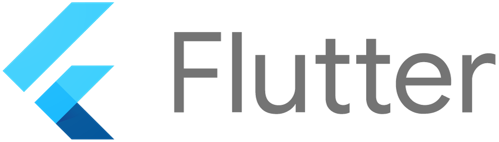
FLUTTER VSCODE

BEGUM UNER

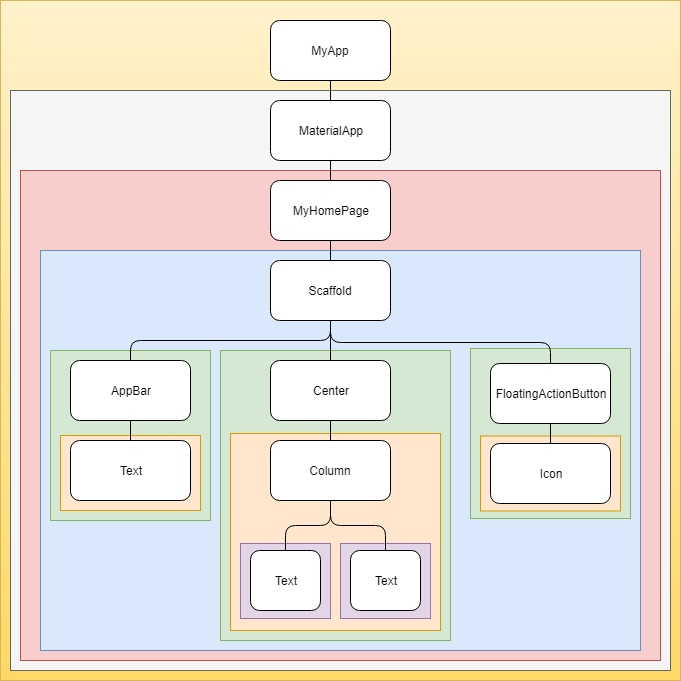
Istanbul University

****

**Flutter – VsCode**

* Flutter SDK kurulum ve Dart dilinin temellerinden daha önce bahsetmiştim. Şimdi Flutter’a başlıyoruz.
* Flutter, Google tarafından oluşturulan açık kaynaklı bir UI yazılım geliştirme kitidir. Android, IOS, Windows, Mac, Linux ve web için uygulamalar geliştirmek için kullanılıyor.
* Flutter, tamamen ücretsizdir.
* Flutter ile yazılım geliştirmesi yapabilmek için **Dart** adında bir programlama dili kullanılır. Dil Google tarafından Ekim 2011'de oluşturulmuştur.
* Flutter iki ana parçadan meydana geliyor;
  + **SDK (Yazılım Geliştirme Kiti):** Uygulamalarınızı daha kolay geliştirebilmeniz için birçok aracı hizmetinize sunar. SDK yazdığınız kodu hem IOS hemde Android için derleyebilmenizi sağlar.
  + **Framework (Birçok UI ve kütüphaneler):** Yazılım geliştirirken kullanılan birçok UI bileşeni (buttonlar, text inputlar vb.) hizmetinize sunar. Bu bileşenleri projelerinize göre özelleştirebilirsiniz.
* Flutter birçok geliştirme ortamında bulunmayan Hot reload adında bir gerçek zamanlı düzenleme özelliğine sahiptir. *Hot Reload* ile geliştiriciler uygulamalarını hazırlar veya düzenlerken, gerçek zamanlı olarak yaptıkları değişikliklerin uygulama üzerindeki yansımalarını anında görebilir
* VsCode kullanırken Control + S ye basıp yaptığınız değişikliği emulatorde görebilirsiniz.
* Flutter yapısı içerisinde gördüğümüz şeylerin %99’u widget. Her yer, her şey, her yapı widget’lar olarak tasarlanmış. Ve biz bu yapıda widget içerisinde widget, hatta yeri geldiğinde bir widget içerisinde onlarca widget kullanabiliyoruz. Kullanımı oldukça esnek olan bu widgetlerı istediğimiz gibi düzenleyebilmemiz ve kendi widgetlerimizi hızlıca yazabilmemiz flutterın artı özelliklerinden bir tanesi olarak gösterilebilir.

**(Genel widget yapısı)**

****

**Başlıyoruz!**

* İlk olarak VsCode’a girip command + shift + p yapıyoruz ve command palette açılıyor orada flutter: new project’i seçip ilk projemizi isimlendiriyoruz. Ve yerini seçiyoruz. İlk projemiz oluştu.
* Explorer’dan lib klasörünün içindeki main.dart’a giriyoruz ve içindeki her şeyi silip ilk flutter projemizi yazıyoruz.
* Klasik hello world’den başlıyoruz. Kodların üstlerinde açıklamalar var.
* <https://api.flutter.dev/flutter/material/MaterialApp-class.html> buradan girip widgetların özelliklerine bakabiliriz.
* Text widget ile çalışmak

Text: Ekrana yazı yazdırmak için kullandığımı widget türüdür.

* // Her flutter dosyamızda olması gereken import kavramı kütüphane ekler
* import 'package:flutter/material.dart';
* // main ana fonksiyonumuz
* void main()
* {
* //runApp hazır fonksiyonu widgetları ekrana basar.
* //center hazır widget
* runApp(Center(
* //child Center'in child'i denir ve bir daha widget kullanirsin Text widget kullandik
* child: Text("hello world", textDirection: TextDirection.ltr,),
* ));
* }
* Emulator’u çalıştırdık aşağıda görülüyor. Center widget’ı ortaladığı için ortada hello World yazıyor. Text widget’ı tek kullanılmadığı için textDirection’ı yazdık. Ltr-> left to right demek.

ekran, ekran görüntüsü, bilgisayar, dizüstü içeren bir resim

Açıklama otomatik olarak oluşturuldu

**MaterialApp Widget**

MaterialApp: Bu widget uygulamamızı en dıştan sarmalayacak olan yapı. Main sınıfımızda projemizi oluşturduktan sonra run app metodu içerisine genellikle bir material app veririz. Bunun nedenlerinden biride sayfalar arası geçişlerde bizim ihtiyacımız olan ‘routes’ kavramıdır.

**!** MaterialApp ’ın çalışması için home kesin olmalı.

import 'package:flutter/material.dart';

//materialApp class olusturduk runApp içinde

void main() {

//basina new yazadabilirsin yazmayada bilirsin farketmez

runApp(new MaterialApp(

//title koyduk

title: "My First App",

//home materialApp'in başlangıç noktası, içine başka widget istiyor

home: Material(

color: Colors.green,

//material'in alt sinifi center

child: Center(

//center'in alt sinifi text

child: Text(

"hello world", textDirection: TextDirection.ltr,

),

),

) ,

));

}

Emulator görüntüsü:

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Text Widget Style**

Text sınıfının içine style ile özellikler ekledik.

import 'package:flutter/material.dart';

//materialApp class olusturduk runApp icinde

void main() {

//basina new yazadabilirsin yazmayada bilirsin farketmez

runApp( new MaterialApp(

//title koyduk

title: "My First App",

//home materialApp'in baslangic noktasi, icine baska widget istiyor

home: Material(

color: Colors.green,

//material'in alt sinifi center

child: Center(

//center'in alt sinifi text

child: Text(

"hello world", textDirection: TextDirection.ltr,

style: TextStyle(color: Colors.red, fontSize: 50.0

)

),

),

) ,

));

}

* IOS emulator

ekran, elektronik eşyalar, ekran görüntüsü, bilgisayar içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Scaffold widget**

Bu widget bizim sayfalarımızın temel iskeletidir. Sayfamızın gövdesinde neler bulunacak, yandan açılır bir menüsü olacak mı olacaksa nasıl olacak, tepede bulunan header yazısı nasıl olacak bu widget içerisine yerleştiririz.

Scaffold Widget, Flutter uygulamasının temel ‘Material’ dizayn yapısını uygulayan bir çerçevedir. Bu widget bizlere şunları sağlar;

* Appbar
* Body
* Floating Action Button
* Drawer, EndDrawer
* Bottom Navigation Bar
* Background Color

**!** AppBar: Uygulamaların üst kısmında yer alan ve genellikle uygulama veya bulunulan sayfanın ismini yer alan yapıdır.

ekran görüntüsü, ekran, oturma, vitrin içeren bir resim

Açıklama otomatik olarak oluşturuldu

**!** Kodunuzun düzgün durması için sağ tıklayıp format document demeniz yeterli.

**!** Child ve children dite 2 parametre var biri tekil biri çoğul eğer içinde birden fazla eleman bulunan yapılar children diğeri child.

import 'package:flutter/material.dart';

//materialApp class olusturduk runApp icinde

void main() {

//basina new yazadabilirsin yazmayada bilirsin farketmez

runApp( new MaterialApp(

//title koyduk

title: "My First App",

//home materialApp'in baslangic noktasi, icine baska widget istiyor

//Scaffold widgeti bizim sayfalarımızın temel iskeletidir

home: Scaffold(

// AppBar uygulamaların üst kısmında yer alan ve genellikle isminin oldugu yerdir

appBar: AppBar(

backgroundColor: Colors.black,

title: Text("Appbar title", textDirection: TextDirection.ltr,

style: TextStyle(color: Colors.grey, backgroundColor: Colors.black

),

),

),

//scaffold body

body: Material(

color: Colors.green,

//material'in alt sinifi center

child: Center(

//center'in alt sinifi text

child: Text(

"hello world", textDirection: TextDirection.ltr,

style: TextStyle(color: Colors.red, fontSize: 50.0

)

),

),

),

)

));

}

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

**!** Emulatorun içindeki debug yazısını kaldırmak için runApp içine aşağıdaki kodu yazın.

debugShowCheckedModeBanner: false,

**Container widget:**

* Container’ı, Flutter’da istediğimiz gibi konumlandırabileceğimiz, şekillendirebileceğimiz ve içerisine nesneler yerleştirebileceğimiz kutular olarak tanımlayabiliriz. child özelliği ile içerisine tanımlanan nesnenin boyutunu otomatik olarak alabileceği gibi width ve height özellikleri ile kendimiz belirli bir boyutta tanımlayabiliriz. Eğer içerisine child özelliğini tanımlamasanız otomatik olarak ekranda verilen alana göre en büyük boyutu alır.
* **Container widget özellikleri:**

1. Alignment özelliği:

* Container içerisine tanımladığımız nesnenin container içerisindeki konumunu değiştirelim. Bunun için alignment özelliğini kullanacağız.
* Alignment özelliğini Alignment.center, Alignment.topCenter, Alignment.bottomCenter vb. gibi hazır parametrelerle belirleyebileceğimiz gibi koordinat düzlemindeki konumunu gösteren x ve y parametrelerini kullanabiliriz.

Container(

//container'ın yerini alignment ile belirliyoruz.

alignment: Alignment.center ,

color: Colors.amber,

child: Text(

"Asus Laptop",

textDirection: TextDirection.ltr,

style: TextStyle(color: Colors.blue, fontSize: 50.0),

),

);

1. Constraints özelliği:

* Container’a min ve max değer vermemizi sağlar.

import 'package:flutter/material.dart';

void main() {

runApp(MaterialApp(

home: Scaffold(

appBar: AppBar(

title: Text("data"),

),

body: Container(

color: Colors.purple,

constraints: BoxConstraints(

maxHeight: 200.0,

maxWidth: 100.0,

minHeight: 150.0,

minWidth: 50.0

),

child: Text("data"),

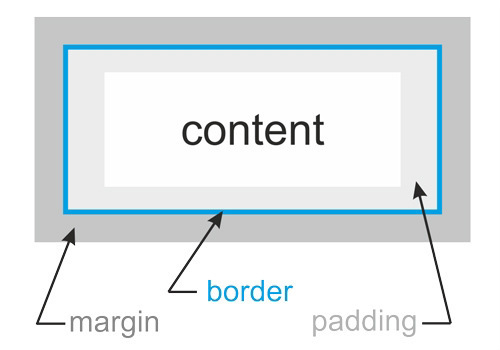
),

),

));

}

1. Margin özelliği:



* margin özelliğini kullanarak container`ın etrafını çevreleyen boş alanın boyutunu özelleştirebiliriz. İçine parametre olarak EdgeInsets alır.
* **EdgeInsets.all ():** Eğer bütün kenarlardan boşluk istiyorsak bu parametreyi kullanırız.
* **EdgeInsets.only ():** Eğer sadece belirli kenarlara margin değeri vermek istiyorsak bu parametreyi kullanırız.
* **EdgeInsets.symmetric ():** Eğer margin değerini yatay (horizontal) ve/veya dikey (vertical) olarak belirlemek istiyorsak bu parametreyi kullanabiliriz.
* **EdgeInsets.fromLTRB ():** Eğer margin değerlerini sırası ile left (sol), top (üst), right (sağ) ve bottom (alt) ile belirlemek isterseniz bu parametreyi kullanabilirsiniz.

