

Biletim



Biletim tanıtımı

Bir bilet satış uygulaması, otobüs, uçak ve tren farklı seyahat türlerinde bilet satın almak isteyen kullanıcıların işlemlerini kolaylaştırmak için tasarlanmıştır. Bu uygulama, kullanıcıların mevcut seferleri listelemesine, bilet aramasına, rezervasyon yapmasına ve iptal işlemleri gerçekleştirmesine olanak tanır.

Uygulama, güvenilir bir veritabanı tasarımlı ve kullanıcı dostu bir arayüz ile desteklenerek kolay kullanılabilir bir çözüm sunmayı amaçlamaktadır.

Hedefler:

Farklı seyahat türleri için bilet işlemlerini bir araya toplamak.

Kullanıcıların bilet arama, satın alma, iptal etme ve güncelleme işlemlerini hızlı ve hatasız bir şekilde gerçekleştirmelerini sağlamak.

İşletmelere, sefer ve bilet bilgilerini verimli bir şekilde yönetme imkanı sunmak.



Örnek Senaryolar

1. Bir kullanıcı, sisteme giriş yaparak Ankara'dan İstanbul'a 25 Aralık 2024 tarihindeki otobüs seferini seçer. Sefer için rezervasyon yapar, ödeme işlemini tamamladıktan sonra bilet oluşturulur ve bilet detaylarını görüntüler.
2. Uygulama, tüm tren seferlerini tarih sırasına göre listeleyerek kullanıcıların arama işlemini kolaylaştırır.



İş Kuralları

1. Genel Kurallar

- Her biletin benzersiz bir ID'si olmalıdır.
- Her seferin bir türü (otobüs, uçak, tren) ve benzersiz bir ID'si olmalıdır.
- Kullanıcılar, bilet id ile arama yapabilir.
- her otobus 40 koltuğu var , 14 tekli 26 çift li.
- her tren 8 vagonu var her vagonda 30 koltuğu var.
- her uçuşta 100 koltuğu var.
- **Doğrulamalar:** Kullanıcılar yanlış veri girerse (örneğin, geçersiz bir bilet ID), sistem anlamlı bir hata mesajı döner.
- Adminler, sistemde yapılan tüm işlemleri izleyebilir ve loglayabilir.

