

FLUTTER

Görsel Seçme ve Dosyalar

29. hafta • Cihaz Özellikleri ve Kalite

36 hafta, 72 bağımsız çalışan Flutter projesi, gerçek uygulama ekranları, uygulamalı dersler ve öğrenci geri bildiriminden sonra incelenebilen çözüm kaynakları.

Bu kitap güncel ders kaynaklarından otomatik üretilmiştir; özet ya da jenerik şablon değildir.

Öğrenme Haritası

Hafta	Ders	Projeler
29	Görsel Seçme ve Dosyalar	Profil Fotoğrafı • Ödev Kanıtı

29. Hafta — Görsel Seçme ve Dosyalar

Bu hafta ne yapacağız?

Telefonun kamera ve galeri özelliklerini Flutter uygulamasına bağlayacağız. `image_picker` paketini kuracak, kullanıcıdan kaynak seçimi alacak, dönen `XFile` nesnesini baytlara dönüştürecek ve seçilen resmi ekranda göstereceğiz. Kullanıcının seçimi iptal etmesi, izin vermemesi veya cihaz hizmetinin hata üretmesi de uygulamanın normal akışının bir parçası olarak ele alınacak.

Kazanımlar

- Bir Flutter paketini `pubspec.yaml` dosyasına ekleme
- Kamera ve galeriyi `ImageSource` ile ayırt etme
- `XFile` üzerinden dosya adı ve baytları okuma
- `Uint8List` verisini `MemoryImage` ve `Image.memory` ile gösterme
- İptal, izin ve hata durumlarında kullanıcıya anlaşılır geri bildirim verme
- Cihaz bağımlılığını arayüz sınıfıyla ayırarak test edilebilir kod yazma
- Gerçek kamera açmadan sahte seçiciyle widget testi yapma

Hazırlık: `image_picker` paketini ekleme

`pubspec.yaml` dosyasına paket eklenir:

```
dependencies:  
  flutter:  
    sdk: flutter  
  image_picker: ^1.2.1
```

Ardından terminalde `flutter pub get` çalıştırılır. Gerçek Android/iOS uygulamasında kamera ve fotoğraf erişim izinleri platform yapılandırmasına da eklenmelidir.

Çözümlü uygulama — Profil Fotoğrafı Seçme

1. Uygulamanın kullanacağı veri modeli

Dosyanın yalnızca yolunu saklamak yerine adı ve okunmuş baytları birlikte taşıyoruz:

```
class SecilenGorsel {  
  const SecilenGorsel({required this.ad, required this.bytes});  
  
  final String ad;  
  final Uint8List bytes;  
  
  double get kb => bytes.length / 1024;  
}
```

Bu sayede arayüz dosya sistemi yoluna bağımlı kalmaz. Görseli bellekten gösterebilir ve boyut bilgisini kullanıcıya sunabilir.

2. Cihaz işlemini soyutlama

```
abstract class GorselSecici {  
  Future<SecilenGorsel?> sec(ImageSource kaynak);  
}
```

Arayüz doğrudan ImagePicker oluşturmaz. Böylece gerçek cihazda CihazGorselSecici, testte ise önceden belirlenmiş bir görsel döndüren sahte sınıf kullanılabilir.

3. Kamera veya galeriden görsel alma

```
final xFile = await picker.pickImage(  
  source: kaynak,  
  maxWidth: 1200,  
  imageQuality: 85,  
);  
  
if (xFile == null) return null;  
  
return SecilenGorsel(  
  ad: xFile.name,  
  bytes: await xFile.readAsBytes(),  
);
```

- source, kameranın mı galerinin mi açılacağını belirler.
- maxWidth, gereksiz büyüklükte dosyaları küçültmeye yardım eder.
- imageQuality, JPEG sıkıştırma kalitesini belirler.
- Kullanıcı seçim ekranını kapatırsa sonuç null olur.

4. Kaynağı düğmelerden gönderme

```
FilledButton.icon(  
  onPressed: () => sec(ImageSource.gallery),  
  icon: const Icon(Icons.photo_library_outlined),  
  label: const Text('Galeriden seç'),  
)
```

Kamera düğmesi aynı metodu ImageSource.camera ile çağırır. Tekrar eden seçim kodu böylece iki ayrı metoda bölünmez.

