Temelden Zirveye: Flutter ile Kusursuz iOS Arayüzleri

Cupertino Kütüphanesi Kullanım Rehberi

Neden iOS'ta 'Farklı' Hissediyor?

Flutter, platformlar arası geliştirme için harikadır, ancak varsayılan Material Design bileşenleri iOS kullanıcılarına yabancı gelebilir. Peki, uygulamanızın her pikselinin iPhone'da 'evindeymiş gibi' hissetmesini nasıl sağlarsınız?

Cevap: Cupertino Kütüphanesi

Flutter'ın Cupertino kütüphanesi, Apple'ın iOS tasarım dilini birebir yansıtan, kullanıma hazır bir bileşen setidir. `MaterialApp`'in iOS dünyasındaki karşılığı olan `CupertinoApp` ile yola çıkıyoruz.

Her Şeyin Başlangıcı: `CupertinoApp`

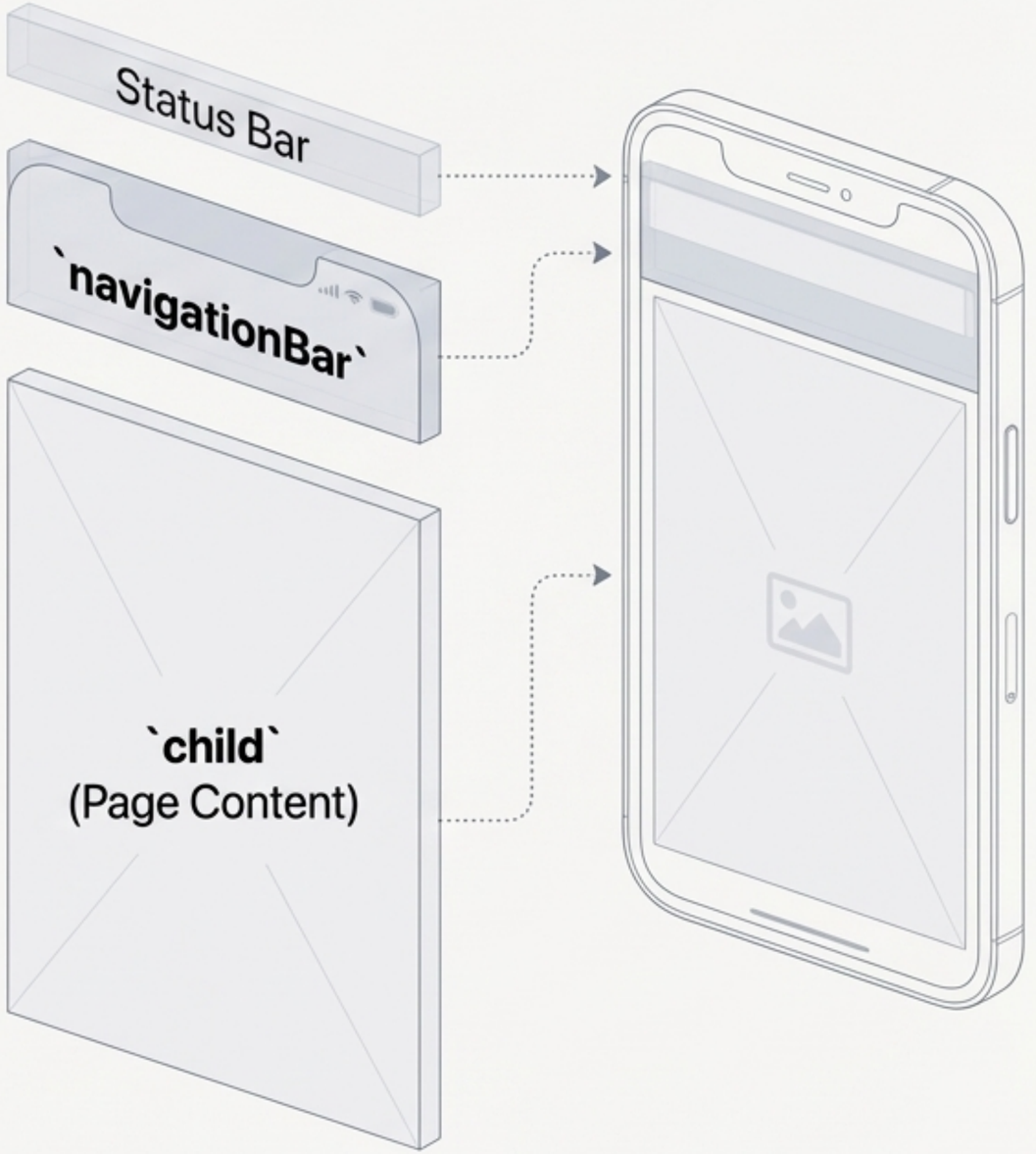
iOS tarzı bir Flutter uygulaması oluşturmanın ilk adımı, widget ağacınızın köküne `CupertinoApp` yerleştirmektir. Bu, uygulamanız için temel yapılandırmayı, temayı ve navigasyonu sağlar.

```dart
// main.dart
Widget build(BuildContext context) {
  return CupertinoApp(
    home: const CupertinoPageScaffold(
      navigationBar: CupertinoNavigationBar(
        middle: Text("Cupertino App"),
      ),
      child: Center(
        child: Text("Cupertino dünyasına hoş geldiniz!"),
      ),
    ),
  );
}
```

