SPRC: Tema #2
Docker, containere și rezervare bilete avion

Termen de predare: 26 Noiembrie 2018

Titulari curs: *Florin Pop*

Responsabil Temă: **Florin Pop, e-mail: florin.pop@cs.pub.ro**

November 8, 2018

1 Obiectivele temei

Scopul acestei teme de casă este de a vă familiariza cu platforma Docker. Veți învăța cum să implementați orice aplicație care rulează în rețea prin construirea unui container simplu pentru client și server utilizând această platformă. Veți învăța să scrieți un conector la o bază de date într-o aplicație și să lucrați cu aceasta.

2 Specificații funcționale

O agenție de voiaj pune la dispoziția clienților un serviciu pentru rezervarea și cumpărarea de bilete de avion. Se cere implementarea unui astfel de serviciu și a unor componente ajutătoare, conform cu:

- serviciul, `AirplaneService`, oferă metode pentru aflarea rutei optime între două destinații, pentru rezervare și pentru cumpărare de bilete;
- evidența zborurilor, a rezervărilor și a biletelor cumpărate se ține într-o baza de date MySQL;
- pentru administrarea informațiilor legate de zboruri se va implementa o aplicație care se conectează la aceeași baza de date;
- pentru testare se va scrie un client pentru serviciul dezvoltat, cu o interfață de tip text ce va conține un meniu cu operațiile posibile. Clientul va avea că parametru în linia de comandă URL-ul serviciului.

2.1 Informații referitoare la zboruri

Pentru fiecare zbor se vor reține cel puțin următoarele informații:

- ID (de tip String);
- sursa și destinația (ambele de tip String);
- ora de plecare (de tip int, între 0 și 23) - se consideră că toate plecările sunt la ore ”fixe”;
- ziua plecării (de tip int, între 1 și 365) - se consideră zborurile dintr-un singur an; zborurile din zile diferite, de la aceeași ora, vor avea ID-uri diferite;
- durata zborului (de tip int) - se consideră că este un număr întreg de ore;
- numărul de locuri disponibile.

2.2 Interfața serviciului

Serviciul va oferi următoarele metode (exemplele sunt oferite orientativ în Java):

```
String[] getOptimalRoute(String source, String dest,
                         int maxFlights, int departureDay);
```

unde: `source` și `dest` au semnificațiile evidente, `maxFlights` este numărul maxim de zboruri pe care îl poate conține ruta, iar `departureDay` este un număr între 1 și 365, reprezentând ziua din an în care se dorește să aibă loc plecarea (se consideră doar ani având 365 de zile).

Se cere găsirea unei rute de timp minim, ținând cont atât de duratele zborurilor cât și de duratele escalelor; de asemenea ruta trebuie să nu aibă mai mult de `maxFlights` zboruri. În plus, orice pasager dorește să fie acasă cu familia de Anul Nou, așa că nu se admit programe de zbor care încep într-un an și se termină în celălalt (în evidența companiei sunt doar zboruri din anul curent). Precizări suplimentare referitoare la ruta:

- NU se ține cont de diferențele de fus orar între diferite orașe (se presupune că toate au aceeași ora);
- o ruta poate dura mai mult de o zi.

Tabloul de șiruri întors de funcție va conține:

- pe prima poziție: un șir cu informații "human-readable" despre ruta (pentru fiecare zbor, ID-ul, sursă, destinația, ora de plecare, durata etc.);
- pe următoarele poziții: ID-urile fiecărui zbor din ruta, în ordinea în sursă ¿ destinație.

```
String bookTicket(String[] flightIDs);
```

Prin această metodă se face o rezervare pentru o anumită ruta. Parametrul `flightIDs` contine ID-urile zborurilor din ruta, care pot fi obținute prin funcția `getOptimalRoute(...)`. Metoda întoarce un ID al rezervarii, sau un șir vid în cazul în care nu s-a putut face rezervarea (nu mai sunt locuri disponibile, unul din zboruri a fost anulat etc.).

```
String buyTicket(String reservationID, String creditCardInformation);
```

Prin această metodă se cumpără un bilet pentru o ruta pentru care s-a făcut deja rezervare. Ca parametru se dă ID-ul rezervarii, iar rezultatul întors de funcție este un boarding pass conținând toate informațiile despre zboruri, sau șirul vid în caz de eroare.

2.3 Aplicația de administrare

Administrarea datelor din baza de date (evidența zborurilor) se va face printr-o aplicație care se conectează la baza de date. Tabelele sunt inițial goale, date despre zboruri putând fi adăugate în ele prin această aplicație de administrare (cu o interfață de tip text). Operațiile de administrare sunt:

```
int adaugare_Zbor(String source, String dest,
                  int departureDay, int departureHour,
                  int duration, int numberOfSeats, String flightID);
```

adaugă un zbor în ziua `departureDay` a anului (între 1 și 365), care decolează la ora `departureHour` (toate zborurile decolează la ora fixă, între 0 și 23), are o durată de `duration` ore, o capacitate de `numberOfSeats` locuri și ID-ul `flightID`. Un zbor poate pleca într-o zi și ajunge la destinație într-o altă zi (poate chiar peste mai multe zile). Se garantează că toate ID-urile zborurilor vor fi unice.

```
void anulare_Zbor (String flightID);
```

funcția va șterge informația referitoare la zborul identificat prin `flightID` - (zborul fiind anulat). Conform politicii standard a companiilor aeriene, se vor putea rezervă maxim 110% din locurile disponibile pentru un zbor (această politică se numește overbooking și se bazează pe faptul că nu toți cei care rezervă un bilet vor și cumpără biletul respectiv). Așadar, atunci când un client dorește să rezerve bilet pentru o serie de zboruri, va trebui ca numărul de locuri rezervate pentru fiecare zbor să nu fi atins valoarea maximă. De asemenea, este obligatoriu ca operația de rezervare pentru mai multe zboruri să se execute atomic (adică ori se rezervă un bilet pentru toate zborurile dorite, ori pentru nici unul). Dacă rezervarea biletului reușește, se întoarce clientului un `reservationID` (unic).

La un moment dat, ulterior rezervării, unii clienți vor dori să cumpere biletul. Pentru această vor apela metoda `buyTicket` a serviciului, trimițând `reservationID`-ul, precum și informații despre cardul de credit (reprezentând modalitatea standard de plata).

3 Cerințe de implementare

Aplicația se va realiza în plaforma Docker (link). Pentru a instala și a începe lucrul cu Docker urmați acest link: Get Started with Dockers.

Veți construi trei containere folosind Docker, câte unul pentru fiecare componentă: serviciul, clientul, aplicația de administrare. Aceste containere pot fi construite pornind de la o imagine oficială de bază. Imaginile oficiale sunt disponibile la următorul link. Porniți de la imaginea potrivită.

Veți crea o stivă de servicii în Docker și să rulați containerele într-o rețea definită de utilizator. Containerele ar trebui să ruleze cele trei componente ale aplicației în mod implicit (adică pe comanda de rulare).

Nu sunt impuse restricții cu privire la ce limbaj de programare folosiți pentru a implementa cele trei componente ale serviciului.

Pentru a lucra cu MySQL în Docker, urmați indicațiile oficiale.

4 Modalitatea de notare

- Construirea și executația containerului serverului - **40 de puncte**
- Construirea și executația containerului client - **10 puncte**
- Construirea și executația containerului aplicației de administrare - **10 puncte**
- Comunicarea corectă între containere - **15 puncte**
- Scripturi pentru a rula containerele - **15 puncte**
- Fișier README, claritate cod - **10 puncte**