

Jetpack Compose ile Android Uygulama Geliştirme Kursu

Çalışma Yapısı

Kasım ADALAN

Elektronik ve Haberleşme Mühendisi

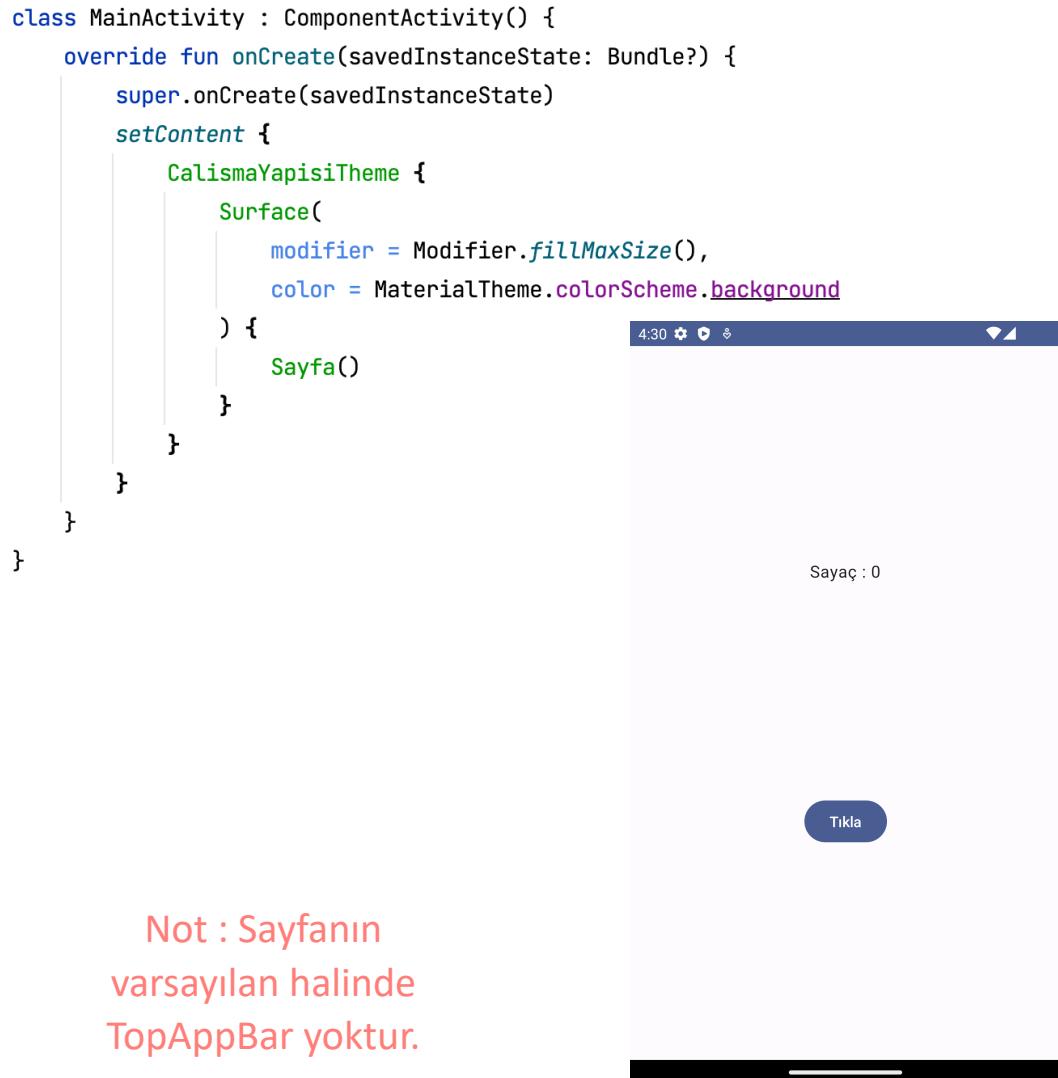
Android - IOS Developer and Trainer

Eğitim İçeriği

- Uygulama Sayfa Yapısı
- State Yapısı
- Uygulama Sayfası Oluşturma
- Sayfalar Arası Geçiş
- Back Stack
- Geri Dönüş Tuşu Kullanımı
- Sayfalar Arası Veri Transferi
- Yaşam Döngüsü
- Uygulama APK'sı oluşturma
- Google Play Üzerinde Yayınlama

Uygulama Sayfa Yapısı

Genel Yapı



```
@Preview(showBackground = true)  
@Composable  
fun GreetingPreview() {  
    CalismaYapisiTheme {  
        Sayfa()  
    }  
}  
  
@Composable  
fun Sayfa() {  
    val sayac = remember { mutableStateOf( value: 0 ) }  
  
    Column(  
        modifier = Modifier.fillMaxSize(),  
        verticalArrangement = Arrangement.SpaceEvenly,  
        horizontalAlignment = Alignment.CenterHorizontally  
    ) {  
        this: ColumnScope  
        Text(text = "Sayaç : ${sayac.value}")  
  
        Button(onClick = {  
            sayac.value = sayac.value + 1  
        }) {  
            this: RowScope  
            Text(text = "Tıkla")  
        }  
    }  
}
```

*Activity sınıfı,
bütün sayfaların
alt yapısını temsil
eder.*



```
class MainActivity : ComponentActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {
```

*Uygulama içinde
gösterilecek
temaları,sayfaları
bu metod içinde
belirtmeliyiz.*

*Uygulama
çalıştırıldığından ilk
burası çalışır.*



```
        super.onCreate(savedInstanceState)  
        setContent {
```

```
            CalismaYapisiTheme {
```

```
                Surface(
```

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

```
                    color = MaterialTheme.colorScheme.background
```

```
                ) {
```

```
                    Sayfa()
```

```
                }
```

*@Composable özelliği olan
Uygulama sayfasını temsil
eden metod buraya
uygulamada çalıştırılmak için
yazılır.*

Uygulama ismi ile oluşan tema tanımlaması

Tasarımın sayfaya

yayılmasını sağlar

*Sayfa arkaplan rengini belirleyen
tanımlama,tüm sayfaların arkaplan
rengini temsil eder.Eğer istedigimiz
sayfa arkaplan rengi değişimi
yapılacaksa o sayfada tekrar Surface
tanımlaması yapılabilir.Alt Surface
üste baskın çıkar.*

Sayfa içerik kodlamaları için Composable metod tanımlamasıdır. Bildiğimiz metod özelliklerine sahiptir. Sadece Composable notasyonu kullanılır.

Not : Composable metodlar Büyük Harf ile başlar.

```
@Composable  
fun Sayfa() {  
  
    val sayac = remember { mutableStateOf( value: 0 ) }  
  
    Column(  
        modifier = Modifier.fillMaxSize(),  
        verticalArrangement = Arrangement.SpaceEvenly,  
        horizontalAlignment = Alignment.CenterHorizontally  
    ) { this: ColumnScope  
        Text(text = "Sayac : ${sayac.value}")  
  
        Button(onClick = {  
            sayac.value = sayac.value + 1  
        }) { this: RowScope  
            Text(text = "Tıkla")  
        }  
    }  
}
```

Sayaç : 0

Sayfa içerik kodlaması

4:30 ⚡ 🌐 ⓘ

Sayaç : 0

Tıkla

Önizleme yapmak istediğiniz sayfayı bu yazarak test edebiliriz.

```
@Preview(showBackground = true)  
@Composable  
fun GreetingPreview() {  
    CalismaYapisiTheme {  
        Sayfa()  
    }  
}
```

Preview , emülatörde çalıştırmadan tasarımi görebilmek için gereklidir. İstenilen composable metoda eklenebilir.

