

# PROJE SONU RAPORU:

## THY Biletleme ve Rezervasyon Sistemi

---

### Grup Üyeleri:

Kaan Baykal 221101028  
Beril Aydın 231101002  
Anıl Özişler 211101081

### Proje GitHub Linki:

[https://github.com/Berilay6/THY\\_Ticketing\\_App/tree/main](https://github.com/Berilay6/THY_Ticketing_App/tree/main)

# 1. Gerçek Dünya Probleminin Tanımı

Havacılık sektöründe binlerce uçuşun, milyonlarca koltuğun ve anlık değişen operasyonel süreçlerin manuel veya izole sistemlerle yönetilmesi imkansızdır. Yolcuların saniyeler içinde biletleme yapma bekłtisi ve havayolu şirketlerinin uçak/bakım envanterini hatalı yönetme zorunluluğu, yüksek performanslı ve bütünsel bir veritabanı çözümünü zorunlu kılmaktadır.

Bu proje, gerçek dünyadaki bu karmaşık yapıyı simüle ederek şu problemleri çözmeyi hedefler:

- Çifte Rezervasyon (Double Booking) Riski:** Aynı koltuğun aynı anda iki farklı yolcuya satılmasının engellenmesi.
- Operasyonel Sürekliklilik:** Uçakların bakıma alınması veya sefer iptalleri durumunda bilet iade gibi süreçlerinin yönetimi.
- Veri Bütünlüğü:** Ödeme alındığı anda biletin oluşması, aksi durumda işlemin tamamen geri alınması (Atomicity).

Projenin akademik bağlamda seçilme nedeni ise; yukarıda belirtilen bu gerçek hayat problemlerinin, Veritabanı Yönetim Sistemleri dersinin temel kazanımları olan **Eşzamanlılık Kontrolü (Concurrency), Transaction Yönetimi ve İlişkisel Normalizasyon** prensiplerini uygulamak için uygun senaryoyu sunmasıdır.

## 2. Gereksinim Analizi

Projenin geliştirilme sürecinde hedeflenen sistem gereksinimleri; kullanıcı etkileşimleri, operasyonel yönetim ve sistemsel kısıtlar olmak üzere üç ana başlıkta analiz edilmiştir.

### 2.1. Fonksiyonel Gereksinimler

Sistem, "User" ve "Admin" olmak üzere iki farklı aktör için aşağıdaki fonksiyonları sağlamalıdır:

#### User Modülü:

- Dinamik Uçuş Arama:** Kullanıcılar, kalkış noktası, varış noktası ve tarih kriterlerine göre veritabanındaki aktif uçuşları filtreleyebilmelidir.
- Rezervasyon ve Biletleme (E-Ticket):** Kullanıcı, seçtiği uçuş için ödeme işlemini tamamlamalıdır. İşlem sonucunda sistemde ticket oluşturulmalıdır.

- **Koltuk Seçimi:** Sistem, biletleme sürecinde uygun koltukları kullanıcıya sunmalıdır.
- **Ödeme Sistemi:** Kullanıcılar kredi kartı bilgileri ile ödeme yapabilmelidir. Bir ödeme işlemi, birden fazla biletin ödemesini tek seferde kapsayabilmelidir.
- **Geçmiş Uçuşlar:** Kullanıcılar, kendi hesapları üzerinden geçmişte yaptıkları seyahatleri ve gelecek rezervasyonlarını "My flights" ekranında listeleyebilmelidir.
- **Profil Yönetimi:** Kullanıcılar kişisel bilgilerini güncelleyebilmeli ve sık kullandıkları kredi kartlarını sistemde saklayabilmelidir.

### **Admin Modülü:**

- **Envanter Yönetimi:** Yönetici, sisteme yeni Havalimanı, Uçak ve Sefer (Flight) ekleyebilmelidir.
- **Otomasyon:** Sistem, bir uçak eklendiğinde uçağın fiziksel koltuklarını ve uçuş eklendiğinde uçuşa ait satılacak koltukları otomatik olarak veri tabanına eklemelidir.
- **Uçuş Durum Yönetimi:** Operasyonel gereklilikler durumunda uçuşlar "Cancelled" statüsüne çekilebilir.
- **İade Süreci:** İptal edilen uçuşlarda, sistem o uçuşa ait satılmış tüm biletleri tespit etmeli ve bilet statülerini otomatik olarak "İade Edildi" konumuna getirerek para iadesini gerçekleştirmelidir.
- **Filo Yönetimi:** Arızalanan uçaklar "Maintenance" moduna alınabilmeli ve bu uçaklara atanmış gelecekteki seferler boş çıkarılarak (Unassigned) yönetici uyarılmalıdır.

## **2.2. Veri ve İşlem Gereksinimleri (Database Constraints)**

Veritabanı tasarıımı, gerçek hayat senaryolarındaki tutarlılığı sağlamak adına aşağıdaki kurallara uymalıdır:

- **Atomik İşlem:** Bilet satın alma süreci (Ödeme Alma > Bilet Oluşturma > Koltuk Kapasitesini Düşürme) tek bir **Transaction** olarak işlenmelidir. Herhangi bir adımda hata olursa (örneğin yetersiz bakiye), tüm işlem geri alınmalı (Rollback) ve veritabanında tutarsız kayıt oluşmamalıdır.
- **Veri Bütünlüğü:** Silme işlemleri (Delete), veri kaybını önlemek amacıyla kısıtlanmalıdır. Bunun yerine "Soft Delete" veya "Durum Değişikliği (Status Change)" yöntemi uygulanarak geçmiş verilerin raporlanabilir kalması sağlanmalıdır.

## 2.3. Fonksiyonel Olmayan Gereksinimler

- **Güvenlik:** Kullanıcı şifreleri veritabanında açık metin olarak değil, BCrypt algoritması ile hashlenerek saklanmalıdır. API erişimleri JWT(JSON Web Token) ile yetkilendirilmelidir.
- **Kullanıcı Dostu Arayüz:** Sistem, kullanıcıların kolayca etkileşim kurabileceği, sezgisel ve temiz bir arayüze sahip olmalıdır. Yapılan tasarım değişiklikleri (giriş, güncelleme, silme) kullanıcı deneyimini hızlıca yansıtmalıdır.
- **Performans:** Uçuş arama gibi yoğun sorgular, veritabanı indeksleri (Indexing) ile optimize edilmeli ve sorgu süreleri minimize edilmelidir.

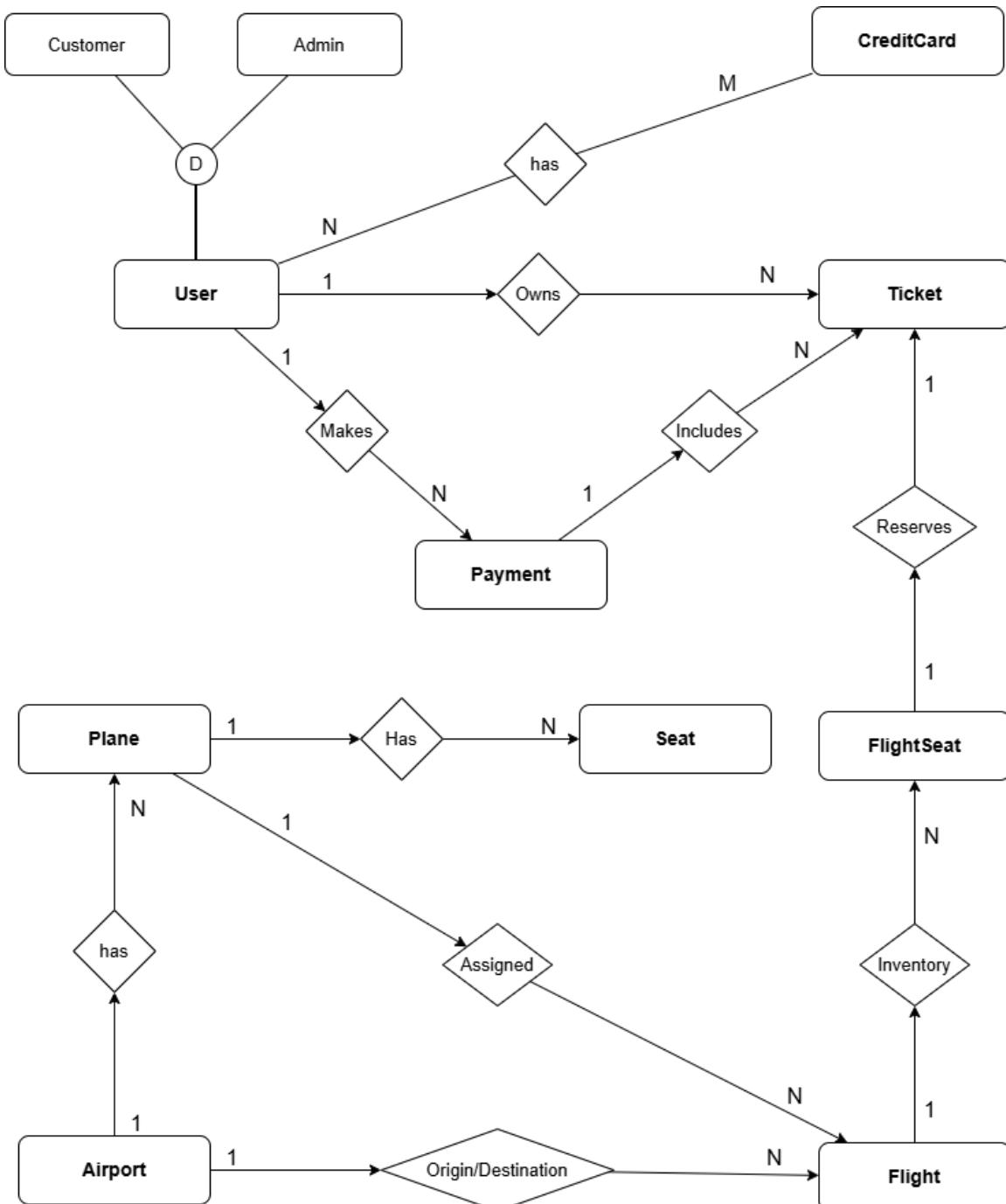
## 3. Kavramsal Tasarım (EER Diyagramı)

Bu bölümde, sistemin veritabanı yapısını oluşturan varlıklar (Entities), bu varlıkların öznitelikleri (Attributes) ve aralarındaki ilişkilerin (Relationships) kardinalite analizleri detaylandırılmıştır. Tasarım sürecinde BCNF normalizasyon kuralları gözetilmiş ve veri tutarlığını garanti altına alacak ilişki tipleri seçilmiştir

### 3.1. Varlık İlişkileri

- **Airport - Plane (1:N):** Bir havalimanında birden fazla uçak bulunabilir.
- **Airport - Flight (1:N):** Bir havalimanından birçok uçuş kalkabilir (Origin) veya inebilir (Destination).
- **Plane - Flight (1:N):** Bir uçak farklı zamanlarda birçok uçuş gerçekleştirebilir.
- **Plane - Seat (1:N):** Bir uçağın fiziksel olarak birçok koltuğu vardır.
- **Flight - FlightSeat (1:N):** Bir uçuşta satılacak birçok koltuk vardır.
- **User - Payment (1:N):** Bir kullanıcı birden fazla ödeme yapabilir.
- **User - Ticket (1:N):** Bir kullanıcının birden fazla biletini olabilir.
- **Payment - Ticket (1:N):** Bir ödeme işleminde birden fazla bilet satın alınabilir.
- **FlightSeat - Ticket (1:1):** Belirli bir uçuşun belirli bir koltuğu sadece tek bir aktif bilette ait olabilir.
- **User - CreditCard (M:N):** Bir kullanıcının çok kartı olabilir, bir kart birden çok kullanıcıkla ekli olabilir.