Uygulamanın İskeleti: `CupertinoPageScaffold`

Cupertino kütüphanesinde iki temel iskelet yapısı bulunur. `CupertinoPageScaffold`, en yaygın senaryo olan, üstte bir gezinme çubuğu (navigation bar) bulunan "düz" sayfalar oluşturmak için kullanılır.

*Ana Fikir:** Tek bir görünüm ve üst bar içeren sayfalar için ilk tercihiniz `CupertinoPageScaffold` olmalıdır.



`CupertinoPageScaffold'un Anatomisi

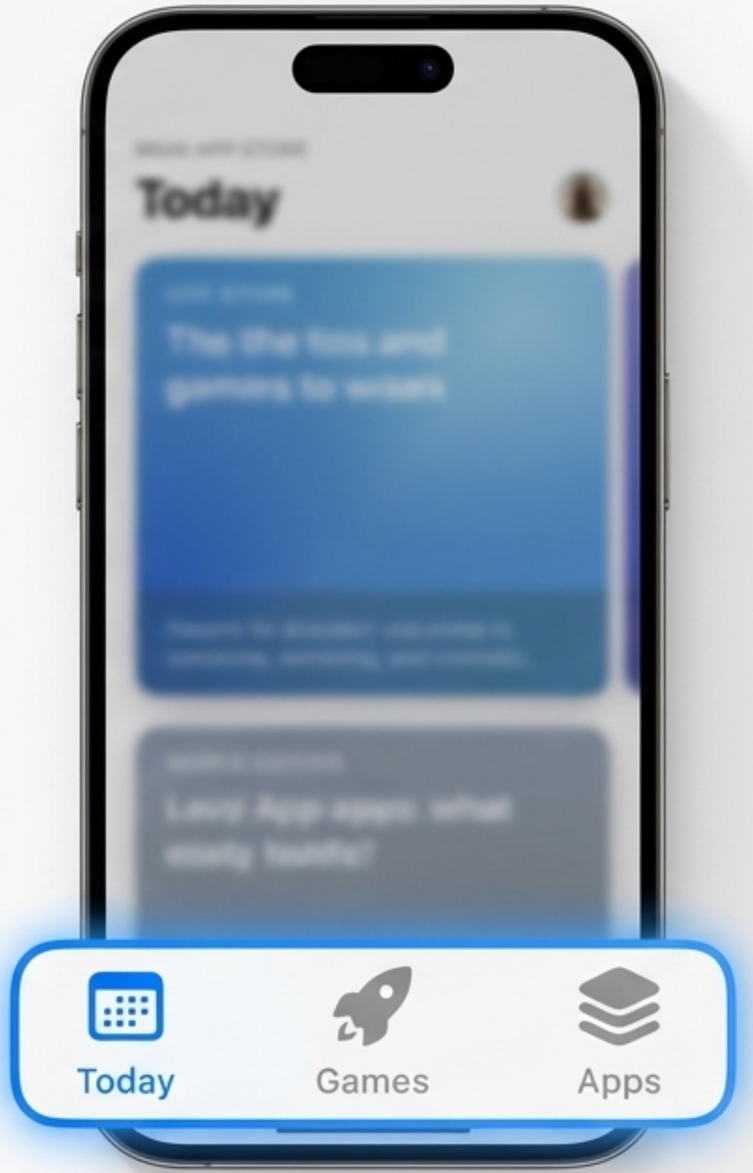
Bu iskelet, iki temel bölümden oluşur: sayfanın üst kısmındaki `navigationBar` ve geri kalan tüm içeriği barındıran `child`.

```
CupertinoApp(
  home: const CupertinoPageScaffold(
    // Sayfanın başlığını ve eylemleri barındıran üst çubuk
    navigationBar: CupertinoNavigationBar(
      middle: Text("Sayfa Başlığı"),
      trailing: Icon(CupertinoIcons.info),
    ),
    // Sayfanın gövdesi, tüm içeriğiniz burada yaşar
    child: Center(
      child: Text("Uygulamanın gövdesi")
    )
  ),
);
```

