

# Jetpack Compose ile Android Uygulama Geliştirme Kursu

## Widgets

Kasım ADALAN

Elektronik ve Haberleşme Mühendisi

Android - IOS Developer and Trainer

# Eğitim İçeriği

- Text
- Button
- TextField
- FloatingActionButton
- Switch
- CheckBox
- GestureDetector
- RadioButton
- ProgressIndicator
- Slider
- WebView
- Image
- DropdownMenu

# Text

- Metinsel ifadeleri kullanıcıya gösterdiğimiz görsel nesnedir.
- Button üzerindeki yazıda dahil.

Gelen Veri :

```
@SuppressLint("UnusedMaterial3ScaffoldPaddingParameter")
@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun SayfaButtonTextTextfield() {
    val tf = remember { mutableStateOf(value: "") }
    val alınanVeri = remember { mutableStateOf(value: "") }

    Column(
        modifier = Modifier.fillMaxSize(),
        verticalArrangement = Arrangement.SpaceEvenly,
        horizontalAlignment = Alignment.CenterHorizontally
    ) { this: ColumnScope
        Text(text = "Gelen Veri : ${alınanVeri.value}",)

        TextField(
            value = tf.value,
            onValueChange = { tf.value = it },
            label = { Text(text = "Veri giriniz")},)

        Button(onClick = {
            alınanVeri.value = tf.value
        }) { this: RowScope
            Text(text = "Veriyi Al")
        }
    }
}
```

Veri giriniz

Veriyi Al

# Text Özelleştirme

```
Text(
```

```
    text = "Gelen Veri : ${alinanVeri.value}",  
    color = Color.White,//Yazı Rengi  
    fontSize = 30.sp,//Yazı boyutu  
    fontWeight = FontWeight.Bold,//Yazı kalınlığı  
    style = TextStyle(  
        background = Color.Blue//Arkaplan rengi  
    )  
)
```

Gelen Veri :

Veriyi Al

# Button

- Bu görsel nesne ile tıklama işlemi yapabiliriz.

```
@SuppressLint("UnusedMaterial3ScaffoldPaddingParameter")
@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun SayfaButtonTextField() {
    val tf = remember { mutableStateOf(value: "") }
    val alinanVeri = remember { mutableStateOf(value: "") }

    Column(
        modifier = Modifier.fillMaxSize(),
        verticalArrangement = Arrangement.SpaceEvenly,
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        this: ColumnScope
        Text(text = "Gelen Veri : ${alinanVeri.value},")

        TextField(
            value = tf.value,
            onValueChange = { tf.value = it },
            label = { Text(text = "Veri giriniz") })

        Button(onClick = {
            alinanVeri.value = tf.value
        }) {
            this: RowScope
            Text(text = "Veriyi Al")
        }
    }
}
```

Gelen Veri :

Veri giriniz

Veriyi Al



# OutlinedButton

- Klasik button ile aynı işlevi vardır sadece görsel farklıdır.

Gelen Veri :

Veri giriniz

Veriyi Al

```
OutlinedButton(onClick = {  
    alinanVeri.value = tf.value  
}) { this: RowScope  
    Text(text = "Veriyi Al")  
}
```

# Button Özelleştirme

```
Button(onClick = {  
    alinanVeri.value = tf.value  
},  
    colors = ButtonDefaults.buttonColors(  
        containerColor = Color.Red, //Arkaplan rengi  
        contentColor = Color.White //Yazı rengi  
    ),  
    border = BorderStroke(1.dp, Color.Black), //Sınır kalınlığı ve rengi  
    shape = RoundedCornerShape(percent: 50) //Köşe kıvrım miktarı %50  
) { this: RowScope  
    Text(text = "Veriyi Al")  
}
```

Gelen Veri :

Veri giriniz

Veriyi Al

# OutlinedButton Özelleştirme

```
OutlinedButton(onClick = {  
    alinanVeri.value = tf.value  
},  
    colors = ButtonDefaults.buttonColors(  
        containerColor = Color.Red, //Arkaplan rengi  
        contentColor = Color.White //Yazı rengi  
    ),  
    border = BorderStroke(1.dp, Color.Black), //Sınır kalınlığı ve rengi  
    shape = RoundedCornerShape(percent: 50) //Köşe kıvrım miktarı %50  
) { this: RowScope  
    Text(text = "Veriyi Al")  
}
```

Gelen Veri :

Veri giriniz

Veriyi Al

# TextField

- TextField ile uygulamaya kullanıcı veri girebilmektedir.
- TextField üzerine girilen veriyi state özelliği olan değişken ile kontrol etmekteyiz.
- Bu değişken sayesinde girilen veriyi okuyabiliyoruz ve girilen veriyi değiştirebiliriz.

```

@Suppress("UnusedMaterial3ScaffoldPaddingParameter")
@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun SayfaButtonTextTextfield() {
    val tf = remember { mutableStateOf(value: "") }
    val alınanVeri = remember { mutableStateOf(value: "") }

    Column(
        modifier = Modifier.fillMaxSize(),
        verticalArrangement = Arrangement.SpaceEvenly,
        horizontalAlignment = Alignment.CenterHorizontally
    ) { this: ColumnScope
        Text(text = "Gelen Veri : ${alınanVeri.value},")

        TextField(
            value = tf.value,
            onValueChange = { tf.value = it },
            label = { Text(text = "Veri giriniz") })
    }
}

```

Gelen Veri :

Veri giriniz

Veriyi Al

*State değişkeni bağlama*

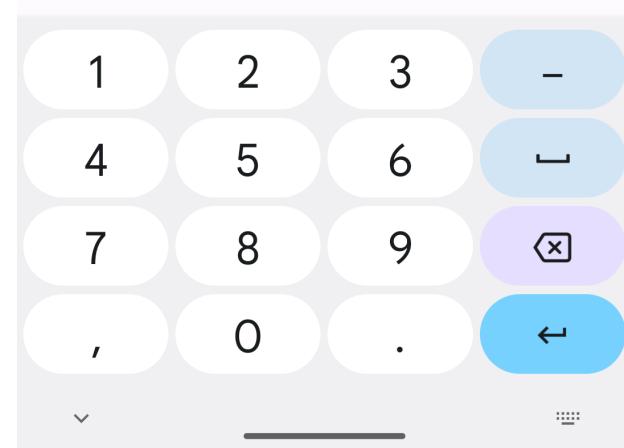
*State değişkene  
textField içine girilen  
veriyi aktarma*

# TextField Özelleştirme

```
TextField(  
    value = tf.value,  
    onValueChange = { tf.value = it},  
    label = { Text(text = "Veri giriniz")},  
    colors = TextFieldDefaults.textFieldColors(  
        containerColor = Color.Gray,//Arkaplan Rengi  
        textColor = Color.Red,//Yazı rengi  
        focusedIndicatorColor = Color.Green,//Belirteç Rengi  
        focusedLabelColor = Color.Yellow//Label Rengi  
    ),  
    visualTransformation = PasswordVisualTransformation(),//Yazıyı Gizleme  
    keyboardOptions = KeyboardOptions(keyboardType = KeyboardType.Number)//Klavye türü  
)
```

Gelen Veri :

Veri giriniz  
...|





# OutlinedTextField

- TextField ile sadece şeiksel farklılıkları vardır.

```
@SuppressLint("UnusedMaterial3ScaffoldPaddingParameter")
@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun SayfaButtonTextTextfield() {
    val tf = remember { mutableStateOf(value: "") }
    val alinanVeri = remember { mutableStateOf(value: "") }

    Column(
        modifier = Modifier.fillMaxSize(),
        verticalArrangement = Arrangement.SpaceEvenly,
        horizontalAlignment = Alignment.CenterHorizontally
    ) { this: ColumnScope
        Text(text = "Gelen Veri : ${alinanVeri.value},")

        OutlinedTextField(
            value = tf.value,
            onValueChange = { tf.value = it },
            label = { Text(text = "Veri giriniz") },)

        Button(onClick = {
            alinanVeri.value = tf.value
        }) { this: RowScope
            Text(text = "Veriyi Al")
        }
    }
}
```

Gelen Veri :

