

**Çizge Renklendirme ile Ders Programı   
 Hazırlama Projesi**

**Yazılım Geliştirme Laboratuvarı-I**

**Yavuz Selim FATİHOĞLU**

**Süleyman EKEN**

**Hazırlayanlar**

Taha Rıdvan ÖZTÜRK– 221307097

Furkan DOKUZOĞLU-221307080

Emrullah DONSAK – 22130710**8**

**İçindekiler**

[Özet: 3](#_Toc155546884)

[1. GİRİŞ 3](#_Toc155546885)

[1.1.Çizge Teorisi Nedir: 3](#_Toc155546886)

[2.UYGULAMA 4](#_Toc155546887)

[2.1.Tanıtım 4](#_Toc155546888)

[2.2.Örnek Bir Ders Programı Hazırlama: 8](#_Toc155546889)

[3.GELİŞTİRME 9](#_Toc155546890)

[3.1.getTeacherNames() Fonksiyonu: 9](#_Toc155546891)

[3.2.\_getHoursFromGridCollection() Fonksiyonu: 9](#_Toc155546892)

[3.3.getClasses() Fonksiyonu: 10](#_Toc155546893)

[3.4. main() Fonksiyonu: 10](#_Toc155546894)

[3.5. MyApp Sınıfı: 10](#_Toc155546895)

[3.6. MyHomePage Sınıfı: 11](#_Toc155546896)

[3.7. \_MyHomePageState Sınıfı: 11](#_Toc155546897)

[3.8.DersEklemeSayfasi Sınıfı: 12](#_Toc155546898)

[3.9.\_DersEklemeSayfasiState Sınıfı: 13](#_Toc155546899)

[3.10.OgretmenEklemeSayfasi Sınıfı: 13](#_Toc155546900)

[3.11.OgretmenEklemeSayfasiState Sınıfı: 14](#_Toc155546901)

[3.12. ClassSchedulePage Sınıfı 15](#_Toc155546902)

[3.13. ManageDataPage Sınıfı: 15](#_Toc155546903)

[3.14. \_ManageDataPageState Sınıfı: 15](#_Toc155546904)

[3.15. KameraEkrani Sınıfı: 16](#_Toc155546905)

#### ÇİZGE RENKLENDİRME İLE DERS PROGRAMI HAZIRLAMA PROJESİ

### Özet:

Bu proje, çeşitli isterler dikkate alınarak çizge renklendirme yöntemleriyle bir ders programı hazırlama uygulaması sürecini kapsamaktadır.

## 1. GİRİŞ

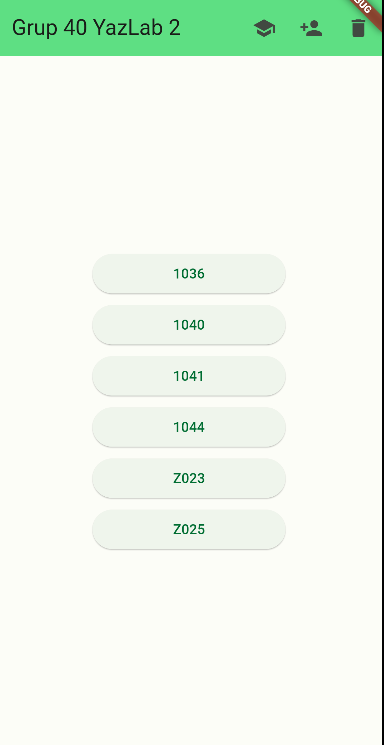
### 1.1.Çizge Teorisi Nedir:

Çizge Teorisi çok farklı disiplinlerin çalışma alanına girmektedir. Sosyolojiden, bilgisayar bilimlerine, işletmeden, endüstri mühendisliğine kadar çok geniş alanlarda kullanımı olan teori, basitçe bir gerçek hayat probleminin çizge ile modellenmesini amaçlamaktadır. Model oluşturulduktan sonra çizge teorisinde bulunan yöntemler kullanılarak problem çözülebilmekte ve ardından da tekrar gerçek hayata uygulanabilmektedir.

## 2.UYGULAMA

### 2.1.Tanıtım

Uygulamayı ilk olarak açtığımızda karşımıza fakültemizdeki Bilişim Sistemleri Mühendisliği bölümü derslikleri çıkmaktadır.

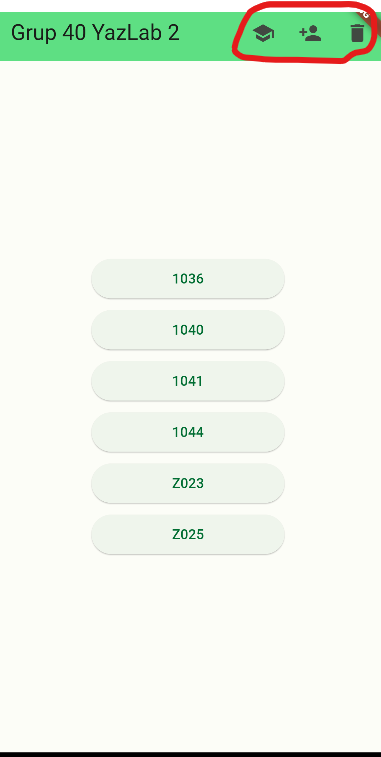


Uygulamamızda gezinirken rastgele herhangi bir dersliğe tıklayınca karşımıza o derslikteki haftalık ders programı çıkmaktadır.

