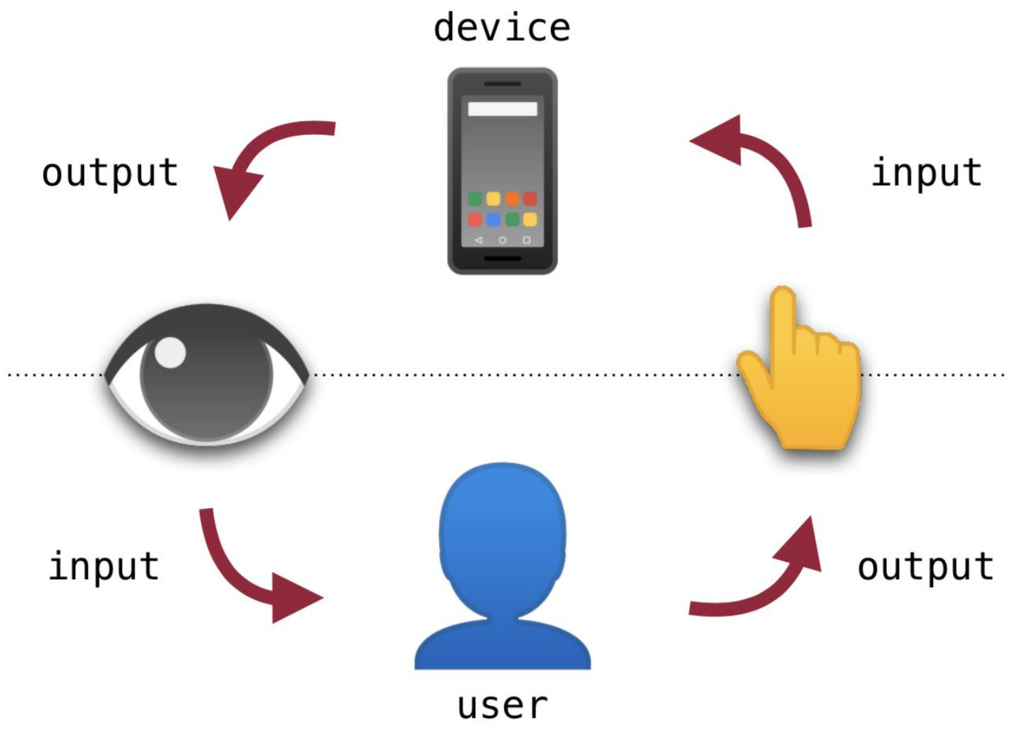
MVI FOR ANDROID

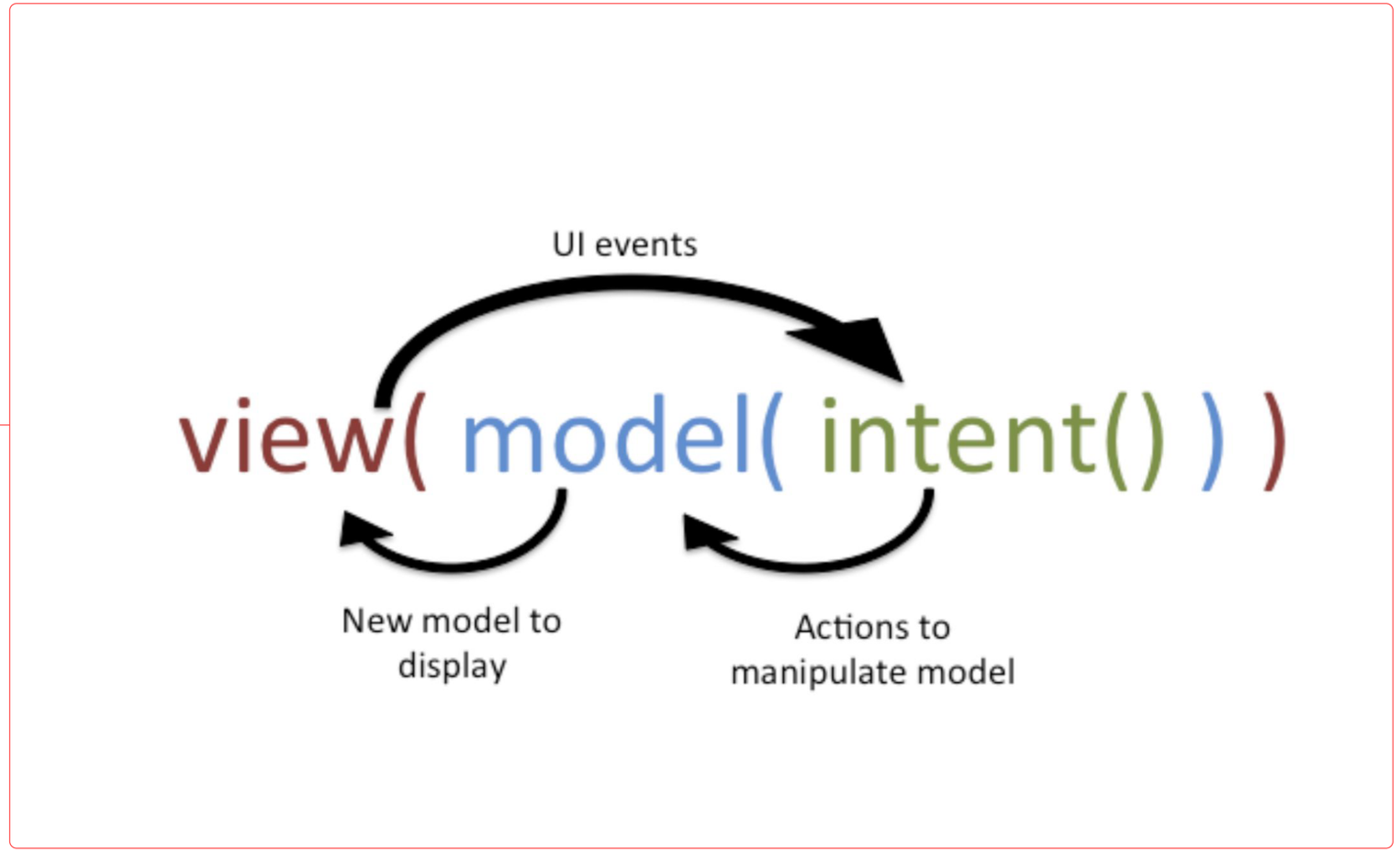
Miguel Alegria Android Developer

About me

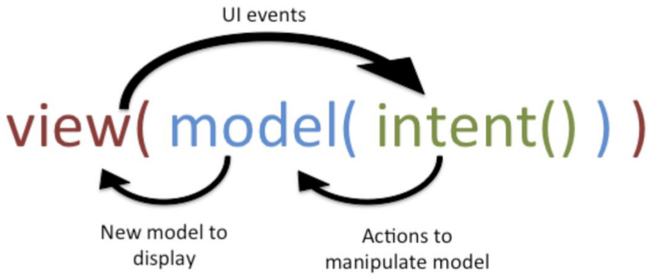
@MikelAlegria

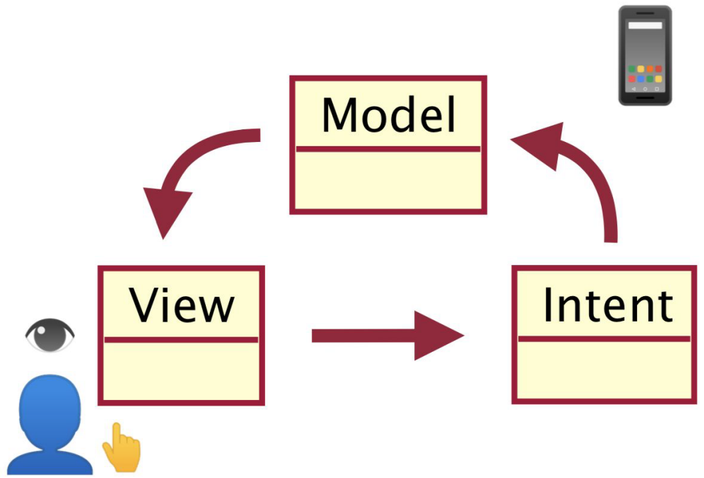
Android developer at Rappi

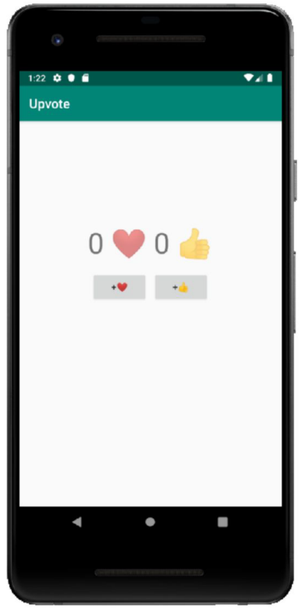

device
output
input
input
output
user

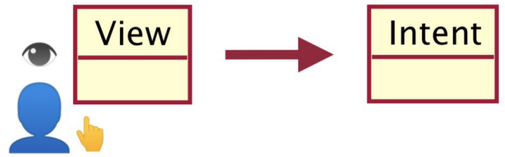