### 3.2 EER Diyagramı



## 4. Mantıksal Tasarım ve Şema Diyagramları

Bu bölümde, kavramsal tasarımin fiziksel veritabanı modeline dönüştürülmüş hali sunulmaktadır.

### 4.1. İlişkisel Şema (Relational Schema)

Tablolarımızın mantıksal yapısı, birincil anahtarlar (PK), birleşik anahtarlar (Composite PK) ve yabancı anahtarlar (FK) ile aşağıda tanımlanmıştır:

- **User:** (user\_id [PK], email [Unique], phone\_num [Unique], password, type, version)
- **Airport:** (airport\_id [PK], iata\_code [Unique], city, country, timezone)
- **Plane:** (plane\_id [PK], model\_type, status, airport\_id [FK])
- **Seat:** (plane\_id [PK, FK], seat\_number [PK], type, status)
- **Flight:** (flight\_id [PK], origin\_id [FK], destination\_id [FK], plane\_id [FK], departure\_time)
- **FlightSeat:** (flight\_id [PK, FK], seat\_number [PK], availability, price, version)
- **Payment:** (payment\_id [PK], user\_id [FK], total\_amount, status)
- **Ticket:** (ticket\_id [PK], flight\_id [FK], seat\_number [FK], payment\_id [FK], user\_id [FK])
- **CreditCard:** (card\_id [PK])
- **User\_CreditCards:** (user\_id[PK], card\_id[PK])

### 4.2. Normalizasyon (BCNF)

Veritabanı tasarımımızda veri tutarlığını sağlamak, güncelleme anomalilerini önlemek ve disk alanını verimli kullanmak amacıyla tablolarımız BCNF seviyesinde normalize edilmiştir.

Normalizasyon sürecinde uyguladığımız spesifik adımlar ve gerekçeleri şöyledir:

#### 1. Çoklu Değerlerin Ayrıştırılması (1NF Uygulaması):

- **Sorun:** Bir kullanıcının birden fazla kredi kartı olabilir. Eğer User tablosunda credit\_cards sütunu açıp kart numaralarını virgülle ayırarak tutsaydık, atomiklik kuralı (1NF) ihlal edilirdi.

- **Çözüm:** Kredi kartı bilgileri CreditCard isimli ayrı bir tabloya taşınmış ve User\_Cards ara tablosu ile kullanıcıya bağlanmıştır. Böylece her hücrede tek bir veri tutulması sağlanmıştır.

## 2. Kısmi Bağımlılıkların Giderilmesi (2NF Uygulaması):

- **Strateji:** Tasarımımızda Doğal Anahtarlar (Örn: TC Kimlik No, Email) yerine, sistem tarafından yönetilen Yapay Anahtarlar Birincil Anahtar olarak tercih edilmiştir.
- **Sonuç:** Tüm tablolarda, anahtar olmayan sütunlar, birleşik bir anahtarın parçasına değil, doğrudan ve tamamen id sütununa bağımlıdır. Bu sayede 2NF kuralı doğal olarak sağlanmıştır.

## 3. Geçişli (Transitive) Bağımlılıkların Giderilmesi (3NF Uygulaması):

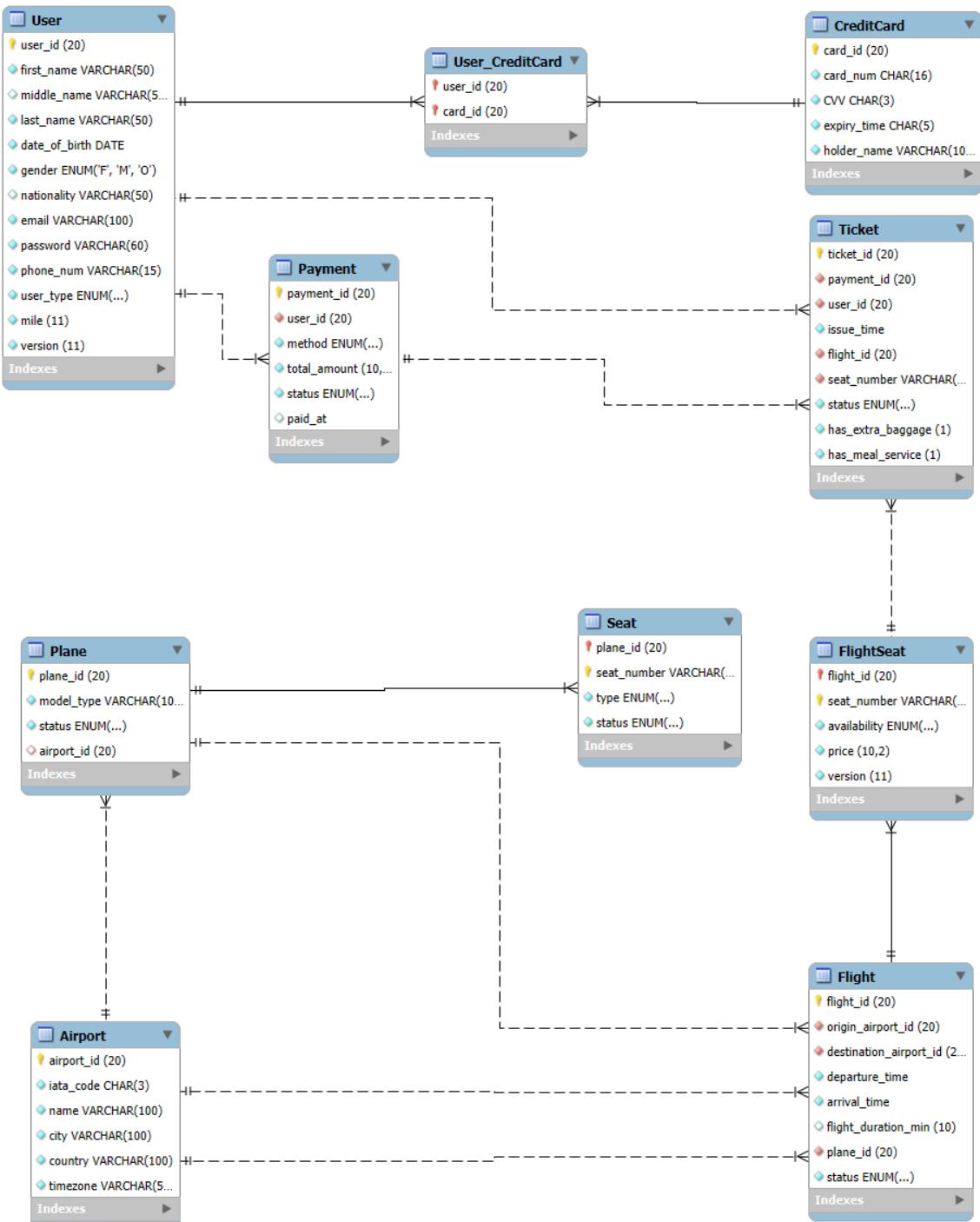
Normalizasyonun en kritik adımı burada uygulanmıştır. Veri tekrarını önlemek için şu ayırtırmalar yapılmıştır:

- **Havalimanı Detayları:** Flight tablosunda havalimanının city, country, timezone bilgileri tutulmamıştır. Bunun yerine sadece airport\_id referans olarak tutulmuş, detaylar Airport tablosuna taşınmıştır. Böylece "İstanbul" ismini değiştirmek istediğimizde binlerce uçuş satırını güncellemek yerine sadece Airport tablosundaki tek bir satırı güncellememiz yeterli olur.
- **Uçak Modeli:** Ticket tablosunda yolcunun bindiği uçağın modeli (Boeing 737 vb.) tutulmaz. Bu bilgiye Ticket -> Flight -> Plane ilişkisi üzerinden erişilir.

## 4. BCNF (Boyce-Codd Normal Form) Analizi: Tasarımımızda 3NF'in ötesine geçilerek, tüm determinantların aday anahtar olması sağlanmıştır.

- **Benzersizlik Kısıtları:** User tablosunda email ve phone\_num alanları, Airport tablosunda iata\_code alanı UNIQUE olarak işaretlenmiştir. Bu alanlar fonksiyonel olarak diğer sütunları belirlese bile, kendileri de birer Candidate Key olduğu için BCNF kuralı ihlal edilmemiştir.

### 4.3. Şema Diyagramı



## 5. Tasarımın Uyarlaması (Implementation)

### 5.1 Yazılım ve Donanım Ortamı

- Backend: Java 21, Spring Boot 3.3.3
- Frontend: React 18 (Vite), Material-UI (MUI)
- Veritabanı: MySQL 8.0 (TIDB Cloud kullanılarak ortak erişim sağlanmıştır.)
- ORM (Object-Relational Mapping): Hibernate (JPA)
- JDBC: Spring Data JPA
- Build Tool: Maven

### 5.2 Tabloların Oluşturulması

Veritabanı tabloları önce Tables.sql dosyası ile manuel olarak oluşturulmuştur. Spring Boot application.properties dosyasında hibernate.ddl-auto=update ayarı ile entity sınıfları ile veritabanı şeması senkronize edilmiştir.

