

FLUTTER

SQLite Temelleri

23. hafta • Yerel Veri Saklama

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
23	SQLite Temelleri	Öğrenci Kayıtları • Ev Envanteri

23. Hafta — SQLite Temelleri

Bu hafta ne yapacağız?

JSON dosyası küçük veri kümeleri için kullanışlıdır; ancak kayıtları koşula göre sorgulamak, benzersizlik kuralları koymak ve satırlara otomatik kimlik vermek için ilişkisel veritabanı daha uygundur. Bu hafta gerçek SQLite tabloları oluşturacak, INSERT ile kayıt ekleyecek ve SELECT sorgularıyla listeleyeceğiz.

Kazanımlar

- Veritabanı, tablo, satır, sütun ve birincil anahtar kavramlarını açıklama
- sqflite ile veritabanı açma ve ilk açılışta tablo oluşturma
- Modeli Map<String, Object?> yapısına dönüştürme
- Parametrelili INSERT ve SELECT işlemleri uygulama
- UNIQUE, NOT NULL ve CHECK kısıtlarını kullanma
- WHERE, ORDER BY ve whereArgs ile güvenli sorgu yazma

Kurulum

```
dependencies:  
  flutter:  
    sdk: flutter  
  path: ^1.9.1  
  sqflite: ^2.4.2
```

Çözümlü uygulama — SQLite Öğrenci Kayıtları

1. Veritabanı yolunu oluşturma

```
final yol = join(await getDatabasesPath(), 'ogrenciler.db');  
final depo = OgrenciDeposu(databaseFactory, yol);
```

join, platformun kullandığı klasör ayıracını doğru biçimde ekler.

2. Tablo şeması

```
CREATE TABLE ogrenciler(  
  id INTEGER PRIMARY KEY AUTOINCREMENT,  
  numara TEXT NOT NULL UNIQUE,  
  ad TEXT NOT NULL,  
  sinif TEXT NOT NULL  
)
```

- id: SQLite tarafından otomatik artırılır.
- NOT NULL: alanın boş veritabanı değeri olmasını engeller.
- UNIQUE: aynı öğrenci numarasının ikinci kez eklenmesini engeller.

3. Veritabanını yalnız gerektiğinde açma

```
Database? _db;

Future<Database> get db async => _db ??= await factory.openDatabase(
  yol,
  options: OpenDatabaseOptions(
    version: 1,
    onCreate: (db, _) => db.execute(createTableSql),
  ),
);
```

Bu yöntem aynı sayfada her işlem için yeni bağlantı açılmasını önler.

4. Modeli ekleme

```
Future<int> ekle(Ogrenci ogrenci) async {
  return (await db).insert(
    'ogrenciler',
    ogrenci.toMap(),
    conflictAlgorithm: ConflictAlgorithm.abort,
  );
}
```

Dönen int, eklenen satırın otomatik kimliğidir. Aynı numara eklenirse veritabanı DatabaseException üretir ve arayüz kullanıcıya Türkçe hata gösterir.

5. Kayıtları sorgulama

```
final maps = await (await db).query(
  'ogrenciler',
  orderBy: 'ad COLLATE NOCASE',
);

return maps.map(Ogrenci.fromMap).toList();
```

Gerçek uygulama ekranı

Test gerçek bellek içi SQLite motorunda tabloyu oluşturmuş, Ece Yılmaz kaydını INSERT etmiş ve SELECT sonucunda numara 1124'ün geri geldiğini doğrulamıştır.

SQLite Öğrenci Kayıtları

Numara

Sınıf

Ad soyad

Öğrenciyi veritabanına ekle

1 kayıt

SELECT • ORDER BY ad

11/B

Ece Yılmaz
No: 1124

#1

SQLite Öğrenci Kayıtları gerçek Flutter ekranı

Özgün uygulama — SQLite Ev Envanteri

Tablo kısıtı

```
CREATE TABLE esyalar(  
  id INTEGER PRIMARY KEY AUTOINCREMENT,  
  ad TEXT NOT NULL,  
  oda TEXT NOT NULL,  
  adet INTEGER NOT NULL CHECK(adet > 0)  
)
```

CHECK(adet > 0), arayüz doğrulaması unutulsa bile sıfır veya negatif adedin veritabanına girmesini engeller.

Odaya göre parametrelili sorgu

```
final maps = await database.query(
  'esyalar',
  where: oda == null ? null : 'oda = ?',
  whereArgs: oda == null ? null : [oda],
  orderBy: 'oda, ad',
);
```

Kullanıcı girdisi SQL metnine doğrudan eklenmez. ? yer tutucusu ve whereArgs, sorguyu güvenli ve okunabilir tutar.

Toplam eşya sayısı

```
final toplam = esyalar.fold<int>(
  0,
  (toplam, esya) => toplam + esya.adet,
);
```

SQLite Ev Envanteri

Oda

Mutfak ▼

Envantere ekle

✓ Tümü

Mutfak

Salon

Çalışma

1 kayıt • 6 eşya

6

Saklama kabı
Mutfak

SQL #1

SQLite Ev Envanteri gerçek Flutter ekranı

SQLite ve JSON karşılaştırması

İhtiyaç	JSON	SQLite
Küçük yapılandırma dosyası	Uygun	Gereksiz olabilir
Koşullu sorgu ve sıralama	Kodla yapılır	SQL ile yapılır
Benzersizlik ve veri kısıtı	Elle denetlenir	Şemada tanımlanır
Çok sayıda kayıt	Zorlaşır	Uygundur
Satır bazında güncelleme	Dosya yeniden yazılır	Doğrudan yapılır

Uygulama görevleri

Kolay — Kitap tablosu

id, ad, yazar ve sayfa sütunlarıyla tablo kurun. Kitap ekleyip ada göre sıralayın.

Orta — Laboratuvar cihazları

Cihaz kodunu benzersiz yapın. Adet için pozitif değer kısıtı koyun ve kategoriye göre sorgulayın.

İleri — Okul kulübü üyeleri

Öğrenci numarası, kulüp, görev ve katılım tarihi saklayın. Kulübe göre filtre, ada göre sıralama ve toplam üye sayısı ekleyin.

Test ve kontrol

- İlk açılışta tablo oluşturulmalı.
- Ekleme işlemi pozitif satır kimliği döndürmeli.
- Aynı benzersiz değer ikinci kez eklenememeli.
- Boş zorunlu alan ve geçersiz adet reddedilmeli.
- WHERE sorgusu yalnız istenen kayıtları döndürmeli.
- Kayıtlar belirtilen ORDER BY sırasıyla gelmeli.

Geri bildirim ve çözüm erişimi

Öğrenci tablo şemasını ve sorgularını gönderdikten sonra İpucunu göster sütun türleri ve SQL akışı hakkında yön verir. Örnek çözümü göster gerçek depo ve arayüz kodunu; indirme bağlantısı ise SQLite bağımlılıklarıyla eksiksiz Flutter projesini sunar.

Kaynak projeler

- Konu uygulaması: 06 - Yerel Veri Saklama/Uygulamalar/Uygulama-23-Öğrenci Kayıtları
- Özgün uygulama: 06 - Yerel Veri Saklama/Uygulamalar/Ozgun-23-Ev Envanteri