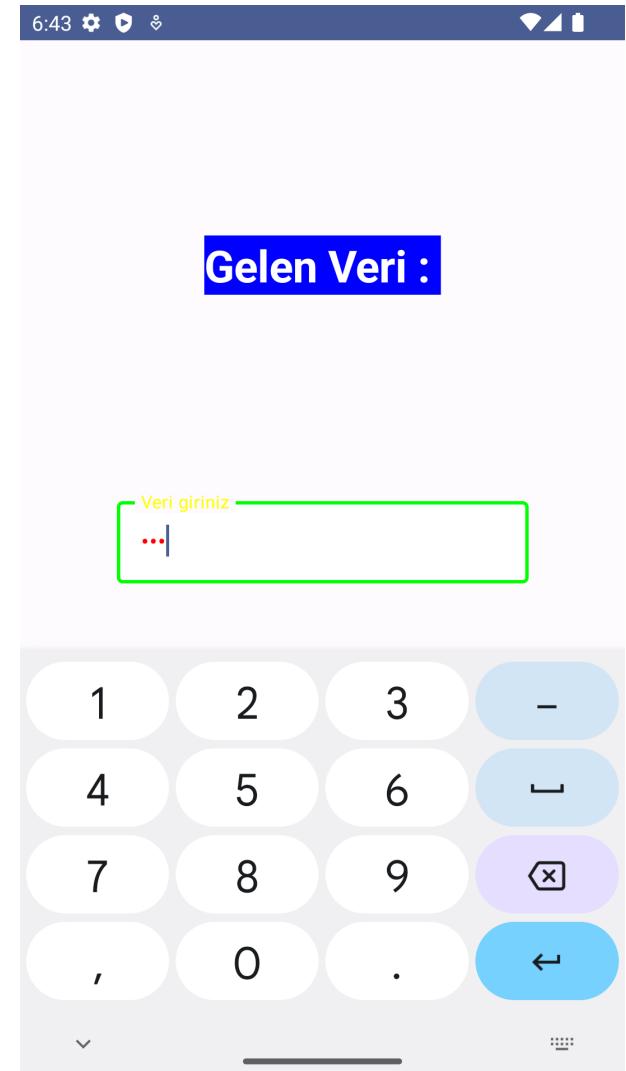
Veri giriniz

merhaba

Veriyi Al

# OutlinedTextField Özelleştirme

```
OutlinedTextField(  
    value = tf.value,  
    onValueChange = { tf.value = it},  
    label = { Text(text = "Veri giriniz")},  
    colors = TextFieldDefaults.outlinedTextFieldColors(  
        containerColor = Color.White,//Arkaplan Rengi  
        textColor = Color.Red,//Yazı rengi  
        focusedBorderColor = Color.Green,//Sınır Rengi  
        focusedLabelColor = Color.Yellow//Label Rengi  
    ),  
    visualTransformation = PasswordVisualTransformation(),//Yazıyı Gizleme  
    keyboardOptions = KeyboardOptions(keyboardType = KeyboardType.Number)//Klavye türü  
)
```



# FloatingActionButton

- Normal bir button gibi kullanılabilir ama genelde scaffold içinde kullanılır ve ekrana duyarlı çalışır.
- Altında liste varsa listeyi yukarı aşağı kaydırırsakta sayfada sabit kalır.

```
@SuppressLint("UnusedMaterial3ScaffoldPaddingParameter")
@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun SayfaFab() {
    Scaffold(
        content = { it: PaddingValues ->
            //Sayfa içeriği kodlaması
        },
        floatingActionButton = {
            FloatingActionButton(
                onClick = {
                    Log.e("Fab", "Tıklındı")
                },
                containerColor = Color.Red,
                content = {
                    Icon(
                        painter = painterResource(id = R.drawable.ekle_resim),
                        contentDescription = "", tint = Color.White)
                }
            )
        }
    )
}
```



# ExtendedFloatingActionButton

```
@SuppressLint("UnusedMaterial3ScaffoldPaddingParameter")
@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun SayfaFab() {
    Scaffold(
        content = { it: PaddingValues
            //Sayfa içeriği kodlaması
        },
        floatingActionButton = {
            ExtendedFloatingActionButton(
                onClick = {
                    Log.e( tag: "Fab", msg: "Tıkladı")
                },
                text = { Text(text = "EKLE", color = Color.White)},
                containerColor = Color.Red,
                icon = {
                    Icon(
                        painter = painterResource(id = R.drawable.ekle_resim),
                        contentDescription = "", tint = Color.White)
                }
            )
        }
    )
}
```



+ EKLE

# Switch

- Çift konumlu bir butondur.

```
@Composable
fun SayfaSwitch() {
    val switchDurum = remember { mutableStateOf( value: false ) }
    Column(
        modifier = Modifier.fillMaxSize(),
        verticalArrangement = Arrangement.SpaceEvenly,
        horizontalAlignment = Alignment.CenterHorizontally
    ) { this: ColumnScope
        Switch(
            checked = switchDurum.value,
            onCheckedChange ={ it: Boolean
                switchDurum.value = it
                Log.e( tag: "Switch seçildi",it.toString())
            },
            colors = SwitchDefaults.colors(
                checkedTrackColor = Color.Red,//True durumu yuvarlak kısmı
                checkedThumbColor = Color.Blue,//True durumu çubuk kısmı
                uncheckedTrackColor = Color.Green,//False durumu yuvarlak kısmı
                uncheckedThumbColor = Color.Black//False durumu çubuk kısmı
            )
        )
        Button(onClick = {
            Log.e( tag: "Switch en son durum",switchDurum.value.toString())
        }) { this: RowScope
            Text(text = "Göster")
        }
    }
}
```



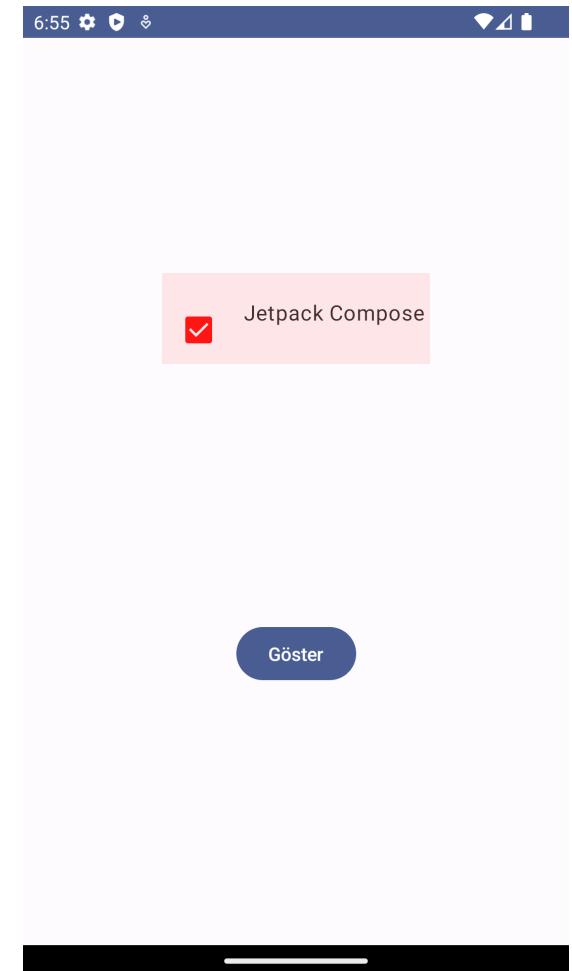
Göster

DALAN

# CheckBox

- Çoklu seçim yaparken kullanılan widgettir.

```
@Composable
fun SayfaCheckBox() {
    val checkboxDurum = remember { mutableStateOf( value: false ) }
    Column(
        modifier = Modifier.fillMaxSize(),
        verticalArrangement = Arrangement.SpaceEvenly,
        horizontalAlignment = Alignment.CenterHorizontally
    ) { this: ColumnScope
        Row { this: RowScope
            Checkbox(
                checked = checkboxDurum.value,
                onCheckedChange = { it: Boolean
                    checkboxDurum.value = it
                    Log.e( tag: "Checkbox seçildi",it.toString() )
                },
                colors = CheckboxDefaults.colors(
                    checkedColor = Color.Red
                )
            )
            Text(text = "Jetpack Compose",modifier = Modifier.padding(start = 10.dp))
        }
        Button(onClick = {
            Log.e( tag: "Checkbox en son durum",checkboxDurum.value.toString() )
        }) { this: RowScope
            Text(text = "Göster")
        }
    }
}
```

