ANKARA ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

Etkinlik Bilet Satış Sistemi

Final Raporu

BLM4537 – IOS ile Mobil Uygulama Geliştirme I

Tarih: 17.01.2026

Video linki: https://drive.google.com/file/d/1vdy8cJBbFrqlxSarkMDJho7jTEzLLYLt/view?usp=sharing

İÇİNDEKİLER

1. Veritabanı Tabloları

Sistemde toplamda 3 ana tablo bulunmaktadır. Veritabanı olarak PostgreSQL kullanılmış, ORM aracı olarak Entity Framework Core tercih edilmiştir.

1.1. Users Tablosu

Kullanıcı bilgilerini saklar. Şifreler BCrypt algoritması kullanılarak hashlenmiş olarak saklanmaktadır.

	Id [PK] integer	Username text	Email text	Password text
1	1	string	string	$2a$11$koxIa0fTFf8PoprEMjeudeH9r8E3BwFH2XGoxLrspWjNdrfNpadmW
2	2	ozaan	kroxsannn@hotmail.c...	$2a$11$IkHi55FvcpaRak8WC8NSE.ApP5sEhMnnFaDEbTBHusOEI.Xfy0W...
3	3	ozan1	ozan@hotmail.com	$2a$11$r4Qani3T0WyuV/qEk2am7uZ601kSPFoj0LJoi42YaMOLnfqtCjVBu
4	4	test1	test@hotmail.com	$2a$11$nr0IpyzItiiiGLF25yRQbemhNscNxZDkkqPe5blmesKItFAvWRszC
5	8	test2	test2@hotmail.com	$2a$11$Rf6cXLGSVocuq1g0NIuXWunACqo65it6//ZAjtuNqVvNK1rqdUez6
6	10	admin1	kroxsan@hotmail.com	$2a$11$uuCZ/wQMI9YYHKo6996WPOsqK9FT5..AVzE9DxgQIF01V4wGiz...

Özellikler:

- Username ve Email alanları unique constraint'e sahiptir

- Şifreler BCrypt.Net kütüphanesi ile hashlenmiştir

- Admin yetkisi username bazlıdır (username: "admin1")

1.2. Events Tablosu

Etkinlik bilgilerini saklar. Her etkinlik bir kullanıcı tarafından oluşturulur.

	Id [PK] integer	Name text	Description text	Category text	City text	Price double precision	Date timestamp with time zone	ImageUrl text	Location text	Capacity integer	UserId integer
1	24	test	dfgdfg	Müzik	Ankara	30	2025-11-20 23:00:00+03	https://i.ytimg.com/vi/SJm5suVpOK0/maxresdefault....	Ankara	350	10
2	25	test1	sdfsdf	Müzik	Ankara	400	2025-11-20 23:00:00+03	https://i.ytimg.com/vi/SJm5suVpOK0/maxresdefault....	Ankara	300	10
3	27	test5	dsfdsf	Müzik	İstanb...	250	2025-11-20 23:00:00+03	https://i.ytimg.com/vi/SJm5suVpOK0/maxresdefault....	İstanbul	250	10

İlişkiler:

- Her etkinlik bir kullanıcıya aittir (Many-to-One)

- Kategori değerleri: Müzik, Komedi, Tiyatro vb.

Özel Özellik:

- Kalan kapasite dinamik olarak hesaplanır: `Capacity - SUM(Tickets.Quantity)`

- DateTime değerleri UTC formatında saklanır

1.3. Tickets Tablosu

Satın alınan biletleri saklar. Her bilet bir kullanıcıya ve bir etkinliğe aittir.

	Id [PK] integer	UserId integer	EventId integer	PurchaseDate timestamp with time zone	Quantity integer	TotalPrice double precision
1	53	10	24	2026-01-16 15:36:25.645452+...	1	30

İlişkiler:

- Her bilet bir kullanıcıya aittir (Many-to-One)

- Her bilet bir etkinliğe aittir (Many-to-One)

1.4. Cascade Delete Davranışı

Veritabanında Cascade Delete politikası uygulanmıştır:

- Kullanıcı silindiğinde: İlgili tüm biletler otomatik silinir

- Etkinlik silindiğinde: İlgili tüm biletler otomatik silinir

Bu veri bütünlüğünü korumak için kritik öneme sahiptir.

