Travel App

I. Descrierea funcționalităților aplicației

Travel App este o aplicație care se ocupă cu gestionarea unui itinerariu de vacanță. Aceasta ia în considerare detalii precum zbor, destinație, client, perioada sejurului și cazare. Aplicația poate fi potrivită pentru companiile de travel, aceasta punând la dispoziție următoarele funcționalități:

1.Clientul își poate alege destinația către care dorește să călătorească din lista de destinații pusă la dispoziție.

- Pentru a vedea lista de destinații se apelează endpoint-ul:
  - **GET:http://localhost:8080/destinations/list**

2. După parcurgerea listei de destinații și alegerea uneia, clientul își poate rezerva o vacanță pentru destinatia respectiva. Vacanta tocmai aleasa va fi setata in sesiunea curenta, iar clientul va putea adauga cazare, ziua de inceput si sfarsit sau zbor:

- **POST:http://localhost:8080/holidays/add**

***Se adauga o vacanta noua care va fi setata in sesiunea curenta("currentHoliday") si care va avea loc la destinatia selectata. Endpointul primeste 2 RequestParams (unul pentru numele destinatiei, iar altul pentru id-ul sesiunii).**

- **POST:http://localhost:8080/holidays/{id}/accommodation**

***Intr-un formular va fi trimis catre id-ul cazarii alese de client.**

- **GET:http://localhost:8080/holidays/currentHoliday/{sessionId}**

  ***Permite clientului sa finalizeze rezervarea vacantei, din acel moment poate sa ii mai modifice numai statusul "cancelled".**

- **POST: http://localhost:8080/holidays/{id}**

  ***Obs. Endpoint-ul primeste un request body de tip HolidayRequest ce are ca atribute numai startDate și endDate.**

- **GET: http://localhost:8080/holidays**

***Afiseaza lista vacantelor utilizatorului autentificat**

- **POST http://localhost:8080/holidays/cancellation/{id}**

  ***Utilizatorul poate anula o vacanta atata timp cat aceasta se va petrece in viitor sau este vacanta in curs de rezervare.**

3. După ce vacanta fost rezervată, utilizatorul poate continua și își poate selecta atât cazarea dorită cât și zborurile potrivite pentru el din listele afisate in paginile sugestive:

- Pentru vizualizarea tuturor cazărilor aflate în acea locație:
    - **GET:http://localhost:8080/accommodations/destination/{destinationName}**

***Obs. În momentul afișării cazărilor vor fi selectate doar acele cazări cu o capacitate mai mare decât 0. În momentul în care un client își alege o cazare, capacitatea acesteia scade cu o unitate, iar în momentul în care clientul renunță aceasta crește.**

- **Pentru a vedea zborurile paginate:**
    - **POST:http://localhost:8080/flights/flightsPAginated/{destinationName}**
- **Pentru signup (s-a folosit Spring Security, insa sistemul a fost personalizat, precum validarile si erorile):**
    - **POST:http://localhost:8080/client**
- **Pentru login (s-a folosit Spring Security, insa sistemul a fost personalizat, precum validarile si erorile):**
    - **POST:http://localhost:8080/**
- **Pentru a adauga o destinatie noua:**
    - **POST:http://localhost:8080/destinations/add**
- **Pentru afisarea tuturor cazarilor, sortate crescator in functie de pret:**
    - **GET:http://localhost:8080/accommodations/priceList/{destinationName}**
- Pentru vizualizarea tuturor zborurilor care au ca destinație sau punct de plecare numele dat ca path variable:
    - **GET: http://localhost:8080/flight/destinations/{destinationName}**
- Pentru a rezerva cazarea acelei vacanțe:
    - **PUT: http://localhost:8080/holidays/{id}/accommodation**
- Pentru a rezerva un zbor:
    - **PUT: http://localhost:8080/holidays/{id}/flight**

