KÜTÜPHANE FLUTTER UYGULAMASI

Projenin mobil ayağında, bir kütüphanenin, kütüphane üyesi kullanıcılar tarafından kullanılacak flutter uygulaması geliştirilmiştir.

Projenin web ayağıyla birlikte ortak bir veri tabanı kullanılmaktadır. Uygulama, veri tabanı ile API aracılığı ile iletişim kurulmaktadır. API'den gelen veri, uygun modele aktarılarak uygulamada kullanılır.

```
Future searchBooks(String searchPhrase) async {
  List<BookModel> books = [];
  var response = await http.get(
    Uri.parse('http://10.0.2.2:44383/api/Book?searchPhrase=$searchPhrase'),
  );
  if (response.statusCode == 200) {
    var jsonresponse = json.decode(response.body);

    for (var b in jsonresponse) {
      BookModel book = BookModel.fromJson(b);
      books.add(book);
    }
  }
  setState(() {
    query = searchPhrase;
    this.books = books;
  });
}
```

Şekil 1.1. searchBooks fonksiyonu

Kodların tanıtmak amacıyla Search Page sayfasının nasıl çalıştığına bakacak olursak, initState'te, Şekil 1.1'deki searcBook fonksiyonu çağırılmıştır. SearchBook fonksiyonu parametre olarak searcPhrase alıyor. İlk olarak API'ye searchPhrase paremetresi ile get isteğinde bulunuyor eğer istek başarılı olursa json dosyasındaki verileri BookModel (şekil 1.5) tipindeki books global listesine atarak depoluyor.

Şekil 1.3'de görüldüğü üzere Build() => Scaffold'un altında Column'da _SearchBox() prosedürü ve listView.builder() widget'ları yer almaktadır. _SearchBox() prosedürü (Şekil 1.2) TextField submit edildiğinde searchBooks fonksiyonunu çağırarak books değişkenini güncellemektedir.

```
Widget _SearchBox() => Container(
      margin: const EdgeInsets.fromLTRB(16, 86, 16, 10),
      child: TextField(
        controller: controller,
        decoration: InputDecoration(
          prefixIcon: const Icon(Icons.search),
          hintText: 'Kitap Adı',
          border: OutlineInputBorder(
              borderRadius: BorderRadius.circular(20),
              borderSide: const BorderSide(color: Colors.yellow)
              ),  // OutlineInputBorder
        ),  // InputDecoration
        onSubmitted: searchBooks,
        //onChanged: filterBooks,
      ),  // TextField
    );  // Container
```

Şekil 1.2. _SearchBox() prosedürü

```
Widget build(BuildContext context) => Scaffold(
      body: Container(
        decoration: const BoxDecoration(
          gradient: LinearGradient(
          colors: [Color(0xff2193b0), Color(0xff6dd5ed)],
          begin: Alignment.topCenter,
          end: Alignment.bottomCenter,
        )),  // LinearGradient, BoxDecoration
        child: Column(
          children: <Widget>[
            _SearchBox(),
            const Divider(
              color: Colors.white,
              thickness: 1,
              height: 2,
            ),  // Divider
            Expanded(
              child: ListView.builder(
                padding: const EdgeInsets.all(8),
                itemCount: books.length,
                itemBuilder: (BuildContext context, int index) {
                  final book = books[index];
                  if(book.Durum.contains("silindi")){
                    return SizedBox.shrink();
                  }
                  return buildBook(book);
                },
              ),  // ListView.builder
            )  // Expanded
          ],  // <Widget>[]
        ),  // Column
```

