FLUTTER FAVORİLER UYGULAMASI

Erciyes Üniversitesi

Mühendislik Fakültesi

Bilgisayar Mühendisliği Bölümü

Ders: Mobil App.Development

Öğretim Üyesi Dr. Öğr. Üyesi FEHİM KÖYLÜ

Proje Ödevi: Vize Projesi

Öğrenci No: 1030510302 -ALİ ENES ERSÖZLÜ

İÇİNDEKİLER

Giriş

Bu rapor, Flutter kullanarak geliştirilen "Flutter Favoriler" adlı mobil uygulama projesini detaylıca incelemektedir. Uygulama, kullanıcıların çeşitli öğeleri listeleyebilecekleri ve favorilere ekleyebilecekleri basit bir arayüz sunmaktadır. Flutter'ın temel bileşenlerini, state yönetimini ve arayüz tasarımını göstermek amacıyla geliştirilmiştir.

Proje, Flutter ve Dart programlama dilinin temel özelliklerini kullanarak, modern bir mobil uygulama mimarisi ve kullanıcı arayüzü tasarımını uygulamaktadır. Uygulamada TabBar, ListView, Card gibi Flutter'ın yaygın kullanılan widget'ları ve StatefulWidget sınıfı aracılığıyla state yönetimi gösterilmiştir.

Uygulama Mimarisi

Uygulama aşağıdaki ana bileşenlerden oluşmaktadır:

1. **MyApp Sınıfı**: Uygulamanın temel yapısını oluşturan ve MaterialApp widget'ını döndüren bir StatelessWidget.
2. **HomePage Sınıfı**: Ana ekranı oluşturan ve iki sekme içeren bir StatefulWidget.
3. **_HomePageState Sınıfı**: State yönetimini sağlayan ve UI'ı oluşturan sınıf.
4. **Item Sınıfı**: Veri modelini temsil eden basit bir sınıf.

Arayüz Tasarımı

Uygulama, iki sekmeye ayrılmış bir arayüze sahiptir:

1. **Tüm Öğeler Sekmesi**: Sistemdeki tüm öğelerin listelendiği ve kullanıcının favorilere ekleyebildiği/çıkarabildiği sekme.
2. **Favoriler Sekmesi**: Kullanıcının favori olarak işaretlediği öğelerin listelendiği ve silinebildiği sekme.

Her iki sekmede de öğeler, Card widget'ları içinde ListTile'lar kullanılarak görüntülenmektedir. Bu, modern ve temiz bir kullanıcı arayüzü sağlamaktadır.

Kod Açıklamaları

main.dart

```dart
void main() {
  runApp(const MyApp());
}
```

Bu fonksiyon, uygulamanın giriş noktasıdır ve MyApp widget'ını Flutter engine'a bağlar.

MyApp Sınıfı

```dart
class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Favoriler -Erciyes University Mobile Development Vize Ödevi 1030510302',
      theme: ThemeData(
        primarySwatch: Colors.blue,
        useMaterial3: true,
      ),
      home: const HomePage(),
    );
  }
}
```

MyApp sınıfı, uygulamanın başlığını, temasını ve ana sayfasını tanımlar. Material Design 3 kullanılarak modern bir görünüm sağlanmıştır.

HomePage ve State Yönetimi

```dart
class HomePage extends StatefulWidget {
  const HomePage({super.key});

  @override
  State<HomePage> createState() => _HomePageState();
}
```

HomePage, uygulamanın durum değişikliklerini yönetebilmesi için bir StatefulWidget olarak tanımlanmıştır.

