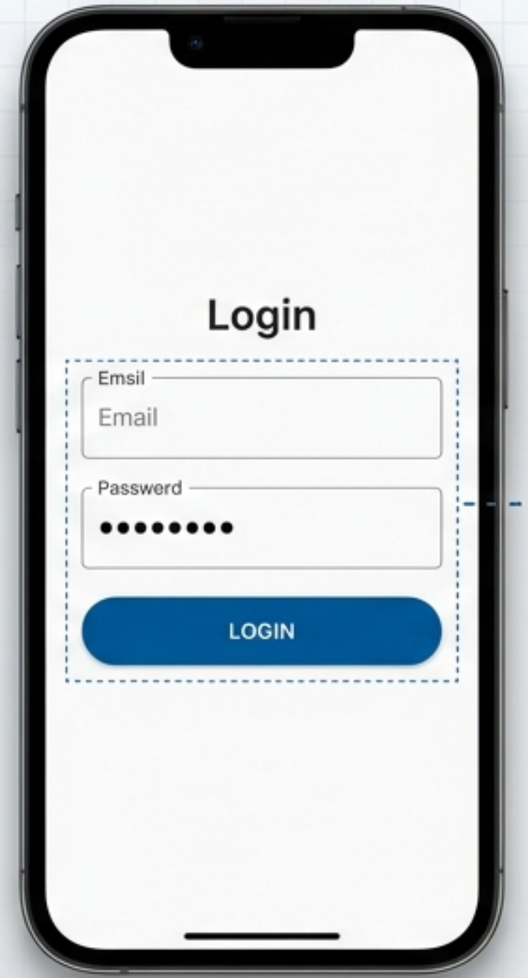
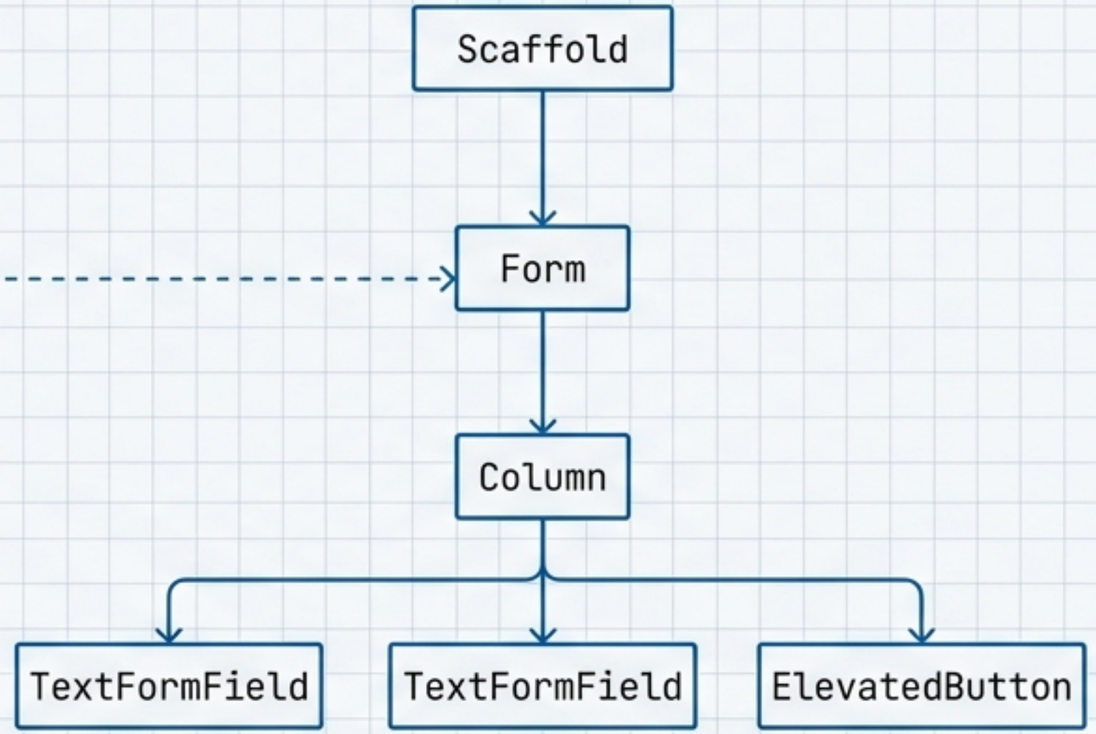
Flutter'da Etkileşim Tasarımı: Formlar ve Doğrulama

