Navigasyon Mimarisi

1. Çoğu uygulama, farklı bilgi türlerini görüntülemek için birkaç ekran içerir. Bu örneğimizde sayfalar arası geçiş standart özelliklerinin tanımlandığı bir **navigasyon mimarisini** kullanacağız. Navigasyon mimarisinin sağlıklı çalışabilmesi için main.dart dosyasından başka dosyalara da ihtiyacımız olacak bu dosyalar. Bu dosyaları lib klasörünün içerisine oluşturun.

   Proje dizininde 4 dosya olmalıdır:

   main.dart → Uygulamayı başlatan ana dosya

   home_page.dart → Ana Sayfa

   notifications_page.dart → Bildirimler Sayfası

   about_page.dart → Hakkımda Sayfası

2. Bu dosyalardan biri olan **main.dart** ana kod bloğunu barındırır ve içerisindeki kodları aşağıdaki şekilde oluşturmaya başlayalım. Bu kodların işlevi:

   - **main.dart** çalıştığında **runApp ile MyApp()** sınıfı çağrılır.
   - **MyApp** sınıfı, içinde bir **MaterialApp** oluşturur (android uygulamanın ana yapısını tanımlayan sınıftır. Bu sınıf, `StatelessWidget`'tan türetilmiştir)
   - **MaterialApp**'in **home** parametresi ile **HomePage()** çağrılır ve uygulama açıldığında bu sayfa ekrana yansıtılır.

```dart
import 'package:flutter/material.dart';
import 'package:flutter_application_1/home_page.dart';

Run | Debug | Profile
void main() {
  runApp(const MyApp());
}
class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return const MaterialApp(
      home: HomePage(),
    ); // MaterialApp
  }
}
```

3. İkinci dosya **home_page.dart** dosyasıdır ve içerisinde bulunan kodları aşağıdaki gibi oluşturabiliriz:

   a. **Gerekli Kütüphanelerin Dahil Edilmesi:** Flutter'ın Material Design bileşenlerini kullanmak için gerekli olan kütüphaneyi projeye ekliyoruz.

   ```
   dart

   import 'package:flutter/material.dart';
   ```

   b. **HomePage Widget'ının Tanımlanması:** Menü elemanları tıklandığında sayfa değişeceği için durumlu bir widget olarak **HomePage Widget'ını oluşturuyoruz.**

   `createState()`: Bu metod, widget için **state** (durum) nesnesini oluşturur ve döndürür. **_HomePageState** sınıfı ile sayfanın davranışını tanımlayacağız.

   ```
   class HomePage extends StatefulWidget {
     const HomePage({super.key});

     @override
     State<HomePage> createState() => _HomePageState();
   }
   ```

   c. **build Metodu ile UI Oluşturulması:** `build()` kullanıcı arayüzünün oluşturulduğu metottur. `Scaffold` Flutter uygulamalarında sayfa yapısını oluşturmak için kullanılan temel widget'tır. AppBar, Drawer, Body gibi bölümleri içerir.

```
class _HomePageState extends State<HomePage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold();
  }
}
```

d. **AppBar – Uygulama Üst Çubuğu:** Scaffold widget'ının içerisinde önemli bir widget'ımız olan AppBar widget'ının oluşturalım. Burada
   - **AppBar**: Sayfanın üst kısmında başlık çubuğu oluşturur.
   - **title**: Çubuğun ortasında görüntülenen metni ayarlar.
   - **centerTitle**: Başlığın ortalanmasını sağlar.
   - **backgroundColor**: AppBar'ın arka plan rengini ayarlar (Deep Orange).

```
appBar: AppBar(
  title: const Text("Mobile Uygulamam"),
  centerTitle: true,
  backgroundColor: Colors.deepOrange,
),
```

