Android MVP Architecture pattern using Kotlin

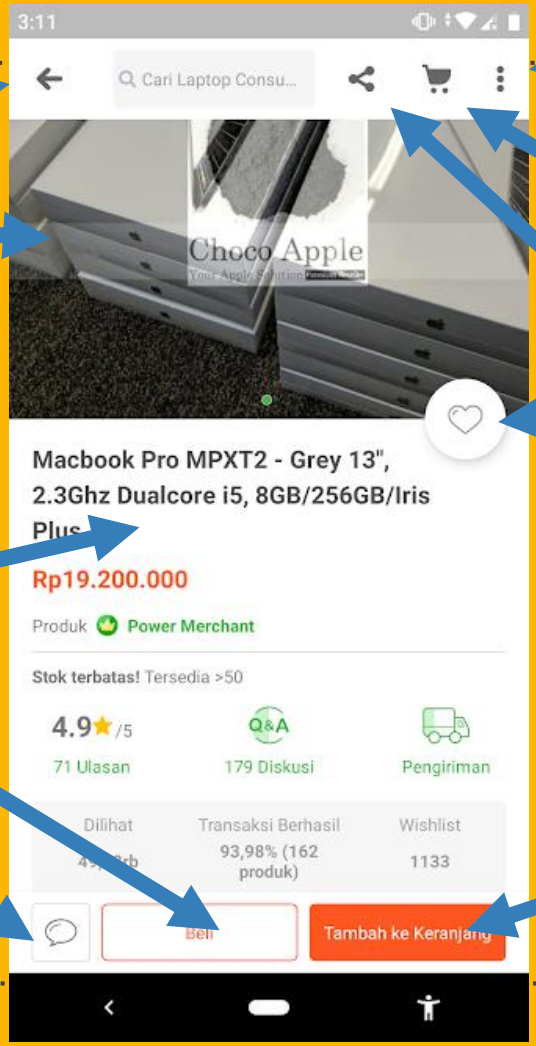
Prieyudha Akadita S
medium.com/@ydhnwb

What is Archictecture Pattern?

Sebuah konsep arsitektur pengembangan aplikasi yang **memisahkan** antara **tampilan/UI** dengan **proses bisnis/LOGIC** yang bekerja pada aplikasi.

Arsitektur ini akan membuat pengembangan aplikasi kita menjadi **lebih terstuktur**, mudah di-*test* dan juga mudah di-*maintain.*


3:11
Cari Laptop Consu...
Choco Apple
Macbook Pro MPXT2 - Grey 13", 2.3Ghz Dualcore i5, 8GB/256GB/Iris Plus
Rp19.200.000
Produk
Power Merchant
Stok terbatas! Tersedia >50
4.9 /5
71 Ulasan
Q&A
179 Diskusi
Pengiriman
Dilihat
49,28rb
Transaksi Berhasil
93,98% (162 produk)
Wishlist
1133
Beli
Tambah ke Keranjang



finish()

setupImageSlider()

fetchData()

bind()

buyItem()

goToChat()

onOptionsSelected()

myCart()

shareMenu()

addToFav()

addToCart()

```
class ProductDetalActivity extends AppCompatActivity{
    private Textview title;
    // tergantung banyaknya komponen ui

    onCreate(){
            findViewById…
    }

    int getIdItem(){  getIntent……     }

    private void fetchData(){
            loading.setVisibility(View.VISIBLE);
            request.enqeue…
            // +35 lines or more code
    }

    private void bind(Product product){
            title.setText(product.getTitle());
            // +10 lines or more code
    }

    //more more and more code
}
```