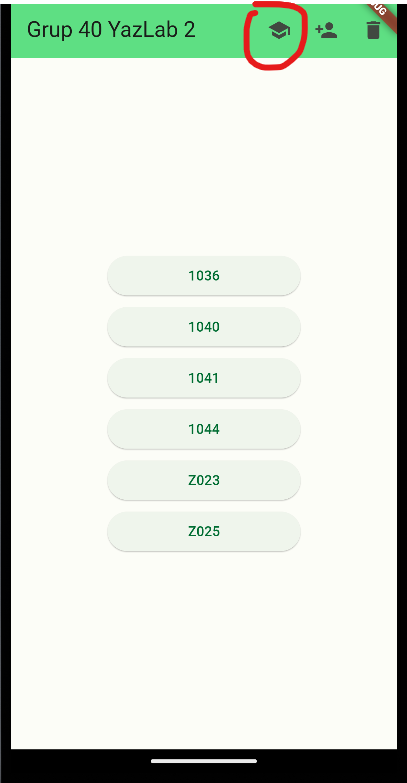
****

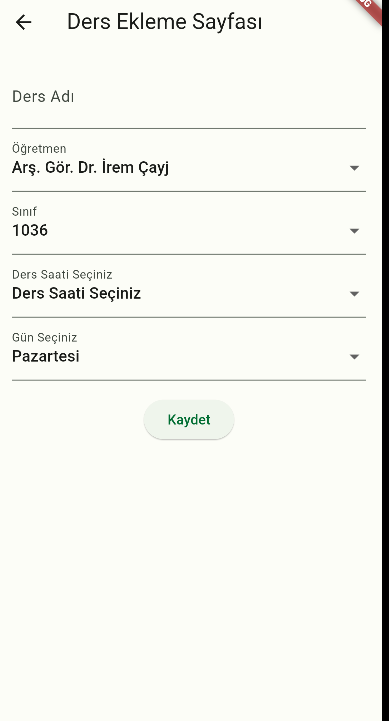


Uygulamamızın arayüzünün üst kısmında üç seçenek olduğu gözükmektedir.

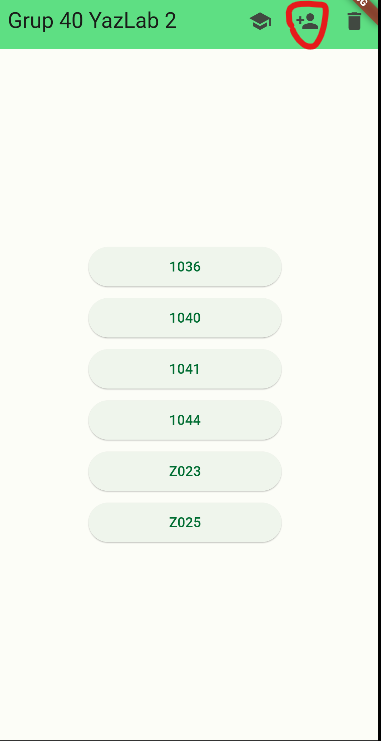


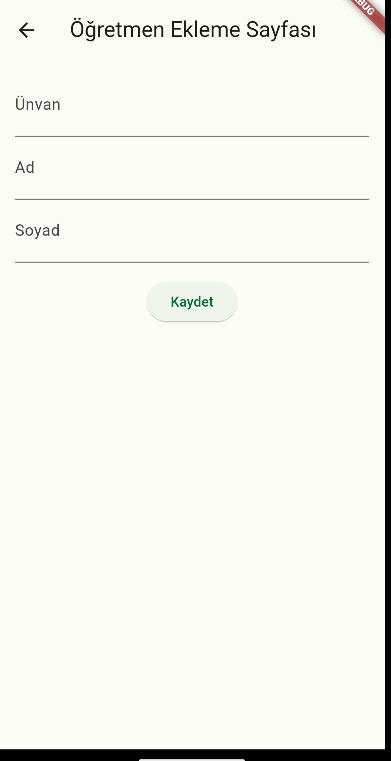
Birinci seçenekteki görsele tıkladığımızda karşımıza ders ekleme sayfası gelmektedir. Burada istediğimiz bir dersi istediğimiz bir öğretim elemanını seçerek haftalık ders programına ekleyebiliriz.



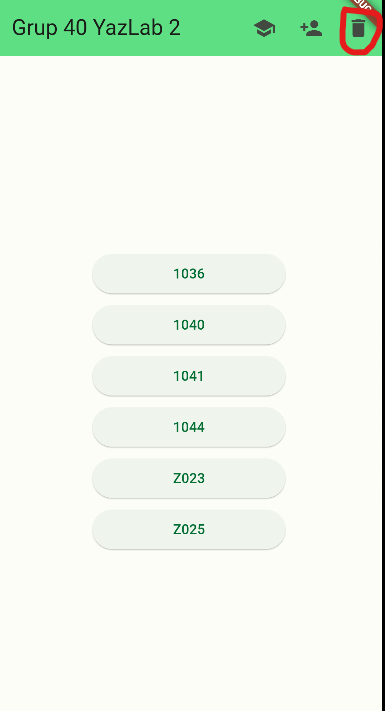


Uygulama arayüzündeki ikinci seçenekteki görsele tıkladığımızda ise karşımıza öğretim görevlisi ekleme sayfası gelmektedir. Burada öğretim görevlisinin unvanı adı ve soyadı girilerek ekleme işlemi yapılır.