#### Temel Tablolar ve Alanları:

##### **User Tablosu:** Kullanıcıların tutulduğu tablo

- user\_id (BIGINT, PK, AUTO\_INCREMENT): Kullanıcı benzersiz kimliği
- first\_name, middle\_name, last\_name (VARCHAR): Kullanıcı ad bilgileri
- date\_of\_birth (DATE): Doğum tarihi
- gender (ENUM: F, M, O): Cinsiyet bilgisi
- email (VARCHAR, UNIQUE): Benzersiz email adresi
- password (VARCHAR 60): Şifrelenmiş parola (BCrypt)
- phone\_num (VARCHAR 15, UNIQUE): Benzersiz telefon numarası
- user\_type (ENUM: customer, admin): Kullanıcı rolü
- mile (INT, DEFAULT 0): Müşteri mil puanı
- version (INT): Optimistic locking için versiyon numarası

##### **Flight Tablosu:** Uçuş bilgilerinin tutulduğu tablo

- flight\_id (BIGINT, PK, AUTO\_INCREMENT): Uçuş benzersiz kimliği
- origin\_airport\_id (BIGINT, FK → Airport): Kalkış havalimanı
- destination\_airport\_id (BIGINT, FK → Airport): Varış havalimanı
- departure\_time (TIMESTAMP): Kalkış zamanı

- arrival\_time (TIMESTAMP): Varış zamanı
- plane\_id (BIGINT, FK → Plane): Uçak referansı
- status (ENUM: SCHEDULED, ACTIVE, COMPLETED, CANCELLED): Uçuş durumu

**FlightSeat Tablosu (Composite Key):** Aktif uçuşları olan uçakların koltuk bilgileri

- flight\_id (BIGINT, PK, FK → Flight): Uçuş referansı
- seat\_number (VARCHAR 3, PK): Koltuk numarası (örn: 12A)
- availability (ENUM: available, reserved, sold): Koltuk durumu
- price (DECIMAL 10,2): Koltuk fiyatı
- version (INT): Race condition önleme için versiyon numarası

**Ticket Tablosu:** Bilet bilgilerinin tutulduğu tablo

- ticket\_id (BIGINT, PK, AUTO\_INCREMENT): Bilet benzersiz kimliği
- payment\_id (BIGINT, FK → Payment): Ödeme referansı
- user\_id (BIGINT, FK → User): Kullanıcı referansı
- flight\_id, seat\_number (FK → FlightSeat): Uçuş ve koltuk bilgisi
- issue\_time (TIMESTAMP): Bilet kesim zamanı
- status (ENUM: booked, cancelled, completed, pending): Bilet durumu
- has\_extra\_baggage (BOOLEAN): Ekstra bagaj seçeneği
- has\_meal\_service (BOOLEAN): Yemek servisi seçeneği

**Payment Tablosu:** Kullanıcının yaptığı harcamalar ve iade işlemleri (hesap hareketleri)

- payment\_id (BIGINT, PK, AUTO\_INCREMENT): Ödeme benzersiz kimliği
- user\_id (BIGINT, FK → User): Ödeme yapan kullanıcı
- method (ENUM: card, mile, cash): Ödeme yöntemi
- total\_amount (DECIMAL 10,2): Toplam tutar
- status (ENUM: pending, paid, refunded, failed): Ödeme durumu
- paid\_at (TIMESTAMP): Ödeme zamanı

**Airport Tablosu:** Havaalanlarının bilgilerinin tutulduğu tablo

- airport\_id (BIGINT, PK, AUTO\_INCREMENT): Havalimanı benzersiz kimliği
- iata\_code (CHAR 3, UNIQUE): IATA kodu (örn: IST, JFK)

- name, city, country (VARCHAR): Havalimanı bilgileri
- timezone (VARCHAR): Zaman dilimi (UTC formatında)

**Plane Tablosu:** Uçaklarının bilgilerinin tutulduğu tablo

- plane\_id (BIGINT, PK, AUTO\_INCREMENT): Uçak benzersiz kimliği
- model\_type (VARCHAR): Uçak modeli
- status (ENUM: active, maintenance, retired): Uçak durumu
- airport\_id (BIGINT, FK → Airport): Bulunduğu havalimanı

**Seat Tablosu (Composite Key):** Tüm uçakların koltuklarının bilgilerinin tutulduğu tablo

- plane\_id (BIGINT, PK, FK → Plane): Uçak referansı
- seat\_number (VARCHAR 3, PK): Koltuk numarası
- type (ENUM: economy, premium\_economy, business, first): Koltuk sınıfı
- status (ENUM: active, unavailable): Koltuk durumu

**CreditCard Tablosu:** Kredi kartlarının bilgilerinin tutulduğu tablo

- card\_id (BIGINT, PK, AUTO\_INCREMENT): Kart benzersiz kimliği
- card\_num (CHAR 16, UNIQUE): Kart numarası
- CVV (CHAR 3): Güvenlik kodu
- expiry\_time (CHAR 5): Son kullanma tarihi (MM/YY)
- holder\_name (VARCHAR 100): Kart sahibi adı

**User\_CreditCard Tablosu (Many-to-Many Join Table):**

- user\_id (BIGINT, PK, FK → User)
- card\_id (BIGINT, PK, FK → CreditCard)

### 5.3 Görünümler (Views)

Projede view kullanılmamıştır. Bunun yerine JPA repository'lerinde JOIN işlemleri runtime'da gerçekleştirilmiştir.

View Kullanılmama Nedenleri:

### **1. Dinamik Veri Yapısı:**

Projede flight ve ticket verileri sürekli değişmektedir (koltuk durumları, uçuş statüleri, ödeme durumları). View'lar statik sorgu sonuçlarını saklar ve her sorgulamada yeniden hesaplanır, ancak materialized view olmadığı için performans avantajı sınırlıdır. ORM (JPA) ile yapılan sorgular zaten optimize edilmiştir ve gerektiğinde cache mekanizması kullanılabilir.

### **2. ORM (JPA) Avantajları:**

Spring Data JPA, JOIN işlemlerini JPQL veya method naming ile otomatik olarak yönetir. Entity ilişkileri (@ManyToOne, @OneToMany) sayesinde code-first yaklaşım ile veritabanı yapısı ve sorguları senkronize tutulur. View kullanımı entity mapping'lerini karmaşıklığından ve Hibernate'in lazy loading, caching gibi özelliklerinden faydalananmayı zorlaştırır.

### **3. Flexibility (Esneklik):**

Farklı endpoint'lerde farklı JOIN kombinasyonları gerekebilir. Örneğin:

- Uçuş arama: Flight + Origin Airport + Destination Airport
- Bilet listeleme: Ticket + Flight + Airport + Payment + User
- Ödeme geçmişi: Payment + User (Flight bilgisi gerekmez)

Her kombinasyon için ayrı view oluşturmak ve sürekli yenilemek yerine, repository method'larda @Query ile ihtiyaca göre JOIN yapılması daha espektifdir.

## **5.4 İndisler (Indexes)**

Performans optimizasyonu için sık kullanılan sorgulara yönelik indexler oluşturulmuştur:

### **User Tablosu:**

- email: Email ile login işlemi (authentication)
- phone\_num: Telefon numarası ile kullanıcı arama

İki attribute de UNIQUE constraint ile otomatik indexlidir, bu nedenle ekstra index oluşturulmamıştır.

### **Airport Tablosu:**

- idx\_airport\_iata: IATA kodu ile hızlı havalimanı bulma
- idx\_airport\_city: Şehre göre havalimanı listeleme
- idx\_airport\_country: Ülkeye göre havalimanı filtreleme

### **Flight Tablosu :**

- idx\_flight\_origin\_dest: (origin\_airport\_id, destination\_airport\_id) composite index - Uçuş arama sayfasında en sık kullanılan soru
- idx\_flight\_departure: departure\_time ile tarih bazlı filtreleme
- idx\_flight\_plane: plane\_id ile belirli uçağın uçuşlarını listeleme
- idx\_flight\_status: Status'e göre aktif/iptal edilmiş uçuşları filtreleme

### **FlightSeat Tablosu:**

- idx\_flightseat\_flight: flight\_id ile uçuşun tüm koltuklarını getirme
- idx\_flightseat\_availability: (flight\_id, availability) composite index - Müsait koltukları hızlı bulma

### **Ticket Tablosu:**

- idx\_ticket\_user: user\_id ile kullanıcının tüm biletlerini listeleme (My Flights sayfası)
- idx\_ticket\_payment: payment\_id ile ödemeye ait biletleri bulma
- idx\_ticket\_flight: flight\_id ile uçuşa ait biletleri listeleme

### **Payment Tablosu:**

- idx\_payment\_user: user\_id ile kullanıcının ödeme geçmişini getirme (Payment History sayfası)
- idx\_payment\_status: Ödeme durumuna göre filtreleme (pending, paid, failed)

### **Plane Tablosu:**

- idx\_plane\_airport: airport\_id ile havalimanındaki uçakları listeleme

- idx\_plane\_status: Status'e göre aktif/bakımdaki uçakları filtreleme

## 5.5 Sorgu Tasarımları

### JOIN Stratejileri

- INNER JOIN: Zorunlu ilişkilerde kullanılmıştır (Flight-Airport ilişkisi, her uçuşun mutlaka origin ve destination airport'u olmalı)
- LEFT JOIN FETCH: N+1 problemini önlemek için eager loading yapılmıştır (Örnek: TicketRepository'de kullanıcının biletleri çekilirken flight, airport, payment bilgileri tek sorguda getirilir)

### Composite Key Kullanımı:

- FlightSeat: (flight\_id, seat\_number) - Belirli bir uçuştaki belirli koltuğu unique olarak tanımlar
- Seat: (plane\_id, seat\_number) - Belirli bir uçaktaki belirli koltuğu unique olarak tanımlar
- User\_CreditCard: (user\_id, card\_id) - Many-to-Many ilişki için join table

### Özel Sorgular (Custom Queries):

- TicketRepository: @Query ile kullanıcının biletlerini flight ve airport detayları ile birlikte getiren optimize edilmiş JPQL sorgusu
- FlightRepository: Origin ve destination airport'a göre, belirli tarih aralığında uçuş arama sorgusu
- PaymentRepository: Kullanıcının ödeme geçmişini paid\_at tarihine göre sıralı getirme

