

Android Mobil Uygulama Geliştirme Eğitimi | Kotlin

Android Uygulama Mimarisi - MVVM

Kasım ADALAN

Elektronik ve Haberleşme Mühendisi

Android - IOS Developer and Trainer

MVVM

MVVM (Model - View - ViewModel) Mimarisi

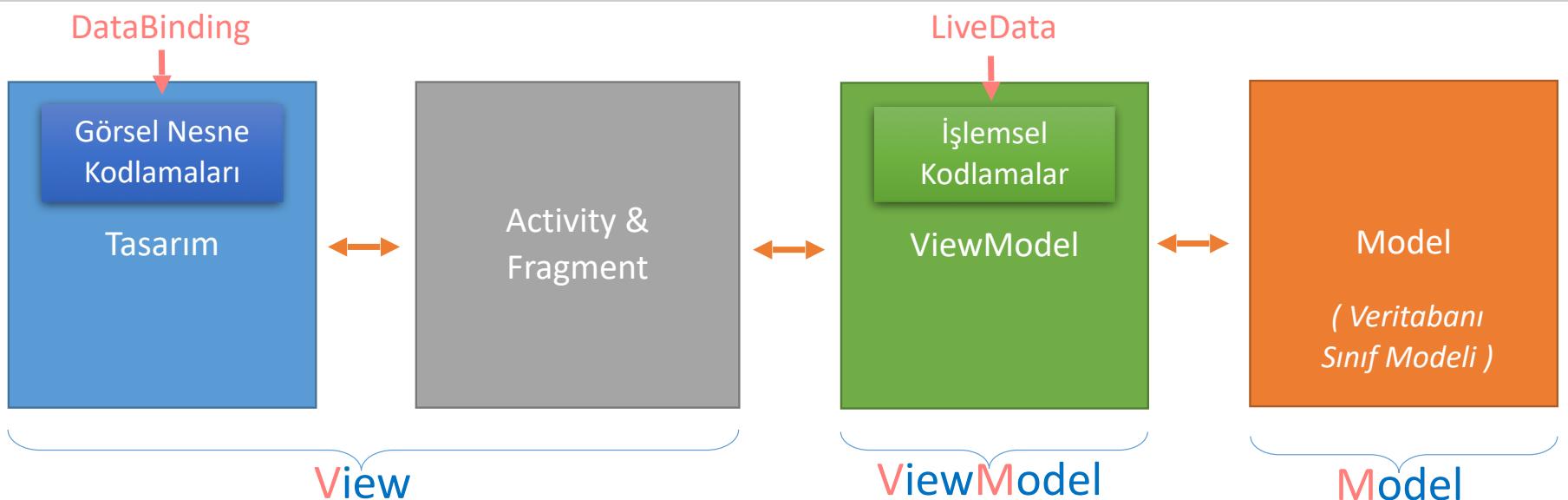
- Bu mimari sayesinde daha profesyonel kodlamalar yapabiliriz.
- Özellikle büyük projelerde oluşan kodlama karmaşasını azaltır.
- Mimarının amacı proje içerisindeki işlemsel kodları ve tasarımsal kodları ayırip daha modüller ve düzenli hale getirmektir.
- Bu mimari kullanımını öğrenmek için mimarının temel yapılarını öğrenmemiz gereklidir.
- Başlıca yapılar ; ViewModel,LiveData ve DataBinding'dir.
- ViewModel yapısı kullanımı aslında MVVM mimarisinin temel alt yapısını oluşturur.
- Temel alt yapıya DataBinding yapısı eklenirse daha profesyonel ve düzenli bir MVVM mimarisi olmuş olur.
- DataBinding,Activity içinde görsel nesne tanımlamalarını azaltır.
- ViewModel,Activity içinde işlemsel kodlamaları azaltır.
- Her ikiside aynı modeli kullanırlar ve kompact bir yapı olmuş olur.
- LiveData,ViewModel içerisinde kullanılır ve ViewModel kullanımının yetegini artırır.

MVVM (Model - View - ViewModel) Mimarisi Süreci

Klasik
Yöntem



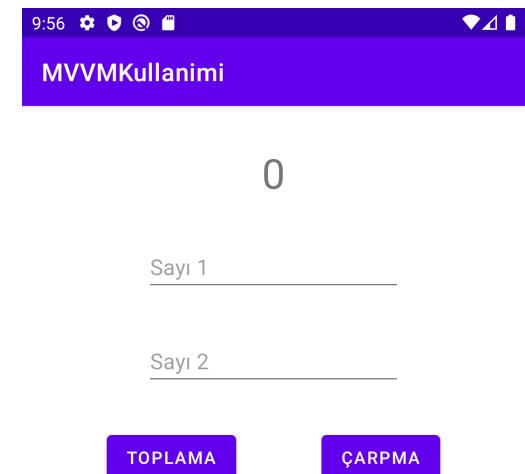
MVVM



Klasik Yöntem

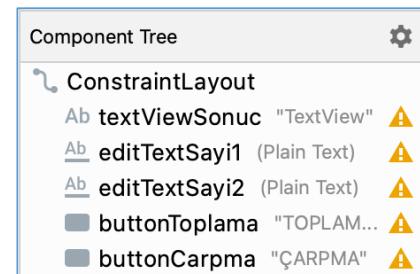
```
class MainActivity : AppCompatActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)  
  
        textViewSonuc.text = "0"  
  
        buttonToplama.setOnClickListener { it: View!  
            val alinanSayi1 = editTextSayi1.text.toString()  
            val alinanSayi2 = editTextSayi2.text.toString()  
  
            val sayi1 = alinanSayi1.toInt()  
            val sayi2 = alinanSayi2.toInt()  
  
            val toplam = sayi1 + sayi2  
  
            textViewSonuc.text = toplam.toString()  
        }  
    }  
}
```

```
buttonCarpma.setOnClickListener { it: View!  
    val alinanSayi1 = editTextSayi1.text.toString()  
    val alinanSayi2 = editTextSayi2.text.toString()  
  
    val sayi1 = alinanSayi1.toInt()  
    val sayi2 = alinanSayi2.toInt()  
  
    val carpma = sayi1 * sayi2  
  
    textViewSonuc.text = carpma.toString()  
}
```



Synthetic binding yapılarak görsel nesnelere erişildi.

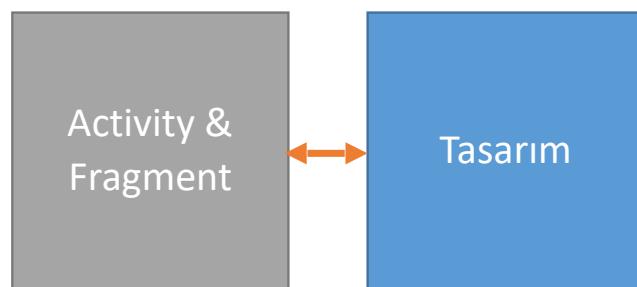
```
plugins {  
    id 'com.android.application'  
    id 'kotlin-android'  
    id 'kotlin-android-extensions'  
}
```



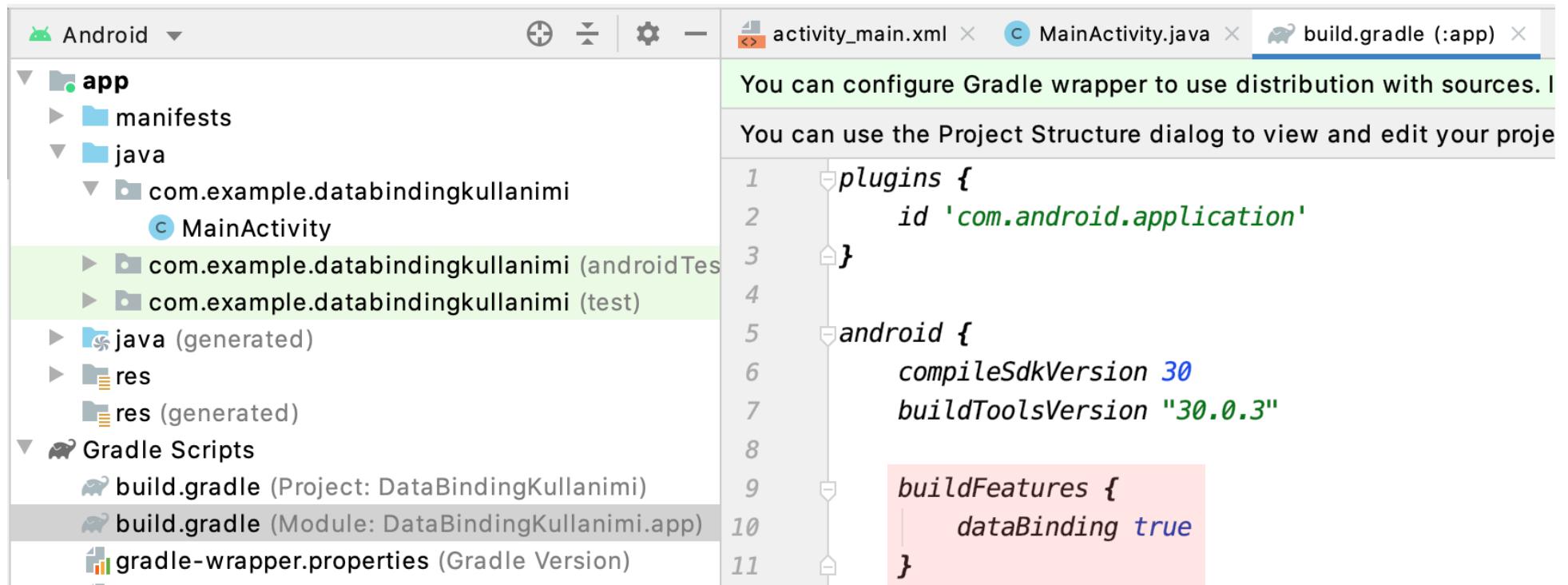
Data Binding

DataBinding

- Temel olarak tasarım alanına veri göndermektir.
- Bu veriler değişken veya nesne olabilir.
- Sınıftan tasarım alanına değişken veya nesne göndererek arayüzde bunları kullanabiliriz.
- Daha az kodlama ile tasarımını ve sınıf modelini ile yönetebiliriz.
- Tasarım üzerindeki görsel nesneleri yönetmek için kolaylık sağlar.
- Kullanıcı arayüzü manuel müdahale olmadan otomatik olarak güncellenmesini sağlayan bir kütüphanedir.
- Klasik yöntemlere göre daha hızlı çalışır ve kod kalabılığından kurtarır.