Berhubungan dg UI

Berhubungan dg Logic

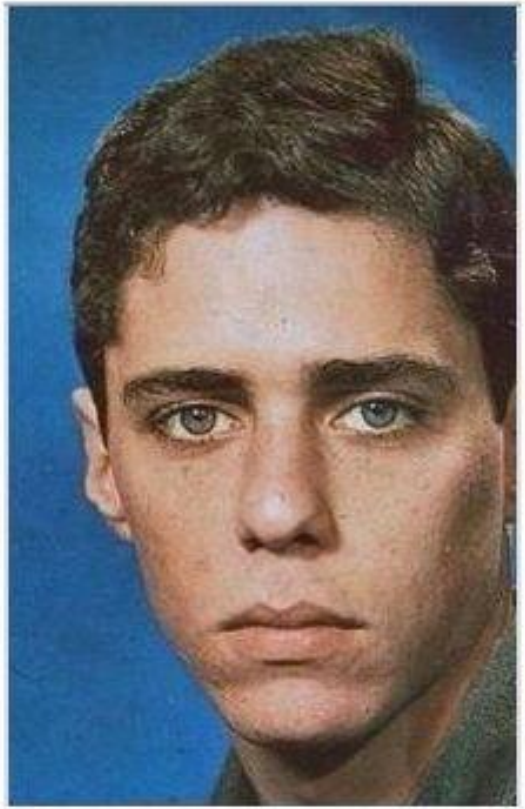
Without pattern

logic

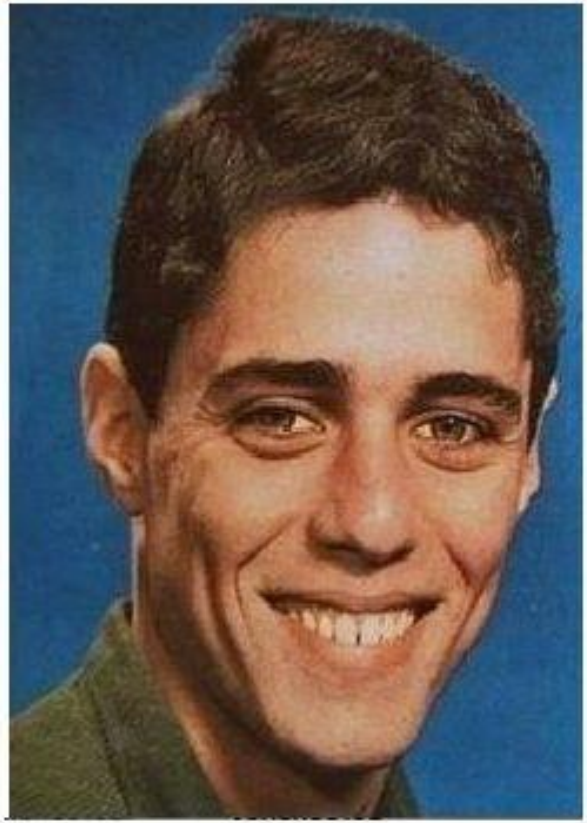
ui

With pattern

Software tester



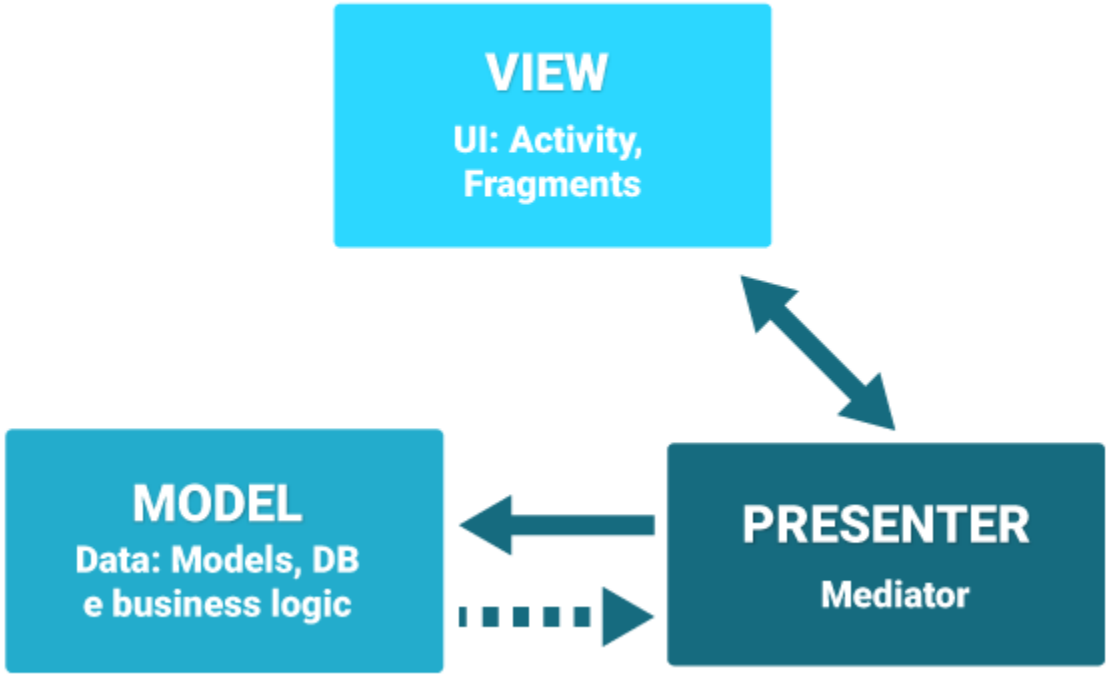
Before using pattern



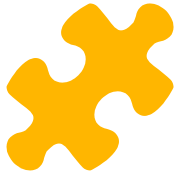
After using pattern

MVP concept

Model View Presenter



contracts



Contracts adalah suatu **perjanjian** mengenai apa-apa saja yang bisa dilakukan oleh **View** dan **Presenter.**

Secara **teknis**, Contracts adalah sebuah **Interface** yang dibuat untuk **UI** dan **Logic** .