### Repository Method Naming Convention:

SELECT sorgularıdır, JPA Repository kolaylık sağlar. Örneğin:

- findByEmail, findByPhoneNum (UserRepository)
- findByOriginAirportAirportIdAndDestinationAirportAirportId (FlightRepository)
- findByFlightIdAndAvailability (FlightSeatRepository)
- findByUserUserId (TicketRepository)

## 5.6 Transaction Yönetimi

### @Transactional Kullanımı:

- Bilet satın alma işlemi: Payment oluşturma, Ticket oluşturma ve FlightSeat güncelleme işlemleri tek transaction içinde yapılır. Herhangi bir hata durumunda tüm işlem geri alınır (rollback)
- Kredi kartı ekleme işlemi: User ve CreditCard many-to-many ilişkisi transaction içinde güncellenir
- Uçuş oluşturma: Flight oluşturulurken ilgili FlightSeat kayıtları da otomatik oluşturulur

### Optimistic Locking (@Version):

- FlightSeat tablosunda version kolonu: Aynı koltuğu aynı anda birden fazla kullanıcının satın almasını önler (race condition)
- User tablosunda version kolonu: Kullanıcı bilgileri güncellenirken veri tutarlılığını sağlar
- Çakışma durumunda OptimisticLockException fırlatılır ve kullanıcıya bilgi verilir

### Cascade İşlemleri:

- User-CreditCard: CascadeType.PERSIST, MERGE (Kullanıcı silindiğinde kartlar silinir - DELETE CASCADE)
- Flight-FlightSeat: FK constraint ile ON DELETE RESTRICT (Uçuşa ait bilet varsa uçuş silinemez)
- Payment-Ticket: ON DELETE RESTRICT (Ödeme kaydı korunmalı)

## 5.7 REST API Tasarımı ve Endpoint'ler:

Tüm API endpoint'leri RESTful mimari prensiplerine göre tasarlanmıştır.

### Authentication Controller (/api/auth):

- POST /api/auth/signup: Kullanıcı kaydı
- POST /api/auth/login: Kullanıcı girişi, JWT token döner
  - Request: { email, password }
  - Response: { token, userId, email, firstName, lastName, userType }

### **Flight Controller (/api/flights):**

- GET /api/flights: Tüm uçuşları listele
- GET /api/flights/{id}: ID'ye göre uçuş detayı
- POST /api/flights/search: Uçuş arama
  - Request: { originAirportId, destinationAirportId, departureDate }
  - Response: Flight listesi (origin, destination airport bilgileriyle)
- POST /api/flights (Admin): Yeni uçuş oluşturma
- GET /api/flights/{id}/seats: Uçuşun koltuk durumları

### **Ticket Controller (/api/tickets):**

- GET /api/tickets/user/{userId}: Kullanıcının biletleri
- POST /api/tickets: Bilet satın alma (Payment ile birlikte transaction)

### **Payment Controller (/api/payments):**

- POST /api/payments: Ödeme işlemi
- GET /api/payments/user/{userId}: Kullanıcının ödeme geçmişi
- POST /api/tickets/{ticketId}/cancel: Bilet iptali

### **User Controller (/api/users):**

- GET /api/users/{id}: Kullanıcı bilgileri
- PUT /api/users/{id}: Kullanıcı güncelleme
- PUT /api/users/{id}/change-password: Şifre değiştirme
- GET /api/users/{emailOrPhoneNum}: Email veya telefon ile kullanıcı bilgisi
- DELETE /api/users/{userId}: Kullanıcı silme
- GET /api/users/{userId}/credit-cards: Kullanıcının kayıtlı kartları
- POST /api/users/{userId}/credit-cards: Kredi kartı ekleme
- DELETE /api/users/{userId}/credit-cards/{cardId}: Kredi kartı silme
- PUT /api/users/{userId}/change-password: Şifre değiştirme

### **Admin Controller (/api/admin):**

- POST /api/admin/planes: Uçak ekleme
- GET /api/admin/planes: Uçak listesi

- GET /api/admin/planes: Tüm uçakları liste
- POST /api/admin/planes: Yeni uçak ekleme
- DELETE /api/admin/planes/{planeId}: Uçak silme
- GET /api/admin/planes/{planeId}: Uçak detayı
- GET /api/admin/planes/{planeId}/seats: Uçağın koltuk yapısı
- GET /api/admin/airports: Tüm havalimanlarını liste
- POST /api/admin/airports: Havalimanı ekleme
- DELETE /api/admin/airports/{airportId}: Havalimanı silme
- GET /api/admin/airports/{airportId}: Havalimanı detayı
- GET /api/admin/airports/{airportId}/planes: Havalimanındaki uçaklar
- GET /api/admin/flights: Tüm uçuşları liste
- POST /api/admin/flights: Yeni uçuş oluşturma
- POST /api/admin/flights/{flightId}/cancel: Uçuş iptal etme
- POST /api/admin/planes/{planeId}/malfunction: Koltuk arızası bildirme
- PUT /api/admin/planes/{planeId}/status: Uçak durumu güncelleme
- PUT /api/admin/planes/{planeId}/airport/{airportId}: Uçağın havalimanını değiştirme
- PUT /api/admin/planes/{planeId}/seats/{seatNumber}/status: Koltuk durumu güncelleme

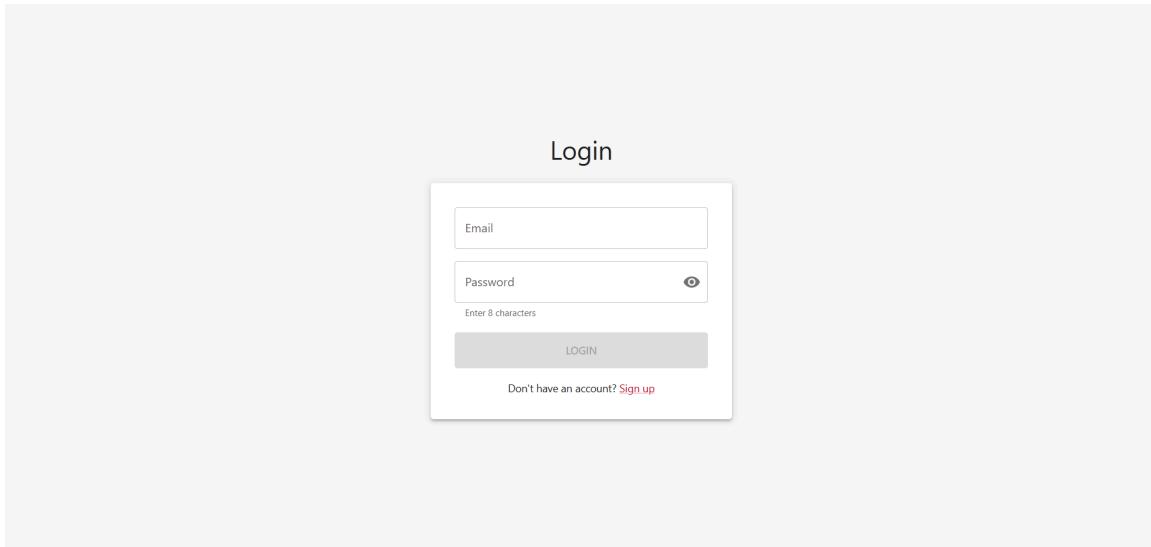
## 6. Uygulama Program Tanıtımı ve Örnek Kullanım

Geliştirilen sistemin arayüzleri, son kullanıcı ve admin deneyimini optimize edecek şekilde React (Material UI) ile tasarlanmıştır. Aşağıda temel kullanım senaryoları ve ekran görüntüleri yer almaktadır.

### 6.1. Giriş ve Kayıt Ekranı (Authentication)

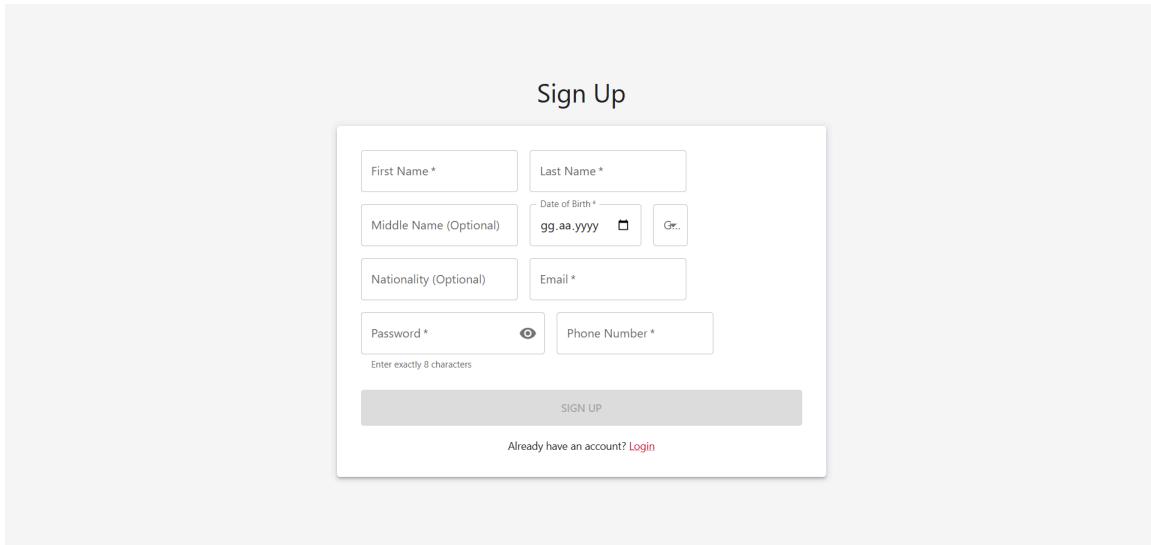
Kullanıcıların güvenli bir şekilde sisteme giriş yapmasını sağlayan modüldür.

- **Teknik Detay:** Giriş yapıldığında Backend'den dönen JWT Token, Frontend tarafında localStorage'da saklanır ve sonraki tüm API isteklerinin Header kısmına Authorization: Bearer olarak eklenir.