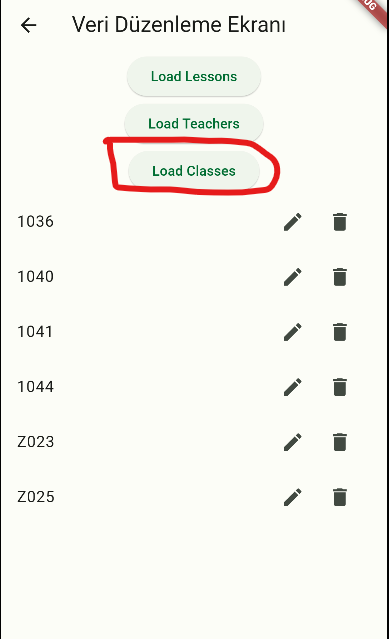


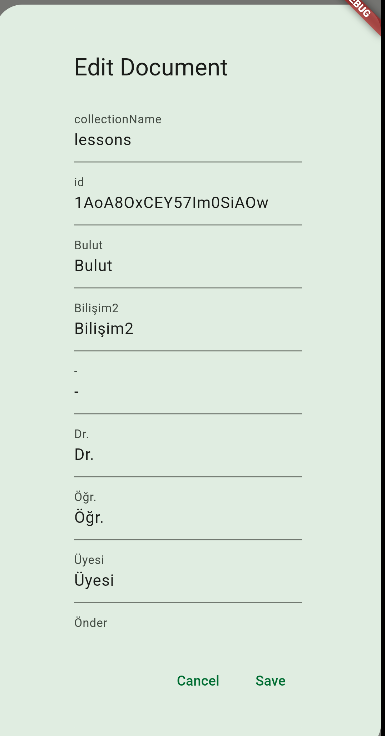


Son olarak son seçenekteki görsele tıkladığımızda ise karşımıza bir veri düzenleme ekranı gelmektedir. Burada uygulamadaki tüm derslikleri, öğretim üyelerini ve dersleri görebilir ve düzenlemeler yapabiliriz.



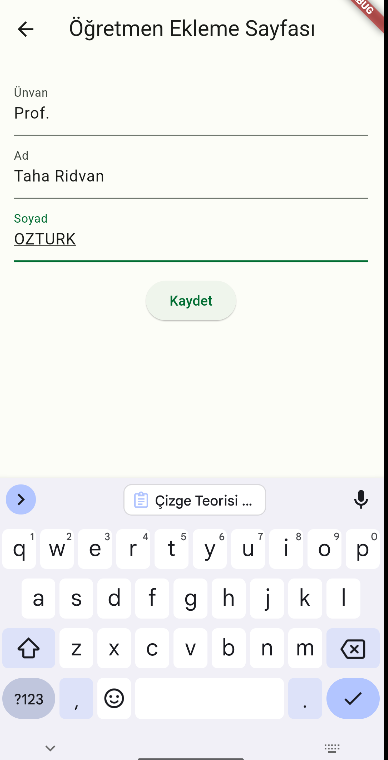




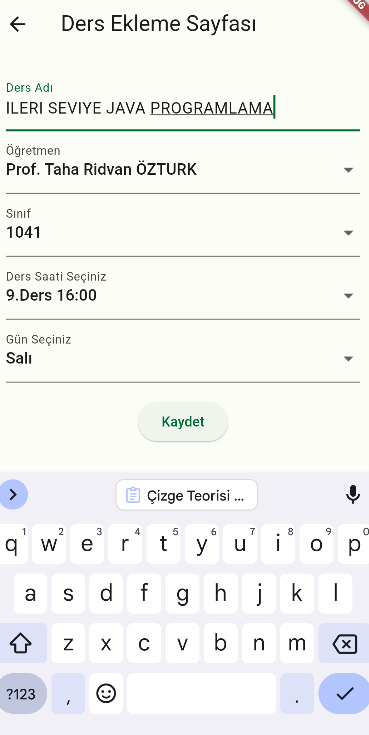


### 2.2.Örnek Bir Ders Programı Hazırlama:

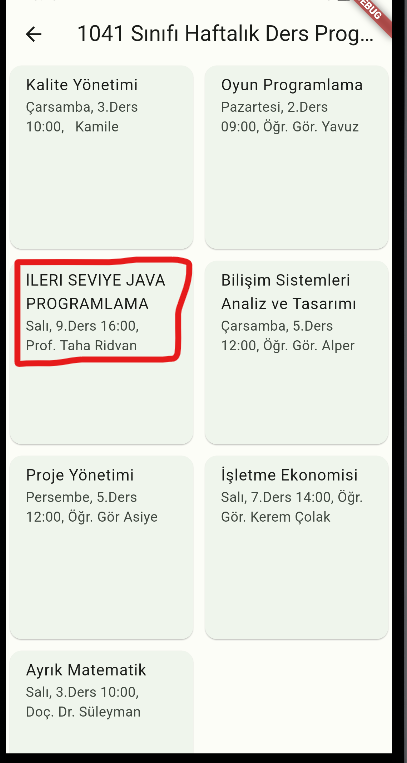
İlk olarak bir öğretim görevlisi eklenir.

****

Daha sonra yeni bir ders eklenip öğretim görevlisi seçilir.



Görüldüğü gibi ders ve öğretim görevlisi ders programına eklenmiş oldu.



## 3.GELİŞTİRME

### 3.1.getTeacherNames() Fonksiyonu:

Bu Dart fonksiyonu, Firebase Firestore veritabanındaki 'teachers' koleksiyonundan öğretmenlerin bilgilerini almak için kullanılabilir. Fonksiyon, bir Future<List<String>> döndürür, yani asenkron olarak çalışır ve bir dize listesi içinde öğretmen isimlerini içerir**.**

Bu fonksiyonun adı “\_getTeacherNames”, ve “Future<List<String>>” türünde bir değer döndürüyor. Fonksiyon içerisinde await anahtar kelimesi kullanılarak asenkron bir işlem gerçekleştiriliyor.

“Firestore.instance.collection('teachers').get():” Firestore'daki 'teachers' koleksiyonundan tüm dokümanları alır. Bu işlem, asenkron olduğu için “await” kullanılarak beklenir.