e. **Debug Yazısından Kurtulma:** debugShowCheckedModeBanner Flutter'da MaterialApp bileşenine ait bir özelliktir. Bu kod, **yalnızca geliştirme aşamasında** kullanılır. Uygulama **yayına alındığında** (release modunda), bu banner zaten otomatik olarak gösterilmez. Biz bunu geliştirme aşamasında gizlemek için **main.dart** dosyasına debugShowCheckedModeBanner: false kodunu ekleyelim.

```
return const MaterialApp(
  home: HomePage(),
  debugShowCheckedModeBanner: false,
); // MaterialApp
```

f. **Drawer – Yan Menü Ekleme:** home_page.dart dosyasına geri dönelim ve AppBar'dan sonra bir yan menü ekleyelim. `Drawer`, uygulamalarda ekranın sol tarafından açılabilen **yan menü (hamburger menü)** yapısını sağlar. **Drawer** genellikle uygulamanın farklı sayfalarına veya işlevlerine hızlıca erişim sağlamak için kullanılır.

```
  ), // AppBar
  drawer: Drawer(),
); // Scaffold
```

g. **Safearea Kullanımı:** Emülatörde AppBar veya Drawer'ın üstteki durum çubuğuyla çakıştığını görüyorsunuz. `SafeArea`, Flutter'da ekranın çentik (notch), durum çubuğu (status bar), alt gezinme çubuğu gibi alanlarla çakışmasını önlemek için kullanılır. Bir widget'ın, bu alanlardan uzak durarak ekranın kullanılabilir alanı içinde kalmasını sağlar.

```
return SafeArea(
  child: Scaffold(
    appBar: AppBar(
```

h. **Drawer İçerisinde ListwView Kullanımı:** ListView menü öğelerini liste halinde tutar ve sayfa ekranını aşması durumunda kaydırılabilir hale getirir.

```
drawer: Drawer(
  child: ListView(
  ), // ListView
```

i. **İlk Menü Elemanını Ekleme:** ListView içerisine birden fazla widget eklemek için **children** propertysi kullanılır. Listenin ilk elemanı olarak da DrawerHeader Drawer içinde üstte görünen **başlık** kısmı olarak eklenmelidir. Bu alan genellikle kullanıcı bilgisi veya uygulama adını göstermek için kullanılır.

- **DrawerHeader**: Drawer'ın başlık kısmını temsil eder.
- **Container**: Drawer içerisinde bir kutu (box) alanı oluşturur. Burada sadece arka plan rengi **deepOrange** olarak belirlenmiştir.

```
child: ListView(
  children: [
    DrawerHeader(
        child: Container(
      color: Colors.deepOrange,
    )) // Container // DrawerHeader
  ],
), // ListView
```

j. **Öğeleri Yatay Eksende Ekleme: Row widget'ı** kullanılarak bir profil ikonu ve kullanıcı adı metnini DrawerHeader alanına ekleyelim. **Row**, öğeleri yatay düzlemde yan yana yerleştirir. **Children** parametresi içinde birden fazla widget yer alabilir (Icon ve Text gibi). Ek olarak, `mainAxisAlignment`, row içindeki öğelerin yatay eksende nasıl hizalanacağını belirler. Burada `MainAxisAlignment.center` kullanılarak ikon ve metin yatay eksende ortalanmıştır.

```
color: Colors.deepOrange,
child: const Row(
  mainAxisAlignment: MainAxisAlignment.center,
  children: [
    Icon(
      Icons.account_circle,
    ), // Icon
    Text("  Kullanıcı Adı"),
  ],
), // Row
```

k. **Menü veya Liste Öğeleri Oluşturma:** `ListTile` widget'ı, menü veya liste öğeleri oluşturmak için idealdir. Başlık (title) ve ikon (trailing) kullanarak, kullanıcı dostu bir arayüz sunar.

```
    )), // Container // DrawerHeader
    const ListTile(
      title: Text("Ana Sayfa"),
      trailing: Icon(Icons.home),
    ), // ListTile
    const ListTile(
      title: Text("Bildirimler"),
      trailing: Icon(Icons.notification_important),
    ) // ListTile
  ],
```