Container(

//container'ın yerini alignment ile belirliyoruz.

alignment: Alignment.topLeft,

//margin: EdgeInsets.all(50.0) bu genel bir 50 her yonden

//bu asagidaki margin tek yonden 50

margin: EdgeInsets.only(left: 50.0, top: 50.0),

color: Colors.amber,

child: Text(

"Asus Laptop",

textDirection: TextDirection.ltr,

style: TextStyle(color: Colors.blue, fontSize: 30.0),

),

)

);

1. Padding özelliği:

* margin özelliğini kullanarak container ‘ın dışındaki boşluğu belirlemiştik. Şimdi ise padding özelliğini kullanarak container`ın içinde nesnelerle arasındaki boşluğu ayarlayabiliriz. İçine parametre olarak margin yapısıyla aynı olarak EdgeInsets parametresi kullanır. Kullanım detaylarını Margin başlığı altında inceleyebilirsiniz.

Container(

//container'ın yerini alignment ile belirliyoruz.

alignment: Alignment.topLeft,

//margin: EdgeInsets.all(50.0) bu genel bir 50 her yonden

//bu asagidaki margin tek yonden 50

margin: EdgeInsets.only(left: 50.0, top: 30.0),

padding: EdgeInsets.only(left: 50.0, right: 50.0, top: 50.0),

color: Colors.amber,

child: Text(

"Asus Laptop",

textDirection: TextDirection.ltr,

style: TextStyle(color: Colors.blue, fontSize: 30.0),),));

**Row widget**

* Row widgeti temel anlamda aynı satırda yan yana birden fazla widgeti göstermek istediğimiz zamanlar kullandığımız bir widgettir.
* Projemizi oluşturduğumuz zaman Scaffold içerisindeki body kısmına bir container eklemiştik. Eklemiş olduğumuz bu container’ın içerisine child olarak row ekliyorum. Tabi ki de bu row’un içerisine hemen her şeyi ekleyemiyoruz. Ekleme yapmak için children özelliğini kullanmamız gerekli.

//hepsi ortalansin diye center widget icine koyduk

Center(

child: Container(

//container'ın yerini alignment ile belirliyoruz.

alignment: Alignment.topLeft,

//margin: EdgeInsets.all(50.0) bu genel bir 50 her yonden

//bu asagidaki margin tek yonden 50

margin: EdgeInsets.only(left: 50.0, top: 30.0),

padding: EdgeInsets.only(left: 50.0, right: 50.0, top: 50.0),

color: Colors.amber,

child: Row(

//rowdaki diizilisi belirtmek icin textdirection kullaniyoruz

textDirection: TextDirection.ltr,

//row widgetinin children ozelliginden faydalandik icine istedigimiz kadar widget ekledik

children: <Widget>[

Text(

"Asus Laptop",

textDirection: TextDirection.ltr,

style: TextStyle(color: Colors.blue),

),

//Text ekrana sığmadığında asagi inmesi icin expended'in child'ina ekliyoruz.

Expanded(

child: Text( " --> 16 Gb ram, i7",

textDirection: TextDirection.ltr,

style: TextStyle(color: Colors.blue),

), ),

],

)));

**Column widget**

* Genel olarak column yapısı ve row yapısı aslında birbirinin aynısı diyebiliriz. Tek fark column içerisinde widgetleri alt alta sıralama imkânı bulurken, row içerisinde yan yana sıralayabiliyoruz. Bu yüzden bu yapılardan birisini iyi anladığınız zaman otomatik olarak ikinci yapıyı da anlayacaksınız.

Center(

child: Container(

//container'ın yerini alignment ile belirliyoruz.

alignment: Alignment.topLeft,

//margin: EdgeInsets.all(50.0) bu genel bir 50 her yonden

//bu asagidaki margin tek yonden 50

margin: EdgeInsets.only(left: 50.0, top: 30.0),

padding: EdgeInsets.only(left: 50.0, right: 50.0, top: 50.0),

color: Colors.amber,

child: Column(

children: [

Row(

//rowdaki diizilisi belirtmek icin textdirection kullaniyoruz

textDirection: TextDirection.ltr,

//row widgetinin children ozelliginden faydalandik icine istedigimiz kadar widget ekledik

children: <Widget>[

Text(

"Asus Laptop",

textDirection: TextDirection.ltr,

style: TextStyle(color: Colors.blue, fontSize: 20.0),

),

//Text ekrana sığmadığında asagi inmesi icin expended'in child'ina ekliyoruz.

Expanded(

child: Text( " --> 16 Gb ram, i7",

textDirection: TextDirection.ltr,

style: TextStyle(color: Colors.blue,fontSize: 20.0),

),

), ],

)],)));

Row ve Column Widgetlarının özelliği

* mainAxisAlignment: *MainAxisAlignment.center* olarak ayarlanırsa, Column içerisine eklenen tüm (children) widgetlar, yukarıdan aşağıya hizada ortalanır.
* crossAxisAlignment ise, tahmin edebileceğiniz gibi, sağdan-sola hizalama yapar. Örneğin, crossAxisAlignment: *CrossAxisAlignment.start* olarak ayarlarsanız, Column içerisindeki tüm widgetları, hepsi en sağa gelecek şekilde hizalamış olursunuz.

Row(

mainAxisAlignment: MainAxisAlignment.center,

crossAxisAlignment: CrossAxisAlignment.center,

children: <Widget>[

Icon(Icons.add\_circle, size: 65.0),

Icon(Icons.add\_circle, size: 65.0),

Icon(Icons.add\_circle, size: 65.0),

Icon(Icons.add\_circle, size: 65.0)

],

),

**Flutter image**

1. Bilgisayarımızdan image eklemek

* Bunun için, ilk olarak görsellerimizi saklayacağımız bir dosya oluştururuz.
* Dosyamızın içine de resimlerimizi ekliyoruz. Ardından pupspec.yaml dosyasında, eklediğimiz resimleri tanıtıyoruz. Assets kısminin yorum satiri işaretini kaldırıp kendi resmimizin adını – olan yere yazıyoruz
* Image.asset() widget’ı ile içinde, tırnak içine “kaynak/resim” şeklinde yazarak görselimizi ekleyebiliriz. Dilersek, width ve height ile de boyutlarını belirleyebiliriz.

Image.asset(

"images/a.png",

width: 100.0,

height: 60.0,),

1. Network ile image ekleme:

* URL adresi ile fotoğraf eklemek. Image.network kullanılır.

Image.network(

"https://static.simpsonswiki.com/images/8/81/Count\_Burns.png",

width: 100.0,

height: 60.0,),

1. Image’ları yuvarlak yapmak:

* CircleAvatar ile yapılıyor.

CircleAvatar(

backgroundImage: NetworkImage("https://avatars2.githubusercontent.com/u/32497416?s=460&u=ff4e1226e80b90a2ffe345cbf592e597cc0b73d4&v=4"),

backgroundColor: Colors.purple,

radius: 30.0,),

1. FadeinImage kullanımı

* Network ile internetten resim aldığımızda geç açıldığı için onu beklerken loading GIF’ı koymayı sağlar. GIF’ı pupspec.yaml dosyasına tanıtıyoruz.

FadeInImage.assetNetwork(placeholder: "images/k.gif",

image: "https://static.simpsonswiki.com/images/8/81/Count\_Burns.png",),

1. FlutterLogo kullanmak

* Flutter logosunu basmamızı sağlar.

FlutterLogo(size: 60),

1. PlaceHolder kullanmak

İsminden de anlaşıldığı gibi yer tutucudur. Henüz tasarım aşamasında uygulamalar içindir. Mesela buraya bir resim koyacağım ama daha sonra hallederim yeri belli olsun denildiği durumlarda kullanılır. Boyut belirtilmez ise varsayılan boyutu 400×400 dür. Bir farklı boyutlu küçük container veya başka bir widget içinde ise sığma taşma problemlerini aşmak için placeHolder ı bir Expanded içerisine alırız.

Expanded(child:

Placeholder(color: Colors.red, strokeWidth: 100.0)),

**Temel Buton Türleri**

1. **FloatingAction Button**

* FloatingActionButton (FAB) material design felsefesinin sağladığı, uygulama sayfasında birincil aksiyonu sağlayan genellikle dairesel butondur. Ekrandaki en belirgin buton olduğu için genelde pozitif aksiyonlarda kullanılması tavsiye edilir (Sayfa Geçişleri, İndirme, Ekleme). Diğer bir tavsiye ise ekranda tek bir FAB kullanmaktır. Farklı konumlarda kullanılabilse de en yaygın kullanımı ekranın sağ alt bölgesinde olanıdır.
* FAB, Flutter’daki diğer widgetlar gibi method olarak yazılır. onPressed parametresi zorunludur(required) ve butona tıklanınca yapılacak eylemi belirler. Eğer null döndürürse devre dışı kalır ve butona tıkladığımızda FAB tepki vermez. Yaygın kullanımda Scaffold(iskelet) yapısının içinde Scaffold.floatingActionButton şeklinde kullanılır.
* Scaffold(  
  floatingActionButton: FloatingActionButton(  
  onPressed: () {}, ),  
  );
* Böyle bir kullanımda onPressed null döndürür ve henüz bir child parametresi tanımlamadığımız için aşağıda görülebileceği gibi boş ve devre dışı butonla karşılaşırız.
* Şimdi child parametresi tanımlayalım. Child parametresi, FAB’ın içinde yer alacak olan widget’ı belirler ve genellikle icon widget’ı tanımlanır. Şimdi bir örnekte inceleyelim.
* Scaffold (  
  floatingActionButton: FloatingActionButton (  
  onPressed: () {},   
  child: Icon (Icons.alarm),),  
  );
* Icon widget’ının içinde Flutter’ın kendi iconlarını sunduğu Icons sınıfını kullandık.
* Eğer FAB’ daki iconun rengini değiştirmek istersek foreground parametresini kullanacağız.
* Scaffold(floatingActionButton: FloatingActionButton(  
  onPressed: counterAdd,  
  child: Icon(Icons.alarm),  
  foregroundColor: Colors.black,),  
  );
* Eğer FAB’ın arka plan rengini değiştirmek istersek backgroundColor parametresini kullanmamız gerekiyor.
* Scaffold(floatingActionButton: FloatingActionButton(  
  onPressed: counterAdd,  
  child: Icon(Icons.alarm),  
  backgroundColor: Colors.black,),  
  );
* Material design FloatingActionButton’da iki çeşit boyut seçeneği var: Default, Mini
* Scaffold(floatingActionButton: FloatingActionButton(  
  onPressed: counterAdd,  
  child: Icon(Icons.*alarm*,),  
  backgroundColor: Colors.*black*,  
  mini: true,),  
  );
* Material design, FAB’ı varsayılan olarak ekranın sağ alt köşesinde tanımlar. Bu konumu değiştirmek için Scaffold yapısının içinde floatingActionButtonLocation parametresini kullanacağız. Şimdi bir örnekle inceleyelim.
* Scaffold(floatingActionButton: FloatingActionButton(  
  onPressed: counterAdd,  
  child: Icon(Icons.*alarm*,),  
  backgroundColor: Colors.*black*,  
  ),  
  floatingActionButtonLocation:  
  FloatingActionButtonLocation.centerFloat, );

**!** navigationBar’ ın tam üstünde yer aldı. Daha iyi bir görünüm için bottomNavigationBar içine shape:CircularNotchedRectangle parametresini ekleyebiliriz

import 'package:flutter/material.dart';

void main() {

runApp(MaterialApp(

theme: ThemeData(

primarySwatch: Colors.red

),

home: Scaffold(

appBar: AppBar(

title: Text("new project"),

),

//onPressed icine fonksiyon bdekliyor bizde dartta bahsettigimiz lambda expression koyduk

floatingActionButton: FloatingActionButton(

onPressed: () {

debugPrint("FAB TIKLANDi");

},

child: Icon(

Icons.access\_alarm,

size: 48.0,

),

),

),

));

}

1. **Raised Button ve Directionality widget:**

* Her yerde gördüğümüz arka planında bir resim ve içerisinde bir yazı olan butonumuz işte bu. Raised Button en sık kullandığımız buton türüdür.
* Raised Button’un sık kullanılan özelliklerine bakacak olursak;
* **Color:** Butonumuzun rengidir. Eğer burada bir değer girmezsek butonumuz pasif olarak ekrana gelecektir.
* **HighlightColor:** Butonumuzun tıklandıktan sonra elimiz butona basılıyken aldığı renktir.
* **Elevation:** Butonumuzun etrafındaki gölge miktarıdır.
* **Child:** Butonumuzun içine alacağı widgetler burada girilir. Örnek vermek gerekirse resimde butonun içerisine sadece merhaba dünya yazdığımız için child olarak bir text widget verdik.
* **Padding:** Buton içerisine eklediğimiz widget ile butonun arasındaki mesafedir.
* **TextColor:** Buton içerisindeki yazıların alacağı renktir.
* **OnPressed:** Butonumuza tıklandıktan sonra nelerin olacağını buraya gireriz.

import 'package:flutter/material.dart';

void main() {

runApp(MaterialApp(

home: Scaffold(

appBar: AppBar(

title: Text("hello"),

),

body:

RaisedButton(

child: Text("Begum"),

elevation: 12,

textColor: Colors.red,

onPressed: () => debugPrint("you can")

)

), ));}

ekran görüntüsü, ekran, bilgisayar, dizüstü içeren bir resim

Açıklama otomatik olarak oluşturuldu

* Butonu başka bir class ’ta yazıp çalıştırmak
* Event kısmında bu kodun tamamı vardır.

