

4.1.3.4 ManageUsers()

```
void Admin::ManageUsers (
            User & user )
```

Zarządza użytkownikami.

Parametry

user	Użytkownik do zarządzania.

Dokumentacja dla tej klasy została wygenerowana z plików:

- admin/admin.h
- admin/admin.cpp

4.2 Dokumentacja klasy Authentication

Ta klasa obsługuje uwierzytelnianie użytkownika.

`#include <authentication.h>`

Metody publiczne

- Authentication (const std::string &uri_str, const std::string &db_name, const std::string &collection_name)

  Konstruuje nowy obiekt Authentication.
- bool RegisterUser (const std::string &username, const std::string &email, const std::string &password)

  Rejestruje nowego użytkownika.
- void AuthenticateUser (const std::string &username, const std::string &password, std::promise< bool > &&promise, User &user)

  Uwierzytelnia użytkownika.

Statyczne metody publiczne

- static std::string HashPassword (const std::string &password)

  Haszuje hasło.

4.2.1 Opis szczegółowy

Ta klasa obsługuje uwierzytelnianie użytkownika.

4.2.2 Dokumentacja konstruktora i destruktora

4.2.2.1 Authentication()

```
Authentication::Authentication (
            const std::string & uri_str,
            const std::string & db_name,
            const std::string & collection_name )
```

Konstruuje nowy obiekt Authentication.

Parametry

uri_str	Ciąg URI.
db_name	Nazwa bazy danych.
collection_name	Nazwa kolekcji.

4.2.3 Dokumentacja funkcji składowych

4.2.3.1 AuthenticateUser()

```
void Authentication::AuthenticateUser (
            const std::string & username,
            const std::string & password,
            std::promise< bool > && promise,
            User & user )
```

Uwierzytelnia użytkownika.

Parametry

username	Nazwa użytkownika.
password	Hasło użytkownika.
promise	Obietnica wyniku uwierzytelnienia.
user	Obiekt User.

4.2.3.2 HashPassword()

```
std::string Authentication::HashPassword (
            const std::string & password )  [static]
```

Haszuje hasło.

Parametry

password	Hasło do zahashowania.

Zwraca

Zahashowane hasło.

4.2.3.3 RegisterUser()

```
bool Authentication::RegisterUser (
            const std::string & username,
            const std::string & email,
            const std::string & password )
```

Rejestruje nowego użytkownika.

Parametry

username	Nazwa użytkownika.
email	Email użytkownika.
password	Hasło użytkownika.

Zwraca

Wartość logiczna wskazująca, czy rejestracja była udana.

Dokumentacja dla tej klasy została wygenerowana z plików:

- authentication/authentication.h
- authentication/authentication.cpp

4.3 Dokumentacja klasy EnvParser

Ta klasa służy do analizy zmiennych środowiskowych.

```
#include <env.h>
```

Metody publiczne

- **EnvParser** ()

  Konstruuje nowy obiekt EnvParser.
- void **ParseEnvFile** ()

  Analizuje plik środowiskowy.
- std::string GetValue (const std::string &key) const

  Pobiera wartość określonej zmiennej środowiskowej.

4.3.1 Opis szczegółowy

Ta klasa służy do analizy zmiennych środowiskowych.

4.3.2 Dokumentacja funkcji składowych

4.3.2.1 GetValue()

```
std::string EnvParser::GetValue (
            const std::string & key ) const
```

Pobiera wartość określonej zmiennej środowiskowej.

Parametry

key	Klucz zmiennej środowiskowej.

Zwraca

Wartość zmiennej środowiskowej.

Dokumentacja dla tej klasy została wygenerowana z plików:

- env/env.h
- env/env.cpp

4.4 Dokumentacja klasy FlightConnection

Ta klasa obsługuje połączenia lotnicze.

`#include <flight_connection.h>`

Metody publiczne

- FlightConnection (const std::string &uri_str, const std::string &db_name, const std::string &collection_name)

  Konstruuje nowy obiekt FlightConnection.
- FlightConnection (std::string flight_id, std::string departure_city, std::string destination_city, std::string departure_time, std::string arrival_time, int available_seats, double price)

  Konstruuje nowy obiekt FlightConnection.
- std::string GetDepartureCity () const

  Pobiera miasto odlotu.
- std::string GetDestinationCity () const

  Pobiera miasto docelowe.
- std::string GetDepartureTime () const

  Pobiera czas odlotu.
- std::string GetArrivalTime () const

  Pobiera czas przyjazdu.
- std::string GetIdentifier () const

  Pobiera identyfikator lotu.
- int GetAvailableSeats () const

  Pobiera liczbę dostępnych miejsc.
- double GetPrice () const

  Pobiera cenę lotu.
- std::vector< FlightConnection > FindAllConnections ()

  Znajduje wszystkie połączenia lotnicze.
- FlightConnection FindConnection (const std::string &departure_city, const std::string &destination_city)

  Znajduje połączenie lotnicze według miasta odlotu i docelowego.
- std::vector< FlightConnection > FindConnectionByPrice (double &min_price, double &max_price)

  Znajduje połączenia lotnicze w zakresie cenowym.
- FlightConnection FindConnectionById (const std::string &id)

  Znajduje połączenie lotnicze według ID.
- std::vector< FlightConnection > FindConnectionsByDeparture (const std::string &departure_city)

  Znajduje połączenia lotnicze według miasta odlotu.
- std::vector< FlightConnection > FindConnectionsByDestination (const std::string &destination_city)

  Znajduje połączenia lotnicze według miasta docelowego.
- std::vector< int > GetSeatsTaken (const std::string &flight_identifier)

  Pobiera zajęte miejsca dla określonego lotu.
- void UpdateSeatsTaken (const std::string &flight_identifier, const std::vector< int > &seats_taken)

  Aktualizuje zajęte miejsca dla określonego lotu.

4.4.1 Opis szczegółowy

Ta klasa obsługuje połączenia lotnicze.

4.4.2 Dokumentacja konstruktora i destruktora

4.4.2.1 FlightConnection() [1/2]

```
FlightConnection::FlightConnection (
            const std::string & uri_str,
            const std::string & db_name,
            const std::string & collection_name )
```

Konstruuje nowy obiekt FlightConnection.

Parametry

uri_str	Ciąg URI.
db_name	Nazwa bazy danych.
collection_name	Nazwa kolekcji.

4.4.2.2 FlightConnection() [2/2]

```
FlightConnection::FlightConnection (
            std::string flight_id,
            std::string departure_city,
            std::string destination_city,
            std::string departure_time,
            std::string arrival_time,
            int available_seats,
            double price )
```

Konstruuje nowy obiekt FlightConnection.

Parametry

flight_id	ID lotu.
departure_city	Miasto odlotu.
destination_city	Miasto docelowe.
departure_time	Czas odlotu.
arrival_time	Czas przyjazdu.
available_seats	Liczba dostępnych miejsc.
price	Cena lotu.

4.4.3 Dokumentacja funkcji składowych

4.4.3.1 FindAllConnections()

```
std::vector< FlightConnection > FlightConnection::FindAllConnections ( )
```

Znajduje wszystkie połączenia lotnicze.

Zwraca

Wektor wszystkich połączeń lotniczych.

4.4.3.2 FindConnection()

```
FlightConnection FlightConnection::FindConnection (
            const std::string & departure_city,
            const std::string & destination_city )
```

Znajduje połączenie lotnicze według miasta odlotu i docelowego.

Parametry

departure_city	Miasto odlotu.
destination_city	Miasto docelowe.

Zwraca

Znalezione połączenie lotnicze.

4.4.3.3 FindConnectionById()

```
FlightConnection FlightConnection::FindConnectionById (
            const std::string & id )
```

Znajduje połączenie lotnicze według ID.

Parametry

id	ID lotu.

Zwraca

Znalezione połączenie lotnicze.

4.4.3.4 FindConnectionByPrice()

```
std::vector< FlightConnection > FlightConnection::FindConnectionByPrice (
```

```
            double & min_price,
            double & max_price )
```

Znajduje połączenia lotnicze w zakresie cenowym.

Parametry

min_price	Minimalna cena.
max_price	Maksymalna cena.

Zwraca

Wektor znalezionych połączeń lotniczych.

4.4.3.5 FindConnectionsByDeparture()

```
std::vector< FlightConnection > FlightConnection::FindConnectionsByDeparture (
            const std::string & departure_city )
```

Znajduje połączenia lotnicze według miasta odlotu.

Parametry

departure_city	Miasto odlotu.

Zwraca

Wektor znalezionych połączeń lotniczych.

4.4.3.6 FindConnectionsByDestination()

```
std::vector< FlightConnection > FlightConnection::FindConnectionsByDestination (
            const std::string & destination_city )
```

Znajduje połączenia lotnicze według miasta docelowego.

Parametry

destination_city	Miasto docelowe.

Zwraca

Wektor znalezionych połączeń lotniczych.

4.4.3.7 GetArrivalTime()

```
std::string FlightConnection::GetArrivalTime ( ) const
```

Pobiera czas przyjazdu.

Zwraca

Czas przyjazdu.

4.4.3.8 GetAvailableSeats()

```
int FlightConnection::GetAvailableSeats ( ) const
```

Pobiera liczbę dostępnych miejsc.

Zwraca

Liczba dostępnych miejsc.

4.4.3.9 GetDepartureCity()

```
std::string FlightConnection::GetDepartureCity ( ) const
```

Pobiera miasto odlotu.

Zwraca

Miasto odlotu.

4.4.3.10 GetDepartureTime()

```
std::string FlightConnection::GetDepartureTime ( ) const
```

Pobiera czas odlotu.

Zwraca

Czas odlotu.

4.4.3.11 GetDestinationCity()

```
std::string FlightConnection::GetDestinationCity ( ) const
```

Pobiera miasto docelowe.

Zwraca

Miasto docelowe.

4.4.3.12 GetIdentifier()

```
std::string FlightConnection::GetIdentifier ( ) const
```

Pobiera identyfikator lotu.

Zwraca

Identyfikator lotu.

4.4.3.13 GetPrice()

```
double FlightConnection::GetPrice ( ) const
```

Pobiera cenę lotu.

Zwraca

Cena lotu.

4.4.3.14 GetSeatsTaken()

```
std::vector< int > FlightConnection::GetSeatsTaken (
            const std::string & flight_identifier )
```

Pobiera zajęte miejsca dla określonego lotu.

Parametry

flight_identifier	Identyfikator lotu.

Zwraca

Wektor zajętych miejsc.

4.4.3.15 UpdateSeatsTaken()

```
void FlightConnection::UpdateSeatsTaken (
            const std::string & flight_identifier,
            const std::vector< int > & seats_taken )
```

Aktualizuje zajęte miejsca dla określonego lotu.

Parametry

flight_identifier	Identyfikator lotu.
seats_taken	Wektor zajętych miejsc.

Dokumentacja dla tej klasy została wygenerowana z plików:

- flights/flight_connection.h
- flights/flight_connection.cpp

4.5 Dokumentacja struktury FlightInfo

Zawiera informacje o locie.

`#include <user_tickets_prints.h>`

Atrybuty publiczne

- int **flight_number**

  Numer lotu.
- std::string **flight_id**

  ID lotu.
- std::string **departure**

  Miejsce odlotu.
- std::string **destination**

  Miejsce docelowe.
- std::string **departure_time**

  Czas odlotu.
- double **price**

  Cena lotu.
- std::vector< int > **seats**

  Miejsca w locie.
- bool **checkin**

  Czy użytkownik odprawił się.
- bool **luggage_checkin**

  Czy użytkownik odprawił bagaż.

4.5.1 Opis szczegółowy

Zawiera informacje o locie.

Dokumentacja dla tej struktury została wygenerowana z pliku:

- user/user_functions/user_tickets/user_tickets_prints.h

4.6 Dokumentacja klasy Item

Ta klasa reprezentuje przedmiot w systemie lotniska.

