Opis

Napraviti aplikaciju koja omogucava rezervaciju i administraciju avio-karata.

Postoje dve vrste korisnika, na osnovu razlicitih privilegija pruzicemo korisniku drugacije mogucnosti:

- **Obican korisnik** koji moze da pretrazuje i rezervise karte i manipulise rezervacijama.
- **Administrator** koji moze da kreira nove korisnike i vrsi CRUD operacije nad kartama i kompanijama.

Sistem treba da se sastoji iz dva dela:

- Za frontend mozete koristiti **jQuerry** ili **Vue.js.**
- Za backend **Jax.rs** ili **SparkJava.**

Entiteti

Korisnik:

- ID:Integer - mora biti jedinstven, ne prikazuje se na frontendu
- Username:String - mora biti jedinstven
- Password :String - mora imati bar 6 karaktera i sadrzati slova i brojeve
- Tip-korisnika:Enum - User ili Admin
- Bookings:List - lista rezervacija (samo kod **obicnog** korisnika)

Avionska kompanija:

- ID:Integer - mora biti jedinstven, ne prikazuje se na frontendu
- Name: String - mora biti jedinstveno

Grad:

- ID:Integer - mora biti jedinstven, ne prikazuje se na frontendu
- Name:String - ime grada, mora biti jedinstveno

Avionska karta:

- ID:Integer - mora biti jedinstven, ne prikazuje se na frontendu
- Kompanija:Kompanija - kompanija za koju je vezana karta
- One-way: Boolean - da li je karta u jednom pravcu ili povratna
- Depart: Date - datum polaska
- Return: Date - datum povratka (samo kod povratnih karata)
- Flight: Integer - ID leta za koji je karta vezana
- Count: Long - broj (>=0) dostupnih karata

Let:

- ID:Integer - mora biti jedinstven, ne prikazuje se na frontendu
- Ticket: List - lista karta za ovaj let
- Origin: Grad - grad iz kog se polece
- Destination: Grad - grad u koji se putuje

Rezervacija:

- ID:Integer - mora biti jedinstven, ne prikazuje se na frontendu
- IsAvailable: Boolean - da li je rezervacija dostupna ili je istekla (ako je prosao datum polaska)
- Flight: Let - rezervisani let
- Ticket: Karta - rezervisana karta
- User: Korisnik - korisnik koji je rezervisao kartu

Parametri koji su u entitetima naznaceni da mogu da budu prisutni samo kod jednog korisnika su opcioni i mogu da budu null u bazi.
Veze ka drugim entitetima mogu biti samo ID, kao foreign key.

Perzistencija

Sve entitete cuvati u bazi ili datotekama.

Ukoliko se radi sa bazom podataka, potrebno je implementirati Optimistic lock mehanizam zastite od konkurentne izmene.
Dakle, dodati u svaku tabelu jos jednu kolonu, sa verzijom ili timestamp-om.
Pre svakog upisa treba proveriti da li je verzija ista kao i kada smo preuzeli iz baze.
Ukoliko nije, neko drugi je vec promenio taj zapis i treba odustati od promene i prijaviti gresku.

Studenti koji se odluce za snimanje u datoteku, ne moraju da vode racuna o ovom problemu, ali je potrebno da obezbede konkurentan pristup datoteci.

Autentikacija i autorizacija

Backend mora da bude stateless - ne treba da zna koji je korisnik ulogovan, sto znaci da mora da postoji **autorizacija** zahteva.
Za svaki zahtev, server proverava da li korisnik koji ga je poslao ima pravo da pristupi tom endpointu.
Autentikaciju i autorizaciju implementirati pomocu JWT tokena.
Token moze da traje beskonacno dugo i nije potrebno implementirati rok trajanja.

Login stranica

Stranica za login treba da ima polja **username** i **password** i dugme za login.
Ukoliko korisnik sa takvim kredencijalima postoji, potrebno je proslediti ga na glavnu stranicu. U suprotnom ostati na istoj stranici i obavestiti klijenta o greski.

Glavna stranica

U gornjem desnom uglu treba da stoje **Username**, **tip** korisnika, ikonica koja predstavlja **rezervisane karte** sa indikatorom broja karata (samo za obicne korisnike) i **logout** dugme.