//butun ortamlarda dogru calismasi icin directionality widgeti kullaniyoruz

ve calistirmak istediğimiz yere yapistiriyoruz

Directionality(

child: OrderButton() ,

textDirection: TextDirection.ltr,

)

//Buttonda bir classtir

class OrderButton extends StatelessWidget {

@override

Widget build(BuildContext context) {

var button = Container(

margin: EdgeInsets.only(top: 50.0),

padding: EdgeInsets.only(top: 50.0),

child: RaisedButton(

child: Text("go to checkout"),

elevation: 5.0,

onPressed: () {},

),

);

return button;

}}

1. **Icon Button**

Icon Button adından da anlaşılacağı üzere ikon şeklinde olan butonlarımızdır. İnstagram’da bulunan resim beğenme butonu veya bazı uygulamalarda bulunan ses artırıp azaltma butonları örnek verilebilir

IconButton(

icon: Icon(Icons.add\_circle, ),

onPressed: () => debugPrint("hi"))

1. **Flat Button**

Flat Button ise raised buttona göre bir arka plan rengi verilmediğinde dahi çalışabilen buton türüdür. Hatırlayacağınız üzere raised buttonda arka plana bir renk vermediğimiz zaman button otomatik olarak disabled geliyordu. Flat buttonda ise renk verilirse arka plana uygulanır ama verilmezse buton yine çalışır.

FlatButton( child:

Text("data",

style: TextStyle(color: Colors.red))

)

ekran, ekran görüntüsü, bilgisayar, iç mekan içeren bir resim

Açıklama otomatik olarak oluşturuldu

Eventlerle çalışmak:

import 'package:flutter/material.dart';

class Home extends StatelessWidget {

@override

Widget build(BuildContext context) {

return Center(

child: Row(

//rowdaki diizilisi belirtmek icin textdirection kullaniyoruz

textDirection: TextDirection.ltr,

//row widgetinin children ozelliginden faydalandik icine istedigimiz kadar widget ekledik

children: <Widget>[

//butun ortamlarda dogru calismasi icin directionality widgeti kullaniyoruz

Directionality(

child: OrderButton(),

textDirection: TextDirection.ltr,

)

],

)));

}

}

//Buttonda bir classtir

class OrderButton extends StatelessWidget {

@override

Widget build(BuildContext context) {

var button = Container(

margin: EdgeInsets.only(top: 50.0),

padding: EdgeInsets.only(top: 50.0),

child: RaisedButton(

child: Text("go to checkout"),

elevation: 5.0,

onPressed: () {

order(context);

},

),

);

return button;

}

}

// bir event calistiracak fonksiyon yaziyoruz

order(BuildContext context) {

var alert = AlertDialog(

title: Text("go to checkout"),

content: Text("i will go to checkout"),

);

//hazir fonksiyon

showDialog(

context: context,

builder: (BuildContext)=> alert

);}

**Kendi widget’ımızı yazalım**

Kendi widget’larımızı 2 şekilde yazabiliyoruz. Stateless ve stateful olarak.

**Stateless** kelimesini dilimize çevirdiğimizde durumsuz, belirtimsiz gibi anlamlara geldiğini görüyoruz. Flutter içerisinde ise bu sınıf ekran ilk oluşturulduğunda bir kez çalışan ve orada sabit bir şekilde kalan widget’lar için tasarlanmış. Örnek verecek olursak Text bir stateless widget’dir ve uygulama oluştuğu anda biz onu ekranın en tepesine koyarız, onunla herhangi bir işlem yapmayız ve o hala orada durmaya devam eder. Ta ki uygulama kapatılana kadar.

**Stateful** ise değişken halinde olabilecek widget’ler için kullanılır. Örnek verecek olursak ekranımızda ortadaki butona tıkladıkça sayının artmasını istiyoruz (Yeni bir proje oluşturduğumuz da otomatik gelen uygulama gibi…) sayının uygulama içinde kullanıcı tıkladıkça artmasını göstermek için bu sınıfı stateful olarak yapmak zorundayız.

**!** Son bir defa bu konuyu toplamak gerekirse ben bir hesap makinesi uygulaması yazıyorum diyelim. Uygulama açılış ekranında kullanıcının girdiği sayıları işlem yaptırıp ekrana yazdıracağım. Burada sınıfımı yazarken kullanacağım sınıf Stateful sınıfı, ama ben hakkımızda sayfası da yapmak istiyorum ve burada uygulama detay bilgileri iletişim bilgileri yer alacak. Kısaca burada herhangi bir işlem durumu yok. Bu durumda bu sayfayı da Stateless olarak oluşturmalıyım.

**1-)** Lib klasörünün altında dart uzantılı bir dosya oluştur ve içinde bir sınıf oluşturup extends ile bu sınıfı StatelessWidget’tan miras aldırıyorum.

**2-)** Stateless sınıfında tanımlı metotları override ediyorum. Hata veren yerin üzerine tıkladığınızda görebilirsiniz.

**3-)** Bu metodun geri dönüş tipini normal bir text widget yaparak istediğim özellikleri oraya giriyorum.

**4-)** İstediğim her yerde artık bu sınıfı import ederek kendi widget’ımı kullanabilirim. Main.dart’ta runApp içinde kendi class’ımı çağırdım.

import 'package:flutter/material.dart';

class HelloWorld extends StatelessWidget

{

@override

Widget build(BuildContext context) {

return MaterialApp(

title: "hello world",

home: Scaffold(

appBar: AppBar(

title: Text("widget"

),

),

body: Material(

child: Center(

child: Text("data"),

), ) ) );

throw UnimplementedError(); }

}

import 'package:flutter/material.dart';

import 'package:my\_widget/widget.dart';

void main() {

runApp(HelloWorld());

}

ekran görüntüsü, ekran, dizüstü, bilgisayar içeren bir resim

Açıklama otomatik olarak oluşturuldu

2 widget oluşturup birinin içinde diğerini çağırmak

import 'package:flutter/material.dart';

class HelloWorld extends StatelessWidget

{

@override

Widget build(BuildContext context) {

return MaterialApp(

title: "hello world",

home: Scaffold(

appBar: AppBar(

title: Text("widget"

),

),

body: MainPage()

));

throw UnimplementedError();

}

}

class MainPage extends StatelessWidget

{

@override

Widget build(BuildContext context) {

return Material(

child: Center(

child: Text("data"),

),

);

throw UnimplementedError();

}}

Fonksiyonlarla çalışmak

SelamVer fonksiyonu yazdık ve bunu mainpage classinda kullandık.

import 'package:flutter/material.dart';

class HelloWorld extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

title: "hello world",

home: Scaffold(

appBar: AppBar(

title: Text("widget"),

),

body: MainPage()));

throw UnimplementedError();

}

}

class MainPage extends StatelessWidget {

@override

Widget build(BuildContext context) {

return Material(

child: Center(

//text widget icinde selamVer fonk cagirdik

child: Text(selamVer()),

),

);

throw UnimplementedError();

}

}

String selamVer() {

//DateTime sinifindan datetime.now ile gunun simdiki saatini cektik

DateTime now = new DateTime.now();

int hour = now.hour;

if (hour < 12) {

return "gunaydin";

} else {

return "iyi aksamlar";

}

}

Main’de çağırdık

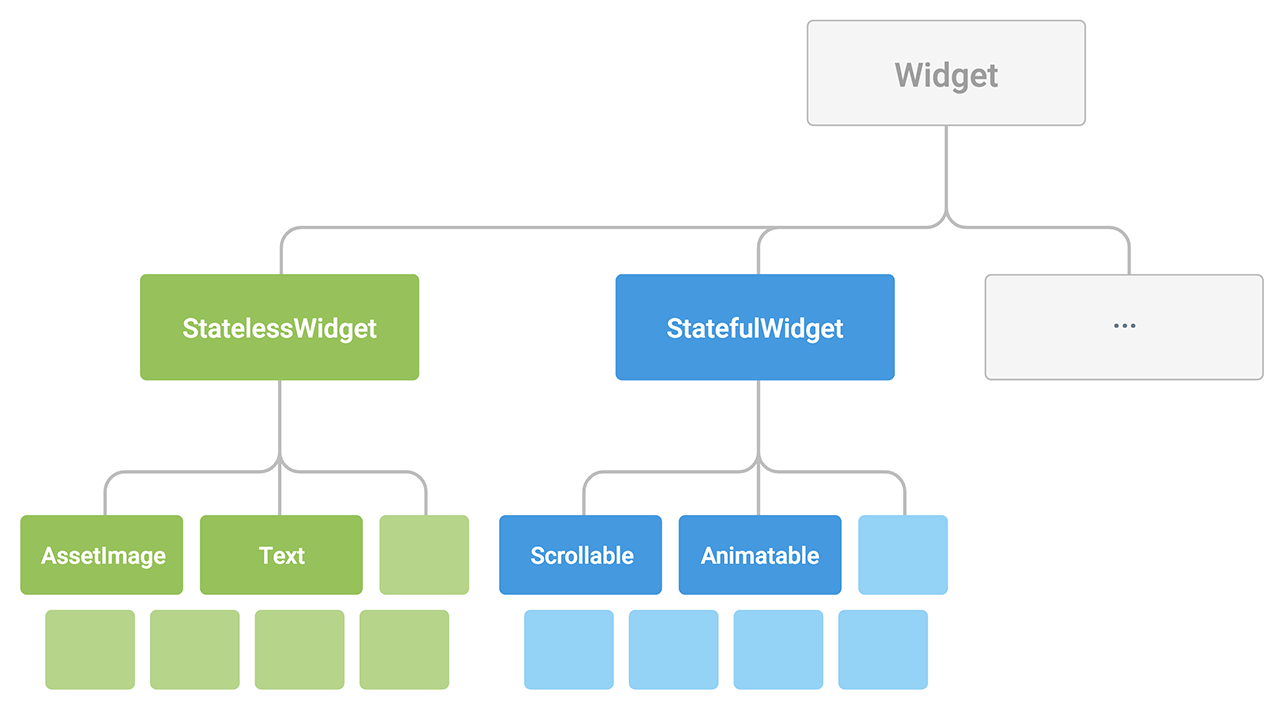
void main() {

runApp(HelloWorld());

}

**Statefull widget ile çalışmak**

* İçerisinde veri yönetimi yapacağımız widget’lar gibi düşünebiliriz.



* Stateless Widget’lar immutable’dır. Yani özellikleri değiştirilemez, durum değişikliği söz konusu değildir. Tüm değerler sabittir(final). Stateful Widget’lar da ise widget var olduğu süre boyunca state’ deki değişikliklerle kendilerini yeniden oluşturan widgetlardır ve stateful widget kullanabilmemiz için en az iki classa ihtiyaç duyarız:
* Stateful widget class
* State class

import 'package:flutter/material.dart';

void main() {

runApp(SuTakip());

}

class SuTakip extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

theme: ThemeData(primarySwatch: Colors.cyan), home: su());

}

}

class su extends StatefulWidget {

@override

\_suState createState() => \_suState();

}

//degistirecegimiz durumlari state widgetinda olusturuyoruz

class \_suState extends State<su> {

int sayi = 0;

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text("Su takip uygulamasi"),

centerTitle: true,

),

body: Center(

child: Text("bardak sayısı $sayi"),

),

floatingActionButton: FloatingActionButton(

onPressed: () => setState(() {

sayi++;

}),

child: Icon(Icons.add)),

);

}

}

**Flutter Liste Yapıları**

Flutter’da birden çok liste yapısı bulunuyor. Mobil uygulama geliştirirken çoklukla kullandığımız bu yapıları size anlatmaya başlıyorum.

