Jetpack Compose ile Android Uygulama Geliştirme Kursu

Material Design

Kasım ADALAN
Elektronik ve Haberleşme Mühendisi
Android - IOS Developer and Trainer

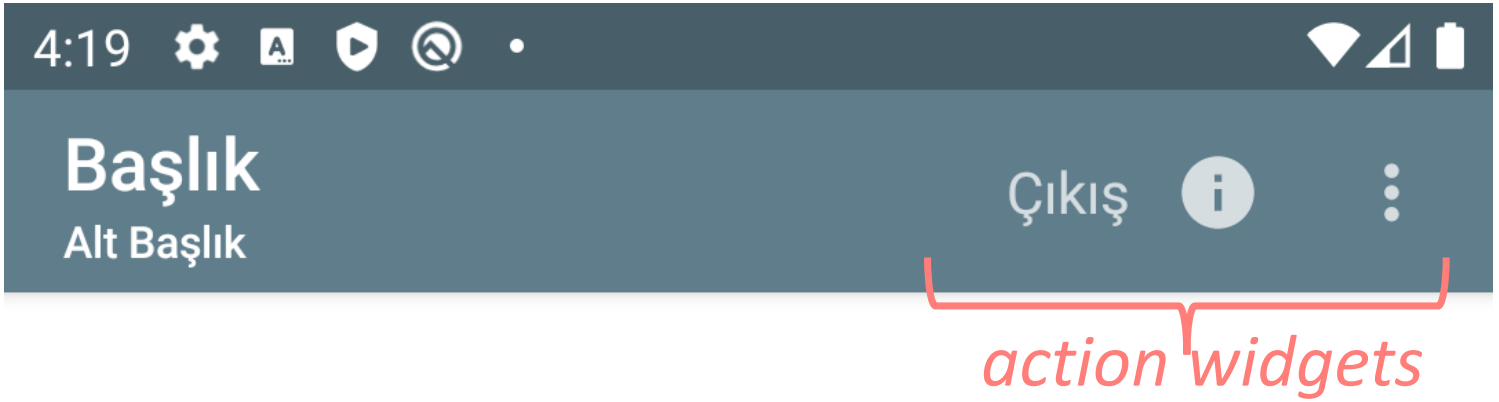
Eğitim İçeriği

- TopAppBar
- Card
- LazyColum
- LazyRow
- LazyColumn Dinamik Liste
- BottomBar
- Drawer

TopAppBar

TopAppBar

- AppBar, sayfanın en üstünde bulunan bir widgettır.
- Varsayılan olarak her sayfada gelmektedir.



TopAppBar Kullanımı

```
@SuppressLint("UnusedMaterial3ScaffoldPaddingParameter")
@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun Sayfa() {
    Scaffold(
        topBar = {
            TopAppBar(
                title = { Text(text = "Başlık")},
            )
        },
        content = { it: PaddingValues
            //Sayfa tasarım içeriği
        }
    )
}
```

Status bar rengi değişimi

- Uygulama durum barının rengini değiştirmeliyiz.

Renk Tanımlama *colors.xml*

```xml
<resources>
    <color name="purple_200">#FFBB86FC</color>
    <color name="purple_500">#FF6200EE</color>
    <color name="purple_700">#FF3700B3</color>
    <color name="teal_200">#FF03DAC5</color>
    <color name="teal_700">#FF018786</color>
    <color name="black">#FF000000</color>
    <color name="white">#FFFFFFFF</color>
    <color name="anaRenk">#607D8B</color>
    <color name="anaRenkKoyu">#485E68</color>
</resources>
```

Status Bar Renk Değişimi *Theme.kt*

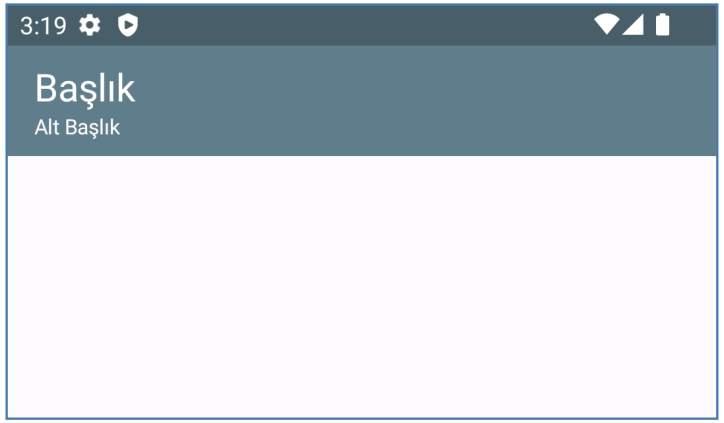
```
val view = LocalView.current

val statusBarRenk = colorResource(id = R.color.anaRenkKoyu).toArgb()

if (!view.isInEditMode) {
    SideEffect {
        val window = (view.context as Activity).window
        window.statusBarColor = statusBarRenk
        WindowCompat.getInsetsController(window, view).isAppearanceLightStatusBars = darkTheme
    }
}
```

TopAppBar Özelleştirme

```
@SuppressLint("UnusedMaterial3ScaffoldPaddingParameter")
@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun Sayfa() {
    Scaffold(
        topBar = {
            TopAppBar(
                title = { Text(text = "Başlık")},
                colors = TopAppBarDefaults.smallTopAppBarColors(
                    containerColor = colorResource(id = R.color.anaRenk),//Arkaplan rengi
                    titleContentColor = colorResource(id = R.color.white),//İçerik Rengi
                )
            )
        },
        content = { it: PaddingValues
            //Sayfa tasarım içeriği
        }
    )
}
```

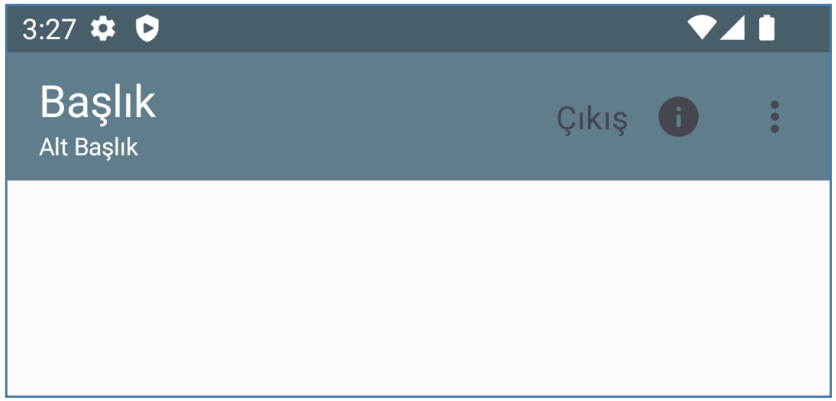
TopAppBar Başlık Özelleştirme

```
@SuppressLint("UnusedMaterial3ScaffoldPaddingParameter")
@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun Sayfa() {
    Scaffold(
        topBar = {
            TopAppBar(
                title = {
                    Column { this: ColumnScope
                        Text(text = "Başlık")
                        Text(text = "Alt Başlık",fontSize = 12.sp)
                    }
                },
                colors = TopAppBarDefaults.smallTopAppBarColors(
                    containerColor = colorResource(id = R.color.anaRenk),//Arkaplan rengi
                    titleContentColor = colorResource(id = R.color.white),//İçerik Rengi
                ),
            )
        },
        content = { it: PaddingValues
            //Sayfa tasarım içeriği
        }
    )
}
```



TopAppBar Action Ekleme

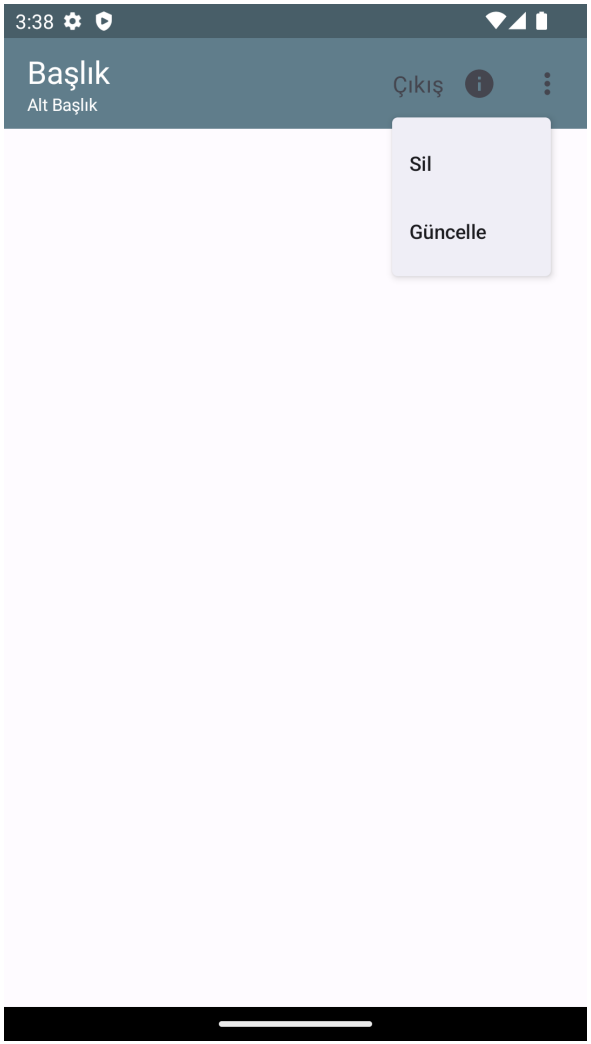
```
TopAppBar(
    title = {
        Column { this: ColumnScope
            Text(text = "Başlık")
            Text(text = "Alt Başlık",fontSize = 12.sp)
        }
    },
    colors = TopAppBarDefaults.smallTopAppBarColors(
        containerColor = colorResource(id = R.color.anaRenk),//Arkaplan rengi
        titleContentColor = colorResource(id = R.color.white),//İçerik Rengi
    ),
    actions = { this: RowScope
        Text(text = "Çıkış",
            modifier = Modifier.clickable { Log.e( tag: "TopBar",  msg: "Çıkış Seçildi") })
        IconButton(onClick = { Log.e( tag: "TopBar",  msg: "Info Seçildi") }) {
            Icon(painter = painterResource(id = R.drawable.info_resim),
                contentDescription = "")
        }
        IconButton(onClick = { Log.e( tag: "TopBar",  msg: "More Seçildi") }) {
            Icon(painter = painterResource(id = R.drawable.more_resim),
                contentDescription = "")
        }
    }
)
```