`#include <item.h>`

Metody publiczne

- Item (const std::string &item_name, const std::string &description, const std::vector< std::string > &hints, bool forbidden, bool registered_luggage, bool hand_luggage, bool pilot_allowance, double max_count, double weight, std::string &profession, std::string &category)

  Konstruuje nowy obiekt Item.
- Item (const std::string &item_name, const std::string &description, const std::vector< std::string > &hints, bool forbidden, bool registered_luggage, bool hand_luggage, bool pilot_allowance, double max_count, double weight, std::string &category)

  Konstruuje nowy obiekt Item.
- const std::string & GetItemName () const

  Pobiera nazwę przedmiotu.
- const std::string & GetDescription () const

  Pobiera opis przedmiotu.
- const std::string & GetProfession () const

  Pobiera zawód związany z przedmiotem.
- const std::vector< std::string > & GetHints () const

  Pobiera wskazówki dotyczące przedmiotu.
- bool IsForbidden () const

  Sprawdza, czy przedmiot jest zabroniony.
- bool IsRegisteredLuggage () const

  Sprawdza, czy przedmiot może być transportowany jako bagaż rejestrowany.
- bool IsHandLuggage () const

  Sprawdza, czy przedmiot może być transportowany jako bagaż podręczny.
- bool IsPilotAllowance () const

  Sprawdza, czy przedmiot wymaga zgody pilota do transportu.
- double GetMaxCount () const

  Pobiera maksymalną liczbę przedmiotów.
- double GetWeight () const

  Pobiera wagę przedmiotu.
- std::string GetCategory () const

  Pobiera kategorię przedmiotu.

4.6.1 Opis szczegółowy

Ta klasa reprezentuje przedmiot w systemie lotniska.

4.6.2 Dokumentacja konstruktora i destruktora

4.6.2.1 Item() [1/2]

```
Item::Item (
            const std::string & item_name,
            const std::string & description,
            const std::vector< std::string > & hints,
            bool forbidden,
            bool registered_luggage,
            bool hand_luggage,
            bool pilot_allowance,
            double max_count,
            double weight,
            std::string & profession,
            std::string & category )
```

Konstruuje nowy obiekt Item.

Parametry

item_name	Nazwa przedmiotu.
description	Opis przedmiotu.
hints	Wskazówki dotyczące przedmiotu.
forbidden	Wskazuje, czy przedmiot jest zabroniony.
registered_luggage	Wskazuje, czy przedmiot może być transportowany jako bagaż rejestrowany.
hand_luggage	Wskazuje, czy przedmiot może być transportowany jako bagaż podręczny.
pilot_allowance	Wskazuje, czy przedmiot wymaga zgody pilota do transportu.
max_count	Maksymalna liczba przedmiotów.
weight	Waga przedmiotu.
profession	Zawód związany z przedmiotem.
category	Kategoria przedmiotu.

4.6.2.2 Item() [2/2]

```
Item::Item (
            const std::string & item_name,
            const std::string & description,
            const std::vector< std::string > & hints,
            bool forbidden,
            bool registered_luggage,
            bool hand_luggage,
            bool pilot_allowance,
            double max_count,
            double weight,
            std::string & category )
```

Konstruuje nowy obiekt Item.

Parametry

item_name	Nazwa przedmiotu.
description	Opis przedmiotu.
hints	Wskazówki dotyczące przedmiotu.
forbidden	Wskazuje, czy przedmiot jest zabroniony.
registered_luggage	Wskazuje, czy przedmiot może być transportowany jako bagaż rejestrowany.
hand_luggage	Wskazuje, czy przedmiot może być transportowany jako bagaż podręczny.
pilot_allowance	Wskazuje, czy przedmiot wymaga zgody pilota do transportu.
max_count	Maksymalna liczba przedmiotów.
weight	Waga przedmiotu.
category	Kategoria przedmiotu.

4.6.3 Dokumentacja funkcji składowych

4.6.3.1 GetCategory()

```
std::string Item::GetCategory ( ) const
```

Pobiera kategorię przedmiotu.

Zwraca

Kategoria przedmiotu.

4.6.3.2 GetDescription()

```
const std::string & Item::GetDescription ( ) const
```

Pobiera opis przedmiotu.

Zwraca

Opis przedmiotu.

4.6.3.3 GetHints()

```
const std::vector< std::string > & Item::GetHints ( ) const
```

Pobiera wskazówki dotyczące przedmiotu.

Zwraca

Wskazówki dotyczące przedmiotu.

4.6.3.4 GetItemName()

```
const std::string & Item::GetItemName ( ) const
```

Pobiera nazwę przedmiotu.

Zwraca

Nazwa przedmiotu.

4.6.3.5 GetMaxCount()

```
double Item::GetMaxCount ( ) const
```

Pobiera maksymalną liczbę przedmiotów.

Zwraca

Maksymalna liczba przedmiotów.

4.6.3.6 GetProfession()

```
const std::string & Item::GetProfession ( ) const
```

Pobiera zawód związany z przedmiotem.

Zwraca

Zawód związany z przedmiotem.

4.6.3.7 GetWeight()

```
double Item::GetWeight ( ) const
```

Pobiera wagę przedmiotu.

Zwraca

Waga przedmiotu.

4.6.3.8 IsForbidden()

```
bool Item::IsForbidden ( ) const
```

Sprawdza, czy przedmiot jest zabroniony.

Zwraca

True, jeśli przedmiot jest zabroniony, false w przeciwnym razie.

4.6.3.9 IsHandLuggage()

```
bool Item::IsHandLuggage ( ) const
```

Sprawdza, czy przedmiot może być transportowany jako bagaż podręczny.

Zwraca

True, jeśli przedmiot może być transportowany jako bagaż podręczny, false w przeciwnym razie.

4.6.3.10 IsPilotAllowance()

```
bool Item::IsPilotAllowance ( ) const
```

Sprawdza, czy przedmiot wymaga zgody pilota do transportu.

Zwraca

True, jeśli przedmiot wymaga zgody pilota do transportu, false w przeciwnym razie.

4.6.3.11 IsRegisteredLuggage()

```
bool Item::IsRegisteredLuggage ( ) const
```

Sprawdza, czy przedmiot może być transportowany jako bagaż rejestrowany.

Zwraca

True, jeśli przedmiot może być transportowany jako bagaż rejestrowany, false w przeciwnym razie.

Dokumentacja dla tej klasy została wygenerowana z plików:

- luggage/item/item.h
- luggage/item/item.cpp

4.7 Dokumentacja klasy Luggage

Ta klasa reprezentuje bagaż w systemie lotniska.

```
#include <luggage.h>
```

Metody publiczne

- Luggage (const std::vector< Item > &items, double total_weight)
  Konstruuje nowy obiekt Luggage.
- double ProcessItemsAndGetWeight ()
  Przetwarza przedmioty i pobiera wagę.
- std::tuple< bool, std::string > ConfirmItems (User &user)
  Potwierdza przedmioty w bagażu.
- double CalculateOverweightFee (double weight) const
  Oblicza opłatę za nadbagaż.

Atrybuty publiczne

- const double **overweight_fee_per_kg_** = 2.0
  Opłata za nadbagaż na kg.
- const double **euro_to_pln_** = 4.32
  Kurs wymiany euro na pln.
- const double **max_allowed_weight_** = 32.0
  Maksymalna dozwolona waga.
- double **max_weight_** = 20.0
  Maksymalna waga.

4.7.1 Opis szczegółowy

Ta klasa reprezentuje bagaż w systemie lotniska.

4.7.2 Dokumentacja konstruktora i destruktora

4.7.2.1 Luggage()

```
Luggage::Luggage (
            const std::vector< Item > & items,
            double total_weight )  [inline]
```

Konstruuje nowy obiekt Luggage.

Parametry

items	Przedmioty w bagażu.
total_weight	Całkowita waga bagażu.

4.7.3 Dokumentacja funkcji składowych

4.7.3.1 CalculateOverweightFee()

```
double Luggage::CalculateOverweightFee (
            double weight ) const
```

Oblicza opłatę za nadbagaż.

Parametry

weight	Waga bagażu.

Zwraca

Opłata za nadbagaż.

4.7.3.2 ConfirmItems()

```
std::tuple< bool, std::string > Luggage::ConfirmItems (
            User & user )
```

Potwierdza przedmioty w bagażu.

Parametry

user	Użytkownik potwierdzający przedmioty.

Zwraca

Krotka zawierająca boolean wskazujący, czy potwierdzenie było udane, i wiadomość typu string.

4.7.3.3 ProcessItemsAndGetWeight()

```
double Luggage::ProcessItemsAndGetWeight ( )
```

Przetwarza przedmioty i pobiera wagę.

Zwraca

Waga przedmiotów.

Dokumentacja dla tej klasy została wygenerowana z plików:

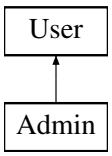
- luggage/luggage.h
- luggage/luggage.cpp

4.8 Dokumentacja klasy User

Reprezentuje użytkownika w systemie.

`#include <user.h>`

Diagram dziedziczenia dla User



Metody publiczne

- User (mongocxx::client &client)

  Konstruuje nowy obiekt User.
- **User** (std::string username, std::string email, double discount, std::string discount_type, std::string premium_card, std::string payment_method, mongocxx::client &client, std::string profession, std::string registration_date, double money_spent, double money_saved, int ticket_bought, std::vector< bsoncxx::document::value > user_flights, bool is_admin)

  Konstruuje nowy obiekt User z określonymi parametrami.
- void **Reset** ()

  Resetuje użytkownika.
- mongocxx::collection & **GetCollection** ()

  Zwraca kolekcję.
- mongocxx::collection **GetSpecificCollection** (const std::string &collection_name)

  Zwraca określoną kolekcję.
- std::string **GetPassword** ()

  Zwraca hasło.
- void **SetPassword** (const std::string &password)

  Ustawia hasło.
- void **SetPremiumCard** (User &user, const std::string &card)

  Ustawia kartę premium.
- void **SetBlik** (const std::string &payment_method)

  Ustawia metodę płatności Blik.
- void **SetVisa** (const std::string &card_number, const std::string &card_cvv)

  Ustawia metodę płatności Visa.
- void **ChangeUsername** (const std::string &username)

  Zmienia nazwę użytkownika.

- void **ChangeEmail** (const std::string &email)

  Zmienia email.
- void **ChangePassword** (const std::string &password)

  Zmienia hasło.
- void **SetDiscount** (double discount, const std::string &discount_type)

  Ustawia zniżkę.
- double **GetDiscount** () const

  Zwraca zniżkę.
- std::string **RecognizeDiscount** () const

  Rozpoznaje zniżkę.
- void **AddTicketToUser** (const std::vector< int > &seats, const FlightConnection &flight_connection)

  Dodaje bilet do użytkownika.
- void **UpdateMoneySaved** (double normal_price, double discount_price)

  Aktualizuje zaoszczędzone pieniądze.
- Admin * **LoginAsAdmin** ()

  Loguje jako admin.
- bool **CheckIfAdmin** () const

  Sprawdza, czy użytkownik jest administratorem.
- void **SetIsAdmin** (bool is_administrator)

  Ustawia, czy użytkownik jest administratorem.
- void **LuggageCheckin** (int flight_number)

  Odprawia bagaż.
- mongocxx::cursor **FindUserInDatabase** ()

  Znajduje użytkownika w bazie danych.
- template<typename T >
  void UpdateUserInDatabase (const std::string &value_in_database, const T &value_to_set)

  Aktualizuje użytkownika w bazie danych.

Atrybuty publiczne

- std::string **username_**

  Nazwa użytkownika.
- std::string **profession_**

  Zawód użytkownika.
- std::string **email_**

  Email użytkownika.
- std::string **discount_type_**

  Typ zniżki użytkownika.
- double **discount_**

  Wartość zniżki użytkownika.
- std::string **premium_card_**

  Karta premium użytkownika.
- std::string **payment_method_**

  Metoda płatności użytkownika.
- std::string **registration_date_**

  Data rejestracji użytkownika.
- double **money_spent_**

  Całkowita kwota wydana przez użytkownika.
- double **money_saved_**

  Całkowita kwota zaoszczędzona przez użytkownika.