1. **ListView (küçük listeler)**

Çok büyük listeler yapmayacaksak bize genel olarak yetecek bir liste türüdür. ListTile widget’i kullanarak görsel bir yapı sağlarız.

class EtkinlikListesi extends StatelessWidget {

@override

Widget build(BuildContext context) {

return ListView(

children: [

ListTile(

title: Text("Map"),

leading: Icon(Icons.map),

),

ListTile(

title: Text("Phone"),

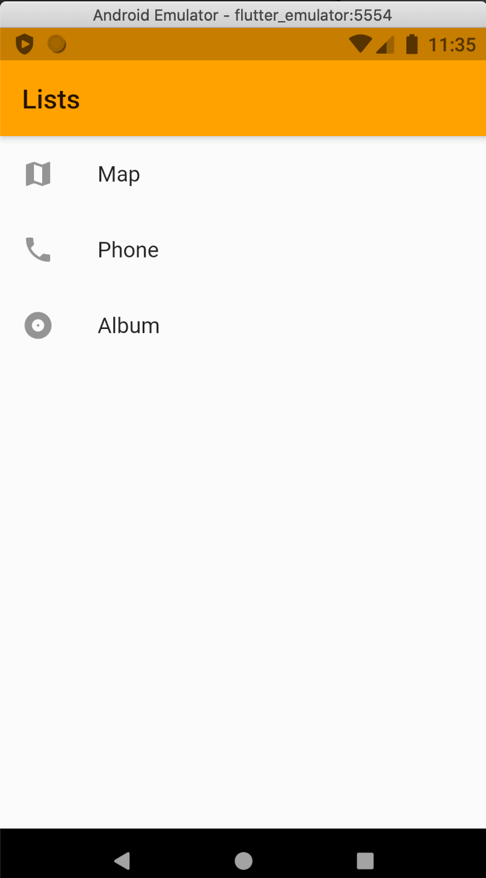
leading: Icon(Icons.phone),

),

ListTile(

title: Text("Album"),

leading: Icon(Icons.album), )]); }}



1. **Yatay Listeler (ListView)**

Yatay listeleri yine ListView kullanarak yapıyoruz. ListView’ı öğrendiysek çok kolay sadece bir satır ekliyoruz (scrollDirection : Axis.horizantal) ve ListTile yerine Container’ları kullanıyoruz.

class listeler extends StatelessWidget {

@override

Widget build(BuildContext context) {

return ListView(scrollDirection: Axis.horizontal, children: [

Container(

width: 130,

color: Colors.red,

),

Container(

width: 150,

color: Colors.amber,

),

Container(

width: 130,

color: Colors.black,

),

Container(

width: 130,

color: Colors.blue,

),

Container(

width: 130,

color: Colors.red,

),]);}}



1. **ListView.builder (büyük listeler)**

ListView.builder daha büyük listelerde kullanılır. İlk olarak bir data kaynağına ihtiyacımız var. Bunun için generate kavramını öğreneceğiz.

* **List.generate constructor**

Generate constructor’ina 2 parametre veriyoruz. Liste uzunluğu ve index değerimiz , index değerimiz 0 dan liste uzunluğu-1’e kadar uzanır.

Aşağıda bir örnek oluşturdum. Benim generate constructor’ım her index’i ekrana yazdıracak.

List items = List.generate(1000, (index) => print("$index"));

* **ListView.builder kullanımı**

Data kaynağını oluşturduk şimdi her index’i kendi satırında gösterelim. Bunun için ListView.builder kullanıyoruz. ListView.builder içinde itemBuilder kullanmak zorunlu 2 parametresi var biri override fonksiyonumuzun parametresi buildContext türünde o yüzden ona gelen context değerini verdik direk diğeri ise index biz bu fonksiyonun içine ListTile koyduk daha güzel bir görüntüsü olması için

import 'package:flutter/material.dart';

class listeler extends StatelessWidget {

@override

Widget build(BuildContext context) {

List items = List.generate(1000, (index) => print("item $index"));

return ListView.builder(itemBuilder: (context, index) {

return ListTile(

title: Text("index $index"),

); }); }}

tablo içeren bir resim

Açıklama otomatik olarak oluşturuldu

1. **Grid Listeleri**

* Elemanlarınızı alt alta göstermek yerine 1 satırda 2,3 tane göstermek istediğimizde kullanırız. GridView.count kullanarak yapıyoruz. crossAxisCount her satırda kaç eleman istediğimizi yazıyoruz. Children içinde ben bir liste oluşturdum ve generate metodu ile ekrana veri yazdırdık.

import 'package:flutter/material.dart';

class listeler extends StatelessWidget {

@override

Widget build(BuildContext context) {

return GridView.count(

crossAxisCount: 2,

children: List.generate(100, (index) {

return Center(

child: Container(

alignment: Alignment.center,

color: Colors.amber,

height: 150,

width: 150,

child: Text("index $index" , style: TextStyle(color: Colors.teal),),

)

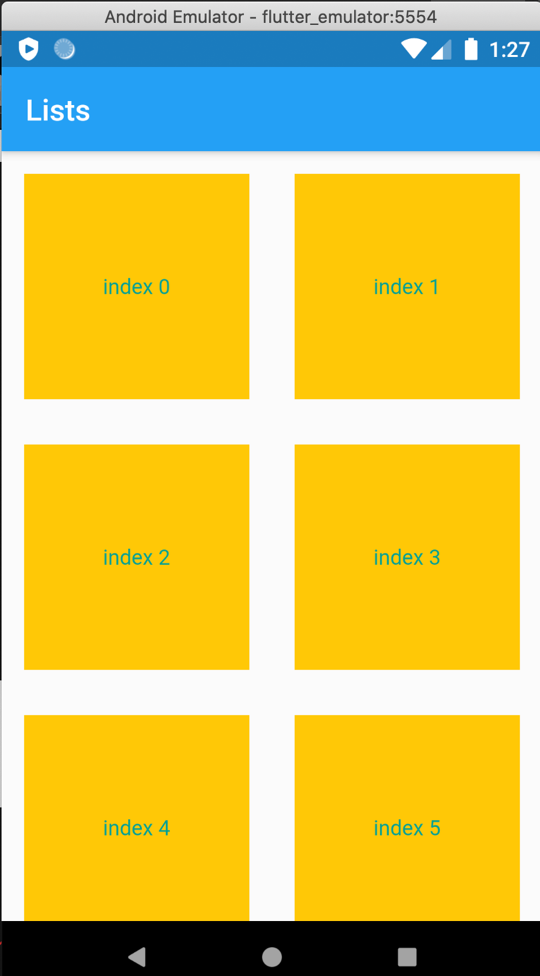
);

}),

);

}

}



* GridView.builder

SliverGridDelegateWithFixedCrossAxisCount (crossAxisCount: 4) ile her satırda 4 eleman olmasını istedik. ItemCount içine kaç elemanımızın olacağını yazdık. ListView.builder’ da yaptığımız gibi itembuilder kullanılır.

import 'package:flutter/material.dart';

class EtkinlikListesi extends StatelessWidget {

@override

Widget build(BuildContext context) {

return GridView.builder(

itemCount: 100,

gridDelegate:

SliverGridDelegateWithFixedCrossAxisCount(crossAxisCount: 4),

itemBuilder: (BuildContext context, int index) {

return Container(

alignment: Alignment.center,

child: Text("data $index"),

color: Colors.teal[100 \* (index%9)],

);

},

);

}

}

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

1. **Listenin başında appBar kullanmak**

appBar ozelligini scaffold içinde kullanıyorduk bu sefer onun içine customscroll yazıp içinde kullanacağız.

* 3 adımda yapabiliriz
* İlk olarak CustomScrollView oluşturacağız.
* Daha sonra SliverAppBar kullanacağız
* En son SliverList kullanarak listeyi ekleyeceğiz
* CustomScrollView oluşturmak

//import 'package:deneme5/new.dart';

import 'package:deneme5/new.dart';

import 'package:flutter/material.dart';

void main() {

runApp(MaterialApp(

title: "flutter dersleri",

debugShowCheckedModeBanner: false,

home: Scaffold(

body: CustomScrollView(

slivers: [],

),

),

));

}

* SliverAppBar kullanıyoruz.

//import 'package:deneme5/new.dart';

import 'package:flutter/material.dart';

void main() {

runApp(MaterialApp(

title: "flutter dersleri",

debugShowCheckedModeBanner: false,

home: Scaffold(

body: CustomScrollView(

slivers: [

SliverAppBar(

title: Text("degisen appbar"),

floating: true,

)

* SliverList

//import 'package:deneme5/new.dart';

import 'package:flutter/material.dart';

void main() {

runApp(MaterialApp(

title: "flutter dersleri",

debugShowCheckedModeBanner: false,

home: Scaffold(

body: CustomScrollView(

slivers: [

SliverAppBar(

title: Text("degisen appbar"),

floating: true,

),

SliverList(

delegate: SliverChildBuilderDelegate (

(context,index) =>ListTile(title: Text("index $index")),

childCount: 100,

), ), ], ), ) ));}

tablo içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Card classi ve divider**

Materyal tasarım kartı: Hafifçe yuvarlatılmış köşelere ve bir yükseklik gölgesine sahip bir panel.

class ListeKonuAnlatimi extends StatelessWidget {

@override

Widget build(BuildContext context) {

//card bir widget olmadigi icin ama fonksiyonda widget dondurmasini bekledigi icin bir widget ile cevreledik container kullandik

return Column(

children: [

Container(

//listeleri kullanirken genel olarak her bir elemani carda aliriz

child: Card(

color: Colors.teal,

margin: EdgeInsets.all(4),

elevation: 20,

//genellikle childlari listTile isimli sinifi kullanarak yapariz

child: ListTile(

leading: CircleAvatar(child: Icon(Icons.ios\_share)),

title: Text("liste eleman baslik"),

subtitle: Text("liste elemani alt baslik"),

trailing: Icon(Icons.add\_circle),

)),), //ince bir cizgi

Divider(color:Colors.red, height: 20),

//buraya istenilen kadar container eklenebilir

], ); }}

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Boxdecoration kullanmak**

Container’larımızı tasarlamamızı sağlar.

import 'package:flutter/material.dart';

class EtkinlikListesi extends StatelessWidget {

@override

Widget build(BuildContext context) {

return GridView.builder(

itemCount: 100,

gridDelegate:

