JETPACK COMPOSE: FIRST BLOOD

ABOUT ME

ABOUT ME

- VLADIMIR IVANOV

ABOUT ME

- VLADIMIR IVANOV
- DESIGNING MOBILE-CENTRIC SOLUTIONS FOR LIVING

ABOUT ME

- VLADIMIR IVANOV
- DESIGNING MOBILE-CENTRIC SOLUTIONS FOR LIVING
- PRIMARY SKILL: ANDROID

ABOUT ME

- VLADIMIR IVANOV
- DESIGNING MOBILE-CENTRIC SOLUTIONS FOR LIVING
- PRIMARY SKILL: ANDROID
- CERTIFIED GOOGLE CLOUD ARCHITECT.

WHAT WE USUALLY DO?

IMPERATIVE PROGRAMMING

```
override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)
    emergency_call_layout.setOnClickListener { onEmergencyCallPressed() }

    filter_recycler_view.layoutManager =
        LinearLayoutManager(context, LinearLayoutManager.HORIZONTAL, false)
    val filterAdapter =
        FilterAdapter(arrayListOf("One", "Two", "Three"), null, this::onFilterSelected)
    filter_recycler_view.adapter = filterAdapter

    crisis_centers_list.layoutManager = LinearLayoutManager(context)
    viewModel.crisisCenters.observe(this, crisisCenterObserver)
}
```

WHAT WE USUALLY DO?

WHAT WE USUALLY DO?

- XML

WHAT WE USUALLY DO?

- XML
- FRAGMENT

WHAT WE USUALLY DO?