2. Kullanıcı İşlemleri

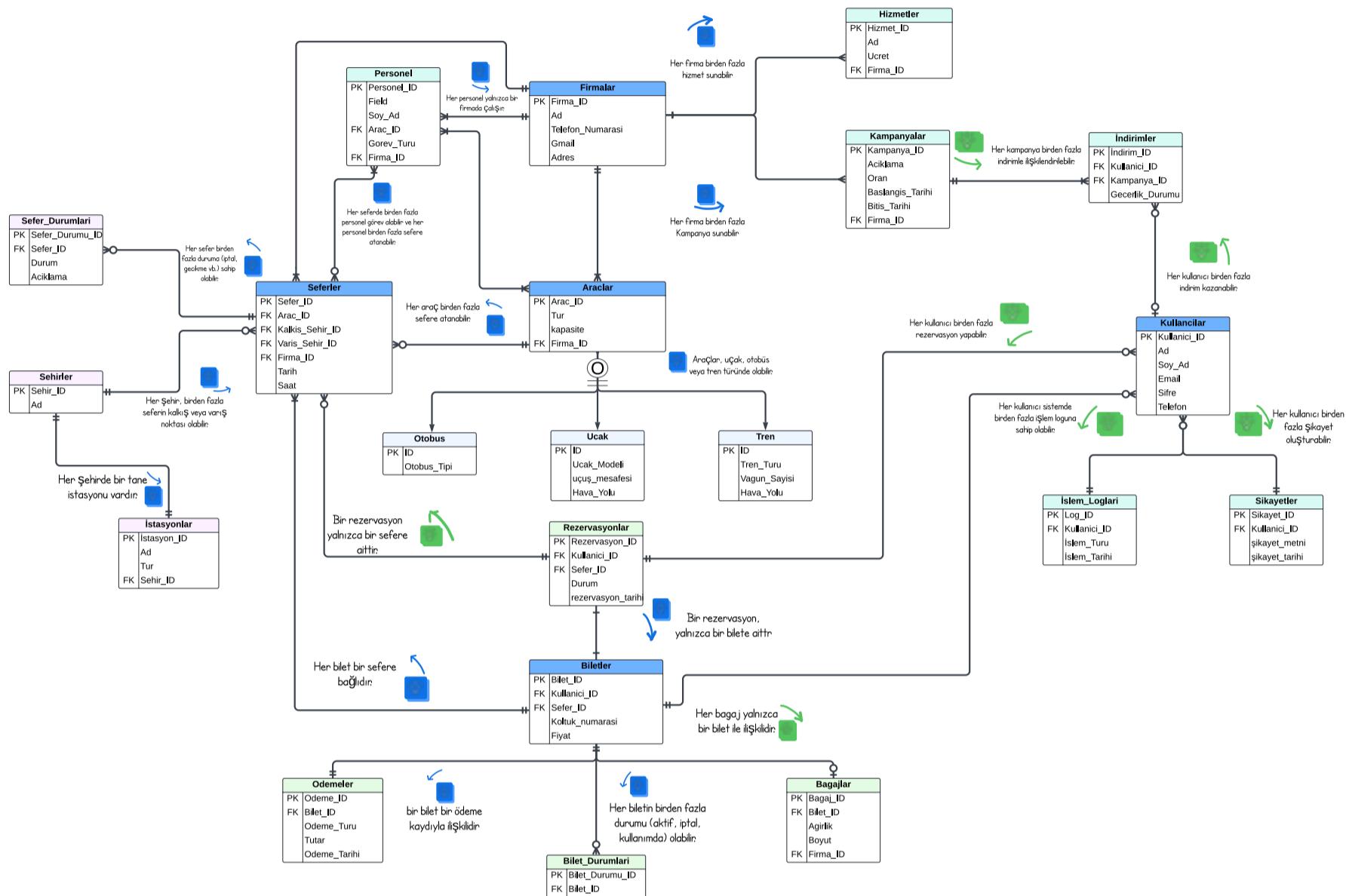
- Kullanıcılar, satın aldığıları biletleri iptal güncelleme edebilir.
- Biletlerin tüm işlemleri bilet id ile gerçekleşecek.
- Kullanıcılar satın aldığıları biletin detaylarını (sefer ID, koltuk numarası, tarih, saat, fiyat) görüntüleyebilir.
- Kullancılar ilk olarak uygun seferi seçerler sonra rezervasyon yaparlar sonra ödeme ile bilet dönüşür.
- Kullanıcılar yalnızca kendi biletlerini görebilir, iptal edebilir veya güncelleyebilir.
- Sisteme üye olmayan kullanıcılar, yalnızca sefer sorgulaması yapabilir ancak bilet satın alamaz.
- Bir kullanıcı aynı sefer için birden fazla bilet rezerve edemez.
- Ödeme sırasında hata alınırsa rezervasyon iptal edilmez, ancak yeniden ödeme yapılabilir.

İlişkisel şema

```
Kullanıcılar(kullanici_id: int PK, ad: varchar, soyad: varchar, email: varchar UNIQUE, kullanici_Sifre: varchar, telefon: varchar)
Firmalar(firma_id: int PK, ad: varchar, telefon_numarası: varchar, gmail: varchar, adres: varchar)
Araçlar(arac_id: int PK, tur: varchar, kapasite: int, firma_id: int FK)
ucaklar(ucak_id :int Pk, havayolu: varchar, business_koltuk_sayisi: int, ekonomi_koltuk_sayisi: int, araç_id: int FK)
trenler(tren_id: int PK, vagon_sayisi: int, yemekli_vagon: boolean, araç_id: int FK)
otobusler(otobus_id: int PK, wifi: boolean, araç_id: int FK)
sehirler(sehir_id: int PK, ad: varchar)
Seferler(sefer_id: int PK, araç_id: int FK, kalkis_sehir_id: int FK, varis_sehir_id: int FK, firma_id: int FK, tarih: date, saat: time)
Biletler(bilet_id: int PK, kullanici_id: int FK, sefer_id: int FK, koltuk_numarası: varchar, fiyat: decimal, rezervasyon_id: int FK)
rezervasyonlar(rezervasyon_id: int PK, kullanici_id: int FK, sefer_id: int FK, durum: varchar, rezervasyon_tarihi: timestamp)
Kampanyalar(kampanya_id: int PK, firma_id: int FK, aciklama: varchar, oran: decimal, baslangis_tarihi: date, bitis_tarihi: date)
indirimler(indirim_id: int PK, kullanici_id: int FK, kampanya_id: int FK, gecerlik_durumu: boolean)
Ödemeler(odeme_id: int PK, bilet_id: int FK, odeme_turu: varchar, tutar: decimal, odeme_tarihi: timestamp)
Bagajlar(bagaj_id: int PK, bilet_id: int FK, ağırlık: decimal, boyut: varchar, firma_id: int FK)
sikayetler(sikayet_id: int PK, kullanici_id: int FK, sikayet_metni: text, sikayet_tarihi: timestamp)
sefer_durumları(sefer_durumu_id: int PK, sefer_id: int FK, durum: varchar, aciklama: text)
bilet_durumları(bilet_durumu_id: int PK, bilet_id: int FK, durum: varchar)
```

