# IOS İLE UYGULAMA GELİŞTİRME

# YÜZÜK ÖLÇÜSÜ ALMA UYGULAMASI

Öğrenci: İlayda Nihal Kavanoz

No:22291015

Github: github.com/nia1da/ballade_art_ring_sizer

Video: https://drive.google.com/file/d/10sS9XGYDMoowJxkdeRHje8a28HVtlLrT/view?usp=sharing

Bu proje, kullanıcıların telefon ekranını fiziksel bir cetvel gibi kullanarak kendi yüzük ölçülerini milimetrik hassasiyetle bulmalarını, bu verileri buluta kaydetmelerini ve paylaşabilmelerini sağlar.

## 2. Kullanılan Teknolojiler ve Mimari

Projeyi geliştirirken **Backend-Agnostik** (Sunucudan bağımsız) bir yapı kurmaya özen gösterdim. Kullandığım teknolojiler şunlardır:

- **Flutter & Dart:** UI ve logic işlemleri için.
- **MethodChannel (Native Code):** Flutter'ın erişemediği donanım tabanlı fiziksel ekran yoğunluğunu (DPI) çekmek için Android (Java/Kotlin) katmanıyla iletişim kuruldu.
- **MockAPI:** Backend servisi ve NoSQL veritabanı simülasyonu için kullanıldı.
- **HTTP Paketi:** RESTful servislerle (GET/POST) haberleşmek için.

## 3. Veritabanı Yapısı (MockAPI)

Veri tabanı olarak bulut tabanlı MockAPI servisi kullanıldı. `logs` isminde bir kaynak (resource) oluşturuldu. JSON yapısı şu şekildedir:

- **id:** Otomatik üretilen benzersiz kimlik.
- **createdAt:** Kayıt zamanı.
- **size:** Yüzük numarası (Örn: 14.5).
- **diameter:** Yüzük çapı (Örn: 17.2 mm).

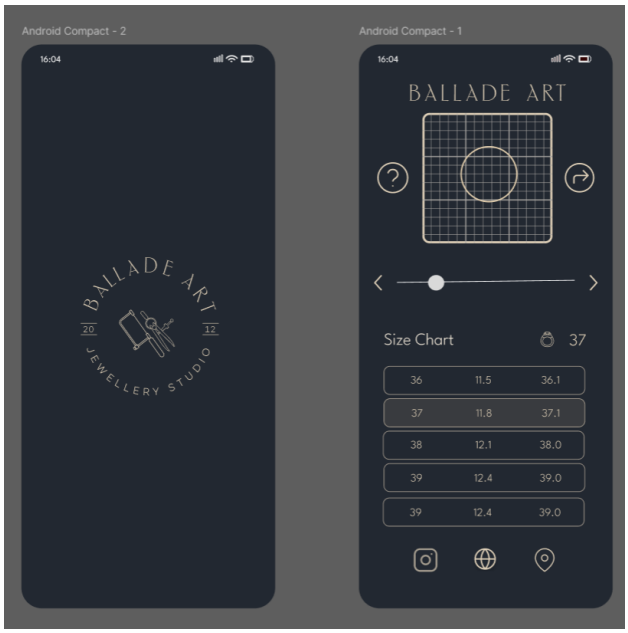
## 3. Uygulama Özellikleri

Projenin teknik altyapısını oluşturan temel fonksiyonlar şunlardır:

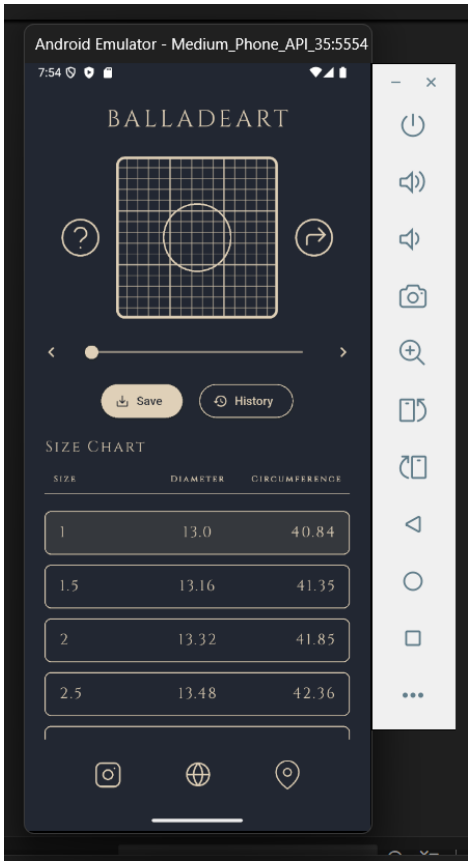
- **Fiziksel Ekran Kalibrasyonu (Native MethodChannel):** Standart piksel kullanımı yerine, `MethodChannel` ile Android işletim sisteminden cihazın gerçek **DPI (Dots Per Inch)** verisi çekilmiştir. Bu veri matematiksel formülle işlenerek, yüzük halkasının her cihazda milimetrik olarak doğru boyutta çizilmesi sağlanmıştır.
- **REST API Entegrasyonu (HTTP & Async Mimari):** Bulut tabanlı MockAPI servisi ile haberleşme sağlanmıştır. Veri gönderimi (**POST**) ve veri çekme (**GET**) işlemleri `http` paketi kullanılarak yapılmış; `async/await` yapısı ile arayüzün donması (UI blocking) engellenmiştir.
- **State Management ve Dinamik Çizim:** Kullanıcı slider'ı hareket ettirdiğinde `setState` mekanizması optimize bir şekilde kullanılarak, ekranın tamamı yerine

sadece değişen halka grafiğinin ve milimetre değerinin anlık olarak yeniden çizilmesi sağlanmıştır.