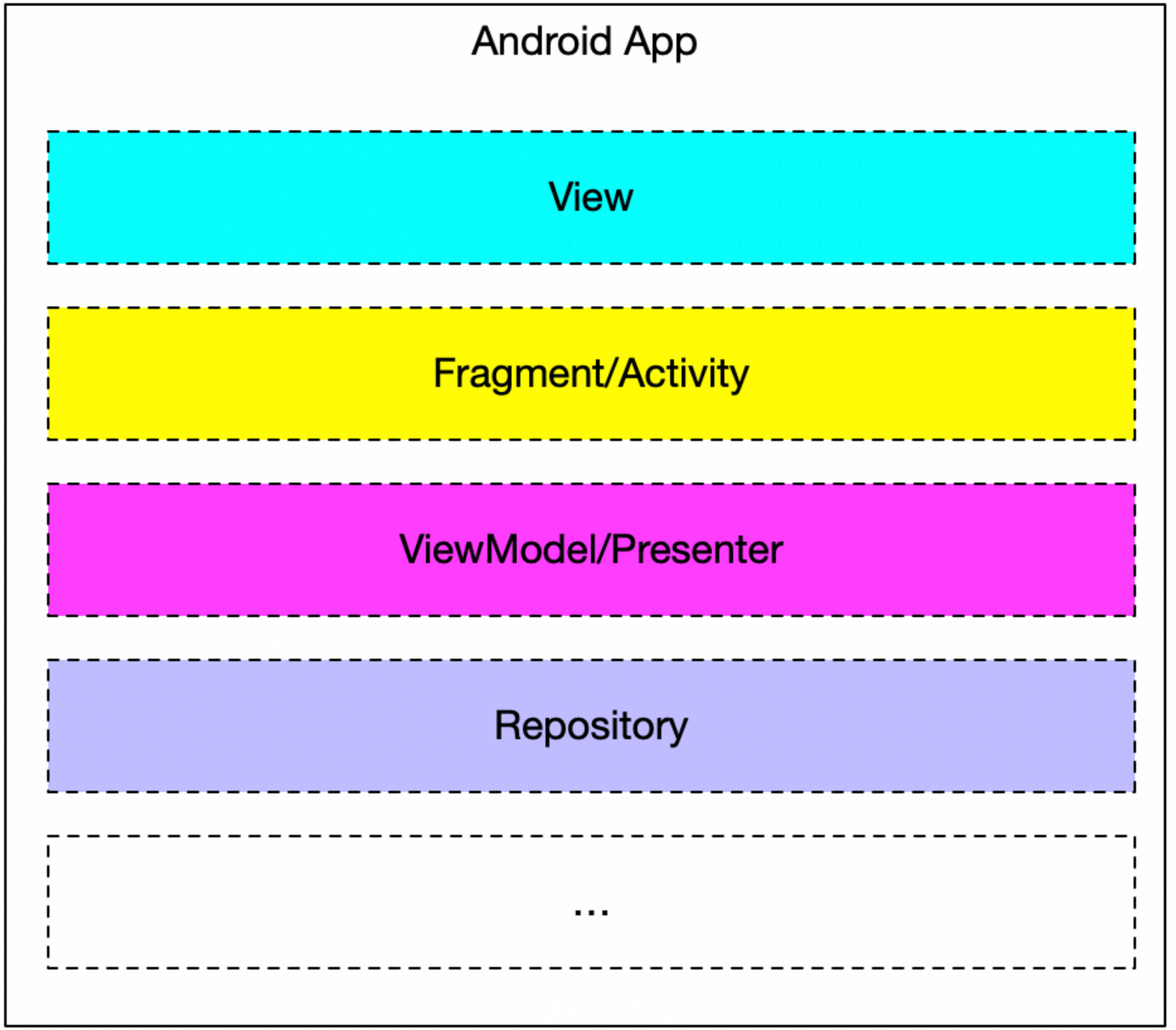
- XML
- FRAGMENT
- PRESENTER/VIEWMODEL

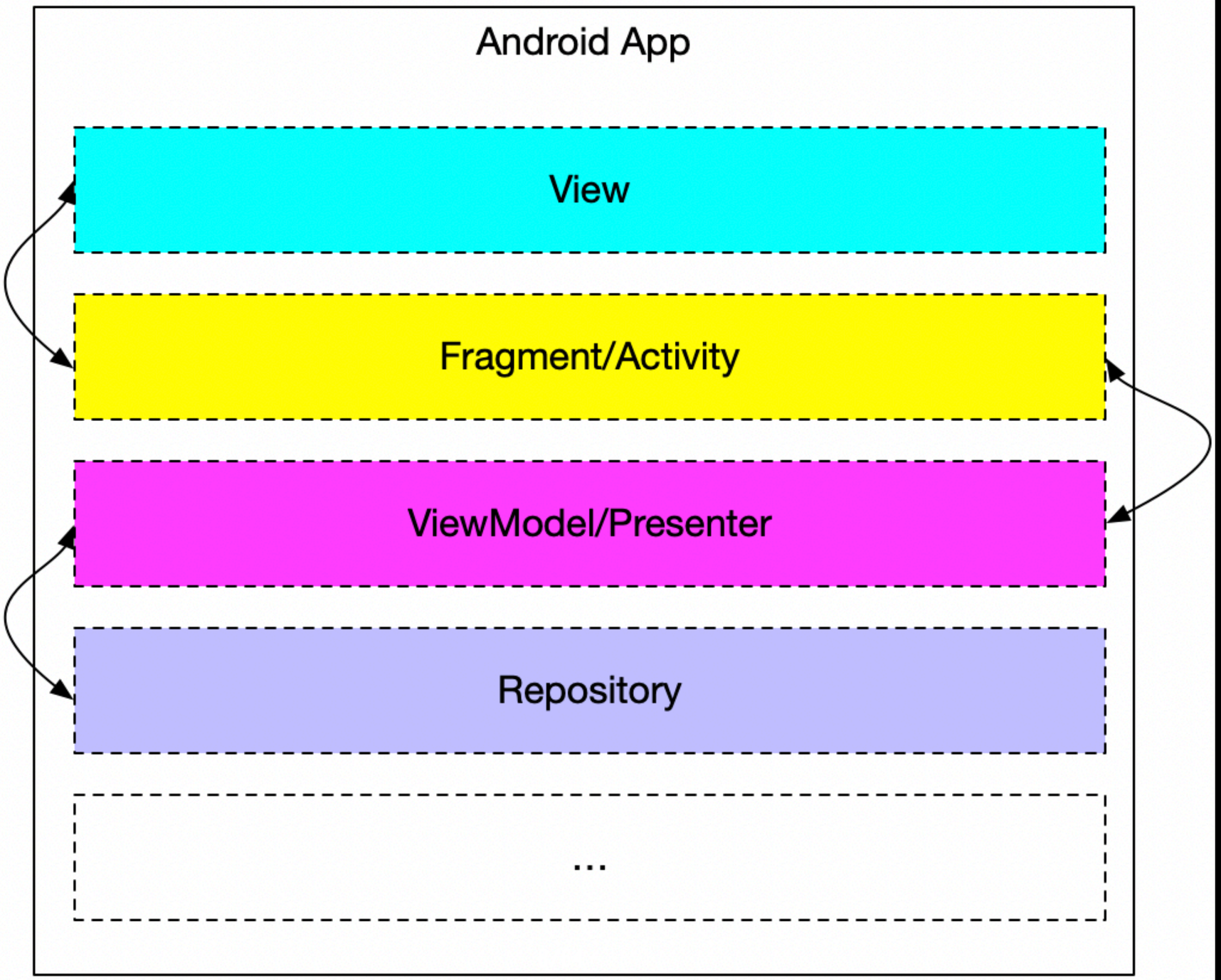
WHAT WE USUALLY DO?