UI events
view(model(intent()))
New model to display
Actions to manipulate model

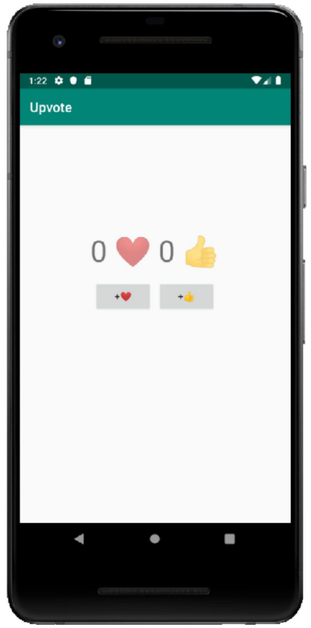
- View -> Fragment/Activity
- Model -> Data/POJO/Sealed
- Intent -> Magic


UI events
view(model(intent()))
New model to display
Actions to manipulate model

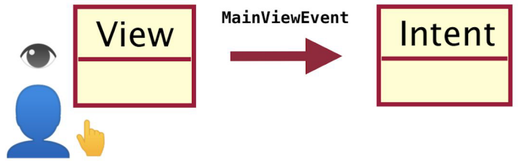

Model
View
Intent

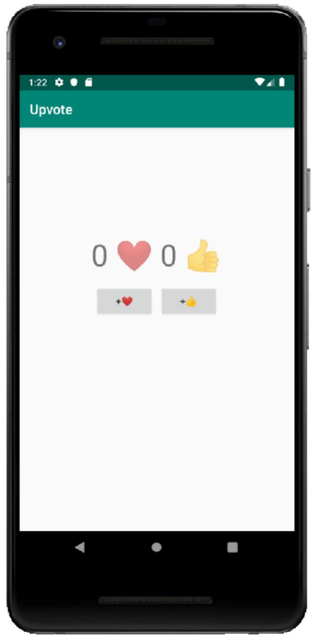

1:22
Upvote
0