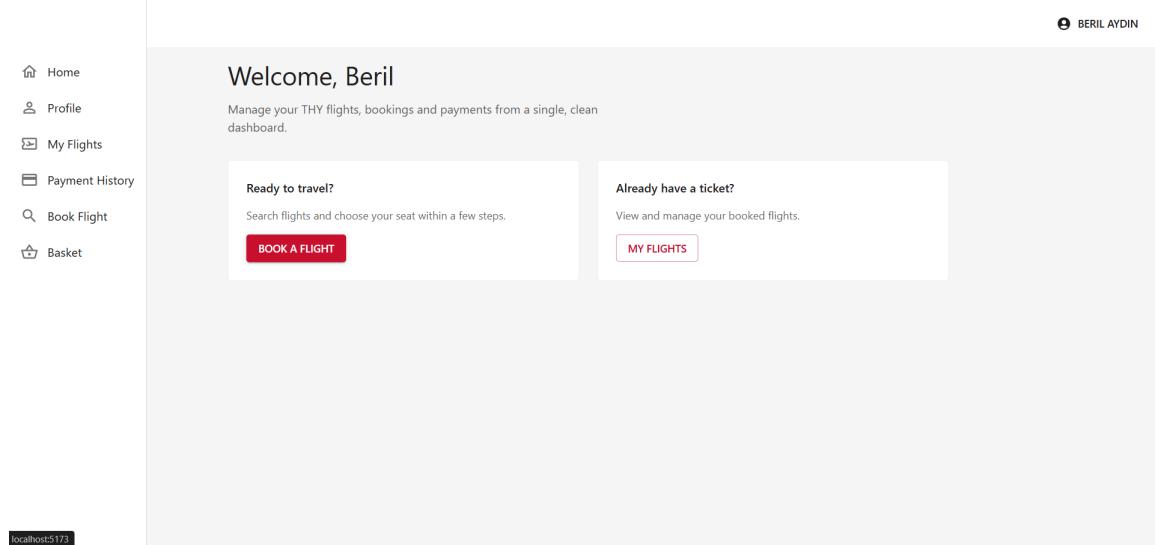
The image shows a login interface titled "Login". It features two input fields: "Email" and "Password". Below the password field is a placeholder "Enter 8 characters". A "LOGIN" button is centered below the fields. At the bottom, there is a link "Don't have an account? [Sign up](#)".

*Görsel 1: Login (Giriş) Ekrani*



The image shows a sign-up interface titled "Sign Up". It includes fields for "First Name \*", "Last Name \*", "Middle Name (Optional)", "Date of Birth \*", "Nationality (Optional)", "Email \*", "Password \*", and "Phone Number \*". The "Date of Birth" field includes a date input and a "Gr." dropdown. The "Password" field has a placeholder "Enter exactly 8 characters". A "SIGN UP" button is at the bottom, and a "Login" link is at the very bottom.

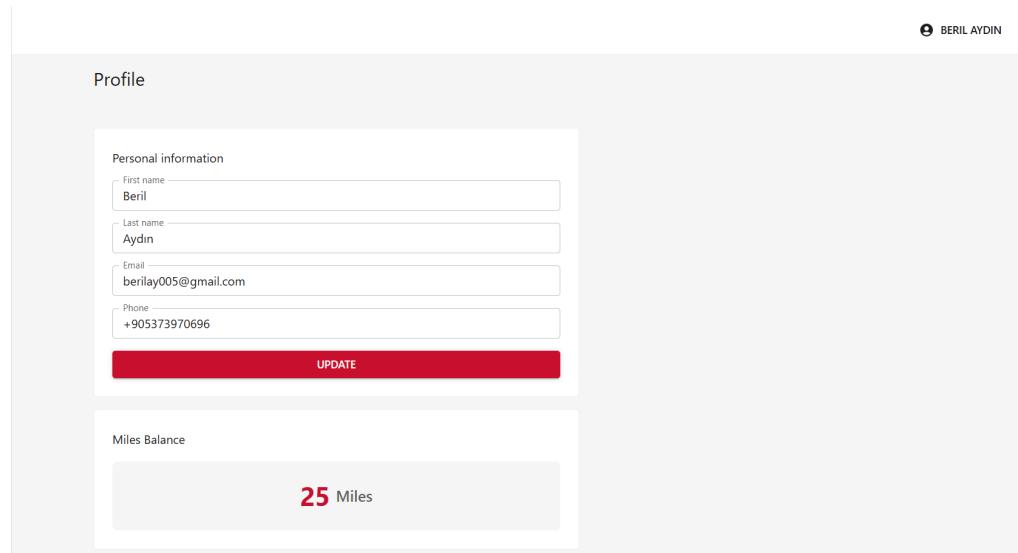
*Görsel 2: Signup (Kayıt) Ekrani*

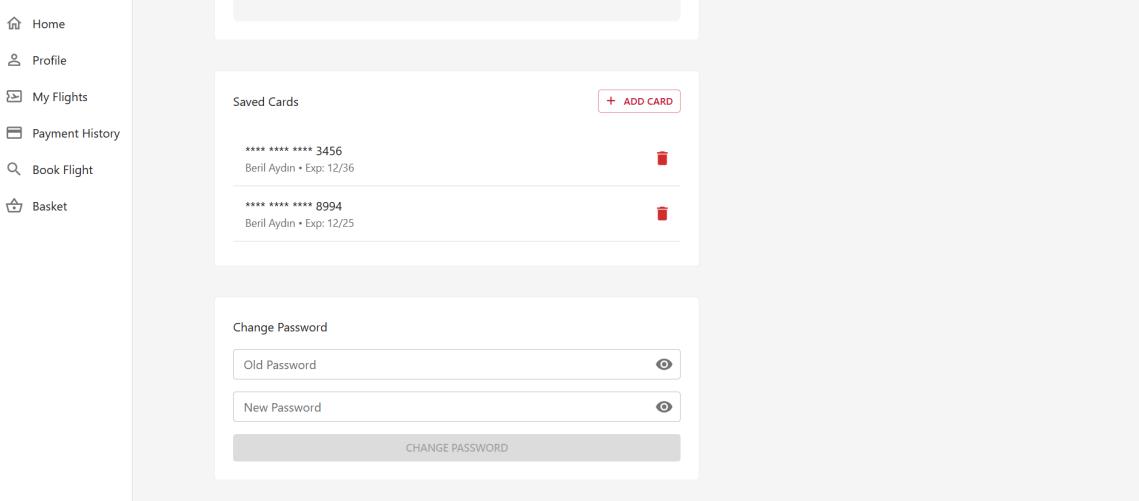


Görsel 3: Ana Ekran

## 6.2. Profil Ekranı (Profile)

Kullanıcının bilgilerini görebildiği ve güncelleme yapabildiği ekranıdır.



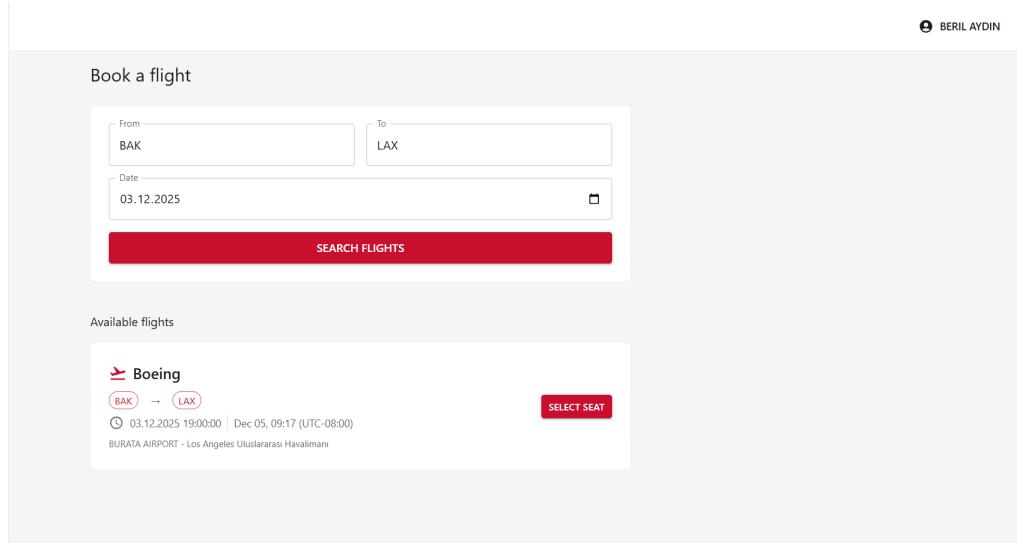


Görsel 4-5: Profil Ekranı

### 6.3. Uçuş Arama ve Listeleme (Flight Search)

Kullanıcının kalkış/varış noktası ve tarih seçerek uygun uçuşları listelediği ekrandır.

- Teknik Detay:** Bu ekrandaki soru, veritabanında idx\_flight\_origin\_dest indeksini kullanarak çalışır. Flight tablosu Airport tablosu ile INNER JOIN FETCH yapılarak, havalimanı isimleri tek bir sorguda getirilir.