SliverGridDelegateWithFixedCrossAxisCount(crossAxisCount: 2),

itemBuilder: (BuildContext context, int index) {

return Container(

//kutu decorasyonu yapmak icin decoration kullaniyoruz

decoration: BoxDecoration(

// koselerini renklendiriyoruz

border: Border.all(

color: Colors.green, width: 5, style: BorderStyle.solid),

color: Colors.red[100 \* (index % 9)],

//renk gecisi yapmak

gradient: LinearGradient(

colors: [Colors.black, Colors.red],

begin: Alignment.topCenter,

end: Alignment.bottomCenter),

image: DecorationImage(

image: NetworkImage(

"https://upload.wikimedia.org/wikipedia/tr/8/84/Figüran\_Bob.png"),

fit: BoxFit.contain),

//boxlarin sekli icin

shape: BoxShape.circle

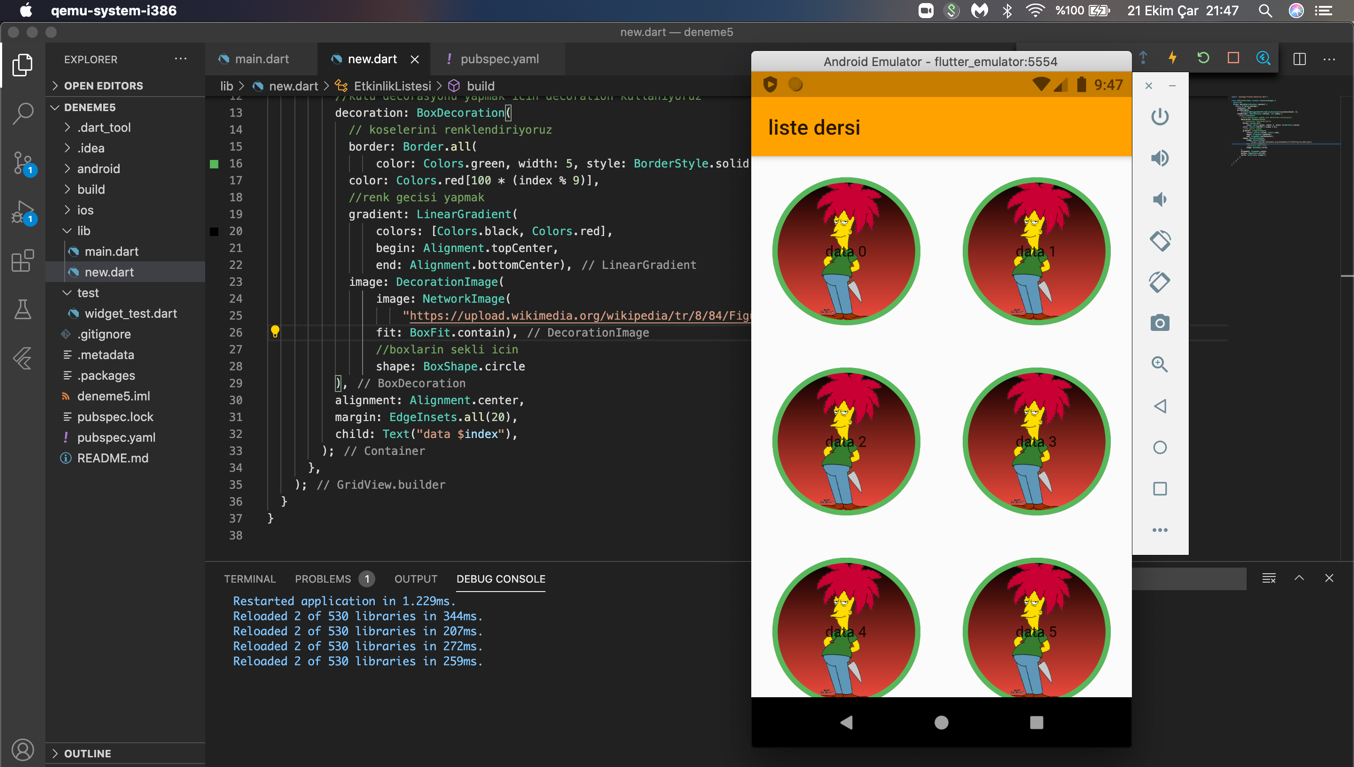
),

alignment: Alignment.center,

margin: EdgeInsets.all(20),

child: Text("data $index"),

); }, );}}



**Navigate işlemleri**

Başka bir sayfaya geçmek için kullanılır.

3 adımda gerçekleştirilir

* Ilk olarak 2 sayfa oluşturulur
* Navigator.push() eklenir anasayfaya
* 2.sayfaya navigatör.pop() oluşturur
* 2 sayfa olusturduk

import 'package:flutter/material.dart';

void main() {

runApp(MaterialApp(

home: AnaSayfa(),

));

}

class AnaSayfa extends StatelessWidget {

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text("ANASAYFA"),

),

body: Center(

child: RaisedButton(

child: Text("Diger sayfaya gec"),

onPressed: (){}),

),

);

}

}

class IkinciSayfa extends StatelessWidget {

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text("2. SAYFA"),

),

body: Center(

child: RaisedButton(

child: Text("anasayfaya geri dön"),

onPressed: (){}),

),

);}}

* Anasayfaya navigator.push ekledik

class AnaSayfa extends StatelessWidget {

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text("ANASAYFA"),

),

body: Center(

child: RaisedButton(

child: Text("Diger sayfaya gec"),

onPressed: (){

Navigator.push(context, MaterialPageRoute(builder: (context) => IkinciSayfa()));

}),

),

);

}

}

* 2. Sayfaya navigator.pop ekledik

class IkinciSayfa extends StatelessWidget {

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text("2. SAYFA"),

),

body: Center(

child: RaisedButton(

child: Text("anasayfaya geri dön"),

onPressed: () {

Navigator.pop(context);

}),

),

);

}

Butun proje:

import 'package:flutter/material.dart';

void main() {

runApp(MaterialApp(

home: AnaSayfa(),

));

}

class AnaSayfa extends StatelessWidget {

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text("ANASAYFA"),

),

body: Center(

child: RaisedButton(

child: Text("Diger sayfaya gec"),

onPressed: () {

Navigator.push(context,

MaterialPageRoute(builder: (context) => IkinciSayfa()));

}),

),

);

}

}

class IkinciSayfa extends StatelessWidget {

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text("2. SAYFA"),

),

body: Center(

child: RaisedButton(

child: Text("anasayfaya geri dön"),

onPressed: () {

Navigator.pop(context);

}),

),

);

}

}

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Argüman göndermek

3 tane sınıf olusturduk yukarıdaki gibi birbirine bağladık ve b sınıfına veri gönderdik.

import 'package:flutter/material.dart';

void main() {

runApp(MaterialApp(

home: AnaSayfa(),

));

}

class AnaSayfa extends StatelessWidget {

String baslik = "aa";

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text("ANASAYFA"),

),

body: Center(

child: Column(

children: [

RaisedButton(

child: Text("2. sayfaya gec"),

onPressed: () {

Navigator.push(

context,

MaterialPageRoute(

builder: (context) => IkinciSayfa(baslik)));

}),

RaisedButton(

child: Text("3. sayfaya gec"),

onPressed: () {

Navigator.push(context,

MaterialPageRoute(builder: (context) => UcuncuSayfa()));

}),

],

)),

);

}

}

class IkinciSayfa extends StatelessWidget {

//yukarida ikincisayfa iciine parametre verdik o yuzden bunu tanimladik

String gelen;

IkinciSayfa(this.gelen) {}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text("2. SAYFA $gelen"),

),

body: Center(

child: RaisedButton(

child: Text("anasayfaya geri dön"),

onPressed: () {

Navigator.pop(context);

}),

),

);

}

}

class UcuncuSayfa extends StatelessWidget {

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text("3. SAYFA"),

),

body: Center(

child: RaisedButton(

child: Text("anasayfaya geri dön"),

onPressed: () {

Navigator.pop(context);

}),

),

);

}

}

PushReplacement

Uygulamadan çıkmak için kullanılır başka bir sayfaya yönlendirmez direk çıkar.

onPressed: () {

Navigator.pushReplacement(context,

MaterialPageRoute(builder: (context) => UcuncuSayfa()));

İsimlendirilmiş Rota

Rotalara isim verip push değil pushreplacementnamed yapıp string yazarak ulaşabiliyoruz.

2 farklı klasörde yazdım daha düzenli olsun diye

Main klasörü:

import 'package:flutter/material.dart';

import 'package:navigate/new.dart';

void main() {

runApp(MaterialApp(

routes: {

// "/" isareti Anasayfa demek bu yuzden home u anasayfadan baslatmaya gerek yok bu isaret nereden baslayacagini gosteriyor

'/' : (context) => AnaSayfa(),

'/Bpage' : (context) => IkinciSayfa(),

'/Cpage' : (context) => UcuncuSayfa(),

},

//home: AnaSayfa(),));}

Diğer Klasör:

import 'package:flutter/material.dart';

class AnaSayfa extends StatelessWidget {

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text("ANASAYFA"),

),

body: Center(

child: RaisedButton(

child: Text("Diger sayfaya gec"),

onPressed: () {

Navigator.pushNamed(context, '/Bpage');

}),

),

);

}

}

class IkinciSayfa extends StatelessWidget {

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text("2. SAYFA"),

),

body: Center(

child: RaisedButton(

child: Text("3.sayfaya git"),

onPressed: () {

Navigator.pushNamed(context, '/Cpage');

}), ), ); }}

class UcuncuSayfa extends StatelessWidget {

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text("3. SAYFA"),

),

body: Center(

child: RaisedButton(

child: Text("2. sayfaya git"),

onPressed: () {

Navigator.pushNamed(context, '/Bpage');

}),

),

);

}}

**GestureDetector Widget**

Hareketleri algılayan bir widget. Kısaca kapsadığı alanı dokunmaya duyarlı hale getirir.

GestureDetector( // GestureDetector

child: Text(

'Hello world',

style: TextStyle(

fontSize: 30,

color: textColor,

),

),

onTap: () { // onTap

\_doSomething();

},

);

}

void \_doSomething() {

setState(() {

// have to import 'dart:math' in order to use Random()

int randomHexColor = Random().nextInt(0xFFFFFF);

int opaqueColor = 0xFF000000 + randomHexColor;

textColor = Color(opaqueColor);

});

}

}

# ! VsCode extension kısmından “Getter and Setter Generator” yazın ve en baştakini indirin. Get ve set metodu olmasını istediğiniz değişkenleri seçip comand+shift+p yapın ve generate set and get methods yazın ve tıklayın. Bazı yerlerde hatalar olabilir düzeltmeyi unutmayın.

**Input Widget**

* Form Altyapısının kurulması:

Statefull widget içine yazılır. Form’ un içine birçok şey eklemek için child kısmına column verdik.

class CustomerAdd extends StatefulWidget {

@override

\_CustomerAddState createState() => \_CustomerAddState();

}

class \_CustomerAddState extends State<CustomerAdd> {

@override

Widget build(BuildContext context) {

return Container(

margin: EdgeInsets.all(25.0),

child: Form(child: Column(

children: [

Text("data buraya gelecek")

],

)),

);

}

}

* Form yapısının yazılması:

Daha düzenli durması için bütün textField alanlarını fonksiyon olarak yazdık.

import 'package:flutter/material.dart';

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

home: Scaffold(

appBar: AppBar(

title: Text("data"),

),

body: CustomerAdd(),

),

);

}

}

class CustomerAdd extends StatefulWidget {

@override

\_CustomerAddState createState() => \_CustomerAddState();

}

class \_CustomerAddState extends State<CustomerAdd> {

@override

Widget build(BuildContext context) {

return Container(

margin: EdgeInsets.all(25.0),

child: Form(

child: Column(

children: [

fistName(),

secondName(),

eMail(),

password(),

submitButton(),

],

)),

);

}

Widget fistName() {

return TextFormField(

decoration:

InputDecoration(labelText: "first name", hintText: "your name"),

);

}

}

Widget secondName() {

return TextFormField(

decoration: InputDecoration(

labelText: "second name", hintText: "your secondname")

);

}

Widget eMail() {

return TextFormField(

keyboardType: TextInputType.emailAddress,

decoration: InputDecoration(labelText: "Email", hintText: "Email"));

}

Widget password() {

return TextFormField(

obscureText: true,

decoration: InputDecoration(labelText: "password", hintText: "password"));

}

Widget submitButton() {

return RaisedButton(child: Text("submit"), onPressed: () {});

}

* Input type ve Obscure Text ile çalışmak

Obscure değerini true yapınca password’u göremiyoruz.

Input type özelliğini email içinde kullandik.