- XML
- FRAGMENT
- PRESENTER/VIEWMODEL
- STYLES

WHAT WE USUALLY DO?

- XML
- FRAGMENT
- PRESENTER/VIEWMODEL
- STYLES
- ATTRIBUTES


Android App
View
Fragment/Activity
ViewModel/Presenter
Repository
...


Android App
View
Fragment/Activity
ViewModel/Presenter
Repository
...

MINUSES

MINUSES

- SEVERAL PLACES TO RIGHT CODE

MINUSES

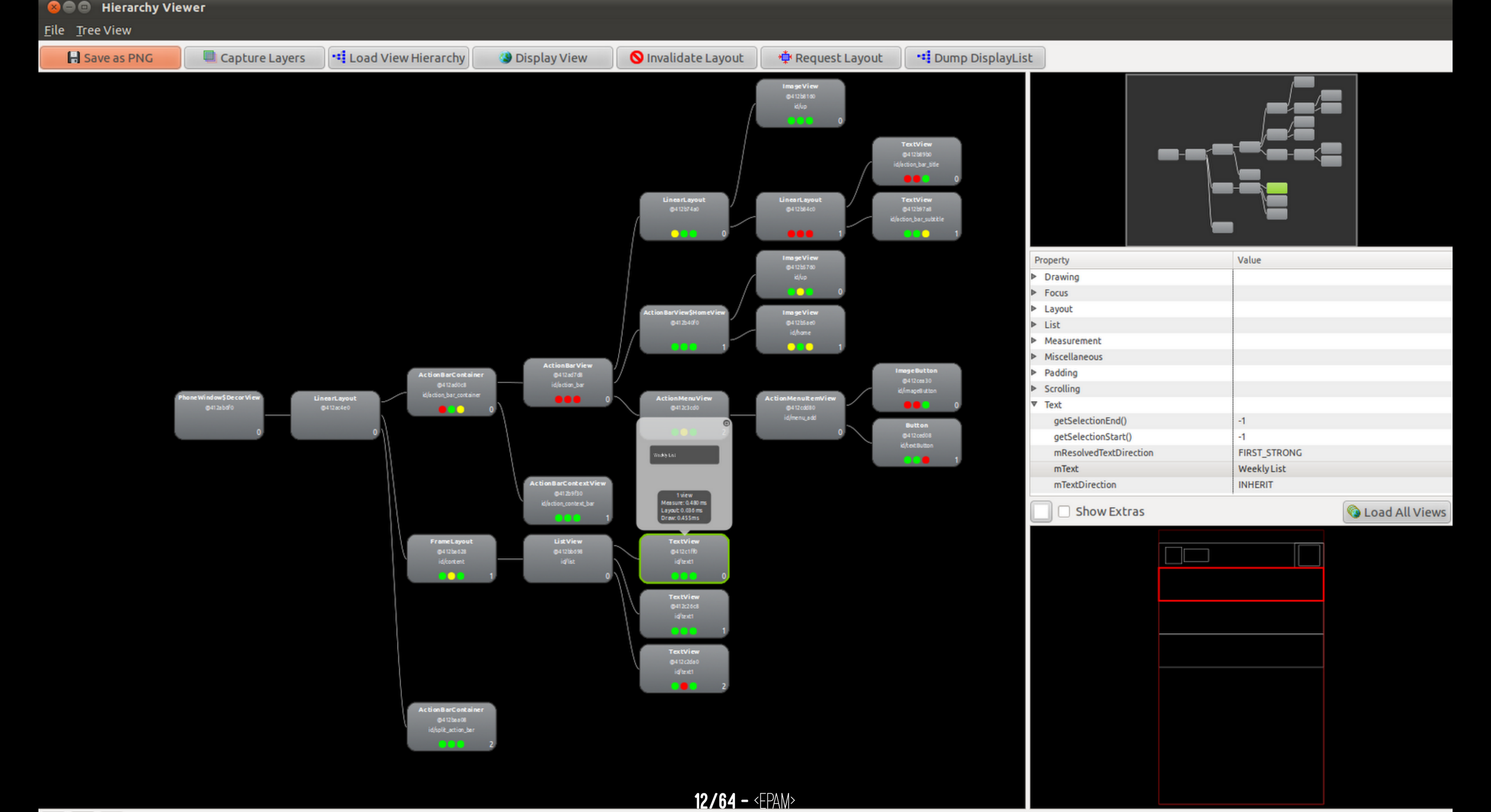
- SEVERAL PLACES TO RIGHT CODE
- HARD TO DEBUG