5. Görseli bellekten gösterme

```
CircleAvatar(  
  radius: 82,  
  backgroundImage:  
    gorsel == null ? null : MemoryImage(gorsel!.bytes),  
  child: gorsel == null  
    ? const Icon(Icons.person, size: 86)  
    : null,  
)
```

Görsel seçilmeden önce kişi simgesi, seçimden sonra gerçek dosyanın baytları gösterilir.

6. İptal ve hata yönetimi

```

try {
  final sonuc = await widget.secici.sec(kaynak);
  if (!mounted) return;

  setState(() {
    seciliyor = false;
    if (sonuc != null) {
      gorsel = sonuc;
      mesaj = 'Görsel başarıyla seçildi';
    } else {
      mesaj = 'Seçim iptal edildi';
    }
  });
} catch (_) {
  if (!mounted) return;
  setState(() {
    seciliyor = false;
    mesaj = 'Görsel açılmadı. Zinleri kontrol edin.';
  });
}

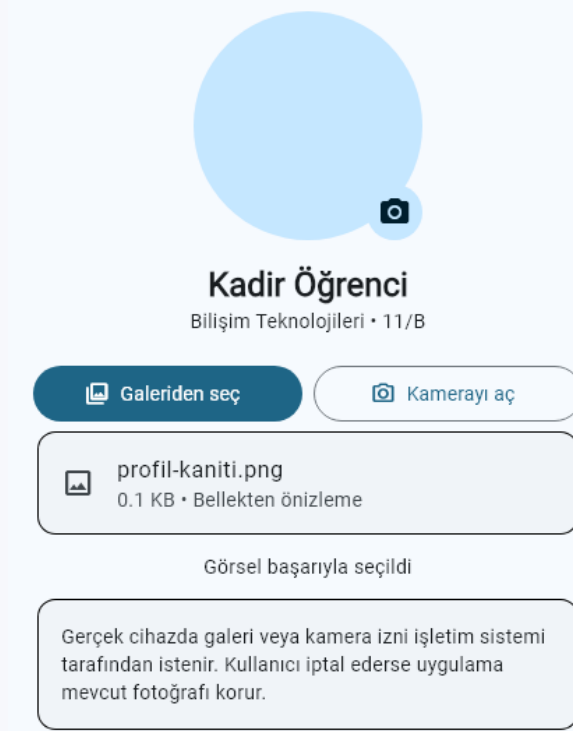
```

await sonrasında sayfa kapanmış olabilir. mounted kontrolü kapatılmış bir widget üzerinde setState çağrılmasını önler.

Gerçek Flutter ekranı

Aşağıdaki ekran Flutter widget testi sırasında uygulamanın kendisi tarafından render edildi. Test galeri kaynağını seçti, gerçek PNG baytlarını uygulamaya verdi, dosya adını ve başarı mesajını doğruladı.

Profil Fotoğrafları Seçme



Profil Fotoğrafları Seçme gerçek Flutter ekranı

Özgün uygulama — Ödev Kanıtı Hazırlama

Bu uygulamada öğrenci yaptığı çalışmanın ekran fotoğrafını kamera veya galeriden ekler, ne yaptığını açıklar ve kanıt paketini hazırlar.

Görsel seçimi

```
final sonuc = await widget.secici.sec(kaynak);
if (mounted && sonuc != null) {
  setState(() {
    gorsel = sonuc;
    durum = 'Görsel hazır';
  });
}
```

Zorunlu alanları doğrulama

```

void gonder() {
  if (gorsel == null || aciklama.text.trim().isEmpty) {
    setState(() => durum = 'Açıklama ve görsel zorunludur');
    return;
  }

  setState(
    () => durum = 'Ödev kanıtı öğretmene gönderilmeye hazır',
  );
}

```

Yalnızca dosya seçmek yeterli değildir. Öğrencinin yaptığı işlemi açıklaması da zorunludur.