Personel(personel_id: int PK, ad: varchar, soyad: varchar, araç_id: int FK, görev_türü: varchar, firma_id: int FK)
 Hizmetler(id: int PK, ad: varchar, soy_ad: varchar, görev_turu: varchar, firma_id: int FK)
 İşlem_Logları(log_id: int PK, kullanıcı_id: int FK, işlem_turu: varchar, işlem_tarihi: timestamp)
 İstasyonlar(id: int PK, ad: varchar, tur: varchar, şehir_id: int FK)

Varlık Bağıntı



Veritabanını oluşturmayı sağlayan SQL ifadeleri :

-- Kullanıcılar Tablosu

CREATE TABLE Kullanıcılar

(kullanici_id SERIAL PRIMARY KEY, ad VARCHAR(100), soyad VARCHAR(100), email VARCHAR(100) UNIQUE, kullanıcı_sifre VARCHAR(100), telefon VARCHAR(15));

-- Araçlar Tablosu

CREATE TABLE Araclar

(arac_id SERIAL PRIMARY KEY, tur VARCHAR(50), kapasite INT, firma_id INT REFERENCES Firmalar(firma_id));

-- Uçaklar Tablosu

CREATE TABLE Uçaklar

```

(ucak_id SERIAL PRIMARY KEY, havayolu VARCHAR(100), business_koltuk_sayisi INT, ekonomi_koltuk_sayi
si INT, arac_id INT REFERENCES Araçlar(arac_id));

-- Trenler Tablosu
CREATE TABLE Trenler
(tren_id SERIAL PRIMARY KEY, vagon_sayisi INT, yemekli_vagon BOOLEAN, arac_id INT REFERENCES Araç
lar(arac_id));

-- Otobüsler Tablosu
CREATE TABLE Otobüsler
(otobus_id SERIAL PRIMARY KEY, wifi BOOLEAN, arac_id INT REFERENCES Araçlar(arac_id));

-- Şehirler Tablosu
CREATE TABLE Şehirler
(sehir_id SERIAL PRIMARY KEY, ad VARCHAR(100));

-- Seferler Tablosu
CREATE TABLE Seferler
(sefer_id SERIAL PRIMARY KEY, arac_id INT REFERENCES Araçlar(arac_id), kalkis_sehir_id INT REFERENC
ES Şehirler(sehir_id), varis_sehir_id INT REFERENCES Şehirler(sehir_id), firma_id INT REFERENCES Firmalar(f
irma_id), tarih DATE, saat TIME);

-- Biletler Tablosu
CREATE TABLE Biletler
(bilet_id SERIAL PRIMARY KEY, kullanici_id INT REFERENCES Kullanıcılar(kullanici_id), sefer_id INT REFERE
NCES Seferler(sefer_id), koltuk_numarasi VARCHAR(10), fiyat DECIMAL(10,2), rezervasyon_id INT REFERENC
ES Rezervasyonlar(rezervasyon_id));

-- Rezervasyonlar Tablosu
CREATE TABLE Rezervasyonlar
(rezervasyon_id SERIAL PRIMARY KEY, kullanici_id INT REFERENCES Kullanıcılar(kullanici_id), sefer_id INT
REFERENCES Seferler(sefer_id), durum VARCHAR(50), rezervasyon_tarihi TIMESTAMP);

-- Kampanyalar Tablosu
CREATE TABLE Kampanyalar
(kampanya_id SERIAL PRIMARY KEY, firma_id INT REFERENCES Firmalar(firma_id), aciklama VARCHAR(25
5), oran DECIMAL(5,2), baslangis_tarihi DATE, bitis_tarihi DATE);

-- İndirimler Tablosu
CREATE TABLE İndirimler
(indirim_id SERIAL PRIMARY KEY, kullanici_id INT REFERENCES Kullanıcılar(kullanici_id), kampanya_id INT
REFERENCES Kampanyalar(kampanya_id), gecerlik_durumu BOOLEAN);

-- Ödemeler Tablosu
CREATE TABLE Ödemeler
(odeme_id SERIAL PRIMARY KEY, bilet_id INT REFERENCES Biletler(bilet_id), odeme_turu VARCHAR(50), t
utar DECIMAL(10,2), odeme_tarihi TIMESTAMP);

-- Bagajlar Tablosu
CREATE TABLE Bagajlar
(bagaj_id SERIAL PRIMARY KEY, bilet_id INT REFERENCES Biletler(bilet_id), ağırlık DECIMAL(5,2), boyut V
ARCHAR(50), firma_id INT REFERENCES Firmalar(firma_id));

-- Şikayetler Tablosu
CREATE TABLE Şikayetler
(sikayet_id SERIAL PRIMARY KEY, kullanici_id INT REFERENCES Kullanıcılar(kullanici_id), sikayet_methi TE
XT, sikayet_tarihi TIMESTAMP);