Kütüphane Kurulumu



The screenshot shows the Android Studio interface with the following details:

- Project Structure:** The left sidebar shows the project structure under the `app` module. It includes `manifests`, `java` (containing `MainActivity`), `res`, and `java (generated)`.
- Gradle Scripts:** The `Gradle Scripts` section lists `build.gradle` (Project: DataBindingKullanimi), `build.gradle` (Module: DataBindingKullanimi.app), and `gradle-wrapper.properties` (Gradle Version).
- Code Editor:** The right panel displays the `build.gradle` file for the module. The `buildFeatures` block is highlighted with a pink background.

```
plugins {
    id 'com.android.application'

}

android {
    compileSdkVersion 30
    buildToolsVersion "30.0.3"

    buildFeatures {
        dataBinding true
    }
}
```

```
buildFeatures {
    dataBinding true
}
```

Tasarımsal Kurulum

- Tasarım alt yapımızın dışına layout tag’i eklenmelidir ve tasarım alt yapısının sahip olduğu xmlns özelliklerini bu tag’e aktarılmalıdır.

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools">

<androidx.constraintlayout.widget.ConstraintLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textViewSonuc"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="32dp"
        android:text="TextView"
        android:textSize="34sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/buttonCarpma"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="32dp"
        android:text="ÇARPMA"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toEndOf="@+id/buttonToplama"
        app:layout_constraintTop_toBottomOf="@+id/editTextSayi2" />

</androidx.constraintlayout.widget.ConstraintLayout>
</layout>
```

DataBinding ile Tasarıma Erişim : Activity

```
class MainActivity : AppCompatActivity() {  
    private lateinit var tasarim:ActivityMainBinding  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        tasarim = DataBindingUtil.setContentView( activity: this,R.layout.activity_main)
```

Önemli : Tasarım Sınıfı ismi tasarım ismine göre otomatik oluşturulur.

activity_main olduğu için ActivityMainBinding olur.

Tasarımı kodlama olarak sınıfı aktardık ve bu nesne sayesinde tasarımındaki görsel nesnelere erişebiliriz.

DataBinding ile Tasarıma Erişim : Fragment

```
class KisiKayitFragment : Fragment() {  
  
    private lateinit var tasarim:FragmentKisiKayitBinding  
  
    override fun onCreateView(  
        inflater: LayoutInflater, container: ViewGroup?,  
        savedInstanceState: Bundle?  
    ): View? {  
  
        tasarim = DataBindingUtil.inflate(inflater,  
            R.layout.fragment_kisi_kayit, container, attachToParent: false)  
  
        tasarim.kisiKayitFragment = this  
        tasarim.kisiKayitToolbarBaslik = "Kişi Kayıt"  
  
        return tasarim.root  
    }  
}
```

Görsel Nesnelere Erişim

Tasarım nesnesi ile
tasarımda yer alan
görsel nesnelere
idleri ile erişebiliriz.

```
class MainActivity : AppCompatActivity() {  
  
    private lateinit var tasarim:ActivityMainBinding  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        tasarim = DataBindingUtil.setContentView( activity: this,R.layout.activity_main)  
  
        → tasarim.textViewSonuc.text = "0"  
  
        tasarim.buttonToplama.setOnClickListener { it: View!  
            val alinanSayi1 = tasarim.editTextSayi1.text.toString()  
            val alinanSayi2 = tasarim.editTextSayi2.text.toString()  
  
            val sayi1 = alinanSayi1.toInt()  
            val sayi2 = alinanSayi2.toInt()  
  
            val toplam = sayi1 + sayi2  
  
            tasarim.textViewSonuc.text = toplam.toString()  
    }  
}
```

Event Handle

- Button'a tıklanılma gibi işlemleri daha pratik hale getirebiliriz.
- Tasarım Alanından Sınıf Metodlarını Tetikleme

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools">

    <data>
        <variable name="mainActivityNesnesi" type="com.example.mvvmkullanimi.MainActivity" />
    </data>
```

Tasarım alanında kullanmak için nesne isimlendirmesi

```
<androidx.constraintlayout.widget.ConstraintLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
```

```
<TextView
    android:id="@+id/textViewSonuc"
    android:layout_width="wrap_content"
```

Hangi sınıfın metodu çalıştırırmak istiyorsak o sınıfın paketi ve adı

Nesne yardımıyla sınıfın metoda erişiyoruz ve tıklanılma özelliğine ekliyoruz.

Not : Tasarım alanında variable oluşturduysak bu değere tasarımın kullanıldığı yerden activity veya sınıfın veri aktarmamız şart. Aksi halde veriyi kullanamayız.

```
<Button
    android:id="@+id/buttonToplama"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="32dp"
    android:text="TOPLAMA"
    android:onClick="@{() -> mainActivityNesnesi.buttonToplamaTikla()}"
    app:layout_constraintEnd_toStartOf="@+id/buttonCarpma"
    app:layout_constraintHorizontal_bias="0.5"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/editTextSayi2" />
```

```
<Button
    android:id="@+id/buttonCarpma"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="32dp"
    android:text="ÇARPMA"
    android:onClick="@{() -> mainActivityNesnesi.buttonCarpmaTikla()}"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.5"
    app:layout_constraintStart_toEndOf="@+id/buttonToplama"
    app:layout_constraintTop_toBottomOf="@+id/editTextSayi2" />
</androidx.constraintlayout.widget.ConstraintLayout>
</layout>
```

Event Handle

- Butona tıklanması gibi işlemleri daha pratik hale getirebiliriz. Tasarım Alanından Sınıf Metodlarını Tetikleme

```
class MainActivity : AppCompatActivity() {  
  
    private lateinit var tasarim:ActivityMainBinding  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        tasarim = DataBindingUtil.setContentView( activity: this,R.layout.activity_main)  
        tasarim.mainActivityNesnesi = this  
  
        tasarim.textViewSonuc.text = "0"  
    }  
  
}
```

```
fun buttonToplamaTikla(){  
    val alinanSayi1 = tasarim.editTextSayi1.text.toString()  
    val alinanSayi2 = tasarim.editTextSayi2.text.toString()  
  
    val sayi1 = alinanSayi1.toInt()  
    val sayi2 = alinanSayi2.toInt()  
  
    val toplam = sayi1 + sayi2  
  
    tasarim.textViewSonuc.text = toplam.toString()  
}
```

Bu tanımlama ile tasarımında oluşturduğumuz activity nesnesine tasarımından activity'e erişim yetkisi veriyoruz. Bu tanımlama olmaz ise buttonlar metodları çalıştırılamaz.

metodun ismi tasarımında oluşturulan nesne ismi ile aynı olmalıdır.

```
fun buttonCarpmaTikla(){  
    val alinanSayi1 = tasarim.editTextSayi1.text.toString()  
    val alinanSayi2 = tasarim.editTextSayi2.text.toString()  
  
    val sayi1 = alinanSayi1.toInt()  
    val sayi2 = alinanSayi2.toInt()  
  
    val carpma = sayi1 * sayi2  
  
    tasarim.textViewSonuc.text = carpma.toString()  
}
```

Event Handle : Parametre ile Veri Gönderme

- Tasarım alanından activity'e veri göndermek isteniyorsa kullanılabilir.

```
<Button  
    android:id="@+id/buttonToplama"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_marginTop="32dp"  
    android:text="TOPLAMA"  
    android:onClick="@{() -> mainActivityNesnesi.buttonToplamaTikla(editTextSayi1.getText().toString(),editTextSayi2.getText().toString())}"  
    app:layout_constraintEnd_toStartOf="@+id/buttonCarpma"  
    app:layout_constraintHorizontal_bias="0.5"  
    app:layout_constraintStart_toStartOf="parent"  
    app:layout_constraintTop_toBottomOf="@+id/editTextSayi2" />  
  
<Button  
    android:id="@+id/buttonCarpma"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_marginTop="32dp"  
    android:text="ÇARPMA"  
    android:onClick="@{() -> mainActivityNesnesi.buttonCarpmaTikla(editTextSayi1.getText().toString(),editTextSayi2.getText().toString())}"  
    app:layout_constraintEnd_toEndOf="parent"  
    app:layout_constraintHorizontal_bias="0.5"  
    app:layout_constraintStart_toEndOf="@+id/buttonToplama"  
    app:layout_constraintTop_toBottomOf="@+id/editTextSayi2" />  
/>/androidx.constraintlayout.widget.ConstraintLayout>  
</layout>
```

TASARIM
AŞAMASI

1

Tasarım alanından
gönderilen parametre

Tasarım alanından
gönderilen parametre

*Not : getText() metodu kullanılmalı kotlin metodunda
hata oluşuyor.*

Event Handle : Parametre ile Veri Gönderme

- Tasarım alanından activity'e veri göndermek isteniyorsa kullanılabilir.

```
fun buttonToplamaTikla(alinanSayi1:String, alinanSayi2:String){  
    val sayi1 = alinanSayi1.toInt()  
    val sayi2 = alinanSayi2.toInt()  
  
    val toplam = sayi1 + sayi2  
  
    tasarim.textViewSonuc.text = toplam.toString()  
}
```

KODLAMA
AŞAMASI

2

Tasarım alanından gönderilen parametreyi alma

```
fun buttonCarpmaTikla(alinanSayi1:String, alinanSayi2:String){  
    val sayi1 = alinanSayi1.toInt()  
    val sayi2 = alinanSayi2.toInt()  
  
    val carpma = sayi1 * sayi2  
  
    tasarim.textViewSonuc.text = carpma.toString()  
}
```