“.map((doc) => '${doc['unvan']} ${doc['ad']} ${doc['soyad']}'):” Alınan dokümanlardan her birini “map” ederek, doküman içindeki 'unvan', 'ad' ve 'soyad' alanlarını kullanarak bir isim string'i oluşturur.

“.toList():” Oluşturulan isim ‘string’'lerini içeren bir liste elde edilir.

Son olarak, oluşturulan dize listesini döndürür.

### 3.2.\_getHoursFromGridCollection() Fonksiyonu:

Bu Dart fonksiyonu, Firebase Firestore'daki 'grid' koleksiyonundan saat bilgilerini almak için kullanılabilir. Fonksiyon, bir “Future<List<Map<String, String>>>” döndürür, yani asenkron olarak çalışır ve her bir saat için bir harita içinde 'id' ve 'text' değerlerini içerir.

Bu fonksiyonun adı “\_getHoursFromGridCollection”, ve “Future<List<Map<String, String>>>” türünde bir değer döndürüyor. Fonksiyon içerisinde ‘await’ anahtar kelimesi kullanılarak asenkron bir işlem gerçekleştiriliyor.

“Firestore.instance.collection('grid').get():” Firestore'daki 'grid' koleksiyonundan tüm dokümanları alır. Bu işlem, asenkron olduğu için ‘await’ kullanılarak beklenir.

“.map((doc) => {'id': doc['id'].toString(), 'text': doc['text'].toString(),}):” Alınan dokümanlardan her birini ‘map’ ederek, 'id' ve 'text' değerlerini içeren bir harita oluşturur.

“.toList():” Oluşturulan harita listesini elde eder.

Son olarak, oluşturulan harita listesini döndürür.

### 3.3.getClasses() Fonksiyonu:

Bu Dart fonksiyonu, Firebase Firestore'daki 'classes' koleksiyonundan sınıf isimlerini almak için kullanılabilir. Fonksiyon, bir “Future<List<String>>” döndürür, yani asenkron olarak çalışır ve her bir sınıf için bir dize içerir.

Bu fonksiyonun adı “\_getClasses”, ve “Future<List<String>>” türünde bir değer döndürüyor. Fonksiyon içerisinde ‘await’ anahtar kelimesi kullanılarak asenkron bir işlem gerçekleştiriliyor.

“Firestore.instance.collection('classes').get():” Firestore'daki 'classes' koleksiyonundan tüm dokümanları alır. Bu işlem, asenkron olduğu için ‘await’ kullanılarak beklenir.

“.map((doc) => doc['name'] as String):” Alınan dokümanlardan her birinden 'name' alanını alarak bir dize listesi oluşturur.

“.toList():” Oluşturulan dize listesini elde eder.

Son olarak, oluşturulan dize listesini döndürür.

### 3.4. main() Fonksiyonu:

Bu Dart kodu, Flutter uygulamasının başlangıcını temsil eder. Aşağıda, kodun adım adım açıklaması bulunmaktadır:

“main” fonksiyonu: Flutter uygulamasının giriş noktasıdır. Bu fonksiyon, uygulamanın başlatılmasını ve çalıştırılmasını sağlar.

“WidgetsFlutterBinding.ensureInitialized():” Flutter'ın ‘widget’ sisteminin başlatıldığını ve hazır olduğunu belirtir. Bu, Flutter uygulamasının ‘widget’'larını başlatmadan önce gerekli ön koşulların karşılandığından emin olur.

“await Firebase.initializeApp(options: DefaultFirebaseOptions.currentPlatform):” Firebase servislerini başlatmak için kullanılır. Bu, Firebase'in başlatılmasını bekler ve uygulama başlatılmadan önce Firebase'in kullanıma hazır olmasını sağlar. Bu noktada, “DefaultFirebaseOptions.currentPlatform” seçenekleri ile varsayılan Firebase ayarları kullanılır.

“runApp(const MyApp()):” Flutter uygulamasının ana ‘widget’'ını başlatır. Bu örnekte, “MyApp” isimli bir ‘widget’ kullanılır. Bu, uygulamanın genel görünüm ve davranışını belirler.

Bu kod parçası, Flutter uygulamasının Firebase'i başlatmasını bekler ve ardından “MyApp” widget'ını başlatarak uygulamayı çalıştırır. ‘await’ anahtar kelimesi kullanılarak asenkron işlemlerin tamamlanmasını bekler, bu nedenle “main” fonksiyonu da ‘async’ olarak işaretlenir.

### 3.5. MyApp Sınıfı:

Bu Dart kodu, bir Flutter uygulamasının temel ‘widget’'ını temsil eden “MyApp” sınıfını içerir. Aşağıda, kodun adım adım açıklaması bulunmaktadır:

“MyApp” sınıfı: “StatelessWidget” sınıfını genişletir. “Stateless Widget”'lar, bir kez oluşturulduktan sonra değiştirilemeyen (‘state’ içermeyen) ‘widget’'lardır.

“const MyApp({Key? key}) : super(key: key);:” “MyApp” sınıfının kurucu metodudur. Burada, üst sınıfın (StatelessWidget) kurucu metoduna ‘key’ parametresini ileterek ‘widget’'ı tanımlar. ‘const’ anahtar kelimesi, bu sınıfın sabit bir değere sahip olduğunu belirtir.

“Widget build(BuildContext context):” “Stateless Widget”'ların oluşturulduğu metottur. “BuildContext” nesnesi, ‘widget’'ın konumunu ve diğer bilgileri içerir.

“MaterialApp:” Flutter uygulamalarının temel ‘widget’'ıdır. Uygulamanın genel temel özelliklerini belirtir. Örneğin, uygulama başlığı, tema ve ana sayfa gibi özellikler burada tanımlanır.

