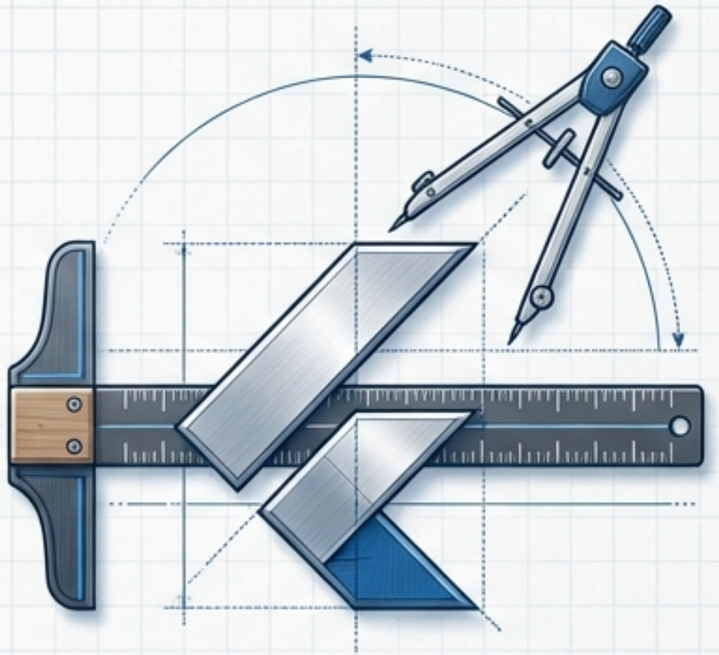


Flutter ile Materyal Arayüzler İnşa Etmek: Bir Mimarın Rehberi

Flutter'ın `material.dart` kütüphanesini kullanarak sağlam, estetik ve tutarlı uygulamalar yaratmanın temel prensipleri.

Projenin Temelini Atmak: İki Yaklaşım

Sıfırdan İnşa Etmek

Tüm düzeni, stateless ve stateful widget'ları kullanarak sıfırdan oluşturmak. Ekran boyutları, konumlandırma ve butonlarla uğraşmak ciddi bir iş yükü anlamına gelir.

Hazır Bir Plan Kullanmak

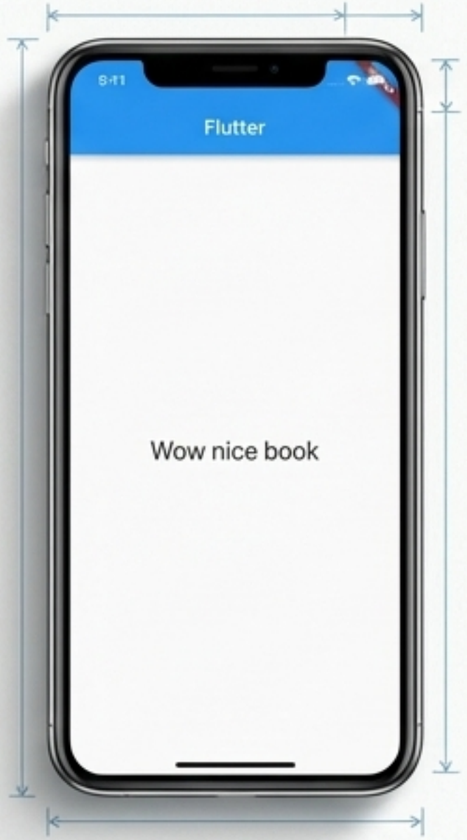
`material.dart` paketini içe aktarmak ve Flutter'ın sunduğu `MaterialApp` widget'ını Bu yaklaşım, Material Design yönergelerini izleyen bir kullanıcı arayüzünün "iskeletini" oluşturur.

Flutter size ihtiyacınız olan her şeyi zaten sunuyor. `MaterialApp` ile güzel bir materyal uygulama yaratacağınız garanti.