Bölüm 19.1 Özeti - Tepkisel ve Güvenli Bir Login Ekranı İnşası





Kapsam: Yerleşim • Kimlik • Dekorasyon • Mantık • Hata Yönetimi

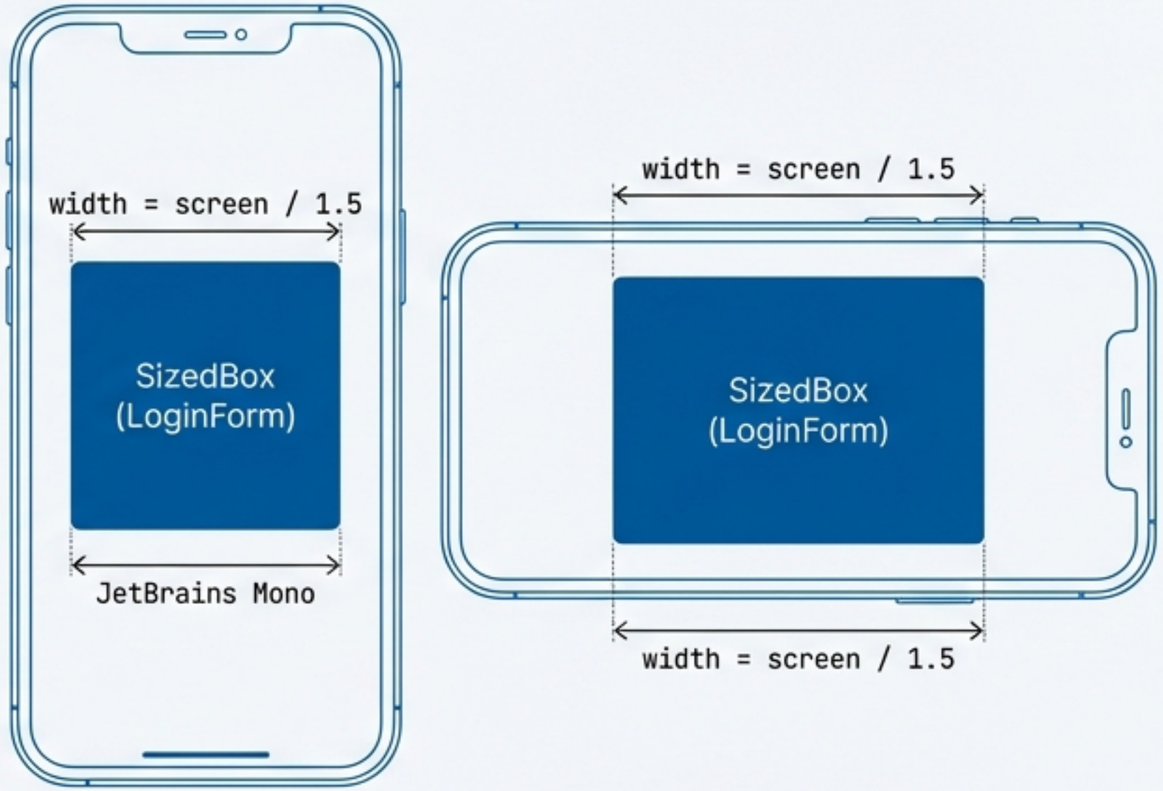
Temel Yapı: Duyarlı Yerleşim (Responsive Layout)

Code Card (Flutter Blue)

```dart
LayoutBuilder(
  builder: (context, dimensions) {
    // (a) Dinamik genişlik hesabı
    final width = dimensions.maxWidth / 1.5;
    final height = dimensions.maxHeight / 3;

    return Center(
      // (b) Basit kutu yapısı
      child: SizedBox(
        width: width,
        height: height,
        child: LoginForm(),
      ),
    );
  },
),
```