```

```
-- Sefer Durumları Tablosu
CREATE TABLE Sefer_Durumları
    (sefer_durumu_id SERIAL PRIMARY KEY, sefer_id INT REFERENCES Seferler(sefer_id), durum VARCHAR(50), aciklama TEXT);

-- Bilet Durumları Tablosu
CREATE TABLE Bilet_Durumları
    (bilet_durumu_id SERIAL PRIMARY KEY, bilet_id INT REFERENCES Biletler(bilet_id), durum VARCHAR(50));

-- Personel Tablosu
CREATE TABLE Personel
    (personel_id SERIAL PRIMARY KEY, ad VARCHAR(100), soyad VARCHAR(100), araç_id INT REFERENCES Araclar(arac_id), görev_türü VARCHAR(100), firma_id INT REFERENCES Firmalar(firma_id));

-- Hizmetler Tablosu
CREATE TABLE Hizmetler
    (id SERIAL PRIMARY KEY, ad VARCHAR(100), soy_ad VARCHAR(100), gorev_turu VARCHAR(100), firma_id INT REFERENCES Firmalar(firma_id));

-- İşlem Logları Tablosu
CREATE TABLE İşlem_Logları
    (log_id SERIAL PRIMARY KEY, kullanici_id INT REFERENCES Kullanıcılar(kullanici_id), işlem_turu VARCHAR(100), işlem_tarihi TIMESTAMP);

-- İstasyonlar Tablosu
CREATE TABLE İstasyonlar
    (id SERIAL PRIMARY KEY, ad VARCHAR(100), tur VARCHAR(50), sehir_id INT REFERENCES Şehirler(sehir_id));
```

veri ekleme

```
DO $$

DECLARE
    baslangic_tarih DATE := '2024-12-21'; -- Bugün tarihi
    bitis_tarih DATE := '2025-01-05'; -- 15 gün sonrası
    mevcut_tarih DATE := baslangic_tarih;
    kalkis_sehir_id INT;
    varis_sehir_id INT;
    firma_id INT;
    arac_id INT;
    saat TIME;
BEGIN
    -- Tarih döngüsü
    WHILE mevcut_tarih <= bitis_tarih LOOP
        -- Örnek şehir ID'leri arasında sefer oluştur
        FOR kalkis_sehir_id, varis_sehir_id IN
            SELECT * FROM (VALUES (34, 6), (6, 34), (1, 34), (34, 1), (6, 1), (1, 6)) AS sehirler(kalkis, varis)
        LOOP
            -- Firmaları ve araçları rastgele seç
            firma_id := (SELECT FLOOR(RANDOM() * 3 + 1)::INT); -- 1-3 arasında firma
            arac_id := (SELECT FLOOR(RANDOM() * 3 + 1)::INT); -- 1-3 arasında araç
            saat := (SELECT TIME '06:00:00' + (INTERVAL '1 hour' * FLOOR(RANDOM() * 15))); -- 06:00 ile 21:00
            arası rastgele saat

            -- Sefer ekle
            INSERT INTO Seferler (arac_id, kalkis_sehir_id, varis_sehir_id, firma_id, tarih, saat)
            VALUES (arac_id, kalkis_sehir_id, varis_sehir_id, firma_id, mevcut_tarih, saat);
```

```
END LOOP;

-- Tarihi bir gün artır
mevcut_tarih := mevcut_tarih + INTERVAL '1 day';
END LOOP;
END $$;
```

fonksiyonlar :