Projenin Ana Planı: `MaterialApp`

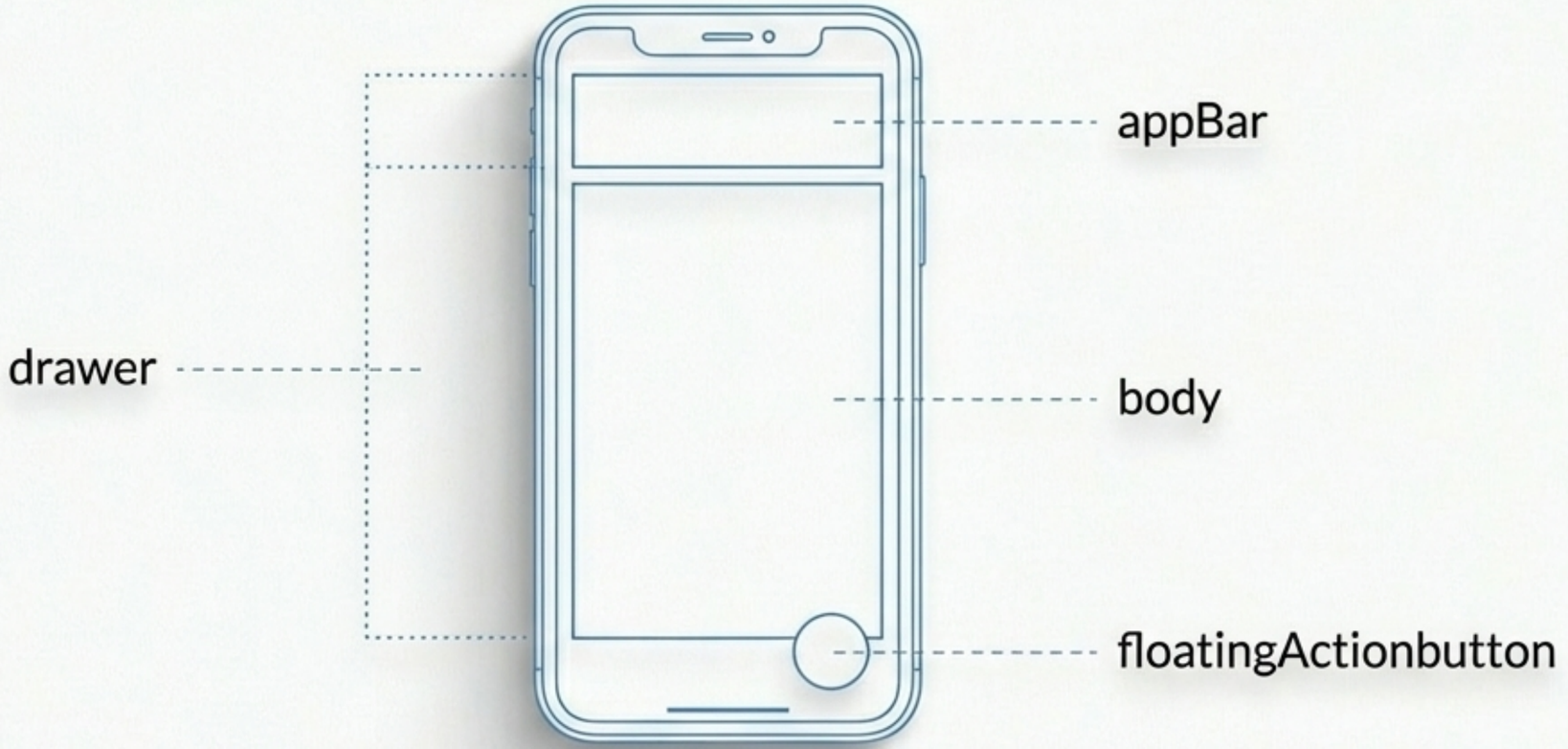
`MaterialApp`, uygulamanızın temelini oluşturan ve tüm materyal tasarım bileşenlerini bir araya getiren ana widget'tır. Sayfa navigasyonu ve yerelleştirme gibi birçok kritik özelliği yönetir.

```
// MaterialApp bir materyal tasarım uygulamasının
// "iskeletini" temsil eder.
Widget build(BuildContext context) {
  return MaterialApp(
    home: Scaffold(
      appBar: AppBar(
        title: const Text("Flutter"),
      ),
      body: const Center(
        child: Text("Wow nice book"),
      ),
    ),
  );
}
```



Her Ekranın Temeli: `Scaffold`

`Scaffold` sınıfı, temel materyal tasarım düzen yapısını uygular. Sadece tipik Android 'görünüm ve hissini' sağlamakla kalmaz, aynı zamanda `AppBar`, `Drawer` ve `FloatingActionButton` gibi birçok önemli widget'ı yönetme olanağı tanır.