_HomePageState Sınıfı

Bu sınıf, uygulamanın durumunu (state) yönetir ve kullanıcı arayüzünü oluşturur:

dart

```dart
class _HomePageState extends State<HomePage> {
  // Örnek veri listesi
  final List<Item> _items = [
    // ... öğeler ...
  ];

  // Favorileri tutacak liste
  final List<Item> _favorites = [];

  // ... build metodu ve diğer metotlar ...
}
```

Sekme Yapısı

dart

```dart
DefaultTabController(
  length: 2,
  child: Scaffold(
    appBar: AppBar(
      title: const Text('Flutter Favoriler -Erciyes University Mobile Development Vize Ödevi 10
      bottom: const TabBar(
        tabs: [
          Tab(icon: Icon(Icons.list), text: 'Tüm Öğeler'),
          Tab(icon: Icon(Icons.favorite), text: 'Favoriler'),
        ],
      ),
    ),
    body: TabBarView(
      // ... içerik ...
    ),
  ),
)
```

DefaultTabController ve TabBar, uygulamanın sekme yapısını oluşturmaktadır. Sekmelerde simgeler ve metin birlikte kullanılarak kullanıcı deneyimi iyileştirilmiştir.

Liste Görünümü

```dart
Widget _buildItemsList(List<Item> items, bool showAddButton) {
  return ListView.builder(
    padding: const EdgeInsets.all(8),
    itemCount: items.length,
    itemBuilder: (context, index) {
      // ... öğe görünümü ...
    },
  );
}
```

ListView.builder, öğeleri verimli bir şekilde listelemek için kullanılmıştır. Bu yapı, büyük listelerde bile performans kaybı yaşanmadan çalışabilir.

Veri Modeli

```dart
class Item {
  final int id;
  final String name;
  final String description;

  Item({required this.id, required this.name, required this.description});
}
```

Item sınıfı, uygulamanın veri modelini temsil eder ve her bir öğe için id, name ve description alanlarını içerir.

Temel Flutter Bileşenleri

Uygulamada kullanılan önemli Flutter bileşenleri şunlardır:

1. **MaterialApp**: Temel Material Design uygulaması oluşturur.
2. **Scaffold**: Temel Material Design görsel yapısını sağlar (AppBar, TabBar, vb.).
3. **DefaultTabController**: Sekme kontrolünü yönetir.
4. **TabBar ve TabBarView**: Sekmeli arayüz oluşturur.
5. **ListView.builder**: Verimli liste görünümü sağlar.
6. **Card ve ListTile**: Liste öğeleri için modern, kart tabanlı tasarım elemanları.
7. **IconButton**: Kullanıcı etkileşimi için simgeli butonlar sağlar.

State Yönetimi

Uygulama, Flutter'ın temel state yönetim mekanizmasını kullanmaktadır. `setState()` methodu, kullanıcı favorilere bir öğe eklediğinde veya çıkardığında arayüzün yeniden oluşturulmasını sağlar:

```dart
setState(() {
  if (_favorites.contains(item)) {
    _favorites.remove(item);
  } else {
    _favorites.add(item);
  }
});
```

Bu basit yaklaşım, küçük ve orta ölçekli uygulamalar için yeterli olmakla birlikte, daha karmaşık uygulamalarda Provider, Bloc veya Riverpod gibi state yönetim çözümleri tercih edilebilir.

Sonuç

"Flutter Favoriler" uygulaması, Flutter'ın temel özellikleri ve widget'larını kullanarak basit ancak işlevsel bir mobil uygulama oluşturmanın mümkün olduğunu göstermektedir. Uygulama, aşağıdaki Flutter kavramlarını başarıyla uygulamaktadır:

- StatelessWidget ve StatefulWidget kullanımı
- TabBar ve sekmeli navigasyon
- ListView ve Card yapıları ile liste görünümleri
- Temel state yönetimi
- Icon kullanımı ve kullanıcı etkileşimleri

Gelecekteki geliştirmeler arasında, verilerin yerel depolama veya bulut veritabanında saklanması, daha karmaşık filtreleme seçenekleri ve gelişmiş kullanıcı arayüzü animasyonları eklenebilir.

Bu proje, Flutter'ın mobil uygulama geliştirme için ne kadar güçlü ve esnek bir framework olduğunu göstermektedir ve daha karmaşık uygulamalar geliştirmek için sağlam bir temel oluşturmaktadır.