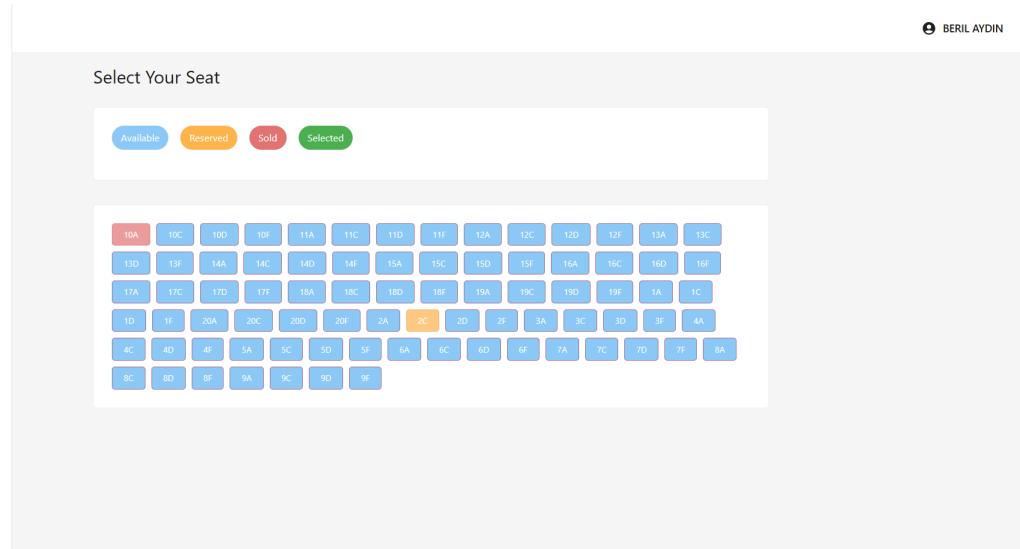


Görsel 6: Uçuş Arama Ekranı ve Örnek Sonuç

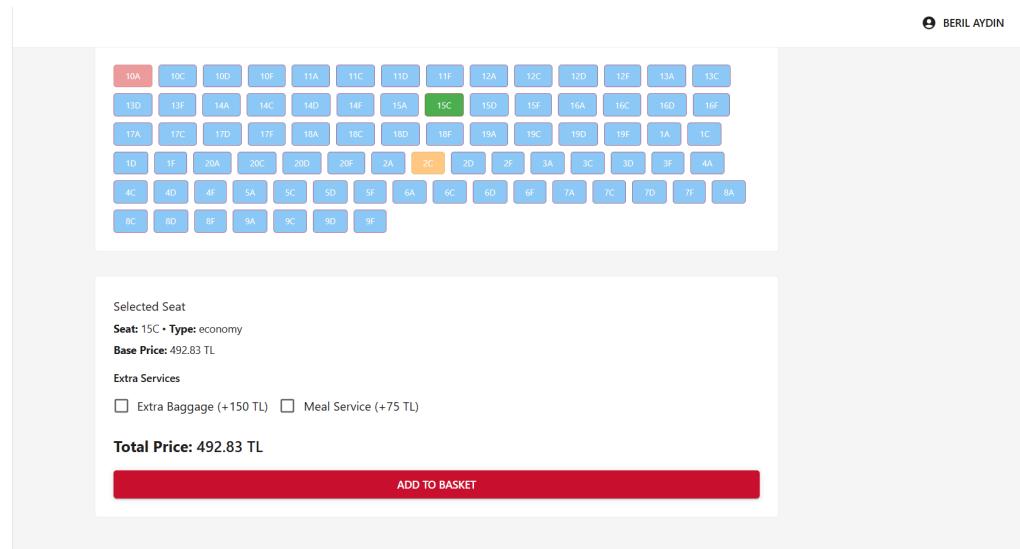
#### **6.4. Koltuk Seçimi (Seat Selection)**

Kullanıcının uçağın güncel doluluk durumuna göre koltuk seçtiği interaktif ekrandır.

- **Teknik Detay:** Bu ekran açıldığında GET /api/flights/{id}/seats endpoint'i çağrılır. FlightSeat tablosundan o uçuşa ait tüm koltukların availability durumu çekilir. "Sold" olanlar kırmızı, "Reserved" olanlar sarı, "Available" olanlar mavi gösterilir.



*Görsel 7: Koltuk Seçim Ekranı*

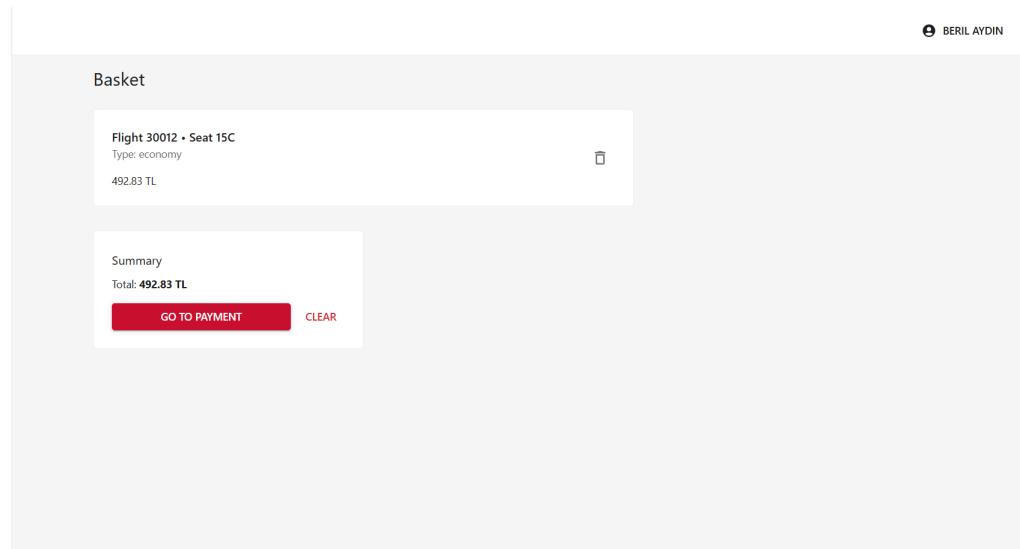


*Görsel 8: Seçilen Koltuk Bilgileri ve Ekstra Servis Seçeneği*

## 6.5. Ödeme ve Biletleme (Payment & Transaction)

Sepet ekranından ödeme ekranına gidilir. Burada ödeme yöntemi seçilir, kredi kartı seçildiyse kullanıcının kredi kartı bilgilerini girer veya kayıtlı kart seçilir ve işlem tamamlanır. Alınan biletler My Flight ekranında, harcamalar ve iadeler Payments ekranında gösterilir.

- **Teknik Detay (ACID):** "Ödeme Yap" butonuna basıldığında Backend'de @Transactional blok başlar.
  1. Ödeme (Payment) kaydı oluşturulur.
  2. Seçilen koltuğun FlightSeat tablosundaki durumu SOLD olarak güncellenir.
  3. Bilet (Ticket) oluşturulur.
  4. Tüm işlemler başarılıysa Commit edilir.



Görsel 9: Sepet ekranı

- [!\[\]\(df97338b4ed9c84de0aa8b40b09055fa\_img.jpg\) Home](#)
- [!\[\]\(b22c7e8560907ffb4e97a8ca6af278e9\_img.jpg\) Profile](#)
- [!\[\]\(0a0ac6cc6d55a8d77e6e348c940ad042\_img.jpg\) My Flights](#)
- [!\[\]\(a194910639352cc7e6ba1807c01ec960\_img.jpg\) Payment History](#)
- [!\[\]\(39e21277690d79d1c52b3a86fa83368d\_img.jpg\) Book Flight](#)
- [!\[\]\(2b9dc98ffaf75730ad126d509d400942\_img.jpg\) Basket](#)

## Payment

### Payment Method

- Credit/Debit Card
- Cash
- Miles (Available: 0)

### Card Information

- Use saved card
- Use new card

Card holder \*

Card number \*

16 digits

*Görsel 10: Ödeme Ekranı*

- [!\[\]\(8c807e59e51969c50b31748894b36a15\_img.jpg\) Home](#)
- [!\[\]\(89330e1edf7db2f683a90d2ccec8e3db\_img.jpg\) Profile](#)
- [!\[\]\(24ca82658f65ef56383e4092e47e062c\_img.jpg\) My Flights](#)
- [!\[\]\(01eacffd919a93b0bd1c6b410a8b3f41\_img.jpg\) Payment History](#)
- [!\[\]\(f2d932e90cf76c8edb77631ef15f4601\_img.jpg\) Book Flight](#)
- [!\[\]\(13ee9f07b952d1d00a2b5054c2e78019\_img.jpg\) Basket](#)

## My Flights

[ACTIVE](#) [BOOKED](#) [PENDING](#) [CANCELLED](#) [ALL](#)

BAK → LAX

Departure: 03.12.2025 19:00:00  
Arrival: Dec 05, 09:17 (UTC-08:00) • Seat: 15C

[booked](#)

[CANCEL TICKET](#)

Total: 492.83 TL  
Status: booked

*Görsel 11: Bilet Ekranı (My Flights)*

**Payments**

- Payment • card**  
BAK → LAX • 15C  
03.12.2025 19:00:00 – Dec 05, 09:17 (UTC-08:00)  
**Total: 492.83 TL**
- Refund • cash** Refunded  
Refund: 985.67 TL
- Refund • card** Refunded  
Refund: 492.83 TL
- Payment • cash**  
BAK → LAX • 2C  
03.12.2025 19:00:00 – Dec 05, 09:17 (UTC-08:00)  
**Total: 985.67 TL**

Görsel 12: Hesap Hareketleri Ekranı (Payments)

## 6.6. Admin Paneli (Operasyonel Yönetim)

Yöneticinin yeni uçak/uçuş eklediği veya uçuşları iptal ettiği yönetim panelidir.

- Teknik Detay:** Admin yeni bir uçuş eklediğinde, sistem arka planda o uçağın Seat tablosundaki fiziksel koltuk şablonunu kopyalar ve FlightSeat tablosuna o uçuş için binlerce satır (Envanter) oluşturur.

Plane Info #30004

Seat Map (80 seats) Status: Active (Green Border), Unavailable (Red Border) Type: Business (Blue), Economy (Orange), Premium Economy (Purple), First Class (Gold)

