

FATMA ÜNAL

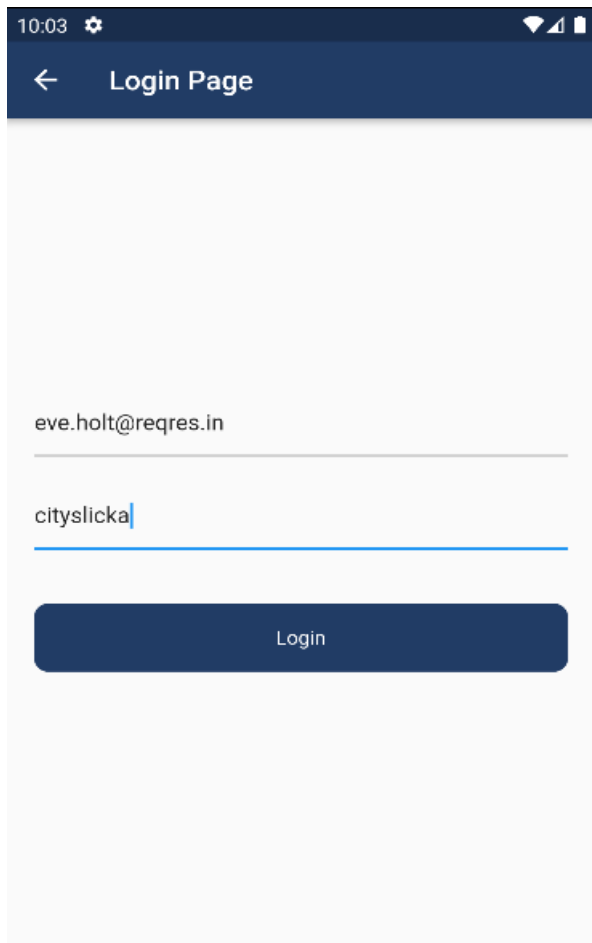
19290276

BLM4537-İOS MOBİL UYGULAMA GELİŞTİRME
FOOD ORDER UYGULAMASI RAPORU

ÖZET

Projemde flutter ile bir yemek sipariş uygulaması geliştirilmiştir.Kullanıcı girişi api ile yapılmıştır.Yemeklerin ismi,renge ve sepettekiler firebase ile alınır.Projenin içinde en çok tercih edilenlerin ve tüm yemeklerin olduğu bir anasayfa vardır.Bu anasayfada bir sepet butonu vardır.Bu sepet butonuna basılınca Sepet ekranına gider. Tüm ürünlerdeki herhangi bir yemeğe tıklayınca yemeğin detayı karşımıza çıkar

1-Login Ekranı

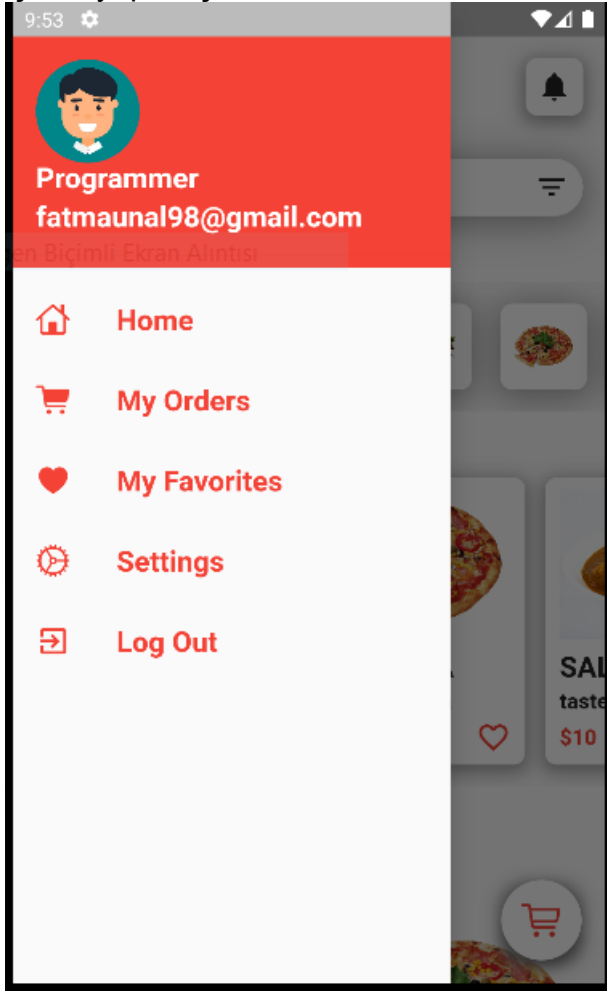


```
class _LoginScreenState extends State<LoginScreen> {}  
  TextEditingController emailController = TextEditingController();  
  TextEditingController passwordController = TextEditingController();  
  
  void login(String email, password) async {  
    try {  
      Response response = await post(Uri.parse('https://reqres.in/api/login'),  
        body: {'email': 'eve.holt@reqres.in', 'password': 'cityslicka'});  
  
      if (response.statusCode == 200) {  
        var data = jsonDecode(response.body.toString());  
        print(data['token']);  
        print('Login successfully');  
        Navigator.push(  
          context, MaterialPageRoute(builder: (context) => (Anasayfa())));  
      } else {  
        print('failed');  
      }  
    } catch (e) {  
      print(e.toString());  
    }  
  }  
}
```

