Android LiveData

Anton Saatze

University of applied Sciences Munich

22.may.2020

Table of Contents

Motivation

- core concept to present data
- clean way to hand data across the archictecture
- guarantee that latest data is presented in view
- first thing to learn, when joining a real android project team

Technical Background

- first thing to learn, when joining a real android project team
- 1/2 year internship in a android project
- learned android basics (Activty, Fragment, Lifecycle, Viewmodel, Views, LiveData, Databinding)
- my first task have been "redesign view xyz regarding to the designers wishes"

Technical Background

- learn how to design views with different layouts
- how to bind data to the view
- how adapter and lists work
- how to apply Data to a viewmodel
- how to publish Data by a viewmodel
- (Spoiler) its no magic, its LiveData

LiveData

> *LiveData is an observable data holder class. Unlike a regular observable, LiveData is lifecycle-aware, meaning it respects the lifecycle of other app components, such as activities, fragments, or services. This awareness ensures LiveData only updates app component observers that are in an active lifecycle state*

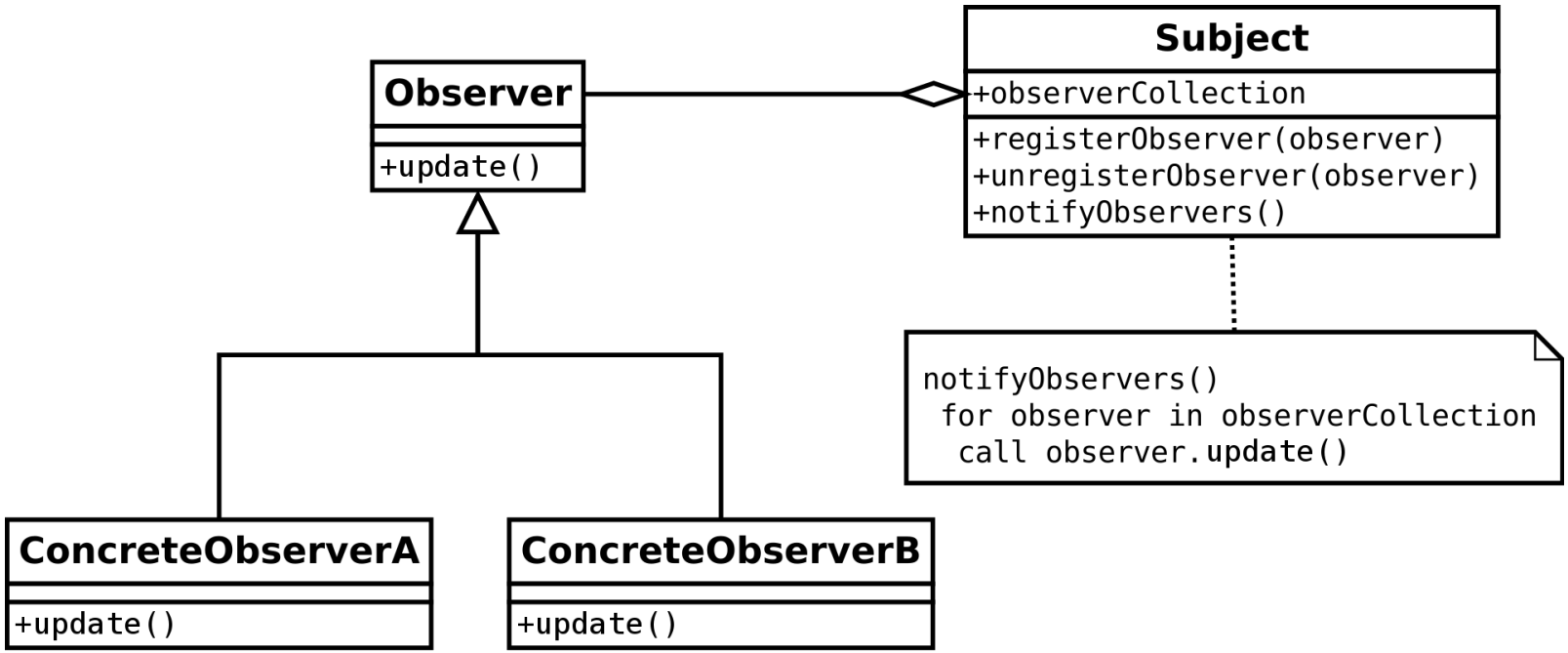
developer.android.com
(official documentaion of android)

