Erciyes Üniversitesi

Bilgisayar Mühendisliği Bölümü

Mobil Application Projesi

1030510314

Sinem Kadakal

Flutter'da Layout ve Tasarım

1. Giriş

Flutter, Google tarafından geliştirilen ve tek bir kod tabanından iOS, Android ve diğer platformlar için uygulama geliştirmeye olanak sağlayan açık kaynaklı bir UI araç takımıdır. Flutter'ın en güçlü yanlarından biri zengin ve özelleştirilebilir kullanıcı arayüzleri oluşturabilme yeteneğidir.

2. Flutter'da Widget Kavramı

Flutter'da her şey bir widget'tır. Widget'lar, kullanıcı arayüzünün yapı taşlarıdır ve uygulamanın görünümünü ve davranışını tanımlarlar.

Widget Tipleri:

- Stateless Widget: Durumu değişmeyen, sabit widget'lar
- Stateful Widget: Kullanıcı etkileşimiyle veya zamanla durumu değişebilen widget'lar
- Inherited Widget: Veri paylaşımını widget ağacı boyunca sağlayan widget'lar

Widget Ağacı:

Flutter, kullanıcı arayüzlerini bir widget ağacı olarak yapılandırır. Her widget, başka widget'ları içerebilir ve böylece karmaşık arayüzler oluşturulabilir.

dart

```
MaterialApp(
 home: Scaffold(
  appBar: AppBar(title: Text('Flutter Layout Örneği')),
  body: Center(
   child: Column(
    children: [
     Text('Merhaba Flutter'),
     ElevatedButton(onPressed: () {}, child: Text('Tıkla'))
    ],
   ),
  ),
 ),
)
```

3. Layout Widget'ları ve Temel Tasarım İlkeleri

Flutter'da layout oluşturmak için kullanılan temel widget'lar:

Tek Çocuklu Layout Widget'ları:

- Container: Padding, margin, sınırlar ve boyut ekleyebilen temel layout widget'ı
- Center: İçeriğini ortalar
- Padding: İçeriğine padding ekler
- Align: İçeriğini belirtilen konuma hizalar
- SizedBox: Belirli genişlik ve yükseklikte alan tanımlar

Çok Çocuklu Layout Widget'ları:

- Row: Çocuklarını yatay düzende yerleştirir
- Column: Çocuklarını dikey düzende yerleştirir
- Stack: Çocuklarını üst üste yerleştirir
- Wrap: Çocuklarını satır ve sütunlarda sarar
- ListView: Kaydırılabilir bir liste oluşturur
- GridView: Kaydırılabilir bir ızgara oluşturur

Tasarım İlkeleri:

- Constraints (Kısıtlamalar): Flutter'da widget'lar, ebeveynlerinden boyut kısıtlamaları alır ve bu kısıtlamalara uymak zorundadır.
- Flex Layout: Ekran boyutuna göre esneyebilen esnek layout'lar oluşturmak için Expanded ve Flexible widget'ları kullanılır.
- Aspect Ratio: AspectRatio widget'ı ile belirli en/boy oranında alanlar oluşturulabilir.

dart

```
Column(
 children: [
  Container(color: Colors.red, height: 100),
  Expanded(
   flex: 2,
   child: Container(color: Colors.blue),
  ),
  Expanded(
   flex: 1,
   child: Container(color: Colors.green),
  ),
 ],
)
```

4. Responsif Tasarım

Flutter uygulamalarının farklı ekran boyutlarına ve yönelimlerine uyum sağlaması önemlidir.

Responsif Tasarım Teknikleri:

- MediaQuery: Ekran boyutu, yönelimi ve piksel yoğunluğu gibi bilgilere erişim sağlar
- LayoutBuilder: Üst widget'ın kısıtlamalarına göre dinamik layout oluşturmayı sağlar
- OrientationBuilder: Ekran yönelimine göre farklı widget'lar döndürür
- FractionallySizedBox: Ekran boyutunun belirli bir yüzdesini kaplayan widget'lar oluşturur

dart

```
LayoutBuilder(
 builder: (context, constraints) {
  if (constraints.maxWidth > 600) {
   // Tablet veya geniş ekran için görünüm
   return WideLayout();
  } else {
   // Telefon veya dar ekran için görünüm
   return NarrowLayout();
  }
 },
)
```

5. Temalar ve Stil Uygulamaları

Flutter'da tutarlı bir kullanıcı arayüzü için temalar ve stiller kullanılır.

