

Flutter Görsel Temelleri: Butonlar ve Container Dünyasına Giriş

Flutter ekosisteminde bir müfredat mimarı olarak bilmeniz gereken ilk kural şudur: **"Her şey bir widget'tır."** Ancak bu ifadeyi daha teknik bir derinlikle ele alırsak; Flutter arayüzü, widget'ların bir "ağaç hiyerarşisi" (Widget Tree) içinde iç içe geçmesiyle oluşur. Bu hiyerarşinin en kritik yapı taşları, kullanıcıyla iletişimi kuran **Butonlar** ve bu butonları sarmalayıp onlara kimlik kazandıran **Container** bileşenleridir. Bu iki yapı, hem işlevsel etkileşimi hem de görsel disiplini temsil eder.

Kullanıcıyla ilk teması kurduğumuz etkileşim araçlarıyla, yani butonlarla keşfimize başlayalım.

1. Etkileşimin Anahtarı: Buton Türleri

Flutter'da bir buton, mantıksal bir fonksiyon (onPressed) ve görsel bir taşıyıcının (child) birleşimidir. child parametresi genellikle bir Text widget'ı olsa da, tasarım ihtiyacına göre bir Icon da barındırabilir. Kaynak verilerimizde yer alan iki temel butonu mimari açıdan inceleyelim:

Buton Karşılaştırma Tablosu

Buton Tipi	Görsel Karakteristik	Kullanım Amacı
ElevatedButton	Belirgin bir arka planı ve derinlik katan gölgesi vardır.	Sayfadaki ana eylemi (örneğin "Kaydet") vurgulamak için tercih edilir.
TextButton	Arka planı ve gölgesi yoktur; sadece metin veya ikondan oluşur.	İkincil eylemler veya form iptalleri gibi daha az vurgu gerektiren alanlar için uygundur.

Önemli Not: Geliştirme sürecinde onPressed bloğu içine eklenen print("butona tıklandı") ifadesi, uygulamanın mantıksal akışını takip etmek için kullanılan bir **"konsol geri bildirimi"** yöntemidir. Bu, UI ile kod arasındaki bağlantının koptuğu noktaları tespit etmenizi sağlar.

Butonlarla etkileşimi sağladık, şimdi bu öğeleri içine koyup şekillendirebileceğimiz en esnek kutumuzu, yani Container'ı tanıyalım.

2. Container: Arayüzün İsviçre Çakısı

Container, Flutter'da sadece bir kutu değil, düzenleme ve dekorasyonun merkez üssüdür. Temel olarak width (genişlik) ve height (yükseklik) özellikleriyle sınırlandırılır.

- **Hizalama Esnekliği:** Bir Container içindeki içeriği ortalamak için alignment: Alignment.center parametresi kullanılır.
- **Pro İpucu (Mimari Alternatif):** Eğer sadece içeriği ortalamak istiyorsanız, Container'ın içindeki child parametresini doğrudan bir Center widget'ı ile

sarmalamak da (örneğin `child: Center(child: Text("hikayemiz"))`) aynı sonucu verir. Bu, Flutter'da bir hedefe giden birden fazla mimari yol olduğunun kanıtıdır.

Sadece bir kutu oluşturmak yetmez; şimdi bu kutuyu sanatsal bir dokunuşla nasıl süsleyeceğimizi öğrenelim.

3. BoxDecoration ile Stil Verme Sanatı

Container bileşenine görsel derinlik ve karakter katan yapı `decoration`: `BoxDecoration` sınıfıdır. Bu sınıf, bir kutuyu standart formundan çıkarıp modern bir tasarım öğesine dönüştürür.

Kenarlık ve Köşeler

- **BorderRadius:** `BorderRadius.circular(20)` tüm köşeleri yumuşatırken; `BorderRadius.horizontal(left: Radius.circular(20))` kullanımı sadece sol tarafı yuvarlayarak bir "sekme" (tab) veya özel bir kart efekti oluşturmanıza olanak tanır.
- **Border:** `Border.all` ile kutuya belirgin bir çerçeve ekleyebilirsiniz. Örneğin, `color: Colors.white` ve `width: 5` değerleriyle kutunuzu bir çerçeve içine alabilirsiniz.

Renk Geçişleri ve Gölgeler

- **LinearGradient:** Statik renkler yerine, `begin: Alignment.topLeft` ve `end: Alignment.bottomRight` noktalarıyla yönlendirilen renk geçişleri tasarıma dinamizm katar.
- **BoxShadow:** Derinlik algısı için şu spesifik değerler kullanılır:
 - `color: Colors.red.shade500`
 - `blurRadius: 10` (Bulanıklık miktarı)
 - `spreadRadius: 5` (Gölgenin yayılma alanı)
 - `offset: Offset(0, 10)` (Gölgenin X ve Y eksenindeki konumu).

Hayati İpucu: Eğer bir Container'da `decoration` tanımlanmışsa, `color` parametresi mutlaka bu `BoxDecoration` nesnesinin içinde olmalıdır. Eğer renk dışarıda bırakılırsa, Flutter bir çalışma zamanı hatası (Exception) fırlatır ve uygulama çöker.

Görsel stilimiz tamam, ancak öğelerin birbirine olan mesafesi de en az stil kadar önemlidir.

4. Yerleşim Disiplini: Padding ve Margin Farkı

Tasarımda "nefes alma alanı" yaratmak, kullanıcı deneyimi (UX) için kritiktir:

- **Padding (İç Boşluk):** Kutunun kenarı ile içindeki child (içerik) arasındaki mesafedir. Örneğin `EdgeInsets.all(20)` içeriği her taraftan merkeze doğru iter.
- **Margin (Dış Boşluk):** Kutunun kendisi ile dışarıdaki diğer bileşenler arasındaki mesafedir. `EdgeInsets.all(50)` gibi bir değer, kutuyu komşu öğelerden belirgin şekilde uzaklaştırır.

Özel Durum (Görüntüler): Image gibi bazı widget'ların kendi padding özelliği yoktur. Bir görsele boşluk vermek için onu bir Padding widget'ı ile sarmalamanız gerekir:

Padding(

padding: const EdgeInsets.only(top: 10),

child: Image.asset("assets/image/foto.png", width: 50),

)

Tüm bu teknik detayları bir araya getirerek büyük resmi görelim.

5. Özet ve Pratik Uygulama Önerisi

Öğrenilen tüm bileşenler birleştiğinde, profesyonel bir "Hazır Card Yapısı" ortaya çıkar. Bir arayüz mimarı olarak kendi kart tasarımınızı şu hiyerarşiyle oluşturabilirsiniz:

1. **Temel İskelet:** Bir Container oluşturun, boyutlarını (örneğin 100x100) belirleyin ve margin: `EdgeInsets.all(50)` ile dış dünyadan ayırın.
2. **Stil ve Kimlik:** `BoxDecoration` içine girin; `LinearGradient` ile renk geçişi verin ve `BoxShadow (Offset(0, 10))` ile zeminden yükseltin.
3. **Köşe Yumuşatma:** `BorderRadius.circular(20)` ile modern bir görünüm sağlayın.
4. **İçerik Yerleşimi:** child olarak bir Center veya Padding widget'ı kullanarak içine `ElevatedButton` veya `Text` ekleyerek kartınızı etkileşime açın.

Flutter dünyası, bu temel taşları yaratıcı şekillerde üst üste dizmenizle şekillenir. Şimdi kendi özgün kart tasarımlarınızı oluşturmak için kodun başına geçme vakti!