“title:” 'Grup 40 YazLab 2': Uygulamanın başlığını belirtir.

“theme: ThemeData(...):” Uygulamanın temasını belirtir. “colorScheme” özelliği, uygulamanın renk şemasını tanımlar. Bu örnekte, özel bir renk şeması kullanılarak oluşturulan bir “ColorScheme” kullanılır.

“ColorScheme.fromSeed(seedColor: const Color.fromARGB(255, 8, 163, 79)):” Renk şemasını oluşturan ve başlatan bir yöntemdir. Bu örnekte, belirli bir renk değeri kullanılarak özel bir renk şeması oluşturulur.

“useMaterial3: true,: useMaterial3” özelliği, “Material 3” tasarım dilini kullanıp kullanmama durumunu belirtir. Bu örnekte, ‘true’ değeri ile “Material 3” kullanılması belirtilmiştir.

“home: const MyHomePage(title: 'Grup 40 YazLab 2'),:” Uygulamanın ana sayfasını belirtir. “MyHomePage” ‘widget’'ı, uygulamanın içeriğini oluşturacak olan ‘widget’'tır. “title” parametresi, “MyHomePage” ‘widget’'ına başlık bilgisini iletilir.

Bu kod parçası, bir temel Flutter uygulamasını başlatan ve temel tasarım özelliklerini belirleyen “MyApp” sınıfını içerir.

### 3.6. MyHomePage Sınıfı:

Bu Dart kodu, bir `MyHomePage` adlı StatefulWidget'ın tanımını içerir. Aşağıda, kodun adım adım açıklaması bulunmaktadır:

1. `MyHomePage` sınıfı: `StatefulWidget` sınıfını genişletir. StatefulWidget'lar, içerdikleri durum nesnesi (state) ile widget'larını güncelleyebilen widget'lardır.

2. `const MyHomePage({Key? key, required this.title}) : super(key: key);`: `MyHomePage` sınıfının kurucu metodudur. Burada, üst sınıfın (StatefulWidget) kurucu metoduna `key` parametresini ileterek widget'ı tanımlar. Ayrıca, `title` adlı bir zorunlu parametre eklenir. Bu parametre, sayfanın başlığını içerir.

3. `final String title;`: `MyHomePage` sınıfının bir özelliğidir. Bu özellik, sayfanın başlığını içerir ve sınıfın kurucu metodunda zorunlu olarak alınır.

4. `@override State<MyHomePage> createState() => \_MyHomePageState();`: `createState` metodu, `MyHomePage` widget'ının durum nesnesini oluşturan metottur. `\_MyHomePageState` adlı özel bir durum sınıfını oluşturur ve geri döndürür. State nesnesi, widget'ın durumuyla ilgili verileri içerir ve widget'ın güncellenmesini sağlar.

Bu kod parçası, bir sayfa başlığı içeren ve bir durum nesnesi içeren `MyHomePage` StatefulWidget'ını tanımlar. `\_MyHomePageState` sınıfı, bu widget'ın durumuyla ilgili işlevselliği sağlar. Bu tasarım, sayfa içeriğindeki durumun güncellenmesi gereken durumlarda kullanılır.

### 3.7. \_MyHomePageState Sınıfı:

Bu Dart kodu, `\_MyHomePageState` adlı bir `State` sınıfını içerir. Bu sınıf, `MyHomePage` widget'ının durumunu yönetir ve `build` metoduyla sayfanın içeriğini oluşturur. Aşağıda, kodun adım adım açıklaması bulunmaktadır:

1. `\_MyHomePageState` sınıfı: `State` sınıfını genişletir. Bu sınıf, `MyHomePage` widget'ının durumunu yönetir.

2. `Widget build(BuildContext context)`: Sayfanın içeriğini oluşturan metottur. Bu metot, kullanıcının sayfaya her girdiğinde çağrılır ve sayfa içeriğini oluşturur.

3. `Scaffold`: Temel materyal tasarım öğelerini içeren bir yapıdır. Uygulamanın ana çerçevesini sağlar.

4. `AppBar`: Sayfanın üst kısmındaki çubuğu temsil eder. `backgroundColor` özelliği, tema renk şemasından tersine çevrilen birincil rengi kullanır.

5. `title: Text(widget.title)`: Sayfa başlığını gösteren bir `Text` widget'ıdır. Başlık, `MyHomePage` widget'ının `title` özelliğinden alınır.

6. `actions`: AppBar'ın sağ tarafında bulunan eylemleri temsil eder. Burada, üç tane `IconButton` bulunur.

- `IconButton` 1: Ders ekleme sayfasına yönlendiren bir ikon düğmesi.

- `IconButton` 2: Öğretmen ekleme sayfasına yönlendiren bir ikon düğmesi.

- `IconButton` 3: Veri yönetim sayfasına yönlendiren bir ikon düğmesi.

7. `body`: Sayfanın ana içeriğini temsil eder. Burada, `FutureBuilder` kullanılarak gelecekteki bir değeri bekleyen bir widget oluşturulur. `\_getClasses` fonksiyonu, sınıf isimlerini getiren bir asenkron işlemdir.

8. `FutureBuilder<List<String>>`: Gelecekte bir `List<String>` değeri döndüren bir işlemin sonucunu bekleyen bir widget'tır. Bu durumda, `\_getClasses` fonksiyonundan gelecek olan sınıf isimlerini bekler.

9. `snapshot.connectionState == ConnectionState.waiting`: Veri beklenirken ekrana bir döner yükleyici gösterir.

10. `snapshot.hasError`: Eğer bir hata oluşursa, hatayı ekrana yazar.