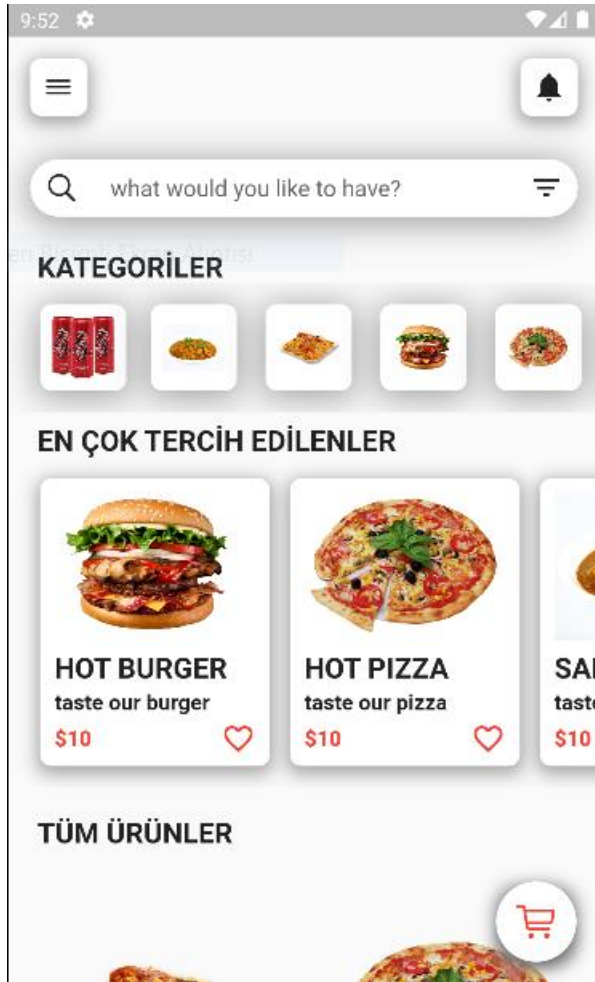
Login sayfası api ile geliştirilmiştir. Json api, ilgili veri kaynaklarının getirilmesi ve işlenmesi için istemci tarafından oluşturulacak isteklerin nasıl yapılması gerektiğini ve sunucunun bu isteklere hangi şekilde yanıt vermesi gerektiğini belirler. İlk olarak http paketi eklenir .Çünkü http package kullanarak json verilere ulaşabilir.Constructor ile bir model yapılandırıp json ile gelen veri bir nesneye çekilir.Daha sonra model map formatına çevirerek json veri map formatında kullanılır. API deki veriyi token kullanarak çekme işlemini gerçekleştirme için response.body işlemi ile json string alınır ve bu **dart:convert** paketini kullanarak conversion işlemi yapılır. statusCode==200 kullanarak istemcinin eylemi başarıyla gerçekleştiyse gelen veri conversion yapılır. Yoksa hata mesajı almasını sağlanır.

