

Akİllİ Kampüs Saİlık ve Güvenlik Bildirim Uygulaması

Teknik Rapor

1. Proje Özeti

Bu proje, kampüs içinde ortaya çıkan saİlık, güvenlik, çevre, kayıp-buluntu ve teknik arıza gibi olayların kullanİcılar tarafından mobil uygulama üzerinden hızlı biçimde raporlanması, harita üzerinde görüntülenmesini, durumlarının takip edilebilmesini ve yöneticiler tarafından güncellenmesini saİlayan bir Flutter tabanlı mobil uygulamadır.

1.1. Amaç

Kampüs içinde güvenliİ ve iletiİmi artırmayı hedefleyen merkezi bir dijital bildirim platformu sunmak.

1.2. Hedef Kitle

- Kullanİcılar (User)**: Kampüs içindeki öğrenciler ve personel
- Yöneticiler (Admin)**: Bildirimleri yöneten ve acil durum uyarılarını yayınlayan yetkili kişiler

2. Kullanılan Teknolojiler

2.1. Frontend Framework

- Flutter SDK**: 3.0.0+ (Dart programlama dili)
- Material Design 3**: Modern ve kullanıcı dostu arayüz

2.2. Backend ve Veritabanı

- Firebase Authentication**: Kullanıcı kimlik doğrulama (email/İfre)
- Cloud Firestore**: NoSQL veritabanı (bildirimler, kullanıcılar)
- Firebase Storage**: Fotoğraf depolama
- Firebase Cloud Messaging (FCM)**: Push notification sistemi

2.3. Harita ve Konum Servisleri

- **Google Maps Flutter**: Harita görüntüleme ve pin yerleştirme
- **Geolocator**: Cihaz konumu alma ve izin yönetimi

2.4. State Management

- **Provider**: Uygulama durumu yönetimi (reactive programming)

2.5. Diğer Kütüphaneler

- **http**: REST API çağrılar (FCM için)
- **image_picker**: Fotoğraf seçme ve yükleme
- **shared_preferences**: Yerel veri saklama
- **intl**: Tarih/saat formatlama

3. Mimari Yapı

3.1. Proje Yapısı

```
lib/
├── main.dart # Uygulama giriş noktası
├── models/ # Veri modelleri
│   ├── user_model.dart # Kullanıcı modeli
│   ├── notification_model.dart # Bildirim modeli
│   └── services/ # Mantık servisleri
│       ├── auth_service.dart # Kimlik doğrulama servisi
│       ├── notification_service.dart # Bildirim CRUD işlemleri
│       └── fcm_service.dart # Push notification servisi
├── providers/ # State management
│   ├── auth_provider.dart # Kullanıcı durumu
│   └── notification_provider.dart # Bildirim durumu
├── screens/ # UI ekranlar
│   ├── auth/ # Giriş/kayıt ekranları
│   ├── home/ # Ana sayfa
│   ├── map/ # Harita ekranı
│   ├── notification/ # Bildirim ekranları
│   ├── admin/ # Admin paneli
│   └── profile/ # Profil ekranları
```

3.2. Tasarım Desenleri

- **Service Layer Pattern**: Mantık servislerde ayrılm
- **Provider Pattern**: State management için
- **Repository Pattern**: Firestore işlemleri için
- **Model-View-ViewModel (MVVM)**: UI ve mantık ayrım

4. Temel Özellikler

4.1. Kullanıcı Özellikleri

1. ****Giriş ve Kayıt****: Email/Şifre ile kimlik doğrulama
2. ****Bildirim Oluşturma****: Tür, başlık, açıklama, konum ve fotoğraf ile bildirim oluşturma
3. ****Bildirim Listeleme****: Filtreleme (tür, durum, takip edilenler) ve arama
4. ****Harita Görüntüleme****: "Açık" ve "İnceleniyor" durumundaki bildirimlerin harita üzerinde renkli pinlerle gösterilmesi
5. ****Bildirim Takibi****: Bildirimleri takip etme/takipten çıkarma
6. ****Push Notification****: Takip edilen bildirimlerin durum değişikliklerinde bildirim alma

4.2. Admin Özellikleri

1. ****Bildirim Yönetimi****: Tüm bildirimleri görüntüleme ve silme
2. ****Durum Güncelleme****: Bildirim durumlarının güncelleme (Açık → İnceleniyor → Çözüldü)
3. ****Acil Durum Bildirimi****: Tüm kullanıcılara anında acil durum uyarısı gönderme