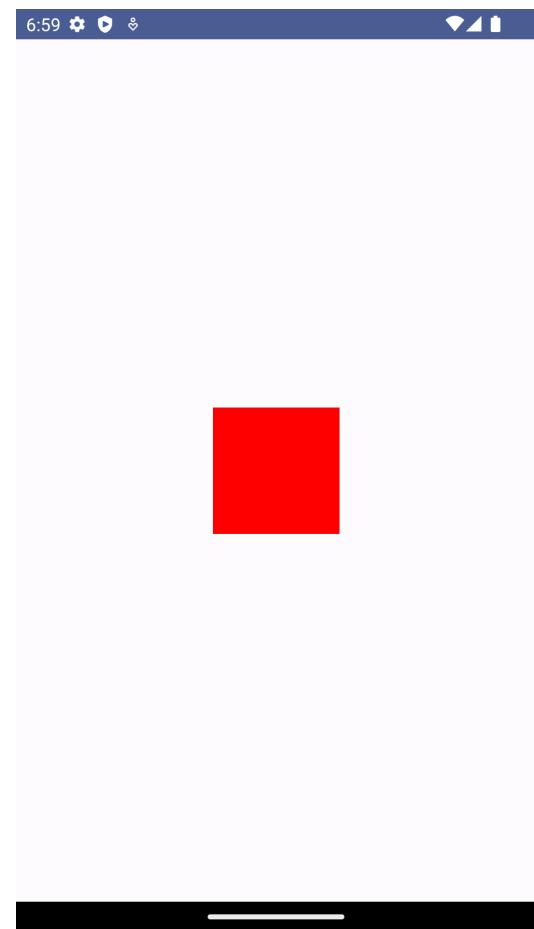


# Clickable

- İstenilen widgeta pratik olarak tıklanılma özelliği verebiliriz.

```
@Composable
fun Sayfa() {
    Column(
        modifier = Modifier.fillMaxSize(),
        verticalArrangement = Arrangement.SpaceEvenly,
        horizontalAlignment = Alignment.CenterHorizontally
    ) { this: ColumnScope

        Box(Modifier.size(100.dp).background(Color.Red)
            .clickable {
                Log.e( tag: "Box", msg: "Tıklandı")
            }
        )
    }
}
```

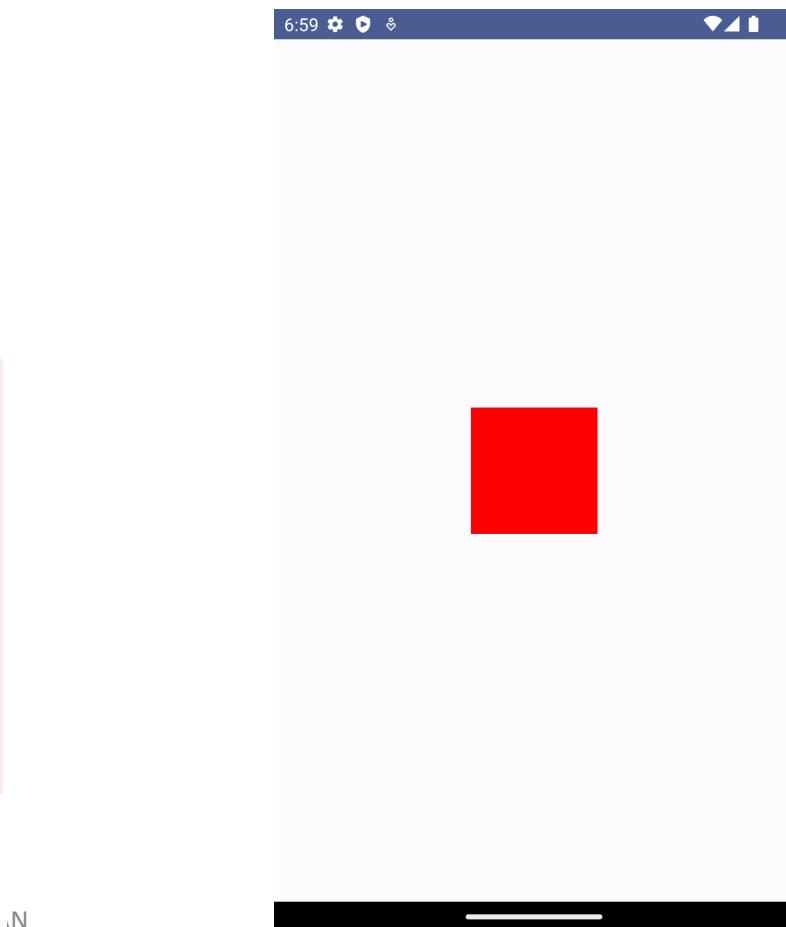


# Gesture Detector

- İstenilen widgeta tıklanılma , çift tıklama , uzun basma gibi özellikler verbiliriz.

```
@Composable
fun Sayfa() {
    Column(
        modifier = Modifier.fillMaxSize(),
        verticalArrangement = Arrangement.SpaceEvenly,
        horizontalAlignment = Alignment.CenterHorizontally
    ) { this: ColumnScope

        Box(Modifier.size(100.dp).background(Color.Red)
            .pointerInput(Unit) { detectTapGestures (
                onTap = { it: Offset
                    Log.e( tag: "Box", msg: "Tıklandı")
                },
                onDoubleTap = { it: Offset
                    Log.e( tag: "Box", msg: "Çift Tıklandı")
                },
                onLongPress = { it: Offset
                    Log.e( tag: "Box", msg: "Üzerine uzun basıldı")
                }
            )
        })
    }
}
```

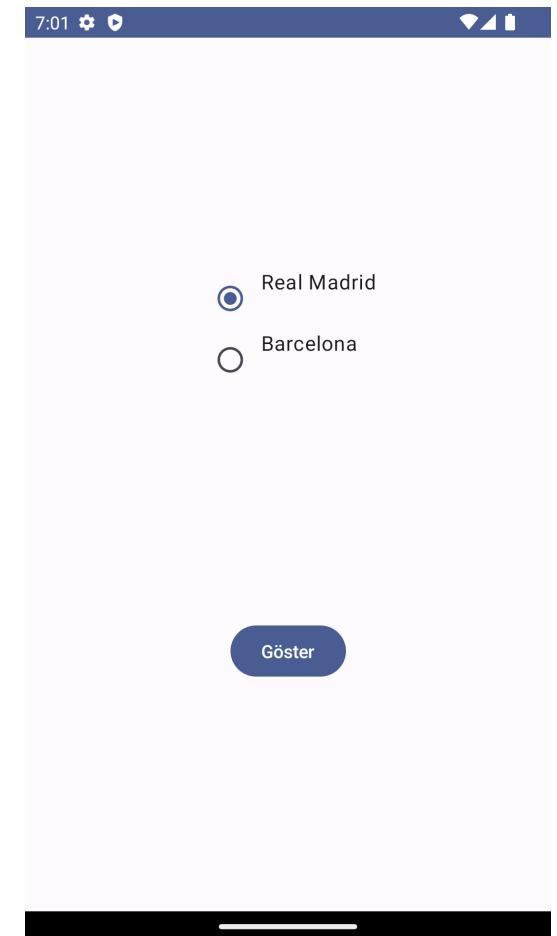


# RadioButton

- Kullanıcıyı tek bir seçime zorlamak istediğimizde kullanırız.

```
@Composable
```