2-DRAWER

Anasayfada drawer ile yandan açılan ekran geliştirilmiştir.Burda kişinin mail hesabı bulunur.Home ile anasayfaya gider.Log out ile çıkış yapar.ListTile ile listelenme işlemi yapılmıştır.



3-ANASAYFA









```
child: Padding(  
  padding: EdgeInsets.symmetric(vertical: 15, horizontal: 15),  
  child: Row(children: [  
    Padding(  
      padding: EdgeInsets.symmetric(horizontal: 7),  
      child: Container(  
        padding: EdgeInsets.all(8),  
        width: 60,  
        height: 60,  
        decoration: BoxDecoration(boxShadow: [  
          BoxShadow(  
            color: Colors.grey,  
            spreadRadius: 2,  
            blurRadius: 20,  
            offset: Offset(0, 3)) // BoxShadow  
        ], color: Colors.white, borderRadius: BorderRadius.circular(8)), // BoxDecoration  
        child: Image.asset("images/drink.png"),  
      ), // Container  
    ], // Padding  
  ),
```

Anasayfada konumlandırma ve boyutlandırma widget'larını birleştiren kullanışlı bir widget olan Container ile "Kategoriler" ve "En çok tercih edilenler" kısmı yapılmıştır. Padding özelliğini kullanarak container'ın içindeki nesnelerle arasındaki boşluk ayarlanır. Parametre olarak Edgelnsets kullanılır. Container kutularını şekillendirmek, boyamak ve istenilen türde kutu yaratmak için BoxDecoration kullanılır. Bu projede Container'ın bir özelliği olan BoxShadow ile Container'ın etrafında gri bir gölge oluşturulmuştur. BorderRadius özelliği ile Container'ın kenarına yuvarlaklık verilmiştir. spreadRadius ile gölgenin bulanıklaştırılması ayarlanır. Container'ın içinde yemek resmi, ismi ve fiyatı bulunur.

4-Anasayfa-Tüm Ürünler

Anasayfanın alt kısmında tüm ürünler vardır. Bu ürünler firebase den çekilir. Ürünlerden herhangi birine tıklayınca ürün detayı gelir ve bu kısımda sepet butonu vardır. Bu butona tıklayınca da sepet ekranına gelir.

TÜM ÜRÜNLER

	
Biryani \$15	Pizza \$10
	
Cola \$15	Hamburger \$8
	
Ciaköfte \$10	Tantuni \$4

foodorder-70d8b	yemekler	5G5ncwnQDGAKMx9BESeY
+ Start collection	+ Add document	+ Start collection
cart	2VWeZ5GroCibzV63dM03	+ Add field
yemekler >	5G5ncwnQDGAKMx9BESeY >	id: "5G5ncwnQDGAKMx9BESeY"
	EJ13Zs2UT1mr9eb6inSQ	name: "Pizza"
	IypmBSsJsQ0zjFD4SrJT	picture: "images/pizza.png"
	JLeHgZQnr8RWbMdFvyHI	price: 10
	sEIauS6AWTay0qu2bktb	

Firestore verileri koleksiyonlara, belgelere, alanlara ve alt koleksiyonlara ayrılır.

```
return StreamBuilder(
  stream: FirebaseFirestore.instance.collection('yemekler').snapshots(),
  builder: (BuildContext context, AsyncSnapshot<dynamic> snapshot) {
    if (snapshot.hasError) {
      return Center(
        child: Text(snapshot.error.toString()),
      ); // Center
    }
  }
)
```