Right Visual Diagram Column



Bottom Text Area

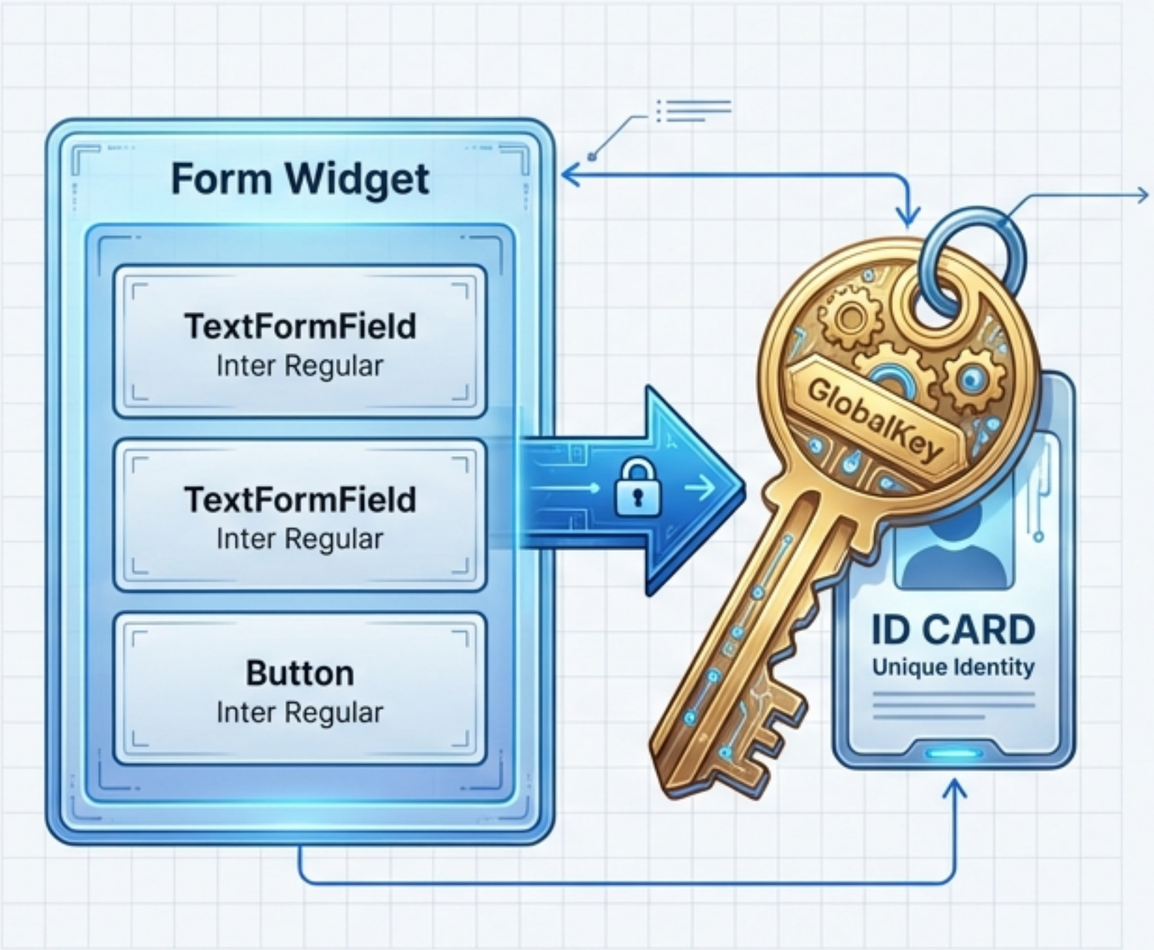
- **LayoutBuilder Avantajı:** Ekran döndürüldüğünde veya tablet kullanıldığında arayüzün tutarlılığını korur.
- **Mantık:** Form genişliği ekranın 2/3'ü, yüksekliği ise 1/3'ü olarak hesaplanır.
- **Neden Container Değil?** Stil veya efekt gerekmediği için **SizedBox** performans açısından daha uygundur.

Form Kapsayıcısı ve GlobalKey Kimliği

Form Widget: TextFormField gibi çoklu giriş alanlarını tek bir çatı altında toplar.

GlobalKey: Widget ağacı (widget tree) içerisinde forma benzersiz bir kimlik kazandırır.

Kritik: Doğrulama (validation) işlemini tetiklemek için bu anahtara ihtiyacımız vardır.