11. `snapshot.hasData && snapshot.data != null`: Eğer veri başarıyla alındıysa, sınıf isimlerini içeren bir liste oluşturur ve bu listeyi kullanarak bir dizi `ElevatedButton` oluşturur. Her düğme, sınıf ismini temsil eder ve tıklanabilir.

12. `return Center(...`: Eğer veri başarıyla alındıysa, ekranda sınıf isimlerini içeren düğmeleri gösterir. Her düğme tıklandığında, ilgili sınıfın ders programını gösteren bir sayfaya yönlendirme yapılır.

13. `else`: Eğer sınıf bilgileri bulunamazsa veya kayıtlı sınıf yoksa ekrana bir hata mesajı yazdırır.

Bu kod parçası, Flutter uygulamasının ana sayfasını ve sınıf isimlerini listeleyen bir sayfayı temsil eder. Sayfa içeriği, asenkron bir işlemle sınıf isimlerini alır ve bu bilgileri kullanarak bir arayüz oluşturur.

### 3.8.DersEklemeSayfasi Sınıfı:

Bu Dart kodu, `DersEklemeSayfasi` adlı StatefulWidget'ın tanımını içerir. Aşağıda, kodun adım adım açıklaması bulunmaktadır:

1. `DersEklemeSayfasi` sınıfı: `StatefulWidget` sınıfını genişletir. StatefulWidget'lar, içerdikleri durum nesnesi (state) ile widget'larını güncelleyebilen widget'lardır.

2. `const DersEklemeSayfasi({Key? key}) : super(key: key);`: `DersEklemeSayfasi` sınıfının kurucu metodudur. Burada, üst sınıfın (StatefulWidget) kurucu metoduna `key` parametresini ileterek widget'ı tanımlar.

3. `@override \_DersEklemeSayfasiState createState() => \_DersEklemeSayfasiState();`: `createState` metodu, `DersEklemeSayfasi` widget'ının durum nesnesini oluşturan metottur. `\_DersEklemeSayfasiState` adlı özel bir durum sınıfını oluşturur ve geri döndürür. State nesnesi, widget'ın durumuyla ilgili verileri içerir ve widget'ın güncellenmesini sağlar.

Bu kod parçası, `DersEklemeSayfasi` adlı bir sayfa oluşturur ve bu sayfanın durumunu yöneten özel bir durum sınıfını içerir. Bu tasarım, sayfanın içeriğindeki durumun güncellenmesi gereken durumlarda kullanılır.

### 3.9.\_DersEklemeSayfasiState Sınıfı:

Bu Dart kodu, `DersEklemeSayfasi` adlı bir sayfanın durumunu yöneten `\_DersEklemeSayfasiState` sınıfını içerir. Bu sınıf, bir ders eklemek için gerekli olan formu ve bu formun işlevselliğini sağlayan metotları içerir. Aşağıda, kodun adım adım açıklaması bulunmaktadır:

1. `final GlobalKey<FormState> \_formKey`: Formun anahtarını tanımlar. Bu anahtar, formun durumunu kontrol etmek ve doğrulamak için kullanılır.

2. `final TextEditingController \_dersAdiController`: Ders adını almak için bir `TextEditingController` tanımlar.

3. `final TextEditingController \_saatController`: Ders saati almak için bir `TextEditingController` tanımlar.

4. `String? \_selectedDay`: Seçilen günü tutan bir değişken.

5. `String \_selectedClass = 'Sınıf Seçiniz'`: Seçilen sınıfı tutan bir değişken.

6. `\_saveLessonData(BuildContext context)`: Yeni bir dersin kaydedilmesini sağlayan metottur. Bu metodun içerisinde, gerekli bilgileri alır, çakışan ders olup olmadığını kontrol eder ve ardından Firebase Firestore'a yeni dersi ekler.

7. `initState()`: State sınıfının initState metodu, sayfa oluşturulduğunda bir kere çağrılır ve özellikle başlangıç durumunu ayarlamak için kullanılır. Bu örnekte `\_loadTeacherNames` metodu çağrılarak öğretmen isimleri yüklenir.

8. `\_loadTeacherNames()`: Firebase Firestore'dan öğretmen isimlerini alarak, dropdown menu için varsayılan öğretmeni seçer.

9. `build(BuildContext context)`: Sayfanın içeriğini oluşturan metottur. Bu metot, kullanıcı formu doldurduktan sonra kaydet butonuna bastığında `\_saveLessonData` metodu çağrılır. Form elemanları, `TextFormField`, `DropdownButtonFormField`, ve `ElevatedButton` gibi Flutter widget'larını içerir.

Bu kod parçası, bir dersin eklenmesi için gerekli olan sayfanın durumunu yönetir ve kullanıcıdan gerekli bilgileri alır. `FirebaseFirestore.instance` kullanılarak Firebase Firestore'a veri eklenir ve olası çakışmalar kontrol edilir. Sonuç olarak, kullanıcıya bir Snackbar mesajı ile işlemin başarılı olup olmadığı bildirilir.

### 3.10.OgretmenEklemeSayfasi Sınıfı:

Bu Dart kodu, `OgretmenEklemeSayfasi` adlı StatefulWidget'ın tanımını içerir. Aşağıda, kodun adım adım açıklaması bulunmaktadır:

1. `OgretmenEklemeSayfasi` sınıfı: `StatefulWidget` sınıfını genişletir. StatefulWidget'lar, içerdikleri durum nesnesi (state) ile widget'larını güncelleyebilen widget'lardır.

2. `const OgretmenEklemeSayfasi({Key? key}) : super(key: key);`: `OgretmenEklemeSayfasi` sınıfının kurucu metodudur. Burada, üst sınıfın (StatefulWidget) kurucu metoduna `key` parametresini ileterek widget'ı tanımlar.