Klik na ikonicu vodi na stranicu sa rezervacijama za tog korisnika.

Ukoliko je korisnik:

1. User
    - Forma za pretragu po:
        - Mestu polaska
        - Destinaciji
        - Datumu polaska i povratka (ukljucujuci uneseni datum)

            Ako se navede samo datum polaska, prikazati sve karte nakon tog datuma bez obzira na datum povratka.
            Slicno i ako je naveden samo povratni datum.
            Ako su navedena oba datuma, prikazati sve karte sa vremenima polazaka i dolazaka izmedju navedenih datuma.

2. Admin
    - Forma za registraciju novog korisnika sa poljima za **username**, **password** i **tip korisnika**
    - Forma za kreiranje nove karte sa svim potrebnim poljima
      Lista mogucih kompanija treba da bude u combobox-u. Nju cete popuniti podacima koje dobijete sa servera.
      Implementirati combobox i za listu letova.

Neki mehanizam da korisnik odabere da li ce mu se prikazati:
- Samo karte u jednom pravcu,
- Samo karte u oba pravca,
- Sve karte.

Tabela karata

Nalazi se na glavnoj stranici i sadrzi polja:

- One-way - da li je karta u jednom pravcu
- Origin - grad iz kog se polece
- Destination - grad u koji se putuje
- Depart - datum polaska
- Return - datum povratka (samo kod povratnih karata)
- Company - naziv avio-kompanije koji je link ka stranici avio-kompanije
- Count - broj dostupnih karata

Tabela mora imati paginaciju.

Paginacija je funkcionalnost gde ne prikazujemo sve rezultate na jednom mestu, vec ih rasporedjujemo po “stranicama”. Kroz stranice se prolazi sa dugmetom za prethodnu ili za sledecu stranu (ako postoje). Ako na BE postoji 25 karata, imacemo tacno 3 puta sansu da kliknemo dugme za sledecu stranicu. Treci put ce se prikazati samo 5 karata ako svaka stranica ima 10 rezultata prikazanih.

U nasem slucaju imacemo tabelu karata sa dva dugmeta ispod za sledece i prethodne rezultate. Tabela ce na osnovu toga da prikazuje odredjenih 10 karata. Svaki put kad se klikne neko od dugmica, potrebno je pozvati BE i popuniti tabelu sa odgovorajucih 10 karata. Na FE ne sme biti ucitano vise od 10 karata sto znaci da se paginacija vrsi na BE.

Da biste postigli ovakav efekat predlazem da napravite pomocnu klasu PaginationResponse koja bi imala jedan atribut tipa Object (tu biste smestili vas model) i jedan atribut tipa PageInfo koji je takodje model koji poseduje informacije o tome koliko ima stranica ukupno, koja je stranica u pitanju, koliko podataka ima po stranici.

Ako je korisnik **administrator** treba da postoji dugme za brisanje karte i dugme koje vodi na stranicu za izmenu karte.

Obican korisnik ima dugme u tabeli za rezervaciju karte, ukoliko je dostupna.
Broj dostupnih karata ne moze biti manji od 0.
Rezervacija karte dekrementira broj karata.

Kada broj karata padne na 0, obrisati kartu.

Stranica za izmenu karte

U gornjem desnom uglu treba da stoje **Username**, **tip** korisnika, ikonica koja predstavlja **rezervisane karte** sa indikatorom broja karata (samo za obicne korisnike) i **logout** dugme.

Sadrzi:

- Formu za izmenu karte sa svim potrebnim poljima, dugme za slanje i za ponistavanje operacije.

Stranica avio-kompanije

U gornjem desnom uglu treba da stoje **Username**, **tip** korisnika, ikonica koja predstavlja **rezervisane karte** sa indikatorom broja karata (samo za obicne korisnike) i **logout** dugme.

Sadrzi:

- Naziv kompanije
- Tabelu svih karata za tu kompaniju (uz mogucnost brisanja i izmene za **administratora** i rezervisanje za **user-a**)
- Dugme za brisanje kompanije (Ako je korisnik **administrator**)
  Brisanjem kompanije brisu se i sve karte vezane za obrisanu kompaniju
  Kao i obrisane karte iz svih letova.
- Forma za izmenu naziva kompanije (Ako je korisnik **administrator**)
  Izmena naziva kompanije povlaci sa sobom izmenu u svim entitetima povezanim sa tom kompanijom.
- Formu za kreiranje kompanije samo sa poljem za naziv (Ako je korisnik **administrator**)

Stranica rezervacija

U gornjem desnom uglu treba da stoje **Username**, **tip** korisnika, ikonica koja predstavlja **rezervisane karte** sa indikatorom broja karata (samo za obicne korisnike) i **logout** dugme.