LiveData

> **LiveData** *is an* **observable data holder** *class. Unlike a regular observable, LiveData is lifecycle-aware, meaning it respects the lifecycle of other app components, such as activities, fragments, or services. This awareness ensures LiveData only updates app component observers that are in an active lifecycle state*

developer.android.com
(official documentaion of android)

Observer Pattern



LiveData

> **LiveData** *is an* **observable data holder** *class. Unlike a regular observable,* **LiveData is lifecycle-aware**, *meaning it respects the lifecycle of other app components, such as activities, fragments, or services. This awareness ensures* **LiveData only updates** *app component observers that are in an* **active lifecycle** *state*

developer.android.com
(official documentaion of android)

LifeCycle Awarenes

- Observer almost everytime need an LifeCycle
- only exception when using "observeForever" method
- LifeCycles are provided by the Activity and/or Fragments
- Observer only observe while the passed LifeCycle is (STARTED/RESUMED)
- Observer do not observe when the LifeCycle is PAUSED or in a dead state

Fazit: make sure, to have an lifecycle when you want to observe

Code example

Create MutableLiveData

```
val liveData = MutableLiveData<String>().apply{
value = "Hello World"
}
```

Modify MutableLiveData with operations like *postValue()* or *setValue()* (which only can be called from the main-thread)

Create LiveData

```
@Query("SELECT * FROM PERSON ORDER BY NAME")
LiveData<List<Person>> loadAllPersons();
```

Code example

Create LiveData using MutableLiveData

```
private val privateLiveData =
MutableLiveData<String>().apply{
value = "Hello World"
}
val publicLiveData : LiveData<String> =
privateLiveData

fun changeString(string : String){
privateLiveData.postValue(string)
}
```

Code example

Observing LiveData with Observer object

```
val nameObserver = Observer<String> { newName ->
        nameTextView.text = newName
    }
publicLiveData.observe(lifeCycleOwner,
nameObserver)
```

Observing LiveData with Lambda

```
// not observed
val name = repository.publicLiveData
// make line above observing indirectly
val nameLetters = name.observe(
lifeCycleOwner,
Observer{it.toCharArray().asList()}
)
```

Observing Livedata + Databinding

Fragment

```
override fun onCreateView(
    inflater: LayoutInflater, container: ViewGroup?,
    savedInstanceState: Bundle?
): View? {
    val binding : FragmentDetailWithBindingBinding! = DataBindingUtil
        .inflate<FragmentDetailWithBindingBinding>(inflater,
            R.layout.fragment_detail_with_binding,
            container,
            attachToParent: false)
    binding.model = viewModel
    binding.lifecycleOwner = this
    return binding.root
}
```

XML

```
<data>

    <variable
        name="model"
        type="com.example.livedatapresentation.tshirt.screens.TShirtSelectorViewModel" />
</data>
```