Tasarım Alanına Veri (Değişken) Gönderme

- Data Binding yapısının temel işlemi tasarım alanına veri göndermektir.
- Bunun en basit yolu ise değişken gönderme işlemidir.

```
class MainActivity : AppCompatActivity() {  
    private lateinit var tasarim:ActivityMainBinding  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        tasarim = DataBindingUtil.setContentView( activity: this,R.layout.activity_main)  
        tasarim.mainActivityNesnesi = this  
        tasarim.hesaplamaSonucu = "0"  
    }  
  
    fun buttonToplamaTikla(alinanSayi1:String,alinanSayi2:String){  
        val sayı1 = alinanSayi1.toInt()  
        val sayı2 = alinanSayi2.toInt()  
        val toplam = sayı1 + sayı2  
        tasarim.hesaplamaSonucu = toplam.toString()  
    }  
  
    fun buttonCarpmaTikla(alinanSayi1:String,alinanSayi2:String){  
        val sayı1 = alinanSayi1.toInt()  
        val sayı2 = alinanSayi2.toInt()  
        val carpma = sayı1 * sayı2  
        tasarim.hesaplamaSonucu = carpma.toString()  
    }  
}
```

Tasarım alanından gönderilen değişken set metodu ile aktarılır.
Bu metodun ismi tasarım alanındaki değişken adına göre otomatik olur.

```
<?xml version="1.0" encoding="utf-8"?>  
<layout xmlns:android="http://schemas.android.com/apk/res/android"  
       xmlns:app="http://schemas.android.com/apk/res-auto"  
       xmlns:tools="http://schemas.android.com/tools">  
  
    <data>  
        <variable name="mainActivityNesnesi" type="com.example.mvvmkullanimi.MainActivity" />  
        <variable name="hesaplamaSonucu" type="String" />  
    </data>  
  
    <androidx.constraintlayout.widget.ConstraintLayout  
        android:layout_width="match_parent"  
        android:layout_height="match_parent"  
        tools:context=".MainActivity">  
  
        <TextView  
            android:id="@+id/textViewSonuc"  
            android:layout_width="wrap_content"  
            android:layout_height="wrap_content"  
            android:layout_marginTop="32dp"  
            android:text="@{hesaplamaSonucu}" ←  
            android:textSize="34sp"
```

Tasarım alanında değişken oluşturma işlemi.
name : Değişken adı
type : Değişken türünü belirtir.

Değişkenin arayüzde kullanılması

Sınıf içinde değişkene set metodu ile veri aktarıldığı anda arayüzde görülmüş olur.

Tasarım Alanında Parantez İçinde Kodlama Yapma

- Tür dönüşümü , koşul gibi kodlamaları tasarım alanında yapabiliriz.
- Görsel nesnenin her özelliğinde kodlama yapılabilir.
- Aşağıdaki kodlamada hesap sonucu 10 dan büyükse renk değişimi olacak.

```
<TextView  
    android:id="@+id/textViewSonuc"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_marginTop="32dp"  
    android:text="@{hesaplamaSonucu}"  
    android:textColor="@{Integer.parseInt(hesaplamaSonucu) > 10 ? @color/purple_700 : @color/teal_700}"  
    android:textSize="34sp"  
    app:layout_constraintEnd_toEndOf="parent"  
    app:layout_constraintHorizontal_bias="0.5"  
    app:layout_constraintStart_toStartOf="parent"  
    app:layout_constraintTop_toTopOf="parent" />
```

Not : sp gibi boyut belirten ifadeleri dimens.xml dosyasında tanımlanıp buraya akatarılmalıdır.

Tasarım alanına istediğimiz sınıfı import edebiliriz.

- Örnek görsel nesnenin görünürlüğü için View nesnesi import etmek gibi

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools">

    <data>
        <variable name="mainActivityNesnesi" type="com.example.mvvmkullanimi.MainActivity" />
        <variable name="hesaplamaSonucu" type="String" />
        <import type="android.view.View"/>
    </data>      Import işlemi

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:context=".MainActivity">

        <TextView
            android:id="@+id/textViewSonuc"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="32dp"
            android:text="@{hesaplamaSonucu}"
            android:textColor="@{Integer.parseInt(hesaplamaSonucu) > 10 ? @color/purple_700 : @color/teal_700}"
            android:visibility="@{Integer.parseInt(hesaplamaSonucu) > 20 ? View.INVISIBLE : View.VISIBLE}"
            android:textSize="34sp"          Import edilen nesnenin kullanımı
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.5"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent" />
    
```

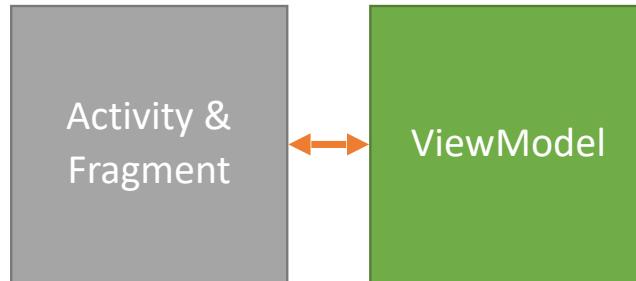
ViewModel

ViewModel

- Viewmodel'in asıl amacı arayüzü besleyecek verileri organize etmektir.
- Ayrıca sayfa rotasyonlarında (Dikey ve Yatay konuma geçişte) veriyi korumaktadır.
- Arayüzü (Activity) verilerden ayırarak daha kontrollü yapı oluşturabiliriz.
- ViewModel içinde sayfa üzerinde yapılacak işlemleri bulundurabiliriz.
- Activity'de verileri arayüze aktarma işlemi yaparız.
- View Modelde veritabanından veri alma , arayüzde matematiksel işlem vb. gibi şeyler yapılır.
- Verilerdeki değişimleri gözlemlerek için *LiveData* yapısını kullanabiliriz.
- DataBinding ile karışmaması lazım , Data binding tasarım alanına verileri kolay şekilde aktarmamızı sağlar.
- View Model ise Activity içinde yapılan arayüz ile ilgili veri işlemlerini kontrol eder.
- Genel kullanım her sayfaya özgü ViewModel oluşturmaktır.

Not : ViewModel aslında MVVM mimarisinin alt yapısıdır.Tek eksik DataBinding yapısıdır onuda ViewModelle entegre edebilirsek MVVM mimarisi olmuş olur.

Ama çoğu kaynak DataBinding yapısı olmadan da MVVM olabileceğini söylemektedir.



Neler Yapamaz

- Toast,SnackBar,Alert gösteremez.
- Sayfa geçiş için intent kullanılamaz.
- Görsel nesne ile ilgili işlemler olmaz.

Kütüphane Kurulumu

The screenshot shows the Android Studio interface. The left sidebar displays the project structure under the 'app' module. The 'build.gradle (Module: ViewModelKullanimi.app)' file is open in the main editor. The code in the editor is as follows:

```
You can configure Gradle wrapper to use distribution with sources. It will provide IDE wit... Hide the
You can use the Project Structure dialog to view and edit your project configuration
34 }
35
36 dependencies {
37
38     implementation "org.jetbrains.kotlin:kotlin-stdlib:$kotlin_version"
39     implementation 'androidx.core:core-ktx:1.3.2'
40     implementation 'androidx.appcompat:appcompat:1.2.0'
41     implementation 'com.google.android.material:material:1.3.0'
42     implementation 'androidx.constraintlayout:constraintlayout:2.0.4'
43     testImplementation 'junit:junit:4.+'
44     androidTestImplementation 'androidx.test.ext:junit:1.1.2'
45     androidTestImplementation 'androidx.test.espresso:espresso-core:3.3.0'
46
47     //ViewModel
48     implementation "androidx.lifecycle:lifecycle-viewmodel:2.3.0"
49     //by viewModels icin gerekli
50     implementation "androidx.activity:activity-ktx:1.2.0"
51 }
```

```
implementation "androidx.lifecycle:lifecycle-viewmodel:2.3.0"
implementation "androidx.activity:activity-ktx:1.2.0"
```

ViewModel Sınıfı Oluşturma

- Arayüzde kullanacağımız veriyi bu sınıfta tanımlarız ve yönetiriz.

*Yönetilecek ve arayüzde
kullanılacak veri*

*Yönettiğimiz değere
matematiksel işlemi aktardık.*

*Yönettiğimiz değere
matematiksel işlemi aktardık.*

```
class MainActivityViewModel : ViewModel(){ ViewModel özelliği aktarılır.
```

```
    var sonuc = "0"
```

```
    fun toplamaYap(alinanSayi1:String, alinanSayi2:String){  
        val sayı1 = alinanSayi1.toInt()  
        val sayı2 = alinanSayi2.toInt()  
        val toplam = sayı1 + sayı2  
        sonuc = toplam.toString()  
    }
```

```
    fun carpmaYap(alinanSayi1:String, alinanSayi2:String){  
        val sayı1 = alinanSayi1.toInt()  
        val sayı2 = alinanSayi2.toInt()  
        val carpma = sayı1 * sayı2  
        sonuc = carpma.toString()  
    }
```

*Buradaki metodların amacı
arayüzü değiştirecek veya
besleyecek verilere değer
üretmektir.*

*Bundan dolayı return demeden
yönettiğimiz veriye elde
ettiğimiz değer aktarılır.
Aktarılma işlemi olduğunda
yönettiğimiz verinin içeriği
değişmiş olur.*

ViewModel Kullanımı

- Activity içinde oluşturduğumuz ViewModel sınıfını kullanabiliriz.

```
class MainActivity : AppCompatActivity() {
    private lateinit var tasarim:ActivityMainBinding

    Oluşturduğumuz ViewModel
    sınıfını Activity'e bağlıyoruz. →    private val viewModel:MainActivityViewModel by viewModels()

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        tasarim = DataBindingUtil.setContentView( activity: this,R.layout.activity_main)
        tasarim.mainActivityNesnesi = this
        tasarim.hesaplamaSonucu = viewModel.sonuc
    }

    View model içindeki yönettiğimiz
    veriye hesaplanan değeri
    aktarmak için View model
    içindeki metod çalıştırılır.