2. Kullanıcı Kayıt ve Giriş Sayfası

Kayıt (Register) İşlemi

- Kullanıcı adı, email ve şifre ile kayıt olunur

- Email ve username unique olmalıdır

- Şifre BCrypt ile hashlenir

- Backend endpoint: `POST /api/auth/register`

Giriş (Login) İşlemi

- Username ve şifre ile giriş yapılır

- Başarılı girişte JWT token üretilir

- Token'da kullanıcı bilgileri (ID, username, email) bulunur

- Token 2 saat geçerlidir

- Backend endpoint: `POST /api/auth/login`

Güvenlik:

- Plain text şifre saklanmaz

- BCrypt work factor: default (10)

- JWT token HS256 algoritması ile imzalanır

- Token SharedPreferences'da saklanır (Flutter)

Özellikler:

- Giriş yapan kullanıcı ana sayfaya yönlendirilir

- Form validasyonu mevcuttur

- Hata mesajları kullanıcıya gösterilir

- Loading state implementasyonu var

3. Ana Sayfa (Home Page)

3.1. Etkinlik Listesi

- Tüm etkinlikler grid formatında gösterilir
- Her etkinlik kartı şunları içerir:
  - Etkinlik görseli
  - Etkinlik adı
  - Kategori
  - Şehir
  - Fiyat
  - Tarih
  - Kalan kapasite

3.2. Arama ve Filtreleme

Arama:

- Etkinlik adına göre arama yapılabilir
- Real-time arama (her karakter girişinde)

Filtreler:

- Kategori filtresi: Tümü, Müzik, Komedi, Tiyatro
- Şehir filtresi: Tümü, İstanbul, Ankara, İzmir
- Filtre değişiminde sayfa otomatik 1'e döner

3.3. Sayfalama (Pagination)

- Sayfa başına 5 etkinlik gösterilir
- İleri/geri navigasyon butonları
- Mevcut sayfa ve toplam sayfa sayısı gösterilir
- Responsive tasarım: Mobilde 1 sütun, büyük ekranlarda 3 sütun

3.4. Navbar (Gezinme Çubuğu)

- Ana Sayfa linki

- Giriş Yap/Çıkış Yap butonu
- Admin kullanıcı için "Admin Paneli" butonu
- Giriş yapmış kullanıcı için "Biletlerim" butonu
- Kullanıcı adı gösterimi

4. Etkinlik Detay Sayfası

Bir etkinlik kartına tıklandığında açılan detay sayfası:

Gösterilen Bilgiler:

- Büyük etkinlik görseli
- Etkinlik adı
- Detaylı açıklama
- Kategori badge'i
- Şehir ve tam adres
- Tarih ve saat
- Bilet fiyatı
- Kalan kapasite

Bilet Satın Alma:

- Bilet adedi seçimi
- Toplam tutar hesaplaması (adet × fiyat)
- "Bilet Satın Al" butonu
- Kapasite kontrolü (yeterli bilet yoksa uyarı)

İşlem Akışı:

1. Kullanıcı bilet adedi seçer
2. Toplam tutar otomatik hesaplanır
3. "Satın Al" butonuna tıklar
4. Backend'e POST isteği gönderilir
5. Kapasite kontrolü yapılır
6. Başarılıysa bilet oluşturulur

7. Kullanıcıya bildirim gösterilir

Validasyonlar:

- Giriş yapmamış kullanıcı bilet alamaz
- Kapasite aşımı engellenir
- Minimum 1, maksimum 10 bilet alınabilir

5. Admin Panel Sayfası

5.1. Erişim Kontrolü

- Sadece admin kullanıcı (username: "admin1") erişebilir
- Yetkisiz erişim denemelerinde ana sayfaya yönlendirme
- JWT token kontrolü

5.2. Etkinlik Ekleme Formu

Form Alanları:

- Etkinlik Adı (zorunlu)
- Açıklama (çok satırlı)
- Kategori
- Şehir
- Konum/Adres
- Fiyat (numerik)
- Görsel URL
- Kapasite (zorunlu, numerik)
- Tarih (YYYY-MM-DD formatında, zorunlu)

Özellikler:

- Form validasyonu
- Boş alanlar için varsayılan değerler
- Konum boşsa otomatik "Şehir, Konum Belirtilmemiş" ataması

5.3. Etkinlik Güncelleme

- Mevcut etkinlik bilgileri forma yüklenir
- Düzenleme modunda "Güncelle" butonu görünür
- İptal butonu ile form temizlenebilir
- Backend endpoint: `PUT /api/events/{id}`

5.4. Etkinlik Silme

- Her etkinlik kartında "Sil" ikonu
- Silme işlemi sonrası liste otomatik yenilenir
- Cascade delete ile ilgili biletler de silinir
- Backend endpoint: `DELETE /api/events/{id}`

5.5. Etkinlik Listesi

- Tüm etkinlikler liste formatında
- Küçük önizleme görseli
- Etkinlik adı, şehir ve tarih bilgisi
- Her satırda düzenle ve sil butonları

6. Biletlerim Sayfası

6.1. Bilet Listesi

Kullanıcının satın aldığı tüm biletler gösterilir:

Her Bilet Kartı İçeriği:

- Etkinlik görseli
- Etkinlik adı
- Konum bilgisi
- Etkinlik tarihi
- Bilet adedi
- Toplam ödenen tutar
- Satın alma tarihi

- Bilet iptal butonu

6.2. Özellikler

- Pull-to-refresh desteği
- Responsive kart tasarımı
- Konum boş ise otomatik "Şehir, Konum Belirtilmemiş" gösterimi
- Görsel yüklenemezse placeholder ikon
- Backend endpoint: `GET /api/tickets/my-tickets`

6.3. Bilet İptali

- Her bilet için "Bileti İptal Et" butonu
- Onay dialogu ile doğrulama
- İptal işlemi sonrası liste yenilenir
- Backend endpoint: `DELETE /api/tickets/{id}`

Güvenlik:

- Sadece kendi biletlerini görebilir/iptal edebilir
- JWT token ile yetkilendirme
- UserId kontrolü backend'de yapılır

7. Teknik Detaylar

7.1. Backend (ASP.NET Core Web API)

Kullanılan Teknolojiler:

- ASP.NET Core 9.0
- Entity Framework Core 9.0
- PostgreSQL (Npgsql)
- JWT Bearer Authentication
- BCrypt.Net-Next
- Swagger/OpenAPI

Proje Yapısı:

```
site_backend/
├── Controllers/
│   ├── AuthController.cs      (Kayıt/Giriş)
│   ├── EventsController.cs    (Etkinlik CRUD)
│   └── TicketsController.cs   (Bilet işlemleri)
├── Models/
│   ├── User.cs
│   ├── Event.cs
│   └── Ticket.cs
├── Data/
│   └── AppDbContext.cs        (EF Core DbContext)
├── Migrations/                (Veritabanı migration'ları)
└── Program.cs                 (Konfigürasyon)
```

API Endpoints:

Authentication:

- `POST /api/auth/register` - Kayıt ol
- `POST /api/auth/login` - Giriş yap

Events:

- `GET /api/events` - Tüm etkinlikleri listele
- `GET /api/events/{id}` - Etkinlik detayı
- `POST /api/events` - Yeni etkinlik oluştur (Auth required)
- `PUT /api/events/{id}` - Etkinlik güncelle (Auth required)
- `DELETE /api/events/{id}` - Etkinlik sil (Auth required)

Tickets:

- `GET /api/tickets/my-tickets` - Kullanıcının biletleri (Auth required)

- `POST /api/tickets` - Bilet satın al (Auth required)

- `DELETE /api/tickets/{id}` - Bilet iptal et (Auth required)

Önemli Özellikler:

- CORS politikası: Tüm origin'lere izin (development)