5. Veritabanı Yapısı

5.1. Firestore Collections

users

- `id`: String (Firebase Auth UID)
- `name`: String
- `email`: String
- `department`: String
- `role`: String ("user" veya "admin")
- `createdAt`: Timestamp
- `fcmToken`: String (push notification için)

notifications

- `id`: String
- `type`: String (health, safety, environment, lostFound, technical)
- `title`: String
- `description`: String
- `createdAt`: String (ISO 8601)
- `status`: String (open, underReview, resolved)
- `userId`: String
- `userName`: String
- `latitude`: Number
- `longitude`: Number
- `photoUrls`: Array (opsiyonel)
- `followedBy`: Array (takip eden kullanıcı ID'leri)

config

- `fcm`: Document
- `serverKey`: String (FCM Server Key)

6. Push Notification Sistemi

6.1. FCM Entegrasyonu

- Her kullanıcının girişi yapıldığında FCM token'ı alınır ve Firestore'a kaydedilir
- Token'lar kullanıcının dokümanında `fcmToken` alanında saklanır

6.2. Bildirim Türleri

1. ****Durum Güncelleme Bildirimleri****: Takip edilen bildirimlerin durumu değiştiğinde, takip eden kullanıcılara gönderilir
2. ****Acil Durum Bildirimleri****: Admin tarafından yayınlanan acil durum uyarıları tüm kullanıcılara gönderilir

6.3. Bildirim Gönderme Mekanizması

- FCM REST API kullanılarak HTTP POST istekleri ile bildirimler gönderilir
- Server Key Firestore'dan (`config/fcm`) alınır
- Her bildirim için ayrı HTTP isteği gönderilir

7. Harita Entegrasyonu

7.1. Google Maps API

- Google Maps Flutter plugin kullanılarak harita görüntülenir
- API key `AndroidManifest.xml` dosyasında tanımlanmıştır

7.2. Pin Gösterimi

- Sadece "Açık" ve "İnceleniyor" durumundaki bildirimler haritada gösterilir
- Her bildirim türü için farklı renkli pinler kullanılır:
- ****Sağlık (Health)****: Kırmızı
- ****Güvenlik (Safety)****: Turuncu
- ****Çevre (Environment)****: Yeşil
- ****Kayıp-Buluntu (Lost & Found)****: Mavi
- ****Teknik (Technical)****: Mor

7.3. Konum İzini

- Uygulama bařlangıcında konum izinleri istenir
- İzin verilmezse varsayılan konum (Atatürk Üniversitesi) kullanılır

8. Güvenlik ve Yetkilendirme

8.1. Kimlik Doğrulama

- Firebase Authentication ile email/şifre doğrulaması
- JWT token tabanlı oturum yönetimi

8.2. Rol Tabanlı Erişim

- Kullanıcı rolleri Firestore'da saklanır
- Admin paneli sadece admin rolüne sahip kullanıcılar tarafından erişilebilir

8.3. Veri Güvenliği

- Firestore Security Rules ile veri erişim kontrolü (önerilir)
- Hassas bilgiler (FCM Server Key) Firestore'da saklanır

9. Test ve Hata Ayıklama

9.1. Geliştirme Ortamı

- Android Studio / VS Code
- Flutter SDK 3.0.0+
- Android Emulator (emulator-5554)

9.2. Çözülen Sorunlar

- NDK versiyon uyumsuzluğu (NDK 27.0.12077973 kullanıldı)
- Firebase initialization hataları
- Google Maps API key yapılandırması
- FCM token yönetimi ve push notification gönderimi

10. Sonuç ve Gelecek Geliştirmeler

10.1. Tamamlanan Özellikler

- Kullanıcı giriř/kayıt sistemi
- Bildirim oluřturma, listeleme, filtreleme
- Harita üzerinde bildirim görüntüleme
- Admin paneli ve bildirim yönetimi
- Push notification sistemi
- Bildirim takip sistemi

10.2. Önerilen iyileřtirmeler

- Firestore Security Rules eklenmesi
- Offline veri senkronizasyonu
- Bildirim istatistikleri ve raporlama
- Çoklu dil desteđi
- Bildirim fotoğrafları için görüntü optimizasyonu
- Kullanıcı profil fotoğrafları

Hazırlayan: [Öğrenci Adı]

Tarih: [Tarih]

Versiyon: 1.0.0