Birden Fazla Sayfa Birden Fazla Sayfa Gerektiğinde: `CupertinoTabScaffold`

Uygulamanızda alt navigasyon çubuğu üzerinden erişilen birden fazla ana bölüm varsa, doğru araç `CupertinoTabScaffold'dur. Bu yapı, iOS'un tipik sekme navigasyon düzenini kolayca oluşturmanızı sağlar.

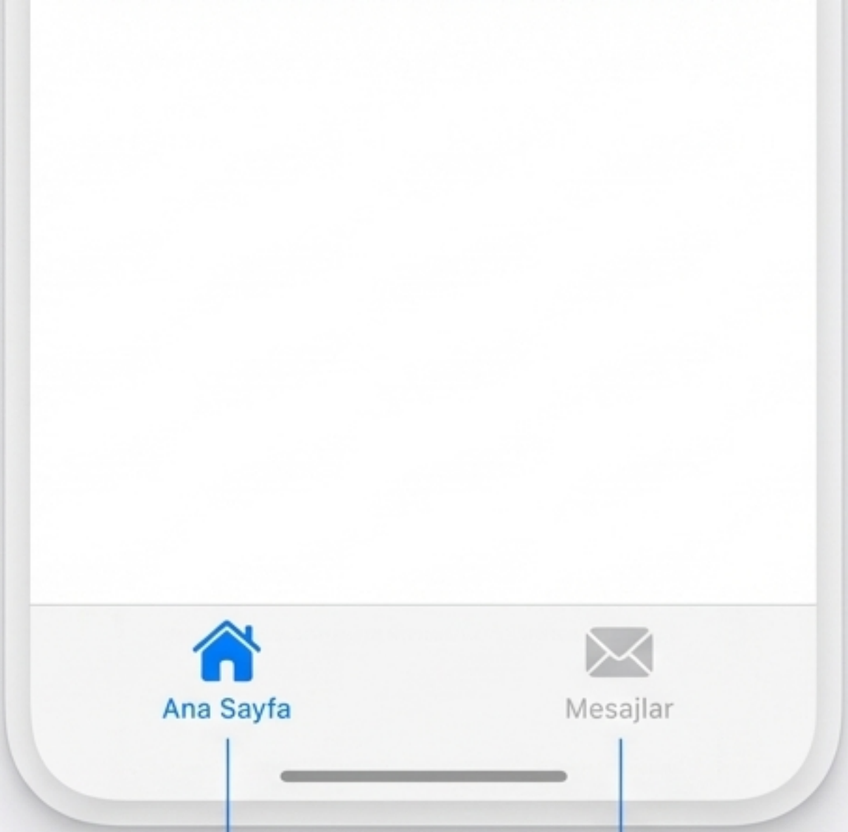
Önemli Not: Bu yapı, sekmeler arasında kaydırma (swipe) hareketini desteklemez; yalnızca dokunma (tap) ile çalışır.