    Bu metod yönettiğimiz değerin
    içeriğini değiştirir. →    fun buttonToplamaTikla(alinanSayi1:String,alinanSayi2:String){
        viewModel.toplamaYap(alinanSayi1,alinanSayi2)
        tasarim.hesaplamaSonucu = viewModel.sonuc
    }

    Yukardaki metod sayesinde
    yönettiğimiz değerin içeriğinin
    değişmiş en son halini alırız ve
    arayüzde gösteririz.

    fun buttonCarpmaTikla(alinanSayi1:String,alinanSayi2:String){
        viewModel.carpmaYap(alinanSayi1,alinanSayi2)
        tasarim.hesaplamaSonucu = viewModel.sonuc
    }
}
```

Oluşturduğumuz ViewModel sınıfını Activity'e bağlıyoruz.

View model içinde yönettiğimiz veriye hesaplanan değeri aktarmak için View model içindeki metod çalıştırılır.

Bu metod yönettiğimiz değerin içeriğini değiştirir.

ViewModel Fragment İçinde Kullanımı

```
class KisiKayitFragment : Fragment() {
    private lateinit var tasarim:FragmentKisiKayitBinding
    private lateinit var viewModel: KisiKayitFragmentViewModel

    override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?, savedInstanceState: Bundle?): View {

        tasarim = DataBindingUtil.inflate(inflater, R.layout.fragment_kisi_kayit, container, attachToParent: false)
        tasarim.kisiKayitFragment = this
        tasarim.kisiKayitToolbarBaslik = "Kişi Kayıt"

        return tasarim.root
    }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        //Fragment içinde viewModel tanımlaması
        val tempViewModel: KisiKayitFragmentViewModel by viewModels()
        this.viewModel = tempViewModel
    }

    fun buttonKaydetTikla(kisi_ad:String,kisi_tel:String){
        viewModel.kayit(kisi_ad,kisi_tel)
    }
}
```

ViewModel İçinde Context Kullanımı

ViewModel içinde veritabanı gibi çalışmalarda context gereklı olabilir.

1

```
class MainActivityViewModelFactory(private val application: Application)
    : ViewModelProvider.NewInstanceFactory() {

    override fun <T : ViewModel?> create(modelClass: Class<T>): T {
        return MainActivityViewModel(application) as T
    }

}
```

2

```
class MainActivityViewModel(application: Application) : AndroidViewModel(application) {

    private val kdaor = KisilerDaoRepository(application)
    var kisilerListesi = MutableLiveData<List<Kisiler>>()
```

3

```
class MainActivity : AppCompatActivity(), SearchView.OnQueryTextListener {
    private lateinit var adapter: KisilerAdapter
    private val viewModel: MainActivityViewModel by viewModels() {MainActivityViewModelFactory(application)}
```

LiveData

LiveData

- View Model ile kullanılır.
- LiveData View Model'in kullanımı kolaylaştırır.
- Temelde yaptığı işlem ViewModel içinde yönetilen verinin tetiklenmesini sağlamak ve değişimi dinlemektir.
- Bu tetikleme ve değişimi dinleme işlemi kodlama yapımızı sadeleştirir.
- Örnek : TextView içeriği çok fazla yerde değiştiriliyorsa , normalde her değişim kodlamasında textView kodlamasını yazmamız gereklidir.
- LiveData sayesinde dinleme yapısı kurularak tek bir textView tanımlaması yapılarak , değişen değeri dinleriz ve değeri textView içeriğine aktarırız.
- Bu işlem gereksiz textView kodlamalarından bizi kurtarır.

Kütüphane Kurulumu

The screenshot shows the Android Studio interface with the project structure on the left and the build.gradle file open in the editor on the right.

Project Structure:

- app
- manifests
- java
 - com.example.viewmodelkullanimi
 - MainActivity
 - MainActivityViewModel
 - com.example.viewmodelkullanimi (androidTest)
 - com.example.viewmodelkullanimi (test)
 - java (generated)
 - res
 - res (generated)
- Gradle Scripts
 - build.gradle (Project: ViewModelKullanimi)
 - build.gradle (Module: ViewModelKullanimi.app)
 - gradle-wrapper.properties (Gradle Version)
 - proguard-rules.pro (ProGuard Rules for ViewModelKullanimi)
 - gradle.properties (Project Properties)
 - settings.gradle (Project Settings)
 - local.properties (SDK Location)

build.gradle (Module: ViewModelKullanimi.app) Content:

```
plugins {
    id 'com.android.application'
    id 'kotlin-android'
    id 'kotlin-android-extensions'
}

apply plugin: 'kotlin-kapt'

dependencies {
    implementation "org.jetbrains.kotlin:kotlin-stdlib:$kotlin_version"
    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.constraintlayout:constraintlayout:2.0.4'
    testImplementation 'junit:junit:4.+'
    androidTestImplementation 'androidx.test.ext:junit:1.1.2'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.3.0'

    //ViewModel
    implementation "androidx.lifecycle:lifecycle-viewmodel:2.3.0"
    //by viewModels icin gerekli
    implementation "androidx.activity:activity-ktx:1.2.0"
    //LiveData
    implementation "androidx.lifecycle:lifecycle-livedata:2.3.0"
    kapt "androidx.lifecycle:lifecycle-compiler:2.2.0"
}
```

Selected Lines:

- apply plugin: 'kotlin-kapt'
- implementation "androidx.lifecycle:lifecycle-livedata:2.3.0"
- kapt "androidx.lifecycle:lifecycle-compiler:2.2.0"

LiveData - ViewModel içinde Kullanımı

- ViewModel yapısını güçlendirmek için kullanılır.

Yönetilecek veri live data
türüne dönüştürülür.

```
class MainActivityViewModel : ViewModel(){  
    var sonuc = MutableLiveData<String>()  
  
    init {  
        sonuc = MutableLiveData<String>( value: "0")  
    }  
}
```

Verimize varsayılan değer atanır.
Atama işlemi constructor ile yapılır.

Bu değer atandığı anda dinleme
işlemi tetikler.

```
fun toplamaYap(alinanSayi1:String,alinanSayi2:String){  
    val sayı1 = alinanSayi1.toInt()  
    val sayı2 = alinanSayi2.toInt()  
    val toplam = sayı1 + sayı2  
    sonuc.value = toplam.toString()  
}
```

value metodunu kullanarak hesaplama sonucunda
oluşan değeri verimizde atadık ve dinleme
işlemi tetikledik.

```
fun carpmaYap(alinanSayi1:String,alinanSayi2:String){  
    val sayı1 = alinanSayi1.toInt()  
    val sayı2 = alinanSayi2.toInt()  
    val carpma = sayı1 * sayı2  
    sonuc.value = carpma.toString()  
}
```

LiveData - Activity İçinde Kullanımı

Yönettiğimiz verinin
değişimini dinleme.
view model içindeki
value metodunu bu
dinlemeyi tetikler.

```
class MainActivity : AppCompatActivity() {  
    private lateinit var tasarim:ActivityMainBinding  
  
    private val viewModel:MainActivityViewModel by viewModels()  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        tasarim = DataBindingUtil.setContentView( activity: this,R.layout.activity_main)  
        tasarim.mainActivityNesnesi = this  
  
        viewModel.sonuc.observe( owner: this,{ s ->  
            tasarim.hesaplamaSonuclu = s  
        })  
    }  
  
    fun buttonToplamaTikla(alinanSayi1:String,alinanSayi2:String){  
        viewModel.toplamaYap(alinanSayi1,alinanSayi2)  
    }  
  
    fun buttonCarpmaTikla(alinanSayi1:String,alinanSayi2:String){  
        viewModel.carpmaYap(alinanSayi1,alinanSayi2)  
    }  
}  
getLifecycle().addObserver(viewModel);
```

value metodunu veride değişim
olduğu anda en son değer buraya
gelir ve arayüzde kullanılabilir.

Bu metod ile ViewModel
içindeki verimizin içeriğini
hesaplama sonucusu ile
değiştirebiliriz ve içinde
value metodunu olduğu için
dinleme işlemini tetikler.

Repository

Repository Sınıfı

- Ortak kullanım için oluşturduğumuz metodların yer aldığı sınıfır.
- Veritabanı işlemlerinde bazı metodları bir çok sayfa kullanabilir.
- Ortak erişebilecek bir sınıf oluşturup kodlama tekrarlarını azaltırız.
- Aslında dao (Database access object) sınıfıdır.
- Alt yapısı [View Model](#) örnek alınarak oluşturulur.

Repository Oluşturma

Yönetilen veriye erişmek
için metod.

Yönetilen verinin içeriğini
değiştirme

```
class MatematikRepository {  
    var matematiselSonuc = MutableLiveData<String>()  
  
    init {  
        matematiselSonuc = MutableLiveData<String>( value: "0")  
    }  
  
    fun matematiselSonunuGetir() : MutableLiveData<String>{  
        return matematiselSonuc  
    }  
  
    fun topla(alinanSayi1:String,alinanSayi2:String){  
        val sayi1 = alinanSayi1.toInt()  
        val sayi2 = alinanSayi2.toInt()  
        val toplam = sayi1 + sayi2  
        matematiselSonuc.value = toplam.toString()  
    }  
  
    fun carp(alinanSayi1:String,alinanSayi2:String){  
        val sayi1 = alinanSayi1.toInt()  
        val sayi2 = alinanSayi2.toInt()  
        val carpma = sayi1 * sayi2  
        matematiselSonuc.value = carpma.toString()  
    }  
}
```

Yönetilecek veri

Varsayılan değer

Bir sorun olursa en
azından gösterilecek
varsayılan veri
tanımlanmış olur.

ViewModel İçinde Kullanımı

```
class MainActivityViewModel : ViewModel(){  
    var sonuc = MutableLiveData<String>()  
    var mrepo = MatematikRepository() ← Repository sınıfına erişmek için nesne
```

ViewModel çalıştığı anda
repo dan veriyi alır ve
arayüzde gösterir.

Bu durumda repo içinde
belirlenmiş varsayılan
değer gösterilir.