TopAppBar DropDownMenu Ekleme

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

```
actions = { this: RowScope
    Text(text = "Çıkış", modifier = Modifier.clickable { Log.e( tag: "TopBar",  msg: "Çıkış Seçildi") })
    IconButton(onClick = { Log.e( tag: "TopBar",  msg: "Info Seçildi") }) {
        Icon(painter = painterResource(id = R.drawable.info_resim), contentDescription = "")
    }
    IconButton(onClick = { menuAcilisKontrol.value = true }) {
        Icon(painter = painterResource(id = R.drawable.more_resim), contentDescription = "")
    }
    DropdownMenu(
        expanded = menuAcilisKontrol.value,
        onDismissRequest = { menuAcilisKontrol.value = false }) { this: ColumnScope
        DropdownMenuItem(
            onClick = {
                Log.e( tag: "TopBar",  msg: "Menu sil seçildi")
                menuAcilisKontrol.value = false
            }, text = { Text( text: "Sil") },
        )
        DropdownMenuItem(
            onClick = {
                Log.e( tag: "TopBar",  msg: "Menu güncelle seçildi")
                menuAcilisKontrol.value = false
            }, text = { Text( text: "Güncelle") },
        )
    }
}
```



TopAppBar Arama Özelliği

- TopAppBar üzerinde arama işlemi yapmak için textfield kullanılabilir.
- İki tasarım oluşturulur arama yapıldığı durumda için ve arama yapılmadığı durum için.
- Arama yapıldığını state özelliği olan bir değişken ile takip edebiliriz.


```
val aramaYapiliyorMu = remember { mutableStateOf( value: false) }
```

TopAppBar Arama Özelliği

```
@SuppressLint("UnusedMaterial3ScaffoldPaddingParameter")
@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun SayfaTopAppBarArama() {
    val aramaYapiliyorMu = remember { mutableStateOf( value: false) }
    val tf = remember { mutableStateOf( value: "") }
    Scaffold(
        topBar = {
            TopAppBar(
                title = {
                    if(aramaYapiliyorMu.value){
                        TextField(
                            value = tf.value,
                            onValueChange = { it: String
                                tf.value = it
                                Log.e( tag: "TopBar",  msg: "Arama : $it")
                            },
                            label = { Text(text = "Ara")},
                            colors = TextFieldDefaults.textFieldColors(
                                containerColor = Color.Transparent,//Arkaplan rengi
                                focusedIndicatorColor = Color.White,//Seçildiğinde Belirteç rengi
                                unfocusedIndicatorColor = Color.White,//Seçilmediğinde Belirteç rengi
                                textColor = Color.Black//Yazı rengi
                            )
                        )
                    }else{ Text(text = "Başlık") }
                },
                actions = { this: RowScope
                    if(aramaYapiliyorMu.value){
                        IconButton(onClick = {
                            aramaYapiliyorMu.value = false
                            tf.value = ""
                        }) {
                            Icon(painter = painterResource(id = R.drawable.kapat_resim),
                                contentDescription = "açıklama")
                        }
                    }else{
                        IconButton(onClick = { aramaYapiliyorMu.value = true }) {
                            Icon(painter = painterResource(id = R.drawable.arama_resim),
                                contentDescription = "açıklama")
                        }
                    }
                }
            )
        },
        content = { it: PaddingValues
            //Tasarım içeriği
        },
    )
}
```

Card

Card

- Tasarım üzerinde kullanabileceğimiz kart etkisi yaratan widgettır.
- Listeleme ve sayfa üzerinde sabit tasarımlarda kullanılabilir.

```
@Composable
fun SayfaCard() {
    Column(modifier = Modifier.fillMaxSize(),
        verticalArrangement = Arrangement.Center) { this: ColumnScope
        Card(modifier = Modifier
            .padding(all = 10.dp)
            .fillMaxWidth(),
            elevation = CardDefaults.cardElevation(
                defaultElevation = 10.dp
            ),
            colors = CardDefaults.cardColors(
                containerColor = Color.Blue,
            ),
            shape = RoundedCornerShape(corner = CornerSize(16.dp)),
            border = BorderStroke(2.dp,Color.Magenta)
        ) { this: ColumnScope
            Row(horizontalArrangement = Arrangement.Center,
                modifier = Modifier.fillMaxWidth().clickable { Log.e( tag: "Card", msg: "Tıklandı") }) {
                Column(
                    horizontalAlignment = Alignment.CenterHorizontally,
                    modifier = Modifier.padding(all = 10.dp),
                ) { this: ColumnScope
                    Image(painter = painterResource(id = R.drawable.gunes_resim),
                        contentDescription = "")
                    Text(text = "Güneş",color = Color.White,fontSize = 36.sp)
                }
            }
        }
    }
```

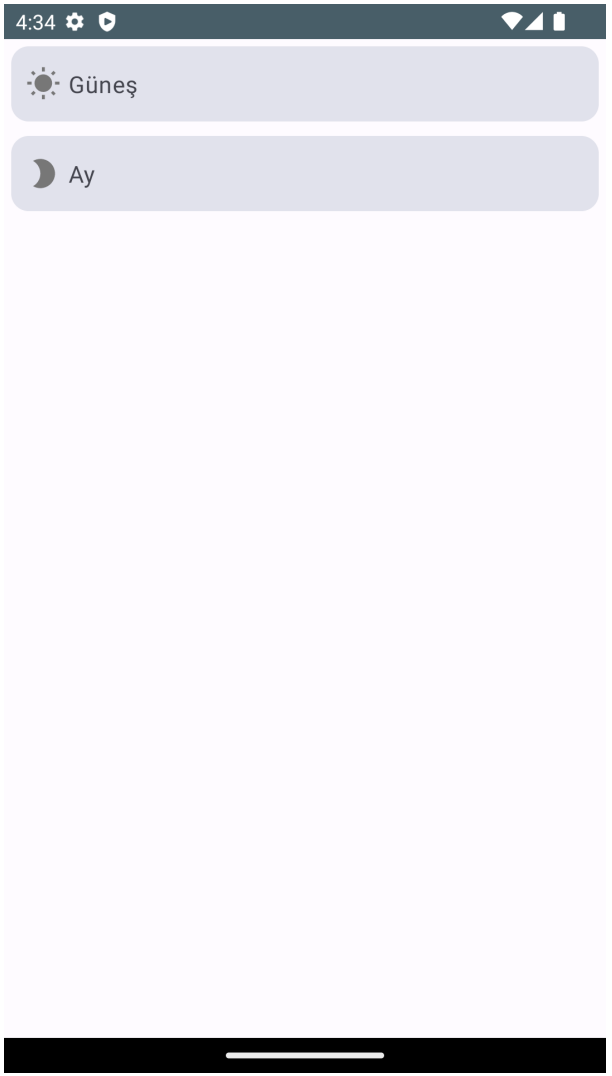
Listeleme

LazyColumn Sabit Liste

LazyColumn : Sabit Liste

- Dikey listemele yapabiliriz.

```
@Composable
fun ListelemeDikeySayfa() {
    LazyColumn { this: LazyListScope
        item { this: LazyItemScope
            Card(modifier = Modifier.padding(all = 5.dp).fillMaxWidth()) {
                Row(modifier = Modifier.clickable {
                    Log.e( tag: "Güneş", msg: "Tıklandı")
                }){ this: RowScope
                    Row(verticalAlignment = Alignment.CenterVertically,
                        modifier = Modifier.padding(all = 10.dp)) { this: RowScope
                        Image(painter = painterResource(id = R.drawable.gunes_resim)
                            , contentDescription = "açıklama")//24 dp resim
                        Text(text = "Güneş",modifier = Modifier.padding(all = 5.dp))
                    }
                }
            }
        }
        item { this: LazyItemScope
            Card(modifier = Modifier.padding(all = 5.dp).fillMaxWidth()) {
                Row(modifier = Modifier.clickable {
                    Log.e( tag: "Ay", msg: "Tıklandı")
                }){ this: RowScope
                    Row(verticalAlignment = Alignment.CenterVertically,
                        modifier = Modifier.padding(all = 10.dp)) { this: RowScope
                        Image(painter = painterResource(id = R.drawable.ay_resim)
                            , contentDescription = "açıklama")//24 dp resim
                        Text(text = "Ay",modifier = Modifier.padding(all = 5.dp))
                    }
                }
            }
        }
    }
}
```