Sekmeleri Oluşturmak: `CupertinoTabBar`

`CupertinoTabScaffold'un `tabBar` özelliği, görünecek sekmeleri tanımladığımız yerdir. `items` listesi, her sekme için bir `BottomNavigationBarItem` (ikon ve isteğe bağlı başlık) alır.

```
// CupertinoTabScaffold içinde...
tabBar: CupertinoTabBar(
    onTap: (index) { /* Sekme değiştiğinde tetiklenir */ },
    activeColor: Colors.blue, // Aktif sekmenin rengi
    items: const <BottomNavigationBarItem>[
      BottomNavigationBarItem(
        icon: Icon(CupertinoIcons.home),
        label: 'Ana Sayfa',
      ),
      BottomNavigationBarItem(
        icon: Icon(CupertinoIcons.mail),
        label: 'Mesajlar',
      ),
    ],
  ),
// ...
```

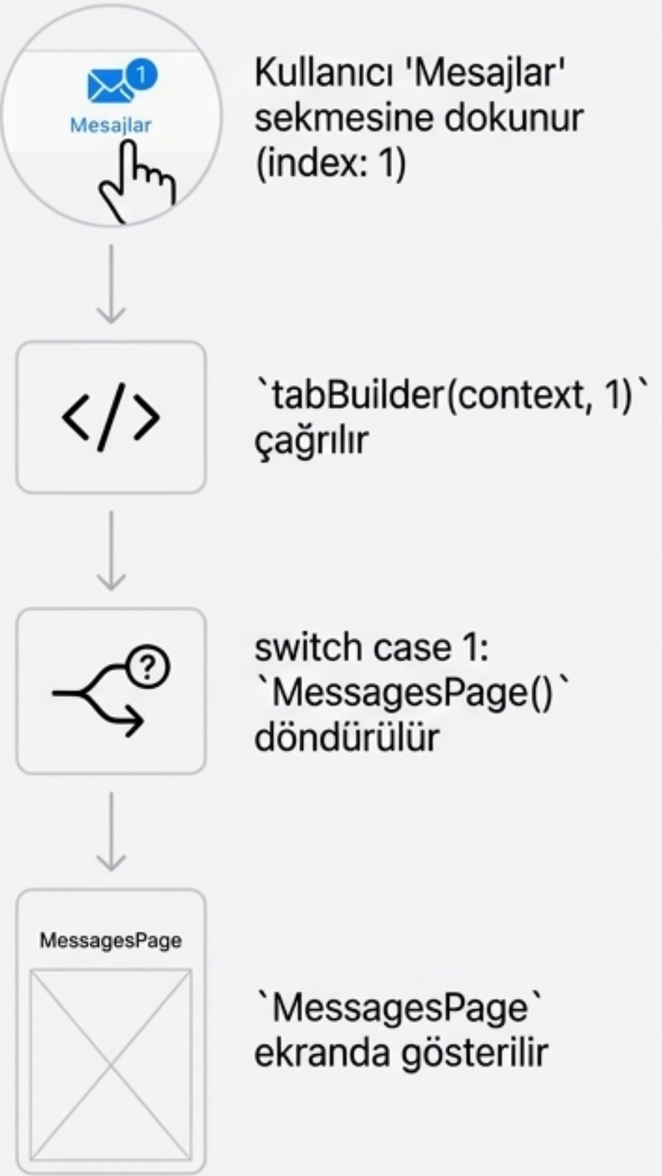


Her Sekmeye Bir Sayfa: `tabBuilder`

Sekmeler tanımlandıktan sonra, `tabBuilder` fonksiyonu hangi sekmenin hangi sayfayı (widget) göstereceğini belirler. Kullanıcı bir sekmeye dokunduğunda, `tabBuilder` ilgili `index` ile çağrılır.

Performans Notu: Kullanılmayan sekmelerin içeriği, daha sonra hızlıca yeniden kullanılabilmek için widget ağacında otomatik olarak önbelleğe alınır.

```
// CupertinoTabScaffold içinde...
tabBuilder: (BuildContext context, int index) {
  return CupertinoTabView(
    builder: (BuildContext context) {
      switch (index) {
        case 0:
          return const HomePage(); // Ana Sayfa içeriği
        case 1:
        default:
          return const MessagesPage(); // Mesajlar içeriği
      }
    },
  );
},
```



Etkileşimi Başlatmak: `CupertinoButton`

`CupertinoButton`, iOS'un standart, kenarlıksız ve düz (flat) düğme stilini yansıtır. Varsayılan olarak arka plan rengi yoktur, ancak `color` özelliği ile kolayca özelleştirilebilir.

```
CupertinoButton(
  child: const Text("iOS Düğmesi",
    style: TextStyle(
      color: CupertinoColors.white
    ),
  ),
  color: CupertinoColors.activeBlue,
  onPressed: () {
    // Düğmeye basıldığında çalışacak kod
  },
),
```

Kullanıcıyla Diyalog: `CupertinoAlertDialog`

Bazen kullanıcıyı bilgilendirmek veya bir eylemini onaylatmak gerekir. `CupertinoAlertDialog`, iOS'un standart uyarı diyaloglarını oluşturmak için kullanılır. Genellikle **`showDialog()`** fonksiyonu ile ekranda gösterilir.