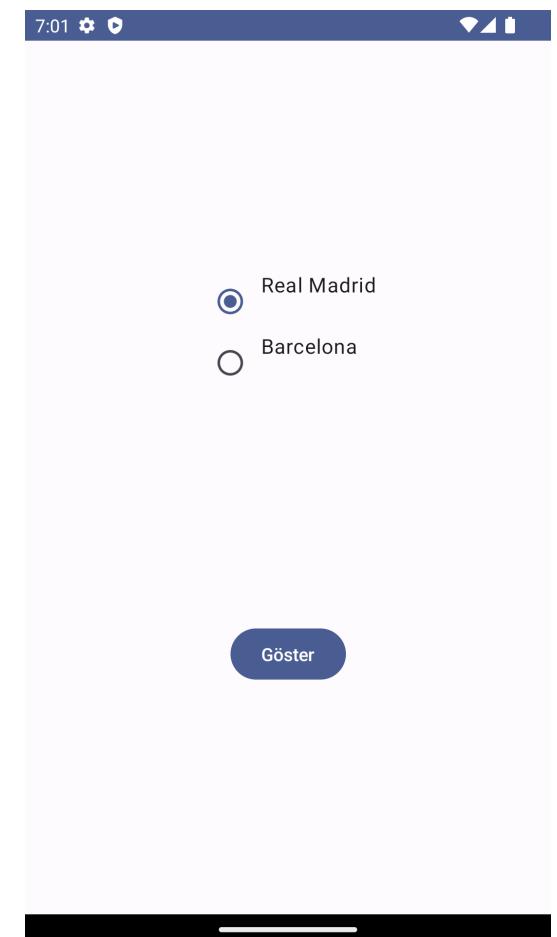
```
fun SayfaRadioButton() {  
    val secilenIndeks = remember { mutableStateOf( value: 0) }  
    val takimListesi = listOf("Real Madrid","Barcelona")  
  
    Column(  
        modifier = Modifier.fillMaxSize(),  
        verticalArrangement = Arrangement.SpaceEvenly,  
        horizontalAlignment = Alignment.CenterHorizontally  
    ) { this: ColumnScope
```



# RadioButton

```
Column { this: ColumnScope
    takimListesi.forEachIndexed { indeks, takim ->
        //Döngü ile liste içeriği alınır ve içinde radio button olan satırlar oluşturulur.
        Row(modifier = Modifier.clickable(
            onClick = {
                secilenIndeks.value = indeks
                //Satıra tıklanılmayı yakalama,
                // amaç hem satıra hep radiobuttona tıklanılmayı yakalama
                Log.e( tag: "Seçilen takım", takimListesi[secilenIndeks.value])
            })) { this: RowScope
            RadioButton(
                //Döngü ile gelen takım ismi ile en son seçilen ülke aynı ise
                // radio button seçili görünür.
                selected = (takim == takimListesi[secilenIndeks.value]),
                onClick = { //Radiobuttona tıklanılmayı yakalama
                    secilenIndeks.value = indeks
                    Log.e( tag: "Seçilen takım", takimListesi[secilenIndeks.value])
                }
            )
            Text(text = takim)
        }
    }
}

Button(onClick = {
    Log.e( tag: "En son seçilmiş takım", takimListesi[secilenIndeks.value])
}) { Text(text = "Göster") }
```



# ProgressIndicator

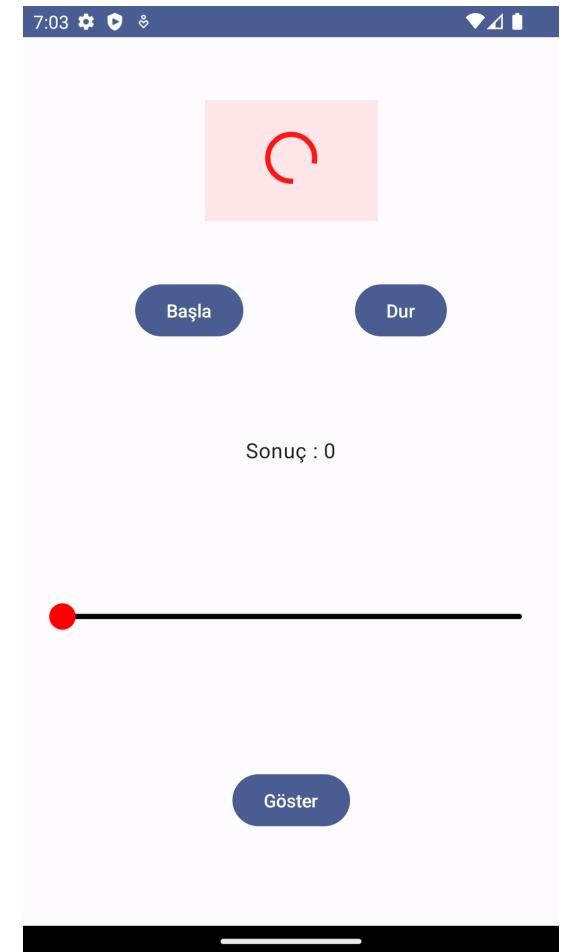
- Yükleme işlemi yaparken temsili olarak gösterdiğimiz widgettir.

```
@Composable
fun Sayfa() {
    val progressDurum = remember { mutableStateOf( value: false) }
    val sliderDeger = remember { mutableStateOf( value: 0f) }
    Column(modifier = Modifier.fillMaxSize(),
        verticalArrangement = Arrangement.SpaceEvenly,
        horizontalAlignment = Alignment.CenterHorizontally
    ) { this: ColumnScope

        if (progressDurum.value){
            CircularProgressIndicator(color = Color.Red)
        }

        Row(horizontalArrangement = Arrangement.SpaceEvenly,
            modifier = Modifier.fillMaxWidth()){ this: RowScope
            Button(onClick = {
                progressDurum.value = true
            }) { Text(text = "Başla") }

            Button(onClick = {
                progressDurum.value = false
            }) { Text(text = "Dur") }
        }
    }
}
```



# Slider

- Belirli değerler arasında kaydırma işlemi ile seçim yapmamızı sağlar.

```
@Composable
fun Sayfa() {
    val progressDurum = remember { mutableStateOf( value: false ) }
    val sliderDeger = remember { mutableStateOf( value: 0f ) }

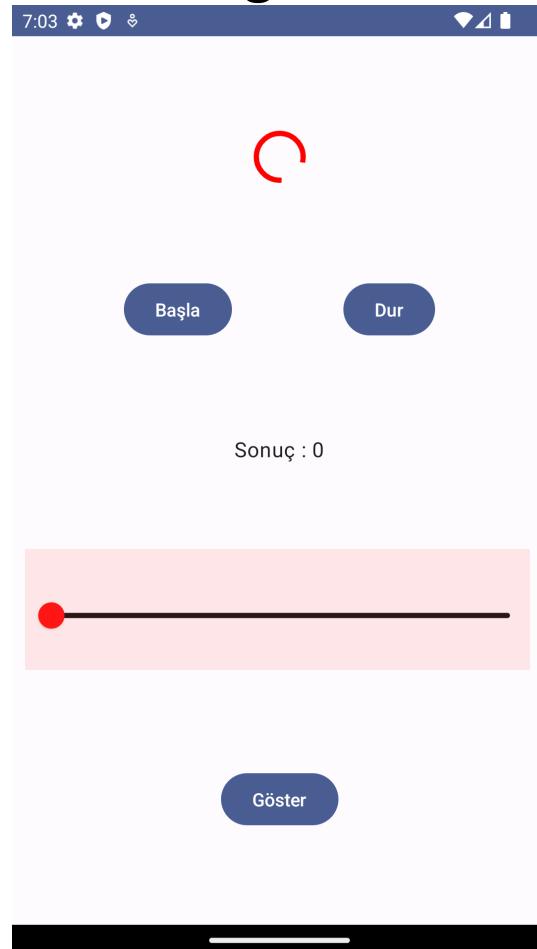
    Column(modifier = Modifier.fillMaxSize(),
```

```
    Text(text = "Sonuç : ${sliderDeger.value.toInt()}")

    Slider(
        value = sliderDeger.value,
        onValueChange = { sliderDeger.value = it },
        valueRange = 0f..100f, //Değer aralığı 0 ile 100
        modifier = Modifier.padding(all = 20.dp),
        colors = SliderDefaults.colors(
            thumbColor = Color.Red, //Yuvarlak rengi
            activeTrackColor = Color.Blue, //Aktif çubuk rengi
            inactiveTrackColor = Color.Black, //Aktif olmayan çubuk rengi
        )
    )

    Button(onClick = {
        Log.e( tag: "Slider en son değeri",sliderDeger.value.toInt().toString() )
    }) { Text(text = "Göster") }
}
```