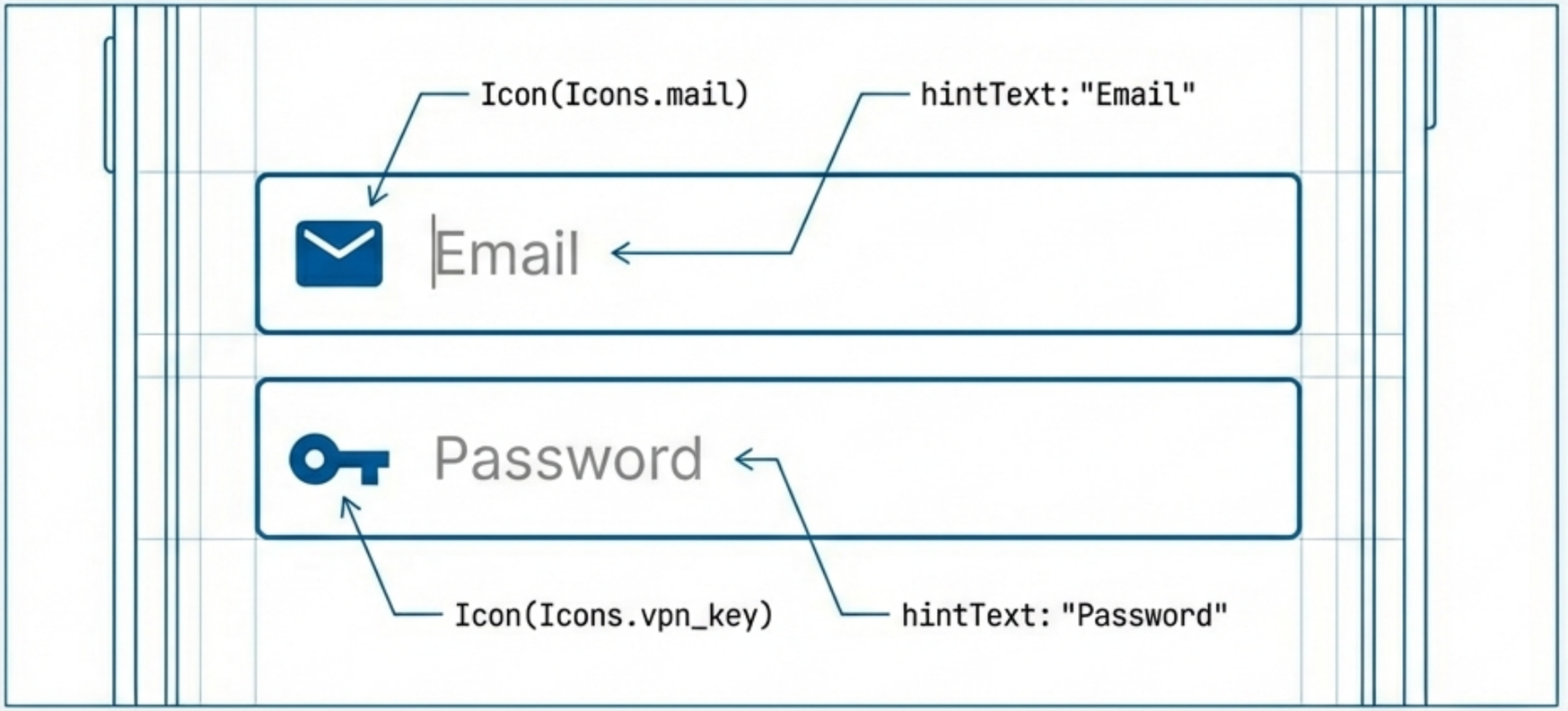


</> CODE BLOCK JetBrains Mono

```
class LoginForm extends StatelessWidget {
  // Kimlik kartı oluşturma
  final _key = GlobalKey();

  @override
  Widget build(BuildContext context) {
    return Form(
      key: _key, // Kimliği atama
      child: Column(...),
    );
  }
}
```

Görsel Tasarım: TextFormField ve Dekorasyon

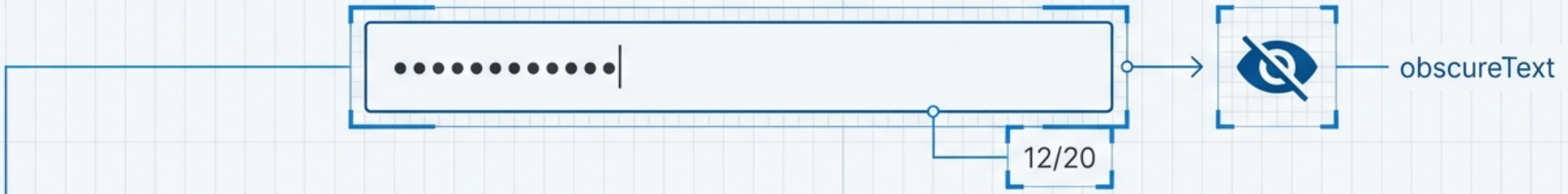



```
TextFormField(
  decoration: const InputDacoration(
    icon: Icon(Icons.mail),
    hintText: 'Email',
  ),
  validator: _validateEmail,
),
```

- **InputDecoration:** Metin kutusunun görünümünü özelleştirmek için kullanılır.
- **İpucu Metin:** Kullanıcıya ne yazması gerektiğini belirtir (hintText).
- **Görsel İkonlar:** `icon: Icon(Icons.vpn_key)` gibi parametrelerle kullanıcı deneyimi zenginleştirilir.