3. `@override OgretmenEklemeSayfasiState createState() => OgretmenEklemeSayfasiState();`: `createState` metodu, `OgretmenEklemeSayfasi` widget'ının durum nesnesini oluşturan metottur. `OgretmenEklemeSayfasiState` adlı özel bir durum sınıfını oluşturur ve geri döndürür. State nesnesi, widget'ın durumuyla ilgili verileri içerir ve widget'ın güncellenmesini sağlar.

Bu kod parçası, `OgretmenEklemeSayfasi` adlı bir sayfanın durumunu yöneten `OgretmenEklemeSayfasiState` sınıfını içerecek şekilde tasarlanmış bir StatefulWidget'ı temsil eder. Bu sınıfın içeriğini ve işlevselliğini sağlayan kodun tamamını görmek gerekir.

### 3.11.OgretmenEklemeSayfasiState Sınıfı:

Bu Dart kodu, `OgretmenEklemeSayfasiState` adlı bir durum sınıfını içerir. Bu sınıf, öğretmen eklemek için gerekli olan formu ve bu formun işlevselliğini sağlayan metotları içerir. Aşağıda, kodun adım adım açıklaması bulunmaktadır:

1. `final GlobalKey<FormState> \_formKey`: Formun anahtarını tanımlar. Bu anahtar, formun durumunu kontrol etmek ve doğrulamak için kullanılır.

2. `final TextEditingController \_unvanController`: Öğretmenin ünvanını almak için bir `TextEditingController` tanımlar.

3. `final TextEditingController \_adController`: Öğretmenin adını almak için bir `TextEditingController` tanımlar.

4. `final TextEditingController \_soyadController`: Öğretmenin soyadını almak için bir `TextEditingController` tanımlar.

5. `build(BuildContext context)`: Sayfanın içeriğini oluşturan metottur. Bu metot, kullanıcı formu doldurduktan sonra kaydet butonuna bastığında `\_saveTeacherData` metodu çağrılır. Form elemanları, `TextFormField` ve `ElevatedButton` gibi Flutter widget'larını içerir.

6. `\_saveTeacherData(BuildContext context)`: Yeni bir öğretmenin kaydedilmesini sağlayan metottur. Bu metodun içerisinde, gerekli bilgileri alır, belirtilen dokümanın Firestore'da var olup olmadığını kontrol eder ve ardından Firestore'a yeni öğretmeni ekler.

7. Bu metot, kaydedilen öğretmenin ismini içeren bir dokümanın Firestore koleksiyonunda var olup olmadığını kontrol eder. Eğer öğretmen zaten ekliyse bir hata mesajı gösterir, aksi takdirde yeni öğretmenin bilgilerini Firestore'a ekler.

8. Formun durumu, `GlobalKey<FormState>` ile kontrol edilir ve form doğrulama işlevselliği içerir.

Bu kod parçası, öğretmen eklemek için gerekli olan sayfanın durumunu yönetir ve kullanıcıdan gerekli bilgileri alır. Firestore kullanılarak veri eklenir ve olası hatalar kontrol edilir. Sonuç olarak, kullanıcıya bir Snackbar mesajı ile işlemin başarılı olup olmadığı bildirilir.

### 3.12. ClassSchedulePage Sınıfı

Bu Dart kodu, bir sınıfın haftalık ders programını gösteren bir sayfa olan `ClassSchedulePage` adlı StatelessWidget'ı içerir. Aşağıda, kodun adım adım açıklaması bulunmaktadır:

1. `final String className`: Sayfanın oluşturulurken alınan, gösterilecek sınıfın adını temsil eden bir özelliktir.

2. `ClassSchedulePage({super.key, required this.className});`: `ClassSchedulePage` sınıfının kurucu metodudur. Bu metodun içinde sınıf adı alınır ve sınıfın haftalık ders programını gösteren bir sayfa oluşturulur.

3. `build(BuildContext context)`: Sayfanın içeriğini oluşturan metottur. Bu metot, Firestore'dan belirli bir sınıfa ait ders bilgilerini alır ve bu bilgileri kullanarak bir `GridView` içinde gösterir.

4. `FutureBuilder<QuerySnapshot>`: Bu widget, asenkron bir işlem sonucunda verileri göstermek için kullanılır. Bu durumda, Firestore'dan sınıfa ait ders bilgilerini almak için kullanılır.

5. `GridView.builder`: Ders bilgilerini göstermek için kullanılan bir GridView oluşturur. Her bir ders bilgisini temsil eden bir `Card` içindeki bir `ListTile` ile oluşturulur.

6. `Card`: Her ders bilgisini temsil eden bir karttır. `ListTile` içinde ders adı (`lessonName`), gün (`lessonDay`), saat (`lessonHour`) ve öğretmen adı (`teacherName`) gösterilir.

Bu kod parçası, belirli bir sınıfa ait haftalık ders programını gösteren bir sayfa oluşturur. Firestore'dan verilerin alınması için `FutureBuilder` kullanılır ve sonuçlara göre uygun widget'lar oluşturulur.

### 3.13. ManageDataPage Sınıfı:

Bu Dart kodu, `ManageDataPage` adlı bir StatefulWidget'ın tanımını içerir. Aşağıda, kodun adım adım açıklaması bulunmaktadır:

1. `ManageDataPage` sınıfı: `StatefulWidget` sınıfını genişletir. StatefulWidget'lar, içerdikleri durum nesnesi (state) ile widget'larını güncelleyebilen widget'lardır.

2. `const ManageDataPage({Key? key}) : super(key: key);`: `ManageDataPage` sınıfının kurucu metodudur. Burada, üst sınıfın (StatefulWidget) kurucu metoduna `key` parametresini ileterek widget'ı tanımlar.