0


View
Intent



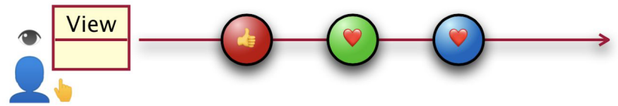

```
sealed class MainViewEvent {
    object ThumbsUpClick : MainViewEvent()
    object LoveItClick : MainViewEvent()
}
```

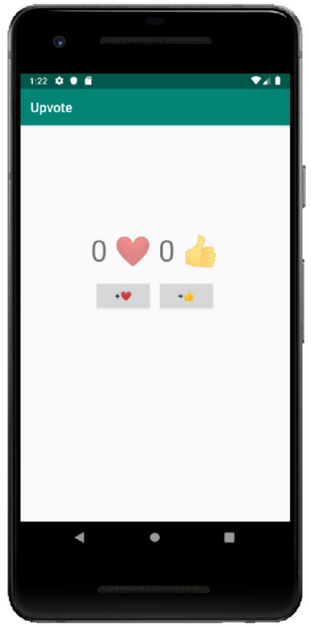




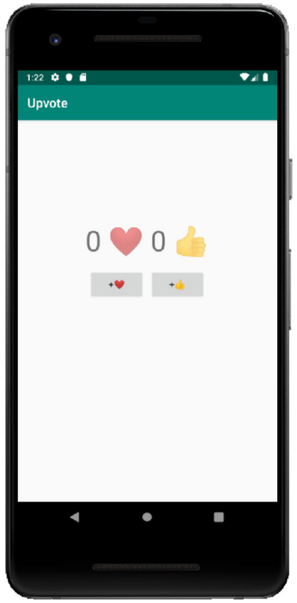

```
sealed class MainViewEvent {
    object ThumbsUpClick : MainViewEvent()
    object LoveItClick : MainViewEvent()
}
```