Widget eMail() {

return TextFormField(

keyboardType: TextInputType.emailAddress,

decoration: InputDecoration(labelText: "Email", hintText: "Email"));

}

Widget password() {

return TextFormField(

obscureText: true,

decoration: InputDecoration(labelText: "password", hintText: "password"));

}

* Validator

Widget fistName() {

return TextFormField(

decoration:

InputDecoration(labelText: "first name", hintText: "your name"),

validator: (value) {

if (value.length < 2) {

return "Name field must least two character";

}

}, ); }}

* GlobalKey

Globalkey ile id atayip yönetebiliyoruz.

import 'package:flutter/material.dart';

final formKey = GlobalKey<FormState>();

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

home: Scaffold(

appBar: AppBar(

title: Text("data"),

),

body: CustomerAdd(),

),

);

}

}

class CustomerAdd extends StatefulWidget {

@override

\_CustomerAddState createState() => \_CustomerAddState();

}

class \_CustomerAddState extends State<CustomerAdd> {

@override

Widget build(BuildContext context) {

return Container(

margin: EdgeInsets.all(25.0),

child: Form(

key: formKey,

child: Column(

children: [

fistName(),

secondName(),

eMail(),

password(),

submitButton(),

],

)),

);

}

Widget fistName() {

return TextFormField(

decoration:

InputDecoration(labelText: "first name", hintText: "your name"),

validator: (value) {

if (value.length < 2) {

return "Name field must least two character";

}

},

);

}

}

Widget secondName() {

return TextFormField(

decoration: InputDecoration(

labelText: "second name", hintText: "your secondname"));

}

Widget eMail() {

return TextFormField(

keyboardType: TextInputType.emailAddress,

decoration: InputDecoration(labelText: "Email", hintText: "Email"));

}

Widget password() {

return TextFormField(

obscureText: true,

decoration: InputDecoration(labelText: "password", hintText: "password"));

}

Widget submitButton() {

return RaisedButton(

child: Text("submit"),

onPressed: () {

formKey.currentState.validate();

});

}

* validationMixin ile calismak

main.dart

import 'package:deneme2/validation.dart';

import 'package:flutter/material.dart';

final formKey = GlobalKey<FormState>();

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

home: Scaffold(

appBar: AppBar(

title: Text("data"),

),

body: CustomerAdd(),

),

);

}

}

class CustomerAdd extends StatefulWidget {

@override

\_CustomerAddState createState() => \_CustomerAddState();

}

class \_CustomerAddState extends State<CustomerAdd> with ValidationMixin {

@override

Widget build(BuildContext context) {

return Container(

margin: EdgeInsets.all(25.0),

child: Form(

key: formKey,

child: Column(

children: [

fistName(),

secondName(),

eMail(),

password(),

submitButton(),

],

)),

);

}

Widget fistName() {

return TextFormField(

decoration:

InputDecoration(labelText: "first name", hintText: "your name"),

validator: validateFirstName);

}

Widget secondName() {

return TextFormField(

decoration: InputDecoration(

labelText: "second name", hintText: "your secondname"),

validator: validateSecondName,

);

}

Widget eMail() {

return TextFormField(

keyboardType: TextInputType.emailAddress,

decoration: InputDecoration(labelText: "Email", hintText: "Email"),

validator: validateEmail,

);

}

Widget password() {

return TextFormField(

obscureText: true,

decoration:

InputDecoration(labelText: "password", hintText: "password"));

}

Widget submitButton() {

return RaisedButton(

child: Text("submit"),

onPressed: () {

formKey.currentState.validate();

});

}

}

validation.dart

import 'package:flutter/material.dart';

class ValidationMixin {

String validateFirstName(String value){

if (value.length < 2) {

return "Name field must least two character";

}

return null;

}

String validateSecondName(String value){

if (value.length < 2) {

return "SecondName field must least two character";

}

return null;

}

String validateEmail(String value){

if (!value.contains("@")) {

return "Mail is not true";

}

return null;

}

}

Save

customer

class Customer {

String firstName;

String secondName;

String eMail;

String password;

}

main

import 'package:deneme2/customer.dart';

import 'package:deneme2/validation.dart';

import 'package:flutter/material.dart';

final formKey = GlobalKey<FormState>();

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

home: Scaffold(

appBar: AppBar(

title: Text("data"),

),

body: CustomerAdd(),

),

);

}

}

class CustomerAdd extends StatefulWidget {

@override

\_CustomerAddState createState() => \_CustomerAddState();

}

class \_CustomerAddState extends State<CustomerAdd> with ValidationMixin {

final customer = Customer();

@override

Widget build(BuildContext context) {

return Container(

margin: EdgeInsets.all(25.0),

child: Form(

key: formKey,

child: Column(

children: [

fistName(),

secondName(),

eMail(),

password(),

submitButton(),

],

)),

);

}

Widget fistName() {

return TextFormField(

decoration:

InputDecoration(labelText: "first name", hintText: "your name"),

validator: validateFirstName,

onSaved: (String value) {

customer.firstName = value;

},

);

}

Widget secondName() {

return TextFormField(

decoration: InputDecoration(

labelText: "second name", hintText: "your secondname"),

validator: validateSecondName,

);

}

Widget eMail() {

return TextFormField(

keyboardType: TextInputType.emailAddress,

decoration: InputDecoration(labelText: "Email", hintText: "Email"),

validator: validateEmail,

);

}

Widget password() {

return TextFormField(

obscureText: true,

decoration:

InputDecoration(labelText: "password", hintText: "password"));

}

Widget submitButton() {

return RaisedButton(

child: Text("submit"),

onPressed: () {

if (formKey.currentState.validate == true) {

formKey.currentState.save();

saveCustomer(customer);

}

});

}

}

void saveCustomer(Customer customer) {}

CheckBox Slider Radio ve Switch Widget

Tanımladık

bool checkBox = false;

kullandık

CheckboxListTile(

value: checkBox,

onChanged: (secildi) {

setState(

() {

checkBox = secildi;

},

);

},

activeColor: Colors.teal,

title: Text("Are you student?"),

secondary: Icon(Icons.school),

)

radioButton:

RadioListTile<String>(

value: "Ankara",

groupValue: sehir,

onChanged: (deger) {

setState(() {

sehir = deger;

debugPrint("$deger");

});

},

title: Text("Ankara"),

),

RadioListTile<String>(

value: "Istanbul",

groupValue: sehir,

onChanged: (deger) {

setState(() {

sehir = deger;

debugPrint("Istanbul");

});

},

title: Text("$sehir")),

switchButton:

bool switchState = false;

SwitchListTile(

value: switchState,

onChanged: (deger) {

setState(() {

switchState = deger;

}

);

}, title: Text("data"),

)

Slider:

double sliderDeger = 0;

Slider(value: sliderDeger, onChanged: (deger){

setState(() {

sliderDeger = deger;

});

},

min: 0, max: 100, divisions: 10, label: sliderDeger.toString(),

)

dropdownbutton

DropdownButton<String>(items: [

DropdownMenuItem<String>(child: Text("data")),

DropdownMenuItem<String>(child: Text("data")),

DropdownMenuItem<String>(child: Text("data"))

], onChanged: (secilen) {},

hint: Text("data"),

)

Tarih ve saat seçme

DateTime suan = DateTime.now();

DateTime ucAyOncesi = DateTime(2020, suan.month - 3);

DateTime yirmiGunSonra = DateTime(2020,1, suan.day + 20);

showDatePicker(

context: context,

initialDate: suan,

firstDate: ucAyOncesi,

lastDate: yirmiGunSonra)

.then((secilenTarih) {

debugPrint(secilenTarih.toString());

});

TimeOfDay suankiSaat = TimeOfDay.now();

showTimePicker(

context: context, initialTime: suankiSaat)

.then((secilenSaat) {

debugPrint(secilenSaat.format(context));

});

Stepper Widget

Adım adım form yazma işlemi

import 'package:flutter/material.dart';

void main() {

runApp(MaterialApp(home: Deneme()));

}

class Deneme extends StatefulWidget {

@override

\_DenemeState createState() => \_DenemeState();

}

class \_DenemeState extends State<Deneme> {

int \_aktifStep = 0;

String ad, soyad, mail;

List<Step> tumStepler;

@override

void initState() {

// TODO: implement initState

super.initState();

tumStepler = \_tumStepler();

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text("form"),

),

body: SingleChildScrollView(

child: Stepper(

steps: tumStepler,

currentStep: \_aktifStep,

onStepTapped: (tiklanilanStep) {

setState(() {

\_aktifStep = tiklanilanStep;

});

},

onStepContinue: () {

setState(() {

{

\_aktifStep++;

}

});

},

onStepCancel: () {

setState(() {

\_aktifStep--;

});

})),

);

}

List<Step> \_tumStepler() {

List<Step> stepler = [

Step(

title: Text("ad"),

state: StepState.indexed,

isActive: true,

content: TextFormField(

decoration: InputDecoration(labelText: "ad"),

)),

Step(

title: Text("soyad"),

state: StepState.indexed,

isActive: true,

content: TextFormField(

decoration: InputDecoration(labelText: "soyad"),

)),

Step(

title: Text("mail"),

state: StepState.indexed,

isActive: true,

content: TextFormField(

decoration: InputDecoration(labelText: "mail"),

))

];

return stepler;

}

}

**JSON**

* **JSON** (JavaScript Object Notation– JavaScript Nesne Notasyonu) insanlar için okunabilir olan bilgi saklama ve alışveriş formatıdır.
* Bir JSON dosyası sadece metin kapsar ve **.json**uzantısını kullanır.
* JSON yapılı veri depolayan ve genellikle bir sunucu ve istemci arasında veri alışverişi için kullanılan bir formattır.
* JSON nedir öğrendiğinize göre artık JSON’ın nasıl çalıştığına geçebiliriz. Bir **.json**dosyası oluşturmak için doğru sözdizimini takip etmeniz gerekir.
* Bir JSON nesnesinin iki adet çekirdek elementi mevcuttur: Anahtarlar ve Değerler.
* **Anahtarlar** diziler olmalıdır. Tırnak işaretleriyle çevirili karakterler dizisinden oluşurlar.
* **Değerler** geçerli bir JSON veri tipidir. Dizin, nesne, dizi, boole değeri, rakam veya boşluk şeklinde olabilirler.
* Bir JSON nesnesi **{}**küme parantezleriyle başlar ve biter. İçinde iki veya daha fazla **anahtar/değer**çiftlerine sahip olabilir ve bu ikisini bir **virgül**ile ayırır. Bu sırada, her bir anahtardan sonra değerden ayrıştırmak için iki nokta üst üste gelir.
* İşte bir örnek:

{

"book": [

{

"id": "01",

"language": "Java",

"edition": "third",

"author": "Herbert Schildt"

},

{

"id": "07",

"language": "C++",

"edition": "second",

"author": "E.Balagurusamy"

}

]

}

**API**

API (Application Programming Interface/Uygulama Programlama Arayüzü), JSON formatında veriler sağlayan veri kaynağıdır.

Araba.json

[

{

"araba adi" : "mercedes",

"ulke adi " : "almanya",

"model" : [

{

"modeladi" : "A serisi",

"fiyat" : "3737774"

},

{

"modeladi" : "C serisi",

"fiyat" : "3737774"

}

]

},

{

"araba adi" : "honda",

"ulke adi " : "japonya",

"model" : [

{

"modeladi" : "A serisi",

"fiyat" : "3737774"

},

{

"modeladi" : "C serisi",

"fiyat" : "3737774"

}

]

}

]

Pubspec.yaml

assets:

- lib/araba.json

Main.dart

import 'dart:convert';

import 'package:flutter/material.dart';

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

// This widget is the root of your application.

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'Flutter Demo',

theme: ThemeData(

primarySwatch: Colors.blue,

visualDensity: VisualDensity.adaptivePlatformDensity,

),

home: LocalJson(),

);

}

}

class LocalJson extends StatefulWidget {

@override

\_LocalJsonState createState() => \_LocalJsonState();

}

class \_LocalJsonState extends State<LocalJson> {

List tumArabalar = [];

@override

void initState() {

super.initState();

veriKaynaginiOku().then((gelenArabaListesi) {

setState(() {

tumArabalar = gelenArabaListesi;

});

});

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text("JSON"),

),

body: tumArabalar != null ? Container(

child: Center(child: ListView.builder(itemBuilder: (context, index) {

return ListTile(

title: Text(

tumArabalar[index]["araba adi"].toString(),

),

subtitle: Text(

tumArabalar[index]["ulke adi "].toString(),

));

}, itemCount: tumArabalar.length,)),

): Center(child: CircularProgressIndicator(),)

);

}

Future<List> veriKaynaginiOku() async {

/\* Future<String> jsonOku =

DefaultAssetBundle.of(context).loadString("lib/araba.json");

jsonOku.then((okunanJson) {

debugPrint(okunanJson);

return okunanJson;

});

\*/

var jsonOku =

await DefaultAssetBundle.of(context).loadString("lib/araba.json");

debugPrint(jsonOku);

List arabaListesi = json.decode(jsonOku.toString());

debugPrint(arabaListesi.length.toString());

for (int i = 0; i < arabaListesi.length - 1; i++) {

debugPrint(arabaListesi[i]["araba adi"].toString());

debugPrint(arabaListesi[i]["ulke adi "].toString());

}

return arabaListesi;

}

}

Farkli method

import 'dart:convert';

import 'package:flutter/material.dart';

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

// This widget is the root of your application.