Na nju moze da ode samo obican korisnik.

Sadrzi:

- Tabelu svih rezervacija u kojoj se nalaze:
  - Kolone iz tabele karata
  - Kolona da li je rezervacija istekla
  - Dugme za brisanje rezervacije, rezervacija se moze obrisati **do 24 sata pre datuma polaska**

Brisanjem rezervacije, treba osveziti listu rezervacija tog korisnika u bazi.

Error handling

Error handling i validacija su obavezni i svaka funkcija mora da prijavi gresku ukoliko ulazni parametri ili interno stanje ne dozvoljavaju izvrsenje operacije.
Validacija na **backendu** je **obavezna**!

- Voditi racuna za unose koji su **prazne vrednosti.**
- Mesto polaska i destinacija ne mogu biti isto mesto.
- Vreme polaska ne moze biti nakon vrema povratka.
- Korisnika obavestiti kad god napravi gresku.
- Ne moze da se registruje korisnik sa username-om koji vec postoji (slicno i za druge entitete koji moraju imati jedinstvena polja).

Napomene:

Ako radite sa fajlom, obratite paznju da update/ delete operacije nad kartama/ kompanijama uticu na entitete sa kojima su u relaciji.
Npr. izmena naziva kompanije, zahteva da se izmeni polje kompanija u svim kartama.

Ukoliko koristite Google Chrome, on kesira stranice pa se moze desiti da ste napravili izmenu a da stranica i dalje izgleda isto.
Zato kada napravite izmenu, otvorite stranicu u incognito prozoru.

Sve nazive (User, Admin, One-Way, Ticket, Booking...) mozete nazvati proizvoljnim imenima u vasim projektima.

Napravite odredjeni broj predefinisanih karata/ kompanija/ letova/ gradova/ korisnika radi lakseg testiranja i demonstracije.

CORS problemi

- Probajte da radite SparkJava projekat, umesto Jersey.

- U AuthServisu parsiranje tokena stavite u try/catch blok, npr.

```
try {
        Jws<Claims> claims =
Jwts.parser().setSigningKey(generalKey()).parseClaimsJws(jwt);
      }catch (Exception e) {
         return false;
      }
```

- Ako se problem i dalje javlja, dodajte neki dependency za CORS umesto ovog sa materijala.
  Na primer ovaj dependency:

  ```
  <dependency>
   <groupId>com.thetransactioncompany</groupId>
   <artifactId>cors-filter</artifactId>
   <version>2.5</version>
  </dependency>
  ```

  a zatim u web.xml dodajte:

  ```
  <filter>
   <filter-name>CORS</filter-name>
   <filter-class>com.thetransactioncompany.cors.CORSFilter</filter-class>
  </filter>
  <filter-mapping>
   <filter-name>CORS</filter-name>
   <url-pattern>/*</url-pattern>
  </filter-mapping>
  ```

Predaja

Rok za predaju projekta za **avgustovski** ispitni rok je 16.08. u ponoc.

Projekat nosi 40 poena, a minimum je potrebno 20 da se polozi predmet.
Za uslov je potrebno implementirati sve do isprekidane linije.

Na Git, Drive, .zip je dovoljno poslati samo kod, bez .jar, node_modules, .idea i drugih nepotrebnih foldera i fajlova koji su u vezi sa framework-om, bibliotekom ili okruzenjem koje koristite. Pogotovo na Git nije ni preporucljivo komitovati te foldere.

Domaci se salje na **amicic@raf.rs**
Naslov imejla i treba da glasi: Projekat <Ime> <Prezime> <Indeks>
Telo imejla treba da bude jedno od sledecih:

- Zip
- Link ka zip-u
- Link ka Drive-u
- Link ka GitHub/GitLab repou

  Ako saljete Git, stavite da je repozitorijum private i dodajte me kao collaborator-a
  https://github.com/alemicic