Flight Reservation Application

- Spring boot başladıktan sonra test edilebilirliğin kolaylaşması için uygulama DummyDataInitializer sınıfı içerisinde @PostConstruct anatasyonu ile veritabanına dummy data koymaktadır.

- Bu veriler yardımıyla proje içerisinde bulunan postman collection ile uygulama test edilebilir.

- Uygulamada Client'a dönen hata response larının anlamlı olması için Custom Exception sınıfları kullanılmıştır ve bu sınıflar üzerinden servislere cevap vermek için @ControllerAdvice anotasyonu ile GlobalExceptionHandler sınıfı oluşturulmuştur.

- Uygulama çalıştırılmadan önce docker-compose.yaml içerisinde bulunan postgresql servisi çalıştırılmalıdır. Kullanılan bilgisayarda docker-compose yoksa https://docs.docker.com/compose/install/ adresinden bilgisayara günlenmelidir.
    - Terminal yardımıyla uygulama paketi içerisinde bulunan docker-compose.yaml dosya yoluna gidilir ve aşağıdaki komut çalıştırılır.
        - docker-compose up
- buySeat api'ı Mockito'dan faydalanarak test edilmiştir.

Services

addFlight

- Bu api uçuş eklemek için kullanılmaktadır.
- Response'da dönen bilgilerden bazıları öteki api testlerinde kullanılacaktır.
- Bir uçuş ilk kayıt olduğunda 'ON_TIME durumunu alır.

deleteFlight

- Bu api uçuş iptal etmek için kullanılmaktadır.
- addFlight response'unda bulunan flightNumber bu api'da path variable olarak kullanılır.
- Uçuşların durumunu 'CANCALLED' olarak günceller.

updateFlight

- Bu api uçuş bilgilerini güncellemek için kullanılır.
- addFlight response'unda bulunan flightNumber bu api'da path variable olarak kullanılır.
- RequestBody olarak UpdateFlightRequestDto nesnesini alır.
- Burada ilgili servise gönderilen planlanan zaman var olan olanlanmış zamandan ileri yada geri olma durumuna göre flight situation 'DELAYED' yada 'ADVANCE' durumlarını alır.

addSeatToFlight

- Bu api varolan uçuşlarda bulunan uçağa koltuk eklemek için kullanılır.
- Bir uçakta koltuk sınırına ulaşmış ise daha fazla koltuk uçağa eklenmeyecektir.
- addFlight response'unda bulunan flightNumber bu api'da path variable olarak kullanılır.
- RequestBody olarak AddSeatRequestDto Nesnesini liste olarak alır.
- Response'da dönen bilgilerden bazıları öteki api testlerinde kullanılacaktır.
- Bu servis ile eklenmiş koltuklar 'PURCHASABLE' durumunu alır.

removeSeat

- Bu api varolan bir uçuşta bulunan uçağın koltuklarını kaldırmak için kullanılmaktadır.
- addSeatToFlight response'unda bulunan koltuk numaraları (seats listesi içerisinde bulunan koltukların number alanları) bu serviste path variable olarak kullanılır.
- addFlight response'unda bulunan flightNumber bu api'da path variable olarak kullanılır.
- addSeatToFlight response'unda bulunan number bilgisi bu serviste path variable olarak kullanılır.
- Bu servis tetiklendikten sonra ilgili koltuk 'UN_PURCHASABLE' durumunu alır.

updateSeat

- Bu api varolan bir uçuşta bulunan uçağın koltuk bilgilerini güncellemek için kullanılır.
- addFlight response'unda bulunan flightNumber bu api'da path variable olarak kullanılır.
- addSeatToFlight response'unda bulunan number bilgisi bu serviste path variable olarak kullanılır.
- RequesyBody olarak UpdateSeatRequestDto nesnesini alır.
- koltukların durumu 'BROKEN', 'PURCHASABLE', 'UN_PURCHASABLE', 'SOLD' olarak güncellenebilir.

getFlightDetails

- Bu api var uçuşların detaylarını görmek için kullanılır.
- addFlight response'unda bulunan flightNumber bu api'da path variable olarak kullanılır.

buySeat

- Bu api client tarafından koltuk satın alınması için oluşturulmuştur.
-
- Koltuk iki farklı kişi tarafından satın alınması engellemek için buySeat metodunda @Transactional annotasyonu ile ilgili veride bir işlem olduğunda hibernate tarafından farklı bir işlem görmemesi kilitlenmiştir. Aynı kayıt, aynı anda farklı kişiler tarafından değiştirilmek istenirse GlobalExceptionHandler sınıfı içerisinde handleSeatAlreadySoldException metodu ile client'a anlamlı bir response gönderilir.