l. **Sayfa Gövdesini Oluşturma:** body:, Scaffold widget'ının ana içeriğini belirler. Sayfanın gövdesine hangi widget'ların yerleştirileceği burada tanımlanır. Burada Center widget'ı kullanılarak, sayfa içeriğini ekranın tam ortasına yerleştirelim.

```
    ), // Drawer
    body: const Center(
      child: Text("Ana Sayfa"),
    ), // Center
), // Scaffold
```

m. **Menü Öğeleri İçin Widget Fonksiyonu Oluşturma:** Kodlarımız menü itemları ile baya kalabalıklaştı burada eğer menü itemı ekleyen bir fonksiyon tanımlarsak yapılan menü tanımlama işlemi biraz daha kolaylaşacaktır. Ayrıca tüm menü itmelarını bu fonksiyonu kullanarak kolaylıkla değiştirebiliriz. Bunun için tüm menü itemlarını silin ve **myMenuItem** fonksiyonu StateFull widget sınıfının içerisine yazın. Burada her menü item'ı için değişken değerleri fonksiyonun içerisinde parametreler ksımında belirttiğimizi unutmayın.

```
81      )); // Scaffold // SafeArea
82    }
83
84    Widget myMenuItem(IconData myicon, String mytitle) => ListTile(
85          leading: Icon(
86            myicon,
87          ), // Icon
88          title: Text(mytitle),
89          trailing: const Icon(Icons.arrow_forward),
90        ); // ListTile
91  }
92
```

n. Yeni bir menü item'ı oluşturmak için DrawerHeader'dan sonra myMenuItem yazın. Çağırılan fonksiyonun sizden ihitiyaç duyulan parametreleri istediğini göreceksiniz.

```
    ), // DrawerHeader
    myMenuItem(Icons.home, "Ana Sayfa"),
    myMenuItem(Icons.notifications, "Bildirimler"),
    myMenuItem(Icons.accessibility, "Hakkımda")
  ])), // ListView // Drawer
```

o. Tüm menü öğelerini değiştirebilmek için artık ilgili fonksiyonu düzenlememiz yeterli olacaktır. Mesela icon rengini siyah yapmak istersek:

```
Widget myMenuItem(IconData myicon, String mytitle) => ListTile(
      leading: Icon(
        myicon,
        color: Colors.black,
      ), // Icon
```

4. Menü itemlerimiz hazır olduğuna göre diğer sayfaları oluşturup bağlantılarını verebiliriz. Bunun için **notifications.dart** ve **about_me.dart** isminde iki dosyayı lib klasörü içerisinde oluşturalım. Bu dosyalara material.dart paketini import ettirelim ve birer tane stateful widget ekleyip sınıf isimlerini Notifications ve AboutMe şeklinde, sayfa içeriklerini isei istediğimiz şekilde değiştirelim.

lib > notifications.dart > _NotificationsState

```
1  import 'package:flutter/material.dart';
2
3  class Notifications extends StatefulWidget {
4    const Notifications({super.key});
5
6    @override
7    State<Notifications> createState() => _NotificationsState();
8  }
9
10 class _NotificationsState extends State<Notifications> {
11   @override
12   Widget build(BuildContext context) {
13     return Scaffold(
14         appBar: AppBar(
15           title: const Text("Bildirimeler"),
16         ), // AppBar
17         body: const Center(
18           child: Text("Bildirimler sayfası"),
19         )); // Center // Scaffold
20   }
21 }
22
```

```dart
lib > about_me.dart > ...
1   import 'package:flutter/material.dart';
2
3   class AboutMe extends StatefulWidget {
4     const AboutMe({super.key});
5
6     @override
7     State<AboutMe> createState() => _AboutMeState();
8   }
9
10  class _AboutMeState extends State<AboutMe> {
11    @override
12    Widget build(BuildContext context) {
13      return Scaffold(
14          appBar: AppBar(
15            title: const Text("Hakkımda"),
16          ), // AppBar
17          body: const Center(
18            child: Text("Hakkımda Sayfası"),
19          )); // Center // Scaffold
20    }
21  }
```