Tıkla

State Yapısı

State Yapısı

- Arayüzde değişim yapılacak değişkenler için kullanılır.
- `MutableStateOf` yapısıyla oluşturulur ve varsayılan değer verilebilir.
- Oluşan değişkenin değerine ulaşmak ve değerini değiştirmek için `value` metodu kullanılır.
- `value` metodu ile değişkende değişim yapıldığında arayüz yenilenmektedir ve arayüzdeki tüm görsel nesne içerikleri güncellenmiş olur.

```
val sayac = remember { mutableStateOf( value: 0 ) }
```

Değişken Adı

Varsayılan
Değeri

```
@Composable
fun Sayfa() {
    val sayac = remember { mutableStateOf( value: 0 ) }

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

        Button(onClick = {
            sayac.value = sayac.value + 1
        }) { this: RowScope
            Text(text = "Tıkla")
        }
    }
}
```

Klasik Yöntem ve State Yapısı Karşılaştırması.

```
class MainActivity : AppCompatActivity() {  
  
    var sayac = 0  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)  
  
        textView.text = "Sayac : $sayac"  
  
        button.setOnClickListener { it: View!  
            sayac = sayac + 1  
            textView.text = "Sayac : $sayac"  
        }  
    }  
}
```

Uygulama ilk açıldığı anda textView içinde sayıç değeri 0 görünür. Butona tıklanıldığı anda sayıç artar ve sayıç değeri button metodu içinde anlık olarak arayüzde görünmesi için textView içine aktarılır.

```
@Composable  
fun Sayfa() {  
    val sayac = remember { mutableStateOf( value: 0 ) }  
  
    Column(  
        modifier = Modifier.fillMaxSize(),  
        verticalArrangement = Arrangement.SpaceEvenly,  
        horizontalAlignment = Alignment.CenterHorizontally  
    ) { this: ColumnScope  
        Text(text = "Sayac : ${sayac.value}")  
  
        Button(onClick = {  
            sayac.value = sayac.value + 1  
        }) { this: RowScope  
            Text(text = "Tıkla")  
        }  
    }  
}
```

Uygulama ilk açıldığı anda textView içinde sayıç değeri 0 görünür. Butona tıklanıldığı anda sayıç artar ve .value metodı ile sayıç değeri kodlama içinde var olan her yerdeki değerini değiştirir ve kodları günceller. Sayac arttırma başka bir yerde , sayacı okuma başka bir yerde gerçekleşir. Bu durum state özelliği ile oluşmaktadır.

Yeni Sayfa Oluşturma

Yeni Sayfa Oluşturma

- Yeni bir sayfa oluşturmak için `@Composable` özelliği olan bir metod oluşturmak yeterlidir.

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

        Text(text = "Anasayfa", fontSize = 50.sp)

        Button(onClick = {

            }) { this: RowScope
                Text(text = "Sayfa A'ya Git")
            }
    }
}
```

Sayfa tasarım
ve
kodlama içeriği

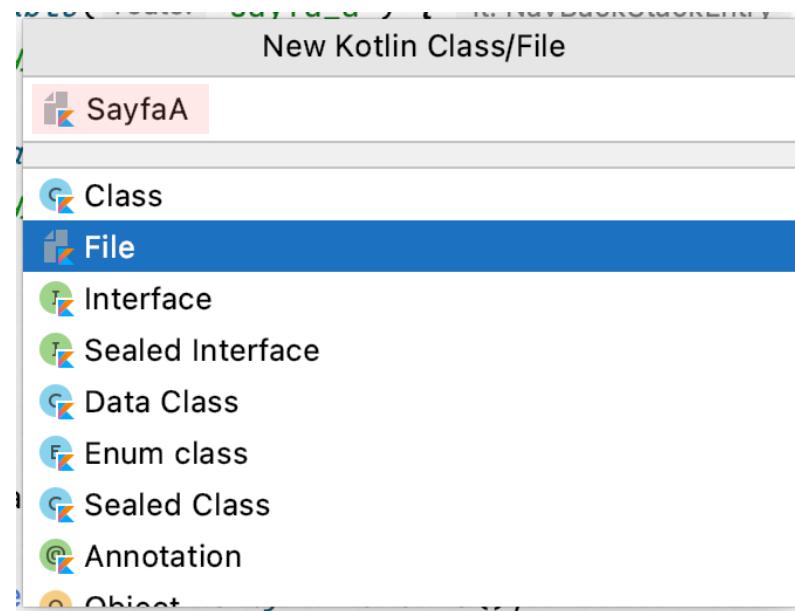
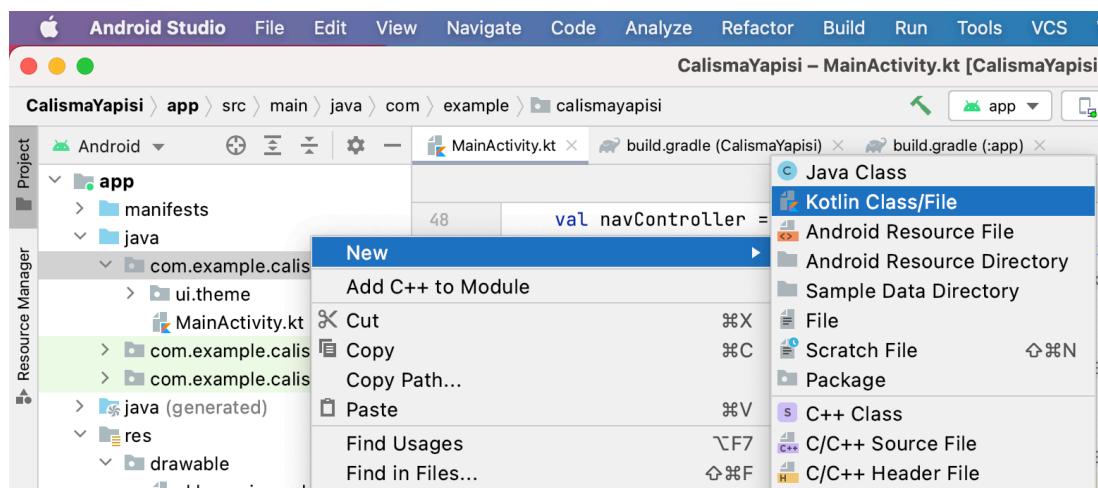
Yöntem 1 : Yeni Sayfa Oluşturma

- MainActivity kodlamasının bulduğu kotlin dosyasında oluşturabiliriz.

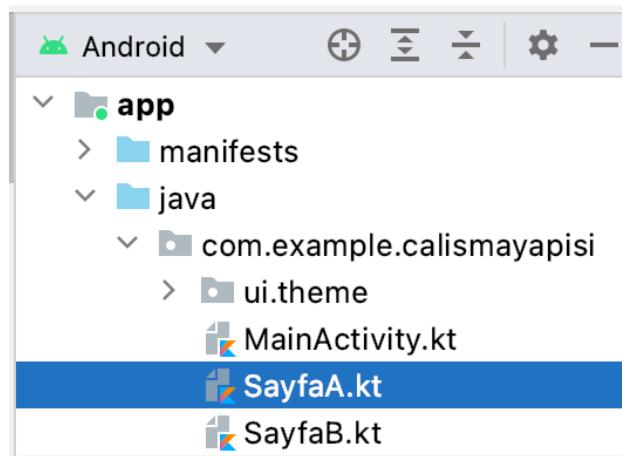
```
class MainActivity : ComponentActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContent {  
            CalismaYapisiTheme {  
                Surface(color = MaterialTheme.colors.background) {  
                    Anasayfa()  
                }  
            }  
        }  
    }  
  
    @Preview(showBackground = true)  
    @Composable  
    fun DefaultPreview() {  
        CalismaYapisiTheme {  
            Anasayfa()  
        }  
    }  
  
    @Composable  
    fun Anasayfa() {  
        Column(  
            modifier = Modifier.fillMaxSize(),  
            verticalArrangement = Arrangement.SpaceEvenly,  
            horizontalAlignment = Alignment.CenterHorizontally  
        ) {  
            this: ColumnScope  
  
            Text(text = "Anasayfa", fontSize = 50.sp)  
  
            Button(onClick = {  
                this: RowScope  
                Text(text = "Sayfa A'ya Git")  
            })  
        }  
    }  
}
```

Yöntem 2 : Yeni Sayfa Oluşturma

- Ayrı bir kotlin dosyası oluşturup içinde @Composable özellikli sayfa tanımlamaktır.
- Bu yöntem kodlama olarak daha verimlidir.
- Sayfaları modüler olarak parçalامış oluruz ve düzenli bir yapı oluşturur.



Yöntem 2 : Yeni Sayfa Oluşturma



```
package com.example.calismayapisi

import ...