3. `@override \_ManageDataPageState createState() => \_ManageDataPageState();`: `createState` metodu, `ManageDataPage` widget'ının durum nesnesini oluşturan metottur. `\_ManageDataPageState` adlı özel bir durum sınıfını oluşturur ve geri döndürür. State nesnesi, widget'ın durumuyla ilgili verileri içerir ve widget'ın güncellenmesini sağlar.

Bu kod parçası, bir veri yönetim sayfasının durumunu yöneten `ManageDataPage` adlı bir StatefulWidget'ı temsil eder.

### 3.14. \_ManageDataPageState Sınıfı:

Bu Dart kodu, bir veri yönetim sayfasını temsil eden `\_ManageDataPageState` adlı bir State sınıfını içerir. Bu sayfa, Firestore'dan ders, öğretmen ve sınıf verilerini yükleyebilir, bu verileri görüntüleyebilir, düzenleyebilir ve silebilir. İşte bu kodun adım adım açıklaması:

1. `\_ManageDataPageState` sınıfı: `ManageDataPage` widget'ının durumunu yöneten bir State sınıfıdır.

2. `\_dataList`: Ders, öğretmen ve sınıf verilerini depolayan bir liste. Bu liste, Firestore sorgularından elde edilen verileri içerir.

3. `initState()`: Widget'ın durumu oluşturulduğunda çağrılan bir metottur. Bu metot, sayfa oluşturulduğunda Firestore'dan ders, öğretmen ve sınıf verilerini yüklemek için `\_loadLessonsData()`, `\_loadTeachersData()`, `\_loadClassesData()` metodlarını çağırır.

4. `\_loadLessonsData()`, `\_loadTeachersData()`, `\_loadClassesData()`: Bu metotlar, sırasıyla ders, öğretmen ve sınıf verilerini Firestore'dan yükler ve `\_dataList` listesini günceller.

5. `\_loadDataFromCollection(String collectionName)`: Belirli bir koleksiyondan veri yüklemek için kullanılan genel bir metottur.

6. `\_deleteDocument(List<Map<String, dynamic>> dataList)`: Belirli bir belgeyi silmek için kullanılan bir metottur. Kullanıcıya silme işlemini onaylaması için bir pencere gösterir.

7. `\_editDocument(Map<String, dynamic> documentData)`: Belirli bir belgeyi düzenlemek için kullanılan bir metottur. Kullanıcıya değişiklikleri onaylaması için bir düzenleme penceresi gösterir.

8. `build(BuildContext context)`: Sayfa içeriğini oluşturan metottur. Bu metot, yüklenmiş verileri bir liste olarak gösterir ve her bir veri öğesi için düzenleme ve silme işlemlerini gerçekleştirmek için düğmeler sağlar.

Bu kod parçası, Firestore'dan veri yüklemek, düzenlemek ve silmek için kullanılan bir veri yönetim sayfasını temsil eder.

### 3.15. KameraEkrani Sınıfı:

Bu Dart kodu, bir veri yönetim sayfasını temsil eden `\_ManageDataPageState` adlı bir State sınıfını içerir. Bu sayfa, Firestore'dan ders, öğretmen ve sınıf verilerini yükleyebilir, bu verileri görüntüleyebilir, düzenleyebilir ve silebilir. İşte bu kodun adım adım açıklaması:

1. `\_ManageDataPageState` sınıfı: `ManageDataPage` widget'ının durumunu yöneten bir State sınıfıdır.

2. `\_dataList`: Ders, öğretmen ve sınıf verilerini depolayan bir liste. Bu liste, Firestore sorgularından elde edilen verileri içerir.

3. `initState()`: Widget'ın durumu oluşturulduğunda çağrılan bir metottur. Bu metot, sayfa oluşturulduğunda Firestore'dan ders, öğretmen ve sınıf verilerini yüklemek için `\_loadLessonsData()`, `\_loadTeachersData()`, `\_loadClassesData()` metodlarını çağırır.

4. `\_loadLessonsData()`, `\_loadTeachersData()`, `\_loadClassesData()`: Bu metotlar, sırasıyla ders, öğretmen ve sınıf verilerini Firestore'dan yükler ve `\_dataList` listesini günceller.

5. `\_loadDataFromCollection(String collectionName)`: Belirli bir koleksiyondan veri yüklemek için kullanılan genel bir metottur.

6. `\_deleteDocument(List<Map<String, dynamic>> dataList)`: Belirli bir belgeyi silmek için kullanılan bir metottur. Kullanıcıya silme işlemini onaylaması için bir pencere gösterir.

7. `\_editDocument(Map<String, dynamic> documentData)`: Belirli bir belgeyi düzenlemek için kullanılan bir metottur. Kullanıcıya değişiklikleri onaylaması için bir düzenleme penceresi gösterir.

8. `build(BuildContext context)`: Sayfa içeriğini oluşturan metottur. Bu metot, yüklenmiş verileri bir liste olarak gösterir ve her bir veri öğesi için düzenleme ve silme işlemlerini gerçekleştirmek için düğmeler sağlar.

Bu kod parçası, Firestore'dan veri yüklemek, düzenlemek ve silmek için kullanılan bir veri yönetim sayfasını temsil eder.

## KAYNAKÇA

1. <https://flutter.dev/>

2. <http://pub.dev/>

3. <https://www.geeksforgeeks.org/>

4. <https://www.phind.com/>

5. <https://firebase.google.com/docs>

6. <https://medium.com/@ravipatel84184/image-to-text-in-flutter-implement-ocr-for-efficient-text-extraction-71ac8ac1217a>

7. <https://github.com/vbalagovic/flutter_scalable_ocr/tree/main>