Şekil 1.3. Search Page Scaffold

listView.builder() altında books değişkenindeki kitaplar Şekil 1.4'teki buildBook fonksiyonuna gönderilmiştir. buildBook fonksiyonunda kitaplar ListTile widget'ının içine yazılarak döndürülmüştür.

```
Widget buildBook(BookModel book) => Container(
      decoration: const BoxDecoration(
      border: Border(bottom: BorderSide(width: .5, color: Colors.black26))),  // BoxDecoration
      child: ListTile(
        title: Text(book.Isim,
            style:
                TextStyle(color: Colors.black87, fontWeight: FontWeight.normal)),  // Text
        subtitle: Text(book.Yazar,
            style:
                TextStyle(color: Colors.black54, fontWeight: FontWeight.w400)),  // Text
        trailing: Text("${book.Durum.replaceAll(' ', '').replaceAll('u', 'ü')}de",style: TextStyle(color: Colors.black54),),
      ),  // ListTile
    );  // Container
```

Şekil 1.4. buildBook fonksiyonu

```
part 'book_model.g.dart';

@JsonSerializable()
class BookModel{
  final int Id;
  final String Isim;
  final String Yazar;
  final String Tur;
  final int Sayfa;
  final String Durum;

  BookModel(this.Id, this.Isim, this.Yazar, this.Tur, this.Sayfa, this.Durum);

  factory BookModel.fromJson(Map<String, dynamic> json) => _$BookModelFromJson(json);
  Map<String, dynamic> toJson() => _$BookModelToJson(this);
```

Şekil 1.5. BookModel modeli

Kullanıcı Deneyimi

Uygulama ilk açıldığında kullanıcı girişi sayfasıyla karşılaşıyoruz (Şekil 2.1) Üye ol yazısına tıklandığında üye ol sayfasına gidilebilmektedir (Şekil 2.2).

Giriş yaptıktan sonra ana sayfada (şekil 2.3), Kullanıcının şu anda kütüphaneden ödünç alıp elinde bulunan kitaplar ve bu kitapların son teslim tarihleri görülmektedir.

Popüler sayfasında (şekil 2.4), tüm kullanıcıların oylarının ortalaması sonucunda en yüksek oy alan kitaplar görünmektedir.

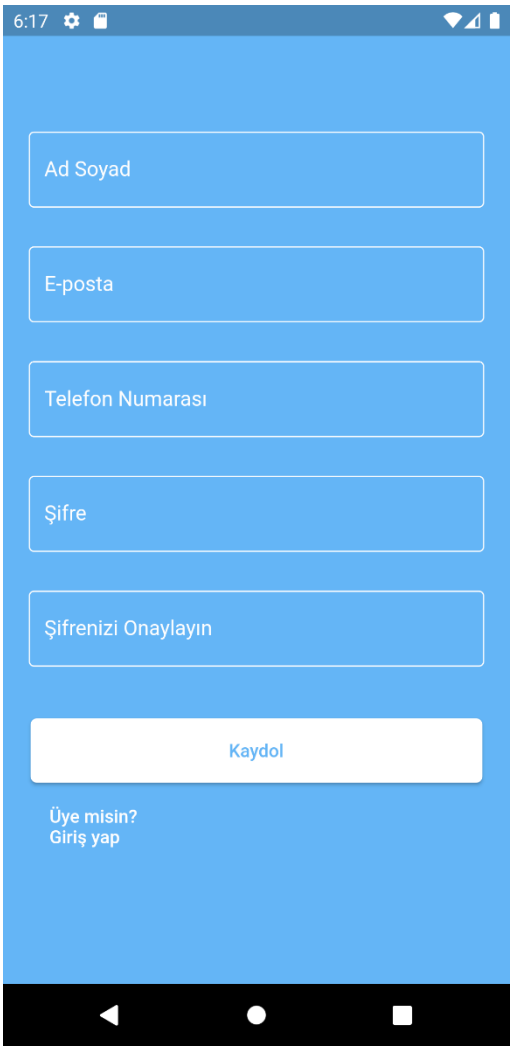
Kitaplarım sayfasında (şekil 2.5), kullanıcının daha önceden okuduğu kitaplar görünmektedir. Kullanıcı bu sayfadan kitapları puanlayabilmektedir.