```
init {  
    sonuc = mrepo.matematikselSonucuGetir()  
}  
  
fun toplamaYap(alinanSayi1:String,alinanSayi2:String){  
    mrepo.topla(alinanSayi1,alinanSayi2) ← Repository sınıfındaki  
                                                metodу çalıştırır ve repo  
                                                içindeki değer değişir,  
                                                değişen değeri  
                                                matematikselSonucuGetir  
                                                metodу view modele  
                                                aktarır.  
}  
  
fun carpmaYap(alinanSayi1:String,alinanSayi2:String){  
    mrepo.carp(alinanSayi1,alinanSayi2)  
}  
}
```

Yardımcı Kaynaklar

Recyclerview

—

Data Binding Kullanımı

Recyclerview - Data Binding Kullanımı

```
data class Kisiler(var kisi_id:Int,  
                   var kisi_ad:String,  
                   var kisi_tel:String) {  
}
```

```
<?xml version="1.0" encoding="utf-8"?>  
<layout xmlns:android="http://schemas.android.com/apk/res/android"  
        xmlns:app="http://schemas.android.com/apk/res-auto"  
        xmlns:tools="http://schemas.android.com/tools">  
  
    <data>  
        <variable name="kisiNesnesi" type="com.example.kisileruygulamasidatabinding.Kisiler"/>  
    </data>  
  
    <androidx.constraintlayout.widget.ConstraintLayout  
        android:layout_width="match_parent"  
        android:layout_height="wrap_content"  
        android:orientation="vertical">  
  
        <androidx.cardview.widget.CardView  
            android:layout_width="match_parent"  
            android:layout_height="50dp"  
            android:layout_margin="5dp"  
            app:cardCornerRadius="5dp"  
            app:layout_constraintBottom_toBottomOf="parent"  
            app:layout_constraintEnd_toEndOf="parent"  
            app:layout_constraintStart_toStartOf="parent"  
            app:layout_constraintTop_toTopOf="parent">  
            <androidx.constraintlayout.widget.ConstraintLayout  
                android:layout_width="match_parent"  
                android:layout_height="match_parent">  
  
                <TextView  
                    android:id="@+id/textViewKisiBilgi"  
                    android:layout_width="wrap_content"  
                    android:layout_height="wrap_content"  
                    android:layout_centerHorizontal="true"  
                    android:layout_centerVertical="true"  
                    android:text='@(kisiNesnesi.kisi_ad" - "+kisiNesnesi.kisi_tel)'  
                    app:layout_constraintBottom_toBottomOf="parent"  
                    app:layout_constraintEnd_toEndOf="parent"  
                    app:layout_constraintHorizontal_bias="0.5"  
                    app:layout_constraintStart_toStartOf="parent"  
                    app:layout_constraintTop_toTopOf="parent" />  
  
            </androidx.constraintlayout.widget.ConstraintLayout>  
        </androidx.cardview.widget.CardView>  
    </androidx.constraintlayout.widget.ConstraintLayout>  
</layout>
```

Card Tasarımı

RecyclerView Adapter

```
class KisilerAdapter(private val mContext: Context, private val kisilerListe: List<Kisiler>)
    : RecyclerView.Adapter<KisilerAdapter.CardTasarimTutucu>(){

    inner class CardTasarimTutucu(cardTasarimBinding: CardTasarimBinding)
        : RecyclerView.ViewHolder(cardTasarimBinding.root){
        var cardTasarimBinding:CardTasarimBinding
        init {
            this.cardTasarimBinding = cardTasarimBinding
        }
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): CardTasarimTutucu {
        val layoutInflater = LayoutInflater.from(mContext)
        val cardTasarimBinding = CardTasarimBinding
            .inflate(layoutInflater, parent, attachToRoot: false)
        return CardTasarimTutucu(cardTasarimBinding)
    }

    override fun onBindViewHolder(holder: CardTasarimTutucu, position: Int) {
        val kisi = kisilerListe.get(position)
        holder.cardTasarimBinding.kisiNesnesi = kisi
    }

    override fun getItemCount(): Int {
        return kisilerListe.size
    }
}
```

card tasarim dosyası ismine
göre bu sınıf ismi otomatik
olur.

Activity

```
class MainActivity : AppCompatActivity() {  
    private lateinit var kisilerListe:ArrayList<Kisiler>  
    private lateinit var adapter: KisilerAdapter  
  
    private lateinit var tasarimBinding:ActivityMainBinding  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        tasarimBinding = DataBindingUtil  
            .setContentView( activity: this, R.layout.activity_main)  
  
        kisilerListe = ArrayList()  
  
        val k1 = Kisiler( kisi_id: 1, kisi_ad: "Ahmet", kisi_tel: "888888")  
        val k2 = Kisiler( kisi_id: 2, kisi_ad: "Zeynep", kisi_tel: "666666")  
  
        kisilerListe.add(k1)  
        kisilerListe.add(k2)  
  
        adapter = KisilerAdapter( mContext: this, kisilerListe)  
        tasarimBinding.kisilerAdapter = adapter  
    }  
}
```

```
<?xml version="1.0" encoding="utf-8"?>  
<layout xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:app="http://schemas.android.com/apk/res-auto"  
    xmlns:tools="http://schemas.android.com/tools">  
  
    <data>  
        <variable name="kisilerAdapter"  
            type="com.example.kisileruygulamasidatabinding.KisilerAdapter" />  
    </data>  
  
    <androidx.constraintlayout.widget.ConstraintLayout  
        android:layout_width="match_parent"  
        android:layout_height="match_parent"  
        tools:context=".MainActivity">  
  
        <androidx.recyclerview.widget.RecyclerView  
            android:id="@+id/rv"  
            android:layout_width="0dp"  
            android:layout_height="0dp"  
            android:adapter="@{kisilerAdapter}"  
            app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"  
            app:layout_constraintBottom_toBottomOf="parent"  
            app:layout_constraintLeft_toLeftOf="parent"  
            app:layout_constraintRight_toRightOf="parent"  
            app:layout_constraintTop_toTopOf="parent" />  
  
    </androidx.constraintlayout.widget.ConstraintLayout>  
    </layout>
```

Activity

11:32 4G

KisilerUygulamasiDataBinding

Ahmet - 2712637
Zeynep - 19812123

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools">

    <data>
        <variable name="kisilerAdapter"
            type="com.example.kisileruygulamasidatabinding.KisilerAdapter" />
    </data>

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:context=".MainActivity">

        <androidx.recyclerview.widget.RecyclerView
            android:id="@+id/rv"
            android:layout_width="0dp"
            android:layout_height="0dp"
            android:adapter="@{kisilerAdapter}"
            android:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintLeft_toLeftOf="parent"
            app:layout_constraintRight_toRightOf="parent"
            app:layout_constraintTop_toTopOf="parent" />

    </androidx.constraintlayout.widget.ConstraintLayout>
</layout>
```

11:34 4G

KisilerUygulamasiDataBinding

Ahmet - 2712637
Zeynep - 19812123

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools">

    <data>
        <variable name="kisilerAdapter"
            type="com.example.kisileruygulamasidatabinding.KisilerAdapter" />
    </data>

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:context=".MainActivity">

        <androidx.recyclerview.widget.RecyclerView
            android:id="@+id/rv"
            android:layout_width="0dp"
            android:layout_height="0dp"
            android:adapter="@{kisilerAdapter}"
            android:layoutManager="androidx.recyclerview.widget.StaggeredGridLayoutManager"
            android:orientation="vertical"
            app:spanCount="2"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintLeft_toLeftOf="parent"
            app:layout_constraintRight_toRightOf="parent"
            app:layout_constraintTop_toTopOf="parent" />

    </androidx.constraintlayout.widget.ConstraintLayout>
</layout>
```

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools">

    <data>
        <variable name="kisilerAdapter"
            type="com.example.kisileruygulamasidatabinding.KisilerAdapter" />
    </data>

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:context=".MainActivity">

        <androidx.recyclerview.widget.RecyclerView
            android:id="@+id/rv"
            android:layout_width="0dp"
            android:layout_height="0dp"
            android:adapter="@{kisilerAdapter}"
            android:layoutManager="androidx.recyclerview.widget.StaggeredGridLayoutManager"
            android:orientation="vertical"
            app:spanCount="2"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintLeft_toLeftOf="parent"
            app:layout_constraintRight_toRightOf="parent"
            app:layout_constraintTop_toTopOf="parent" />

    </androidx.constraintlayout.widget.ConstraintLayout>
</layout>
```

Recyclerview

View Model ve Live Data
Kullanımı

ViewModel ve LiveData Yapısı

```
class MainActivityViewModel : ViewModel {
```

Adaptera aktaracağımız
ArrayList veri kümesi

```
    val kisilerListesi: MutableLiveData<List<Kisiler>> by lazy {  
        MutableLiveData<List<Kisiler>>()  
    }
```

ViewModel ilk
oluşturulduğunda işlem
yapmak için init
constructorımız.

```
    init {  
        kisileriYukle()  
    }
```

Bu örnekte ViewModel ilk
oluşturulduğunda veri
yükleme işlemi yapılır
çünkü bu metod value
metodunu çalıştırır ve
dinlemeyi tetikler.

```
    fun kisileriYukle() {  
        val kisilerListe = ArrayList<Kisiler>()  
  
        val k1 = Kisiler( kisi_id: 1, kisi_ad: "Ahmet", kisi_tel: "888888")  
        val k2 = Kisiler( kisi_id: 2, kisi_ad: "Zeynep", kisi_tel: "666666")  
  
        kisilerListe.add(k1)  
        kisilerListe.add(k2)  
  
        kisilerListesi.value = kisilerListe  
    }  
}
```

Verileri bu metod
sayesinde oluştururuz ve
dinlemeyi tetikleyebiliriz.