LazyRow Sabit Liste

LazyRow : Sabit Liste

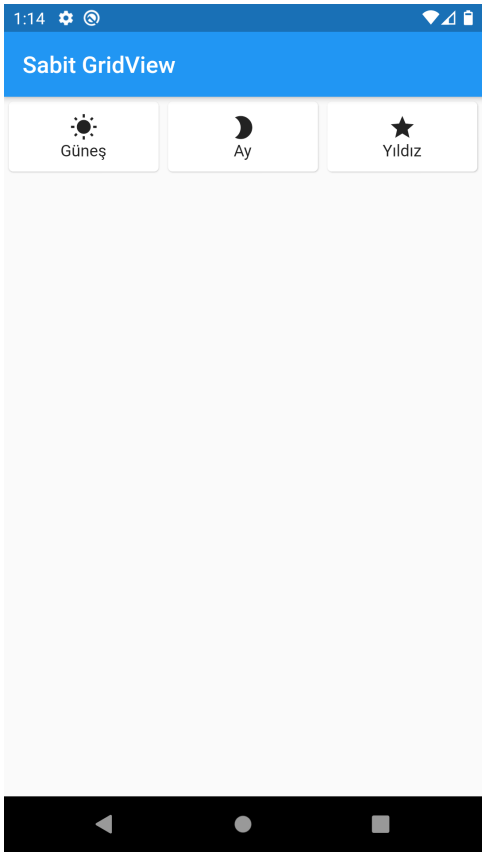
- Yatay listemele yapabiliriz.

```
@Composable
fun SayfaSabitListeleme() {
    LazyRow{ this: LazyListScope
        item { this: LazyItemScope
            Card(modifier = Modifier.padding(all = 5.dp).size(100.dp)) { this: ColumnScope
                Row(horizontalArrangement = Arrangement.Center,
                    verticalAlignment = Alignment.CenterVertically,
                    modifier = Modifier.fillMaxSize().clickable {
                        Log.e( tag: "Liste", msg: "Güneş tıklandı")
                    }) { this: RowScope
                    Row(
                        verticalAlignment = Alignment.CenterVertically,
                        modifier = Modifier.padding(all = 10.dp)
                    ) { this: RowScope
                        Image(painter = painterResource(id = R.drawable.gunes_resim_24),
                            contentDescription = "")
                        Text(text = "Güneş",modifier = Modifier.padding(all = 5.dp))
                    }
                }
            }
        }
        item { this: LazyItemScope
            Card(modifier = Modifier.padding(all = 5.dp).size(100.dp)) { this: ColumnScope
                Row(horizontalArrangement = Arrangement.Center,
                    verticalAlignment = Alignment.CenterVertically,
                    modifier = Modifier.fillMaxSize().clickable {
                        Log.e( tag: "Liste", msg: "Ay tıklandı")
                    }) { this: RowScope
                    Row(
                        verticalAlignment = Alignment.CenterVertically,
                        modifier = Modifier.padding(all = 10.dp)
                    ) { this: RowScope
                        Image(painter = painterResource(id = R.drawable.ay_resim),
                            contentDescription = "")
                        Text(text = "Ay",modifier = Modifier.padding(all = 5.dp))
                    }
                }
            }
        }
```

GridView Sabit Liste

GridView : Sabit Liste

- Satırlar tek tek yan yana veya alt alta oluşturduğumuz liste türüdür.
- İstenilen türde satır oluşturulabilir.

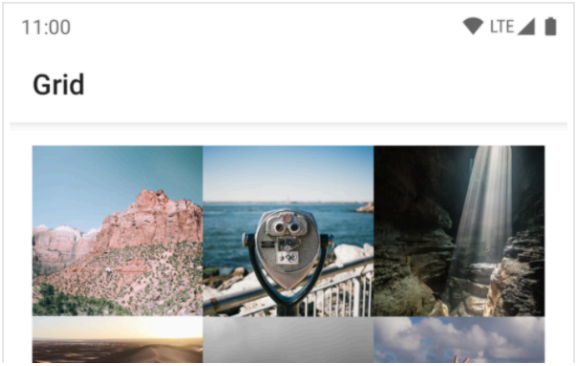


NOT : Android kaynalarında bu yapı henüz hazır değil.

Grids (experimental)

Caution: Experimental APIs can change in the future or may be removed entirely.

The `LazyVerticalGrid` composable provides experimental support for displaying items in a grid.

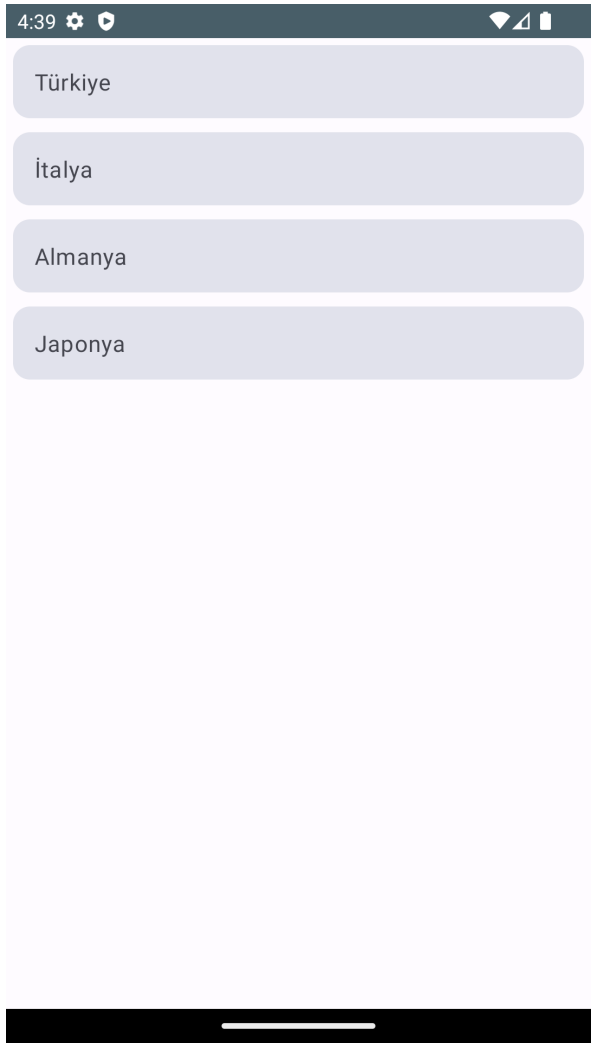


LazyColumn Dinamik Liste

LazyColumn : Dinamik Liste

```
@Composable
fun Sayfa() {
    val ulkeListesi = remember { mutableStateListOf("Türkiye","İtalya","Almanya","Japonya") }

    LazyColumn { this: LazyListScope
        items(
            count = ulkeListesi.count(),
            itemContent = { this: LazyItemScope
                val ulke = ulkeListesi[it]

                Card(modifier = Modifier.padding(all = 5.dp).fillMaxWidth()) {
                    Row{ this: RowScope

                        Row(verticalAlignment = Alignment.CenterVertically,
                            modifier = Modifier.padding(all = 10.dp)) { this: RowScope
                            Text(text = ulke,modifier = Modifier.padding(all = 5.dp))
                        }
                    }
                }
            })
    }
}
```

Satır Tasarımı



Satır Tasarımına Tıklama

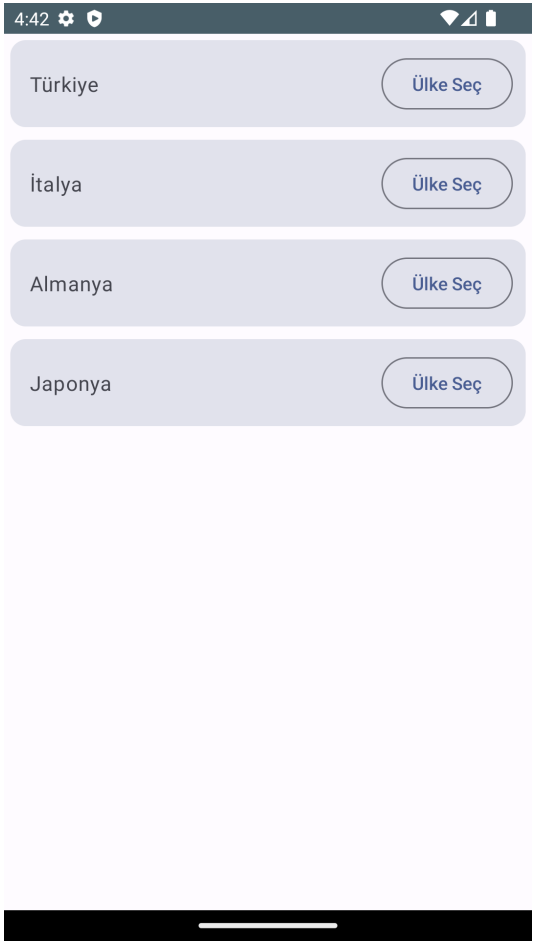
```
@Composable
fun Sayfa() {
    val ulkeListesi = remember { mutableStateListOf("Türkiye","İtalya","Almanya","Japonya") }

    LazyColumn { this: LazyListScope
        items(
            count = ulkeListesi.count(),
            itemContent = { this: LazyItemScope
                val ulke = ulkeListesi[it]

                Card(modifier = Modifier.padding(all = 5.dp).fillMaxWidth()) {
                    Row(modifier = Modifier.clickable {
                        Log.e(ulke, msg: "Seçildi")
                    }){ this: RowScope

                        Row(verticalAlignment = Alignment.CenterVertically,
                            modifier = Modifier.padding(all = 10.dp)) { this: RowScope
                            Text(text = ulke,modifier = Modifier.padding(all = 5.dp))
                        }
                    }
                }
            })
    }
}
```

Card tıklanılma özelliğini çalıştıramadığı için onun içini kapsayacak Row ile tıklanılmayı tespit edebiliriz.