@Composable
fun SayfaA() {

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

        Text(text = "Sayfa A", fontSize = 50.sp)

        Button(onClick = {

        }) { this: RowScope
            Text(text = "Sayfa B'ye Git")
        }
    }
}
```

Sayfalar Arası Geçiş

Sayfalar Arası Geçiş Ön Kurulum

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

Sayfalar Arası Geçiş

- Sayfa geçişinde kullanılacak navController nesnesini sayfaya parametre olarak almamız gereklidir.
- Bu nesne sayesinde sayfa geçişlerini çalıştırabiliriz.

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

        Text(text = "Anasayfa", fontSize = 50.sp)

        Button(onClick = {

        }) { this: RowScope
            Text(text = "Sayfa A'ya Git")
        }
    }
}
```

Sayfalar Arası Geçiş

- Sayfa geçişlerini temsil edecek Composable bir metod oluşturmalıyız.

Sayfa geçişlerinde
çalışacak sayfanın
Key tanımlaması

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

Sayfa geçişlerinde çalışacak sayfanın Key tanımlaması

Sayfa geçişlerini yapabilmek için gerekli navController.

İlk açılacak sayfanın tanımlaması.

Sayfanın key ifadesi yazılır.

Sayfa geçişlerinde çalışacak olan Sayfa tanımlaması

Sayfa içinde geçişleri çalıştırabilme (tetikleyebilmek) için sayfaya navController nesnesi parametre olarak gönderilir.

Sayfalar Arası Geçiş

- Sayfa geçişleri için oluşturulan metoduyu MainActivity içine ekleme.
- Uygulama ilk açıldığı anda sayfa geçişleri metodundaki startDestination yani varsayılan anasayfa açılır.

```
class MainActivity : ComponentActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContent {  
            CalismaYapisiTheme {  
                Surface(color = MaterialTheme.colors.background) {  
                    SayfaGecisleri()  
                }  
            }  
        }  
    }  
  
    @Preview(showBackground = true)  
    @Composable  
    fun DefaultPreview() {  
        CalismaYapisiTheme {  
        }  
    }  
}
```

Sayfalar Arası Geçiş

- navController nesnesi ile geçişini çalışma.

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

        Text(text = "Anasayfa", fontSize = 50.sp)

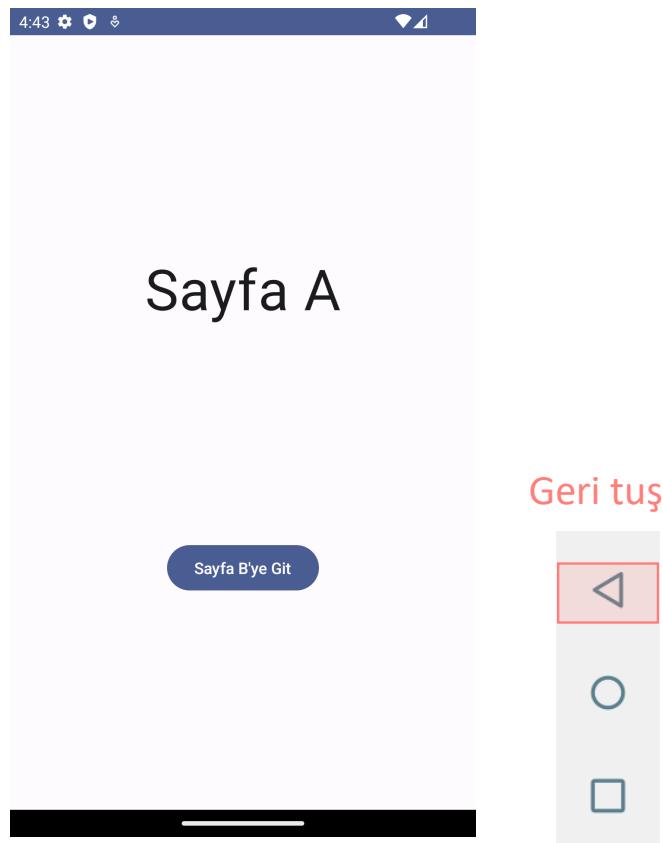
        Button(onClick = {
            navController.navigate(route: "sayfa_a") ←
        }) { this: RowScope
            Text(text = "Sayfa A'ya Git")
        }
    }
}
```

Sayfa geçişini
çalıştırmak için
sayfanın geçiş keyini
yazmalıyız.

Gidilmek İstenen
Sayfa

Otomatik Geri Dönüş

- Geri gelmek için geri tuşuna bastığımızda veya soldan kaydırma yaptığımızda bir önceki sayfaya otomatik olarak geri dönebiliriz.



Sayfalar Arası Geçiş

```
Button(onClick = {  
    navController.navigate( route: "sayfa_a")  
}) { this: RowScope  
    Text(text = "Sayfa A'ya Git")  
}
```

4:46 ⚡ 📱 🔋

```
Button(onClick = {  
    navController.navigate( route: "sayfa_b")  
}) { this: RowScope  
    Text(text = "Sayfa B'ye Git")  
}
```

4:43 ⚡ 📱 🔋

4:49 ⚡ 📱 🔋

Anasayfa

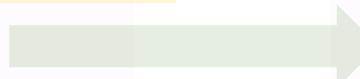
Sayfa A

Sayfa B

navController.popBackStack()
*Geri tuşunun işlevini yapar.
İstersek button altında kullanabiliriz.*

Sayfa A'ya Git

Sayfa B'ye Git



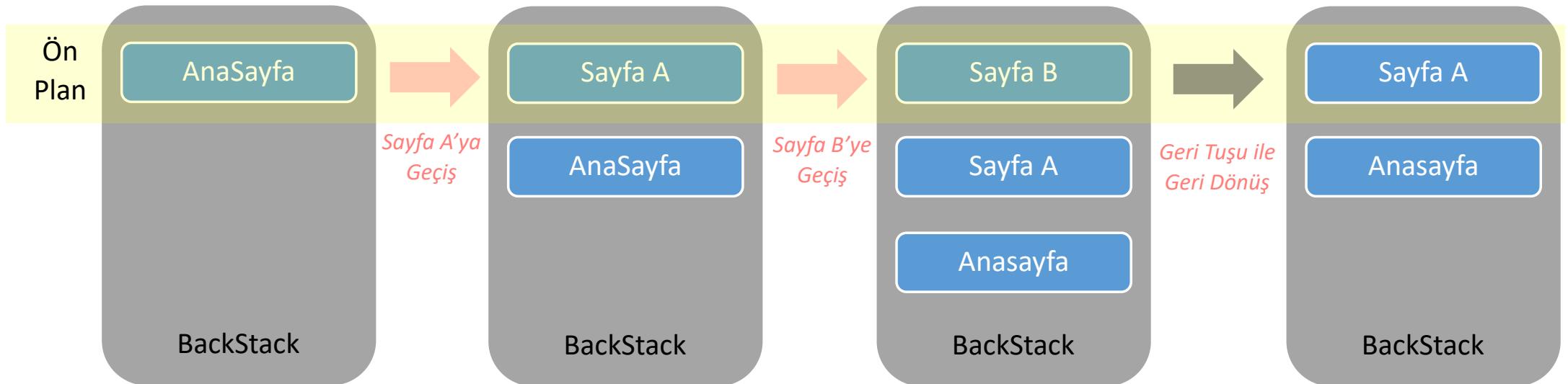
BACK STACK

BACK STACK

- **Task** nedir daha iyi anlamak için web tarayıcının çalışmasını örnek olarak alalım.
- **Task** sekme, etkileşim pencere (**Activity**) ve link **Intent** varsayıyalım. İstediğiniz zaman **Back**(Geri) butona basıp geri dönebilirsiniz ama ileriye dönemezsınız çünkü ilerdeki sayfa siliniyor.
- İleriye dönmek için yine linke (**Intent**) basmamız gerekecek.
- Sonuçta web tarayıcının ve Android arasındaki benzerliği şöyle gösterebiliriz:
 - Tarayıcı – Android
 - Sekme – **Task** (Görev)
 - Pencere – Etkileşim (**Activity**)
 - Link – **Intent**

BACK STACK

Sayfa B
BackStackten
Silinir



Bu işlemler otomatik olarak gerçekleşir.
İstenirse sayfa geçişinde istediğimiz sayfayı back stackten silebiliriz.