JetBrains Mono

Kısıtlamalar ve Biçimlendirme



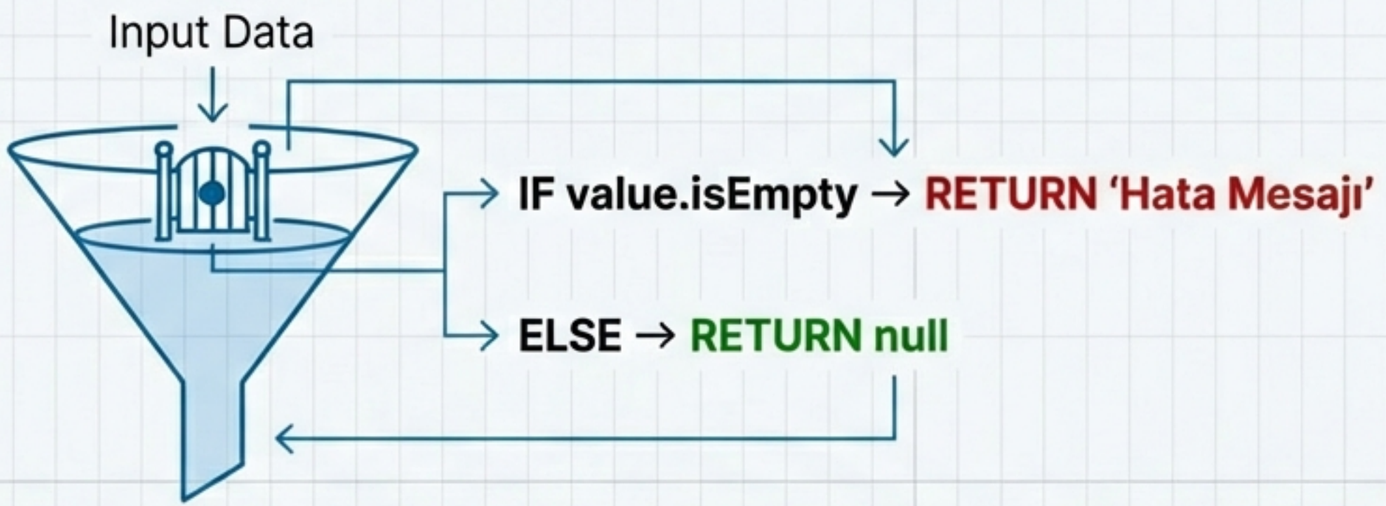
JetBrains Mono

Maksimum uzunluk

```
TextFormField(
  obscureText: true, // Şifreyi gizle
  maxLength: 20,     // Maksimum uzunluk
  decoration: InputDaCorption(
    counterText: '', // Sayacı gizle (opsiyonel)
  ),
),
```

1. **Şifre Gizleme:** `obscureText: true` parametresi ile karakterler noktalarla gizlenir.
2. **Karakter Sınırı:** `maxLength: 20` ile giriş uzunluğu kısıtlanır.
3. **Sayaç Yönetimi:** Sağ altta çıkan sayacı gizlemek için `counterText: ''` kullanılabilir.

Kapı Bekçileri: Doğrulama Mantığı (Validation)



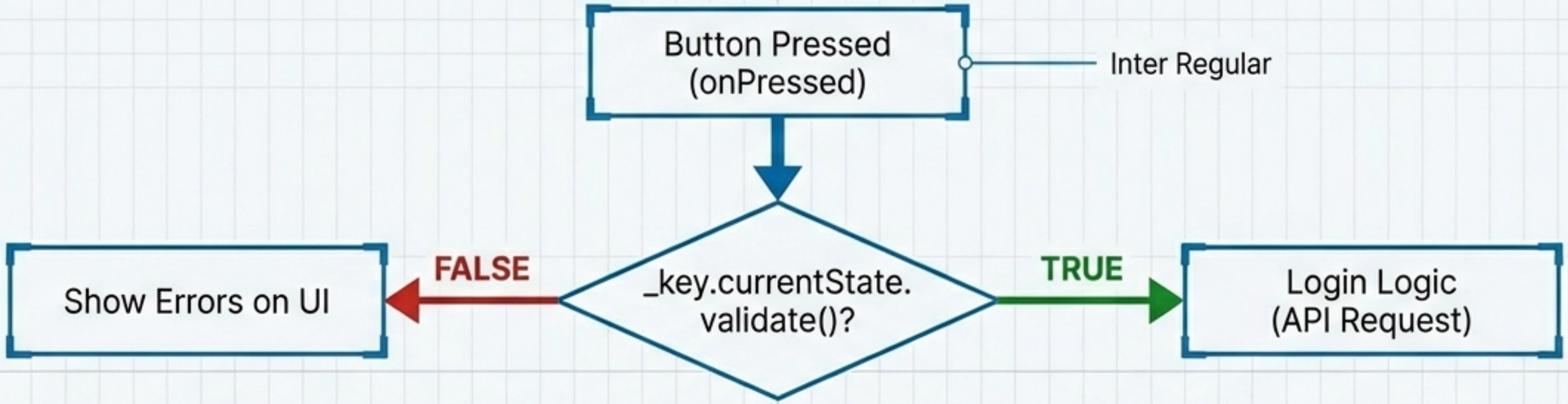

```dart
String? _validateEmail(String value) {
  if (value.isEmpty) {
    return 'Field cannot be empty'; // Hata
  }
  return null; // Başarılı
}

String? _validatePassword(String value) {
  if (value.length < 8) {
    return 'At least 8 chars!'; // Hata
  }
  return null; // Başarılı
}
```

- Modern Technical Blueprint **Mekanizma:** Fonksiyon bir String (hata mesajı) veya null (başarılı) döndürür.
- **Kurallar:** E-posta boş olamaz; Şifre en az 8 karakter olmalıdır.
- **Geri Bildirim:** Hata dönülürse, metin kutusunun altında kırmızı uyarı belirir.