Satır Üzerindeki Widgetlara Tıklama

- Varsayılan olarak tıklama özelliği olan widgetların *onClick* özelliği kullanılabilir.
- Varsayılan olarak tıklama özelliği yoksa *GestureDetector* kullanılabilir.

```
@Composable
fun Sayfa() {
    val ulkeListesi = remember { mutableStateListOf("Türkiye","İtalya","Almanya","Japonya") }
    LazyColumn { this: LazyListScope
        items(
            count = ulkeListesi.count(),
            itemContent = { this: LazyItemScope
                val ulke = ulkeListesi[it]
                Card(modifier = Modifier.padding(all = 5.dp).fillMaxWidth()) {
                    Row(modifier = Modifier.clickable {
                        Log.e(ulke, msg: "Seçildi")
                    }){ this: RowScope

                        Row(verticalAlignment = Alignment.CenterVertically,
                            horizontalArrangement = Arrangement.SpaceBetween,
                            modifier = Modifier.padding(all = 10.dp).fillMaxWidth()) { this: RowScope
                            Text(text = ulke,modifier = Modifier.padding(all = 5.dp).clickable {
                                Log.e( tag: "Text ile $ulke", msg: "Seçildi")
                            })
                            OutlinedButton(onClick = {
                                Log.e( tag: "Button ile $ulke", msg: "Seçildi")
                            }) { Text(text = "Ülke Seç") }
                        }
                    }
                }
            })
    }
}
```

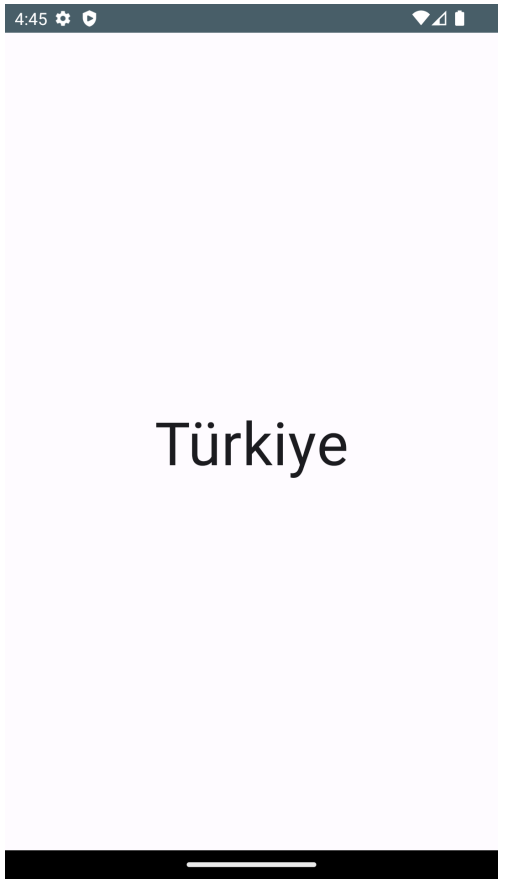


Satıra Tıklayıp Sayfa Geçişi

- Geçiş yapılacak sayfa oluşturulur.

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

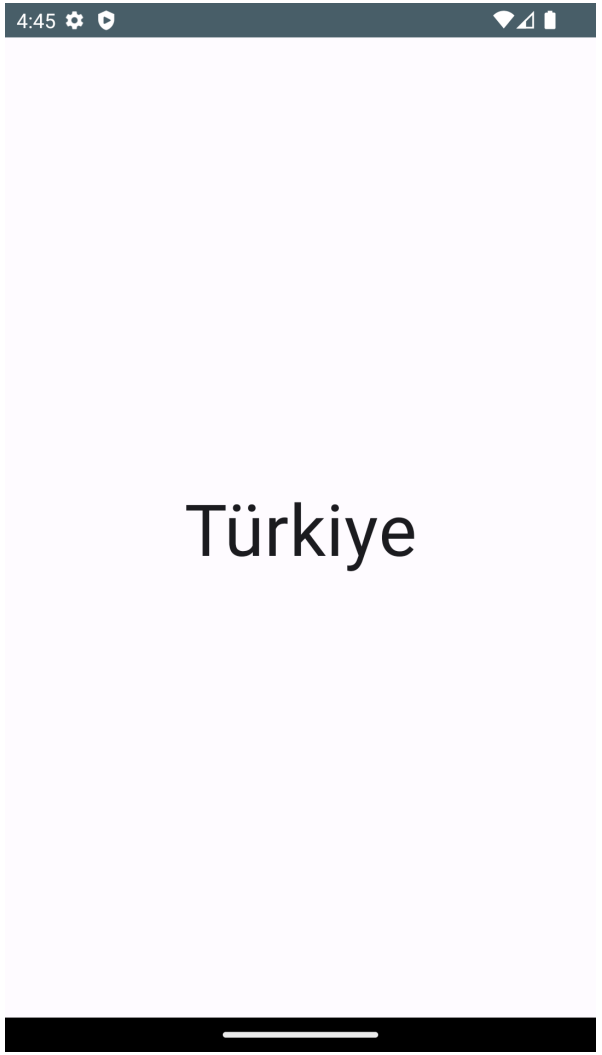
```
@Composable
fun DetaySayfa(navController: NavController,ulkeAdi:String) {
    Column(
        modifier = Modifier.fillMaxSize(),
        verticalArrangement = Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally
    ) { this: ColumnScope

        Text(text = ulkeAdi,fontSize = 50.sp)
    }
}
```



Satıra Tıklayıp Sayfa Geçişi

- Geçişleri oluşturma

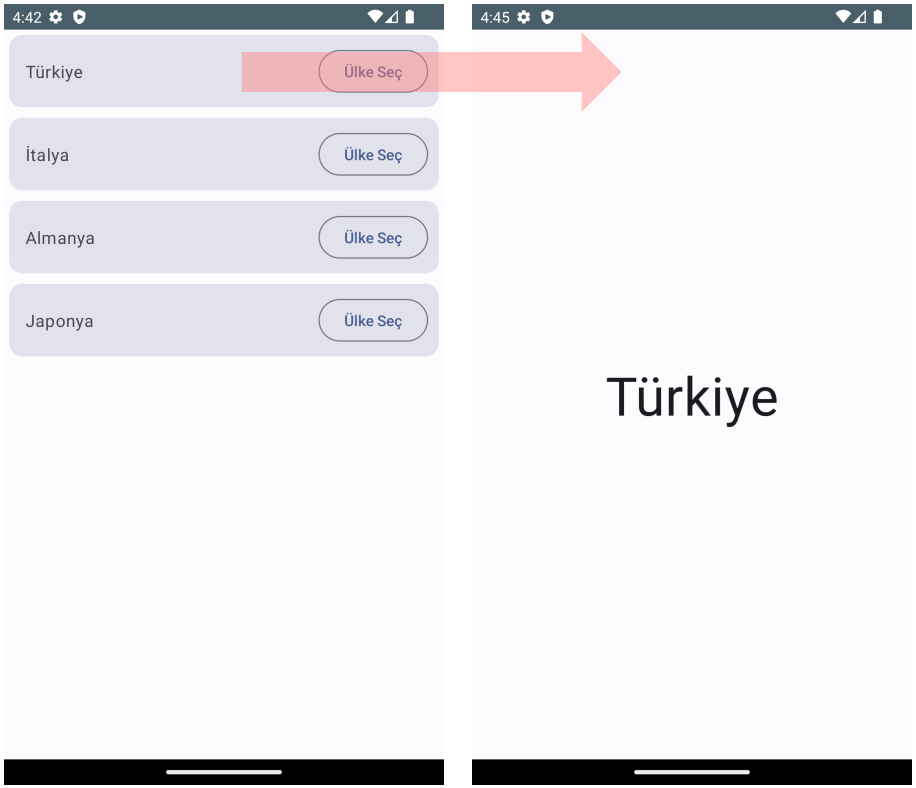
```
@Composable
fun SayfaGecisleri() {
    val navController = rememberNavController()
    NavHost(navController = navController, startDestination = "sayfa") { th
        composable( route: "sayfa") {  it: NavBackStackEntry
            Sayfa(navController = navController)
        }
        composable( route: "detay_sayfa/{ulkeAdi}",
            arguments = listOf(
                navArgument( name: "ulkeAdi") { type = NavType.StringType },
            )
        ) {  it: NavBackStackEntry
            val ulkeAdi = it.arguments?.getString( key: "ulkeAdi")!!
            DetaySayfa(navController = navController,ulkeAdi)
        }
    }
}
```