ThemeData:

MaterialApp widget'ı içinde tanımlanan ThemeData, uygulamanın renk şemasını, yazı tiplerini ve diğer görsel öğelerini belirler.

dart

```
MaterialApp(
 theme: ThemeData(
  primarySwatch: Colors.blue,
  fontFamily: 'Roboto',
  textTheme: TextTheme(
   headline1: TextStyle(fontSize: 24.0, fontWeight: FontWeight.bold),
```

```
    bodyText1: TextStyle(fontSize: 16.0),
   ),
   elevatedButtonTheme: ElevatedButtonThemeData(
    style: ElevatedButton.styleFrom(
     primary: Colors.blue,
     onPrimary: Colors.white,
    ),
   ),
  ),
  home: MyHomePage(),
 )
```

Özel Stiller:

- BoxDecoration: Container widget'ının dekorasyonunu tanımlamak için kullanılır.
- TextStyle: Metin stilini tanımlamak için kullanılır.
- ButtonStyle: Düğme stilini tanımlamak için kullanılır.

dart

```
 Container(
  decoration: BoxDecoration(
   color: Colors.white,
   borderRadius: BorderRadius.circular(10),
   boxShadow: [
    BoxShadow(
     color: Colors.grey.withOpacity(0.5),
     spreadRadius: 2,
     blurRadius: 5,
     offset: Offset(0, 3),
    ),
   ],
  ),
  child: Text(
```

```
  'Özel Stil',
  style: TextStyle(
   fontSize: 18,
   fontWeight: FontWeight.bold,
   color: Colors.blue,
  ),
 ),
)
```

6. İleri Seviye Layout Teknikleri

Karmaşık arayüzler oluşturmak için kullanılan ileri seviye teknikler:

Custom Clipper:

Özel şekillerde widget'lar oluşturmak için ClipPath ve custom Clipper sınıfları kullanılır.

dart

```
ClipPath(
 clipper: MyCustomClipper(),
 child: Container(
  color: Colors.blue,
  height: 200,
 ),
)
class MyCustomClipper extends CustomClipper<Path> {
 @override
 Path getClip(Size size) {
  final path = Path();
  path.lineTo(0, size.height - 50);
  path.quadraticBezierTo(
   size.width / 2, size.height,
   size.width, size.height - 50
  );
  path.lineTo(size.width, 0);
```

```
  path.close();
  return path;
 }
 @override
 bool shouldReclip(CustomClipper<Path> oldClipper) => false;
}
```

CustomPaint: Canvas API kullanarak özel çizimler yapmak için kullanılır.

dart

```
CustomPaint(
 painter: MyPainter(),
 size: Size(200, 200),
)
class MyPainter extends CustomPainter {
 @override
 void paint(Canvas canvas, Size size) {
  final paint = Paint()
   ..color = Colors.blue
   ..style = PaintingStyle.fill;

  canvas.drawCircle(
   Offset(size.width / 2, size.height / 2),
   size.width / 3,
   paint,
  );
 }
 @override
 bool shouldRepaint(CustomPainter oldDelegate) => false;
}
```

Slivers:

Kaydırılabilir arayüzler için yüksek performanslı öğeler:

- SliverAppBar: Kaydırma sırasında davranışı değişen app bar
- SliverList: Kaydırılabilir liste
- SliverGrid: Kaydırılabilir ızgara
- CustomScrollView: Sliver widget'larını bir araya getiren scrollable container

dart

```
CustomScrollView(
 slivers: [
  SliverAppBar(
   floating: true,
   expandedHeight: 200,
   flexibleSpace: FlexibleSpaceBar(
    title: Text('Sliver Örneği'),
    background: Image.network('url_to_image', fit: BoxFit.cover),
   ),
  ),
  SliverList(
   delegate: SliverChildBuilderDelegate(
    (context, index) => ListTile(title: Text('Öğe $index')),
    childCount: 20,
   ),
  ),
 ],
)
```

Sonuç olarak, flutter güçlü ve esnek bir layout sistemi sunarak geliştiricilerin karmaşık ve güzel kullanıcı arayüzleri oluşturmasına olanak tanır.