Projekt Biura podróży. Oferuje ono wycieczki o różnych kategoriach: objazdowe, pobyty stałe, jednodniowe. Ponadto mamy trzy poziomy dostępów w zależności od użytkownika (użytkownik niezalogowany - gość, klient zalogowany, admin). Przy czym każdy z nich może być Clintem. Tylko adminowi będzie przysługiwać prawo modyfikacji zawartości bazy danych. Zalogowani mogą dokonać zakupu, a reszta może wyłącznie przeglądać ofertę. Mamy podstawowy system zamówień (zapisujemy historię dla zalogowanych użytkowników ).

# 1.Opis tabel i ich pól

**UserRoles**

Tabela zarządza poziomami dostępu użytkowników.

* **Role\_ID (INT, PK)**: Unikalny identyfikator roli.
* **Role\_Name (NVARCHAR(50))**: Nazwa roli (np. Gość, Klient, Admin).

**Users**

Przechowuje dane użytkowników systemu.

* **User\_ID (INT, PK)**: Unikalny identyfikator użytkownika.
* **Login (NVARCHAR(50))**: Login użytkownika.
* **Password (NVARCHAR(50))**: Hasło użytkownika zapisane z solą.
* **Role\_ID (INT, FK)**: Odniesienie do tabeli UserRoles w celu określenia roli.
* **Resfresh\_token (nvarchar(max))** Refresh\_token JWT przypisany do użytkownika
* **Expiration\_date (datetime)** Data wygaśnięcia (ważności) refresh tokenu
* **Salt (nvarchar(max))** Sól, z która zapisaliśmy hasło

**Clients**

Reprezentuje klientów biura podróży.

* **User\_ID (PK FK)**: Odniesienie do tabeli Users, gdy klient ma konto.
* **FirstName (NVARCHAR(50))**: Imię klienta.
* **LastName (NVARCHAR(80))**: Nazwisko klienta.
* **Address (NVARCHAR(100))**: Adres zamieszkania.
* **Email (NVARCHAR(50))**: Adres e-mail.

**TripCategories**

Tabela przechowuje kategorie wycieczek.

* **Category\_ID (INT, PK)**: Unikalny identyfikator kategorii.
* **Category\_Name (NVARCHAR(20))**: Nazwa kategorii (np. Objazdowe, Hotele, Jednodniowe).

**Trips**

Przechowuje dane o dostępnych wycieczkach.

* **Trip\_ID (INT, PK)**: Unikalny identyfikator wycieczki.
* **Name (NVARCHAR(30))**: Nazwa wycieczki.
* **Category\_ID (INT, FK)**: Odniesienie do tabeli TripCategories.
* **Country (NVARCHAR(20))**: Kraj, w którym odbywa się wycieczka.
* **Description:** Opis wycieczki
* **Image\_link:** Ściezka do pliku zdjęcia powiązanego z wycieczką (do danej lokalizacji**)**
* **Price decimal(10,2):** Cena za jedną osobę

**Orders**

Rejestruje zamówienia klientów.

* **Order\_ID (INT, PK)**: Unikalny identyfikator zamówienia.
* **Client\_ID (INT, FK)**: Odniesienie do tabeli Clients.
* **Schdeuled\_Trip\_ID (INT, FK)**: Odniesienie do tabeli Trips.
* **Issue\_Date (DATETIME)**: Data utworzenia zamówienia.
* **End\_Date (DATETIME)**: Data zakończenia zamówienia (np. termin wycieczki).
* **Total\_Price (DECIMAL(10, 2))**: Całkowity koszt zamówienia.
* **Number\_of\_participant**: Liczba uczestników zamówienia.

**Permissions**

Tabela definiuje uprawnienia dla różnych ról.

* **Permission\_ID (INT, PK)**: Unikalny identyfikator uprawnienia.
* **Permission\_Name (NVARCHAR(50))**: Nazwa uprawnienia (np. Przeglądanie wycieczek, Modyfikacja danych).

**RolePermissions**

Tabela wiąże uprawnienia z rolami.