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'Flutter Demo',

theme: ThemeData(

primarySwatch: Colors.blue,

visualDensity: VisualDensity.adaptivePlatformDensity,

),

home: LocalJson(),

);

}

}

class LocalJson extends StatefulWidget {

@override

\_LocalJsonState createState() => \_LocalJsonState();

}

class \_LocalJsonState extends State<LocalJson> {

List tumArabalar = [];

@override

void initState() {

super.initState();

veriKaynaginiOku().then((gelenArabaListesi) {

setState(() {

tumArabalar = gelenArabaListesi;

});

});

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text("JSON"),

),

body: Container(

child: FutureBuilder(

future: veriKaynaginiOku(),

builder: (context, sonuc) {

return ListView.builder(

itemBuilder: (context, index) {

if (sonuc.hasData) {

tumArabalar = sonuc.data;

return ListTile(

title: Text(

tumArabalar[index]["araba adi"].toString(),

),

subtitle: Text(

tumArabalar[index]["ulke adi "].toString(),

));

} else {

return Center(child: CircularProgressIndicator(),);

}

},

itemCount: tumArabalar.length);

}),

));

}

Future<List> veriKaynaginiOku() async {

var jsonOku =

await DefaultAssetBundle.of(context).loadString("lib/araba.json");

debugPrint(jsonOku);

List arabaListesi = json.decode(jsonOku.toString());

debugPrint(arabaListesi.length.toString());

for (int i = 0; i < arabaListesi.length - 1; i++) {

debugPrint(arabaListesi[i]["araba adi"].toString());

debugPrint(arabaListesi[i]["ulke adi "].toString());

}

return arabaListesi;

}

}

İnternetten veri çekmek

<https://pub.dev/packages/http/install> bu sayfadan import ve depend on it kismini aldik. Depend on iy in altindaki http: ^0.12.2 pubspec.yaml dosyasına cupertino\_icons un altina ekledik import kismini aldik import ettik

# The following adds the Cupertino Icons font to your application.

# Use with the CupertinoIcons class for iOS style icons.

cupertino\_icons: ^1.0.0

http: ^0.12.2http: ^0.12.2

<https://marketplace.visualstudio.com/items?itemName=hirantha.json-to-dart> bunu vscoda’a install edin ve API’yi kopyalayınca yapıştırıp command+shift+p yapın oraya json to dart yazıp ‘convert from selection’ seçin.

Veya

<https://app.quicktype.io> bu siteden

<https://stackoverflow.com/questions/53744989/flutter-http-package-does-not-exist>

main

import 'package:currencyconvertor/currencyList.dart';

import 'package:flutter/material.dart';

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'Flutter Demo',

debugShowCheckedModeBanner: false,

home: Currency(),

);

}

}

Currency

// To parse this JSON data, do

//

// final welcome = welcomeFromJson(jsonString);

import 'dart:convert';

Welcome welcomeFromJson(String str) => Welcome.fromJson(json.decode(str));

String welcomeToJson(Welcome data) => json.encode(data.toJson());

class Welcome {

Welcome({

this.success,

this.timestamp,

this.base,

this.date,

this.rates,

});

bool success;

int timestamp;

String base;

DateTime date;

Map<String, double> rates;

factory Welcome.fromJson(Map<String, dynamic> json) => Welcome(

success: json["success"],

timestamp: json["timestamp"],

base: json["base"],

date: DateTime.parse(json["date"]),

rates: Map.from(json["rates"])

.map((k, v) => MapEntry<String, double>(k, v.toDouble())),

);

Map<String, dynamic> toJson() => {

"success": success,

"timestamp": timestamp,

"base": base,

"date":

"${date.year.toString().padLeft(4, '0')}-${date.month.toString().padLeft(2, '0')}-${date.day.toString().padLeft(2, '0')}",

"rates": Map.from(rates).map((k, v) => MapEntry<String, dynamic>(k, v)),

};

}

currencyList

import 'package:currencyconvertor/models/currency.dart';

import 'package:flutter/material.dart';

import 'dart:convert';

import 'package:http/http.dart' as http;

class Currency extends StatefulWidget {

@override

\_CurrencyState createState() => \_CurrencyState();

}

class \_CurrencyState extends State<Currency> {

var tryDegeri = 1.0;

List<String> kur = List(4);

List<DovizListem> dovizListesi = List<DovizListem>();

Welcome currency;

String url =

"http://data.fixer.io/api/latest?access\_key=37381f6be5bc25559393ce9dbb6b6159";

Future<Welcome> kurGetir() async {

var responce = await http.get(url);

var decodedJson = json.decode(responce.body);

currency = Welcome.fromJson(decodedJson);

return currency;

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

backgroundColor: Colors.black,

title: Text("Finansal Veriler", textAlign: TextAlign.center),

),

body: FutureBuilder(

future: kurGetir(),

builder: (context, AsyncSnapshot<Welcome> snapshot) {

if (snapshot.connectionState == ConnectionState.waiting) {

return Center(

child: CircularProgressIndicator(),

);

} else if (snapshot.connectionState == ConnectionState.done) {

snapshot.data.rates.forEach((key, value) {

dovizListesi

.add(DovizListem(paraAdi: key, paraDegeri: value));

});

}

//TL BASE

double tlBul() {

for (var i = 0; i < dovizListesi.length; i++) {

if (dovizListesi[i].paraAdi == "TRY") {

return dovizListesi[i].paraDegeri;

}

}

}

tryDegeri = tlBul();

//BELLI KURLAR

List istenilenKUR() {

for (var i = 0; i < dovizListesi.length; i++) {

if (dovizListesi[i].paraAdi == "USD") {

kur[0] =

(tryDegeri / dovizListesi[i].paraDegeri).toString();

} else if (dovizListesi[i].paraAdi == "EUR") {

kur[1] =

(tryDegeri / dovizListesi[i].paraDegeri).toString();

} else if (dovizListesi[i].paraAdi == "GBP") {

kur[2] =

(tryDegeri / dovizListesi[i].paraDegeri).toString();

} else if (dovizListesi[i].paraAdi == "CAD") {

kur[3] =

(tryDegeri / dovizListesi[i].paraDegeri).toString();

}

}

return kur;

}

istenilenKUR();

List<String> isim = ["USD", "EUR", "GBP", "CAD"];

return GridView.builder(

itemCount: kur.length,

gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(

crossAxisCount: 2),

itemBuilder: (context, index) {

return Card(

elevation: 6,

child: Container(

decoration: BoxDecoration(

// koselerini renklendiriyoruz

border: Border.all(

color: Colors.green,

width: 0.5,

style: BorderStyle.solid),

),

child: Column(

mainAxisAlignment: MainAxisAlignment.spaceEvenly,

children: <Widget>[

Text(

isim[index],

style: TextStyle(

color: Colors.green,

fontWeight: FontWeight.bold),

),

Text(kur[index]),

],

),

));

});

}));

}

}

class DovizListem {

DovizListem({this.paraAdi, this.paraDegeri});

String paraAdi;

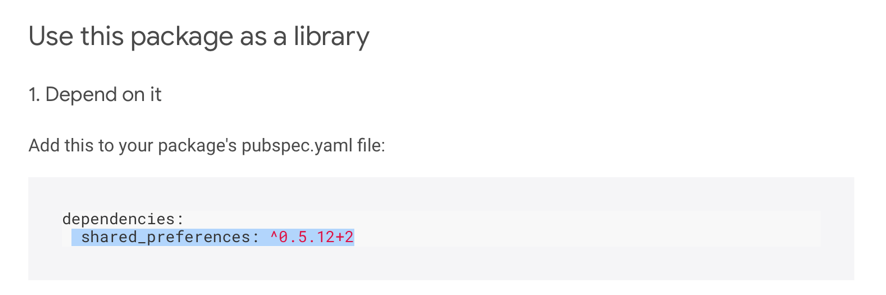
double paraDegeri;

}

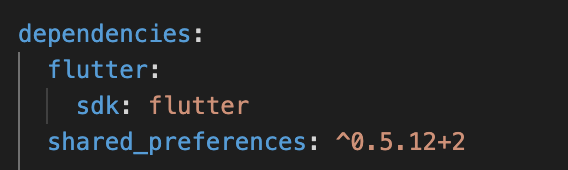
**Sharedpreferences Kullanımı**

1. **Paketi projeye ekle**

<https://pub.dev/packages/shared_preferences/install> sayfasına git install kısmına gir ve asagidaki resimde kopyalanan yeri kopyala



Projede pubspec.yaml dosyasına gidin ve dependencies kısmının altına, kopyaladığınız yeri yapıştırın.

****

import 'package:flutter/material.dart';

import 'package:shared\_preferences/shared\_preferences.dart';

class SharedPrefKullanimi extends StatefulWidget {

@override

\_SharedPrefKullanimiState createState() => \_SharedPrefKullanimiState();

}

class \_SharedPrefKullanimiState extends State<SharedPrefKullanimi> {

String isim;

int id;

bool cinsiyet;

var formKey=GlobalKey<FormState>();

var mySharedPrefences;

@override

void initState() {

// TODO: implement initState

super.initState();

// SharedPreferences.getInstance().then((sf)=>mySharedPrefences=sf);

SharedPreferences.getInstance().then((sf){

mySharedPrefences=sf;

});

}

@override

void dispose() {

mySharedPrefences.dispose();

super.dispose();

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text("Shared Pref Kullanımı"),

),

body: SingleChildScrollView(

child: Form(

key: formKey,

child: Column(

children: <Widget>[

Padding(

padding: const EdgeInsets.all(8.0),

child: TextFormField(

onSaved: (deger){

isim=deger;

},

decoration: InputDecoration(

labelText: "Isminizi giriniz",

border:

OutlineInputBorder(borderRadius: BorderRadius.circular(10)),

),

),

),

Padding(

padding: const EdgeInsets.all(8.0),

child: TextFormField(

onSaved: (deger){

id=int.parse(deger);

},

keyboardType: TextInputType.number,

decoration: InputDecoration(

labelText: "ID giriniz",

border:

OutlineInputBorder(borderRadius: BorderRadius.circular(10)),

),

),

),

Padding(

padding: const EdgeInsets.all(8.0),

child: RadioListTile(value: true, groupValue: cinsiyet, onChanged: (secildi){

setState(() {

cinsiyet = secildi;

});

}, title: Text("Erkek"),),

),

Padding(

padding: const EdgeInsets.all(8.0),

child: RadioListTile(value: false, groupValue: cinsiyet, onChanged: (secildi){

setState(() {

cinsiyet = secildi;

});

}, title: Text("Kadın"),),

),

Row(

mainAxisAlignment: MainAxisAlignment.spaceEvenly,

children: <Widget>[

RaisedButton(onPressed: \_ekle, child: Text("Kaydet"), color: Colors.green,),

RaisedButton(onPressed: \_goster, child: Text("Göster"), color: Colors.blue,),

RaisedButton(onPressed: \_sil, child: Text("Sil"), color: Colors.red,),

],

)

],

)),

),

);

}

void \_ekle() async{

formKey.currentState.save();

await (mySharedPrefences as SharedPreferences).setString("myIsim", isim);

await (mySharedPrefences as SharedPreferences).setInt("myId", id);

await (mySharedPrefences as SharedPreferences).setBool("myCinsiyet", cinsiyet);

}

void \_goster() {

debugPrint("Okunan isim "+(mySharedPrefences as SharedPreferences).getString("myIsim")?? "NULLLL");

debugPrint("Okunan id "+(mySharedPrefences as SharedPreferences).getInt("myId").toString() ?? "NULL");

debugPrint("Cinsiyet erkek mi "+(mySharedPrefences as SharedPreferences).getBool("myCinsiyet").toString() ?? "NULLL");

}

void \_sil() {

(mySharedPrefences as SharedPreferences).remove("myIsim");

(mySharedPrefences as SharedPreferences).remove("myId");

(mySharedPrefences as SharedPreferences).remove("myCinsiyet");

(mySharedPrefences as SharedPreferences).remove("sil");

(mySharedPrefences as SharedPreferences).remove("telNo");

}

}

**SQFLite Kullanımı ve Kalıcı Depolama**

**!** Hem android hem IOS’ta düzgün bir şekilde çalışır

-Uygulama için veri tabanı oluşturur.

-Çok büyük olmayan veriler uygulamanın yüklü olduğu cihazın belleğinde veri tabanı dosyası şeklinde saklanır.

-Bir veri tabanında olan veri ekleme, silme, güncellene gibi işlemler yapılabilir

1. **Paketi projeye ekle**

* Paket adresi: <https://pub.dev/packages/sqflite>, https://pub.dev/packages/path\_provider/install sayfalarına git install kısmına gir ve aşağıdaki resimde kopyalanan yeri kopyala

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

* Projede pubspec.yaml dosyasına gidin ve dependencies kısmının altına,sqflite ve path\_provider paketlerinin güncel versiyonlarını ekleyin.