- int **ticket_bought_**

  Całkowita liczba biletów kupionych przez użytkownika.
- std::vector< bsoncxx::document::value > **user_flights_**

  Loty użytkownika.
- bool **is_admin_**

  Czy użytkownik jest administratorem.

Atrybuty chronione

- mongocxx::client & **_client**

  Klient MongoDB.

4.8.1 Opis szczegółowy

Reprezentuje użytkownika w systemie.

4.8.2 Dokumentacja konstruktora i destruktora

4.8.2.1 User()

```
User::User (
            mongocxx::client & client )  [inline], [explicit]
```

Konstruuje nowy obiekt User.

Parametry

client	Klient MongoDB.

4.8.3 Dokumentacja funkcji składowych

4.8.3.1 UpdateUserInDatabase()

```
template<typename T >
void User::UpdateUserInDatabase (
            const std::string & value_in_database,
            const T & value_to_set )  [inline]
```

Aktualizuje użytkownika w bazie danych.

Parametry Szablonu

T	Typ wartości do ustawienia.

Parametry

value_in_database	Wartość w bazie danych.
value_to_set	Wartość do ustawienia.

Dokumentacja dla tej klasy została wygenerowana z plików:

- user/user.h
- user/user.cpp

Rozdział 5

Dokumentacja plików

5.1 Dokumentacja pliku admin/admin.h

Ten plik zawiera deklarację klasy Admin.

```
#include "../user/user.h"
```

Komponenty

- class Admin

  Ta klasa reprezentuje użytkownika admina.

5.1.1 Opis szczegółowy

Ten plik zawiera deklarację klasy Admin.

5.2 admin.h

Idź do dokumentacji tego pliku.

```
00001
00006 #ifndef AIRPORT_ADMIN_H
00007 #define AIRPORT_ADMIN_H
00008
00009 #include "../user/user.h"
00010
00016 class Admin : public User {
00017  public:
00036   Admin(const std::string &username,
00037         const std::string &email,
00038         double discount,
00039         const std::string &discount_type,
00040         const std::string &premium_card,
00041         const std::string &payment_method,
00042         mongocxx::client &client,
00043         const std::string &profession,
00044         const std::string &registration_date,
00045         double money_spent,
00046         double money_saved,
00047         int ticket_bought,
00048         const std::vector<bsoncxx::document::value> &user_flights,
00049         bool is_admin,
```

```
00050         std::string hashed_admin_password);
00051
00056   Admin(const User &user)
00057       : User(user), hashed_admin_password_("") {}
00058
00059   std::string hashed_admin_password_;
00060
00065   void AddFlight(User &user);
00066
00071   void AddVerificationQuestion(User &user);
00072
00077   void ManageUsers(User &user);
00078
00083   void AddLuggageItem(User &user);
00084 };
00085
00086 #endif  // AIRPORT_ADMIN_H
```

5.3 Dokumentacja pliku admin/admin_functions/admin_functions.h

Ten plik zawiera deklaracje funkcji używanych w panelu administratora.

```
#include "../../user/user.h"
```

Funkcje

- void HandleAdminDashboard (Admin &admin, User &user)

  Obsługuje panel administratora.
- std::string ProcessAddingFlight ()

  Przetwarza dodanie lotu.
- std::string CaptureInputWithValidation (const std::string &title, const std::string &message, const std::function< bool(const std::string &)> &validator)

  Przechwytuje dane wejściowe użytkownika z walidacją.
- std::string CaptureLineWithValidation (const std::string &title, const std::string &message, const std::function< bool(const std::string &)> &validator)

  Przechwytuje linię danych wejściowych z walidacją.
- std::optional< bool > CaptureBoolWithValidation (const std::string &title, const std::string &message)

  Przechwytuje wartość logiczną z walidacją.

5.3.1 Opis szczegółowy

Ten plik zawiera deklaracje funkcji używanych w panelu administratora.

5.3.2 Dokumentacja funkcji

5.3.2.1 CaptureBoolWithValidation()

```
std::optional< bool > CaptureBoolWithValidation (
            const std::string & title,
            const std::string & message )
```

Przechwytuje wartość logiczną z walidacją.

Parametry

title	Tytuł pola wejściowego.
message	Wiadomość do wyświetlenia użytkownikowi.

Zwraca

Opcjonalna wartość logiczna. Jeśli dane wejściowe są prawidłowe, wartością jest przechwycone dane wejściowe. Jeśli dane wejściowe są nieprawidłowe, wartością jest std::nullopt.

5.3.2.2 CaptureInputWithValidation()

```
std::string CaptureInputWithValidation (
            const std::string & title,
            const std::string & message,
            const std::function< bool(const std::string &)> & validator )
```

Przechwytuje dane wejściowe użytkownika z walidacją.

Parametry

title	Tytuł pola wejściowego.
message	Wiadomość do wyświetlenia użytkownikowi.
validator	Funkcja do walidacji danych wejściowych.

Zwraca

Przechwycone dane wejściowe.

5.3.2.3 CaptureLineWithValidation()

```
std::string CaptureLineWithValidation (
            const std::string & title,
            const std::string & message,
            const std::function< bool(const std::string &)> & validator )
```

Przechwytuje linię danych wejściowych z walidacją.

Parametry

title	Tytuł pola wejściowego.
message	Wiadomość do wyświetlenia użytkownikowi.
validator	Funkcja do walidacji danych wejściowych.

Zwraca

Przechwycone dane wejściowe.

5.3.2.4 HandleAdminDashboard()

```
void HandleAdminDashboard (
            Admin & admin,
            User & user )
```

Obsługuje panel administratora.

Parametry

admin	Obiekt administratora.
user	Obiekt użytkownika.

5.3.2.5 ProcessAddingFlight()

```
std::string ProcessAddingFlight ( )
```

Przetwarza dodanie lotu.

Zwraca

Ciąg znaków reprezentujący wynik operacji.

5.4 admin_functions.h

Idź do dokumentacji tego pliku.

```
00001
00006 #ifndef AIRPORT_ADMIN_ADMIN_FUNCTIONS_ADMIN_FUNCTIONS_H_
00007 #define AIRPORT_ADMIN_ADMIN_FUNCTIONS_ADMIN_FUNCTIONS_H_
00008
00009 #include "../../user/user.h"
00010
00016 void HandleAdminDashboard(Admin &admin, User &user);
00017
00022 std::string ProcessAddingFlight();
00023
00031 std::string CaptureInputWithValidation(
00032     const std::string &title,
00033     const std::string &message,
00034     const std::function<bool(const std::string &)> &validator);
00035
00043 std::string CaptureLineWithValidation(
00044     const std::string &title,
00045     const std::string &message,
00046     const std::function<bool(const std::string &)> &validator);
00047
00054 std::optional<bool> CaptureBoolWithValidation(const std::string &title, const std::string &message);
00055
00056 #endif //AIRPORT_ADMIN_ADMIN_FUNCTIONS_ADMIN_FUNCTIONS_H_
```

5.5 Dokumentacja pliku admin/admin_functions/validators.h

Ten plik zawiera deklaracje funkcji walidacyjnych używanych w panelu administratora.

```
#include <string>
```

Funkcje

- bool ValidateFlightId (const std::string &flight_id)

  Waliduje ID lotu.
- bool ValidateCity (const std::string &city)

  Waliduje nazwę miasta.
- bool ValidateDate (const std::string &date)

  Waliduje datę.
- bool ValidateTime (const std::string &time)

  Waliduje czas.
- bool ValidatePrice (const std::string &price)

  Waliduje cenę.
- bool ValidateNonEmpty (const std::string &input)

  Waliduje, czy dane wejściowe nie są puste.
- bool ValidateSolution (const std::string &solution)

  Waliduje rozwiązanie.

5.5.1 Opis szczegółowy

Ten plik zawiera deklaracje funkcji walidacyjnych używanych w panelu administratora.

5.5.2 Dokumentacja funkcji

5.5.2.1 ValidateCity()

```
bool ValidateCity (
            const std::string & city )
```

Waliduje nazwę miasta.

Parametry

city	Nazwa miasta do zweryfikowania.

Zwraca

Prawda, jeśli nazwa miasta jest prawidłowa, w przeciwnym razie fałsz.

5.5.2.2 ValidateDate()

```
bool ValidateDate (
            const std::string & date )
```

Waliduje datę.

Parametry

date	Data do zweryfikowania.

Zwraca

Prawda, jeśli data jest prawidłowa, w przeciwnym razie fałsz.

5.5.2.3 ValidateFlightId()

```
bool ValidateFlightId (
            const std::string & flight_id )
```

Waliduje ID lotu.

Parametry

flight_id	ID lotu do zweryfikowania.

Zwraca

Prawda, jeśli ID lotu jest prawidłowe, w przeciwnym razie fałsz.

5.5.2.4 ValidateNonEmpty()

```
bool ValidateNonEmpty (
            const std::string & input )
```

Waliduje, czy dane wejściowe nie są puste.

Parametry

input	Dane wejściowe do zweryfikowania.

Zwraca

Prawda, jeśli dane wejściowe nie są puste, w przeciwnym razie fałsz.

5.5.2.5 ValidatePrice()

```
bool ValidatePrice (
            const std::string & price )
```

Waliduje cenę.

Parametry

price	Cena do zweryfikowania.

Zwraca

Prawda, jeśli cena jest prawidłowa, w przeciwnym razie fałsz.

5.5.2.6 ValidateSolution()

```
bool ValidateSolution (
            const std::string & solution )
```

Waliduje rozwiązanie.

Parametry

solution	Rozwiązanie do zweryfikowania.

Zwraca

Prawda, jeśli rozwiązanie jest prawidłowe, w przeciwnym razie fałsz.

5.5.2.7 ValidateTime()

```
bool ValidateTime (
            const std::string & time )
```

Waliduje czas.

Parametry

time	Czas do zweryfikowania.

Zwraca

Prawda, jeśli czas jest prawidłowy, w przeciwnym razie fałsz.

5.6 validators.h

Idź do dokumentacji tego pliku.

```
00001
00006 #ifndef AIRPORT_ADMIN_ADMIN_FUNCTIONS_VALIDATORS_H_
00007 #define AIRPORT_ADMIN_ADMIN_FUNCTIONS_VALIDATORS_H_
00008
00009 #include <string>
00010
00016 bool ValidateFlightId(const std::string &flight_id);
```

```
00017
00023 bool ValidateCity(const std::string &city);
00024
00030 bool ValidateDate(const std::string &date);
00031
00037 bool ValidateTime(const std::string &time);
00038
00044 bool ValidatePrice(const std::string &price);
00045
00051 bool ValidateNonEmpty(const std::string &input);
00052
00058 bool ValidateSolution(const std::string &solution);
00059
00060 #endif //AIRPORT_ADMIN_ADMIN_FUNCTIONS_VALIDATORS_H_
```

5.7 Dokumentacja pliku admin/admin_prints/admin_prints.h

Ten plik zawiera deklaracje funkcji używanych do wyświetlania informacji związanych z administratorem.

Funkcje

- void **DisplayAdminMenu** ()

  Wyświetla menu administratora.
- void **DisplayAddingFlightInfo** ()

  Wyświetla informacje związane z dodawaniem lotu.
- std::string DisplayAdminMessageAndCaptureInput (const std::string &title_message, const std::string &text↩ _message)

  Wyświetla wiadomość dla administratora i przechwytuje jego dane wejściowe.
- std::string DisplayAdminMessageAndCaptureLine (const std::string &title_message, const std::string &text↩ _message)

  Wyświetla wiadomość dla administratora i przechwytuje linię jego danych wejściowych.
- void **DisplayManageUsersMenu** ()

  Wyświetla menu zarządzania użytkownikami.

5.7.1 Opis szczegółowy

Ten plik zawiera deklaracje funkcji używanych do wyświetlania informacji związanych z administratorem.

5.7.2 Dokumentacja funkcji

5.7.2.1 DisplayAdminMessageAndCaptureInput()

```
std::string DisplayAdminMessageAndCaptureInput (
            const std::string & title_message,
            const std::string & text_message )
```

Wyświetla wiadomość dla administratora i przechwytuje jego dane wejściowe.