```
Navigator.pushReplacement(context, MaterialPageRoute(builder: (context) => SayfaA() ));  
//Android içindeki Finish metodunu gibi çalışır  
//sayfaya geçiş yapıldığında back stackten kendisini siler
```

Sayfayı Back Stackten Silme

- Android içindeki Finish metodu gibi çalışır
- Bu yapı ile sayfaya geçiş yapıldığında back stackten kendisini siler.
- Kendisini sildiği için geçiş yapıldığı sayfadan geri tuşu ile bu sayfaya gelinemez çünkü back stackten silinmiştir.

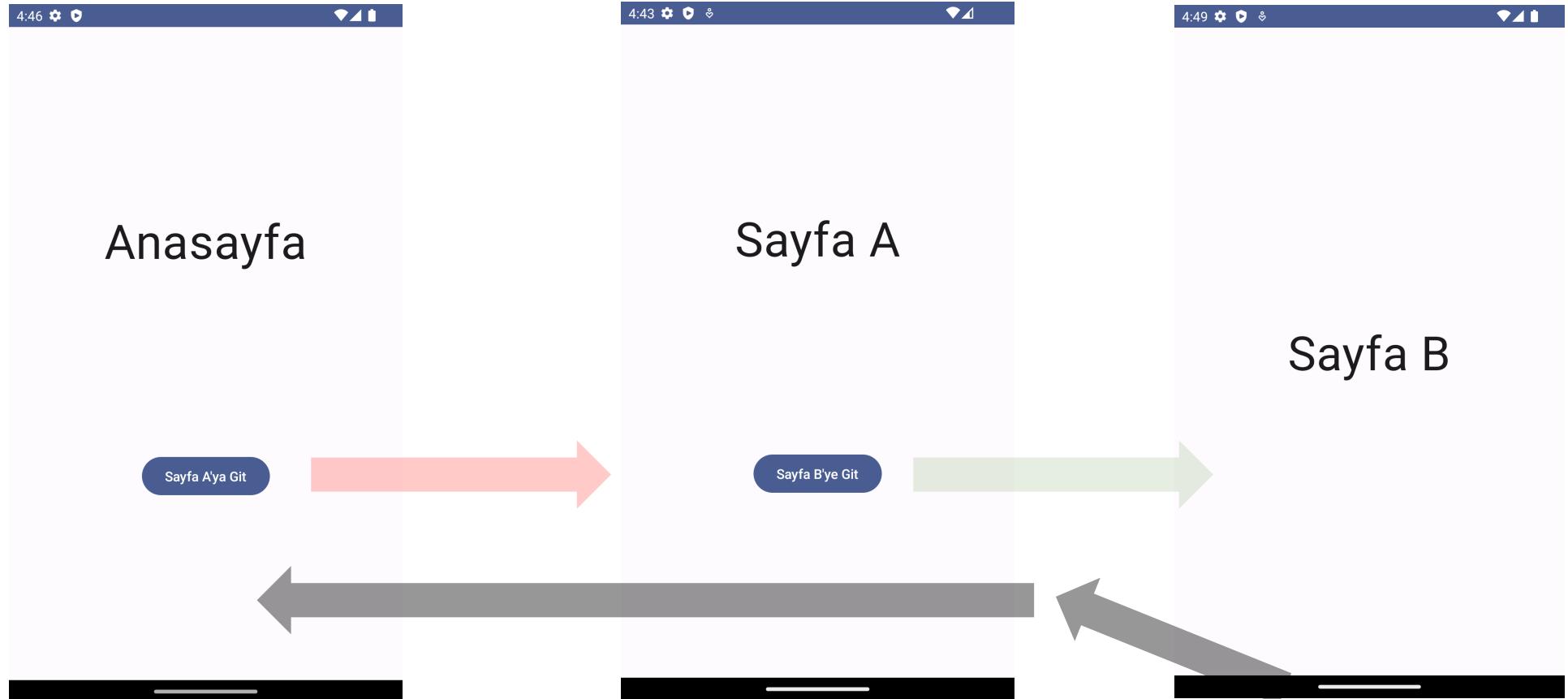
```
@Composable
fun SayfaA(navController: NavController) {
    Column(
        modifier = Modifier.fillMaxSize(),
        verticalArrangement = Arrangement.SpaceEvenly,
        horizontalAlignment = Alignment.CenterHorizontally
    ) { this: ColumnScope
        Text(text = "Sayfa A", fontSize = 50.sp)

        Button(onClick = {
            navController.navigate(route: "sayfa_b") { this: NavOptionsBuilder
                popUpTo(route: "sayfa_a") { inclusive = true }
            }
        }) { this: RowScope
            Text(text = "Sayfa B'ye Git")
        }
    }
}
```

Back stackten silinmek istenen sayfanın key'i buraya yazılır.

Back Stack

```
navController.navigate( route: "sayfa_b") { this: NavOptionsBuilder  
    | popUpTo( route: "sayfa_a") { inclusive = true }  
}
```



Geri Dönüş Tuşu

Geri Dönüş Tuşu

- Geri tuşu varsayılan olarak önceki sayfaya geçiş yapmamızı sağlar.
- İstersen geri tuşunu kontrol edebiliriz.
- Bunun için BackHandler metodunu kullanabiliriz.

```
@Composable
fun SayfaB(navController: NavController) {

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

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

    //Geri tuşu dinleme
    BackHandler(onBack = {
        Log.e( tag: "SayfaB", msg: "Geri tuşuna basıldı.")
    })
}
```

Uygulamayı Kapatma

- Kodlama ile uygulamayı kapatabiliriz.

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

```
BackHandler(onBack = {
    Log.e( tag: "SayfaB", msg: "Geri tuşuna basıldı.")
    activity.finish()
})
```

Sayfalar Veri Transferi

Sayfalar Arası Veri Transferi

- Sayfalar arası veri transferi için geçiş yapılacak sayfaya değişkenleri parametre olarak tanımlamalıyız.