# Final Kodlama

```
@Composable
fun Sayfa() {
    val progressDurum = remember { mutableStateOf( value: false) }
    val sliderDeger = remember { mutableStateOf( value: 0f) }
    Column(modifier = Modifier.fillMaxSize(),
        verticalArrangement = Arrangement.SpaceEvenly,
        horizontalAlignment = Alignment.CenterHorizontally
    ) { this: ColumnScope

        if (progressDurum.value){
            CircularProgressIndicator(color = Color.Red)
        }

        Row(horizontalArrangement = Arrangement.SpaceEvenly,
            modifier = Modifier.fillMaxWidth()){ this: RowScope
            Button(onClick = {
                progressDurum.value = true
            }) { Text(text = "Başla") }

            Button(onClick = {
                progressDurum.value = false
            }) { Text(text = "Dur") }
        }
    }
}

Text(text = "Sonuç : ${sliderDeger.value.toInt()}")
Slider(
    value = sliderDeger.value,
    onValueChange = { sliderDeger.value = it },
    valueRange = 0f..100f, //Değer aralığı 0 ile 100
    modifier = Modifier.padding(all = 20.dp),
    colors = SliderDefaults.colors(
        thumbColor = Color.Red, //Yuvarlak renk
        activeTrackColor = Color.Blue, //Aktif çubuk renk
        inactiveTrackColor = Color.Black, //Aktif olmayan çubuk renk
    )
)
Button(onClick = {
    Log.e( tag: "Slider en son değeri",sliderDeger.value.toInt().toString())
}) { Text(text = "Göster") }
}
```

# Webview

- Uygulama içinde internet sitelerini göstermek için kullandığımız yapıdır.
- Bu yapı flutter içinde varsayılan olarak gelmemektedir.
- Bu yapıyı projemize eklememiz gereklidir.

```
@SuppressLint("SetJavaScriptEnabled")
@Composable
fun Sayfa() {
    val url = "https://gelecegiyazarlar.turkcell.com.tr/"
    AndroidView(factory = { it: Context
        WebView(it).apply { this: WebView
            layoutParams = ViewGroup.LayoutParams(
                ViewGroup.LayoutParams.MATCH_PARENT,
                ViewGroup.LayoutParams.MATCH_PARENT
            )
            webViewClient = WebViewClient()
            loadUrl(url)
        }
    }, update = { it: WebView
        it.loadUrl(url)
    })
}
```



# Güncelleme

- **http** ile başlayan web servisler varsayılan olarak çalışmamaktadır.
- **https** ile başlayan web servislerde sorun yaşamamaktadır.
- Bu sorunu çözmek için aşağıdaki ifadeyi manifest dosyasına eklemeniz gereklidir.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.volleykullanimi">

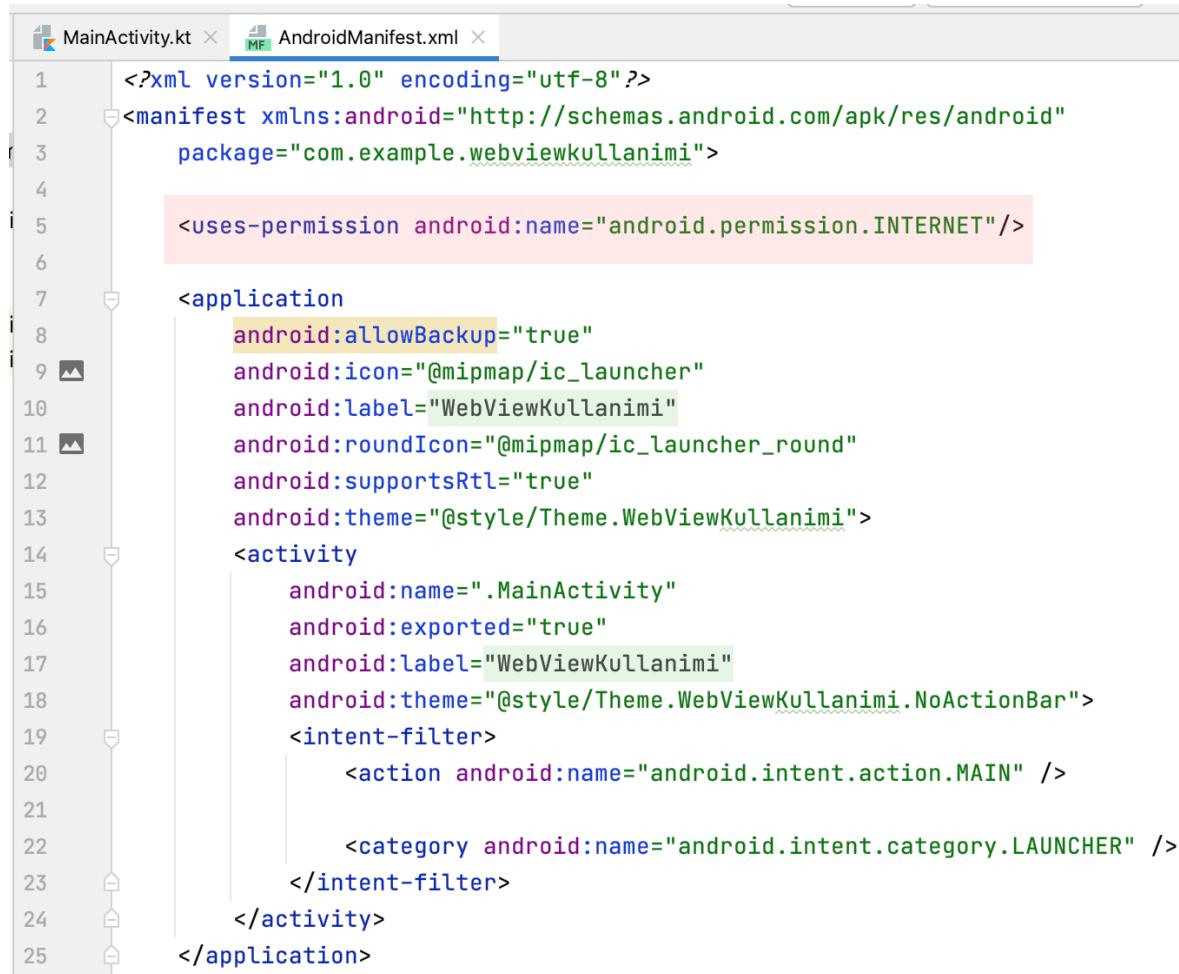
    <uses-permission android:name="android.permission.INTERNET"/>

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="VolleyKullanimi"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme"
        android:usesCleartextTraffic="true">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

# WebView : Internet izni



The screenshot shows the AndroidManifest.xml file in the Android Studio editor. The file contains XML code defining an application with various configurations. A specific line of code, <uses-permission android:name="android.permission.INTERNET"/>, is highlighted with a pink rectangular background, indicating it is selected or being reviewed.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.webviewkullanimi">