Uygulamanın Vitrini: `AppBar`

`AppBar`, genellikle ekranın en üstüne yerleştirilir. Uygulamanızın sayfaları (Flutter dünyasında 'route'lar) arasında gezinirken, `AppBar` otomatik olarak tipik bir sol ok olan 'geri' düğmesini ekler.

```
// Sayfalar arasında gezinirken geri düğmesinin
// otomatik olarak görünmesini istemiyorsanız
// bu özelliği 'false' yapın.
Scaffold(
  appBar: AppBar(
    automaticallyImplyLeading: false,
    title: const Text("Geri Düğmesi Olmayan AppBar"),
  )
```

`AppBar`'a İşlevsellik Eklemek: Eylem Butonları

Başlığın sağında yer alan butonlara eylem butonları denir. Genellikle, o butonun amacını görsel olarak temsil eden tıklanabilir ikonlardır ve `actions` parametresine bir widget listesi geçirilerek ayarlanırlar.

```
Scaffold(
  appBar: AppBar(
    title: const Text("Flutter"),
    actions: [
      IconButton(
        icon: const Icon(Icons.info),
        onPressed: () { /* ... */ }
      ),
      Padding(
        padding: EdgeInsets.only(right: 20),
        child: Icon(Icons.search),
    ] )
  )
```