Parametry

title_message	Tytuł wiadomości.
text_message	Tekst wiadomości.

Zwraca

Dane wejściowe przechwycone od administratora.

5.7.2.2 DisplayAdminMessageAndCaptureLine()

```
std::string DisplayAdminMessageAndCaptureLine (
             const std::string & title_message,
             const std::string & text_message )
```

Wyświetla wiadomość dla administratora i przechwytuje linię jego danych wejściowych.

Parametry

title_message	Tytuł wiadomości.
text_message	Tekst wiadomości.

Zwraca

Linia danych wejściowych przechwycona od administratora.

5.8 admin_prints.h

Idź do dokumentacji tego pliku.

```
00001
00006 #ifndef AIRPORT_ADMIN_ADMIN_PRINTS_ADMIN_PRINTS_H_
00007 #define AIRPORT_ADMIN_ADMIN_PRINTS_ADMIN_PRINTS_H_
00008
00012 void DisplayAdminMenu();
00013
00017 void DisplayAddingFlightInfo();
00018
00025 std::string DisplayAdminMessageAndCaptureInput(const std::string &title_message, const std::string
      &text_message);
00026
00033 std::string DisplayAdminMessageAndCaptureLine(const std::string &title_message, const std::string
      &text_message);
00034
00038 void DisplayManageUsersMenu();
00039
00040 #endif //AIRPORT_ADMIN_ADMIN_PRINTS_ADMIN_PRINTS_H_
```

5.9 Dokumentacja pliku authentication/auth_functions/user_authentication.h

Ten plik zawiera deklaracje funkcji używanych do uwierzytelniania użytkownika.

```
#include <string>
#include <tuple>
#include "../../user/user.h"
#include "../authentication.h"
```

Funkcje

- std::tuple< std::string, std::string, std::string, bool > RegisterUser ()

  Rejestruje nowego użytkownika.
- std::tuple< std::string, std::string, bool > Login ()

  Loguje użytkownika.
- void HandleRegistration (Authentication &auth)

  Obsługuje proces rejestracji.
- bool HandleLogin (Authentication &auth, User &user)

  Obsługuje proces logowania.

5.9.1 Opis szczegółowy

Ten plik zawiera deklaracje funkcji używanych do uwierzytelniania użytkownika.

5.9.2 Dokumentacja funkcji

5.9.2.1 HandleLogin()

```
bool HandleLogin (
            Authentication & auth,
            User & user )
```

Obsługuje proces logowania.

Parametry

auth	Obiekt Authentication.
user	Obiekt User.

Zwraca

Wartość logiczna wskazująca, czy logowanie było udane.

5.9.2.2 HandleRegistration()

```
void HandleRegistration (
            Authentication & auth )
```

Obsługuje proces rejestracji.

Parametry

auth	Obiekt Authentication.

5.9.2.3 Login()

```
std::tuple< std::string, std::string, bool > Login ( )
```

Loguje użytkownika.

Zwraca

Krotka zawierająca nazwę użytkownika, hasło i wartość logiczną wskazującą, czy logowanie było udane.

5.9.2.4 RegisterUser()

```
std::tuple< std::string, std::string, std::string, bool > RegisterUser ( )
```

Rejestruje nowego użytkownika.

Zwraca

Krotka zawierająca nazwę użytkownika, hasło, email i wartość logiczną wskazującą, czy rejestracja była udana.

5.10 user_authentication.h

Idź do dokumentacji tego pliku.

```
00001
00006 #ifndef AIRPORT_AUTHPRINTHANDLER_H
00007 #define AIRPORT_AUTHPRINTHANDLER_H
00008
00009 #include <string>
00010 #include <tuple>
00011 #include "../../user/user.h"
00012 #include "../authentication.h"
00013
00018 std::tuple<std::string, std::string, std::string, bool> RegisterUser();
00019
00024 std::tuple<std::string, std::string, bool> Login();
00025
00030 void HandleRegistration(Authentication &auth);
00031
00038 bool HandleLogin(Authentication &auth, User &user);
00039
00040 #endif  // AIRPORT_AUTHPRINTHANDLER_H
```

5.11 Dokumentacja pliku authentication/authentication.h

Ten plik zawiera deklarację klasy Authentication.

```
#include <future>
#include <string>
#include "../user/user.h"
#include "mongocxx/v_noabi/mongocxx/client.hpp"
#include "mongocxx/v_noabi/mongocxx/database.hpp"
#include "mongocxx/v_noabi/mongocxx/instance.hpp"
```

Komponenty

- class Authentication

  Ta klasa obsługuje uwierzytelnianie użytkownika.

5.11.1 Opis szczegółowy

Ten plik zawiera deklarację klasy Authentication.

5.12 authentication.h

Idź do dokumentacji tego pliku.

```
00001
00006 #ifndef AUTHENTICATION_H
00007 #define AUTHENTICATION_H
00008
00009 #include <future>
00010 #include <string>
00011
00012 #include "../user/user.h"
00013 #include "mongocxx/v_noabi/mongocxx/client.hpp"
00014 #include "mongocxx/v_noabi/mongocxx/database.hpp"
00015 #include "mongocxx/v_noabi/mongocxx/instance.hpp"
00016
00021 class Authentication {
00022  private:
00023   mongocxx::client _client_;
00024   mongocxx::database _db_;
00025   mongocxx::collection _collection_;
00026
00027  public:
00034   Authentication(const std::string &uri_str, const std::string &db_name, const std::string
      &collection_name);
00035
00041   static std::string HashPassword(const std::string &password);
00042
00050   bool RegisterUser(const std::string &username, const std::string &email, const std::string
      &password);
00051
00059   void AuthenticateUser(const std::string &username,
00060                         const std::string &password,
00061                         std::promise<bool> &&promise,
00062                         User &user);
00063 };
00064
00065 #endif  // AUTHENTICATION_H
```

5.13 Dokumentacja pliku checkin/checkin_prints.h

Ten plik zawiera deklaracje funkcji używanych do operacji check-in.

```
#include "../user/user.h"
```

Funkcje

- void PrintCheckinScreen (User &user)

  Wyświetla ekran check-in dla użytkownika.

5.13.1 Opis szczegółowy

Ten plik zawiera deklaracje funkcji używanych do operacji check-in.

5.13.2 Dokumentacja funkcji

5.13.2.1 PrintCheckinScreen()

```
void PrintCheckinScreen (
            User & user )
```

Wyświetla ekran check-in dla użytkownika.

Parametry

user	Użytkownik, dla którego wyświetlany jest ekran check-in.

5.14 checkin_prints.h

Idź do dokumentacji tego pliku.

```
00001
00006 #ifndef AIRPORT_CHECKIN_FUNCTIONS_H
00007 #define AIRPORT_CHECKIN_FUNCTIONS_H
00008
00009 #include "../user/user.h"
00010
00015 void PrintCheckinScreen(User &user);
00016
00017 #endif  // AIRPORT_CHECKIN_FUNCTIONS_H
```

5.15 Dokumentacja pliku env/env.h

Ten plik zawiera deklarację klasy EnvParser.

```
#include <string>
#include <unordered_map>
```

Komponenty

- class EnvParser

  Ta klasa służy do analizy zmiennych środowiskowych.

5.15.1 Opis szczegółowy

Ten plik zawiera deklarację klasy EnvParser.

5.16 env.h

Idź do dokumentacji tego pliku.

```
00001
00006 #ifndef ENVPARSER_H
00007 #define ENVPARSER_H
00008
00009 #include <string>
00010 #include <unordered_map>
00011
00016 class EnvParser {
00017  private:
00018   std::unordered_map<std::string, std::string> _env_map_;
00019
00020  public:
00024   EnvParser();
00025
00029   void ParseEnvFile();
00030
00036   [[nodiscard]] std::string GetValue(const std::string &key) const;
00037 };
00038
00039 #endif  // ENVPARSER_H
```

5.17 Dokumentacja pliku flights/flight_connection.h

Ten plik zawiera deklarację klasy FlightConnection.

```
#include <string>
#include "mongocxx/v_noabi/mongocxx/client.hpp"
#include "mongocxx/v_noabi/mongocxx/collection.hpp"
#include "mongocxx/v_noabi/mongocxx/database.hpp"
```

Komponenty

- class FlightConnection

  Ta klasa obsługuje połączenia lotnicze.

5.17.1 Opis szczegółowy

Ten plik zawiera deklarację klasy FlightConnection.

5.18 flight_connection.h

Idź do dokumentacji tego pliku.

```
00001
00006 #pragma once
00007
00008 #include <string>
00009
00010 #include "mongocxx/v_noabi/mongocxx/client.hpp"
00011 #include "mongocxx/v_noabi/mongocxx/collection.hpp"
00012 #include "mongocxx/v_noabi/mongocxx/database.hpp"
00013
00018 class FlightConnection {
00019  private:
00020   std::string _flight_id_;
00021   std::string _departureCity_;
00022   std::string _destinationTime_;
```

```
00023   std::string _arrivalTime_;
00024   std::string _departureTime_;
00025   std::string _destinationCity_;
00026   int _availableSeats_{};
00027   double _price_{};
00028
00029   mongocxx::client _client_;
00030   mongocxx::database _db_;
00031   mongocxx::collection _collection_;
00032
00033  public:
00040   FlightConnection(const std::string &uri_str, const std::string &db_name, const std::string
      &collection_name);
00041
00052   FlightConnection(
00053       std::string flight_id,
00054       std::string departure_city,
00055       std::string destination_city,
00056       std::string departure_time,
00057       std::string arrival_time,
00058       int available_seats,
00059       double price);
00060
00065   [[nodiscard]] std::string GetDepartureCity() const;
00066
00071   [[nodiscard]] std::string GetDestinationCity() const;
00072
00077   [[nodiscard]] std::string GetDepartureTime() const;
00078
00083   [[nodiscard]] std::string GetArrivalTime() const;
00084
00089   [[nodiscard]] std::string GetIdentifier() const;
00090
00095   [[nodiscard]] int GetAvailableSeats() const;
00096
00101   [[nodiscard]] double GetPrice() const;
00102
00107   std::vector<FlightConnection> FindAllConnections();
00108
00115   FlightConnection FindConnection(const std::string &departure_city, const std::string
      &destination_city);
00116
00123   std::vector<FlightConnection> FindConnectionByPrice(double &min_price, double &max_price);
00124
00130   FlightConnection FindConnectionById(const std::string &id);
00131
00137   std::vector<FlightConnection> FindConnectionsByDeparture(const std::string &departure_city);
00138
00144   std::vector<FlightConnection> FindConnectionsByDestination(const std::string &destination_city);
00145
00151   std::vector<int> GetSeatsTaken(const std::string &flight_identifier);
00152
00158   void UpdateSeatsTaken(const std::string &flight_identifier, const std::vector<int> &seats_taken);
00159 };
```

5.19 Dokumentacja pliku functions/helpers.h

Ten plik zawiera deklaracje różnych funkcji pomocniczych.

```
#include <string>
#include "ftxui/dom/table.hpp"
#include "ftxui/screen/color.hpp"
```

Funkcje

- std::string ExtractFileName (const std::string &path)

  Wyodrębnia nazwę pliku z ścieżki.
- void Countdown (int seconds, const std::string &type)

  Wykonuje odliczanie.
- std::string HashString (const std::string &string_to_hash)

  Hashuje ciąg znaków.
- void SetCellColor (ftxui::Table &table, int col, int row, ftxui::Color color)

  Ustawia kolor komórki w tabeli.

5.19.1 Opis szczegółowy

Ten plik zawiera deklaracje różnych funkcji pomocniczych.

5.19.2 Dokumentacja funkcji

5.19.2.1 Countdown()

```
void Countdown (
            int seconds,
            const std::string & type )
```

Wykonuje odliczanie.

Parametry

seconds	Liczba sekund do odliczenia.
type	Typ odliczania.