MainViewEvent
+👍 ThumbsUpClick
+❤️ LoveItClick

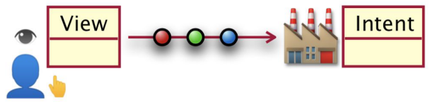




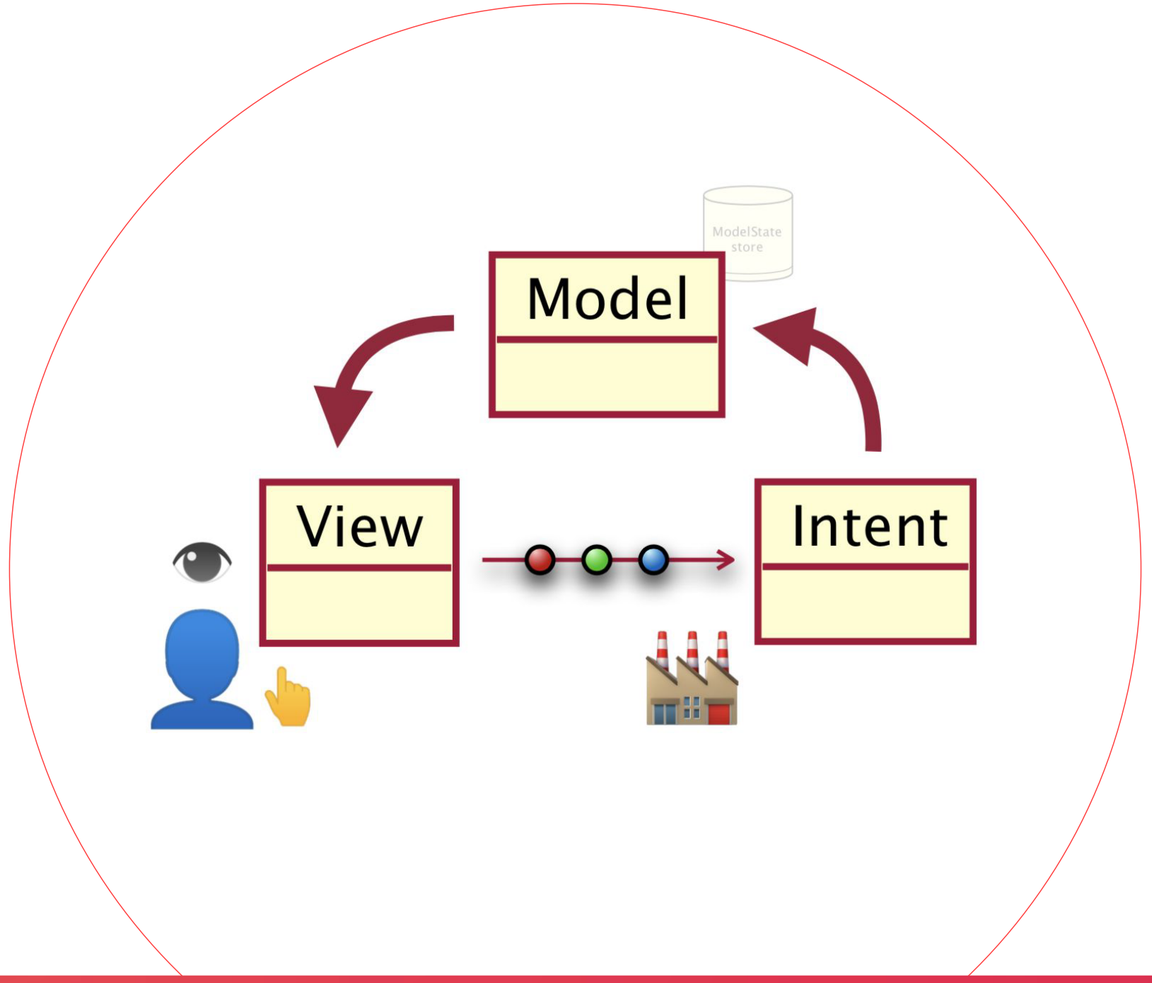

```
class MainActivity : ViewEventObservable<MainViewEvent> {

    private val disposables = CompositeDisposable()

    override fun onResume() {
        super.onResume()
        disposables += viewEvents().subscribe( ? )
    }

    override fun onPause() {
        super.onPause()
        disposables.clear()
    }

    override fun viewEvents(): Observable<MainViewEvent> {
        return Observable.merge(
                heartButton.clicks().map { LoveItClick },
                thumbButton.clicks().map { ThumbsUpClick }
        )
    }

}
```