JetBrains Mono

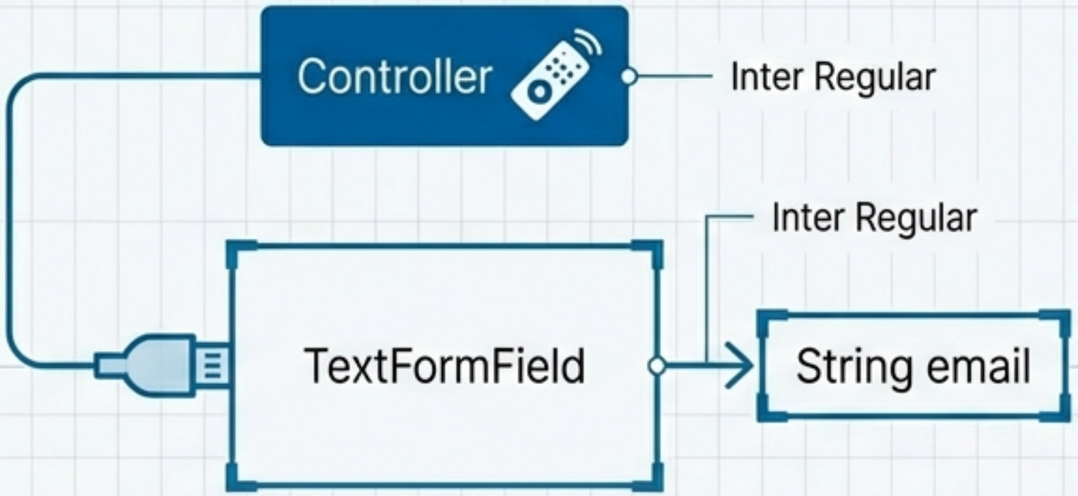
Tetikleyici: Formu Gönderme İşlemi




```dart
RaisedButton(
  child: const Text('Login'),
  onPressed: () {
    // Tüm alanları kontrol et
    if (_key.currentState?.validate() ?? false) {
      _login(); // İşlem Başarılı
    } else {
      // Hataları göster
    }
  },
)
```

- **Bağlantı:** RaisedButton 'onPressed' olayı ile tetiklenir.
- **Toplu Kontrol:** validate() metodu formdaki tüm alanları aynı anda kontrol eder.
- **Sonuç:** Eğer tek bir alan bile hatalıysa işlem durdurulur.

Veriyi Okumak: TextController Kullanımı



1. Definition:

```
final emailController = TextAittingController();
```

2. Usage:

```
TextFormField(controller: emailController, ...)
```

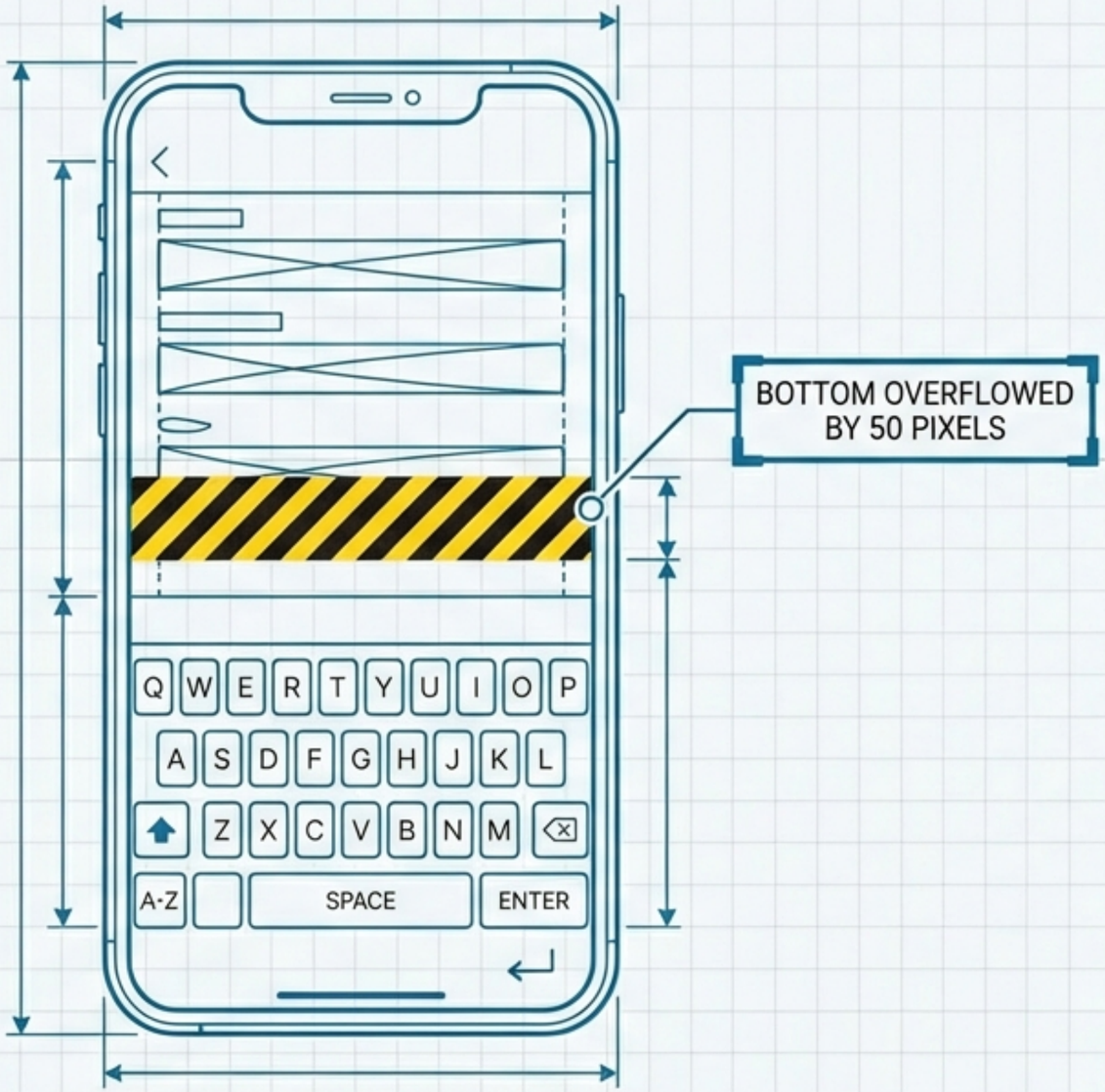
3. Retrieval:

```
final email = emailController.text;
```