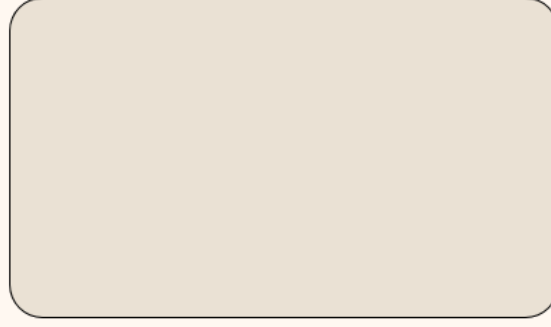
Seçilen resmi önizleme

```

child: gorsel == null
  ? const Column(
    mainAxisAlignment: MainAxisAlignment.center,
    children: [
      Icon(Icons.add_photo_alternate_outlined, size: 64),
      Text('Uygulamanızın ekranı ekleyin'),
    ],
  )
  : Image.memory(gorsel!.bytes, fit: BoxFit.cover),

```

Ödev Kanıtı Hazırla



 Fotoğraf çek

 Galeriden al

Ne yaptığınızı açıklayın

Form doğrulamasını üç farklı veriyle test ettim.



form-dogrulama.png
68 bayt okundu

 Kanıtı hazırla

Ödev kanıtı öğretmene gönderilmeye hazır

Ödev Kanıtı Hazırlama gerçek Flutter ekranı

Testin çalışma biçimi

Widget testi gerçek kamera penceresi açmaz. KanıtSecici arayüzünü uygulayan sahte sınıf geçerli PNG baytları döndürür. Test:

1. “Fotoğraf çek” düğmesine basar.
2. `ImageSource.camera` gönderildiğini doğrular.
3. Açıklama alanına metin yazar.
4. “Kanıtı hazırla” düğmesine basar.
5. Başarı geri bildirimini ve dosya adını doğrular.
6. Flutter tarafından render edilen ekranı PNG olarak kaydeder.

Sık yapılan hatalar

- pickImage sonucunun her zaman dolu olacağını varsaymak
- İzin reddini uygulama hatası gibi yönetmeden bırakmak
- Çok büyük görselleri özgün çözünürlükte belleğe almak
- await sonrasında mounted kontrolü yapmamak
- Dosya yolu ile dosya içeriğini birbirine karıştırmak
- Testte doğrudan cihaz kamerasına bağımlı olmak
- Görsel seçilmeden formu göndermeye izin vermek

Üç uygulama görevi

Kolay — Sınıf Kartı

Öğrenci adı, sınıfı ve galeriden seçilen fotoğrafıyla bir kimlik kartı hazırlayın. Seçim iptal edilirse önceki fotoğraf ekranda kalmalıdır.

Orta — Arıza Bildirim Formu

Okuldaki arızanın fotoğrafını kamera veya galeriden alın. Konum açıklaması ve arıza türü zorunlu olsun. Eksik alanlarda gönderimi durdurun.

İleri — Çoklu Proje Günlüğü

Birden fazla çalışma görseli eklenebilen proje günlüğü hazırlayın. Her görsel için açıklama, tarih ve kategori tutun. Büyük görselleri sıkıştırın; silme ve yeniden sıralama işlemlerini destekleyin.

Kontrol listesi

- Paket bağımlılığı doğru eklenmiş mi?
- Kamera ve galeri ayrı düğmelerden açılıyor mu?
- Kullanıcı iptal edince uygulama çalışmaya devam ediyor mu?
- Seçilen dosyanın adı ve baytları okunuyor mu?
- Görsel bellekten ekrana çiziliyor mu?
- İzin/hata mesajı kullanıcıya gösteriliyor mu?
- Gönderimden önce açıklama ve görsel doğrulanıyor mu?
- Cihaz erişimi test edilebilir bir arayüz arkasında mı?

Geri bildirim ve çözüm erişimi

Öğrenci kendi uygulamasını gönderip geri bildirim aldıktan sonra İpucunu göster seçeneği yalnızca kaynak seçimi, XFile ve doğrulama akışını açıklayacaktır. Örnek çözümü göster seçeneği ise bu dersteki gerçek Dart kaynak kodunu açacaktır. İndirme bağlantısı çalışan Flutter projesinin tamamını sunacaktır; ipucu ile tam çözüm aynı içerik olmayacaktır.

Kaynak projeler

- Konu uygulaması: 08 - Cihaz Özellikleri ve Kalite/Uygulamalar/Uygulama-29-Profil Fotoğraf
- Özgün uygulama: 08 - Cihaz Özellikleri ve Kalite/Uygulamalar/Ozgun-29-Ödev Kanıt