1. Fonksiyon: kontrol_rezervasyon

kontrol_rezervasyon(p_rezervasyon_id INT,p_kullanici_id INT)

İşlev:

- Belirtilen rezervasyonun durumunu kontrol eder.
- Eğer rezervasyon bulunamazsa veya durumu "Ödendi" ise **FALSE**, aksi halde **TRUE** döndürür.
- Kullanıcı ve rezervasyon ID'sini doğrular.

2. Fonksiyon: rezervasyonu_bilete_donustur

rezervasyonu_bilete_donustur(p_rezervasyon_id INT)

İşlev:

- Verilen rezervasyonu bilet tablosuna taşıır (bilet oluşturur).
- Rezervasyonun durumunu "Ödendi" olarak günceller.
- Oluşturulan yeni biletin ID'sini döndürür.

3. Fonksiyon: bilet_iodemeye_ekle (p_bilet_id INT, p_odeme_turu TEXT)

İşlev:

- Verilen bilet için ödeme tablosuna bir ödeme kaydı ekler.
- Ödeme türü, tutar ve ödeme tarihi gibi bilgileri ekler.

4. Fonksiyon: kullaniciya_ait_biletler (p_kullanici_id INT)

İşlev:

- Belirtilen kullanıcıya ait tüm biletleri döndürür.
- Dönen tablo; bilet ID'si, sefer ID'si, fiyat ve rezervasyon ID'sini içerir.

5. Fonksiyon: bilet_iptali (p_bilet_id INT,p_kullanici_id INT)

İşlev:

- Verilen biletin ilgili kullanıcıya ait olup olmadığını kontrol eder.
- Kullanıcıya ait değilse hata fırlatır.
- İlk olarak bilet ile ilişkili ödemeyi, ardından bilet kaydını siler.
- Başarılı işlem sonrası mesaj gösterir.

6. Bilet Değiştirme Fonksiyonu

bilet_degistir(p_kullanici_id INT, p_bilet_id INT, p_yeni_kalkis_sehir_id INT, p_yeni_varis_sehir_id INT, p_yeni_sefer_tarihi DATE)

Amaç:

- Kullanıcının mevcut biletini belirtilen yeni kalkış şehri, varış şehri ve sefer tarihine göre güncellemek.

İşlevler:

1. Biletin Kullanıcıya Ait Olup olmadığını Kontrol Eder:

- Eğer bilet belirtilen kullanıcıya ait değilse hata mesajı döner.

2. Yeni Sefer Bilgilerine Göre Bilet Günceller:

- Verilen kriterlere (kalkış şehri, varış şehri, sefer tarihi) uygun bir sefer aranır.
- Sefer bulunamazsa hata mesajı verir.

7. Gelişmiş Kullanıcıya Ait Biletleri Listeleme Fonksiyonu

kullaniciya_ait_biletler(p_kullanici_id INT)

Amaç:

- Kullanıcıya ait biletlerin detaylı bir listesini döndürmek. Liste, şehir adları ve sefer tarihi gibi ek bilgiler içerir.

İşlevler:

1. Kullanıcıya Ait Biletleri Seçer:

- Kullanıcı ID'sine göre biletleri filtreler.

2. Detaylı Bilgi Döner:

- Bilet bilgilerine ek olarak kalkış ve varış şehir ID'lerini, şehir isimlerini ve sefer tarihini döndürür.

3. Birden Fazla Tabloyu Birleştirir:

- `biletler`, `seferler` ve `sehirler` tablolarından gerekli verileri toplar ve ilişkileri kullanarak birleştirir.

8. UygunSeferler(kalkis INT, varis INT, aramaTarihi DATE)

Amaç: Kalkış ve varış şehirlerine, arama tarihine göre uygun seferleri listelemek.

İşlevler:

- Seferler tablosundan kalkış ve varış şehir ID'lerine göre uygun kayıtları getirir.
- Tarihi eşleşen seferleri filtreler.
- Firma adı, kalkış ve varış şehir adları, tarih ve saat bilgileri ile birlikte sonucu döndürür.