Satıra Tıklayıp Sayfa Geçişi

- Satıra tıklama işlemi

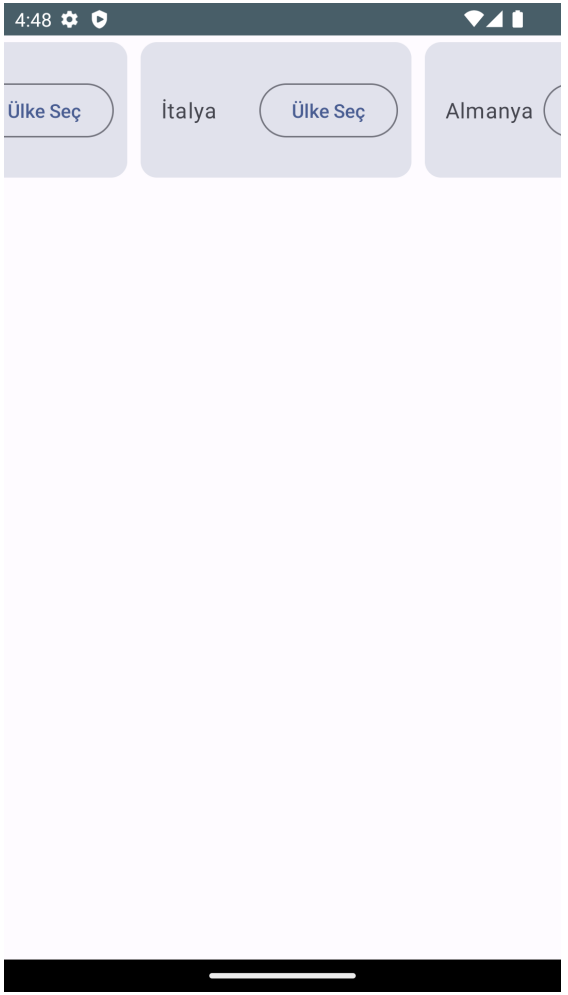
```
@Composable
fun Sayfa(navController: NavController) {
    val ulkeListesi = remember { mutableStateListOf("Türkiye","İtalya","Almanya","Japonya") }
    LazyColumn { this: LazyListScope
        items(
            count = ulkeListesi.count(),
            itemContent = { this: LazyItemScope
                val ulke = ulkeListesi[it]
                Card(modifier = Modifier.padding(all = 5.dp).fillMaxWidth()) {
                    Row(modifier = Modifier.clickable {
                        Log.e(ulke, msg: "Seçildi")
                        navController.navigate( route: "detay_sayfa/$ulke")
                    }){ this: RowScope
                        Row(verticalAlignment = Alignment.CenterVertically,
                            horizontalArrangement = Arrangement.SpaceBetween,
                            modifier = Modifier.padding(all = 10.dp).fillMaxWidth()) { this: RowScope
                            Text(text = ulke,modifier = Modifier.padding(all = 5.dp).clickable {
                                Log.e( tag: "Text ile $ulke", msg: "Seçildi")
                            })
                            OutlinedButton(onClick = {
                                Log.e( tag: "Button ile $ulke", msg: "Seçildi")
                            }) { Text(text = "Ülke Seç") }
                        }
                    }
                }
            })
    }
}
```



LazyRow : Yatay Listeleme

- Yatay scroll edilebilir liste oluşturabiliriz.

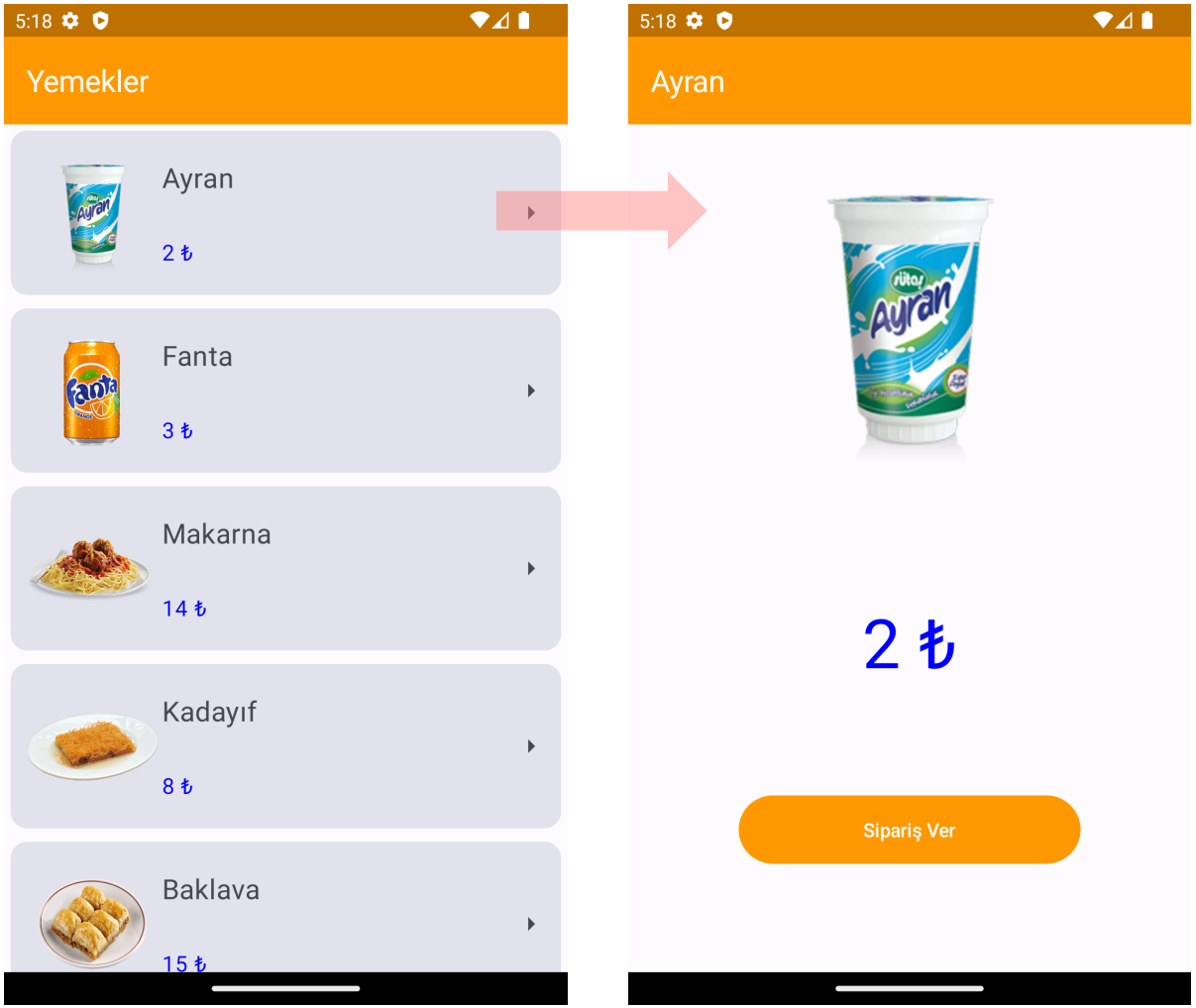
```
@Composable
fun Sayfa(navController: NavController) {
    val ulkeListesi = remember { mutableStateListOf("Türkiye","İtalya","Almanya","Japonya") }
    LazyRow { this: LazyListScope
        items(
            count = ulkeListesi.count(),
            itemContent = { this: LazyItemScope
                val ulke = ulkeListesi[it]
                Card(modifier = Modifier.padding(all = 5.dp).size(200.dp,100.dp)) {
                    Row(verticalAlignment = Alignment.CenterVertically,
                        modifier = Modifier.clickable {
                        Log.e(ulke, msg: "Seçildi")
                        navController.navigate( route: "detay_sayfa/$ulke")
                    }){ this: RowScope
                        Row(verticalAlignment = Alignment.CenterVertically,
                            horizontalArrangement = Arrangement.SpaceBetween,
                            modifier = Modifier.padding(all = 10.dp) .fillMaxSize() ) { this: RowScope
                            Text(text = ulke,modifier = Modifier.padding(all = 5.dp).clickable {
                                Log.e( tag: "Text ile $ulke", msg: "Seçildi")
                            })
                            OutlinedButton(onClick = {
                                Log.e( tag: "Button ile $ulke", msg: "Seçildi")
                            }) { Text(text = "Ülke Seç") }
                        }
                    }
                }
            })
    }
}
```