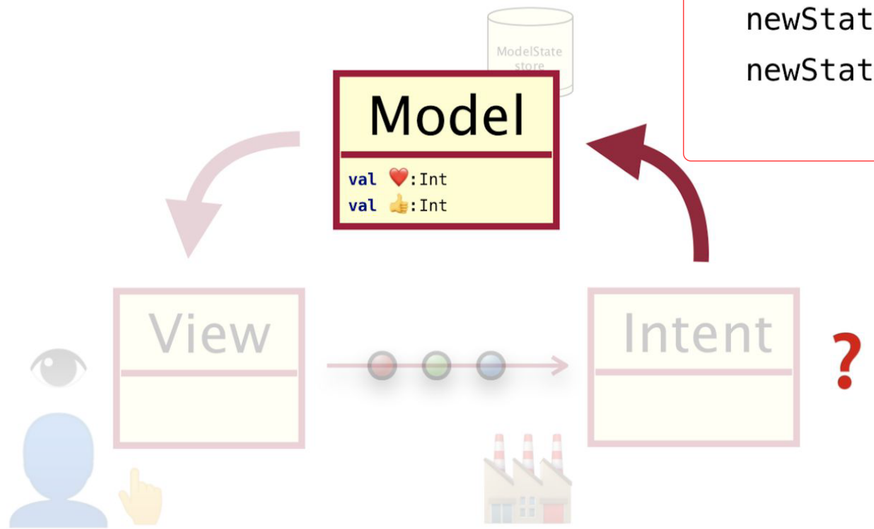



```
class MainActivity : ViewEventObservable<MainViewEvent> {

    private val disposables = CompositeDisposable()

    override fun onResume() {
        super.onResume()
        disposables += viewEvents().subscribe(
            MainViewIntentFactory::process
        )
    }

    override fun onPause() {
        super.onPause()
        disposables.clear()
    }

    override fun viewEvents(): Observable<MainViewEvent> {
        return Observable.merge(
                heartButton.clicks().map { LoveItClick },
                thumbButton.clicks().map { ThumbsUpClick }
        )
    }

}
```

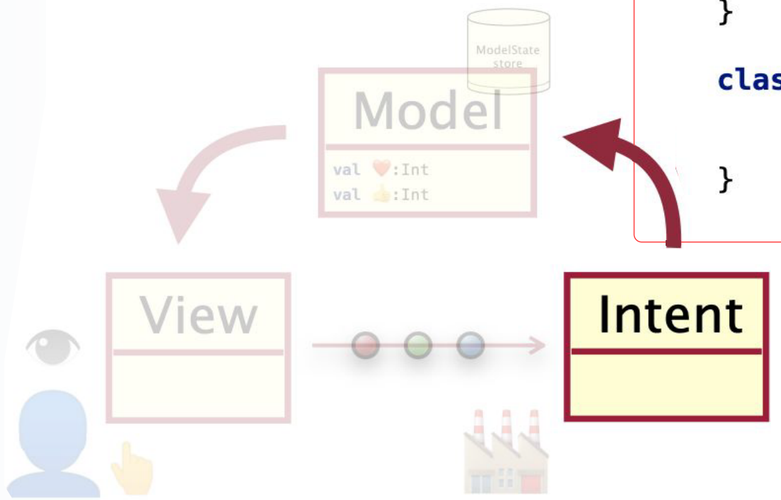


MVI ARCHITECTURE



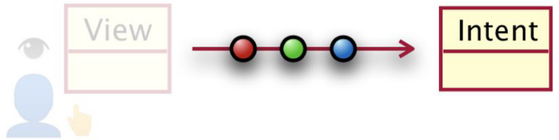



```
data class UpvoteModel(val hearts:Int, val thumbs:Int)

newState = oldState.copy(thumbs = thumbs + 1) // 0, 1
newState = newState.copy(hearts = hearts + 1) // 1, 1
newState = newState.copy(hearts = hearts + 1) // 2, 1
```