Arama sayfasında (şekil 2.6), kullanıcı, okumak istediği kitabın kütüphanede olup olmadığını sorgulayabilmektedir. Arama işlemi kitap adı, yazar adı ya da kitap id'si ile yapılabilmektedir.

Sağdaki yan menüden (şekil 2.7), bildirimler (şekil 2.8) ve şifre değiştir (şekil 2.9) sayfalarına gidilebilmektedir.

Bildirimler panelinde kütüphane görevlisinin gönderdiği bildirimleri görebilmekte; şifre değiştirme sayfasında ise mevcut şifresini girerek şifre değiştirebilmektedir.

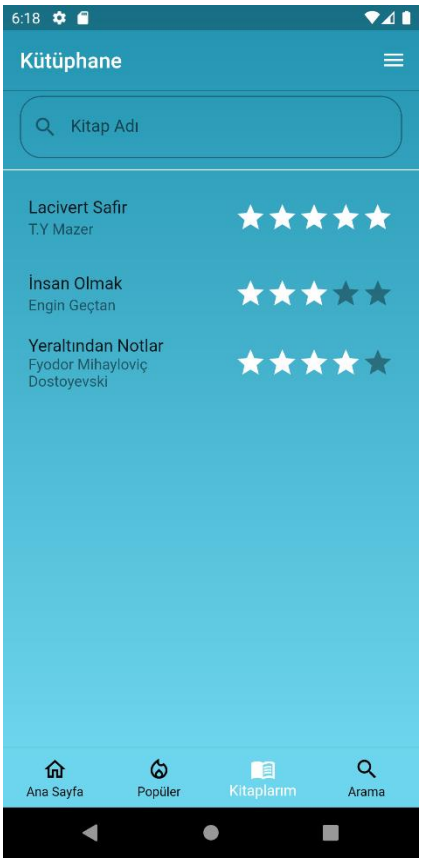
Şekil 2.1. Giriş Yap



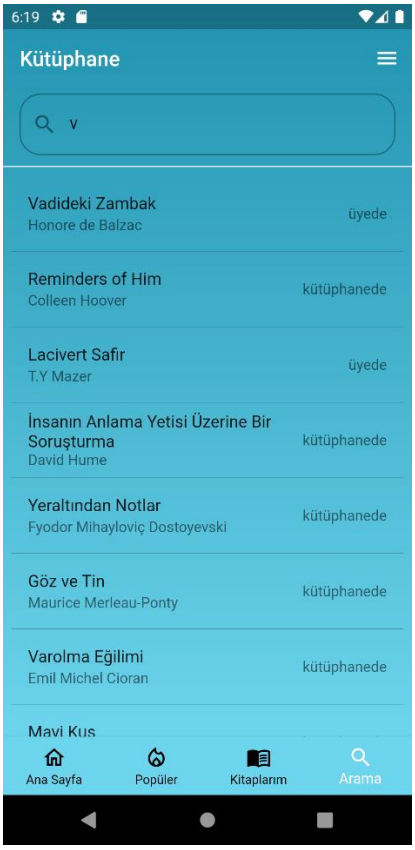
Şekil 2.2. Üye Ol

Şekil 2.3. Ana Sayfa

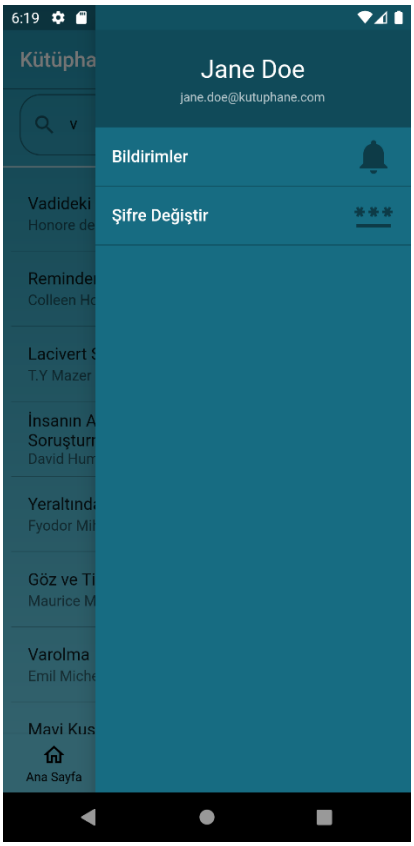
Şekil 2.4. Popüler



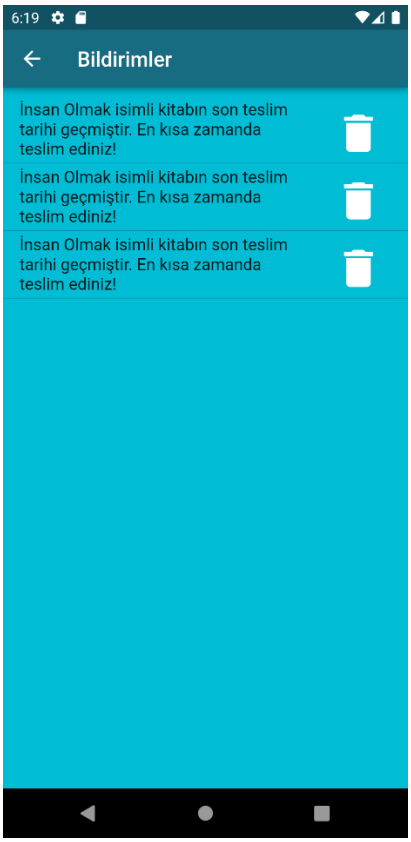