```
@Composable
fun SayfaA(navController: NavController, gelenIsim:String, gelenYas:Int, gelenBoy:Float, gelenBekarMi:Boolean) {
    Column(
        modifier = Modifier.fillMaxSize(),
        verticalArrangement = Arrangement.SpaceEvenly,
        horizontalAlignment = Alignment.CenterHorizontally
    ) { this: ColumnScope

        Text(text = "Sayfa A", fontSize = 50.sp)

        Text(text = gelenIsim)
        Text(text = gelenYas.toString())
        Text(text = gelenBoy.toString())
        Text(text = gelenBekarMi.toString())

        Button(onClick = {
            navController.navigate( route: "sayfa_b"){ this: NavOptionsBuilder
                popUpTo( route: "sayfa_a") { inclusive = true }
            }
        }) { this: RowScope
            Text(text = "Sayfa B'ye Git")
        }
    }
}
```

Sayfalar Arası Veri Transferi

- Veriyi transfer ederken hangi verileri hangi sayfaya göndereceğimizi yani detayları sayfa geçiş metodunda belirtmeliyiz.

Gönderilmek istenen veri için id tanımlanır ve {} içinde

yazılmalıdır. Birden fazla veri göndermek için / işaretini kullanmalıyız.

Gönderilecek verilerin tür tanımlaması

Gönderilecek sayfanın parametresine verileri aktarmak için arguments içinde verilerin id ile alınması.

Gönderilecek sayfanın parametresine verilerin eklenmesi

```
@Composable
fun SayfaGecisleri() {
    val navController = rememberNavController()
    NavHost(navController = navController, startDestination = "anasayfa") {
        composable(route: "anasayfa") { it: NavBackStackEntry
            Anasayfa(navController = navController)
        }
        composable(
            route: "sayfa_a/{isim}/{yas}/{boy}/{bekarMi}",
            arguments = listOf(
                navArgument(name: "isim") { type = NavType.StringType },
                navArgument(name: "yas") { type = NavType.IntType },
                navArgument(name: "boy") { type = NavType.FloatType },
                navArgument(name: "bekarMi") { type = NavType.BoolType }
            )
        ) { it: NavBackStackEntry
            val isim = it.arguments?.getString(key: "isim")!!
            val yas = it.arguments?.getInt(key: "yas")!!
            val boy = it.arguments?.getFloat(key: "boy")!!
            val bekarMi = it.arguments?.getBoolean(key: "bekarMi")!!
            SayfaA(navController = navController, isim, yas, boy, bekarMi)
        }
        composable(route: "sayfa_b") { it: NavBackStackEntry
            SayfaB(navController = navController)
        }
    }
}
```

Sayfalar Arası Veri Transferi

- Verilerin sayfa içinden gönderilmesi, geçiş yapılan sayfaya gönderilmesi

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

        Text(text = "Anasayfa", fontSize = 50.sp)

        Button(onClick = {
            navController.navigate( route: "sayfa_a/ahmet/18/1.78f/true")
        }) { this: RowScope
            Text(text = "Sayfa A'ya Git")
        }
    }
}
```

Sayfalar Veri Transferi

-

Nesne Tabanlı

Sayfalar Arası Veri Transferi - Nesne Tabanlı

- Sayfalar arası veri transferi Serializable veya Parceable interfaceleri ile yapılabilirken.
- Jetpack compose bu iki yöntemde sorunlar çıkarabiliyor.
- Bundan dolayı nesneyi json formatına dönüştürüp transfer edebiliyoruz.
- Bu işlem için Gson kütüphanesini projemize eklememiz gereklidir.

```
dependencies {  
  
    implementation 'androidx.core:core-ktx:1.3.2'  
    implementation 'androidx.appcompat:appcompat:1.2.0'  
    implementation 'com.google.android.material:material:1.3.0'  
    implementation "androidx.compose.ui:ui:$compose_version"  
    implementation "androidx.compose.material:material:$compose_version"  
    implementation "androidx.compose.ui:ui-tooling:$compose_version"  
    implementation 'androidx.lifecycle:lifecycle-runtime-ktx:2.3.1'  
    implementation 'androidx.activity:activity-compose:1.3.0-alpha06'  
    testImplementation 'junit:junit:4.+'  
    androidTestImplementation 'androidx.test.ext:junit:1.1.2'  
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.3.0'  
    androidTestImplementation "androidx.compose.ui:ui-test-junit4:$compose_version"  
  
    implementation "androidx.navigation:navigation-compose:2.4.0-alpha02"//6  
  
    //Nesne transferi için gson  
    implementation "com.google.code.gson:gson:2.8.6"  
}
```

```
implementation "com.google.code.gson:gson:2.8.6"
```

Sınıf Oluşturma

```
data class Kisiler(var isim:String,  
                  var yas:Int,  
                  var boy:Float,  
                  var bekarMi:Boolean) {  
}
```

Sayfa Geçiş Tanımlaması

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

json formatında çalışacağımız için String türü seçiyoruz.

String bilgisi Kisiler nesnesine dönüştürüyoruz.

Verinin Gönderilmesi

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

        Text(text = "Anasayfa", fontSize = 50.sp)

        Button(onClick = {
            val kisi = Kisiler( isim: "ahmet", yas: 18, boy: 1.78f, bekarMi: true)

            val kisiJson = Gson().toJson(kisi) ← Kisiler nesnesini json formatına çeviriyoruz.
            navController.navigate( route: "sayfa_a/$kisiJson")
        }) { this: RowScope
            Text(text = "Sayfa A'ya Git")
        }
    }
}
```

Verinin Alınması

```
@Composable
fun SayfaA(navController: NavController, gelenNesne:Kisiler) {

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

        Text(text = "Sayfa A", fontSize = 50.sp)

        Text(text = gelenNesne.isim)
        Text(text = gelenNesne.yas.toString())
        Text(text = gelenNesne.boy.toString())
        Text(text = gelenNesne.bekarMi.toString())