5.19.2.2 ExtractFileName()

```
std::string ExtractFileName (
            const std::string & path )
```

Wyodrębnia nazwę pliku z ścieżki.

Parametry

path	Ścieżka, z której wyodrębniana jest nazwa pliku.

Zwraca

Wyodrębniona nazwa pliku.

5.19.2.3 HashString()

```
std::string HashString (
            const std::string & string_to_hash )
```

Hashuje ciąg znaków.

Parametry

string_to_hash	Ciąg znaków do zahashowania.

Zwraca

Zahashowany ciąg znaków.

5.19.2.4 SetCellColor()

```
void SetCellColor (
            ftxui::Table & table,
            int col,
            int row,
            ftxui::Color color )
```

Ustawia kolor komórki w tabeli.

Parametry

table	Tabela zawierająca komórkę.
col	Kolumna komórki.
row	Rząd komórki.
color	Kolor, na który ma zostać ustawiona komórka.

5.20 helpers.h

Idź do dokumentacji tego pliku.

```
00001
00006 #ifndef AIRPORT_HELPERS_H
00007 #define AIRPORT_HELPERS_H
00008
00009 #include <string>
00010 #include "ftxui/dom/table.hpp"
00011 #include "ftxui/screen/color.hpp"
00012
00018 std::string ExtractFileName(const std::string &path);
00019
00025 void Countdown(int seconds, const std::string &type);
00026
00032 std::string HashString(const std::string &string_to_hash);
00033
00041 void SetCellColor(ftxui::Table &table, int col, int row, ftxui::Color color);
00042
00043 #endif  // AIRPORT_HELPERS_H
```

5.21 info_prints.h

```
00001
00006 #ifndef FUNCTIONS_H
00007 #define FUNCTIONS_H
00008
00009 #include <string>
00010
00011 #include "../../user/user.h"
00012
00018 void PrintSuccessMessage(const std::string &title_message, const std::string &optional_message);
00019
00025 void PrintErrorMessage(const std::string &title_message, const std::string &optional_message);
00026
00031 void PrintLogout(User &user);
00032
00036 void PrintSeeya();
00037
00038 #endif  // FUNCTIONS_H
```

5.22 main_handler.h

```
00001
00006 #ifndef MAIN_FUNCTIONS_H
00007 #define MAIN_FUNCTIONS_H
00008
00009 #include "../authentication/authentication.h"
00010 #include "../flights/flight_connection.h"
00011
00019 void ProcessChoice(bool is_logged_in, Authentication &auth, User &user, FlightConnection
      &flight_connection);
00020
00021 #endif  // MAIN_FUNCTIONS_H
```

5.23 Dokumentacja pliku functions/main_prints/main_prints.h

Ten plik zawiera deklaracje różnych funkcji wyświetlających i przechwytujących dane wejściowe.

```
#include <memory>
#include "ftxui/dom/elements.hpp"
#include "../../user/user.h"
```

Funkcje

- void PrintScreen (const std::shared_ptr< ftxui::Element > &screen)

  Drukuje ekran.
- void PrintFullWidthScreen (std::shared_ptr< ftxui::Node > container)

  Drukuje ekran o pełnej szerokości.
- void PrintNodeScreen (std::shared_ptr< ftxui::Node > container)

  Drukuje ekran węzła.
- std::string DisplayMessageAndCaptureStringInput (const std::string &title_message, const std::string &text↩ _message)

  Wyświetla wiadomość i przechwytuje dane wejściowe typu string.
- double DisplayMessageAndCaptureDoubleInput (const std::string &title_message, const std::string &text_↩ message)

  Wyświetla wiadomość i przechwytuje dane wejściowe typu double.
- std::string DisplayWarningAndCaptureInput (const std::string &title_message, const std::string &text_↩ message)

  Wyświetla ostrzeżenie i przechwytuje dane wejściowe.
- void DisplayUserMenu (User &user)

  Wyświetla menu użytkownika.
- void **DisplayMenu** ()

  Wyświetla główne menu.

5.23.1 Opis szczegółowy

Ten plik zawiera deklaracje różnych funkcji wyświetlających i przechwytujących dane wejściowe.

5.23.2 Dokumentacja funkcji

5.23.2.1 DisplayMessageAndCaptureDoubleInput()

```
double DisplayMessageAndCaptureDoubleInput (
            const std::string & title_message,
            const std::string & text_message )
```

Wyświetla wiadomość i przechwytuje dane wejściowe typu double.

Parametry

title_message	Tytuł wiadomości.
text_message	Tekst wiadomości.

Zwraca

Przechwycone dane wejściowe typu double.

5.23.2.2 DisplayMessageAndCaptureStringInput()

```
std::string DisplayMessageAndCaptureStringInput (
            const std::string & title_message,
            const std::string & text_message )
```

Wyświetla wiadomość i przechwytuje dane wejściowe typu string.

Parametry

title_message	Tytuł wiadomości.
text_message	Tekst wiadomości.

Zwraca

Przechwycone dane wejściowe typu string.

5.23.2.3 DisplayUserMenu()

```
void DisplayUserMenu (
            User & user )
```

Wyświetla menu użytkownika.

Parametry

user	Użytkownik, dla którego wyświetlane jest menu.

5.23.2.4 DisplayWarningAndCaptureInput()

```
std::string DisplayWarningAndCaptureInput (
            const std::string & title_message,
            const std::string & text_message )
```

Wyświetla ostrzeżenie i przechwytuje dane wejściowe.

Parametry

title_message	Tytuł wiadomości.
text_message	Tekst wiadomości.

Zwraca

Przechwycone dane wejściowe.

5.23.2.5 PrintFullWidthScreen()

```
void PrintFullWidthScreen (
            std::shared_ptr< ftxui::Node > container )
```

Drukuje ekran o pełnej szerokości.

Parametry

container	Kontener do wydrukowania.

5.23.2.6 PrintNodeScreen()

```
void PrintNodeScreen (
            std::shared_ptr< ftxui::Node > container )
```

Drukuje ekran węzła.

Parametry

container	Kontener do wydrukowania.

5.23.2.7 PrintScreen()

```
void PrintScreen (
            const std::shared_ptr< ftxui::Element > & screen )
```

Drukuje ekran.

Parametry

screen	Ekran do wydrukowania.

5.24 main_prints.h

Idź do dokumentacji tego pliku.

```
00001
00006 #ifndef AIRPORT_MAIN_PRINTS_H
00007 #define AIRPORT_MAIN_PRINTS_H
00008
00009 #include <memory>
00010
00011 #include "ftxui/dom/elements.hpp"
00012 #include "../../user/user.h"
00013
00018 void PrintScreen(const std::shared_ptr<ftxui::Element> &screen);
00019
00024 void PrintFullWidthScreen(std::shared_ptr<ftxui::Node> container);
00025
00030 void PrintNodeScreen(std::shared_ptr<ftxui::Node> container);
00031
00038 std::string DisplayMessageAndCaptureStringInput(const std::string &title_message, const std::string
      &text_message);
00039
00046 double DisplayMessageAndCaptureDoubleInput(const std::string &title_message, const std::string
      &text_message);
00047
00054 std::string DisplayWarningAndCaptureInput(const std::string &title_message, const std::string
      &text_message);
00055
00060 void DisplayUserMenu(User &user);
00061
00065 void DisplayMenu();
00066
00067 #endif  // AIRPORT_MAIN_PRINTS_H
```

5.25 Dokumentacja pliku luggage/item/item.h

Ten plik zawiera deklarację klasy Item.

```
#include <string>
#include <vector>
```

Komponenty

- class Item

  Ta klasa reprezentuje przedmiot w systemie lotniska.

5.25.1 Opis szczegółowy

Ten plik zawiera deklarację klasy Item.

5.26 item.h

Idź do dokumentacji tego pliku.

```
00001
00006 #ifndef AIRPORT_ITEM_H
00007 #define AIRPORT_ITEM_H
00008
00009 #include <string>
00010 #include <vector>
00011
00012 class User;
```

```
00013
00018 class Item {
00019   std::string item_name_;
00020   std::string description_;
00021   std::vector<std::string> hints_;
00022   bool forbidden_;
00023   bool registered_luggage_;
00024   bool hand_luggage_;
00025   bool pilot_allowance_;
00026   double max_count_;
00027   double weight_;
00028   std::string profession_;
00029   std::string category_ = "special";
00030
00031  public:
00046   Item(const std::string &item_name,
00047        const std::string &description,
00048        const std::vector<std::string> &hints,
00049        bool forbidden,
00050        bool registered_luggage,
00051        bool hand_luggage,
00052        bool pilot_allowance,
00053        double max_count,
00054        double weight,
00055        std::string &profession,
00056        std::string &category);
00057
00071   Item(const std::string &item_name,
00072        const std::string &description,
00073        const std::vector<std::string> &hints,
00074        bool forbidden,
00075        bool registered_luggage,
00076        bool hand_luggage,
00077        bool pilot_allowance,
00078        double max_count,
00079        double weight,
00080        std::string &category);
00081
00086   [[nodiscard]] const std::string &GetItemName() const;
00087
00092   [[nodiscard]] const std::string &GetDescription() const;
00093
00098   [[nodiscard]] const std::string &GetProfession() const;
00099
00104   [[nodiscard]] const std::vector<std::string> &GetHints() const;
00105
00110   [[nodiscard]] bool IsForbidden() const;
00111
00116   [[nodiscard]] bool IsRegisteredLuggage() const;
00117
00122   [[nodiscard]] bool IsHandLuggage() const;
00123
00128   [[nodiscard]] bool IsPilotAllowance() const;
00129
00134   [[nodiscard]] double GetMaxCount() const;
00135
00140   [[nodiscard]] double GetWeight() const;
00141
00146   [[nodiscard]] std::string GetCategory() const;
00147 };
00148
00149 #endif  // AIRPORT_ITEM_H
```

5.27 Dokumentacja pliku luggage/item/item_handler.h

Ten plik zawiera deklaracje różnych funkcji obsługujących przedmioty.

```
#include <vector>
#include "../../user/user.h"
#include "item.h"
```

Funkcje

- double GetDoubleValue (const bsoncxx::document::view &item, const std::string &key)

Pobiera wartość typu double z dokumentu BSON.

- std::vector< std::string > GetArrayValue (const bsoncxx::document::view &item, const std::string &key)

  Pobiera wartość tablicy z dokumentu BSON.
- std::string GetStringValue (const bsoncxx::document::view &item, const std::string &key)

  Pobiera wartość typu string z dokumentu BSON.
- std::vector< Item > GetItems (User &user)

  Pobiera przedmioty dla użytkownika.

5.27.1 Opis szczegółowy

Ten plik zawiera deklaracje różnych funkcji obsługujących przedmioty.

5.27.2 Dokumentacja funkcji

5.27.2.1 GetArrayValue()

```
std::vector< std::string > GetArrayValue (
            const bsoncxx::document::view & item,
            const std::string & key )
```

Pobiera wartość tablicy z dokumentu BSON.

Parametry

item	Dokument BSON.
key	Klucz wartości do pobrania.

Zwraca

Wartość tablicy.

5.27.2.2 GetDoubleValue()

```
double GetDoubleValue (
            const bsoncxx::document::view & item,
            const std::string & key )
```

Pobiera wartość typu double z dokumentu BSON.

Parametry

item	Dokument BSON.
key	Klucz wartości do pobrania.

Zwraca

Wartość typu double.

5.27.2.3 GetItems()

```
std::vector< Item > GetItems (
            User & user )
```

Pobiera przedmioty dla użytkownika.

Parametry

user	Użytkownik, dla którego pobierane są przedmioty.

Zwraca

Przedmioty dla użytkownika.

5.27.2.4 GetStringValue()

```
std::string GetStringValue (
            const bsoncxx::document::view & item,
            const std::string & key )
```

Pobiera wartość typu string z dokumentu BSON.

Parametry

item	Dokument BSON.
key	Klucz wartości do pobrania.

Zwraca

Wartość typu string.