MINUSES

- SEVERAL PLACES TO RIGHT CODE
- HARD TO DEBUG
- IMPOSSIBLE TO REUSE

LET'S RETHINK THE WHOLE APPROACH

UI - IS A TREE-LIKE DATA STRUCTURE THAT CHANGES OVERTIME


Hierarchy Viewer
File
Tree View
Save as PNG
Capture Layers
Load View Hierarchy
Display View
Invalidate Layout
Request Layout
Dump DisplayList
PhoneWindow$DecorView
LinearLayout
ActionBarContainer
ActionBarView
ImageView
LinearLayout
TextView
TextView
ImageView
ActionBarView$HomeView
ImageView
ActionMenuView
ActionMenuItemView
ImageButton
Button
ActionBarContextView
FrameLayout
ListView
TextView
TextView
TextView
ActionBarContainer
Property
Value
Drawing
Focus
Layout
List
Measurement
Miscellaneous
Padding
Scrolling
Text
getSelectionEnd()
-1
getSelectionStart()
-1
mResolvedTextDirection
FIRST_STRONG
mText
Weekly List
mTextDirection
INHERIT
Show Extras
Load All Views
Filter by class or id:
20%
200%

DECLARATIVE APPROACH

DECLARATIVE APPROACH

- MAKE UI A PURE FUNCTION OF STATE

PROS

PROS

- ALMOST SINGLE PLACE TO WRITE CODE

PROS

- ALMOST SINGLE PLACE TO WRITE CODE
- EASY TO TEST

PROS

- ALMOST SINGLE PLACE TO WRITE CODE
- EASY TO TEST
- EASY TO SCALE

PROS

- ALMOST SINGLE PLACE TO WRITE CODE
- EASY TO TEST
- EASY TO SCALE
- UNBUNDLED IMPLEMENTATION


YOU KNOW WHAT?
THE APPROACH WON!