Observing Livedata + Databinding

XML

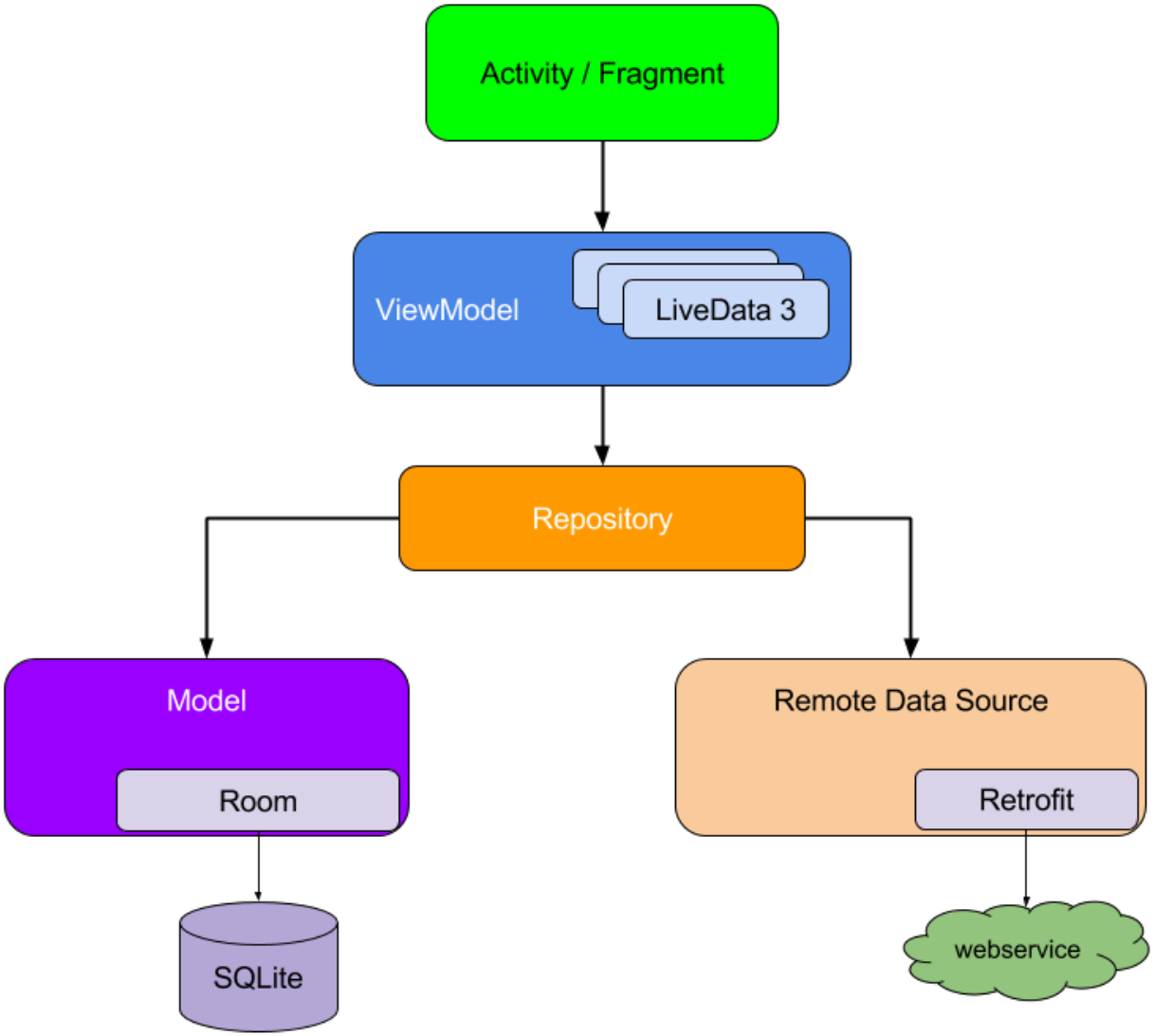
```
<ImageView
    android:id="@+id/t_shirt_image"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:scaleType="fitCenter"