    <uses-permission android:name="android.permission.INTERNET"/>

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="WebViewKullanimi"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.WebViewKullanimi">
        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:label="WebViewKullanimi"
            android:theme="@style/Theme.WebViewKullanimi.NoActionBar">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
```

# Image

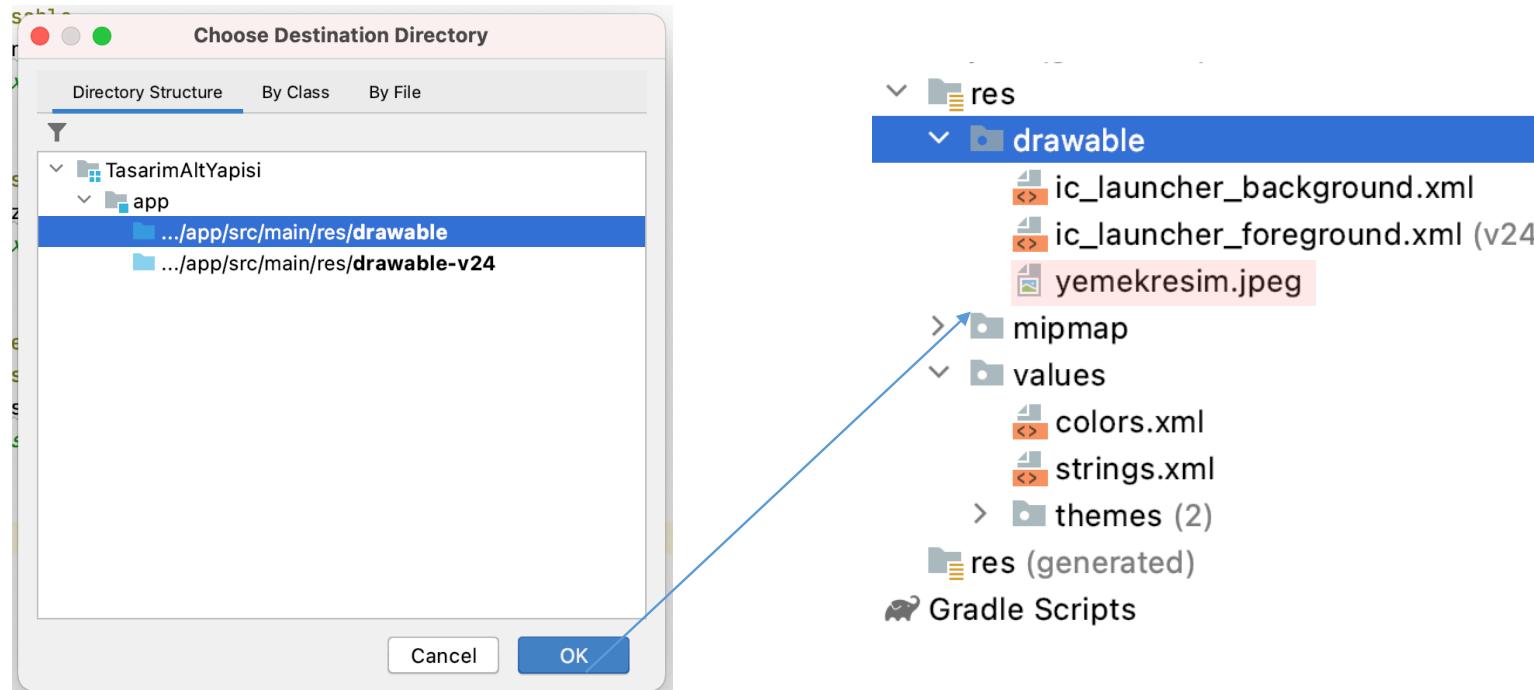
- Belirli bir kaynaktan alınan resimleri gösteren widgettir.
- Resimler lokalde veya internet üzerinde yer alabilir.



# Projeye Resim Ekleme

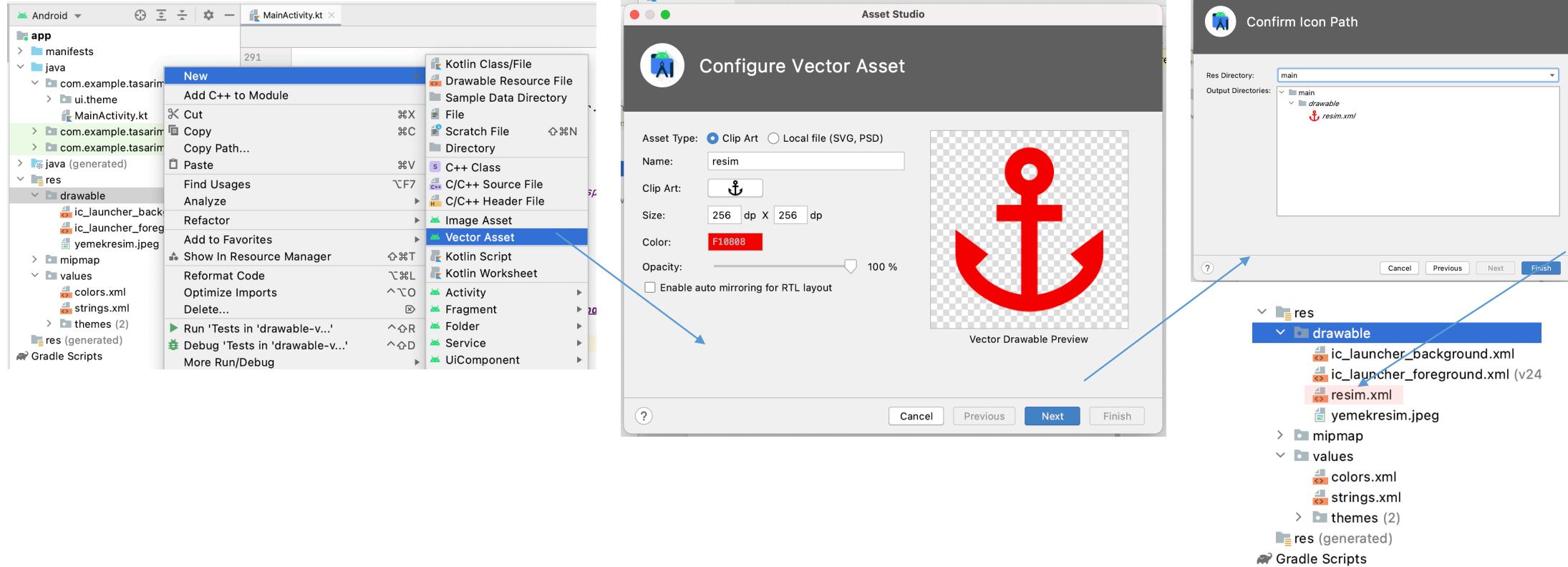
# Dışardan Resim Ekleme

- Android Studio'ya eklemek istediğiniz resmi kopyalayıp drawable klasörüne yapıştırabilirsiniz.



# Android Studio Resimlerini Kullanma

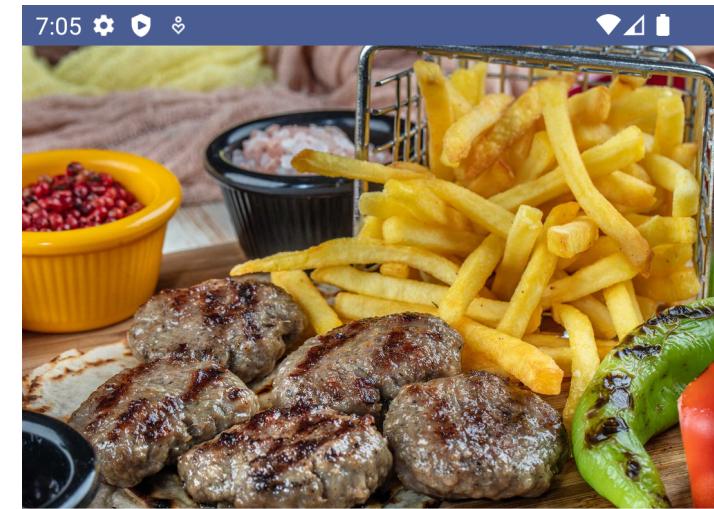
- Google'ın bizlere sunduğu lisanslı resimleri kullanabiliriz.



# Resim kullanımı

@Composable

```
fun ResimKullanimi(){
    Column{    this: ColumnScope
        Image(painter = painterResource(id = R.drawable.yemekresim)
            , contentDescription: "açıklama : yemek resim")
        Image(painter = painterResource(id = R.drawable.resim)
            , contentDescription: "açıklama : vector resim")
    }
}
```



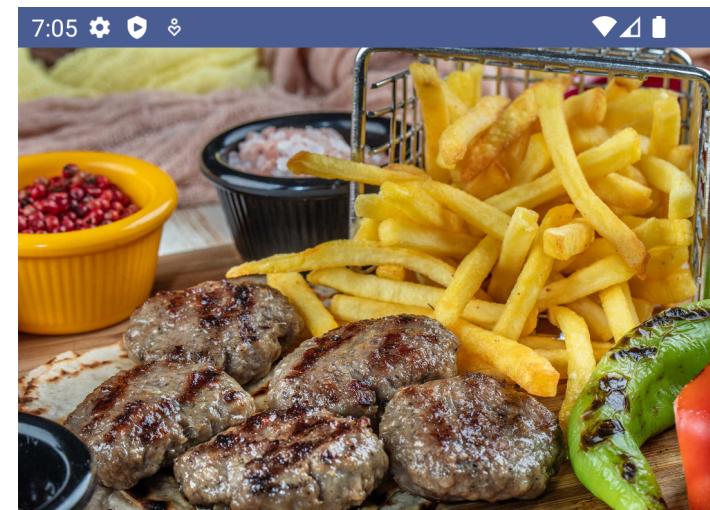
# Resim ismi ile erişim

- Bu işlem sadece bitmap resimlerde geçerlidir. png,jpeg vb.
- Vector resimler için çalışmaz. xml uzantılı resimler için.