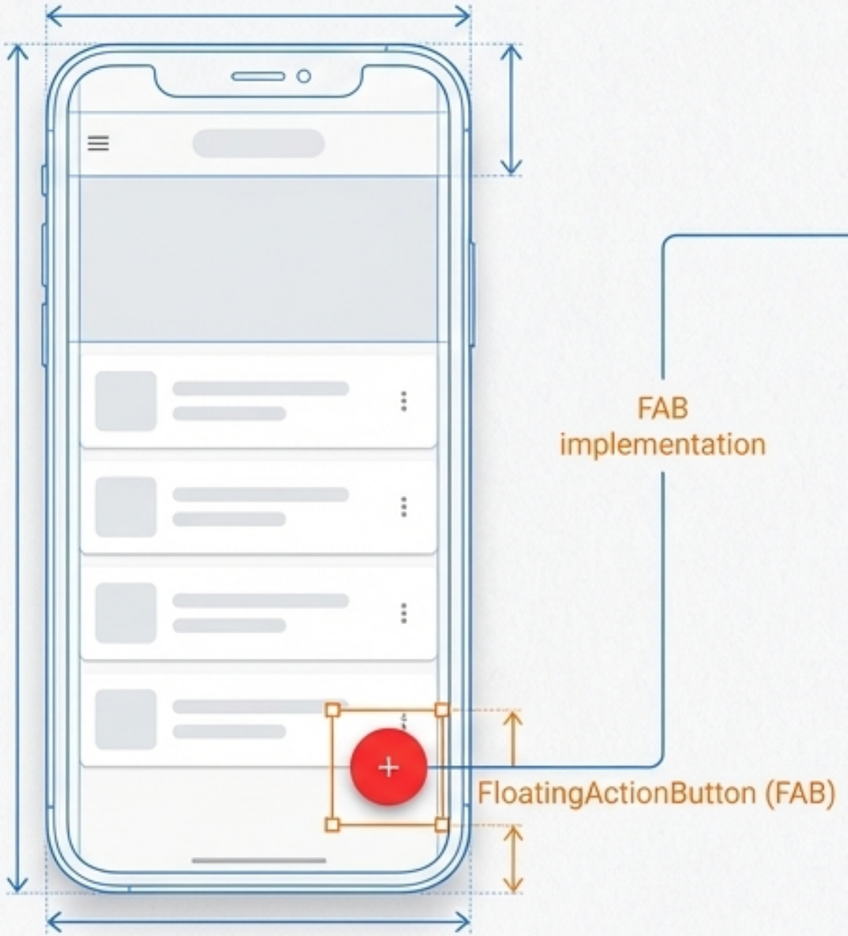
Gizli Alanlar: `Drawer` ve `endDrawer`

`Drawer`, genellikle kullanıcıyı uygulamanın belirli sayfalarına yönlendiren ikon ve metin kombinasyonlarını görüntülemek için kullanılan, ekranın bir tarafından yatay olarak kayan bir paneldir.

```dart
Scaffold(
  // Klasik soldan sağa açılan drawer
  drawer: Drawer(
    child: ListView(children: [
      ListTile(
        leading: const Icon(Icons.people),
        title: const Text("Item 1"),
        onTap: () {},
      ),
      ListTile(
        leading: const Icon(Icons.train),
        title: const Text("Item 2"),
        onTap: () {},
      )
    ]),
  ),
  // Sağdan sola açılan drawer
  endDrawer: Drawer(),
)
```

İmza Dokunuşu: `FloatingActionButton` (FAB)

FAB olarak da bilinen bu özel, yükseltilmiş yuvarlak buton, genellikle ekranın sağ alt köşesinde belirir ve **en önemli eylem** için kullanılır.




```
Scaffold(
  floatingActionButton: FloatingActionButton(
    child: const Icon(Icons.add),
    backgroundColor: Colors.red,
    onPressed: () { /* ... */ },
  ),
)
```

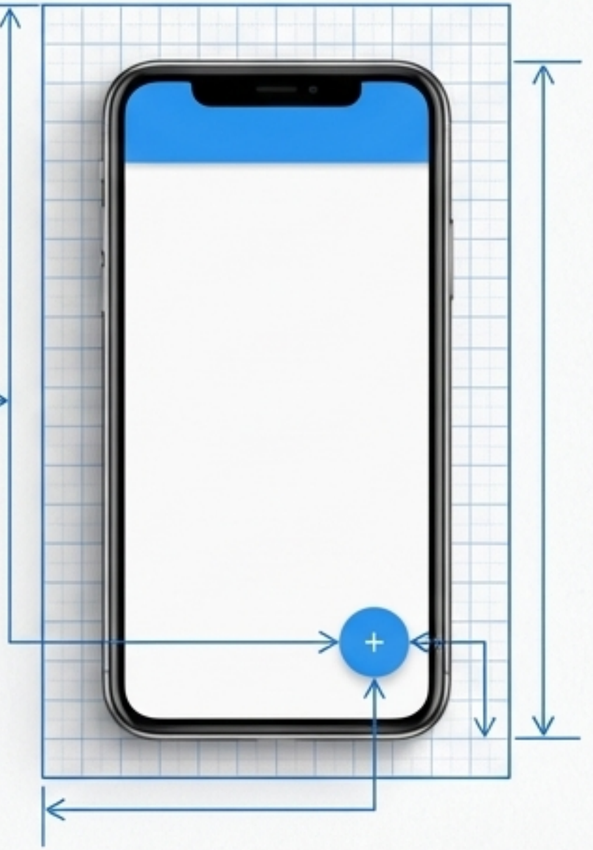
Temel Prensip

Bir `Scaffold` içinde yalnızca bir tane `FloatingActionButton` olabilir.

Hassas Konumlandırma: `floatingActionButtonLocation`

FAB'ın konumunu değiştirmek mi istiyorsunuz? `floatingActionButtonLocation` parametresini kullanarak `Scaffold` içinde birçok önceden tanımlanmış konumdan birini seçebilirsiniz.

```
Scaffold(
  floatingActionButton: FloatingActionButton(
    child: const Icon(Icons.add),
    onPressed: () {},
  ),
  floatingActionButtonLocation:
   FloatingActionButtonLocation.centerFloat,
)
```



Etkileşimli Elemanlar: Buton Çeşitleri

Butonlar, bir eylemi tetiklemenin en sezgisel yoludur. Flutter, materyal yönergelerini izleyen çeşitli buton widget'ları sunar.

