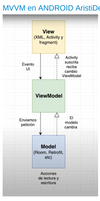
Patron MVVM

MVVM en ANDROID AristiDevs



En Gradle(app)

```
android {
    ...
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }

    kotlinOptions {
        jvmTarget = '1.8'
    }
    buildFeatures{
        viewBinding = true
    }
}
dependencies{
    ...
    // Fragment
    implementation "androidx.fragment:fragment-ktx:1.3.2"
    // Activity
    implementation "androidx.activity:activity-ktx:1.2.2"
    // ViewModel
    implementation "androidx.lifecycle:lifecycle-viewmodel-ktx:2.3.1"
    // LiveData
    implementation "androidx.lifecycle:lifecycle-livedata-ktx:2.3.1"
}
```

ViewBinding

```
class MainActivity : AppCompatActivity() {
    // crea propiedad
    private lateinit var binding: ActivityMainBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        // asocia a la vista
        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)
    }
}
```

Añadir a la vista elementos para ver los valores de las propiedades de nuestra clase modelo..activity_main.xml

```
    <?xml version="1.0" encoding="utf-8"?>
    <androidx.constraintlayout.widget.ConstraintLayout
        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"
        android:id="@+id/viewContainer"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="@color/background"
        tools:context=".view.MainActivity">

        <TextView
            android:id="@+id/tvQuote"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:gravity="center"
            android:padding="16dp"
            android:textColor="@color/white"
            android:textSize="24sp"
            android:textStyle="italic"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintLeft_toLeftOf="parent"
            app:layout_constraintRight_toRightOf="parent"
            app:layout_constraintTop_toTopOf="parent" />

        <TextView
            android:id="@+id/tvAuthor"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:gravity="center"
            android:padding="16dp"
            android:textColor="@color/white"
            android:textSize="24sp"
            android:textStyle="italic"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintLeft_toLeftOf="parent"
            app:layout_constraintRight_toRightOf="parent" />

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

- **Línea 5:** se añade un id al constraint para poder hacer un onclick
- los colores hay que crealos ALT+ INTRO > create color value resource

Model

- en la carpeta del MainActivity creamos una **carpeta model** y creamos una data class

```
data class QuoteModel (val quote:String, val author:String)
```

QuoteProvider

- **Carpeta model** creamos el quoteProvider lista de modelos mas funcion que devuelve una posicion aleatoria

```
class QuoteProvider {
companion object {
    fun random(): QuoteModel {
        val position = (0..(quotes.size-1)).random()
        return quotes[position]
    }
    private val quotes = listOf(
        QuoteModel(
            quote = "It's not a bug.
            It's an undocumented feature!",
            author = "Anonymous"
        ),
        QuoteModel(
            quote = "“Software Developer” – An organism that
            turns caffeine into software",
            author = "Anonymous"
        ),
        QuoteModel(
            quote = "If debugging is the process of removing
            software bugs, then programming must be the process
            of putting them in",
            author = "Edsger Dijkstra"
        ),
        QuoteModel(
            quote = "A user interface is like a joke.
            If you have to explain it, it's not that good.",
            author = "Anonymous"
        ),
        QuoteModel(
            quote = "I don't care if it works on your machine!
            We are not shipping your machine!",
            author = "Vidiu Platon"
        ),
        QuoteModel(
            quote = "Measuring programming progress by lines of code
            is like measuring aircraft building progress by weight.",
            author = "Bill Gates"
        ),
        QuoteModel(
            quote = "My code DOESN'T work, I have no idea why.
            My code WORKS, I have no idea why.",
            author = "Anonymous"
        ),
        QuoteModel(
            quote = "Things aren't always #000000 and #FFFFFF",
            author = "Anonymous"),
        QuoteModel(
            quote = "Talk is cheap. Show me the code.",
            author = "Linus Torvalds"),
        QuoteModel(
            quote = "Software and cathedrals are much the same –
            first we build them, then we pray.",
            author = "Anonymous"
        ),
        QuoteModel(
            quote = "¿A que esperas?, suscribete.",
             author = "AristiDevs")
    )
}
}
```

ViewModel

- crear directorio viewModel y clase QuoteViewModel

```
    class QuoteViewModel : ViewModel() {

        val quoteModel = MutableLiveData<QuoteModel>()

        fun randomQuote() {
            val currentQuote = QuoteProvider.random()
            quoteModel.postValue(currentQuote)
        }
    }
```

- **Línea 3: LiveData** tipo de datos que detecta si hay un cambio en nuestro modelo.
- **Línea 7 postValue** actualiza el valor cada vez que se ejecuta el hilo principal

View

- **MainActivity**

```
    class MainActivity : AppCompatActivity() {
        private lateinit var binding: ActivityMainBinding
        private val quoteViewModel: QuoteViewModel by viewModels()
        override fun onCreate(savedInstanceState: Bundle?) {
            super.onCreate(savedInstanceState)
            binding = ActivityMainBinding.inflate(layoutInflater)
            setContentView(binding.root)

            quoteViewModel.quoteModel.observe(this, Observer {
                binding.tvQuote.text = it.quote
                binding.tvAuthor.text = it.author
            })

            binding.viewContainer.setOnClickListener {
                quoteViewModel.randomQuote()
                }

        }

    }
```

- **Línea 3:** declaramos una variable que delege **"by"** en viewModels gracias a una de las porpiedades que hemos implementado
- **Líneas 9-12:** ponemos un observer que **ejecutara el codigo que queramos cada vez que el valor de "quoteModel" se actualice**, en este caso darle valor al texto de las vista
- **Líneas 14:** ponemos una escucha para que cada vez que se pulse la pantalla se cambie el valor de quotemodel