```
@Composable
fun Sayfa() {
    Column{    this: ColumnScope
        val activity = (LocalContext.current as Activity)

        Image(bitmap = ImageBitmap
            .imageResource(
                id = activity.resources.getIdentifier(
                    name: "resim",
                    defType: "drawable",
                    activity.packageName)
            , contentDescription: "açıklama : yemek resim")

        Image(painter = painterResource(id = R.drawable.resim)
            , contentDescription: "açıklama : vector resim")
    }
}
```



# DropdownMenu

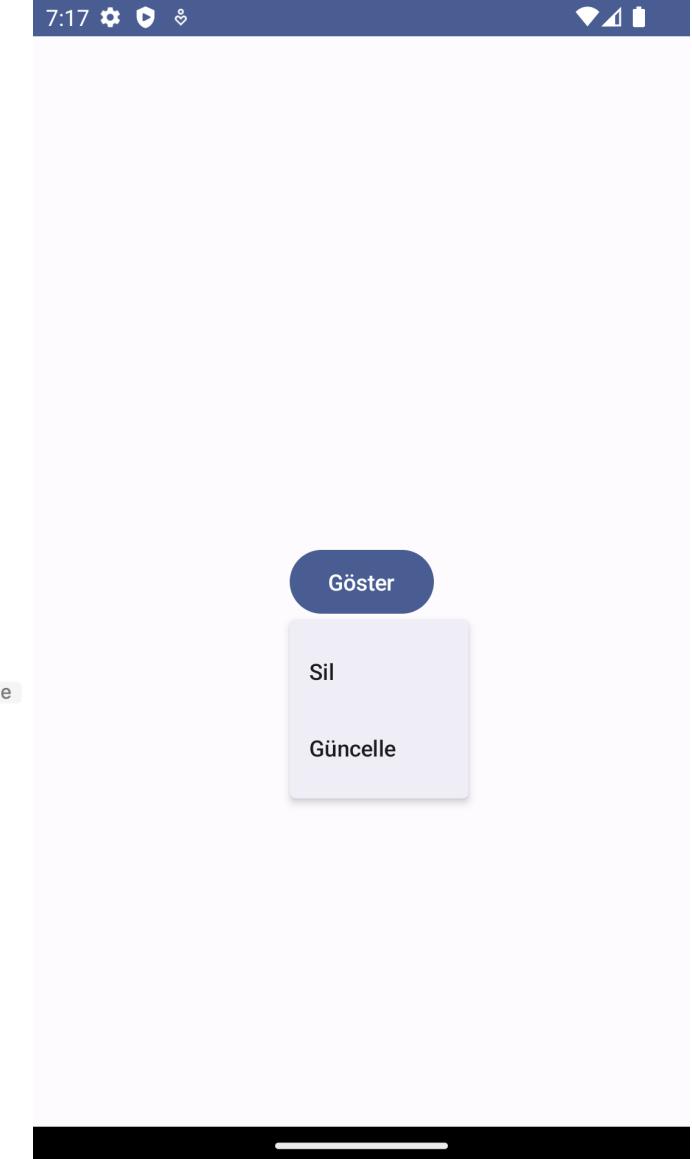
- Sayfa üzerinde tıklanabilir itemlar gösteren bir alt yapıdır.

```
@Composable
fun Sayfa() {
    val menuAcilisKontrol = remember { mutableStateOf( value: false ) }

    Column(
        modifier = Modifier.fillMaxSize(),
        verticalArrangement = Arrangement.SpaceEvenly,
        horizontalAlignment = Alignment.CenterHorizontally
    ) { this: ColumnScope
        Box{ this: BoxScope
            Button(onClick = { menuAcilisKontrol.value = true }) { this: RowScope
                Text(text = "Göster")
            }
        }

        DropdownMenu(
            expanded = menuAcilisKontrol.value,
            onDismissRequest = { menuAcilisKontrol.value = false } ) { this: ColumnScope

            DropdownMenuItem(
                onClick = { menuAcilisKontrol.value = false },
                text = { Text( text: "Sil") },
            )
            DropdownMenuItem(
                onClick = { menuAcilisKontrol.value = false },
                text = { Text( text: "Güncelle") },
            )
        }
    }
}
```

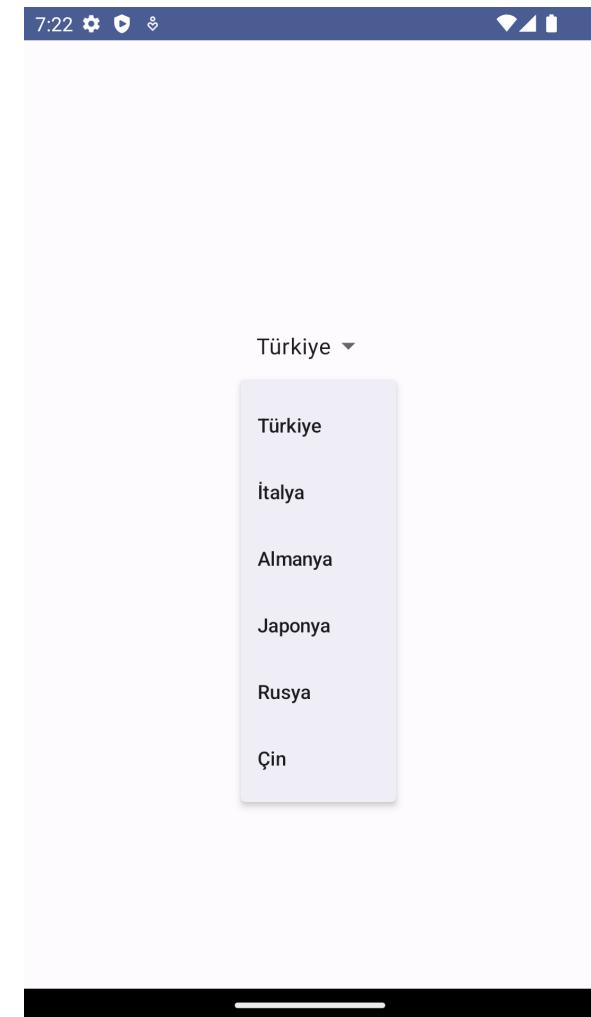


# Dinamik DropdownMenu

- Veri kümesi içeriği kadar DropdownMenuItem Gösterebiliriz.

```
@Composable
fun SayfaDropdownMenu() {
    val ulkeListe = listOf("Türkiye", "İtalya", "Almanya", "Japonya", "Rusya", "Çin")
    val menuAcilisKontrol = remember { mutableStateOf( value: false) }
    val secilenIndeks = remember { mutableStateOf( value: 0) }

    Column(
        modifier = Modifier.fillMaxSize(),
        verticalArrangement = Arrangement.SpaceEvenly,
        horizontalAlignment = Alignment.CenterHorizontally
    ) { this: ColumnScope
        Box{ this: BoxScope
            Row(
                verticalAlignment = Alignment.CenterVertically,
                horizontalArrangement = Arrangement.Center,
                modifier = Modifier
                    .size(100.dp, 50.dp)
                    .clickable {
                        menuAcilisKontrol.value = true
                    }
            )
        } { this: RowScope
            Text(text = ulkeListe[secilenIndeks.value])
            Image(painter = painterResource(id = R.drawable.dropdownmenu_resim),
                contentDescription = "")
        }
    }
}
```



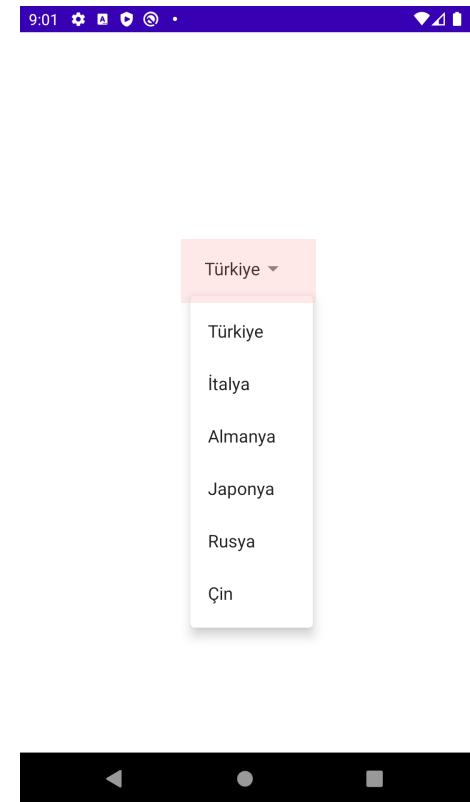
```