Activity Yapısı

```
class MainActivity : AppCompatActivity() {
    private lateinit var adapter: KisilerAdapter
    private val viewModel: MainActivityViewModel by viewModels()

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        rv.layoutManager = LinearLayoutManager( context: this)
    }

    kisilerListesini
    dinleme işlemi.
    init metodu içindeki
    metod sayesinde ilk
    çalıştığında
    arayüzde veriler
    görebiliriz. }
```

→ viewModel.kisilerListesi.observe(owner: this, { kisilerListesi ->
 adapter = KisilerAdapter(mContext: this,kisilerListesi)
 rv.adapter = adapter
})

Activity Yapısı - Tetikleme işlemini tekrar tekrar yapabilme

- Veri yükleme metodunu istediğimiz zaman çalıştırıp arayüze yeni verileri aktarabiliriz.

```
class MainActivity : AppCompatActivity() {  
    private lateinit var adapter: KisilerAdapter  
    private val viewModel: MainActivityViewModel by viewModels()  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)  
  
        rv.layoutManager = LinearLayoutManager(context: this)  
  
        viewModel.kisilerListesi.observe(owner: this, { kisilerListesi ->  
            adapter = KisilerAdapter( mContext: this, kisilerListesi)  
            rv.adapter = adapter  
        })  
    }  
    viewModel.kisileriYukle()  
}
```

ViewModel içindeki
kisilerYukle metodunu
çalıştırırsak verileri alıp
value ile dinleme işlemini
tetikleriz.

Repository

Repository Sınıfı

Ortak liste olmalı hem bütün kişileri alırken hemde başka işlem için kullanılabilir ve bütün metodlar bu listeyi besler.

```
class KisilerDaoRepository {  
  
    private val kisilerListesi: MutableLiveData<List<Kisiler>>  
  
    init {  
        kisilerListesi = MutableLiveData()  
    }  
  
    fun kisilerSonuc(): MutableLiveData<List<Kisiler>> {  
        return kisilerListesi  
    }  
  
    fun kisileriAl() {  
        val kisilerListe = ArrayList<Kisiler>()  
  
        val k1 = Kisiler( kisi_id: 1, kisi_ad: "Ahmet", kisi_tel: "888888")  
        val k2 = Kisiler( kisi_id: 2, kisi_ad: "Zeynep", kisi_tel: "666666")  
  
        kisilerListe.add(k1)  
        kisilerListe.add(k2)  
  
        kisilerListesi.value = kisilerListe  
    }  
}
```

Bu kodlama sayesinde ortak beslenen listeye en son değeri aktarırız.

Ortak listenin oluşturulması

Bu metod sayesinde ortak beslenen listenin hep en son değerini almış oluruz ve bu metod sayesinde ViewModel ike bağlantı kurulur.

Bu işlemin avantajı bu metodları başka viewModel sınıflarıda kullanabilir.

Eğer bunları viewModel içine yazsaydık. Daha sonra başka bir viewModel içinde benzer metodları kullanıcaksak bu metodları oraya tekrar yazmamız gereklidir.

Bu şekilde ortak kullanılacak bir repository oluşturduk.

ViewModel İçinde Kullanımı

```
class MainActivityViewModel : ViewModel() {  
    var kisilerListesi = MutableLiveData<List<Kisiler>>()  
    private val kdoar = KisilerDaoRepository()  
  
    init {  
        kisileriYukle()  
        kisilerListesi = kdoar.kisilerSonuc()  
    }  
  
    fun kisileriYukle() {  
        kdoar.kisileriAl()  
    }  
}
```

val iken var yaptık çünkü listeye başka bir liste aktardık.

Repository sınıfına erişim

Repository ortak listesinin buradaki listeye bağlanması. Artık Repository içinde listede değişim olduğu anda bu listeyi yenilicek ve bu listede activity içindeki dinleme kısmını tetiklicek.

Bu metod ortak listeyi besliyor ve buradaki liste yenilenmiş oluyor.

Activity Yapısı

```
class MainActivity : AppCompatActivity() {
    private lateinit var adapter: KisilerAdapter
    private val viewModel: MainActivityViewModel by viewModels()

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

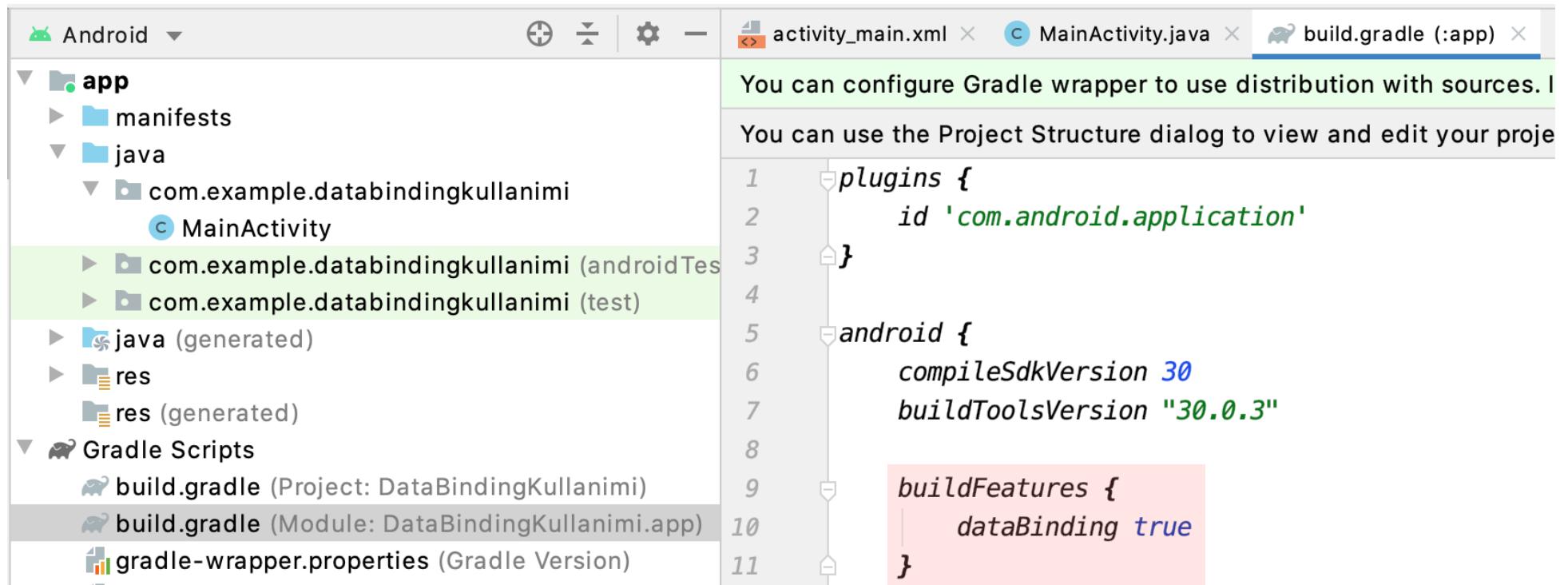
        rv.layoutManager = LinearLayoutManager( context: this)
    }

    kisilerListesini
    dinleme işlemi.
    init metodu içindeki
    metod sayesinde ilk
    çalıştığında
    arayüzde veriler
    görebiliriz. }
```

→ viewModel.kisilerListesi.observe(owner: this, { kisilerListesi ->
 adapter = KisilerAdapter(mContext: this,kisilerListesi)
 rv.adapter = adapter
})

ViewModel ile DataBinding Birlikte Kullanımı

Kütüphane Kurulumu



The screenshot shows the Android Studio interface with the following details:

- Project Structure:** The left sidebar shows the project structure under the `app` module. It includes `manifests`, `java` (containing `MainActivity`), `res`, and `Gradle Scripts`. The `build.gradle` file is currently selected.
- Build Script Content:** The right pane displays the `build.gradle` file content:

```
plugins {
    id 'com.android.application'

}

android {
    compileSdkVersion 30
    buildToolsVersion "30.0.3"

    buildFeatures {
        dataBinding true
    }
}
```
- Toolbars and Status:** The top bar shows standard Android Studio icons for file operations. A status message at the bottom indicates: "You can configure Gradle wrapper to use distribution with sources. I" and "You can use the Project Structure dialog to view and edit your projec".

```
buildFeatures {
    dataBinding true
}
```

LiveData - ViewModel içinde Kullanımı

- ViewModel yapısını güçlendirmek için kullanılır.

Yönetilecek veri live data
türüne dönüştürülür.

```
class MainActivityViewModel : ViewModel() {  
  
    → val sonuc: MutableLiveData<String> by lazy {  
        MutableLiveData<String>( value: "0")  
    }  
  
    fun hesapla(alinanVeri:String){  
        val sayı = alinanVeri.toInt()  
        sonuc.value = (sayı * sayı).toString()  
    }  
}
```

lazy , lateinit var yapısının val
için kullanılan versiyonudur.

Verimize varsayılan değer atanır.
Atama işlemi constructor ile yapılır.
Bu değer atama yapıldığı anda
dinleme işlemini tetikler.

value metodу ile hesap
sonucunda oluşan değeri
verimize atadık ve dinleme
işlemi tetikledik.

```

class MainActivity : AppCompatActivity() {

    private val viewModel: MainActivityViewModel by viewModels()

    private lateinit var tasarimBinding: ActivityMainBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        tasarimBinding = DataBindingUtil
            .setContentView( activity: this, R.layout.activity_main)

        tasarimBinding.mainActivityNesnesi = this

        viewModel.sonuc.observe( owner: this, { s ->
            tasarimBinding.viewModelNesnesi = viewModel
        })
    }

    fun buttonTikla(alinanVeri:String) {
        viewModel.hesapla(alinanVeri)
    }
}

```

Buttona tıklanınca hesaplama işlemi yapılacak ve hesaplanan değer value metoduya sonuca aktarıldığı anda dinleme kısmı tetiklenecek.

```

<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools">

    <data>
        <variable name="mainActivityNesnesi" type="com.example.viewmodelkullanimi.MainActivity"/>
        <variable name="viewModelNesnesi" type="com.example.viewmodelkullanimi.MainActivityViewModel"/>
    </data>

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:context=".MainActivity">

        <TextView
            android:id="@+id/textViewCikti"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="64dp"
            android:text="@{viewModelNesnesi.sonuc}"
            android:textSize="36sp"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.5"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent" />

        <EditText
            android:id="@+id/editTextGirdi"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="64dp"
            android:ems="10"
            android:hint="Sayı giriniz"
            android:inputType="textPersonName"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.5"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/textViewCikti" />

        <Button
            android:id="@+id/buttonHesapla"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="64dp"
            android:text="HESAPLA"
            android:onClick="@{() -> mainActivityNesnesi.buttonTikla(editTextGirdi.getText().toString())}"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0.5"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/editTextGirdi" />
    </layout>

```

Activityden gönderilen viewModelNesnesi ile getSonuc metodunu çalıştırırsak en son değeri almış ve arayüzde görüntülemiş oluruz.