Temel Bir Uyarı Diyaloğu Oluşturmak

Bir `CupertinoAlertDialog` oluşturmak için `title`, `content` ve kullanıcıya seçenekler sunan `actions` listesini tanımlamanız yeterlidir.

```
showDialog(
  context: context,
  builder: (context) {
    return CupertinoAlertDialog(
      title: const Text("Cupertino Uyarısı"),
      content: const Text("Bu bir iOS uyarı diyaloğudur."),
      actions: [
        CupertinoButton(
          child: const Text("Tamam"),
          onPressed: () => Navigator.pop(context),
        )
      ],
    );
  }
);
```

Doğru Araç: `CupertinoDialogAction`

Diyaloglar içindeki eylem düğmeleri için genel `CupertinoButton` yerine `CupertinoDialogAction` kullanmak daha semantik ve güçlü bir yaklaşımdır. Bu widget, eylemin doğasına göre düğmenin stilini otomatik olarak ayarlar.

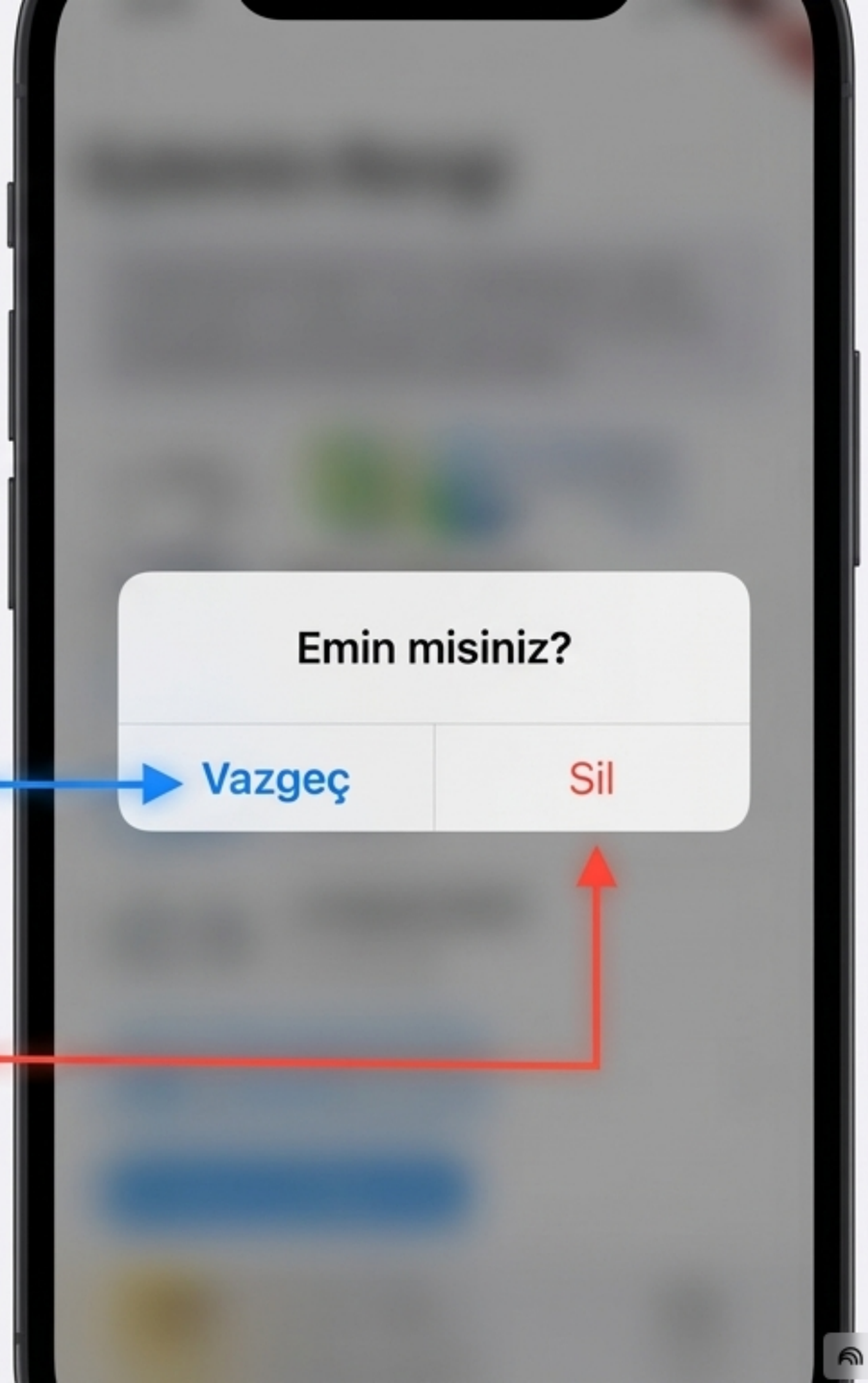
Ana Fikir: İki önemli özelliği vardır: `isDefaultAction` ve `isDestructiveAction`.