DropdownMenu(
    expanded = menuAcilisKontrol.value,
    onDismissRequest = { menuAcilisKontrol.value = false}) { this: ColumnScope

    ulkeListe.forEachIndexed{indeks,ulke ->
        DropdownMenuItem(
            onClick = {
                Log.e( tag: "Menu", msg: "Ülke seçildi : $ulke")
                menuAcilisKontrol.value = false
                secilenIndeks.value = indeks
            },
            text = { Text(text = ulke) },
        )
    }
}

Button(onClick = {
    Log.e( tag: "Menu", msg: "En son seçilen ülke : ${ulkeListe[secilenIndeks.value]}")
}) { this: RowScope
    Text(text = "Göster")
}
}

```



# Sayı Tahmin Uygulaması



Tahmin Oyunu

Kalan Hak : 4

Kazandınız

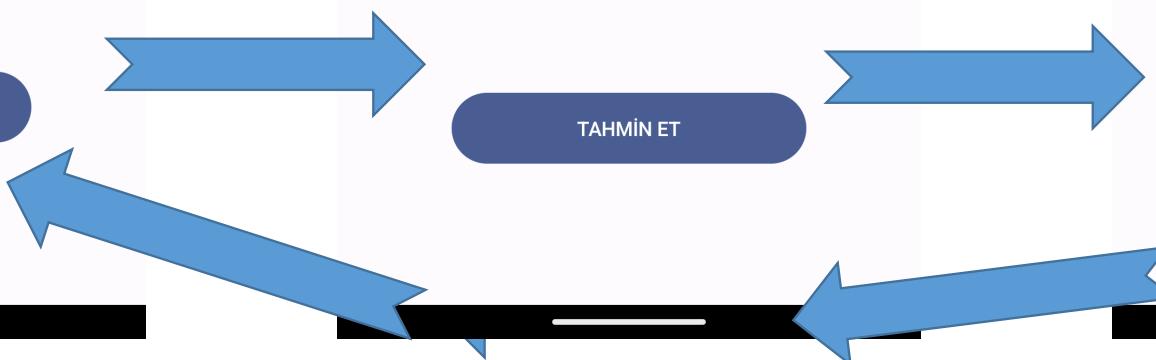
Yardım : Arttır



OYUNA BAŞLA

Tahmin  
|

TAHMİN ET



# Sayfalar Arası Geçiş Ön Kurulum

```
implementation "androidx.navigation:navigation-compose:2.4.0-alpha02"
```

```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            SayiTahminUygulamasıTheme {
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colorScheme.background) @Composable
                ) {
                    SayfaGecisleri()
                }
            }
        }
    }
}

@Composable
fun SayfaGecisleri() {
    val navController = rememberNavController()
    NavHost(navController = navController, startDestination = "anasayfa") {
        composable(route: "anasayfa") { it: NavBackStackEntry
            Anasayfa(navController = navController)
        }

        composable(route: "tahmin_ekrani") { it: NavBackStackEntry
            TahminEkrani(navController = navController)
        }

        composable(route: "sonuc_ekrani/{sonuc}")
            , arguments = listOf(
                navArgument(name: "sonuc"){type = NavType.BoolType})
        ) { it: NavBackStackEntry
            val sonuc = it.arguments?.getBoolean(key: "sonuc")!!
            SonucEkrani(navController = navController, sonuc)
        }
    }
}
```

# Tahmin Oyunu



OYUNA BAŞLA

```
@Composable
fun Anasayfa(navController: NavController) {
    Column(
        modifier = Modifier.fillMaxSize(),
        verticalArrangement = Arrangement.SpaceEvenly,
        horizontalAlignment = Alignment.CenterHorizontally
    ) { this: ColumnScope

        Text(text = "Tahmin Oyunu", fontSize = 36.sp)
        Image(painter = painterResource(id = R.drawable.zar_resim),
              contentDescription = "açıklama")      Resimler : 128dp x 128dp
        Button(
            onClick = { navController.navigate(route: "tahmin_ekrani") },
            modifier = Modifier.size(width = 250.dp, height = 50.dp)
        ) { this: RowScope
            Text(text = "OYUNA BAŞLA")
        }
    }
}
```

# Kalan Hak : 4

Yardım : Arttır



```
@SuppressLint("UnusedMaterial3ScaffoldPaddingParameter")
@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun TahminEkranı(navController: NavController) {
    val tfTahmin = remember { mutableStateOf("") }
    val rasgeleSayı = remember { mutableStateOf(0) }
    val kalanHak = remember { mutableStateOf(5) }
    val yönlendirme = remember { mutableStateOf("") }

    Column(
        modifier = Modifier.fillMaxSize(),
        verticalArrangement = Arrangement.SpaceEvenly,
        horizontalAlignment = Alignment.CenterHorizontally
    ) { this: ColumnScope
        LaunchedEffect(key1 = true){ this: CoroutineScope
            rasgeleSayı.value = Random.nextInt(until: 101)//0 ile 100
            Log.e(tag: "Rasgele sayı",rasgeleSayı.value.toString())
        }

        Text(text = "Kalan Hak : ${kalanHak.value}",fontSize = 36.sp,color = Color.Red)
        Text(text = "Yardım : ${yönlendirme.value}",fontSize = 24.sp)
        TextField(
            value = tfTahmin.value,
            onValueChange = { tfTahmin.value = it},
            label = { Text(text = "Tahmin")} )
    }
}
```

7:35 ☰ 🔍



# Kalan Hak : 4

Yardım : Arttır

TAHMİN ET

```
Button(  
    onClick = {  
        kalanHak.value = kalanHak.value - 1  
        val tahmin = tfTahmin.value.toInt()  
  
        if(tahmin == rasgeleSayi.value){  
            navController.navigate( route: "sonuc_ekrani/true"){ this: NavOptionsBuilder  
                popUpTo( route: "tahmin_ekrani") { inclusive = true }  
            }  
            return@Button  
        }  
        if (tahmin > rasgeleSayi.value){  
            yonlendirme.value = "Azalt"  
        }  
        if (tahmin < rasgeleSayi.value){  
            yonlendirme.value = "Arttır"  
        }  
        if (kalanHak.value == 0){  
            navController.navigate( route: "sonuc_ekrani/false"){ this: NavOptionsBuilder  
                popUpTo( route: "tahmin_ekrani") { inclusive = true }  
            }  
        }  
        tfTahmin.value = ""  
    },  
    modifier = Modifier.size(width = 250.dp, height = 50.dp)  
) { this: RowScope  
    Text(text = "TAHMİN ET")  
}
```



# Kazandınız



```
@Composable
fun SonucEkranı(navController: NavController,sonuc:Boolean) {

    Column(
        modifier = Modifier.fillMaxSize(),
        verticalArrangement = Arrangement.SpaceEvenly,
        horizontalAlignment = Alignment.CenterHorizontally
    ) { this: ColumnScope

        if(sonuc){
            Text(text = "Kazandınız",fontSize = 36.sp)
            Image(painter = painterResource(id = R.drawable.mutlu_resim)
                  ,contentDescription = "açıklama")
        }else{
            Text(text = "Kaybettiniz",fontSize = 36.sp)
            Image(painter = painterResource(id = R.drawable.uzgun_resim)
                  ,contentDescription = "açıklama")
        }
    }
}
```

# Teşekkürler...



kasım-adalan



kasimadalan@gmail.com



kasimadalan