        Button(onClick = {
            navController.navigate( route: "sayfa_b"){ this: NavOptionsBuilder
                popUpTo( route: "sayfa_a") { inclusive = true }
            }
        }) { this: RowScope
            Text(text = "Sayfa B'ye Git")
        }
    }
}
```

Yaşam Döngüsü

Yaşam Döngüsü

- Yaşam döngüsü uygulama sayfasının çalışmasını takip ettiğimiz metodlardır.
- LaunchEffect, SideEffect , DisposableEffect olmak üzere çeşitli metodlar yer almaktadır.
- **LaunchedEffect** : Sayfa her görüntülendiğinde (Uygulama sayfası ilk kez açıldığında veya başka sayfadan geri gelindiğinde) bir kere çalışır.
- **SideEffect** : Sayfa görüntülendiğinde (Uygulama sayfası ilk kez açıldığında veya başka sayfadan geri gelindiğinde) ve arayüz state ile her yenilendiğinde çalışır.
- **DisposableEffect** : Sayfadan ayrıldığımızda çalışır. Örnek : Geri tuşu ile uygulamadan çıkışma veya başka bir sayfaya geçerken sayfadan ayrıldığımızda çalışır.

Yaşam Döngüsü

Back Tuşuna Basıldığından

İlk Çalıştığından



Sayaç : 0

Arttır

Geçiş Yap



Sayaç : 0

Arttır

Geçiş Yap

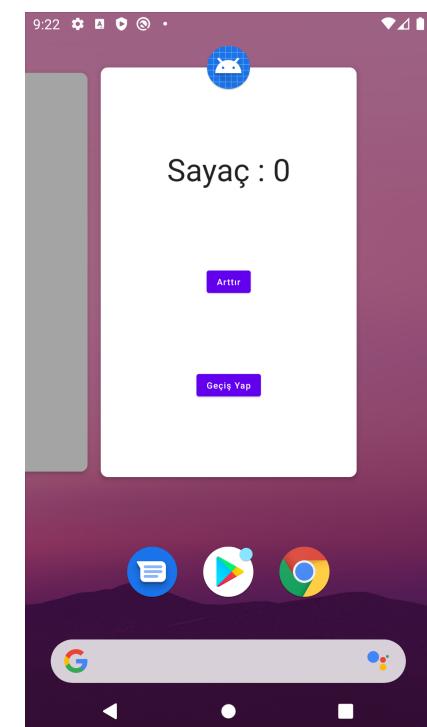
1. DisposableEffect() metodu



1. SideEffect() metodu

2. LaunchedEffect() metodu

Arka Plandan Tekrar Çağrıldığında



Sayaç : 0

Arttır

Geçiş Yap

1. SideEffect() metodu

2. LaunchedEffect() metodu

Not : Yuvarlak ve Kare tuşlarında disposable çalışmaz.

Yaşam Döngüsü

İlk Çalıştığında



Sayaç : 2



Geçiş Yap



1. *SideEffect() metodu*

Arttır butonu ile sayac içeriği state özelliği ile her arttığında arayüz yenilenir ve her arayüz yenilenmesinde SideEffect metodu çalışır.

Yaşam Döngüsü

1

Sayfa Geçişi olduğunda

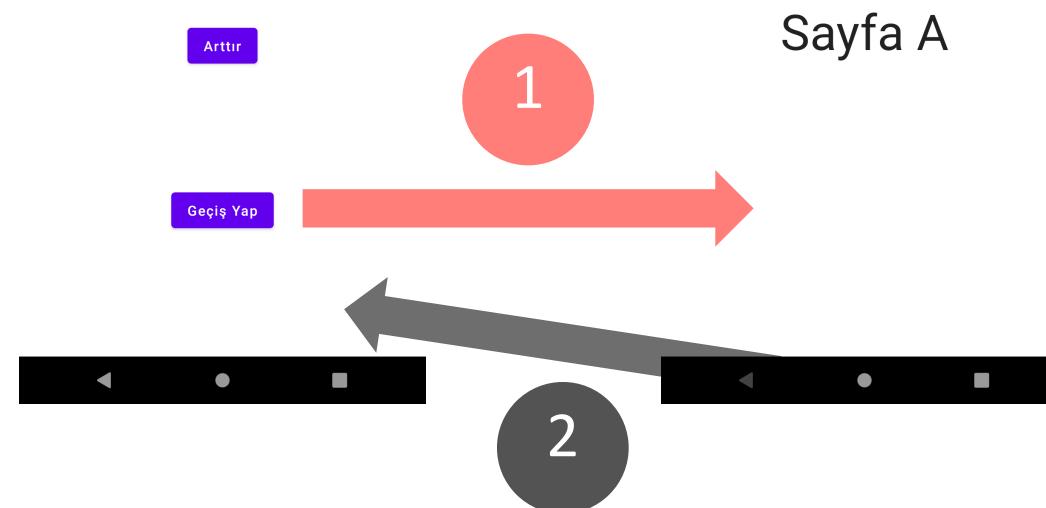
1. *DisposableEffect()* metodu

2

Sayfaya geri gelindiğinde

1. *SideEffect()* metodu
2. *LaunchedEffect()* metodu

Sayaç : 0



Final Kod



Sayfa A

```
@Composable
fun SayfaA() {
    Column(
        modifier = Modifier.fillMaxSize(),
        verticalArrangement = Arrangement.SpaceEvenly,
        horizontalAlignment = Alignment.CenterHorizontally
    ) { this: ColumnScope
        Text(text = "Sayfa A", fontSize = 50.sp)
    }
}
```



Sayaç : 0

ArttırGeçiş Yap

```
@Composable
fun Anasayfa(navController: NavController) {
    val sayac = remember { mutableStateOf( value: 0 ) }
    Column(
        modifier = Modifier.fillMaxSize(),
        verticalArrangement = Arrangement.SpaceEvenly,
        horizontalAlignment = Alignment.CenterHorizontally
    ) { this: ColumnScope
        Text(text = "Sayaç : ${sayac.value}",fontSize = 50.sp)
        Button(onClick = { sayac.value = sayac.value + 1 }) { Text(text = "Arttır") }
        Button(onClick = { navController.navigate( route: "sayfa_a" ) }) { Text(text = "Geçiş Yap") }
    }

    LaunchedEffect( key1: true){ this: CoroutineScope
        Log.e( tag: "Anasayfa", msg: "LaunchedEffect Çalıştı")
    }

    SideEffect {
        Log.e( tag: "Anasayfa", msg: "SideEffect Çalıştı")
    }

    DisposableEffect(Unit) { this: DisposableEffectScope
        onDispose {
            Log.e( tag: "Anasayfa", msg: "DisposableEffect Çalıştı")
        }
    }
}
```

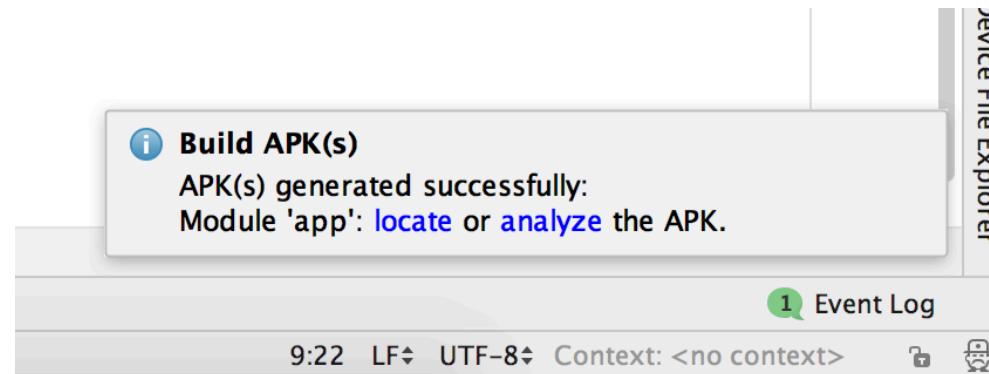
```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            YasamDongusuTheme {
                Surface(color = MaterialTheme.colors.background) {
                    SayfaGecisleri()
                }
            }
        }
    }
}