    android:src="@drawable/t_shirt"
    android:tint="@{model.tshirtColor}"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

Observing Livedata + Databinding

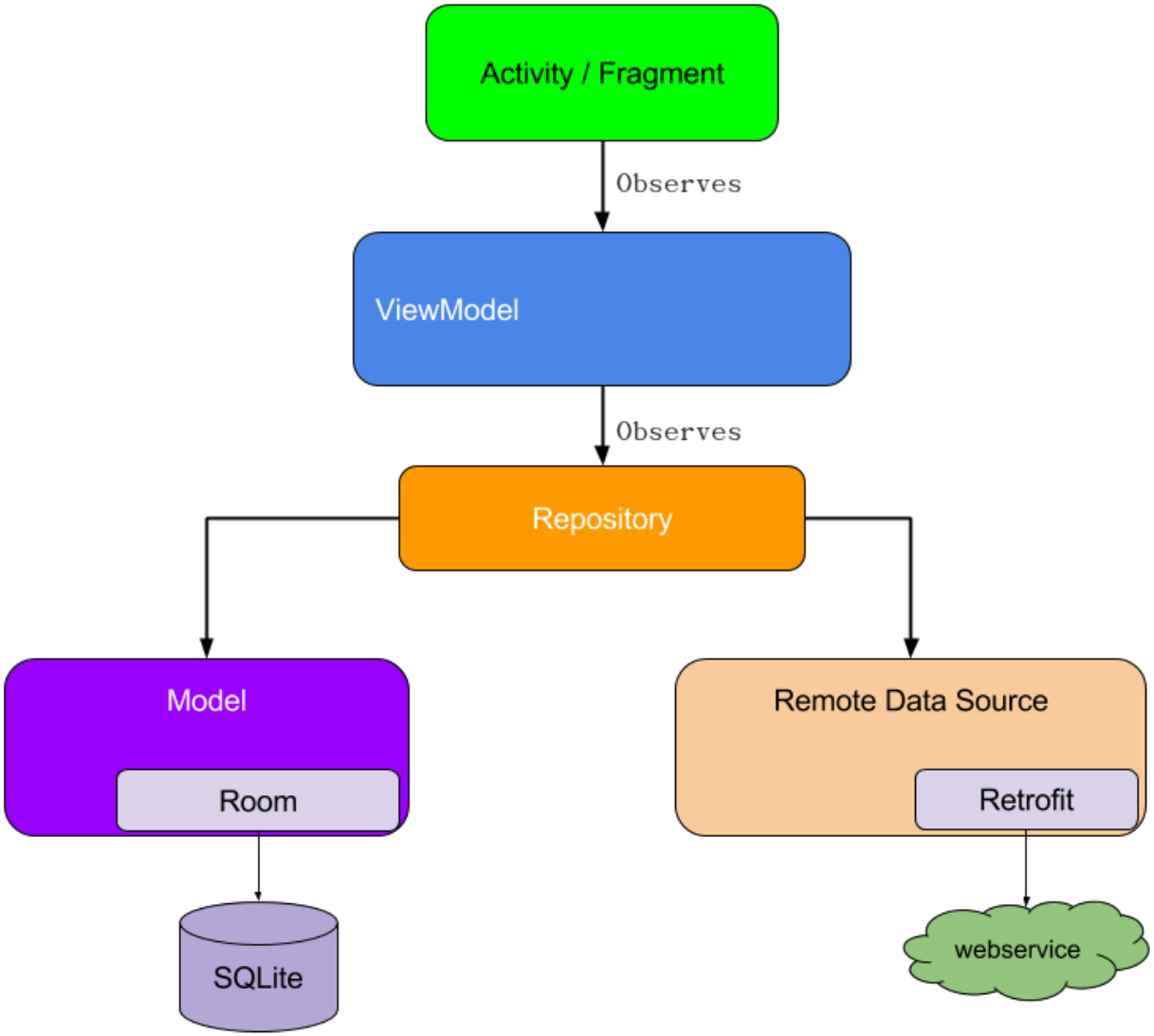
XML