Firestore verileri "collection" içindeki "document" içinde depolar. Her bir yemek türünün "yemekler" içindeki kendi document'leri olur. Kodda görüldüğü üzere StreamBuilder ile Yani stream'den gelen veriler hazır olduğunda ve her yeni veri geldiğinde, builder fonksiyonunu tekrar çağırıyor ve tekrar widget ağacını yaratır.

```
if (!snapshot.hasData) {
  return const Center(child: CircularProgressIndicator());
}
List<DocumentSnapshot> tripPhotos;
tripPhotos = snapshot.data.docs;

return GridView.builder(
  shrinkWrap: true,
  itemCount: snapshot.data.docs.length,
  gridDelegate: new SliverGridDelegateWithFixedCrossAxisCount(
    crossAxisCount: 2), // SliverGridDelegateWithFixedCrossAxisCount
  itemBuilder: (BuildContext context, int index) {
    return Padding(
      padding: const EdgeInsets.all(4.0),
      child: Single_prod(
        prod_id: tripPhotos[index]['id'],
        prod_name: tripPhotos[index]['name'],
        prod_picture: tripPhotos[index]['picture'],
        prod_price: tripPhotos[index]['price'],
      ), // Single_prod
    ); // Padding
  }); // GridView.builder
```

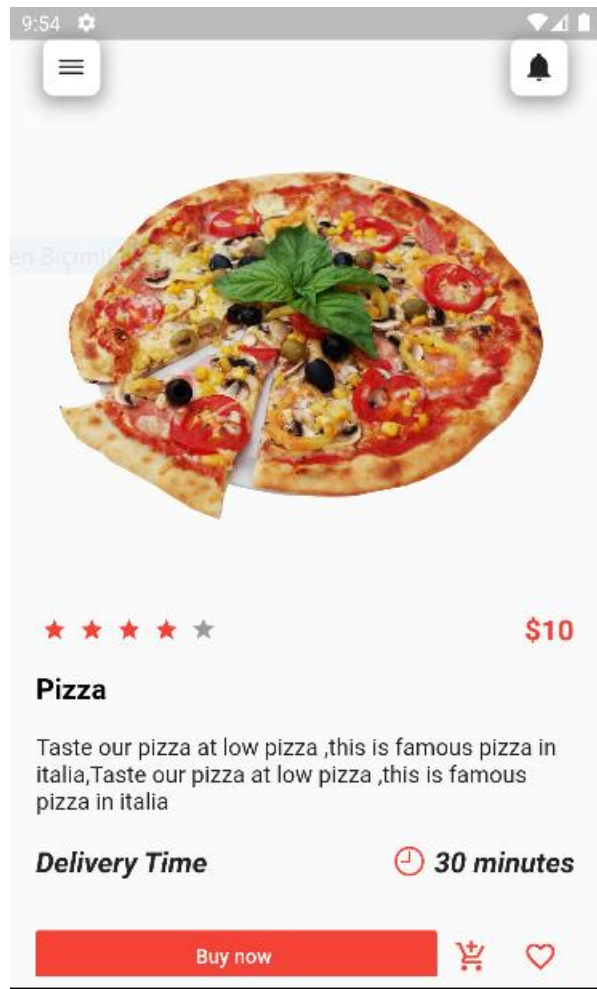

Kodda “yemekler” isimli collection’ı çağırarak bir QuerySnapshot elde edilir. Bu QuerySnapshot’ın içinden get metodu ile bir DocumentSnapshot’ı çeker. Çünkü document’ın ID’sini bilmektedir ve çünkü DocumentSnapshot’lar, QuerySnapshot’lar içerisinde bir liste olarak tutulmaktadır.

Documentsnapshot’ı bir StreamBuilder içerisinde kullanarak snapshot’ını alır.

Bu snapshot ile elde ettiği veriyi map olarak kullanır. Çünkü DocumentSnapshot bir listedir ve bu listenin içerisinde map’ler vardır. Bu map’ler ise anahtar ve değer olarak verilerimizi tutar. Bu map’e erişildiğinde istenilen veriye erişmiş olunur.

Map’in anahtar ve değerlerini Gridview içinde ekrana yazdırır.Bu gridview içinde yemek ismi,yemeğin resmi ve yemeğin fiyatı bulunur.