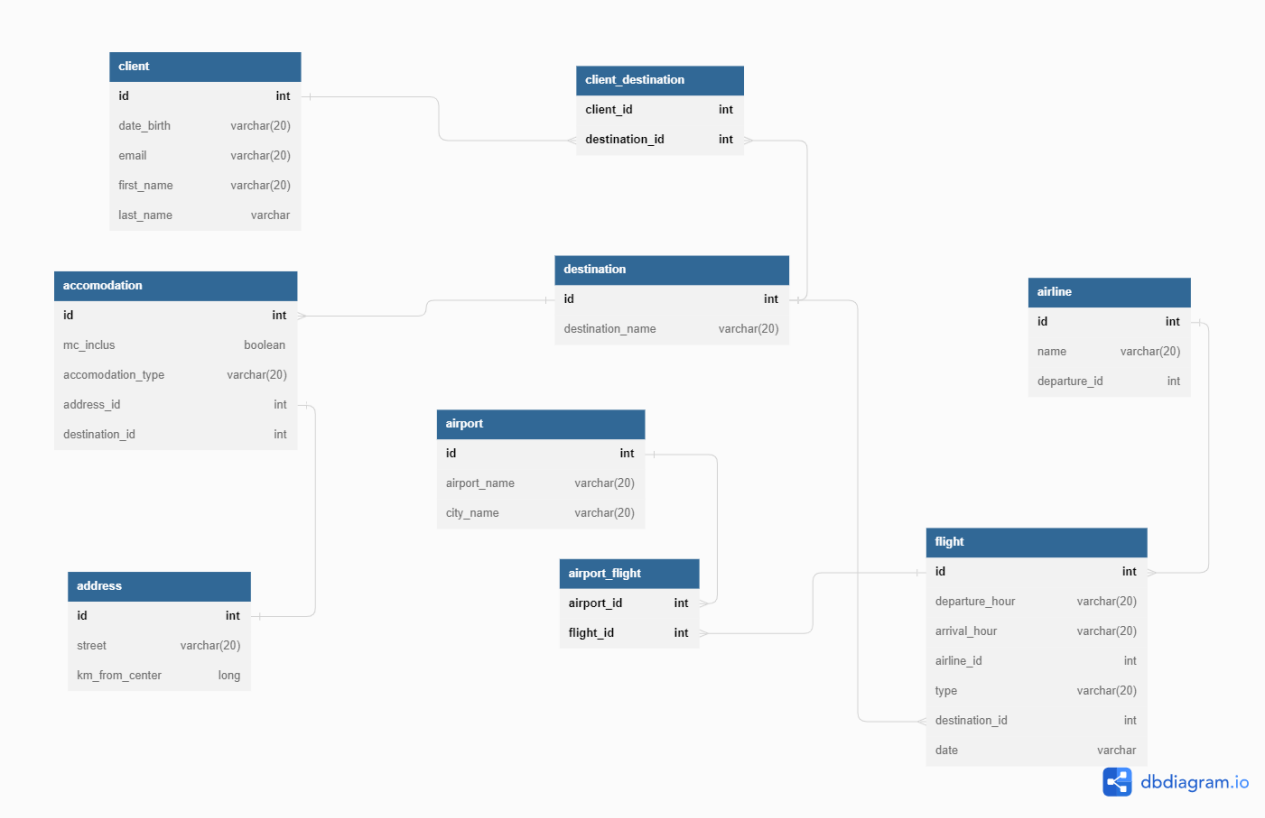
4. În cazul în care clientul se răzgândește în privința zborurilor sau cazărilor, acesta le poate șterge din vacanța rezervată.