4. **main.dart** dosyasında yollarımızı MaterialApp widget'ında **routes** propertysi ile belirleyelim. Burada **string ifadenin dosya ismi,** context içinde **çağırılan nesnenin ise bir sınıf** olduğunu unutmayalım.

```dart
return MaterialApp(
  routes: {
    "/home_page": (context) => HomePage(),
    "/notifications": (context) => Notifications(),
    "/about_me": (context) => AboutMe(),
  },
  home: HomePage(),
); // MaterialApp
```

- **routes**: Uygulamada kullanılacak tüm rotaları (sayfaları) bir **map (anahtar-değer çiftleri)** olarak tanımlar.
  - **Anahtar**: Rota adı (örneğin, "/home_page").
  - **Değer**: Bu rotaya karşılık gelen sayfa widget'ı (örneğin, HomePage)
- Kullanıcı, uygulama içinden bir menü öğesine veya butona tıkladığında, ilgili rota adı kullanılarak **Navigator.pushNamed()** metodu çalıştırılır.

- `context`, bir widget'ın **hangi ağaçta olduğunu** bilmesini sağlar. Bu sayede widget'ın üst widget'larla iletişim kurmasına ve **navigasyon işlemlerinin** doğru bir şekilde gerçekleşmesine yardımcı olur. **Rotalar** tanımlanırken de her yeni sayfa için `context` kullanılır; böylece uygulama, **sayfalar arasında geçiş yaparken** ilgili widget'ları doğru yerleştirebilir.

5. Bu yolları menü itemlerine atamak için, menümüzün olduğu **home_page.dart** dosyasına geri dönelim. Burada tüm menü itemlerının hepsini birden değiştirebileceğimiz **myMenuItem fonksiyonumuzda myroute** isminde bir string **parametre** girelim. Bu parametreyi de fonksiyon içerisinde **onTap: propetysi** ile yeni bir fonksiyon olarak **Navigator.pushNamed** komutu ile tanımlayalım.

```
Widget myMenuItem(IconData myicon, String mytitle, String myRoute) =>
    ListTile(
        leading: Icon(
          myicon,
          color: Colors.black,
        ), // Icon
        title: Text(mytitle),
        trailing: const Icon(Icons.arrow_forward),
        onTap: () => Navigator.pushNamed(context, myRoute)); // ListTile
}
```

6. Menü itemlarının hata verdiğini göreceksiniz. Bunun nedeni fonksiyonda bulunan parametre kadar değişkenin içeride tanımlanmamış olmasıdır. Bu tanımlama için main.dart dosyasındaki dosya yollarını string olarak girmek yeterli olacaktır.

```
myMenuItem(Icons.home, "Ana Sayfa", "/home_page"),
myMenuItem(Icons.notifications, "Bildirimler", "/notifications"),
myMenuItem(Icons.account_box, "Hakkımda", "/about_me"),
```

7. Ana sayfaya Appbar'dan geri döndüğümüzde drawer'ın açık olduğunu farketmişsinizdir. Bunu kapatmanın bir yolu var. Bunun için öncelikle **homepage'te** bulunan **onTap fonksiyonuna** gidelim ve bu fonksiyondaki komutları arttıracağımız için aşağıdaki şekilde kümeli parantezler içerisine alalım.

```
onTap: () => Navigator.pushNamed(context, myRoute),
```

```
onTap: () {
  Navigator.pushNamed(context, myRoute);
}); // ListTile
```

Ek olarak **Navigator.pop()** metodunu geçerli rotayı yönetilen **rota yığınından** kaldırmak için aşağıdaki gibi kullanalım. Böylece her yeni sayfaya gittiğimizde dönüşteki durum hali rota yığınında bulunmayacak homepage ilk açıldığı haliyle karşımıza gelecektir.

```
onTap: () {
  Navigator.pop(context);
  Navigator.pushNamed(context, myRoute);
}); // ListTile
```