> REACT

- REACT
- ANGULAR

- REACT
- ANGULAR
- VUE.JS

- REACT
- ANGULAR
- VUE.JS
- SWIFTUI

- REACT
- ANGULAR
- VUE.JS
- SWIFTUI
- AND NOW: JETPACK COMPOSE

JETPACK COMPOSE

JETPACK COMPOSE

- INSPIRED BY REACT AND FLUTTER

JETPACK COMPOSE

- INSPIRED BY REACT AND FLUTTER
- PRE-PRE-ALPHA

JETPACK COMPOSE

- INSPIRED BY REACT AND FLUTTER
- PRE-PRE-ALPHA
- PART OF AOSP

JETPACK COMPOSE

GENERALLY SPEAKING

GENERALLY SPEAKING

- NODE FOR PIECE OF UI, HAVING CHILDREN

GENERALLY SPEAKING

- NODE FOR PIECE OF UI, HAVING CHILDREN
  - FUNCTION TO RENDER A ROOT NODE

GENERALLY SPEAKING

- NODE FOR PIECE OF UI. HAVING CHILDREN
  - FUNCTION TO RENDER A ROOT NODE
- FUNCTIONS TO RENDER PRIMITIVES: TEXT. IMAGE. ETC.

THEN...

THEN...

- WE NEED TO PASS ROOT NODE DOWN THE BRANCHES

THEN...

- WE NEED TO PASS ROOT NODE DOWN THE BRANCHES
  - WHICH CAN BE KINDA HARSH

SO...

```
interface Composer {
  // add node as a child to the current Node, execute
  // `content` with `node` as the current Node
  fun emit(node: Node, content: () -> Unit = {})
}
```

THUS

THUS

- `ComposerImpl`

THUS

- `ComposerImpl`
- **TOP-LEVEL** `compose` **FUNCTION**

THUS

- `ComposerImpl`
- TOP-LEVEL `compose` FUNCTION
- CALL TOP LEVEL RENDER FUNCTION WITH COMPOSE INVOCATION

ALL THIS CAN BE HIDDEN WITH THE HELP OF COMPILER AND RUNTIME
WHICH JETPACK COMPOSE HAPPENS TO BE

[1] HTTPS://DEVELOPER.ANDROID.COM/JETPACK/COMPOSE

[2] HTTPS://WWW.RAYWENDERLICH.COM/3604589-JETPACK-COMPOSE-PRIMER

HOW TO START: D.ANDROID.COM/JETPACK/COMPOSE[1]

[1] HTTPS://DEVELOPER.ANDROID.COM/JETPACK/COMPOSE

[2] HTTPS://WWW.RAYWENDERLICH.COM/3604589-JETPACK-COMPOSE-PRIMER

- HOW TO START: D.ANDROID.COM/JETPACK/COMPOSE[1]
- HOW TO START PROPERLY: PRIMER BY RAYWENDERLICH.COM[2]

[1] HTTPS://DEVELOPER.ANDROID.COM/JETPACK/COMPOSE

[2] HTTPS://WWW.RAYWENDERLICH.COM/3604589-JETPACK-COMPOSE-PRIMER

START WITH

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContent {
        CraneWrapper {
            MaterialTheme {
                EditAndSlider()
            }
        }
    }
}
```

> LOOKS LIKE DSL

- LOOKS LIKE DSL
- LOOKS LIKE A TREE

- LOOKS LIKE DSL
- LOOKS LIKE A TREE
- EVERY RECORD IS A COMPONENT

COMPONENT IS A FUNCTION (...) -> UNIT

CRANEWRAPPER - ROOT COMPOSABLE

MATERIALTHEME - SOME STYLING DEFAULTS

SELECTIONSCONTROLSDEMO - INVOCATION OF OUR FUNCTION