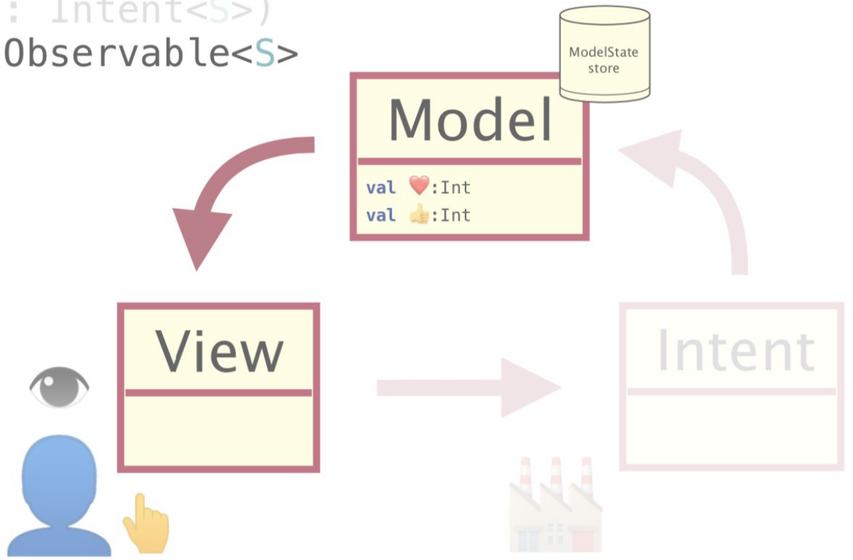


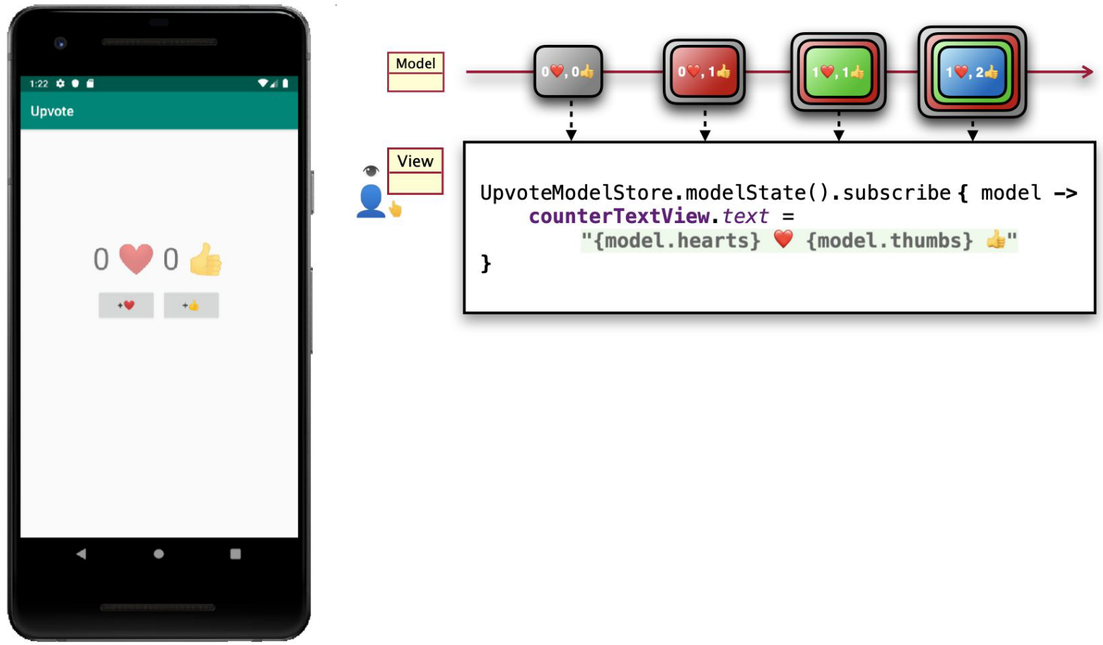

```
data class UpvoteModel(val hearts:Int, val thumbs:Int)

interface Intent<T> {
    fun reduce(oldState: T): T
}

class AddHeart():Intent<UpvoteModel> {
    override fun reduce(oldState: UpvoteModel) =
        oldState.copy(hearts = oldState.hearts + 1)
}
```

```
interface IntentFactory<E> {
    fun process(viewEvent:E)
}

object MainViewIntentFactory : IntentFactory<MainViewEvent> {

    override fun process(viewEvent: MainViewEvent) {
        UpvoteModelStore.process(toIntent(viewEvent))
    }

    private fun toIntent(viewEvent: MainViewEvent):Intent<UpvoteModel> {
        return when (viewEvent) {
            MainViewEvent.LoveItClick -> AddHeart()
            MainViewEvent.ThumbsUpClick -> AddThumb()
        }
    }

}
```



```
object UpvoteModelStore

interface ModelStore<S> {
    fun process(intent: Intent<S>)
    fun modelState(): Observable<S>
}
```