Row	Seat	Type	Status
1A	Business	ACTIVE	
1B	Economy	ACTIVE	
1C	Economy	ACTIVE	
1D	Economy	UNAVAILABLE	
1E	Economy	ACTIVE	
1F	Economy	ACTIVE	
1G	Economy	ACTIVE	
1H	Economy	ACTIVE	
1I	Economy	ACTIVE	
1J	Economy	ACTIVE	
1K	Economy	ACTIVE	
1L	Economy	ACTIVE	
1M	Economy	ACTIVE	
1N	Economy	ACTIVE	
1O	Economy	ACTIVE	
1P	Economy	ACTIVE	
1Q	Economy	ACTIVE	
1R	Economy	ACTIVE	
1S	Economy	ACTIVE	
1T	Economy	ACTIVE	
1U	Economy	ACTIVE	
1V	Economy	ACTIVE	
1W	Economy	ACTIVE	
1X	Economy	ACTIVE	
1Y	Economy	ACTIVE	
1Z	Economy	ACTIVE	
2A	Business	ACTIVE	
2B	Business	ACTIVE	
2C	Business	ACTIVE	
2D	Business	ACTIVE	
2E	Business	ACTIVE	
2F	Business	ACTIVE	
2G	Business	ACTIVE	
2H	Business	ACTIVE	
2I	Business	ACTIVE	
2J	Business	ACTIVE	
2K	Business	ACTIVE	
2L	Business	ACTIVE	
2M	Business	ACTIVE	
2N	Business	ACTIVE	
2O	Business	ACTIVE	
2P	Business	ACTIVE	
2Q	Business	ACTIVE	
2R	Business	ACTIVE	
2S	Business	ACTIVE	
2T	Business	ACTIVE	
2U	Business	ACTIVE	
2V	Business	ACTIVE	
2W	Business	ACTIVE	
2X	Business	ACTIVE	
2Y	Business	ACTIVE	
2Z	Business	ACTIVE	
3A	Economy	ACTIVE	
3B	Economy	ACTIVE	
3C	Economy	ACTIVE	
3D	Economy	ACTIVE	
3E	Economy	ACTIVE	
3F	Economy	ACTIVE	
3G	Economy	ACTIVE	
3H	Economy	ACTIVE	
3I	Economy	ACTIVE	
3J	Economy	ACTIVE	
3K	Economy	ACTIVE	
3L	Economy	ACTIVE	
3M	Economy	ACTIVE	
3N	Economy	ACTIVE	
3O	Economy	ACTIVE	
3P	Economy	ACTIVE	
3Q	Economy	ACTIVE	
3R	Economy	ACTIVE	
3S	Economy	ACTIVE	
3T	Economy	ACTIVE	
3U	Economy	ACTIVE	
3V	Economy	ACTIVE	
3W	Economy	ACTIVE	
3X	Economy	ACTIVE	
3Y	Economy	ACTIVE	
3Z	Economy	ACTIVE	
4A	Economy	ACTIVE	
4B	Economy	ACTIVE	
4C	Economy	ACTIVE	
4D	Economy	ACTIVE	
4E	Economy	ACTIVE	
4F	Economy	ACTIVE	
4G	Economy	ACTIVE	
4H	Economy	ACTIVE	
4I	Economy	ACTIVE	
4J	Economy	ACTIVE	
4K	Economy	ACTIVE	
4L	Economy	ACTIVE	
4M	Economy	ACTIVE	
4N	Economy	ACTIVE	
4O	Economy	ACTIVE	
4P	Economy	ACTIVE	
4Q	Economy	ACTIVE	
4R	Economy	ACTIVE	
4S	Economy	ACTIVE	
4T	Economy	ACTIVE	
4U	Economy	ACTIVE	
4V	Economy	ACTIVE	
4W	Economy	ACTIVE	
4X	Economy	ACTIVE	
4Y	Economy	ACTIVE	
4Z	Economy	ACTIVE	
5A	Economy	ACTIVE	
5B	Economy	ACTIVE	
5C	Economy	ACTIVE	
5D	Economy	ACTIVE	
5E	Economy	ACTIVE	
5F	Economy	ACTIVE	
5G	Economy	ACTIVE	
5H	Economy	ACTIVE	
5I	Economy	ACTIVE	
5J	Economy	ACTIVE	
5K	Economy	ACTIVE	
5L	Economy	ACTIVE	
5M	Economy	ACTIVE	
5N	Economy	ACTIVE	
5O	Economy	ACTIVE	
5P	Economy	ACTIVE	
5Q	Economy	ACTIVE	
5R	Economy	ACTIVE	
5S	Economy	ACTIVE	
5T	Economy	ACTIVE	
5U	Economy	ACTIVE	
5V	Economy	ACTIVE	
5W	Economy	ACTIVE	
5X	Economy	ACTIVE	
5Y	Economy	ACTIVE	
5Z	Economy	ACTIVE	
6A	Economy	ACTIVE	
6B	Economy	ACTIVE	
6C	Economy	ACTIVE	
6D	Economy	ACTIVE	
6E	Economy	ACTIVE	
6F	Economy	ACTIVE	
6G	Economy	ACTIVE	
6H	Economy	ACTIVE	
6I	Economy	ACTIVE	
6J	Economy	ACTIVE	
6K	Economy	ACTIVE	
6L	Economy	ACTIVE	
6M	Economy	ACTIVE	
6N	Economy	ACTIVE	
6O	Economy	ACTIVE	
6P	Economy	ACTIVE	
6Q	Economy	ACTIVE	
6R	Economy	ACTIVE	
6S	Economy	ACTIVE	
6T	Economy	ACTIVE	
6U	Economy	ACTIVE	
6V	Economy	ACTIVE	
6W	Economy	ACTIVE	
6X	Economy	ACTIVE	
6Y	Economy	ACTIVE	
6Z	Economy	ACTIVE	
7A	Economy	ACTIVE	
7B	Economy	ACTIVE	
7C	Economy	ACTIVE	
7D	Economy	ACTIVE	
7E	Economy	ACTIVE	
7F	Economy	ACTIVE	
7G	Economy	ACTIVE	
7H	Economy	ACTIVE	
7I	Economy	ACTIVE	
7J	Economy	ACTIVE	
7K	Economy	ACTIVE	
7L	Economy	ACTIVE	
7M	Economy	ACTIVE	
7N	Economy	ACTIVE	
7O	Economy	ACTIVE	
7P	Economy	ACTIVE	
7Q	Economy	ACTIVE	
7R	Economy	ACTIVE	
7S	Economy	ACTIVE	
7T	Economy	ACTIVE	
7U	Economy	ACTIVE	
7V	Economy	ACTIVE	
7W	Economy	ACTIVE	
7X	Economy	ACTIVE	
7Y	Economy	ACTIVE	
7Z	Economy	ACTIVE	
8A	Economy	ACTIVE	
8B	Economy	ACTIVE	
8C	Economy	ACTIVE	
8D	Economy	ACTIVE	
8E	Economy	ACTIVE	
8F	Economy	ACTIVE	
8G	Economy	ACTIVE	
8H	Economy	ACTIVE	
8I	Economy	ACTIVE	
8J	Economy	ACTIVE	
8K	Economy	ACTIVE	
8L	Economy	ACTIVE	
8M	Economy	ACTIVE	
8N	Economy	ACTIVE	
8O	Economy	ACTIVE	
8P	Economy	ACTIVE	
8Q	Economy	ACTIVE	
8R	Economy	ACTIVE	
8S	Economy	ACTIVE	
8T	Economy	ACTIVE	
8U	Economy	ACTIVE	
8V	Economy	ACTIVE	
8W	Economy	ACTIVE	
8X	Economy	ACTIVE	
8Y	Economy	ACTIVE	
8Z	Economy	ACTIVE	
9A	Economy	ACTIVE	
9B	Economy	ACTIVE	
9C	Economy	ACTIVE	
9D	Economy	ACTIVE	
9E	Economy	ACTIVE	
9F	Economy	ACTIVE	
9G	Economy	ACTIVE	
9H	Economy	ACTIVE	
9I	Economy	ACTIVE	
9J	Economy	ACTIVE	
9K	Economy	ACTIVE	
9L	Economy	ACTIVE	
9M	Economy	ACTIVE	
9N	Economy	ACTIVE	
9O	Economy	ACTIVE	
9P	Economy	ACTIVE	
9Q	Economy	ACTIVE	
9R	Economy	ACTIVE	
9S	Economy	ACTIVE	
9T	Economy	ACTIVE	
9U	Economy	ACTIVE	
9V	Economy	ACTIVE	
9W	Economy	ACTIVE	
9X	Economy	ACTIVE	
9Y	Economy	ACTIVE	
9Z	Economy	ACTIVE	
10A	Economy	ACTIVE	
10B	Economy	ACTIVE	
10C	Economy	ACTIVE	
10D	Economy	ACTIVE	
10E	Economy	ACTIVE	
10F	Economy	ACTIVE	
10G	Economy	ACTIVE	
10H	Economy	ACTIVE	
10I	Economy	ACTIVE	
10J	Economy	ACTIVE	
10K	Economy	ACTIVE	
10L	Economy	ACTIVE	
10M	Economy	ACTIVE	
10N	Economy	ACTIVE	
10O	Economy	ACTIVE	
10P	Economy	ACTIVE	
10Q	Economy	ACTIVE	
10R	Economy	ACTIVE	
10S	Economy	ACTIVE	
10T	Economy	ACTIVE	
10U	Economy	ACTIVE	
10V	Economy	ACTIVE	
10W	Economy	ACTIVE	
10X	Economy	ACTIVE	
10Y	Economy	ACTIVE	
10Z	Economy	ACTIVE	
11A	Economy	ACTIVE	
11B	Economy	ACTIVE	
11C	Economy	ACTIVE	
11D	Economy	ACTIVE	
11E	Economy	ACTIVE	
11F	Economy	ACTIVE	
11G	Economy	ACTIVE	
11H	Economy	ACTIVE	
11I	Economy	ACTIVE	
11J	Economy	ACTIVE	
11K	Economy	ACTIVE	
11L	Economy	ACTIVE	
11M	Economy	ACTIVE	
11N	Economy	ACTIVE	
11O	Economy	ACTIVE	
11P	Economy	ACTIVE	
11Q	Economy	ACTIVE	
11R	Economy	ACTIVE	
11S	Economy	ACTIVE	
11T	Economy	ACTIVE	
11U	Economy	ACTIVE	
11V	Economy	ACTIVE	
11W	Economy	ACTIVE	
11X	Economy	ACTIVE	
11Y	Economy	ACTIVE	
11Z	Economy	ACTIVE	
12A	Economy	ACTIVE	
12B	Economy	ACTIVE	
12C	Economy	ACTIVE	
12D	Economy	ACTIVE	
12E	Economy	ACTIVE	
12F	Economy	ACTIVE	
12G	Economy	ACTIVE	
12H	Economy	ACTIVE	
12I	Economy	ACTIVE	
12J	Economy	ACTIVE	
12K	Economy	ACTIVE	
12L	Economy	ACTIVE	
12M	Economy	ACTIVE	
12N	Economy	ACTIVE	
12O	Economy	ACTIVE	
12P	Economy	ACTIVE	
12Q	Economy	ACTIVE	
12R	Economy	ACTIVE	
12S	Economy	ACTIVE	
12T	Economy	ACTIVE	
12U	Economy	ACTIVE	
12V	Economy	ACTIVE	
12W	Economy	ACTIVE	
12X	Economy	ACTIVE	
12Y	Economy	ACTIVE	
12Z	Economy	ACTIVE	
13A	Economy	ACTIVE	
13B	Economy	ACTIVE	
13C	Economy	ACTIVE	
13D	Economy	ACTIVE	
13E	Economy	ACTIVE	
13F	Economy	ACTIVE	
13G	Economy	ACTIVE	
13H	Economy	ACTIVE	
13I	Economy	ACTIVE	
13J	Economy	ACTIVE	
13K	Economy	ACTIVE	
13L	Economy	ACTIVE	
13M	Economy	ACTIVE	
13N	Economy	ACTIVE	
13O	Economy	ACTIVE	
13P	Economy	ACTIVE	
13Q	Economy	ACTIVE	
13R	Economy	ACTIVE	
13S	Economy	ACTIVE	
13T	Economy	ACTIVE	
13U	Economy	ACTIVE	
13V	Economy	ACTIVE	
13W	Economy	ACTIVE	
13X	Economy	ACTIVE	
13Y	Economy	ACTIVE	
13Z	Economy	ACTIVE	
14A	Economy	ACTIVE	
14B	Economy	ACTIVE	
14C	Economy	ACTIVE	
14D	Economy	ACTIVE	
14E	Economy	ACTIVE	
14F	Economy	ACTIVE	
14G	Economy	ACTIVE	
14H	Economy	ACTIVE	
14I	Economy	ACTIVE	
14J	Economy	ACTIVE	
14K	Economy	ACTIVE	
14L	Economy	ACTIVE	
14M	Economy	ACTIVE	
14N	Economy	ACTIVE	
14O	Economy	ACTIVE	
14P	Economy	ACTIVE	
14Q	Economy	ACTIVE	
14R	Economy	ACTIVE	
14S	Economy	ACTIVE	
14T	Economy	ACTIVE	
14U	Economy	ACTIVE	
14V	Economy	ACTIVE	
14W	Economy	ACTIVE	
14X	Economy	ACTIVE	
14Y	Economy	ACTIVE	
14Z	Economy	ACTIVE	
15A	Economy	ACTIVE	
15B	Economy	ACTIVE	
15C	Economy	ACTIVE	
15D	Economy	ACTIVE	
15E	Economy	ACTIVE	
15F	Economy	ACTIVE	
15G	Economy	ACTIVE	
15H	Economy	ACTIVE	
15I	Economy	ACTIVE	
15J	Economy	ACTIVE	
15K	Economy	ACTIVE	
15L	Economy	ACTIVE	
15M	Economy	ACTIVE	
15N	Economy	ACTIVE	
15O	Economy	ACTIVE	
15P	Economy	ACTIVE	
15Q	Economy	ACTIVE	
15R	Economy	ACTIVE	
15S	Economy	ACTIVE	
15T	Economy	ACTIVE	
15U	Economy	ACTIVE	
15V	Economy	ACTIVE	
15W	Economy	ACTIVE	
15X	Economy	ACTIVE	
15Y	Economy	ACTIVE	
15Z	Economy	ACTIVE	
16A	Economy	ACTIVE	
16B	Economy	ACTIVE	
16C	Economy	ACTIVE	
16D	Economy	ACTIVE	
16E	Economy	ACTIVE	
16F	Economy	ACTIVE	
16G	Economy	ACTIVE	
16H	Economy	ACTIVE	
16I	Economy	ACTIVE	
16J	Economy	ACTIVE	
16K	Economy	ACTIVE	
16L	Economy	ACTIVE	
16M	Economy	ACTIVE	
16N	Economy	ACTIVE	
16O	Economy	ACTIVE	
16P	Economy	ACTIVE	
16Q	Economy	ACTIVE	
16R	Economy	ACTIVE	
16S	Economy	ACTIVE	
16T	Economy	ACTIVE	
16U	Economy	ACTIVE	
16V	Economy	ACTIVE	
16W	Economy	ACTIVE	
16X	Economy	ACTIVE	
16Y	Economy	ACTIVE	
16Z	Economy	ACTIVE	
17A	Economy	ACTIVE	
17B	Economy	ACTIVE	
17C	Economy	ACTIVE	
17D	Economy	ACTIVE	
17E	Economy	ACTIVE	
17F	Economy	ACTIVE	
17G	Economy	ACTIVE	
17H	Economy	ACTIVE	
17I	Economy	ACTIVE	
17J	Economy	ACTIVE	
17K	Economy	ACTIVE	
17L	Economy	ACTIVE	
17M	Economy	ACTIVE	
17N	Economy	ACTIVE	
17O	Economy	ACTIVE	
17P	Economy	ACTIVE	
17Q	Economy	ACTIVE	
17R	Economy	ACTIVE	
17S	Economy	ACTIVE	
17T	Economy	ACTIVE	
17U	Economy	ACTIVE	
17V	Economy	ACTIVE	
17W	Economy	ACTIVE	
17X	Economy	ACTIVE	
17Y	Economy	ACTIVE	
17Z	Economy	ACTIVE	
18A	Economy	ACTIVE	
18B	Economy	ACTIVE	
18C	Economy	ACTIVE	
18D	Economy	ACTIVE	
18E	Economy	ACTIVE	
18F	Economy	ACTIVE	
18G	Economy	ACTIVE	
18H	Economy	ACTIVE	
18I	Economy	ACTIVE	
18J	Economy	ACTIVE	