9. ekle_kullanici(p_ad VARCHAR, p_soyad VARCHAR, p_email VARCHAR, p_sifre VARCHAR, p_telefon VARCHAR)

Amaç: Yeni bir kullanıcıyı veritabanına eklemek.

İşlevler:

- Kullanıcı bilgilerini (`ad`, `soyad`, `email`, `sifre`, `telefon`) kullanarak `kullanicilar` tablosuna yeni bir kayıt ekler.

10. get_kullanici_id(p_ad VARCHAR, p_sifre VARCHAR)

Amaç: Kullanıcının ad ve şifresine göre kullanıcı ID'sini döndürmek.

İşlevler:

- `kullanicilar` tablosundan ad ve şifre bilgileri eşleşen kullanıcıyı sorgular.
- Eşleşen kullanıcı varsa ID bilgisini döndürür, yoksa `NULL` döner.

11. KontrolSeferID(kalkis INT, varis INT, aramaTarihi DATE, secilenSeferID INT)

Amaç: Kullanıcının seçtiği seferin kalkış, varış ve tarihle eşleşip eşleşmediğini kontrol etmek.

İşlevler:

- Seçilen `SeferID` nin kalkış, varış ve tarih bilgileri ile uyumlu olup olmadığını `EXISTS` sorgusu ile kontrol eder.
- Uygun bir eşleşme varsa `TRUE`, aksi halde `FALSE` döndürür.

12. ekle_rezervasyon(p_kullanici_id INT, p_sefer_id INT)

Amaç: Kullanıcının seçtiği sefer için rezervasyon oluşturmak.

İşlevler:

- rezervasyonlar tablosuna kullanıcı ID'si ve sefer ID'si ile yeni bir rezervasyon ekler.
- Rezervasyon durumu varsayılan olarak 'Beklemede' şeklinde ayarlanır.
- Rezervasyon tarihi otomatik olarak CURRENT_TIMESTAMP ile eklenir.

13. get_rezervasyonlar(p_kullanici_id INT)

Amaç: Belirlenen kullanıcı ID'sine ait rezervasyonları listelemek.

İşlevler:

- Kullanıcı ID'sine göre rezervasyonlar tablosundan ilgili rezervasyonları çeker.
- Rezervasyonların durum, tarih, kalkış ve varış şehir bilgileriyle birlikte sefer tarihi ve saatini döndürür.
- rezervasyonlar ve seferler tablolarını birleştirerek detaylı bir tablo oluşturur.

14. SikayetEkle(p_sikayet_metni)

Kullanıcının şikayetlerini veritabanına eklemek için bir kolaylık sağlar.

İşleyiş:

1. Parametreler:

- Kullanıcı ID (p_kullanici_id) ve şikayet metni (p_sikayet_metni) alınır.

2. Fonksiyonun Görevi:

- Alınan bilgilerle sikayetler tablosuna yeni bir kayıt ekler.
- Şikayetin eklenme zamanı otomatik olarak kayıt edilir (CURRENT_TIMESTAMP).

3. Sonuç:

- Kullanıcı şikayeti tablodaki ilgili alanlara eklenir.

Tetikleyiciler

1. BiletLogTrigger

• Fonksiyon Tanımı

YeniBiletLog() adında bir fonksiyon tanımlanmıştır. Bu fonksiyon bir tetikleyici (trigger) ile ilişkilendirilerek çalıştırılır.

• Amaç

Biletler tablosuna yeni bir kayıt eklendiğinde (yeni bir bilet alındığında), Islem_Loglari tablosuna otomatik olarak bir işlem kaydı eklemek.

• İşleyiş Detayları

- NEW.Kullanici_ID ile yeni eklenen biletin kullanıcısının ID'si alınır.
- "Bilet Alındı" mesajı ve işlem tarihi, CURRENT_TIMESTAMP ile kaydedilir.
- INSERT INTO ile bu bilgiler Islem_Loglari tablosuna eklenir.

• Trigger Tanımı

- BiletLogTrigger adında bir tetikleyici tanımlanmıştır.
- AFTER INSERT ifadesi ile, Biletler tablosuna her yeni ekleme işleminden sonra tetiklenir.
- Tetikleyici, tanımlanan YeniBiletLog() fonksiyonunu çalıştırır.