Şekil 2.5. Kitaplarım



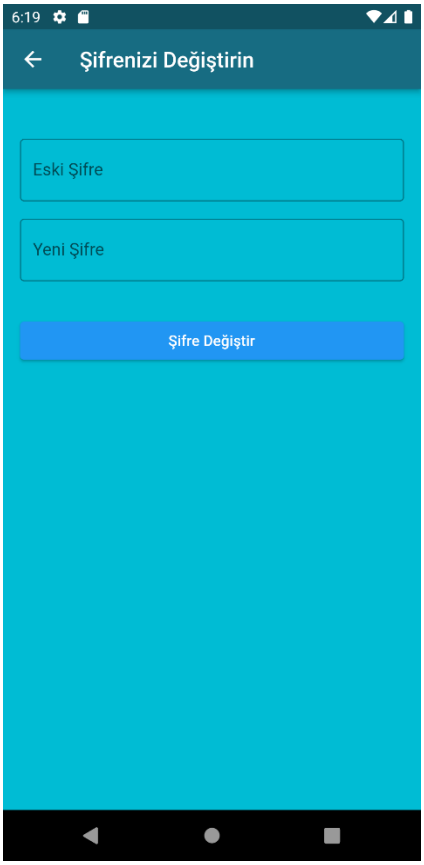
Şekil 2.6. Arama



Şekil 2.7. Yan panel



Şekil 2.8. Bildirimler



Şekil 2.9. Şifre Değiştir

Kurulum

Github'dan indirilen kütüphane_flutter_app dosyası Türkçe karakter içermeyen bir dizine kaydedildikten sonra Android Studio uygulaması ile yüklenen dizindeki dosya seçilerek açılabilir. Çalıştırma işleminin burdan yapılması gerekiyor.

NOT: API bilgisayarın localserver'inde olduğu için ve emulator farklı bir işletim sistemi çalıştırdığı için apk dosyası bilgisayarın localserver'ine bağlanamıyor o yüzden uygulamanın doğrudan android studio ya da visual studio üzerinden başlatılıp emülatörde öyle çalıştırılması gerekiyor.