**metin içeren bir resim

Açıklama otomatik olarak oluşturuldu**

1. **Model sınıfı oluşturuyoruz.**

* Model bizim veritabanımızın kodsal karşlığı olacak
* New folder deyip models klasörünü oluşturuyoruz onun altına new file diyerek product.dart dosyasını oluşturduk.

class Product {

int \_id;

String \_name;

String \_description;

double \_price;

//constructor olusturmamiz gerek

Product(this.\_description, this.\_name, this.\_price) {}

Product.withId(this.\_id, this.\_description, this.\_name, this.\_price);

//getter ve setter olusturalim

int get\_id() {

return this.\_id;

}

set\_id(int \_id) {

this.\_id = \_id;

}

String get\_name() {

return this.\_name;

}

set\_name(String \_name) {

this.\_name = \_name;

}

String get\_description() {

return this.\_description;

}

set\_description(String \_description) {

this.\_description = \_description;

}

double get\_price() {

return this.\_price;

}

set\_price(double \_price) {

this.\_price = \_price;

}

//elimizdeki bir nesneyi map'e cevirecek

Map<String, dynamic> toMao() {

var map = Map<String, dynamic>();

map["name"] = \_name;

map["description"] = \_description;

map["price"] = \_price;

if (["id"] != null) {

map["id"] = \_id;

}

return map;

}

//gelen datayi nesneye çeviriyor datanin turunu bilmedigimiz icin dynamic kabul ediyoruz

Product.fromObject(dynamic o) {

this.\_id = o["id"];

this.\_name = o["name"];

this.\_description = o["description"];

this.\_price = o["price"];

}

}

1. **Dbhelper oluşturuyoruz**

* New folder deyip db klasörünü oluşturuyoruz onun altına new file diyerek dbhelper.dart dosyasını oluşturduk.
* İmport edilecekler
* import 'package:sqflite/sqflite.dart';
* import 'dart:async';
* import 'dart:io';
* import 'package:path\_provider/path\_provider.dart';

class DbHelper {

//tablo ve kolon isimlerini degiskenlerde tutacagiz

String tblProduct = "product";

String colomId = "id";

String colomDescription = "description";

String colomPrice = "price";

String colomName = "name";

//static yapmak icin yaptik

DbHelper.\_internal();

//DbHelper.\_internal'dan nesne olusturduk

static final DbHelper \_dbHelper = DbHelper.\_internal();

//factory bize dbHelper i bir fonk vasitasi ile cagirriyoruz. Her seferinde ayni db gormek icin

factory DbHelper() {

return \_dbHelper;

}

//dbhelper vasitasi ile CRUD islemlerini yazacagiz ilk veritabanina ulasmamiz gerek

//sqflite importu ile bize saglanan Database sinifindan obje olusturduk

static Database \_db;

Future<Database> get Db async {

if (\_db != null) {

return Db;

}

\_db = await initializeDb();

return \_db;

}

Future<Database> initializeDb() async {

//database' imizin oldugu dosyaya girmemizi path provider sagliyor

Directory directory = await getApplicationDocumentsDirectory();

String path = directory.path + "etrade.db";

var dbetrade = await openDatabase(path, version: 1, onCreate: \_createDb);

return dbetrade;

}

void \_createDb(Database db, int version) async {

await db.execute(

"create table $tblProduct($colomId integer primarykey, $colomName text, $colomDescription text, $colomPrice int)");

}

//CRUD islemleri

Future<int> insert(Product product) async {

Database db = await this.Db;

var result = await db.insert(tblProduct, product.toMap());

return result;

}

Future<int> update(Product product) async {

Database db = await this.Db;

var result = await db.update(tblProduct, product.toMap(),

where: "$colomId = ?", whereArgs: [product.get\_id()]);

return result;

}

Future<int> delete(int id) async {

Database db = await this.Db;

var result = await db.rawDelete("delete from $tblProduct where $colomId = $id ")

}

Future<List> GetProducts() async{

Database db = await this.Db;

var result =await db.rawQuery("select \* from $tblProduct");

return result;

}

}

1. dbhelper i uygulamada kullanmak
2. import 'package:flutter/material.dart';
3. import 'package:sqlislemelri/db/dbhelper.dart';
4. import 'package:sqlislemelri/models/product.dart';
5. class ProductList extends StatefulWidget {
6. @override
7. \_ProductListState createState() => \_ProductListState();
8. }
9. class \_ProductListState extends State<ProductList> {
10. //veritabanina urun ekleme
11. DbHelper dbHelper = new DbHelper();
12. List<Product> products;
13. int count = 0;
14. @override
15. Widget build(BuildContext context) {
16. return Scaffold(
17. appBar: AppBar(
18. title: Text("PRODUCT LIST"),
19. ),
20. body: productListItems(),
21. );
22. }
23. productListItems() {
24. getData();
25. return ListView.builder(
26. itemBuilder: (BuildContext context, int index) {
27. return Card(
28. child: ListTile(
29. title: Text(this.products[index].get\_name()),
30. subtitle: Text(this.products[index].get\_description()),
31. ),
32. );
33. },
34. itemCount: count.bitLength,
35. );
36. }
37. void getData() {
38. //db yok ise 0 dan veritabani olusturur veya var ise onu getirir
39. var db = dbHelper.initializeDb();
40. db.then((result) {
41. var productsFuture = dbHelper.GetProducts();
42. productsFuture.then((data) {
43. List<Product> productData = new List<Product>();
44. count = data.length;
45. for (int i = 0; i < count; i++) {
46. productData.add(Product.fromObject(data[i]));
47. }
48. setState(() {
49. products = productData;
50. count = count;
51. });
52. });
53. });
54. }
55. }

**Main**

import 'package:flutter/material.dart';

import 'package:sqlislemelri/screens/productlist.dart';

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'Flutter Demo',

theme: ThemeData(

primarySwatch: Colors.blue,

visualDensity: VisualDensity.adaptivePlatformDensity,

),

home: ProductList(),

);

}

}

**Firebase**

* Firebase: JSON verilerimizi uzak bir sunucuda gerçek zamanlı olarak depolayabildiğimiz, bulut (Cloud) tabanlı bir platformdur
* Firebase platformu 2011 senesinde iki startup tarafından kuruldu. O dönemki ilk adı ise Envolve idi. İlk çıktığında asıl amaç, uygulama üreten yazılımcıların zahmetsiz şekilde chat uygulamasını ekleyebilmeleriydi. Bu uygulamayı API şeklinde tanımlayabiliriz. Buna rağmen tam anlamıyla bu noktayla tatmin olmadılar. Çünkü asıl ihtiyaç olan şey chat uygulaması değildi, bunu fark etmiş oldular.
* Bu nedenle hemen bir sene sonra Firebase isminde yepyeni bir firma kurdular ve burada ağırlıklı olarak data senkronu üzerine hizmet verdiler.
* 2014 senesinde Google, bu hizmeti iki girişimciden satın aldı. Asıl büyüme hikayesi de tam olarak bu noktada başladı. Server-Side dediğimiz arka planda ki verilerin tutulması ve gerektiği zaman kullanıcıya kullanılması her platformun ortak sorunudur ve Google Firebase bu konuda geliştirilmiş ortak bir çözümdür.
* Günümüzde uygulama yapanların kurtarıcısı olan Google Firebase, pek çok özelliğe sahiptir.
* Eşzamanlı veri tabanı
* Kullanıcı girişlerinde yetkilendirme
* Depolama
* Makine öğrenme kiti
* Performans test ortamı
* Platform arası ortak biçimde uygulama analizleri
* Ortak fonksiyon
* Reklam materyalleri
* Bildirim gönderimi
* Genel uygulama yönetimi
* Google Firebase uygulama geliştirme noktasında çok sayıda ürünü ve hizmeti bir arada sunuyor. Bu hizmetlerden ve ürünlerden bazıları şu şekilde:
* **Realtime Database**: Veri tutmaya yarayan büyük bir veri tabanı uygulaması diyebiliriz. Temel amacı anlık olarak verileri senkronize etmektir.
* **Authentication**: Bu serviste işlem yapmak adına giriş yapmanız şarttır.
* **Test Lab**: Ürettiğiniz ya da tasarım aşamasında olduğunuz mobil uygulamalarla ilgili size ön bilgiler sunar. Uygulamada denediğiniz her şeyi hızlıca test edebilirsiniz.
* **Crashlytic**: Anlık olarak loglama yapmanızı sağlar. Ayrıca hatalarınızı anında deşifre eder ve bu sayede güncellemeler yapmanıza olanak sağlar.
* **Firestore**: Anlık senkronizasyonun daha ölçeklenebilir halidir. Realtime Database ile sık sık karşılaştırılır. Günümüzde beta sürümü olarak hizmet verir.

**Firebase ile proje**

1. **Main içine basit form işlemleri ekledik**
2. import 'package:flutter/material.dart';
3. import 'package:flutter/services.dart';
4. void main() {
5. runApp(MaterialApp(
6. debugShowCheckedModeBanner: false,
7. theme: ThemeData(
8. primarySwatch: Colors.blue,
9. visualDensity: VisualDensity.adaptivePlatformDensity,
10. ),
11. home: MyApp(),
12. ));
13. }
14. class MyApp extends StatefulWidget {
15. @override
16. \_MyAppState createState() => \_MyAppState();
17. }
18. class \_MyAppState extends State<MyApp> {
19. String ad, kategori, id;
20. double fiyat;
21. @override
22. Widget build(BuildContext context) {
23. return Scaffold(
24. appBar: AppBar(
25. title: Text("Urun "),
26. ),
27. body: Column(
28. children: [
29. Padding(
30. padding: const EdgeInsets.all(8.0),
31. child: TextFormField(
32. decoration: InputDecoration(
33. labelText: "urun id",
34. fillColor: Colors.white,
35. focusedBorder: OutlineInputBorder(
36. borderSide: BorderSide(color: Colors.blue, width: 2.0))),
37. onChanged: (String urunId) {
38. urunIdAl(urunId);
39. },
40. ),
41. ),
42. Padding(
43. padding: const EdgeInsets.all(8.0),
44. child: TextFormField(
45. decoration: InputDecoration(
46. labelText: "urun fiyat",
47. fillColor: Colors.white,
48. focusedBorder: OutlineInputBorder(
49. borderSide: BorderSide(color: Colors.blue, width: 2.0))),
50. onChanged: (String urunFiyati) {
51. urunFiyatiAl(urunFiyati);
52. },
53. ),
54. ),
55. Padding(
56. padding: const EdgeInsets.all(8.0),
57. child: TextFormField(
58. decoration: InputDecoration(
59. labelText: "urun kategorisi",
60. fillColor: Colors.white,
61. focusedBorder: OutlineInputBorder(
62. borderSide: BorderSide(color: Colors.blue, width: 2.0))),
63. onChanged: (String urunKategorisi) {
64. urunKategorisiAl(urunKategorisi);
65. },
66. ),
67. ),
68. Padding(
69. padding: const EdgeInsets.all(8.0),
70. child: TextFormField(
71. decoration: InputDecoration(
72. labelText: "urun adi",
73. fillColor: Colors.white,
74. focusedBorder: OutlineInputBorder(
75. borderSide: BorderSide(color: Colors.blue, width: 2.0))),
76. onChanged: (String urunAdi) {
77. urunAdiAl(urunAdi);
78. },
79. ),
80. ),
81. Padding(
82. padding: const EdgeInsets.only(top: 8.0),
83. child: Row(
84. mainAxisAlignment: MainAxisAlignment.spaceEvenly,
85. children: [
86. RaisedButton(
87. onPressed: () {
88. veriOku();
89. },
90. color: Colors.green,
91. shape: RoundedRectangleBorder(
92. borderRadius: BorderRadius.circular(16)),
93. child: Text("oku"),
94. ),
95. RaisedButton(
96. onPressed: () {
97. veriEkle();
98. },
99. color: Colors.cyan,
100. shape: RoundedRectangleBorder(
101. borderRadius: BorderRadius.circular(16)),
102. child: Text("ekle"),
103. ),
104. RaisedButton(
105. onPressed: () {
106. veriGuncelle();
107. },
108. color: Colors.red,
109. shape: RoundedRectangleBorder(
110. borderRadius: BorderRadius.circular(16)),
111. child: Text("güncelle"),
112. ),
113. RaisedButton(
114. onPressed: () {
115. veriSil();
116. },
117. color: Colors.yellow,
118. shape: RoundedRectangleBorder(
119. borderRadius: BorderRadius.circular(16)),
120. child: Text("sil"),
121. )
122. ],
123. ),
124. )
125. ],
126. ),
127. );
128. }
129. void urunIdAl(urunId) {
130. this.id = urunId;
131. }
132. void urunFiyatiAl(urunFiyati) {
133. this.fiyat = double.parse(urunFiyati);
134. }
135. void urunKategorisiAl(urunKategorisi) {
136. this.kategori = urunKategorisi;
137. }
138. void urunAdiAl(urunAdi) {
139. this.ad = urunAdi;
140. }
141. veriEkle() {
142. print("eklendi");
143. }
144. veriGuncelle() {
145. print("guncellendi");
146. }
147. veriOku() {
148. print("okundu");
149. }
150. veriSil() {
151. print("silindi");
152. }
153. }