• Sonuç

Bu yapı, manuel bir işlem gerekmeden, yeni bir bilet alındığında otomatik bir log kaydının oluşturulmasını sağlar.

2. sefer_durumu_ekle

Seferler tablosuna yeni bir kayıt eklendiğinde, Sefer_Durumları tablosuna otomatik olarak bir "Planlandı" durumu eklenmesi için bir fonksiyon tanımlanır.

Fonksiyonun Dönüş Değeri

- **Kod:** `RETURN NEW;`
- **Açıklama:** İşlem tamamlandıktan sonra tetikleme olayının çalışmasına devam etmesi sağlanır.

Trigger Tanımı

- **Kod:** `CREATE TRIGGER sefer_eklenince_durum_olustur`
- **Açıklama:** Seferler tablosunda yeni bir kayıt eklendiğinde, tetikleyici otomatik olarak sefer_durumu_ekle fonksiyonunu çalıştırır.

3. bilet_durumu_ekle

- **Amaç:** Yeni eklenen bir bilet için `bilet_durumları` tablosuna bir durum kaydı eklemek.
- **İşleyiş:**
 - `biletler` tablosuna bir kayıt eklendiğinde tetiklenir.
 - Yeni eklenen biletin `bilet_id` bilgisi alınarak `bilet_durumları` tablosuna bir durum ("Oluşturuldu") kaydı ekler.
- **Kullanılan Tablo:** `bilet_durumları`
- **Eklenen Veri:**
 - `bilet_id`: Yeni eklenen biletin ID'si.
 - `durum`: Biletin durumu (örneğin, "Oluşturuldu").

Tetikleyici: `bilet_eklenince_durum_olustur`

- **Amaç:** `biletler` tablosuna yeni bir kayıt eklendiğinde otomatik olarak `bilet_durumu_ekle()` fonksiyonunu çalışırmak.
- **İşleyiş:**
 - `AFTER INSERT` tetikleyicisi olarak çalışır.
 - `biletler` tablosuna yeni bir kayıt eklendiği her seferde, `bilet_durumu_ekle()` fonksiyonunu çağırır.
- **Bağılı Tablo:** `biletler`
- **Fonksiyon:** `bilet_durumu_ekle()` fonksiyonunu tetikler.

4. YeniSikayetLog

Bu fonksiyonun amacı,
`şikayetler` tablosuna yeni bir kayıt eklendiğinde bu işlemi takip etmek ve işlemle ilgili bir log kaydı oluşturmaktır.

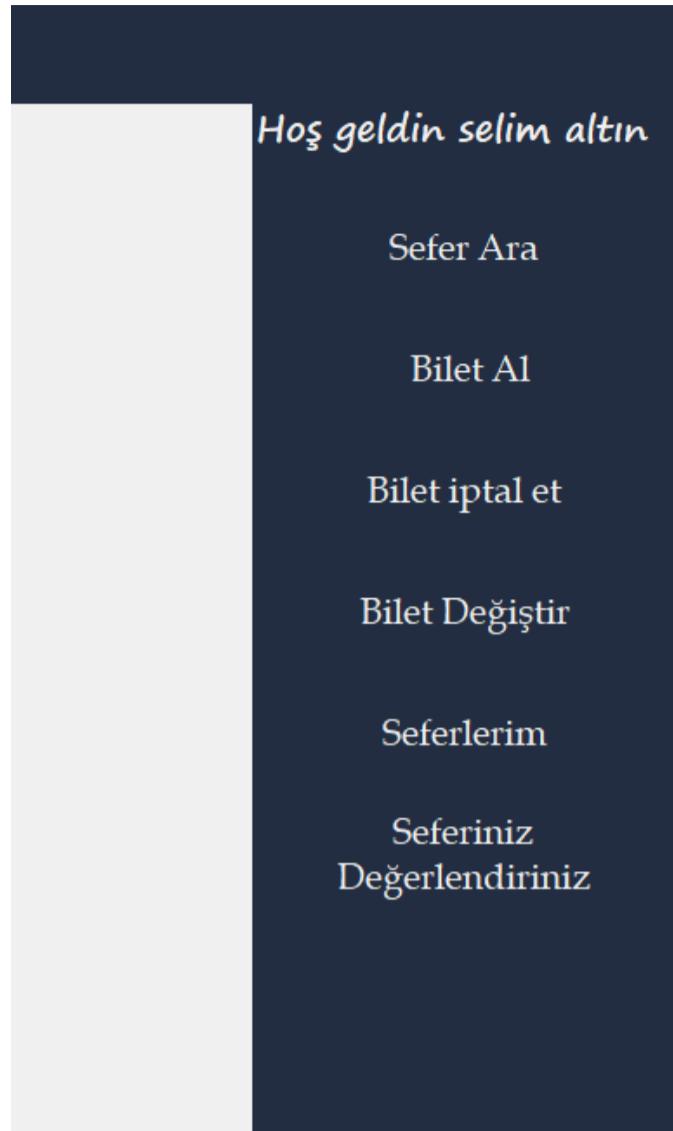
İşleyiş:

1. **YeniSikayetLog Fonksiyonu:**
 - Şikayetler tablosuna eklenen kayıttaki **Kullanıcı_ID** ve işlem türü ("Şikayet Eklendi") ile birlikte, işlem zamanını `islem_Loglari` tablosuna ekler.
2. **SikayetLogTrigger Tetikleyici:**
 - Şikayetler tablosunda bir **INSERT (ekleme)** işlemi gerçekleştiğinde, bu tetikleyici devreye girer ve otomatik olarak **YeniSikayetLog** fonksiyonunu çalıştırır.

Arama işlemi ekran görüntüleri :

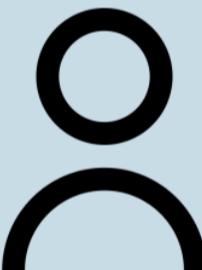
Hesap Bilgilerinizi Giriniz

Kullancı Adı	<input type="text" value="selim"/>	<input type="button" value="Giriş"/>
Şifre	<input type="text" value="Salam1234"/>	



Ekleme işlemi ekran görüntüleri :

Hesap Bilgileriniz Giriniz

	Ad : <input type="text" value="abdulselam"/>
	Soyad : <input type="text" value="almasri"/>
	Gmail : <input type="text" value="abd933231@gmail.com"/>
	Telefon : <input type="text" value="05066710818"/>
	Şifre : <input type="text" value="123"/>
	<input type="button" value="Kaydet"/>

Aramak istediğiniz sefer bilgileri giriniz

Nerden : <input type="text" value="Adana"/>		Nereye : <input type="text" value="Ankara"/>	Tarih : <input type="text" value="24 Aralık 2024 Salı"/>	Siyahet Türü <input checked="" type="radio"/>  <input type="radio"/>  <input type="radio"/> 
<input type="button" value="Uygun Seferler Ara"/>				

seferid	firmaad	kalkisşehir	varisşehir	tarih	saat
50	123 Tren	Adana	Ankara	24.12.2024	20:00:00
*					

rezervasyon id : 50

Silme işlemi ekran görüntüleri :

Hoş geldin abdulselam a

rezervasyon_id	bilet_id	kullanici_id	fiyat	kalkis_sehir_id	varis_sehir_id	sefer_tarihi
22	47	13	100,00	1	6	24.12.2024
*						

Bilet id :

- Sefer Ara
- Bilet Al
- Bilet iptal et
- Bilet Değiştir
- Seferlerim
- Seferiniz Değerlendiriniz

Güncelleme işlemi ekran görüntüleri :

yeni sefer bilgileriniz giriniz

Nerden : Nereye : Tarih : Siyahet Türü

rezervasyon_id	bilet_id	kullanici_id	fiyat	kalkis_sehir_id	varis_sehir_id	sefer_tarihi
24	49	13	100,00	6	34	30.01.2025
*						

Bilet id :

- Sefer Ara
- Bilet Al
- Bilet iptal et
- Bilet Değiştir
- Seferlerim
- Seferiniz Değerlendiriniz

yeni sefer bilgileriniz giriniz

Nerden : Nereye : Tarih : Siyahet Türü

rezervasyon_id	bilet_id	kullanici_id	fiyat	kalkis_sehir_id	varis_sehir_id	sefer_tarihi
24	49	13	100,00	6	1	2.01.2025
*						

Bilet id :

- Sefer Ara
- Bilet Al
- Bilet iptal et
- Bilet Değiştir
- Seferlerim
- Seferiniz Değerlendiriniz

Uygulamanın kaynak kodları :

<https://github.com/sla55m/Biletim>