- **Algoritmik Liste Senkronizasyonu (Auto-Scroll):** Görsel halka boyutu değiştikçe, alttaki ölçü tablosunda en yakın değer matematiksel olarak hesaplanır. `ScrollController` kullanılarak liste otomatik olarak ilgili satıra kaydırılır ve kullanıcıya doğru numara işaretlenir.
- **Yerel Dil Algılama (Localization):** Uygulama açılışta cihazın `locale` bilgisini okur. Eğer cihaz dili Türkçe ise arayüz otomatik olarak Türkçe, değilse İngilizce olarak yapılandırılır.




Figma tasarımı




Emulatör ekran görüntüsü

## 4. Çalışma Mantığı Kod Yapısı

### *A. Native İletişim (DPI Helper)*

Uygulamanın doğru çalışması için ekranın piksel yoğunluğunu bilmemiz gerekiyor. Flutter standart olarak "mantıksal piksel" kullandığı için, gerçek fiziksel boyutu hesaplamak adına **MethodChannel** yapısını kurdum.

```dart
class DpiHelper {
  static const MethodChannel _channel = MethodChannel("dpi_channel");

  /// Native taraftan cihazın xdpi / ydpi değerlerini alır
  static Future<Map<String, double>> getDpi() async {
    final result = await _channel.invokeMethod<Map>("getDpi");
    return {
      "xdpi": (result?["xdpi"] as double?) ?? 0.0,
      "ydpi": (result?["ydpi"] as double?) ?? 0.0,
    };
  }

  /// 1 mm'nin kaç fiziksel px olduğunu hesaplar
  static double pixelsPerMm(double dpi) {
    return dpi / 25.4; // 1 inch = 25.4 mm
  }

  /// mm → fiziksel px dönüşümü
  static double mmToPx(double mm, double dpi) {
    return mm * pixelsPerMm(dpi);
```

Burda telefonun donanım tarafıyla iletişime geçiyoruz ve cihazın ekranının bir inçte kaç pixel olduğu değerini çekiyoruz.

B. API Entegrasyonu (Veri Kaydetme)

Verileri buluta gönderirken **HTTP POST** metodunu kullandım. Uygulamanın donmaması için işlemi `async/await` ile asenkron hale getirdim.

```dart
Future<void> saveToCloud(dynamic size, double diameter) async {
  try {
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(
        content: Text(isTurkish ? "Buluta bağlanıyor..." : "Connecting..."),
        duration: const Duration(milliseconds: 500),
      ), // SnackBar
    );

    // 2. DOĞRU VERİ TİPİ:
    // MockAPI "Number" bekliyor. .toString() ASLA kullanma.
    // size değişkeni zaten sayı (int veya double) geliyor, olduğu gibi yolla.
    final Map<String, dynamic> dataToSend = {
      "size": size,         // Örn: 14.5 (Sayı)
      "diameter": diameter, // Örn: 17.2 (Sayı)
    };

    final response = await http.post(
      Uri.parse(apiUrl),
      headers: {
        "Content-Type": "application/json",
        "Accept": "application/json",
      },
      body: jsonEncode(dataToSend),
    );
```

### *C. Veri Listeleme (FutureBuilder)*

Geçmiş ölçümleri listelerken `setState` karmaşasına girmemek için **FutureBuilder** widget'ını tercih ettim.

```dart
  @override
  void initState() {
    super.initState();
    final langCode = WidgetsBinding.instance.platformDispatcher.locale.languageCode;
    isTurkish = langCode == 'tr';

    _loadDpi();
  }
```

Burda cihazın sistem dili eğer türkçeyse langCode değişkenini tr yapıyoruz diğer durumlarda ise yazıları ingilizce göstereceğiz.

### *API Entegrasyonu (Veri Kaydetme)*

Verileri buluta gönderirken **HTTP POST** metodunu kullandım. Uygulamanın donmaması için işlemi `async/await` ile asenkron hale getirdim.

```dart
final String apiUrl = "https://69601edbe7aa517cb79560bc.mockapi.io/logs";

Future<void> saveToCloud(dynamic size, double diameter) async {
  // 1. Veriyi JSON formatına çevirme (Serialization)
  final Map<String, dynamic> dataToSend = {
    "size": size,
    "diameter": diameter,
  };

  // 2. İsteği Gönderme
  final response = await http.post(
    Uri.parse(apiUrl),
    headers: {"Content-Type": "application/json"},
    body: jsonEncode(dataToSend),
  );

  // 3. Durum Kontrolü
  if (response.statusCode >= 200 && response.statusCode < 300) {
    // Başarılı (Yeşil Tik Göster)
  }
}
```

Mock api ile çalışmaönizlemesi

Logs:

dit/replace data for **logs** resource. Data must be an array and a valid JSON.


```
[
  {
    "createdAt": "2026-01-08T15:02:08.095Z",
    "size": 6,
    "diameter": 14.61,
    "id": "1"
  },
  {
    "createdAt": "2026-01-08T00:24:55.456Z",
    "size": 18,
    "diameter": 18.49,
    "id": "2"
  },
  {
    "createdAt": "2026-01-08T08:51:30.428Z",
    "size": 12,
    "diameter": 16.55,
    "id": "3"
  },
  {
    "createdAt": "2026-01-08T18:37:51.938Z",
    "size": 23.5,
    "diameter": 20.26,
```