DataBinding Olmadan

```
class MainActivity : AppCompatActivity() {  
  
    private val viewModel: MainActivityViewModel by viewModels()  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)  
  
        viewModel.sonuc.observe(owner: this, { s ->  
            textViewCikti.text = s  
        })  
  
        buttonHesapla.setOnClickListener { it: View!  
            val alinanVeri = editTextGirdi.text.toString()  
  
            viewModel.hesapla(alinanVeri)  
        }  
    }  
}
```

DataBinding Uygulandıktan Sonra

```
class MainActivity : AppCompatActivity() {  
  
    private val viewModel: MainActivityViewModel by viewModels()  
  
    private lateinit var tasarimBinding: ActivityMainBinding  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        tasarimBinding = DataBindingUtil  
            .setContentView(activity: this, R.layout.activity_main)  
  
        tasarimBinding.mainActivityNesnesi = this  
  
        viewModel.sonuc.observe(owner: this, { s ->  
            tasarimBinding.viewModelNesnesi = viewModel  
        })  
    }  
  
    fun buttonTikla(alinanVeri: String) {  
        viewModel.hesapla(alinanVeri)  
    }  
}
```

Recyclerview

View Model ve DataBinding Kullanımı

Recyclerview - Data Binding Kullanımı

```
data class Kisiler(var kisi_id:Int,  
                  var kisi_ad:String,  
                  var kisi_tel:String) {  
}
```

```
<?xml version="1.0" encoding="utf-8"?>  
<layout xmlns:android="http://schemas.android.com/apk/res/android"  
        xmlns:app="http://schemas.android.com/apk/res-auto"  
        xmlns:tools="http://schemas.android.com/tools">  
  
    <data>  
        <variable name="kisiNesnesi" type="com.example.kisileruygulamasidatabinding.Kisiler"/>  
    </data>  
  
    <androidx.constraintlayout.widget.ConstraintLayout  
        android:layout_width="match_parent"  
        android:layout_height="wrap_content"  
        android:orientation="vertical">  
  
        <androidx.cardview.widget.CardView  
            android:layout_width="match_parent"  
            android:layout_height="50dp"  
            android:layout_margin="5dp"  
            app:cardCornerRadius="5dp"  
            app:layout_constraintBottom_toBottomOf="parent"  
            app:layout_constraintEnd_toEndOf="parent"  
            app:layout_constraintStart_toStartOf="parent"  
            app:layout_constraintTop_toTopOf="parent">  
            <androidx.constraintlayout.widget.ConstraintLayout  
                android:layout_width="match_parent"  
                android:layout_height="match_parent">  
  
                <TextView  
                    android:id="@+id/textViewKisiBilgi"  
                    android:layout_width="wrap_content"  
                    android:layout_height="wrap_content"  
                    android:layout_centerHorizontal="true"  
                    android:layout_centerVertical="true"  
                    android:text='@(kisiNesnesi.kisi_ad" - "+kisiNesnesi.kisi_tel)'  
                    app:layout_constraintBottom_toBottomOf="parent"  
                    app:layout_constraintEnd_toEndOf="parent"  
                    app:layout_constraintHorizontal_bias="0.5"  
                    app:layout_constraintStart_toStartOf="parent"  
                    app:layout_constraintTop_toTopOf="parent" />  
  
            </androidx.constraintlayout.widget.ConstraintLayout>  
        </androidx.cardview.widget.CardView>  
    </androidx.constraintlayout.widget.ConstraintLayout>  
</layout>
```

Card Tasarımı

RecyclerView Adapter

```
class KisilerAdapter(private val mContext: Context, private val kisilerListe: List<Kisiler>)
    : RecyclerView.Adapter<KisilerAdapter.CardTasarimTutucu>() {

    inner class CardTasarimTutucu(cardTasarimBinding: CardTasarimBinding)
        : RecyclerView.ViewHolder(cardTasarimBinding.root) {
        var cardTasarimBinding: CardTasarimBinding
            init {
                this.cardTasarimBinding = cardTasarimBinding
            }
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): CardTasarimTutucu {
        val layoutInflater = LayoutInflater.from(mContext)
        val cardTasarimBinding = CardTasarimBinding
            .inflate(layoutInflater, parent, attachToRoot: false)
        return CardTasarimTutucu(cardTasarimBinding)
    }

    override fun onBindViewHolder(holder: CardTasarimTutucu, position: Int) {
        val kisi = kisilerListe.get(position)
        holder.cardTasarimBinding.kisiNesnesi = kisi
    }

    override fun getItemCount(): Int {
        return kisilerListe.size
    }
}
```

*card tasarim dosyasi ismine
göre bu sınıf ismi otomatik
olur.*

Activity

```
class MainActivity : AppCompatActivity() {  
    private lateinit var adapter: KisilerAdapter  
    private val viewModel: MainActivityViewModel by viewModels()  
  
    private lateinit var tasarimBinding: ActivityMainBinding  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        tasarimBinding = DataBindingUtil  
            .setContentView( activity: this, R.layout.activity_main)  
  
        viewModel.kisilerListesi.observe( owner: this, { kisilerListesi ->  
            adapter = KisilerAdapter( mContext: this,kisilerListesi)  
            tasarimBinding.kisilerAdapter = adapter  
        })  
    }  
}
```

ViewModelin bize verdiği kişi listesini adaptera aktarırız ve adapterde data binding sayesinde tasarıma aktarırız.

```
<?xml version="1.0" encoding="utf-8"?>  
<layout xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:app="http://schemas.android.com/apk/res-auto"  
    xmlns:tools="http://schemas.android.com/tools">  
  
    <data>  
        <variable name="kisilerAdapter"  
            type="com.example.kisileruygulamasidatabinding.KisilerAdapter" />  
    </data>  
  
    <androidx.constraintlayout.widget.ConstraintLayout  
        android:layout_width="match_parent"  
        android:layout_height="match_parent"  
        tools:context=".MainActivity">  
  
        <androidx.recyclerview.widget.RecyclerView  
            android:id="@+id/rv"  
            android:layout_width="0dp"  
            android:layout_height="0dp"  
            android:adapter="@{kisilerAdapter}"  
            app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"  
            app:layout_constraintBottom_toBottomOf="parent"  
            app:layout_constraintLeft_toLeftOf="parent"  
            app:layout_constraintRight_toRightOf="parent"  
            app:layout_constraintTop_toTopOf="parent" />  
  
    </androidx.constraintlayout.widget.ConstraintLayout>  
</layout>
```

DataBinding Olmadan

```
class MainActivity : AppCompatActivity() {  
    private lateinit var adapter: KisilerAdapter  
    private val viewModel: MainActivityViewModel by viewModels()  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)  
  
        rv.layoutManager = LinearLayoutManager( context: this)  
  
        viewModel.kisilerListesi.observe( owner: this, { kisilerListesi ->  
            adapter = KisilerAdapter( mContext: this,kisilerListesi)  
            rv.adapter = adapter  
        })  
    }  
}
```

DataBinding Uygulandıktan Sonra

```
class MainActivity : AppCompatActivity() {  
    private lateinit var adapter: KisilerAdapter  
    private val viewModel: MainActivityViewModel by viewModels()  
  
    private lateinit var tasarimBinding:ActivityMainBinding  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        tasarimBinding = DataBindingUtil  
            .setContentView( activity: this, R.layout.activity_main)  
  
        viewModel.kisilerListesi.observe( owner: this, { kisilerListesi ->  
            adapter = KisilerAdapter( mContext: this,kisilerListesi)  
            tasarimBinding.kisilerAdapter = adapter  
        })  
    }  
}
```

Kişiler Uygulaması Retrofit

-

MVVM

Retrofit Kurulumu Yapılması

- Bu işlemleri İnternet Tabanlı işlemler bölümünde yer alan Kişiler Uygulaması – Retrofit dersini izleyerek öğrenebilirsiniz.
- Konumuz gereği bunlara deðinip konun daðılmasını istemiyoruz.
- Asıl amacımız MVVM yapısını kullanmak.

Tüm Verileri Okuma

```
class KisilerDaoRepository {

    private val kisilerListesi: MutableLiveData<List<Kisiler>>
    private val kisilerdaoInterface: KisilerDaoInterface

    init {
        kisilerdaoInterface = ApiUtils.getKisilerDaoInterface()
        kisilerListesi = MutableLiveData()
    }

    fun kisilerSonuc(): MutableLiveData<List<Kisiler>> {
        return kisilerListesi
    }

    fun kisileriAl() {
        kisilerdaoInterface.tumKisiler().enqueue(object : Callback<KisilerCevap> {
            override fun onResponse(call: Call<KisilerCevap>, response: Response<KisilerCevap>) {
                val liste = response.body().kisiler
                kisilerListesi.value = liste
            }
            override fun onFailure(call: Call<KisilerCevap>, t: Throwable) {}
        })
    }
}
```

```
class MainActivityViewModel : ViewModel() {

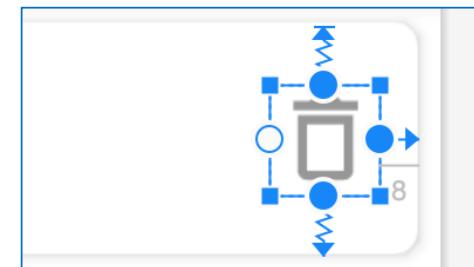
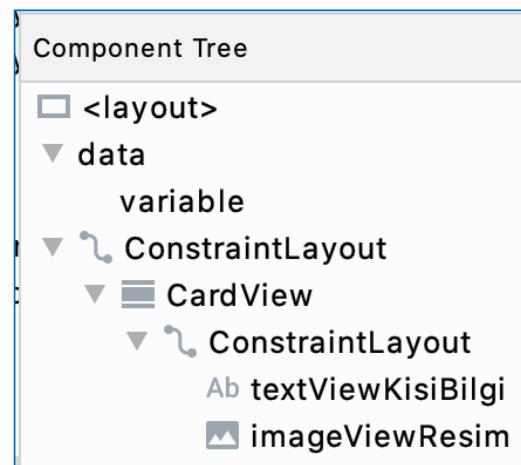
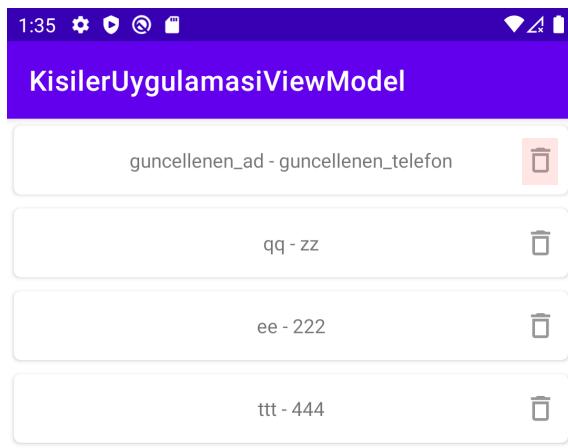
    var kisilerListesi = MutableLiveData<List<Kisiler>>()
    private val kdoar = KisilerDaoRepository()

    init {
        kisileriYukle()
        kisilerListesi = kdoar.kisilerSonuc()
    }

    fun kisileriYukle() {
        kdoar.kisileriAl()
    }
}
```