RAISED BUTTON

`RaisedButton`

Dikdörtgen şekilli ve varsayılan yükseltiye (gölgeye) sahip tipik buton.

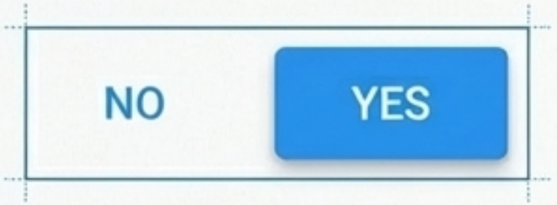
FLAT BUTTON

`FlatButton`

`RaisedButton'a benzer ancak yükseltisi (gölgesi) yoktur.

`IconButton`

İçeriği düz bir metin yerine bir materyal ikonu olan buton.



`ButtunBar`

Yatay bir sırada bir dizi butonu tutan bir konteyner.

Kullanıcıyla İletişim: `AlertDialog`

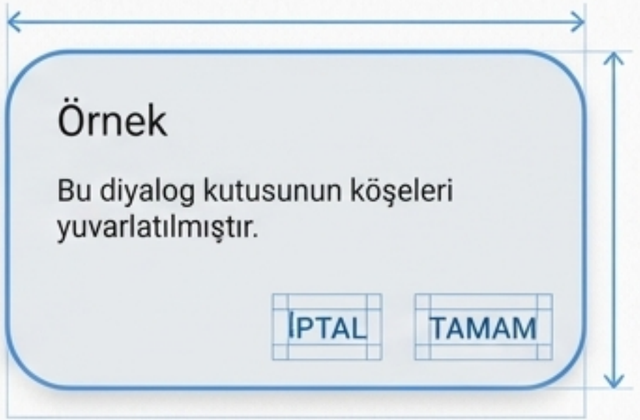
En yaygın diyalog türü, bir eylemin onaylanmasını veya reddedilmesini isteyen `AlertDialog`'dur. Ekranda görünmesi için `showDialog(...)` metodu kullanılmalıdır.

```
showDialog(
  context: context,
  builder: (BuildContext context) {
    return AlertDialog(
      title: const Text("Örnek"),
      content: const Text("Bu kitabı beğendiniz mi?"),
      actions: [
        FlatButton(
          child: const Text("Evet"),
          onPressed: () {},
        ),
        FlatButton(
          child: const Text("Elbette"),
          onPressed: () {},
        )
      ]
    );
  }
);
```

Diyalogları Şekillendirmek ve Kontrol Etmek

Diyaloğun kenarlıklarını shape parametresiyle tamamen özelleştirebilirsiniz.

```
AlertDialog(
  // ...
  shape: RoundedRectangleBorder(
    borderRadius: BorderRadius.circular(30),
  )
);
```



Diyaloğu kapatmak için `Navigator.pop(context)` kullanılır. Varsayılan olarak, kullanıcı diyalog dışına dokunarak da kapatabilir. `barrierDismissible: false` ayarı bunu engeller.

```
FlatButton(
  child: const Text("Kapat"),
  onPressed: () => Navigator.pop(context),
)
```

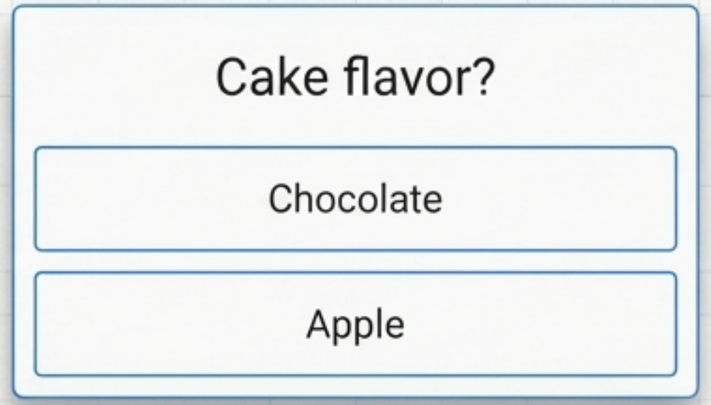
Mimarın Notu

Navigator sınıfı, sayfalar (route'lar) arasında gezinmek ve diyalogları kapatmak için kullanılır.