**Create New Flight**

**Flight Route**  
Origin Airport \*  
**ESB** Esenboga - Ankara, Turkey

Destination Airport \*

**Route**  
BAK - LAX  
Showing airports except origin

**Aircraft Selection**  
Plane \*

No planes available at this airport - try deploying a plane first

**Flight Schedule**  
Quick departure time:  
**IN 2H** **IN 6H** **TOMORROW**

Departure Time \*  
03.12.2025 21:00

Local time at departure airport

Arrival Time \*  
gg.aa.yyyy --:--

Local time at arrival airport

**Pricing Configuration**  
Base Price Per Hour \*  
**10**

Seat multipliers: Economy × 1.0 | Premium Economy × 1.5 | Business × 2.0 | First Class × 3.0

**CLEAR FORM** **CANCEL** **CREATE FLIGHT**

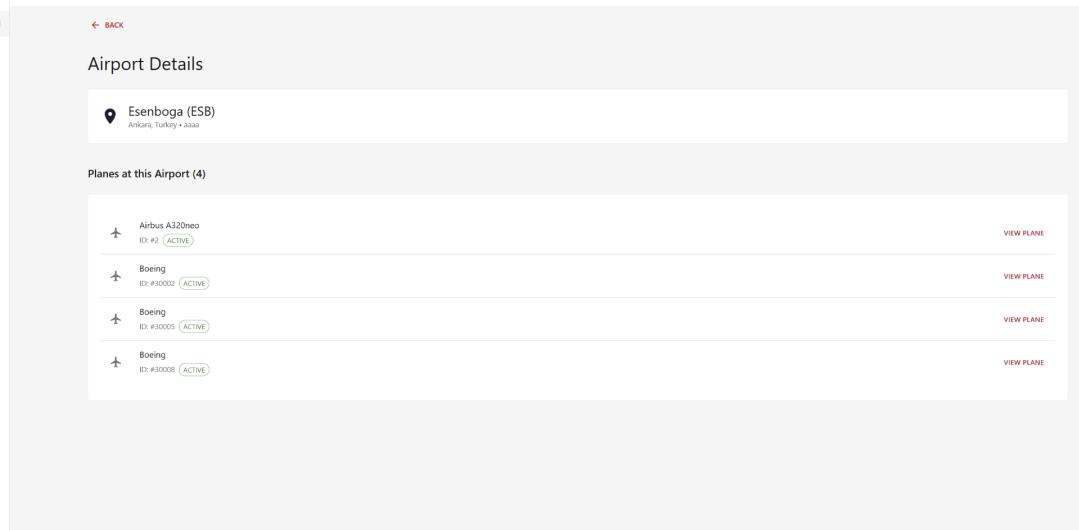
Görsel 14: Uçuş Oluşturma Ekranı

**Storage Inventory**  
List of planes currently in the hangar or storage.

**ADD PLANE**

Plane ID	Model	Status	Notes	Action
#1	Boeing 777-300ER	(RETIRED)		
#2	Airbus A320neo	(ACTIVE)		
#30001	Boeing	(ACTIVE)		
#30002	Boeing	(ACTIVE)		
#30003	Boeing	(ACTIVE)		
#30004	Boeing	(ACTIVE)		
#30005	Boeing	(ACTIVE)		
#30006	Boeing	(ACTIVE)		
#30007	Boeing	(ACTIVE)		
#30008	Boeing	(ACTIVE)		
#30009	Boeing	(ACTIVE)		

Görsel 15: Uçak Depolama Ekranı (Storage Page)



Görsel 15: Havaalanı Detay Ekranı (Airport Details Page)

## 7. Sonuç

Bu proje kapsamında, havacılık sektörünün yüksek eşzamanlılık (High Concurrency) ve veri tutarlılığı gerektiren karmaşık yapısı, ilişkisel veritabanı prensiplerine uygun olarak modellenmiştir.

Projenin en önemli kazanımları şunlardır:

- Veri Tutarlılığı:** Seat ve FlightSeat tablolarının ayrılması ve Composite Key kullanımı sayesinde, veritabanı seviyesinde veri tekrarı önlenmiş ve BCNF normalizasyon formu sağlanmıştır.
- Concurrency Yönetimi:** FlightSeat tablosuna eklenen version sütunu ve **Optimistic Locking** stratejisi sayesinde, "Çifte Rezervasyon (Double Booking)" problemi sistemsel olarak engellenmiştir.
- Performans:** Sık kullanılan sorgular (Uçuş arama, Bilet listeleme) için stratejik **İndeksler (B-Tree)** oluşturulmuş ve JOIN FETCH stratejisi ile N+1 soru problemi çözülmüştür.
- Güvenlik ve Mimari:** Spring Security ile JWT tabanlı güvenli bir mimari kurulmuş, RESTful prensiplerine sadık kalınmıştır.

Sonuç olarak; teorik veritabanı dersi kazanımları (Transaction, Normalizasyon, İndeksleme), gerçek hayat senaryosuna başarıyla uygulanmış ve ölçeklenebilir bir "Havayolu Rezervasyon Sistemi" ortaya çıkarılmıştır.

## 8. Referanslar

1. **Spring Boot Documentation (v3.3): Data Access & Transaction Management.**  
[<https://docs.spring.io/spring-boot/docs/current/reference/html/data.html>]
2. **MySQL 8.0 Reference Manual: InnoDB Locking and Transaction Model.**  
[<https://dev.mysql.com/doc/refman/8.0/en/innodb-locking-transaction-model.html>]
3. **Elmasri, R., & Navathe, S. B. (2016). Fundamentals of Database Systems (7th Edition).** Pearson.
4. **Hibernate ORM Documentation: Optimistic Locking & Versioning.**  
[<https://hibernate.org/orm/documentation/5.6/>]