Repository ortak listesinin buradaki listeye bağlanması. Artık Repository içinde listede değişim olduğu anda bu listeyi yenilicek ve bu listede activity içindeki dinleme kısmını tetiklicek.

Kayıt Silme - Tasarım



Adapter Kodlaması

```
override fun onBindViewHolder(holder: CardTasarimTutucu, position: Int) {
    val kisi = kisilerListe.get(position)
    holder.cardTasarimBinding.kisiNesnesi = kisi

    holder.cardTasarimBinding.imageViewResim.setOnClickListener { it: View!
    }
}
```

Kayıt Silme - Kodlama

```
class KisilerDaoRepository {

    private val kisilerListesi: MutableLiveData<List<Kisiler>>
    private val kisilerdaoInterface: KisilerDaoInterface

    init {
        kisilerdaoInterface = ApiUtils.getKisilerDaoInterface()
        kisilerListesi = MutableLiveData()
    }

    fun kisilerSonuc(): MutableLiveData<List<Kisiler>> {
        return kisilerListesi
    }

    fun kisiSil(kisi_id: Int) {
        kisilerdaoInterface.kisiSil(kisi_id).enqueue(object : Callback<CRUDCevap?> {
            override fun onResponse(call: Call<CRUDCevap?>, response: Response<CRUDCevap?>) {
                kisileriAl()
            }
            override fun onFailure(call: Call<CRUDCevap?>, t: Throwable) {}
        })
    }

    fun kisileriAl() {
        kisilerdaoInterface.tumKisiler().enqueue(object : Callback<KisilerCevap> {
            override fun onResponse(call: Call<KisilerCevap>, response: Response<KisilerCevap>) {
                val liste = response.body().kisiler
                kisilerListesi.value = liste
            }
            override fun onFailure(call: Call<KisilerCevap>, t: Throwable) {}
        })
    }
}

class MainActivityViewModel : ViewModel() {

    var kisilerListesi = MutableLiveData<List<Kisiler>>()
    private val kdoar = KisilerDaoRepository()

    init {
        kisileriYukle()
        kisilerListesi = kdoar.kisilerSonuc()
    }

    fun kisileriYukle() {
        kdoar.kisileriAl()
    }

    fun sil(kisi_id: Int) {
        kdoar.kisiSil(kisi_id)
    }
}
```

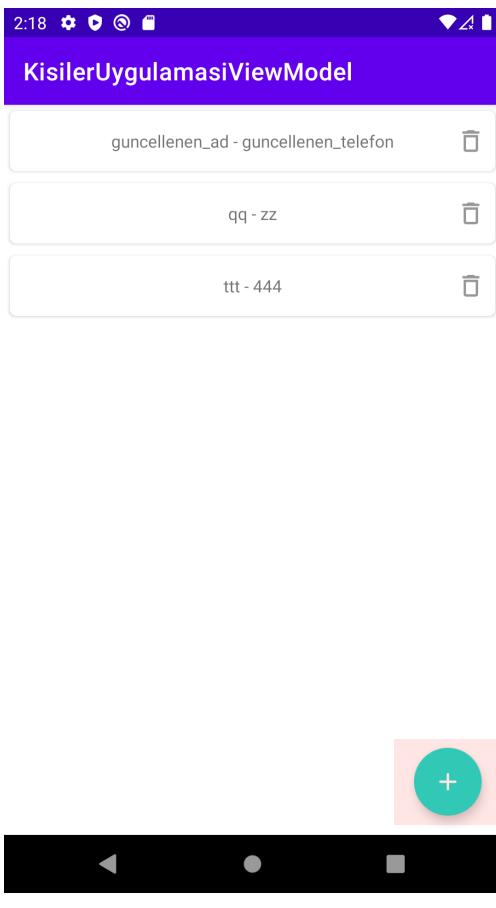
Kayıt Silme - Kodlama

```
class KisilerAdapter(private val mContext: Context,  
                     private val kisilerListe: List<Kisiler>,  
                     private val viewModel: MainActivityViewModel)  
    : RecyclerView.Adapter<KisilerAdapter.CardTasarimTutucu>() {
```

```
override fun onBindViewHolder(holder: CardTasarimTutucu, position: Int) {  
    val kisi = kisilerListe.get(position)  
    holder.cardTasarimBinding.kisiNesnesi = kisi  
  
    holder.cardTasarimBinding.imageViewResim.setOnClickListener { it: View!  
        viewModel.sil(kisi.kisi_id)  
    }  
}
```

```
viewModel.kisilerListesi.observe(owner: this, { kisilerListesi ->  
    adapter = KisilerAdapter( mContext: this, kisilerListesi, viewModel)  
    tasarimBinding.kisilerAdapter = adapter  
})
```

Kayıt Ekleme - Tasarım



```
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools">

    <data>
        <variable name="kisilerAdapter" type="com.example.kisileruygulamasiviewmodel.KisilerAdapter" />
        <variable name="mainActivityNesnesi" type="com.example.kisileruygulamasiviewmodel.MainActivity" />
    </data>

    class MainActivity : AppCompatActivity() {
        private lateinit var adapter: KisilerAdapter
        private val viewModel: MainActivityViewModel by viewModels()

        private lateinit var tasarimBinding: ActivityMainBinding

        override fun onCreate(savedInstanceState: Bundle?) {
            super.onCreate(savedInstanceState)
            tasarimBinding = DataBindingUtil
                .setContentView( activity: this, R.layout.activity_main)
            tasarimBinding.mainActivityNesnesi = this

            viewModel.kisilerListesi.observe( owner: this, { kisilerListesi ->
                adapter = KisilerAdapter( mContext: this,kisilerListesi,viewModel)
                tasarimBinding.kisilerAdapter = adapter
            })
        }

        fun fabTikla(view:View){
        }
    }
}
```

```
<com.google.android.material.floatingactionbutton.FloatingActionButton
    android:id="@+id/fab"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginEnd="16dp"
    android:layout_marginRight="16dp"
    android:layout_marginBottom="16dp"
    android:clickable="true"
    app:tint="@color/white"
    android:onClick="@{mainActivityNesnesi.fabTikla}"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:srcCompat="@drawable/ekle_resim" />

</androidx.constraintlayout.widget.ConstraintLayout>
</layout>
```

Kayıt Ekleme - Kodlama

```
class KisilerDaoRepository {

    private val kisilerListesi: MutableLiveData<List<Kisiler>>
    private val kisilerdaoInterface: KisilerDaoInterface

    init {
        kisilerdaoInterface = ApiUtils.getKisilerDaoInterface()
        kisilerListesi = MutableLiveData()
    }

    fun kisilerSonuc(): MutableLiveData<List<Kisiler>> {
        return kisilerListesi
    }

    fun kisiKayit(kisi_ad: String, kisi_tel: String) {
        kisilerdaoInterface.kisiEkle(kisi_ad, kisi_tel).enqueue(object : Callback<CRUDCevap?> {
            override fun onResponse(call: Call<CRUDCevap?>, response: Response<CRUDCevap?>) {
                kisileriAl()
            }
            override fun onFailure(call: Call<CRUDCevap?>, t: Throwable) {}
        })
    }

    class MainActivityViewModel : ViewModel() {

        var kisilerListesi = MutableLiveData<List<Kisiler>>()
        private val kdoar = KisilerDaoRepository()

        init {
            kisileriYukle()
            kisilerListesi = kdoar.kisilerSonuc()
        }

        fun kayit(kisi_ad: String, kisi_tel: String) {
            kdoar.kisiKayit(kisi_ad, kisi_tel)
        }

        fun kisileriYukle() {
            kdoar.kisileriAl()
        }

        fun sil(kisi_id: Int) {
            kdoar.kisiSil(kisi_id)
        }
    }
}
```

```
class MainActivity : AppCompatActivity() {
    private lateinit var adapter: KisilerAdapter
    private val viewModel: MainActivityViewModel by viewModels()

    private lateinit var tasarimBinding: ActivityMainBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        tasarimBinding = DataBindingUtil
            .setContentView( activity: this, R.layout.activity_main)
        tasarimBinding.mainActivityNesnesi = this

        viewModel.kisilerListesi.observe( owner: this, { kisilerListesi ->
            adapter = KisilerAdapter( mContext: this, kisilerListesi, viewModel)
            tasarimBinding.kisilerAdapter = adapter
        })
    }

    fun fabTikla(view: View){
        viewModel.kayit( kisi_ad: "Test AD", kisi_tel: "Test TEL")
    }
}
```

Teşekkürler...



kasım-adalan



kasimadalan@gmail.com



kasimadalan