`CupertinoButton` `CupertinoDialogAction`

Eylemin Rengi: `isDefault` vs `isDestructive`

`isDestructiveAction: true` parametresi, metni "yıkıcı" eylemler (örn: Sil) için kırmızı yapar.
`isDefaultAction: true` ise, metni varsayılan veya onaylama eylemleri (örn: İzin Ver) için standart mavi ve kalın yapar.

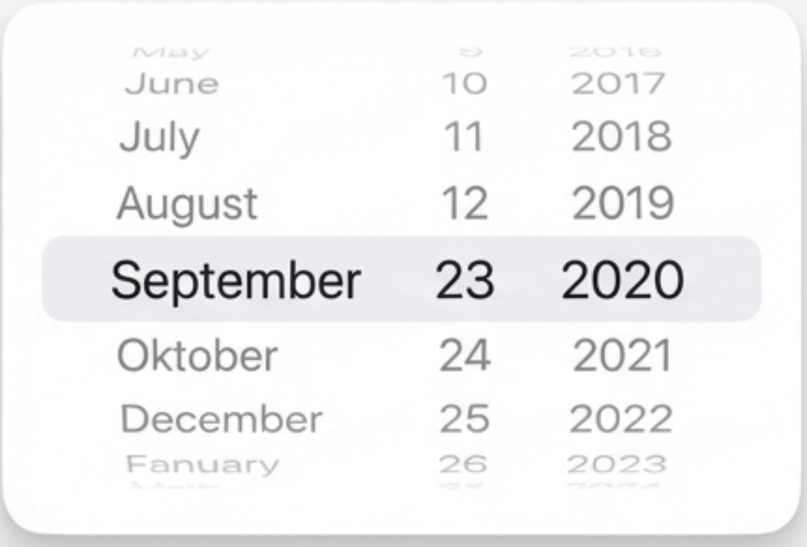
```
CupertinoAlertDialog(
  title: Text("Emin misiniz?"),
  actions: [
    CupertinoDialogAction(
      isDefaultAction: true,
      child: const Text("Vazgeç"),
      onPressed: () { /* ... */ },
    ),
    CupertinoDialogAction(
      isDestructiveAction: true,
      child: const Text("Sil"),
      onPressed: () { /* ... */ },
    ),
  ],
);
```



Cupertino Ekosistemi Büyüyor

Bu sunumda temel yapıları ve en yaygın bileşenleri ele aldık. Cupertino kütüphanesi, Materyal kütüphanesine göre daha az bileşen içerse de, Flutter ekibinin yol haritasında bu koleksiyonu sürekli olarak genişletmek yer alıyor.

Sonraki Adım: iOS için özel olarak tasarlanmış `CupertinoDatePicker`, gibi hezireymeın `CupertinoDatePicker`, `CupertinoSlider`, `CupertinoSwitch` gibi daha birçok bileşeni keşfetmek için resmi dokümantasyona göz atın.



Resmi Cupertino Widget Kataloğuna Göz Atın