- Ștergere zbor din lista de zboruri rezervate în funcție de id-ul din request param (pentru cazul în care zborul nu a fost rezervat se afișează un mesaj):
    - **DELETE:http://localhost:8080/holidays/{id}/flight**
- Ștergerea cazării din vacanța rezervată:
    - **DELETE: http://localhost:8080/holidays/{id}/accommodation/delete**

5. Dacă utilizatorul s-a răzgândit și își dorește anularea vacanței, poate face acest lucru accesând:

- **POST: http://localhost:8080/holidays/cancellation/{id}**

II. Diagrama relațională



III. Descrierea entităților și serviciilor

Entități

În vederea realizării aplicației s-au realizat 8 entități după cum urmează:

1. Accommodation
    1.1. Id
    1.2.AccommodationType
    1.3.Name
    1.4.PricePerNight
    1.5.CheckInHour
    1.6.CheckOutHour
    1.7.Capacity
    1.8.Destination
2. Address
    2.1.Accommodation
    2.2.KmFromCenter
    2.3.StreetAndNumber
3. Airport
    3.1.Id
    3.2.AirportName

3.3.Flights

4. Client
    4.1.Id
    4.2.FirstName
    4.3.LastName
    4.4.BirthDate
    4.5.Holidays
    4.6.AccountNotLocked
    4.7.CredentialsNotExpired
    4.8.AccountNotExpired
    4.9.Enabled
    4.10. Authorities
5. Destination
    5.1.Id
    5.2.DestinationName
    5.3.Accommodations
    5.4.Holidays
6. Flight
    6.1.Id
    6.2.Airline
    6.3.Destination
    6.4.DepartureHour
    6.5.ArrivalHour
    6.6.Date
    6.7.Price
7. Holiday (reprezinta relatia de many to may între destinație și client)
    7.1.Id
    7.2.Client
    7.3.Destination
    7.4.FirstDay
    7.5.EndDay
    7.6.Accommodation
    7.7.Flights
    7.8.IsCanceled
8. Authority

8.1.Id

8.2.Role

8.3.Users

Împreună cu:

1. ClientSignupRequest

1.1. Reprezinta DTO-ul corespunzator lui Client in vederea inregistrarii unui client nou. Contine atributele email si password.

2. ClientRequest
   2.1. Reprezinta DTO-ul corespunzator lui Client in vederea autentificarii. Contine atributele email si password.
3. HolidayRequest
   3.1. Reprezintă DTO-ul corespunzător lui Holiday în vederea rezervării unei vacanțe; conține numai atributele startDate și endDate.
4. AirlineType
   4.1. Reprezintă compania aeriană al unui zbor .(Wizzair, JetBlue etc.)
5. AccommodationType
   5.1. Reprezintă tipul de cazare.(Hotel,motel,cabană,camping etc.)

Servicii

Pentru fiecare entitate descrisă mai sus s-a realizat câte o interfață după modelul: NumeEntitateService care moștenește interfața CrudService<T>. Interfața CrudService are rolul de a reține toate metodele de tip CRUD de bază, pentru a evita scrierea lor în fiecare clasă. Interfața NumeEntitateService este implementata de clasa NumeEntitateServiceImpl care moștenește și clasa AbstractService<T>. Aceasta simulează clasele de service, având atributul de JpaRepository<T,Long> și implementând la rândul ei interfața CrudService<T> căreia îi suprascrie metodele. Astfel avem:

1. HolidayService și HolidayServiceImpl
2. AccommodationService și AccommodationServiceImpl
3. AddressService și AddressServiceImpl
4. ArportService și AirportServiceImpl
5. ClientService și ClientServiceImpl
6. DestinationService și DestinationServiceImpl
7. FlightService și FlightServiceImpl
8. HolidayService și HolidayServiceImpl

Repositories

Pentru fiecare entitate înregistrată în baza de date s-a creat o interfață NumeEntitateRepository care extinde JpaRepository<T,Long>. Fiecare interfață este injectată în clasele de NumeEntitateServiceImpl.

1. AccommodationRepository
2. AddressRepository
3. AirportRepository
4. ClientRepository
5. DestinationRepository
6. FlightRepository
7. HolidayRepository
8. AuthorityRepository

Pachetul exceptions

Conține excepții customizate care se apelează în funcție metodele care se apelează și care nu găsesc obiectele dorite în baza de date. Gestionarea excepțiilor se face în ControllerAdvice-ul definit și anume GeneralExceptionHandler.