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

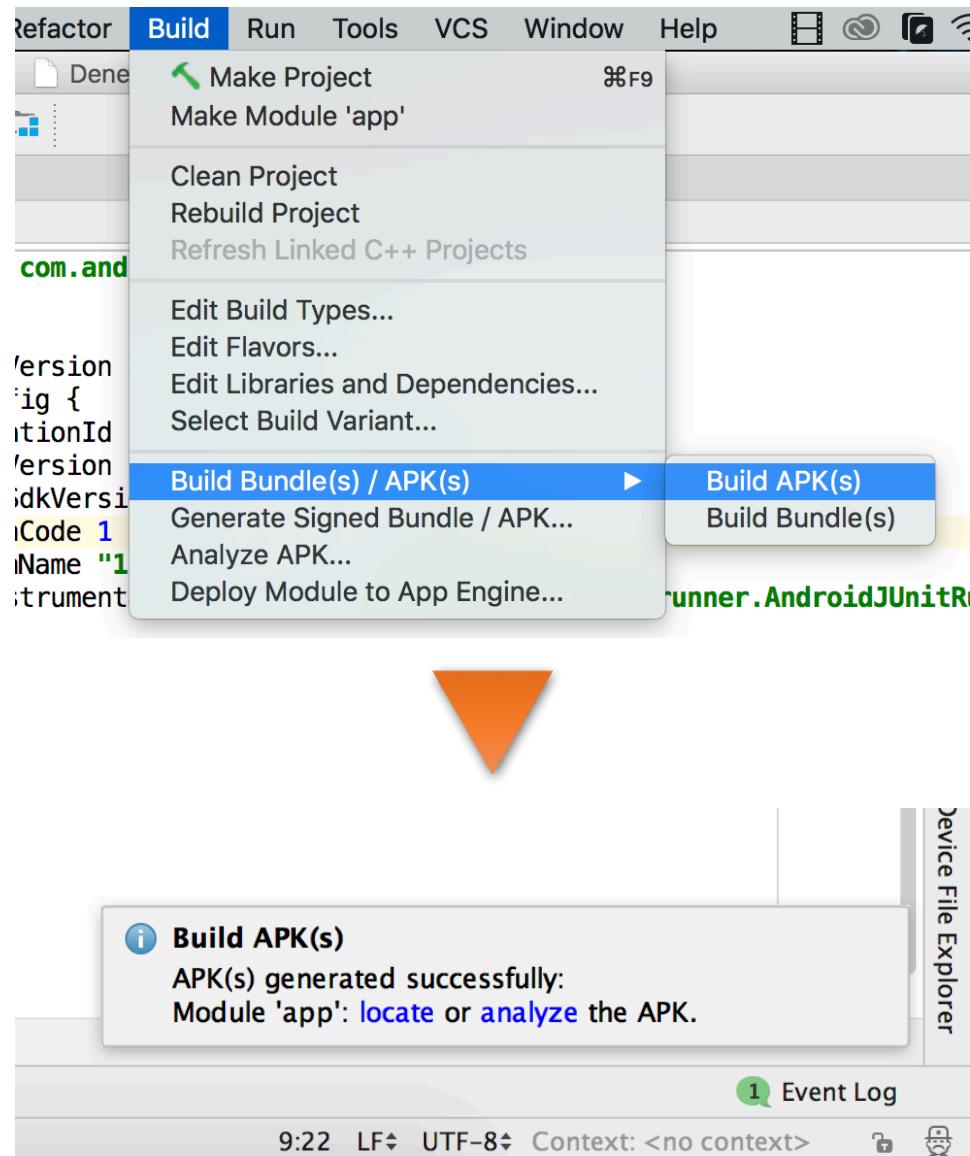
APK Oluşturma

Apk Oluşturma

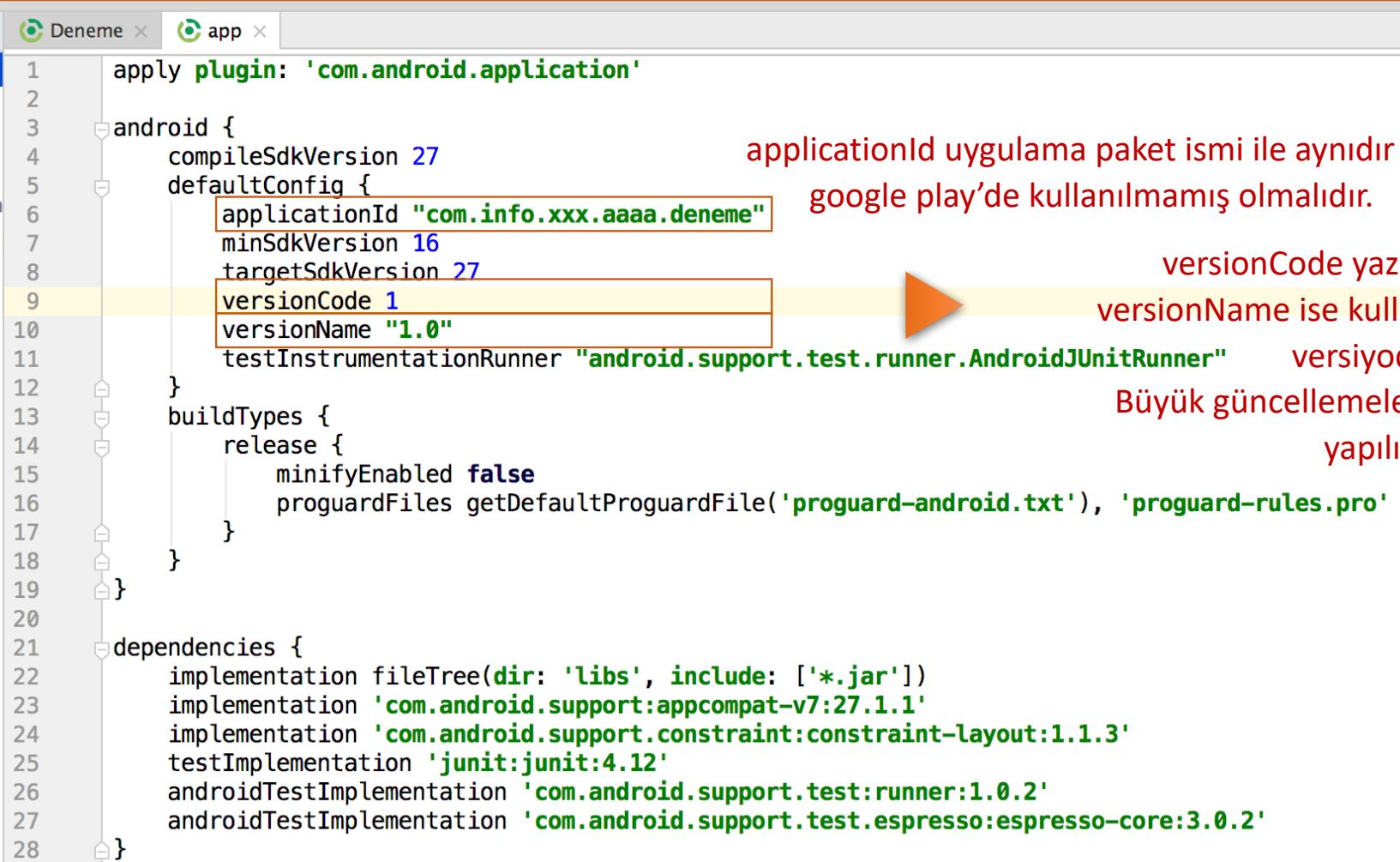
- **İmzalı apk** : Google play'e yüklemek için gerekli olan apk sürümü.
- **Normal apk** : Google play haricinde kullanmak için kullanılır.



Normal APK



İmzalı (Signed) APK



```
1 apply plugin: 'com.android.application'
2
3 android {
4     compileSdkVersion 27
5     defaultConfig {
6         applicationId "com.info.xxx.aaaa.deneme"
7         minSdkVersion 16
8         targetSdkVersion 27
9         versionCode 1
10        versionName "1.0"
11        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
12    }
13    buildTypes {
14        release {
15            minifyEnabled false
16            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
17        }
18    }
19 }
20
21 dependencies {
22     implementation fileTree(dir: 'libs', include: ['*.jar'])
23     implementation 'com.android.support:appcompat-v7:27.1.1'
24     implementation 'com.android.support.constraint:constraint-layout:1.1.3'
25     testImplementation 'junit:junit:4.12'
26     androidTestImplementation 'com.android.support.test:runner:1.0.2'
27     androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'
28 }
```