Özel Amaçlı Diyaloglar

Seçenek Sunmak

Kullanıcının bir dizi seçenek arasından seçim yapabildiği basit bir diyalog.




```
return SimpleDialog( title: ...,
  children: [ SimpleDialogOption(...) ] );
```

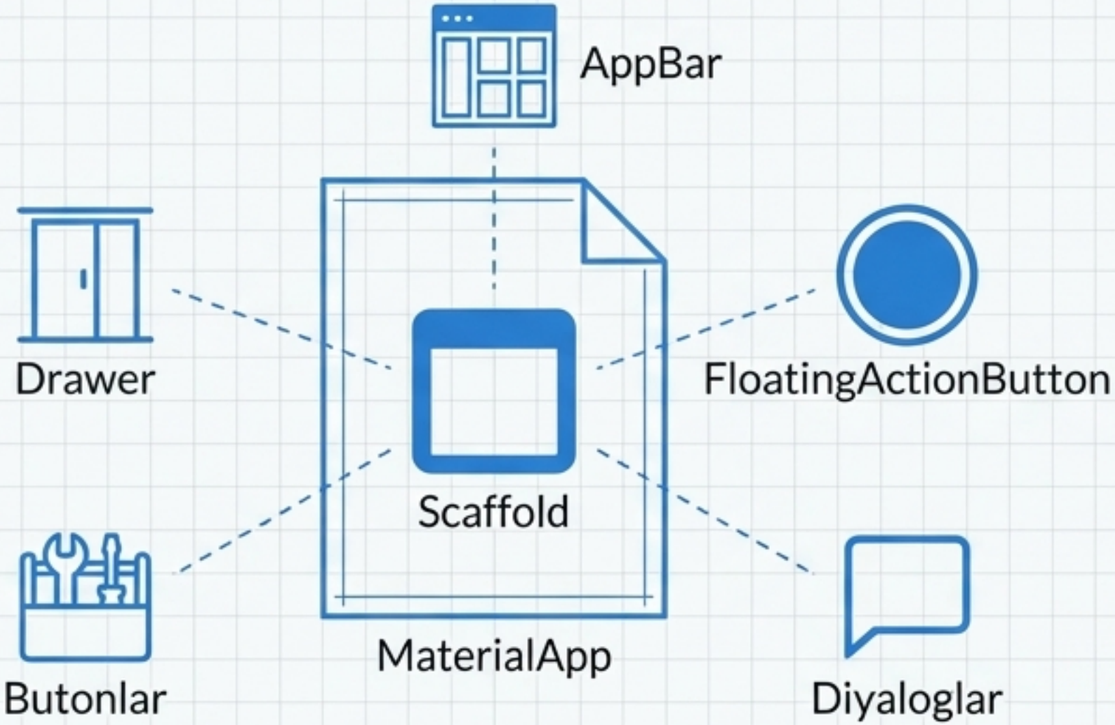
Alttan Bilgi Sunmak

Ekranın altından yukarı doğru kayan bir diyalog animasyonu oluşturur.


```
showBottomSheet( context: context, builder: ... );
```

Özet: Mimarın Araç Seti

Flutter'ın Materyal kütüphanesi, estetik ve işlevsel arayüzler inşa etmek için size eksiksiz ve birbiriyle uyumlu bir bileşen seti sunar.



Doğru araçlarla, her ekranı sağlam bir temel üzerine inşa edebilir ve kullanıcılarınız için sezgisel deneyimler yaratabilirsiniz.

Sadece Kod Değil, Bir Tasarım Felsefesi

Flutter’ın Materyal bileşenleri size yalnızca widget’lar sunmaz; on milyonlarca kullanıcının aşina olduğu kanıtlanmış bir tasarım dilinin gücünü ve tutarlılığını sunar.

Bu araç setini kullanarak, uygulamanızın temelini en başından itibaren kalite ve kullanılabilirlik üzerine kurarsınız.