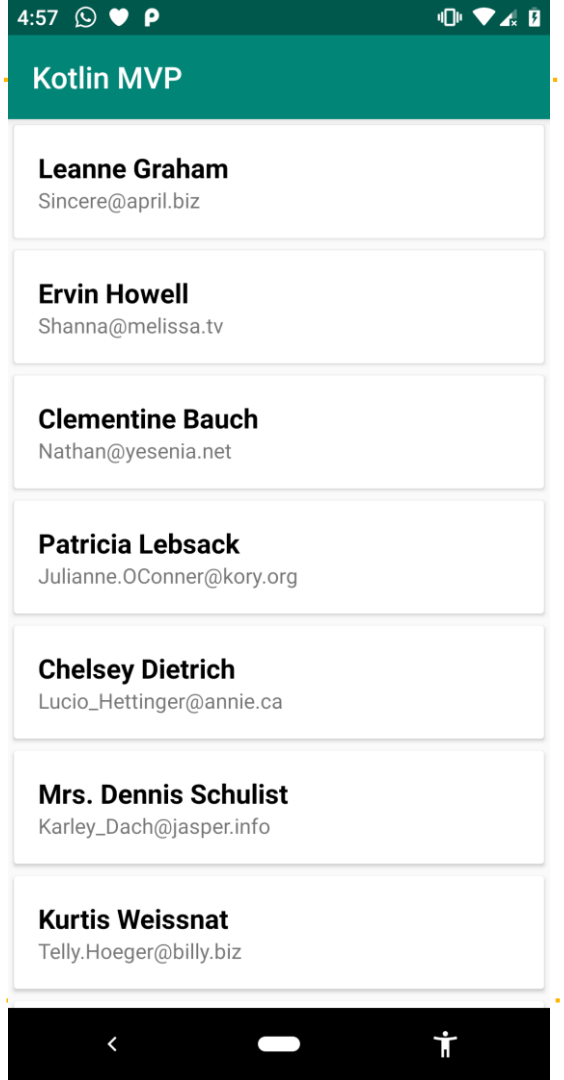
Contracts **merepresentasikan Activity** yang kita buat. Jika kita punya **MainActivity**, kita **perlu membuat MainActivityContract**. Jika suatu hari kita membuat **LoginActivity**, kita juga perlu membuat **LoginActivityContract** juga

View Contract
(diterapkan di Activity.java)

```
isLoading()

attachRecycler()

toast()
```



Logic Contract
(diterapkan pada ActivityPresenter.java)

```
fetchUser()

destroyView()
```

```
interface MainActivityContract {
    interface View {
        fun attachToRecycler(users : List<User>)
        fun toast(message : String)
        fun isLoading(state : Boolean)
    }

    interface Interactor {
        fun fetchUser()
        fun destroyView()
    }
}
```

```kotlin
class MainActivityPresenter(v : MainActivityContract.View) : MainActivityContract.Interactor { ...
    private var view : MainActivityContract.View? = v
    private var api = ApiClient.instance()

    override fun destroyView() { view = null }

    override fun fetchUser() {
        val request = api.allUsers()
        view?.isLoading( state: true)
        request.enqueue(object : Callback<List<User>>{
            override fun onFailure(call: Call<List<User>>, t: Throwable) {...}

            override fun onResponse(call: Call<List<User>>, response: Response<List<User>>) {
                if(response.isSuccessful){
                    val users = response.body()
                    users?.let { it: List<User>
                        view?.attachToRecycler(users)
                    }
                }else{
                    view?.toast("Kesalahan saat mengambil response")
                }
                view?.isLoading( state: false)
            }
        })
    }
}
```

Implement Contract.Interactor

Method hasil generate

```kotlin
class MainActivity : AppCompatActivity(), MainActivityContract.View {
    private var presenter = MainActivityPresenter( v: this)

    override fun attachToRecycler(users: List<User>) {
        rv_main.apply { this: RecyclerView!
            layoutManager = LinearLayoutManager( context: this@MainActivity)
            adapter = UserAdapter(users,  context: this@MainActivity)
        }
    }

    override fun toast(message: String) = Toast.makeText( context: this@MainActivity, message, Toast.LENGTH_LONG)

    override fun isLoading(state: Boolean) {...}

    private fun fetchUser() = presenter.fetchUser()

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

    override fun onDestroy() {...}
}
```

MainActivity implement Contract.View

Practice!