⚠ **Bellek Yönetimi:** Controller'lar kullanıldıktan sonra mutlaka dispose() metodu ile yok edilmelidir.

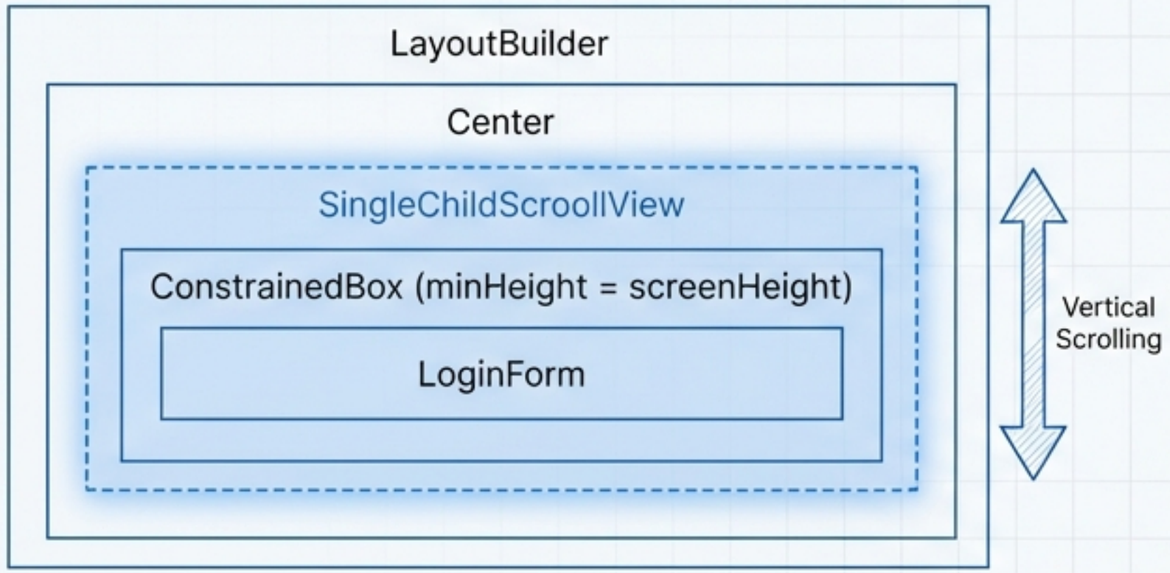
İpucu: 'selection' özelliği ile seçili metin aralıkları (start/end) manipüle edilebilir.

UX Krizi: Klavye ve Taşma (Overflow) Hataları



- **Sorun:** Klavye açıldığında dikey alan daralır.
- **Sonuç:** Form elemanları ekrana sığmaz ve Flutter 'Overflow' hatası verir.
- **Kullanıcı Deneyimi:** Kullanıcı ne yazdığını göremez veya formu gönderemez.

Çözüm A: Kaydırma Davranışı (SingleChildsorollView)



- **Strateji:** Alan yetersiz kaldığında arayüzü kaydırılabilir hale getirmek.
- **ConstrainedBox**: İçeriğin minimum yüksekliğini ekran yüksekliğine eşitler, böylece boşluk varsa form ortalanır, yoksa kaydırılır.
- **En İyi Pratik:** Formlar için en güvenli ve esnek yöntemdir.