1. HolidayAlreadyCancelledException
2. EntityNotFoundException
3. ClientNotRegisteredException
4. AccommodationNotMatchingDestException
5. ClientAlreadyRegistered
6. CustomNumberFormatException
7. HolidayCannotBeCancelled
8. InvalidSessionException

Templates

1. **AccessDenied.html - pentru acces interzis**
2. **AccommodationsSorted.html - unde se afiseaza cazarile sortate crescator in functie de destinatia selectata**
3. **DefaultException.html - template pentru exceptie default**
4. **Destination.html - template unde se afla toate destinatiile**
5. **FlightsPaginated.html - toate zborurile catre destinatia selectata, paginate**
6. **Holiday.html - se afiseaza toate vacantele clientului curent**
7. **Home.html - pagina de redirectionare dupa login/signup**
8. **InvalidSession.html - template pentru cazul de sesiune invalida**
9. **Login.html - template-ul de login**
10. **Signup – template-ul de sign-up**
11. **NumberFormatException.html - template-ul pentru eroarea de formatare**

Controllers

1. AccommodationController conține:
    1.1. Două request-uri de GET (getAllByDest și getById) care returnează toate cazările de la o anumită locație, respectiv cazarea găsită după id.
    1.2. Un request POST care adaugă p cazare.
2. ClientController conține:
    2.1. Un request POST care gestioneaza flow-ul de sign-up.
    2.2. GET care returnează un client după id.
    2.3. GET pentru afisarea paginii de Not Found.
3. DestinationController conține:
    3.1. O metodă POST de tip REST care adaugă o destinație.
    3.2. O metoda POST care adauga o destinatie si redirectioneaza clientul inapoi catre pagina de destinatii.
    3.3. Două metode GET de tip REST care returnează toate destinațiile, respectiv destinația găsită în funcție de id-ul dat.

3.4. O metoda GET care obtine lista cu toate destinatiile din baza de date si gestioneaza mesajele de eroare si succes.
3.5. O metodă PUT care modifică un obiect de tip Destination.
3.6. O metodă DELETE care șterge o destinație în funcție de id.

4. FlightController conține:
   4.1. O metoda POST care adaugă un zbor nou.
   4.2. Patru metode GET care returnează: toate zborurile dintr-o anumită perioadă, toate zborurile spre o anumită destinație, zborul căutat în funcție de id si toate zborurile spre o anumita destinatie, paginate.
5. HolidayController conține:
   5.1. O metodă POST care adaugă o vacanță nouă si o seteaza in sesiunea curenta.
   5.2. O metoda de tip POST care permite clientului sa seteze prima zi si ultima zi de vacanta, insa aceste zile trebuie sa fie in viitor, in caz contrar se afiseaza mesaje de eroare in pagina.
   5.3. O metoda de tip POST care anuleaza vacanta, daca vacanta este deja anulata sau se afla in trecut apare un mesaj de eroare.
   5.4. O metoda de tip POST care adauga cazarea in functie de id-ul dat.
   5.5. O metoda GET care returneaza vacantele clientului autentificat.
   5.6. Două metode DELETE care șterg cazarea sau zborul vacanței.
   5.7. O metoda GET care finalizeaza rezervarea vacantei.
6. LoginController care contine:
   6.1. Metode GET care returneaza paginile html corespunzatoare. (utilizate la redirect)
   6.2. Metoda POST pentru gestionarea login-ului.

Popularea bazei de date

Popularea bazei de date se realizează în clasa BootstrapClass si DataLoader care implementează interfața CommandLineRunner și suprascrie metoda run. La runtime se verifică dacă metodele findAll() returnează liste goale sau populate. În cazul în care listele sunt goale se creează și se salvează pe rand diferite obiecte în baza de date.

Proprietatile definite pentru baza de date (Postgres) se gasesc in application.properties, iar pentru baza de date In Memory se gasesc in application-h2.properties.