5-YEMEK DETAYI



Ekranda görüldüğü üzere yemek türünün resmi,ismi,fiyatı ,sipariş saati ve bir sepete ekle butonu bulunur.İsmi,resmi ve fiyatı firebase’den alınır.Butona basılında firebase’e ekler.Firebase’den çekme işlemi “Tüm ürünler” i çekme işlemi gibidir.Fakat butona tıklayınca cart collection’a ekleme işlemi daha farklıdır.

```

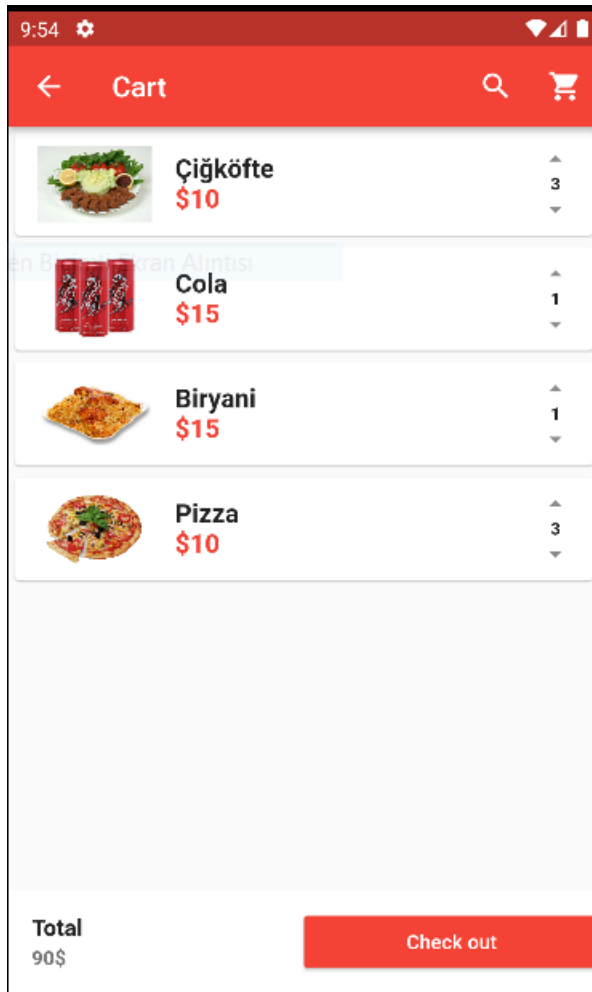
Future<void> sepeteEkle(String id) async {
  try {
    final QuerySnapshot varmiKontrol = await FirebaseFirestore.instance
      .collection('cart')
      .where('prod_id', isEqualTo: id)
      .get();
    if (varmiKontrol.docs.isNotEmpty) {
      if (varmiKontrol.docs[0]['prod_id'] == id) {
        print('bu veri zaten var!');
        await FirebaseFirestore.instance
          .collection('cart')
          .doc(varmiKontrol.docs[0]['id'])
          .update({'quantity': FieldValue.increment(1)});
      }
    } else {
      print(id);
      var key =
        FirebaseFirestore.instance.collection('cart').doc().id.toString();
      print(key);

      await FirebaseFirestore.instance.collection('cart').doc(key).set({
        'id': key,
        'prod_id': widget.product_id,
        'name': widget.product_detail_name,
        'picture': widget.product_detail_picture,
        'price': widget.product_detail_price,
        'quantity': 1,
      });
    }
  }
}

```

Bir Collection'a Document eklemek için CollectionReference'ın add metodu kullanılır; fakat kendi id'sini oluşturmak istediğimiz için DocumentReference'a ait olan set metodu kullanılır. Bu metod ile seçilen yemek türü firebase'e eklenir. Querysnapshot ile "cart" collection'unda id'nin olup olmadığını kontrol eder. Varsa quantity değerini bir artırır; yoksa o yemek türünü "cart" collection'a ekler.

6-CART



Yemek detayından butona basılınca firebase e veri gönderilir.Gönderilen veriyi cart page,firebase'den çeker.

```
total = 0;

tripPhotos.forEach((result) {
  total += result['price'] * result['quantity'];
});

return Text(
  '${total.toString()}\$',
  style: TextStyle(fontWeight: FontWeight.bold),
); // Text
},
```

Yapılan hesaplamada total hesabı; seçilen ürün detayının fiyatı ile sayısı çarpılır ve hepsi toplanır.Bu total text widget'ına gönderilir.