- JWT token 2 saat geçerli

- Kapasite kontrolü transaction içinde

- DateTime UTC formatında

- Cascade delete ilişkileri

7.2. Frontend (Flutter Web)

Kullanılan Teknolojiler:

- Flutter

- Dart

- Provider (State Management)

- HTTP package

- SharedPreferences (Token storage)

Proje Yapısı:

site_frontend/lib/

├── main.dart (Entry point)

├── data/

│ ├── event.dart (Event model)

│ └── mock_events.dart (Mock data - kullanılmıyor)

├── providers/

```
|   ├── auth_provider.dart    (Authentication state)
|   └── event_provider.dart   (Event state & API calls)
├── pages/
|   ├── home_page.dart        (Ana sayfa)
|   ├── login_page.dart       (Giriş sayfası)
|   ├── register_page.dart    (Kayıt sayfası)
|   ├── admin_panel_page.dart (Admin paneli)
|   ├── my_tickets_page.dart  (Biletlerim)
|   └── event_detail_page.dart (Etkinlik detay)
└── widgets/
    ├── event_card.dart       (Etkinlik kartı widget)
    └── navbar.dart           (Navigasyon çubuğu)
```

State Management:

- Provider pattern kullanılmıştır
- AuthProvider: Kullanıcı authentication state
- EventProvider: Etkinlik listesi ve CRUD işlemleri
- ChangeNotifier ile reactive güncellemeler

Routing:

- Named routes kullanılmıştır
- Routes: `/`, `/login`, `/admin`, `/my-tickets`

API Integration:

- Base URL: `http://localhost:5151`
- HTTP headers ile JWT token gönderimi
- JSON serialization/deserialization
- Error handling ve loading states

7.3. Güvenlik Özellikleri

Backend Güvenlik:

- BCrypt ile şifre hashleme
- JWT token bazlı authentication
- HTTPS kullanımı (production)
- SQL Injection koruması (EF Core parametrized queries)
- CORS policy kontrolü

Frontend Güvenlik:

- Token güvenli şekilde saklanır (SharedPreferences)
- Hassas bilgiler plain text olarak saklanmaz
- Route guards (isLoggedIn, isAdmin kontrolü)

7.4. Performans Optimizasyonları

- Pagination ile sayfa başına 5 etkinlik
- Lazy loading
- Efficient state management (Provider)
- Image lazy loading ve error handling
- Database indexing (Username, Email unique)

7.5. Responsive Tasarım

- Grid column count ekran boyutuna göre: mobilde 1, büyük ekranda 3
- Flexible layouts (Expanded, Flexible widgets)
- Media queries
- Adaptive sizing

7.6. Veritabanı Bağlantısı

PostgreSQL bağlantı string'i appsettings.json dosyasında:

json

```
{
  "ConnectionStrings": {
```

"DefaultConnection":
"Host=localhost;Port=5432;Database=EventsDb;Username=postgres;Password=****"

}

}

Docker ile PostgreSQL:

```bash

docker run -d --name events-postgres -p 5432:5432 -e POSTGRES_PASSWORD=**** postgres

Sonuç

Bu projede modern web teknolojileri kullanılarak tam fonksiyonel bir etkinlik bilet satış sistemi geliştirilmiştir.

Başarılan Özellikler:

Kullanıcı kayıt ve giriş sistemi (JWT authentication)

Admin panel ile CRUD işlemleri

Etkinlik listeleme, arama ve filtreleme

Bilet satın alma ve iptal işlemleri

Responsive tasarım

Veritabanı ilişkileri ve cascade delete

State management (Provider pattern)

RESTful API tasarımı

Güvenlik önlemleri (BCrypt, JWT)

Sistem production ortamına alınmadan önce yapılması gerekenler:

- HTTPS zorunlu hale getirilmeli

- CORS policy production için kısıtlanmalı

- Rate limiting eklenmeli

- Logging ve monitoring implementasyonu

- Unit ve integration testler yazılmalı

- Error handling geliştirilmeli

- Payment gateway entegrasyonu (ödeme sistemi)

- Email verification sistemi
```