* **Role\_ID (INT, FK)**: Odniesienie do tabeli UserRoles.
* **Permission\_ID (INT, FK)**: Odniesienie do tabeli Permissions.
* **PRIMARY KEY (Role\_ID, Permission\_ID)**: Klucz główny na poziomie tabeli.

**Trip\_Schedule**

**- Schedule\_ID (int,PK):** Klucz glowny tabeli

**- Trip\_ID (int, FK):** Id wycieczki, która ma zostać zaplanowana

**- Start\_date (datetime) –** data startu wycieczki

**- End\_date –** zakończenie wycieczki

**- Max\_participants –** Maksymalna ilość miejsc (które pozostały)

# 2. Funkcjonalności

**1. Uwierzytelnianie i Autoryzacja**

* **JWT Authentication**:
  + Logowanie użytkownika za pomocą tokenów JWT.
  + Generowanie i walidacja tokenów JWT z zawartością (np. login, rola użytkownika w payloadzie accessTokenu).
* **Role-Based Access Control (RBAC)**:
  + Dostęp do endpointów oparty na rolach użytkownika (Admin, Normal\_user).
  + Admin ma pełny dostęp do zarządzania systemem.
  + Zwykły użytkownik (Normal\_user) może tylko przeglądać i rezerwować wycieczki.
* **Odświeżanie tokenów**:
  + Zarządzanie tokenami odświeżania (RefreshToken) i ich terminem ważności.

**2. Zarządzanie Użytkownikami**

* **Rejestracja użytkowników**:
  + Tworzenie kont użytkowników z hashowaniem haseł i generowaniem soli.
* **Logowanie użytkowników**:
  + Sprawdzanie poprawności danych logowania i generowanie tokenów JWT.
* **Zarządzanie danymi użytkowników**:
  + Pobieranie danych użytkownika na podstawie loginu.
  + Przypisywanie ról użytkownikom (Admin, Normal\_user).

**3. Zarządzanie Wycieczkami**

* **Dodawanie nowych wycieczek**:
  + Administrator ma możliwość tworzenia nowych wycieczek
* **Edytowanie wycieczek**:
  + Możliwość aktualizacji danych o istniejących wycieczkach, np. zmiana ceny lub nazwy
* **Usuwanie wycieczek**:
  + Usuwanie wycieczek z bazy danych, a raczej wszystkich ich wystąpień w zaplanowanym harmonogramie, gdyż mamy zapisane wszystkie wycieczki celem wyświetlania ich użytkownikowi jako historii zamówień.
* **Przeglądanie wycieczek**:
  + Pobieranie listy dostępnych wycieczek z filtrowaniem po kategoriach, kraju, cenie minimalnej i maksymalnej oraz datach.

**4. Harmonogramy Wycieczek**

* **Zarządzanie harmonogramami**:
  + Tworzenie harmonogramów wycieczek z datą rozpoczęcia i zakończenia oraz limitem uczestników, który później jest modyfikowany o ilość osób w zamówieniu (zmniejszany o)

**5. Rezerwacje i Zamówienia**

* **Składanie zamówień**:
  + Zalogowany użytkownik może złożyć zamówienie wybierając dostępną wycieczkę i podając liczbę uczestników .
* **Historia zamówień użytkownika**:
  + Pobieranie listy zamówień przypisanych do zalogowanego użytkownika.
* **Kalkulacja cen**:
  + Automatyczne obliczanie całkowitej ceny na podstawie liczby uczestników.

# 3. Instrukcja uruchomienia

Baza danych to baza MSSQL. Backend bazuje na ASP.NET, EntityFramework (+ Biblioteki z Nuget’a pod SqlServer). Najpierw zatem musimy załadować do naszej bazy przez konsolę dbbuild. Aby uruchomić backend należy użyć komendy „dotnet run” lub otworzyć projekt poprzez plik z rozszerzeniem „.sln” w odpowiednim środowisku takim jak JetBrains Rider czy Visual Studio i tam użyć Run. Frontend napisany w React. Uruchomić przy pomocy komendy „npm start”.