5.28 item_handler.h

Idź do dokumentacji tego pliku.

```
00001
00006 #ifndef ITEM_HANDLER_H
00007 #define ITEM_HANDLER_H
00008
00009 #include <vector>
00010
00011 #include "../../user/user.h"
00012 #include "item.h"
00013
00020 double GetDoubleValue(const bsoncxx::document::view &item, const std::string &key);
00021
00028 std::vector<std::string> GetArrayValue(const bsoncxx::document::view &item, const std::string &key);
00029
```

```
00036 std::string GetStringValue(const bsoncxx::document::view &item, const std::string &key);
00037
00043 std::vector<Item> GetItems(User &user);
00044
00045 #endif  // ITEM_HANDLER_H
```

5.29 Dokumentacja pliku luggage/luggage.h

Ten plik zawiera deklarację klasy Luggage.

```
#include <vector>
#include <iostream>
#include "item/item.h"
```

Komponenty

- class Luggage

  Ta klasa reprezentuje bagaż w systemie lotniska.

5.29.1 Opis szczegółowy

Ten plik zawiera deklarację klasy Luggage.

5.30 luggage.h

Idź do dokumentacji tego pliku.

```
00001
00006 #ifndef AIRPORT_LUGGAGE_H
00007 #define AIRPORT_LUGGAGE_H
00008
00009 #include <vector>
00010 #include <iostream>
00011 #include "item/item.h"
00012
00013 class User;
00014
00019 class Luggage {
00020   std::vector<Item> items_;
00021   double total_weight_ = 0.0;
00022
00023  public:
00029   Luggage(
00030       const std::vector<Item> &items,
00031       double total_weight
00032   ) : items_(items), total_weight_(total_weight) {}
00033
00038   double ProcessItemsAndGetWeight();
00039
00045   std::tuple<bool, std::string> ConfirmItems(User &user);
00046
00047   const double overweight_fee_per_kg_ = 2.0;
00048   const double euro_to_pln_ = 4.32;
00049   const double max_allowed_weight_ = 32.0;
00050   double max_weight_ = 20.0;
00051
00057   double CalculateOverweightFee(double weight) const;
00058 };
00059
00060 #endif //AIRPORT_LUGGAGE_H
```

5.31 Dokumentacja pliku luggage/luggage_handler.h

Ten plik zawiera deklarację funkcji CheckIn.

```
#include "../user/user.h"
```

Funkcje

- void CheckIn (User &user, int flightNumber)

  Przeprowadza odprawę użytkownika na konkretny lot.

5.31.1 Opis szczegółowy

Ten plik zawiera deklarację funkcji CheckIn.

5.31.2 Dokumentacja funkcji

5.31.2.1 CheckIn()

```
void CheckIn (
            User & user,
            int flightNumber )
```

Przeprowadza odprawę użytkownika na konkretny lot.

Parametry

user	Użytkownik do odprawy.
flightNumber	Numer lotu, na który odprawiany jest użytkownik.

5.32 luggage_handler.h

Idź do dokumentacji tego pliku.

```
00001
00006 #ifndef AIRPORT_LUGGAGEHANDLER_H
00007 #define AIRPORT_LUGGAGEHANDLER_H
00008
00009 #include "../user/user.h"
00010
00016 void CheckIn(User &user, int flightNumber);
00017
00018 #endif  // AIRPORT_LUGGAGEHANDLER_H
```

5.33 Dokumentacja pliku luggage/luggage_prints/luggage_prints.h

Ten plik zawiera deklaracje różnych funkcji obsługujących bagaż.

```
#include "../../user/user.h"
#include "../item/item.h"
#include "ftxui/component/component.hpp"
```

Funkcje

- std::vector< ftxui::Component > CreateGroups (const std::vector< ftxui::Component > &checkbox_components)

  Tworzy grupy komponentów.
- void PrintAllItems (User &user)

  Drukuje wszystkie przedmioty dla użytkownika.
- void PrintSpecificItem (Item &item)

  Drukuje konkretny przedmiot.
- void PrintWelcomeInCheckIn (User &user)

  Drukuje powitanie przy odprawie.

Zmienne

- const std::string **AIRPORT_NAME** = "WOLFI AIRPORT "

  Nazwa lotniska.
- const std::string **ITEM_CARD** = "KARTA PRZEDMIOTU"

  Karta przedmiotu.

5.33.1 Opis szczegółowy

Ten plik zawiera deklaracje różnych funkcji obsługujących bagaż.

5.33.2 Dokumentacja funkcji

5.33.2.1 CreateGroups()

```
std::vector< ftxui::Component > CreateGroups (
            const std::vector< ftxui::Component > & checkbox_components )
```

Tworzy grupy komponentów.

Parametry

checkbox_components	Komponenty do pogrupowania.

Zwraca

Grupy komponentów.

5.33.2.2 PrintAllItems()

```
void PrintAllItems (
            User & user )
```

Drukuje wszystkie przedmioty dla użytkownika.

Parametry

user	Użytkownik, dla którego drukowane są przedmioty.

5.33.2.3 PrintSpecificItem()

```
void PrintSpecificItem (
            Item & item )
```

Drukuje konkretny przedmiot.

Parametry

item	Przedmiot do wydrukowania.

5.33.2.4 PrintWelcomeInCheckIn()

```
void PrintWelcomeInCheckIn (
            User & user )
```

Drukuje powitanie przy odprawie.

Parametry

user	Użytkownik, dla którego drukowane jest powitanie.

5.34 luggage_prints.h

Idź do dokumentacji tego pliku.

```
00001
00006 #ifndef AIRPORT_LUGGAGE_PRINTS_H
00007 #define AIRPORT_LUGGAGE_PRINTS_H
00008
00009 #include "../../user/user.h"
00010 #include "../item/item.h"
00011 #include "ftxui/component/component.hpp"
00012
00013 const std::string AIRPORT_NAME = "WOLFI AIRPORT  ";
00014 const std::string ITEM_CARD = "KARTA PRZEDMIOTU";
00015
00021 std::vector<ftxui::Component> CreateGroups(const std::vector<ftxui::Component> &checkbox_components);
00022
00027 void PrintAllItems(User &user);
00028
00033 void PrintSpecificItem(Item &item);
```

```
00034
00039 void PrintWelcomeInCheckIn(User &user);
00040
00041 #endif  // AIRPORT_LUGGAGE_PRINTS_H
```

5.35 Dokumentacja pliku plane/plane.h

Ten plik zawiera deklarację funkcji ProcessSeatSelectionAndPurchase.

```
#include <vector>
#include "../flights/flight_connection.h"
#include "../user/user.h"
```

Funkcje

- void ProcessSeatSelectionAndPurchase (std::vector< int > seat_number, FlightConnection &flight_↩ connection, FlightConnection &found_connection, User &user)

  Przetwarza wybór miejsca i zakup dla lotu.

5.35.1 Opis szczegółowy

Ten plik zawiera deklarację funkcji ProcessSeatSelectionAndPurchase.

5.35.2 Dokumentacja funkcji

5.35.2.1 ProcessSeatSelectionAndPurchase()

```
void ProcessSeatSelectionAndPurchase (
            std::vector< int > seat_number,
            FlightConnection & flight_connection,
            FlightConnection & found_connection,
            User & user )
```

Przetwarza wybór miejsca i zakup dla lotu.

Parametry

seat_number	Wybrane numery miejsc.
flight_connection	Połączenie lotnicze.
found_connection	Znalezione połączenie lotnicze.
user	Użytkownik dokonujący zakupu.

5.36 plane.h

Idź do dokumentacji tego pliku.

```
00001
00006 #ifndef AIRPORT_PLANE_H
00007 #define AIRPORT_PLANE_H
00008
00009 #include <vector>
00010
00011 #include "../flights/flight_connection.h"
00012 #include "../user/user.h"
00013
00021 void ProcessSeatSelectionAndPurchase(std::vector<int> seat_number,
00022                                      FlightConnection &flight_connection,
00023                                      FlightConnection &found_connection,
00024                                      User &user);
00025
00026 #endif  // AIRPORT_PLANE_H
```

5.37 Dokumentacja pliku qr_code/qrcode_prints.h

Ten plik zawiera deklaracje funkcji tworzenia i drukowania kodów QR.

```
#include "qrcodegen.hpp"
```

Funkcje

- void CreateQr (const std::string &email, const std::string &username, const std::string &flight_id, std::vector< int > seats)

  Tworzy kod QR na podstawie informacji o użytkowniku i locie.
- void PrintQr (const QrCode &qr)

  Drukuje kod QR.

5.37.1 Opis szczegółowy

Ten plik zawiera deklaracje funkcji tworzenia i drukowania kodów QR.

5.37.2 Dokumentacja funkcji

5.37.2.1 CreateQr()

```
void CreateQr (
            const std::string & email,
            const std::string & username,
            const std::string & flight_id,
            std::vector< int > seats )
```

Tworzy kod QR na podstawie informacji o użytkowniku i locie.

Parametry

email	Adres email użytkownika.
username	Nazwa użytkownika.
flight_id	ID lotu.
seats	Miejsca wybrane przez użytkownika.

5.37.2.2 PrintQr()

```
void PrintQr (
            const QrCode & qr )
```

Drukuje kod QR.

Parametry

qr	Kod QR do wydrukowania.

5.38 qrcode_prints.h

Idź do dokumentacji tego pliku.

```
00001
00006 #ifndef AIRPORT_QRCODE_PRINTS_H
00007 #define AIRPORT_QRCODE_PRINTS_H
00008
00009 #include "qrcodegen.hpp"
00010
00011 using qrcodegen::QrCode;
00012 using qrcodegen::QrSegment;
00013 using std::uint8_t;
00014
00022 void CreateQr(const std::string &email,
00023               const std::string &username,
00024               const std::string &flight_id,
00025               std::vector<int> seats);
00026
00031 void PrintQr(const QrCode &qr);
00032
00033 #endif  // AIRPORT_QRCODE_PRINTS_H
```

5.39 Dokumentacja pliku tickets/tickets.h

Ten plik zawiera deklaracje różnych funkcji obsługujących bilety.

```
#include "../flights/flight_connection.h"
#include "../user/user.h"
```

Funkcje

- void HandleTicketChoice (FlightConnection &flight_connection, User &user)

  Obsługuje wybór biletu przez użytkownika.
- void HandleBuyTicket (int choice, FlightConnection &flight_connection, User &user)

  Obsługuje zakup biletu.
- void HandleFlightById (FlightConnection &flight_connection, User &user)

  Obsługuje lot według jego ID.
- void HandleFlightByData (FlightConnection &flight_connection, User &user)

  Obsługuje lot według jego danych.
- void ProcessPurchase (FlightConnection &flight_connection, FlightConnection &found_connection, User &user)

  Przetwarza zakup biletu.

Zmienne

- const int **MAX_TICKETS** = 4

  Maksymalna liczba biletów.
- const int **EMERGENCY_SEAT_ONE** = 37

  Numer pierwszego miejsca awaryjnego.
- const int **EMERGENCY_SEAT_TWO** = 45

  Numer drugiego miejsca awaryjnego.

5.39.1 Opis szczegółowy

Ten plik zawiera deklaracje różnych funkcji obsługujących bilety.

5.39.2 Dokumentacja funkcji

5.39.2.1 HandleBuyTicket()

```
void HandleBuyTicket (
            int choice,
            FlightConnection & flight_connection,
            User & user )
```

Obsługuje zakup biletu.

Parametry

choice	Wybór użytkownika.
flight_connection	Połączenie lotnicze.
user	Użytkownik dokonujący zakupu.

5.39.2.2 HandleFlightByData()

```
void HandleFlightByData (
            FlightConnection & flight_connection,
            User & user )
```

Obsługuje lot według jego danych.

Parametry

flight_connection	Połączenie lotnicze.
user	Użytkownik.

5.39.2.3 HandleFlightById()

```
void HandleFlightById (
```

```
            FlightConnection & flight_connection,
            User & user )
```

Obsługuje lot według jego ID.

Parametry

flight_connection	Połączenie lotnicze.
user	Użytkownik.