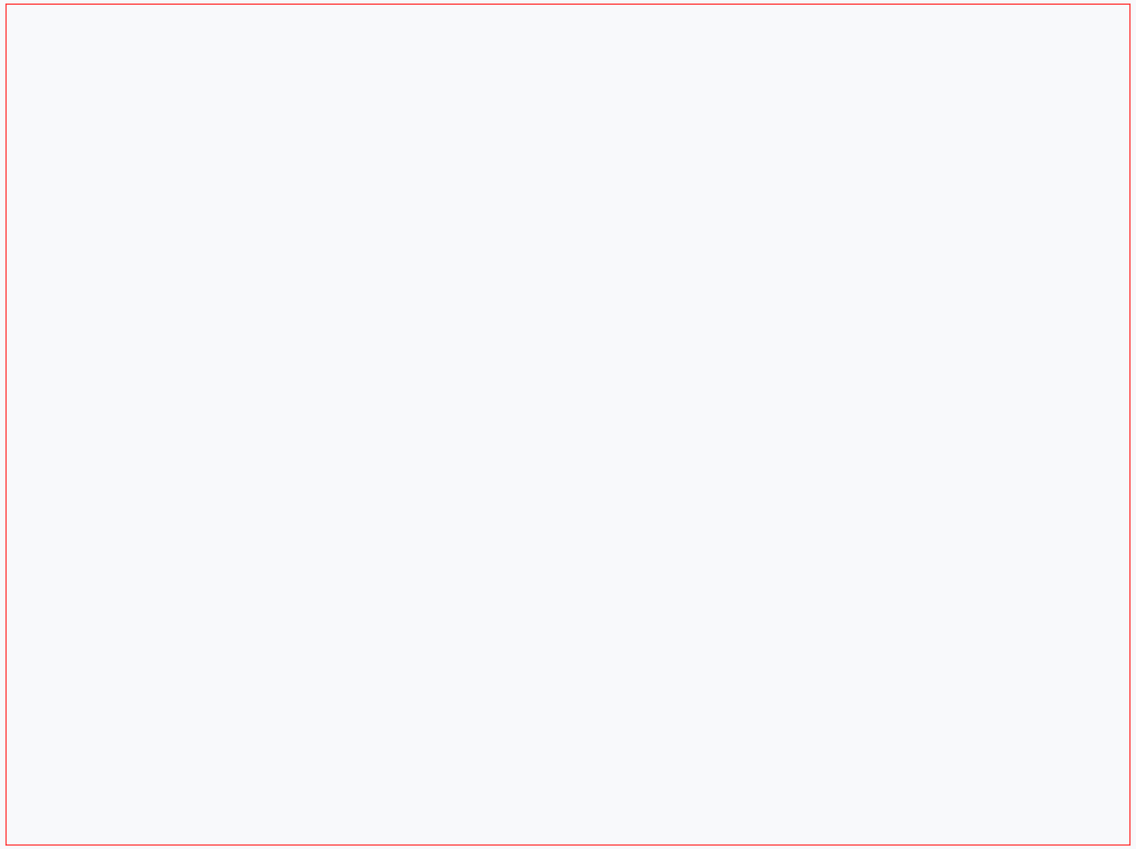



1:22
Upvote
0 ❤️ 0 👍
+❤️
+👍
Model
0❤️, 0👍
0❤️, 1👍
1❤️, 1👍
1❤️, 2👍
View
UpvoteModelStore.modelState().subscribe { model ->
counterTextView.text =
"{model.hearts} ❤️ {model.thumbs} 👍"
}

LIVE DEMO

QUESTIONS & ANSWERS

THE ANSWER IS ALWAYS



"IT DEPENDS"

THANK YOU

Rappi

WE ARE HIRING

https://jobs.lever.co/rappi/