```
<TextView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_weight="0.5"
    android:gravity="end"
    android:text="@{model.selectedTShirt.size.toString()}"
    android:textAlignment="textEnd"/>
</LinearLayout>
```

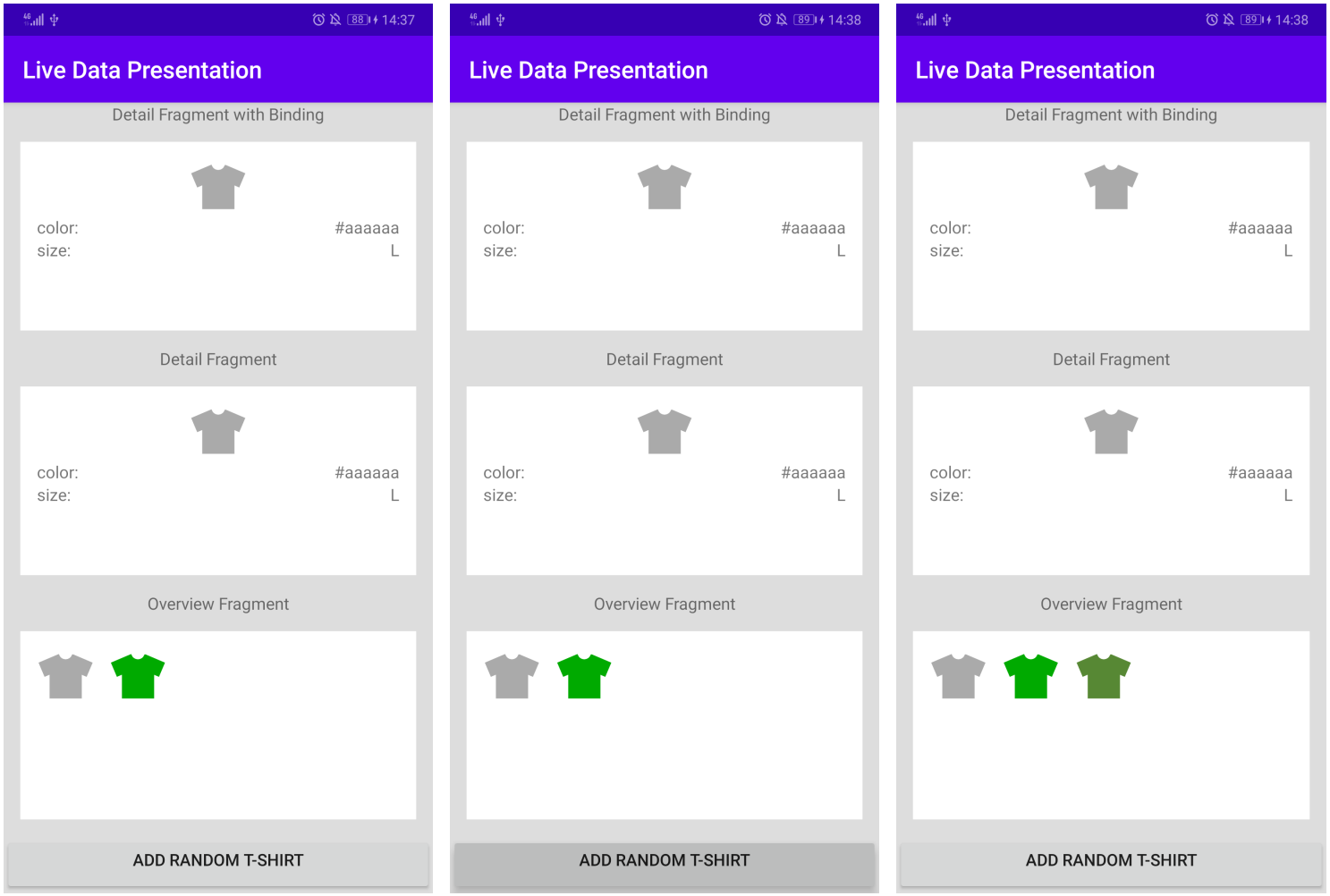
Overall Structure of Android Apps



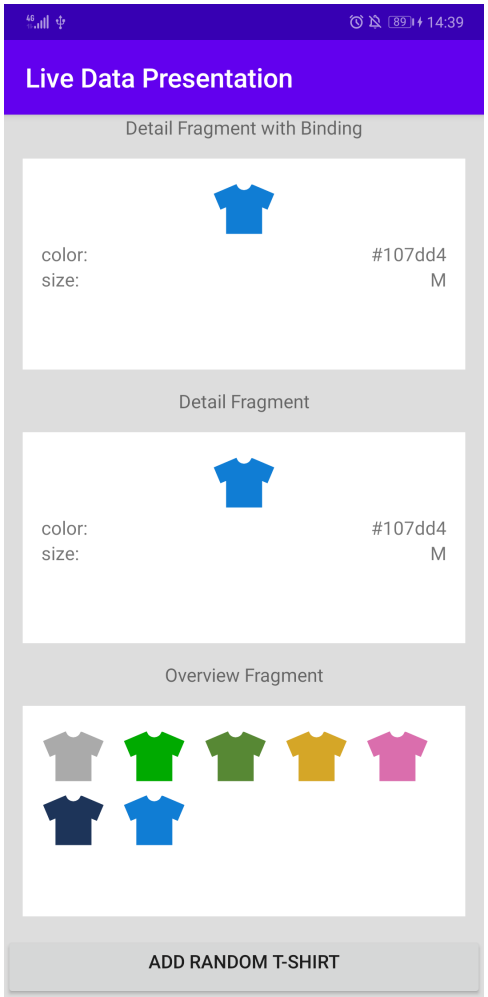
Overall Structure of Android Apps

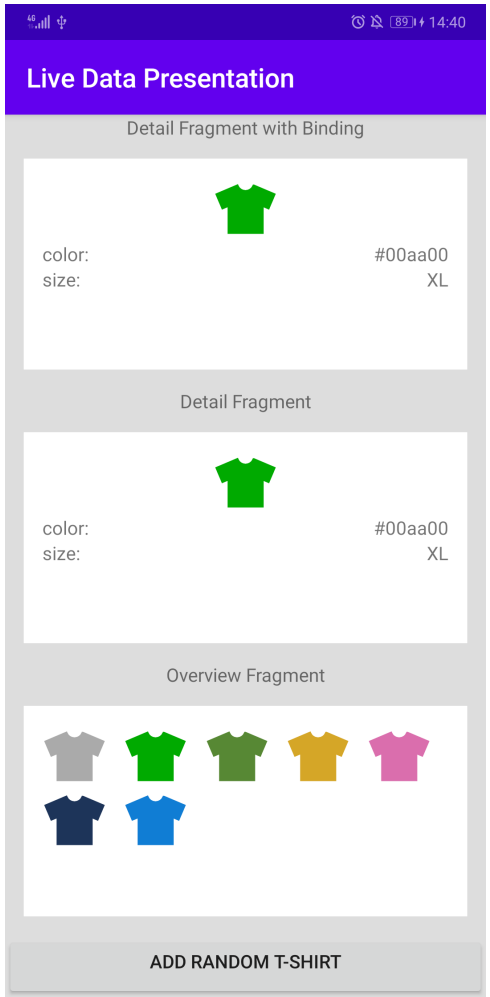


T-Shirt App Fragments

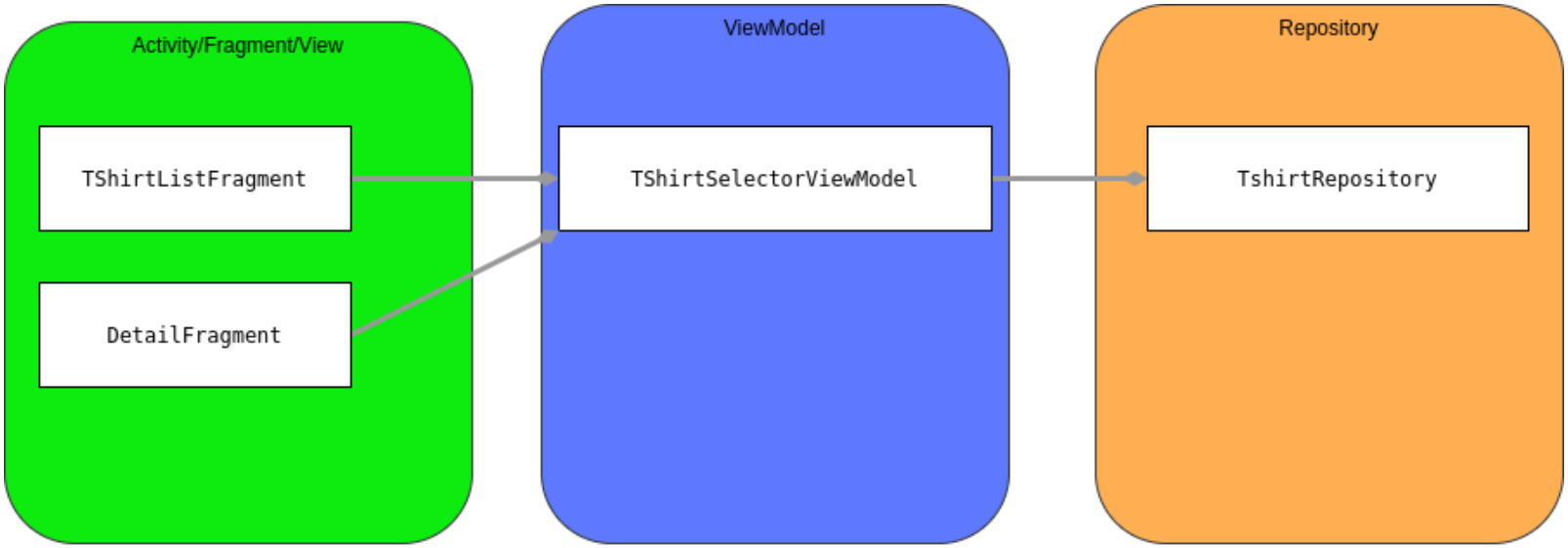