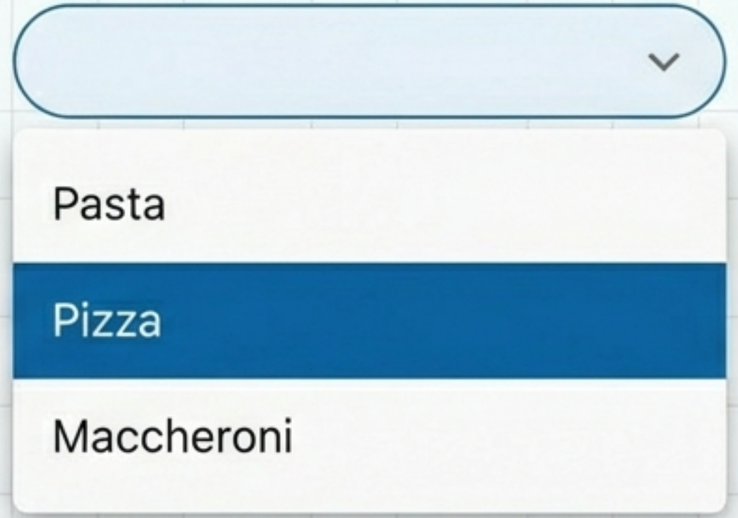
```
SingleChildScrollView(
  child: ConstrainedBox(
    constraints: BoxConstraints(
      minHeight: height, // Ekran yüksekliğini koru
    ),
    child: LoginForm(),
  ),
)
```

Çözüm B: Klavye Örtüşmesi (Overlay)

- **Yöntem:** Klavye arayüzü sıkıştırmaz, bunun yerine arayüzün ‘üzerine’ biner.
- **Avantaj:** Hızlı çözüm, kaydırma mantığı gerektirmez.
- **Dezavantaj:** Klavye form alanlarını kapatabilir. Kullanıcı ne yazdığını göremez. Formlar için önerilmez.

```
Scaffold(
  resizeToAvoidBottomInset: false, // Yeniden boyutlandırmayı kapat
  body: MyBody(...),
)
```

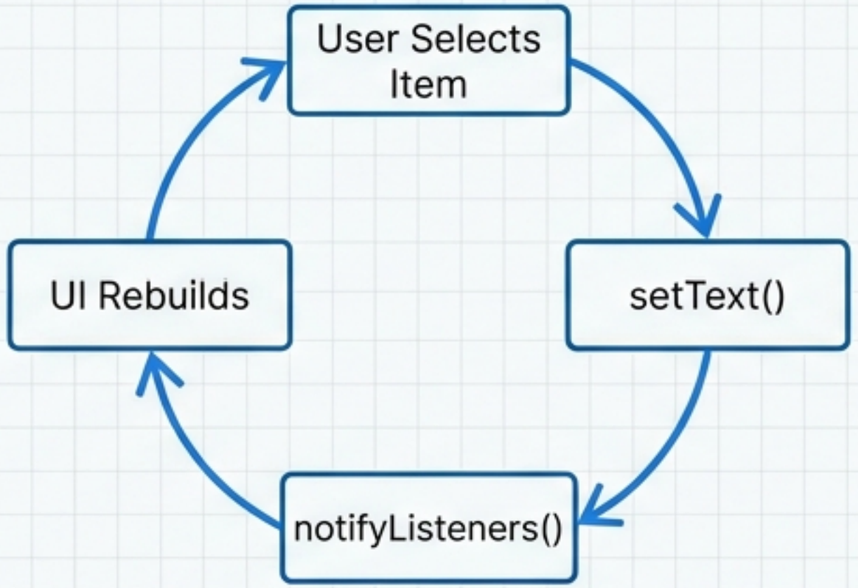
İleri Düzey Girişler: Açılır Menüler (Dropdowns)



- **DropdownButtonFormField**: Form uyumlu açılır menü widget'ı.
- **DropdownMenuItem**: Seçeneklerin her biri.
- **Map Fonksiyonu**: Listeyi menü elemanlarına dönüştürür.

```
items: _list.map((item) {
   return DropdownMenuItem(
     value: item,
     child: Text(item),
   );
}).toList(),
```

Durum Yönetimi (State Management)




```dart
class DropdownText with ChangenNoifier {
  var _text = '';
  void setText(String value) {
    _text = value;
    notifyListeners(); // Arayüzü güncelle
  }
}
```

- **Provider:** Seçilen değeri UI'dan bağımsız tutar.
- **NotifyListeners:** Seçim değiştiğinde arayüzü otomatik olarak yeniden çizilmeye (rebuild) zorlar.
- **Consumer:** Widget ağacında değişikliği dinleyen yapı.

Özet ve En İyi Uygulamalar

- ✓ **Yapı: LayoutBuilder** kullanarak her ekrana uyumlu (responsive) alanlar yaratın.
- ✓ **Kimlik:** Doğrulama işlemleri için her zaman **GlobalKey** kullanın.
- ✓ **Güvenlik:** Kullanıcıyı kısıtlayın (**maxLength**, **obscureText**) ve girdileri doğrulayın (**validator**).
- ✓ **Erişilebilirlik:** Klavye taşmalarını önlemek için **SingleChildScreellView** sarmalayıcısını unutmayın.

Teşekkürler.

Based on Flutter Complete Reference, Chapter 19.