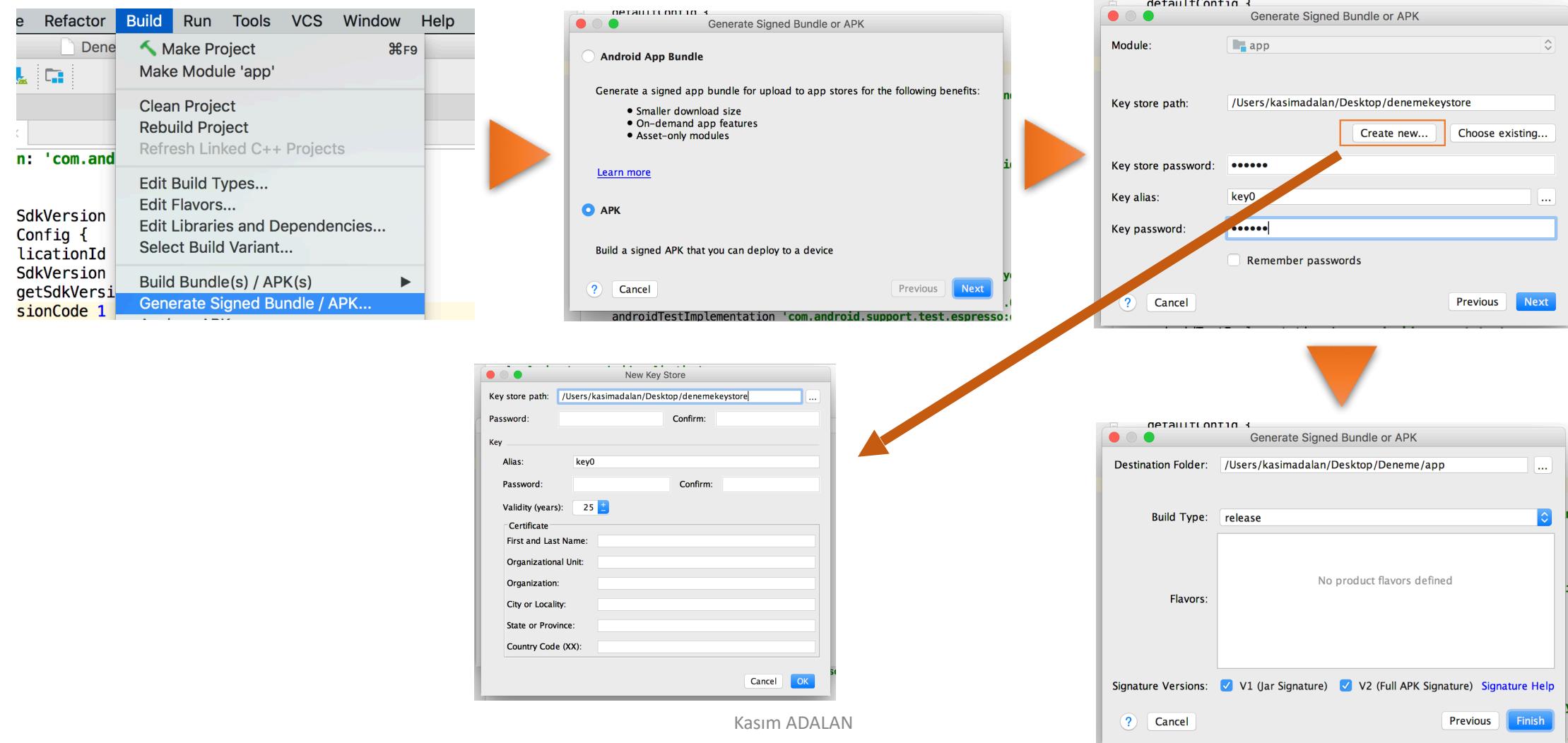
applicationId uygulama paket ismi ile aynıdır ve google play'de kullanılmamış olmalıdır.

versionCode yazılım sırasıdır.

versionName ise kullanıcının göreceği versiyodur.

Büyük güncellemelerde büyük artış yapılır.

İmzalı APK Oluşturma



Kasım ADALAN

Uygulamayı Google Play Üzerinde Yayınlama

Uygulamanın Google Play Yüklenmesi



Uygulamanın Google Play Yüklenmesi

The screenshot shows the Google Play Console interface. On the left, a sidebar lists various sections: 'Tüm uygulamalar', 'Hesap özetи', 'Uygulama sürümleri', 'Android Hazır Uygulamalar', 'Yapı kitaplığı', 'Cihaz kataloğu', 'Uygulama imzalama', 'Mağaza girişи' (highlighted with an orange border), 'İçerik derecelendirme', 'Fiyatlandırma ve dağıtım', and 'Uygulama içi ürünler', 'Çeviri hizmeti'. The main area is titled 'Mağaza girişи' and 'Ürün bilgileri'. It shows a draft application named 'Deneme Taslak' with a placeholder icon. The status bar indicates 'TURKCE - TR-TR' and 'Çevirileri yönet'. A red note in the center states: 'Google Play Console sayfasından uygulama paylaşmak için her yıl 25 dolar verilmelidir.' Below the note, there is a warning about data policies and a large blue button labeled 'TASLAĞI KAYDET'.

Google Play Console sayfasından uygulama paylaşmak
için her yıl 25 dolar verilmelidir.

Uygulama meta verileriyle ilgili sık karşılaşılan bazı ihlallerden kaçınmak için lütfen [Meta Veri politikamıza](#) göz atın. Lütfen uygulamalarınızı göndermeden önce diğer tüm [program politikalarını](#) incelediğinizden de emin olun. Uygulamanız veya mağaza girişiniz Google Play Uygulama İnceleme ekibine [önceden bildirim için uygunsa](#) uygulamanızı yayımlamadan önce [bizimle iletişim kurun](#).

TASLAĞI KAYDET

Mağaza için Görüşellerin Hazırlanması

Ürün bilgileri TÜRKÇE – TR-TR Çeviri

2/8 ekran görüntüsü DOSYALARA

Yüksek çözünürlüklü simge * Varsayılan – Türkçe – tr-TR 512 x 512 32 bit PNG (alfa kanallı)	Öne Çıkan Grafik * Varsayılan – Türkçe – tr-TR 1024 g x 500 y JPG veya 24 bit PNG (alfa kanalsız)	Tanıtım Grafiği Varsayılan – Türkçe – tr-TR 180 g x 120 y JPG veya 24 bit PNG (alfa kanalsız)
		
TV Banner'i Varsayılan – Türkçe – tr-TR 1280 g x 720 y JPG veya 24 bit PNG (alfa kanalsız)	Daydream 360 derecelik stereoskopik resim Varsayılan – Türkçe – tr-TR 4096 g x 4096 y JPG veya 24 bit PNG (alfa kanalsız)	

İmzalı APK'nın yüklenmesi

The screenshot shows the Google Play Developer Console interface. On the left, there's a sidebar with various options: Hesap özeti, Uygulama sürümleri (selected), Android Hazır Uygulamalar, Yapı kitaplığı, Cihaz kataloğu, Uygulama imzalama, Mağaza girişи, İçerik derecelendirme, Fiyatlandırma ve dağıtım, and Uygulama içi ürünler. The main area is titled 'Yeni üretim sürümü' and shows two steps: 'Sürümü hazırlayın' (Step 1) and 'İnceleyin ve kullanıma sunun' (Step 2). A section titled 'Google Play'den uygulama imzalama özelliği' indicates that it is active. Below this, there's a section for 'Eklenecek Android Uygulama Paketleri ve APK'lar' with a note that they will be published on Google Play after being uploaded. A large button labeled 'DOSYALAR GÖZ AT' (View files) is at the bottom.

Hesap özeti

Uygulama sürümleri

Android Hazır Uygulamalar

Yapı kitaplığı

Cihaz katalogu

Uygulama imzalama

Mağaza girişи

İçerik derecelendirme

Fiyatlandırma ve dağıtım

Uygulama içi ürünler

← Yeni üretim sürümü

1 2

Sürümü hazırlayın İnceleyin ve kullanıma sunun

Google Play'den uygulama imzalama özelliği

Etkin.

Eklenecek Android Uygulama Paketleri ve APK'lar

Bu uygulama paketleri ve APK'lar bu sürümün kullanıma sunulmasından sonra Google Play Store'da sunulacak.

KİTAPLIKTAN EKLE

Uygulama paketlerinizi ve APK'larınızı buraya bırakın veya bir dosya seçin.

DOSYALAR GÖZ AT

Teşekkürler...



kasım-adalan



kasimadalan@gmail.com



kasimadalan