5.39.2.4 HandleTicketChoice()

```
void HandleTicketChoice (
            FlightConnection & flight_connection,
            User & user )
```

Obsługuje wybór biletu przez użytkownika.

Parametry

flight_connection	Połączenie lotnicze.
user	Użytkownik dokonujący wyboru.

5.39.2.5 ProcessPurchase()

```
void ProcessPurchase (
            FlightConnection & flight_connection,
            FlightConnection & found_connection,
            User & user )
```

Przetwarza zakup biletu.

Parametry

flight_connection	Połączenie lotnicze.
found_connection	Znalezione połączenie lotnicze.
user	Użytkownik dokonujący zakupu.

5.40 tickets.h

Idź do dokumentacji tego pliku.

```
00001
00006 #ifndef AIRPORT_TICKETS_H
00007 #define AIRPORT_TICKETS_H
00008
00009 #include "../flights/flight_connection.h"
00010 #include "../user/user.h"
00011
00012 const int MAX_TICKETS = 4;
```

```
00013 const int EMERGENCY_SEAT_ONE = 37;
00014 const int EMERGENCY_SEAT_TWO = 45;
00015
00021 void HandleTicketChoice(FlightConnection &flight_connection, User &user);
00022
00029 void HandleBuyTicket(int choice, FlightConnection &flight_connection, User &user);
00030
00036 void HandleFlightById(FlightConnection &flight_connection, User &user);
00037
00043 void HandleFlightByData(FlightConnection &flight_connection, User &user);
00044
00051 void ProcessPurchase(
00052     FlightConnection &flight_connection,
00053     FlightConnection &found_connection,
00054     User &user);
00055
00056 #endif  // AIRPORT_TICKETS_H
```

5.41 Dokumentacja pliku user/discounts/discounts.h

Ten plik zawiera deklaracje różnych funkcji obsługujących zniżki.

```
#include "../user.h"
```

Funkcje

- double GetDiscount (std::string choice)

  Pobiera zniżkę na podstawie wyboru.
- void HandleDiscountChoice (User &user, std::string choice)

  Obsługuje wybór zniżki przez użytkownika.
- void PrintDiscountCard (User &user)

  Drukuje kartę zniżkową dla użytkownika.

5.41.1 Opis szczegółowy

Ten plik zawiera deklaracje różnych funkcji obsługujących zniżki.

5.41.2 Dokumentacja funkcji

5.41.2.1 GetDiscount()

```
double GetDiscount (
            std::string choice )
```

Pobiera zniżkę na podstawie wyboru.

Parametry

choice	Wybór użytkownika.

Zwraca

Zniżka jako double.

5.41.2.2 HandleDiscountChoice()

```
void HandleDiscountChoice (
            User & user,
            std::string choice )
```

Obsługuje wybór zniżki przez użytkownika.

Parametry

user	Użytkownik dokonujący wyboru.
choice	Wybór użytkownika.

5.41.2.3 PrintDiscountCard()

```
void PrintDiscountCard (
            User & user )
```

Drukuje kartę zniżkową dla użytkownika.

Parametry

user	Użytkownik, dla którego drukowana jest karta zniżkowa.

5.42 discounts.h

Idź do dokumentacji tego pliku.

```
00001
00006 #ifndef AIRPORT_DISCOUNTS_H
00007 #define AIRPORT_DISCOUNTS_H
00008
00009 #include "../user.h"
00010
00016 double GetDiscount(std::string choice);
00017
00023 void HandleDiscountChoice(User &user, std::string choice);
00024
00029 void PrintDiscountCard(User &user);
00030
00031 #endif  // AIRPORT_DISCOUNTS_H
```

5.43 Dokumentacja pliku user/premium_cards/premium_cards.h

Ten plik zawiera deklaracje różnych funkcji obsługujących karty premium.

```
#include "../user.h"
```

Funkcje

- void HandlePremiumCard (User &user)

  Obsługuje kartę premium użytkownika.
- void HandleCardChoice (const std::string &card, int price, User &user)

  Obsługuje wybór karty przez użytkownika.
- double GetCardDiscount (const std::string &card)

  Pobiera zniżkę z karty.
- std::string RecognizeDiscountCard (double discount)

  Rozpoznaje kartę zniżkową na podstawie zniżki.

5.43.1 Opis szczegółowy

Ten plik zawiera deklaracje różnych funkcji obsługujących karty premium.

5.43.2 Dokumentacja funkcji

5.43.2.1 GetCardDiscount()

```
double GetCardDiscount (
            const std::string & card )
```

Pobiera zniżkę z karty.

Parametry

card	Karta, dla której pobierana jest zniżka.

Zwraca

Zniżka jako double.

5.43.2.2 HandleCardChoice()

```
void HandleCardChoice (
            const std::string & card,
            int price,
            User & user )
```

Obsługuje wybór karty przez użytkownika.

Parametry

card	Karta wybrana przez użytkownika.
price	Cena karty.
user	Użytkownik dokonujący wyboru.

5.43.2.3 HandlePremiumCard()

```
void HandlePremiumCard (
            User & user )
```

Obsługuje kartę premium użytkownika.

Parametry

user	Użytkownik z kartą premium.

5.43.2.4 RecognizeDiscountCard()

```
std::string RecognizeDiscountCard (
            double discount )
```

Rozpoznaje kartę zniżkową na podstawie zniżki.

Parametry

discount	Zniżka do rozpoznania karty.

Zwraca

Rozpoznana karta jako string.

5.44 premium_cards.h

Idź do dokumentacji tego pliku.

```
00001
00006 #ifndef AIRPORT_PREMIUM_CARDS_H
00007 #define AIRPORT_PREMIUM_CARDS_H
00008
00009 #include "../user.h"
00010
00015 void HandlePremiumCard(User &user);
00016
00023 void HandleCardChoice(const std::string &card, int price, User &user);
00024
00030 double GetCardDiscount(const std::string &card);
00031
00037 std::string RecognizeDiscountCard(double discount);
00038
00039 #endif  // AIRPORT_PREMIUM_CARDS_H
```

5.45 Dokumentacja pliku user/professions/profession_choice.h

Ten plik zawiera deklaracje różnych funkcji obsługujących wybór zawodu przez użytkownika.

```
#include "../user.h"
```

Funkcje

- void MusicProfession (User &user)

  Obsługuje wybór zawodu muzycznego przez użytkownika.
- void MathProfession (User &user)

  Obsługuje wybór zawodu matematycznego przez użytkownika.
- void InformaticProfession (User &user)

  Obsługuje wybór zawodu informatycznego przez użytkownika.
- void DoctorProfession (User &user)

  Obsługuje wybór zawodu lekarskiego przez użytkownika.
- void PoliceProfession (User &user)

  Obsługuje wybór zawodu policyjnego przez użytkownika.

5.45.1 Opis szczegółowy

Ten plik zawiera deklaracje różnych funkcji obsługujących wybór zawodu przez użytkownika.

5.45.2 Dokumentacja funkcji

5.45.2.1 DoctorProfession()

```
void DoctorProfession (
            User & user )
```

Obsługuje wybór zawodu lekarskiego przez użytkownika.

Parametry

user	Użytkownik dokonujący wyboru.

5.45.2.2 InformaticProfession()

```
void InformaticProfession (
            User & user )
```

Obsługuje wybór zawodu informatycznego przez użytkownika.

Parametry

user	Użytkownik dokonujący wyboru.

5.45.2.3 MathProfession()

```
void MathProfession (
            User & user )
```

Obsługuje wybór zawodu matematycznego przez użytkownika.

Parametry

user	Użytkownik dokonujący wyboru.

5.45.2.4 MusicProfession()

```
void MusicProfession (
            User & user )
```

Obsługuje wybór zawodu muzycznego przez użytkownika.

Parametry

user	Użytkownik dokonujący wyboru.

5.45.2.5 PoliceProfession()

```
void PoliceProfession (
            User & user )
```

Obsługuje wybór zawodu policyjnego przez użytkownika.

Parametry

user	Użytkownik dokonujący wyboru.

5.46 profession_choice.h

Idź do dokumentacji tego pliku.

```
00001
00006 #ifndef AIRPORT_PROFESSION_CHOICE_H
00007 #define AIRPORT_PROFESSION_CHOICE_H
00008
00009 #include "../user.h"
00010
00015 void MusicProfession(User &user);
00016
00021 void MathProfession(User &user);
00022
00027 void InformaticProfession(User &user);
00028
00033 void DoctorProfession(User &user);
00034
00039 void PoliceProfession(User &user);
00040
00041 #endif  // AIRPORT_PROFESSION_CHOICE_H
```

5.47 Dokumentacja pliku user/professions/profession_handler.h

Ten plik zawiera deklaracje różnych funkcji obsługujących pytania związane z zawodem.

```
#include <string>
#include "../user.h"
```

Funkcje

- bool GuessMusicAuthor (const std::string &music_link)

  Próbuje odgadnąć autora utworu muzycznego na podstawie linku.
- bool GuessDoctorQuestion (User &user)

  Próbuje odgadnąć odpowiedź na pytanie związane z zawodem lekarza.
- bool GuessInformaticQuestion (User &user)

  Próbuje odgadnąć odpowiedź na pytanie związane z informatyką.
- bool GuessMathQuestion (User &user)

  Próbuje odgadnąć odpowiedź na pytanie związane z matematyką.
- bool DisplayPoliceProfession ()

  Wyświetla informacje związane z zawodem policjanta.

5.47.1 Opis szczegółowy

Ten plik zawiera deklaracje różnych funkcji obsługujących pytania związane z zawodem.

5.47.2 Dokumentacja funkcji

5.47.2.1 DisplayPoliceProfession()

```
bool DisplayPoliceProfession ( )
```

Wyświetla informacje związane z zawodem policjanta.

Zwraca

Prawda, jeśli operacja jest udana, w przeciwnym razie fałsz.

5.47.2.2 GuessDoctorQuestion()

```
bool GuessDoctorQuestion (
            User & user )
```

Próbuje odgadnąć odpowiedź na pytanie związane z zawodem lekarza.

Parametry

user	Użytkownik, który próbuje odgadnąć.

Zwraca

Prawda, jeśli zgadnięcie jest poprawne, w przeciwnym razie fałsz.

5.47.2.3 GuessInformaticQuestion()

```
bool GuessInformaticQuestion (
            User & user )
```

Próbuje odgadnąć odpowiedź na pytanie związane z informatyką.

Parametry

user	Użytkownik, który próbuje odgadnąć.

Zwraca

Prawda, jeśli zgadnięcie jest poprawne, w przeciwnym razie fałsz.

5.47.2.4 GuessMathQuestion()

```
bool GuessMathQuestion (
            User & user )
```

Próbuje odgadnąć odpowiedź na pytanie związane z matematyką.

Parametry

user	Użytkownik, który próbuje odgadnąć.

Zwraca

Prawda, jeśli zgadnięcie jest poprawne, w przeciwnym razie fałsz.

5.47.2.5 GuessMusicAuthor()

```
bool GuessMusicAuthor (
            const std::string & music_link )
```

Próbuje odgadnąć autora utworu muzycznego na podstawie linku.

Parametry

music_link	Link do utworu muzycznego.

Zwraca

Prawda, jeśli zgadnięcie jest poprawne, w przeciwnym razie fałsz.

5.48 profession_handler.h

Idź do dokumentacji tego pliku.

```
00001
00006 #ifndef AIRPORT_PROFESSION_HANDLER_H
00007 #define AIRPORT_PROFESSION_HANDLER_H
00008
00009 #include <string>
00010
00011 #include "../user.h"
00012
00018 bool GuessMusicAuthor(const std::string &music_link);
00019
00025 bool GuessDoctorQuestion(User &user);
00026
00032 bool GuessInformaticQuestion(User &user);
00033
00039 bool GuessMathQuestion(User &user);
00040
00045 bool DisplayPoliceProfession();
00046
00047 #endif  // AIRPORT_PROFESSION_HANDLER_H
```

5.49 Dokumentacja pliku user/professions/profession_prints/profession_prints.h

Ten plik zawiera deklaracje funkcji wyświetlania informacji o zawodzie i walidacji odpowiedzi.

```
#include <string>
#include "../../user.h"
```

Funkcje

- int CreateProfessionScreen ()

  Tworzy ekran zawodu.
- std::string DisplayProfessionInfo ()

  Wyświetla informacje o zawodzie.
- void **InvalidAnswer** ()

  Obsługuje nieprawidłową odpowiedź.
- void ValidAnswer (const std::string &category, User &user)

  Obsługuje prawidłową odpowiedź.