T-Shirt App Fragments

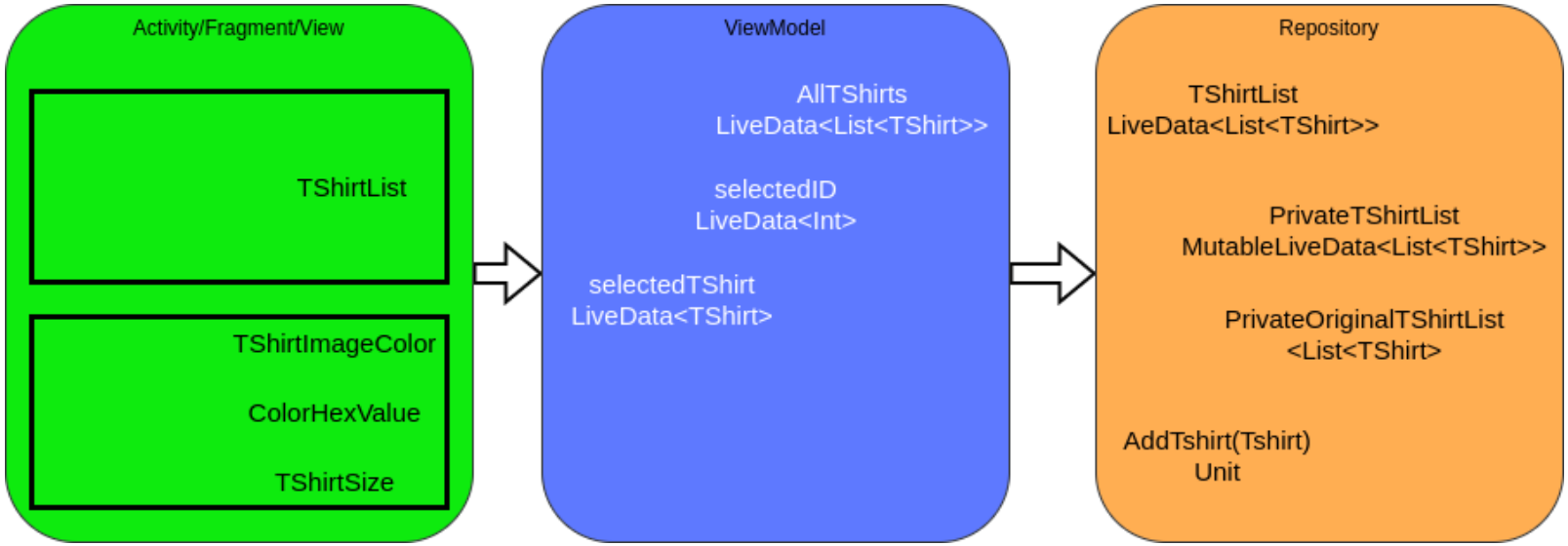




LiveData in the T-Shirt App



LiveData in the T-Shirt App



Summary

- check LifeCycle
  no observation when LifeCycle is Paused
- pass lifecycle to binding
- check if LiveData is observed
  not observed LiveData does not change
- check value is in proper Type in xml
  android:text = "@{viewmodel.number}"
  $\neq$
  android:text = "@{viewmodel.number.toString()}"
- check Livedata is not Mutable acidently

more information

- udemy.com/course/android-jetpack/
- medium.com/androiddevelopers/viewmodels-and-livedata-patterns-antipatterns-21efaef74a54

developer.android.com:

- /topic/libraries/architecture/livedata
- /reference/androidx/lifecycle/Transformations
- /jetpack/docs/guide

Questions?

Thank you for your attention

Feel free to ask me anything about

- LiveData
- Android General
- work in the internship

Repo:
github.com/Tosaa/mobile-application-presentation