GridView Dinamik Liste

Detaylı LazyColumn

Detaylı LazyColumn



Renk Oluşturma

- Uygulamada kullanmak için renk oluşturmalıyız.

colors.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="purple_200">#FFBB86FC</color>
    <color name="purple_500">#FF6200EE</color>
    <color name="purple_700">#FF3700B3</color>
    <color name="teal_200">#FF03DAC5</color>
    <color name="teal_700">#FF018786</color>
    <color name="black">#FF000000</color>
    <color name="white">#FFFFFFFF</color>
    <color name="anaRenk">#FF9801</color>
    <color name="anaRenkKoyu">#BF7100</color>
</resources>
```

Status bar rengi değişimi

- Uygulama durum barının rengini değiştirmeliyiz.

Renk Tanımlama *colors.xml*

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="purple_200">#FFBB86FC</color>
    <color name="purple_500">#FF6200EE</color>
    <color name="purple_700">#FF3700B3</color>
    <color name="teal_200">#FF03DAC5</color>
    <color name="teal_700">#FF018786</color>
    <color name="black">#FF000000</color>
    <color name="white">#FFFFFFFF</color>
    <color name="anaRenk">#FF9801</color>
    <color name="anaRenkKoyu">#BF7100</color>
</resources>
```

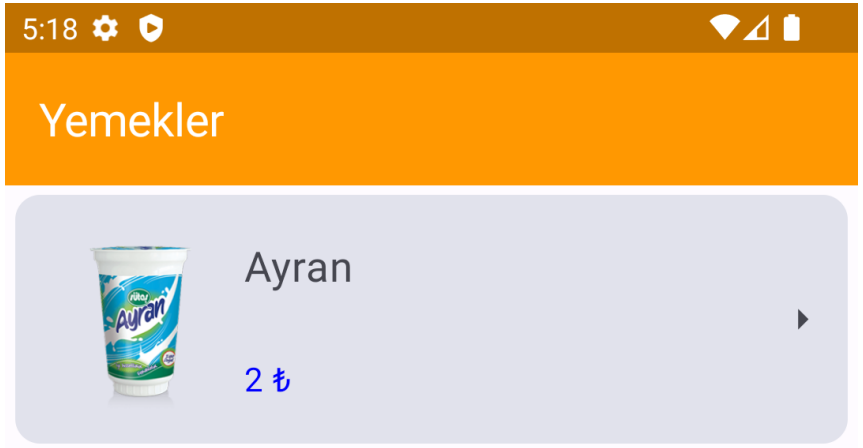
Status Bar Renk Değişimi *Theme.kt*

```
val view = LocalView.current

val statusBarRenk = colorResource(id = R.color.anaRenkKoyu).toArgb()

if (!view.isInEditMode) {
    SideEffect {
        val window = (view.context as Activity).window
        window.statusBarColor = statusBarRenk
        WindowCompat.getInsetsController(window, view).isAppearanceLightStatusBars = darkTheme
    }
}
```

Nesne Tabanlı Çalışma

- Nesne tabanlı çalışmak için satır tasarımına model olacak bir sınıf oluşturmalıyız.

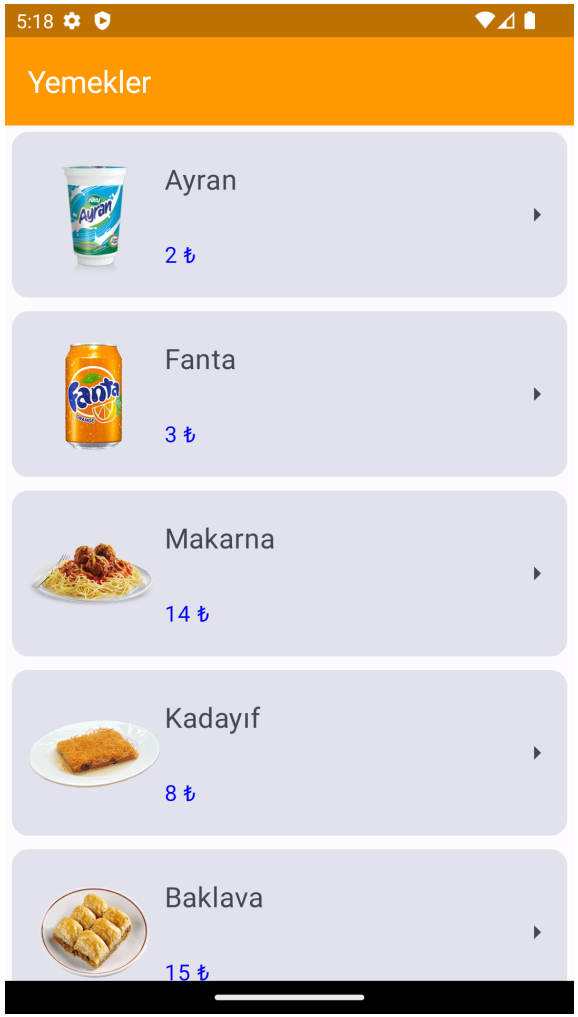
```
data class Yemekler(var yemek_id:Int,
                    var yemek_adi:String,
                    var yemek_resim_adi:String,
                    var yemek_fiyat:Double) {
}
```



Veri Kümesi Oluşturulur.

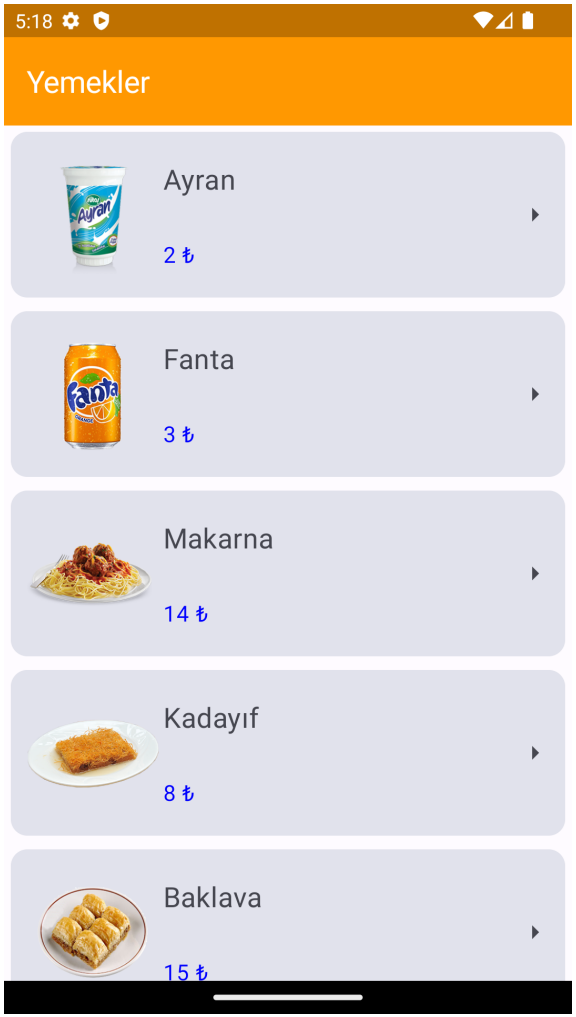
```
@SuppressLint("UnusedMaterial3ScaffoldPaddingParameter")
@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun Anasayfa(navController: NavController) {
    val yemekListesi = remember { mutableStateListOf<Yemekler>() }

    LaunchedEffect(key1 = true){ this: CoroutineScope
        val y1 = Yemekler( yemek_id: 1, yemek_adi: "Köfte", yemek_resim_adi: "kofte", yemek_fiyat: 15)
        val y2 = Yemekler( yemek_id: 2, yemek_adi: "Ayran", yemek_resim_adi: "ayran", yemek_fiyat: 2)
        val y3 = Yemekler( yemek_id: 3, yemek_adi: "Fanta", yemek_resim_adi: "fanta", yemek_fiyat: 3)
        val y4 = Yemekler( yemek_id: 4, yemek_adi: "Makarna", yemek_resim_adi: "makarna", yemek_fiyat: 14)
        val y5 = Yemekler( yemek_id: 5, yemek_adi: "Kadayıf", yemek_resim_adi: "kadayif", yemek_fiyat: 8)
        val y6 = Yemekler( yemek_id: 6, yemek_adi: "Baklava", yemek_resim_adi: "baklava", yemek_fiyat: 15)

        yemekListesi.add(y1)
        yemekListesi.add(y2)
        yemekListesi.add(y3)
        yemekListesi.add(y4)
        yemekListesi.add(y5)
        yemekListesi.add(y6)
    }
```

Satır Tasarımı




```
Scaffold(
    topBar = {
        TopAppBar(
            title = { Text(text = "Yemekler")},
            colors = TopAppBarDefaults.smallTopAppBarColors(
                containerColor = colorResource(id = R.color.anaRenk),
                titleContentColor = colorResource(id = R.color.white),
            )
        )
    },
    content = { it: PaddingValues
        LazyColumn{ this: LazyListScope
            items(
                count = yemekListesi.count(),
                itemContent = { this: LazyItemScope  it: Int
                    val yemek = yemekListesi[it]
                    Card(modifier = Modifier
                        .padding(all = 5.dp)
                        .fillMaxWidth()) { this: ColumnScope
                        Row(modifier = Modifier.clickable {
                            val yemekJson = Gson().toJson(yemek)
                            navController.navigate( route: "detay_sayfa/$yemekJson")
```