5.49.1 Opis szczegółowy

Ten plik zawiera deklaracje funkcji wyświetlania informacji o zawodzie i walidacji odpowiedzi.

5.49.2 Dokumentacja funkcji

5.49.2.1 CreateProfessionScreen()

```
int CreateProfessionScreen ( )
```

Tworzy ekran zawodu.

Zwraca

Liczba całkowita reprezentująca status operacji.

5.49.2.2 DisplayProfessionInfo()

```
std::string DisplayProfessionInfo ( )
```

Wyświetla informacje o zawodzie.

Zwraca

Ciąg znaków zawierający informacje o zawodzie.

5.49.2.3 ValidAnswer()

```
void ValidAnswer (
            const std::string & category,
            User & user )
```

Obsługuje prawidłową odpowiedź.

Parametry

category	Kategoria odpowiedzi.
user	Użytkownik udzielający odpowiedzi.

5.50 profession_prints.h

Idź do dokumentacji tego pliku.

```
00001
00006 #ifndef AIRPORT_PROFESSION_PRINTS_H
00007 #define AIRPORT_PROFESSION_PRINTS_H
00008
00009 #include <string>
00010
00011 #include "../../user.h"
00012
00017 int CreateProfessionScreen();
00018
00023 std::string DisplayProfessionInfo();
00024
00028 void InvalidAnswer();
```

```
00029
00035 void ValidAnswer(const std::string &category, User &user);
00036
00037 #endif  // AIRPORT_PROFESSION_PRINTS_H
```

5.51 Dokumentacja pliku user/professions/user_profession_functions.h

Ten plik zawiera deklaracje funkcji obsługujących zawód użytkownika.

```
#include "../user.h"
```

Funkcje

- void HandleProfessionChoice (int choice, User &user)

  Obsługuje wybór zawodu przez użytkownika.
- void HandleProfession (User &user)

  Obsługuje zawód użytkownika.

5.51.1 Opis szczegółowy

Ten plik zawiera deklaracje funkcji obsługujących zawód użytkownika.

5.51.2 Dokumentacja funkcji

5.51.2.1 HandleProfession()

```
void HandleProfession (
            User & user )
```

Obsługuje zawód użytkownika.

Parametry

user	Użytkownik, którego zawód jest obsługiwany.

5.51.2.2 HandleProfessionChoice()

```
void HandleProfessionChoice (
            int choice,
            User & user )
```

Obsługuje wybór zawodu przez użytkownika.

Parametry

choice	Wybór dokonany przez użytkownika.
user	Użytkownik dokonujący wyboru.

5.52 user_profession_functions.h

Idź do dokumentacji tego pliku.

```
00001
00006 #ifndef AIRPORT_USER_PROFESSION_FUNCTIONS_H
00007 #define AIRPORT_USER_PROFESSION_FUNCTIONS_H
00008
00009 #include "../user.h"
00010
00016 void HandleProfessionChoice(int choice, User &user);
00017
00022 void HandleProfession(User &user);
00023
00024 #endif  // AIRPORT_USER_PROFESSION_FUNCTIONS_H
```

5.53 Dokumentacja pliku user/user.h

Ten plik zawiera deklarację klasy User.

```
#include <string>
#include "../flights/flight_connection.h"
#include "../luggage/luggage.h"
#include "mongocxx/client.hpp"
```

Komponenty

- class User

  Reprezentuje użytkownika w systemie.

5.53.1 Opis szczegółowy

Ten plik zawiera deklarację klasy User.

5.54 user.h

Idź do dokumentacji tego pliku.

```
00001
00006 #ifndef USER_H
00007 #define USER_H
00008
00009 #include <string>
00010
00011 #include "../flights/flight_connection.h"
00012 #include "../luggage/luggage.h"
00013 #include "mongocxx/client.hpp"
00014
00015 class Admin;
```

```
00016
00021 class User {
00022  private:
00023   mongocxx::database _db_;
00024   mongocxx::collection _collection_;
00025   std::string _password_;
00026
00027  protected:
00028   mongocxx::client &_client;
00029
00030  public:
00035   explicit User(mongocxx::client &client)
00036       : _client(client),
00037         _db_(client["projekt"]),
00038         _collection_(_db_["users"]),
00039         username_("gosc"),
00040         email_("brak"),
00041         discount_(1.0),
00042         discount_type_("brak"),
00043         premium_card_("brak"),
00044         payment_method_("blik"),
00045         profession_("brak"),
00046         registration_date_("brak"),
00047         money_spent_(0),
00048         money_saved_(0),
00049         ticket_bought_(0),
00050         user_flights_(std::vector<bsoncxx::document::value>{}),
00051         is_admin_(false) {}
00052
00056   User(std::string username, std::string email, double discount,
00057        std::string discount_type, std::string premium_card,
00058        std::string payment_method, mongocxx::client &client,
00059        std::string profession, std::string registration_date,
00060        double money_spent, double money_saved,
00061        int ticket_bought, std::vector<bsoncxx::document::value> user_flights, bool is_admin);
00062
00063   // Obiekt użytkownika (dane)
00064   std::string username_;
00065   std::string profession_;
00066   std::string email_;
00067   std::string discount_type_;
00068   double discount_;
00069   std::string premium_card_;
00070   std::string payment_method_;
00071   std::string registration_date_;
00072   double money_spent_;
00073   double money_saved_;
00074   int ticket_bought_;
00075   std::vector<bsoncxx::document::value> user_flights_;
00076   bool is_admin_;
00077
00078   // Funkcje użytkownika (metody)
00079   void Reset();
00080   mongocxx::collection &GetCollection();
00081   mongocxx::collection GetSpecificCollection(const std::string &collection_name);
00082   std::string GetPassword();
00083   void SetPassword(const std::string &password);
00084   void SetPremiumCard(User &user, const std::string &card);
00085   void SetBlik(const std::string &payment_method);
00086   void SetVisa(const std::string &card_number, const std::string &card_cvv);
00087   void ChangeUsername(const std::string &username);
00088   void ChangeEmail(const std::string &email);
00089   void ChangePassword(const std::string &password);
00090   void SetDiscount(double discount, const std::string &discount_type);
00091   [[nodiscard]] double GetDiscount() const;
00092   [[nodiscard]] std::string RecognizeDiscount() const;
00093   void AddTicketToUser(const std::vector<int> &seats,
00094                        const FlightConnection &flight_connection);
00095   void UpdateMoneySaved(double normal_price, double discount_price);
00096   Admin *LoginAsAdmin();
00097   [[nodiscard]] bool CheckIfAdmin() const;
00098   void SetIsAdmin(bool is_administrator) { User::is_admin_ = is_administrator; }
00099   void LuggageCheckin(int flight_number);
00100   mongocxx::cursor FindUserInDatabase();
00101
00108   template<typename T>
00109   void UpdateUserInDatabase(
00110       const std::string &value_in_database,
00111       const T &value_to_set) {
00112     bsoncxx::document::value update_builder = bsoncxx::builder::basic::make_document(
00113         bsoncxx::builder::basic::kvp("$set", bsoncxx::builder::basic::make_document(
00114             bsoncxx::builder::basic::kvp(value_in_database, value_to_set))));
00115
00116     bsoncxx::document::view update_view = update_builder.view();
00117     bsoncxx::document::value filter_builder_email_password = bsoncxx::builder::basic::make_document(
00118         bsoncxx::builder::basic::kvp("email_", email_),
00119         bsoncxx::builder::basic::kvp("password", GetPassword()));
```

```
00120
00121     bsoncxx::document::view filter_view_email_password = filter_builder_email_password.view();
00122     _collection_.update_one(filter_view_email_password, update_view);
00123   }
00124 };
00125
00126 #endif  // USER_H
```

5.55 Dokumentacja pliku user/user_functions/user_payments/user_payment_functions.h

Ten plik zawiera deklaracje funkcji obsługujących płatności użytkownika.

```
#include <iostream>
#include "../../user.h"
```

Funkcje

- void HandlePaymentOption (User &user)

  Obsługuje opcję płatności użytkownika.
- bool AuthenticatePayment (User &user, const std::string &payment_method, const std::string &title_message, int target_price)

  Autentykuje płatność dokonaną przez użytkownika.

5.55.1 Opis szczegółowy

Ten plik zawiera deklaracje funkcji obsługujących płatności użytkownika.

5.55.2 Dokumentacja funkcji

5.55.2.1 AuthenticatePayment()

```
bool AuthenticatePayment (
            User & user,
            const std::string & payment_method,
            const std::string & title_message,
            int target_price )
```

Autentykuje płatność dokonaną przez użytkownika.

Parametry

user	Użytkownik dokonujący płatności.
payment_method	Metoda płatności używana przez użytkownika.
title_message	Tytuł wiadomości dla płatności.
target_price	Docelowa cena płatności.

Zwraca

Prawda, jeśli płatność jest uwierzytelniona, w przeciwnym razie fałsz.

5.55.2.2 HandlePaymentOption()

```
void HandlePaymentOption (
            User & user )
```

Obsługuje opcję płatności użytkownika.

Parametry

user	Użytkownik dokonujący płatności.

5.56 user_payment_functions.h

Idź do dokumentacji tego pliku.

```
00001
00006 #ifndef USER_PAYMENT_FUNCTIONS_H
00007 #define USER_PAYMENT_FUNCTIONS_H
00008
00009 #include <iostream>
00010
00011 #include "../../user.h"
00012
00017 void HandlePaymentOption(User &user);
00018
00027 bool AuthenticatePayment(User &user, const std::string &payment_method, const std::string
      &title_message, int target_price);
00028
00029 #endif  // USER_PAYMENT_FUNCTIONS_H
```

5.57 user_prints.h

```
00001
00006 #ifndef AIRPORT_USER_PRINT_FUNCTIONS_H
00007 #define AIRPORT_USER_PRINT_FUNCTIONS_H
00008
00009 #include <string>
00010
00011 #include "../../user.h"
00012 #include "ftxui/dom/elements.hpp"
00013 #include "ftxui/screen/screen.hpp"
00014
00020 std::string DisplaySettingsMenu(const User &user);
00021
00026 int DisplayDefaultPaymentScreen();
00027
00032 void DisplayProfileScreen(User &user);
00033
00034 #endif  // AIRPORT_USER_PRINT_FUNCTIONS_H
```

5.58 user_settings_handler.h

```
00001
00006 #ifndef USER_SETTINGS_FUNCTIONS_H
00007 #define USER_SETTINGS_FUNCTIONS_H
00008
00009 #include <iostream>
00010
```

```
00011 #include "../../user.h"
00012
00017 void HandleSettingsOption(User &user);
00018
00019 #endif  // USER_SETTINGS_FUNCTIONS_H
```

5.59 user_tickets_prints.h

```
00001
00006 #ifndef AIRPORT_USER_TICKETS_PRINT_FUNCTIONS_H
00007 #define AIRPORT_USER_TICKETS_PRINT_FUNCTIONS_H
00008
00009 #include <string>
00010 #include <vector>
00011
00012 #include "../../user.h"
00013
00015 const int PAGE_SIZE = 4;
00016
00021 struct FlightInfo {
00022   int flight_number;
00023   std::string flight_id;
00024   std::string departure;
00025   std::string destination;
00026   std::string departure_time;
00027   double price;
00028   std::vector<int> seats;
00029   bool checkin;
00030   bool luggage_checkin;
00031 };
00032
00039 std::optional<std::string> CreateTicketsScreen(User &user, bool is_checkin = false);
00040
00041 #endif  // AIRPORT_USER_TICKETS_PRINT_FUNCTIONS_H
```

Skorowidz