Satır Tasarımı




```
itemContent = { this: LazyItemScope  it: Int
    val yemek = yemekListesi[it]
    Card(modifier = Modifier.padding(all = 5.dp).fillMaxWidth()) { this: ColumnScope
        Row(modifier = Modifier.clickable {
            val yemekJson = Gson().toJson(yemek)
            navController.navigate( route: "detay_sayfa/$yemekJson")
        }) { this: RowScope
            Row(verticalAlignment = Alignment.CenterVertically,
                modifier = Modifier.padding(all = 10.dp).fillMaxWidth()
            ) { this: RowScope
                val activity = (LocalContext.current as Activity)
                Image(bitmap = ImageBitmap.imageResource(id = activity.resources.getIdentifier(
                    yemek.yemek_resim_adi, defType: "drawable",activity.packageName
                ) ), contentDescription = "",modifier = Modifier.size(100.dp))
                Row(
                    verticalAlignment = Alignment.CenterVertically,
                    horizontalArrangement = Arrangement.SpaceBetween,
                    modifier = Modifier.fillMaxWidth()
                ) { this: RowScope
                    Column(verticalArrangement = Arrangement.SpaceEvenly,
                        modifier = Modifier.fillMaxHeight()
                    ) { this: ColumnScope
                        Text(text = yemek.yemek_adi,fontSize = 20.sp)
                        Spacer(modifier = Modifier.size(30.dp))
                        Text(text = "${yemek.yemek_fiyat} ₺",color = Color.Blue)
                    }
                    Icon(painter = painterResource(id = R.drawable.arrow_resim),
                        contentDescription = "")
                }
```

Sayfa Geçişi


```
@Composable
fun SayfaGecisleri() {
    val navController = rememberNavController()
    NavHost(navController = navController, startDestination = "anasayfa") {
        composable( route: "anasayfa") {  it: NavBackStackEntry
            Anasayfa(navController = navController)
        }
        composable( route: "detay_sayfa/{yemek}",
            arguments = listOf(
                navArgument( name: "yemek") { type = NavType.StringType },
            )
        ) {  it: NavBackStackEntry
            val json = it.arguments?.getString( key: "yemek")
            val yemek = Gson().fromJson(json, Yemekler::class.java)
            DetaySayfa(navController = navController,yemek)
        }
    }
}
```

Sayfa Geçişi


```
@SuppressLint("UnusedMaterial3ScaffoldPaddingParameter")
@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun DetaySayfa(yemek:Yemekler) {
    Scaffold(
        topBar = {
            TopAppBar(
                title = { Text(text = yemek.yemek_adi) },
                colors = TopAppBarDefaults.smallTopAppBarColors(
                    containerColor = colorResource(id = R.color.anaRenk),
                    titleContentColor = colorResource(id = R.color.white),
                )
            )
        },
```

Sayfa Geçişi


```
content = { it: PaddingValues
    Column(
        modifier = Modifier.fillMaxSize(),
        verticalArrangement = Arrangement.SpaceEvenly,
        horizontalAlignment = Alignment.CenterHorizontally
    ) { this: ColumnScope
        val activity = (LocalContext.current as Activity)
        Image(bitmap = ImageBitmap.imageResource(id = activity.resources.getIdentifier(
            yemek.yemek_resim_adi, defType: "drawable",activity.packageName
        ) ), contentDescription = "",modifier = Modifier.size(250.dp))
        Text(text = "${yemek.yemek_fiyat} ₺",color = Color.Blue,fontSize = 50.sp)
        Button(onClick = {
            Log.e( tag: "Yemek", msg: "${yemek.yemek_adi} siparis verildi")
        },
            modifier = Modifier.size(250.dp,50.dp),
            colors = ButtonDefaults.buttonColors(
                containerColor = colorResource(id = R.color.anaRenk),
                contentColor = Color.White
            )
        ) { this: RowScope
            Text(text = "Siparis Ver")
        }
    }
}
```

Detaylı GridView.builder

NavigationBar

NavigationBar

Sayfanın altında çalışacak şekilde tasarlanmıştır.


```
@SuppressLint("UnusedMaterial3ScaffoldPaddingParameter")
@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun NavigationBarSayfa() {
    val items = listOf("Bir", "İki")
    val secilenItem = remember { mutableStateOf( value: 0) }
    Scaffold(
        topBar = { TopAppBar(title = { Text(text = "Başlık")}) },
        content = { it: PaddingValues
            if(secilenItem.value == 0){ SayfaBir() }
            if(secilenItem.value == 1){ SayfaIki() }
        },
        bottomBar = {
           NavigationBar(containerColor = Color.White) { this: RowScope //Arkaplan rengi
               items.forEachIndexed { index, item ->
                   NavigationBarItem(
```


```
bottomBar = {
    NavigationBar(containerColor = Color.White) { this: RowScope
        items.forEachIndexed{ index,item ->
            NavigationBarItem(
                selected = secilenItem.value == index,
                onClick = { secilenItem.value = index },
                label = { Text(text = item)},
                icon = {
                    when(item){
                        "Bir" -> Icon(painter = painterResource(id = R.drawable.resim1),
                            contentDescription = "")
                        "İki" -> Icon(painter = painterResource(id = R.drawable.resim2),
                            contentDescription = "")
                    }
                },
                colors = NavigationBarItemDefaults.colors(
                    selectedIconColor = Color.Blue,
                    unselectedIconColor = Color.Gray,
                    selectedTextColor = Color.Blue,
                    unselectedTextColor = Color.Gray,
                    indicatorColor = Color.Red
                )
            )
        }
    }
}
```

Sayfalar

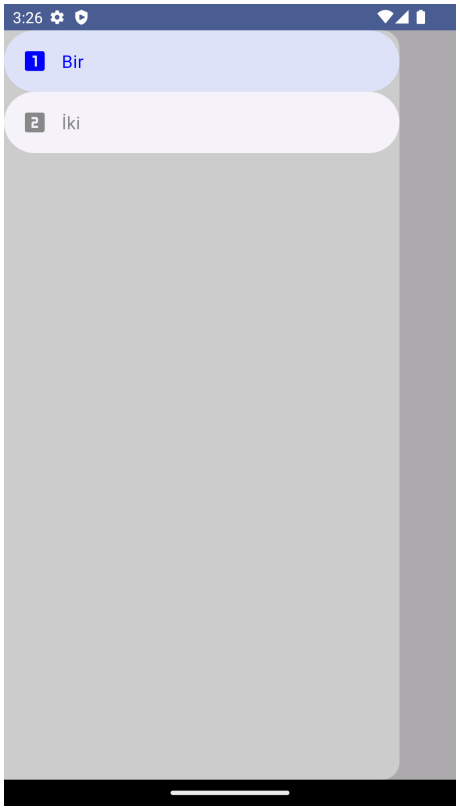
```
@Composable
fun SayfaBir() {
    Column(
        modifier = Modifier.fillMaxSize(),
        verticalArrangement = Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally
    ) { this: ColumnScope

        Text(text = "Sayfa Bir",fontSize = 50.sp)
    }
}
```

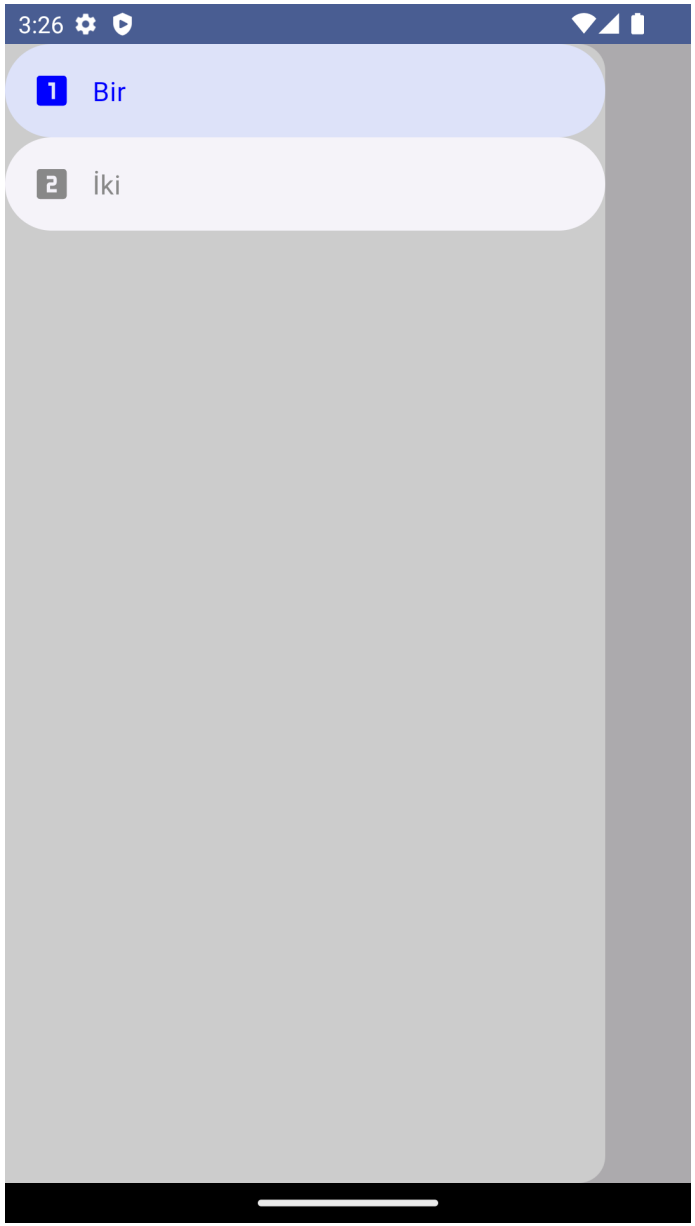
```
@Composable
fun SayfaIki() {
    Column(
        modifier = Modifier.fillMaxSize(),
        verticalArrangement = Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally
    ) { this: ColumnScope

        Text(text = "Sayfa İki",fontSize = 50.sp)
    }
}
```

Drawer

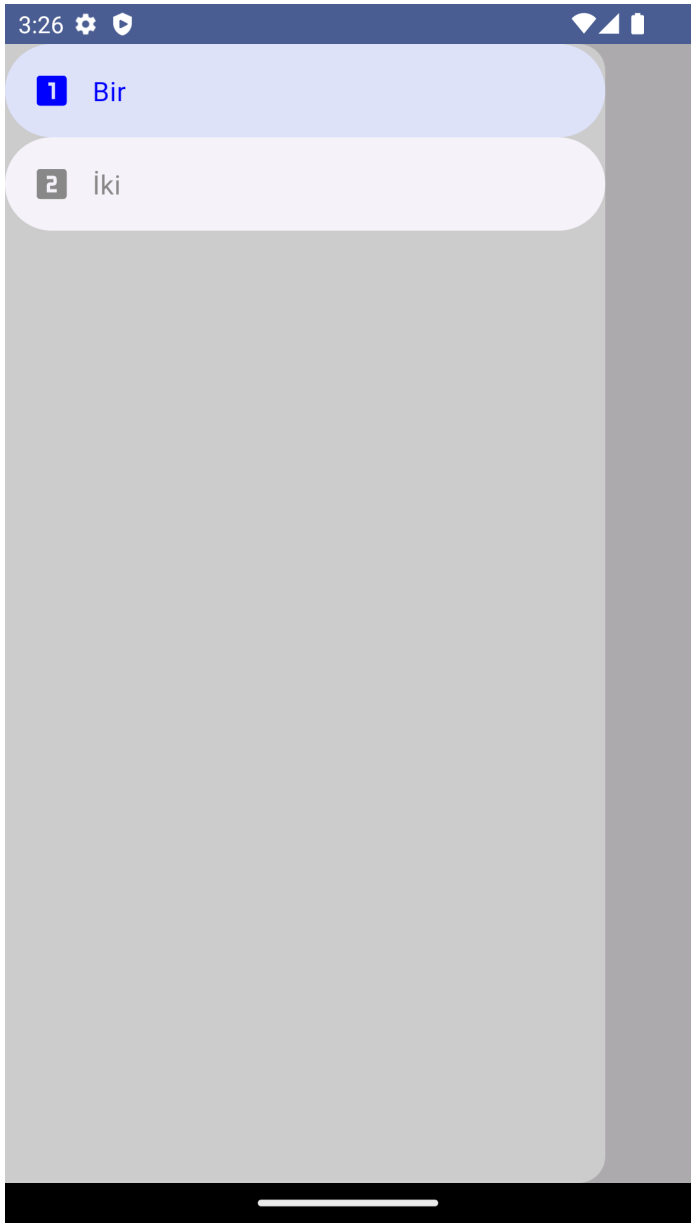
Drawer

- Drawer aynı ekran üzerinde birden fazla sayfa göstermek için kullandığımız bir yapıdır.
- Parmak hareketine duyarlı şekilde açılır ve kapanabilir.
- Geri tuşu ile kapatılabilir.

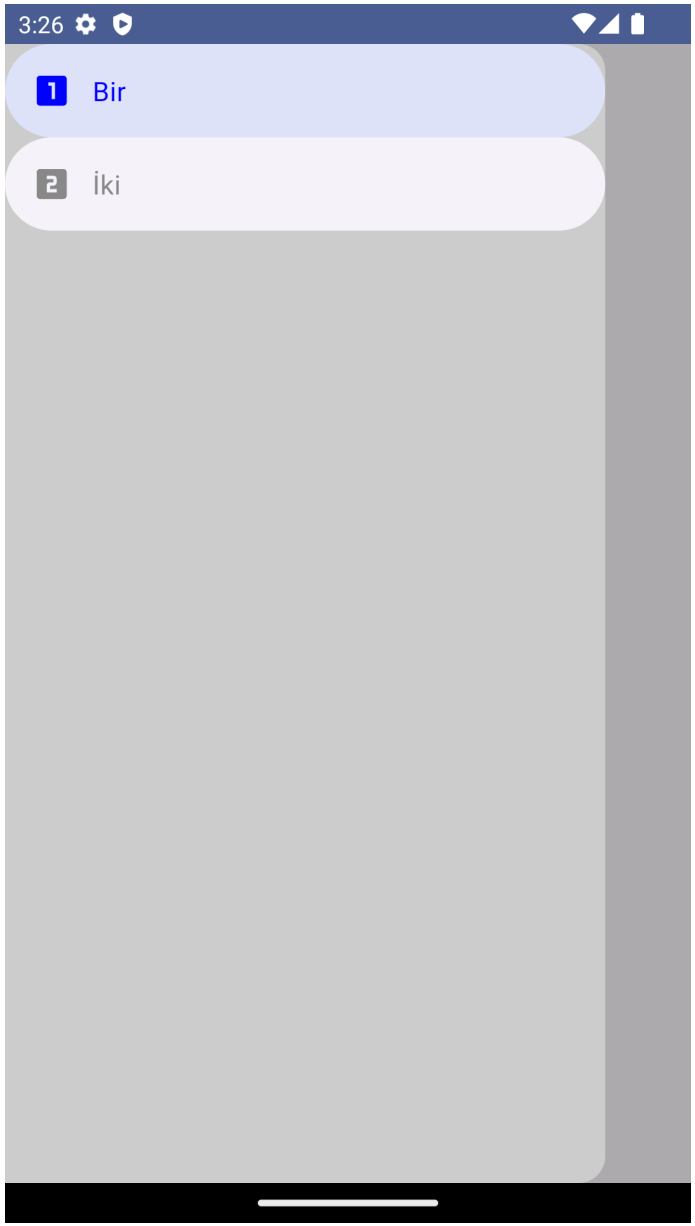





```
@SuppressLint("UnusedMaterial3ScaffoldPaddingParameter")
@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun DrawerSayfa() {
    val items = listOf("Bir", "İki")
    val secilenItem = remember { mutableStateOf( value: 0) }
    val drawerState = rememberDrawerState(DrawerValue.Closed)
    val scope = rememberCoroutineScope()
    ModalNavigationDrawer(
        drawerState = drawerState,
        drawerContent = {
            ModalDrawerSheet(drawerContainerColor = Color.LightGray) { this: ColumnScope //Arkaplan rengi
                items.forEachIndexed { index, item ->
                    NavigationDrawerItem(
                        selected = secilenItem.value == index ,
                        onClick = {
                            secilenItem.value = index
                            scope.launch { drawerState.close() }
                        },
```




```
                    icon = {
                        when (item) {
                            "Bir" -> Icon(painter = painterResource(id = R.drawable.resim1),
                                contentDescription = "")
                            "İki" ->Icon(painter = painterResource(id = R.drawable.resim2),
                                contentDescription = "")
                        }
                    },
                    label = { Text(text = item) },
                    colors = NavigationDrawerItemDefaults.colors(
                        selectedIconColor = Color.Blue,//Seçili icon rengi
                        unselectedIconColor = Color.Gray,//Seçili olmayan icon rengi
                        selectedTextColor = Color.Blue,//Seçili yazı rengi
                        unselectedTextColor = Color.Gray,//Seçili olmayan yazı rengi
                    )
                )
            }
        }
    },
    content = {
        if(secilenItem.value == 0){ SayfaBir() }
        if(secilenItem.value == 1){ SayfaIki() }
    }
)
```




```
    content = {
        if(secilenItem.value == 0){ SayfaBir() }
        if(secilenItem.value == 1){ SayfaIki() }
    }
)
```

```
val activity = (LocalContext.current as Activity)

BackHandler(onBack = {
    if(drawerState.isOpen){
        scope.launch { drawerState.close() }
    }else{
        activity.finish()
    }
})
```

Geri tuşu kodlaması

Sayfalar

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

        Text(text = "Sayfa Bir",fontSize = 50.sp)
    }
}
```

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

        Text(text = "Sayfa İki",fontSize = 50.sp)
    }
}
```

Teşekkürler...

kasım-adalan

kasimadalan@gmail.com

kasimadalan