```
@Composable
fun EditAndSlider() {
    Text(text = "Enter your login")
}
```

COMPOSABLE COMPONENT

COMPOSABLE COMPONENT

- RETURNS ANOTHER FUNCTION INVOCATION

COMPOSABLE COMPONENT

- RETURNS ANOTHER FUNCTION INVOCATION
  - HAS @COMPOSABLE

COMPOSABLE COMPONENT

- RETURNS ANOTHER FUNCTION INVOCATION
  - HAS @COMPOSABLE
  - BASICALLY YIELDS PIECE OF UI

WHY COMPILER IS INVOLVED?

WHY COMPILER IS INVOLVED?

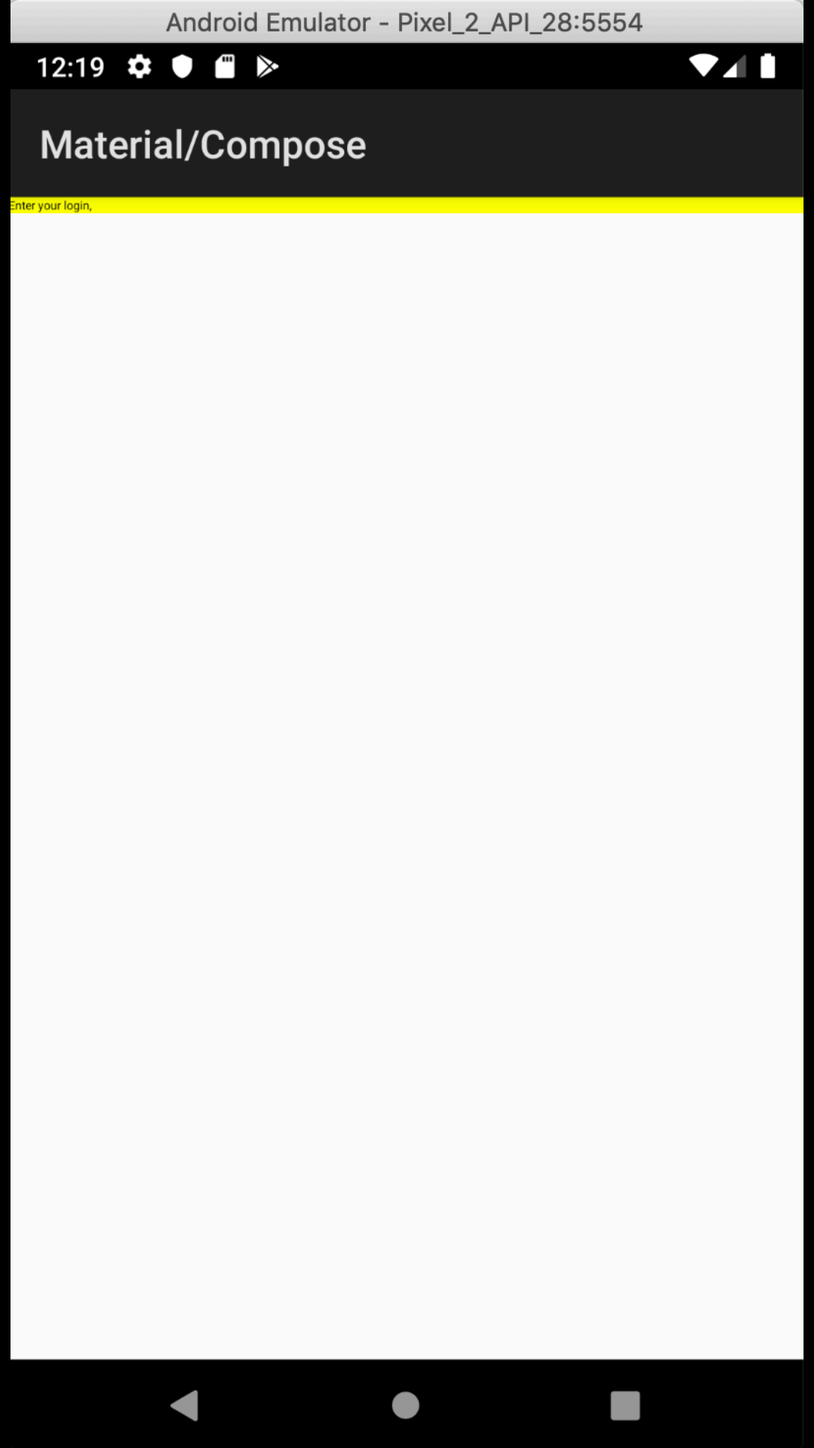
- EVERY @COMPOSABLE SHOULD ACCEPT A COMPOSER AS A LAST ARGUMENT

WHY COMPILER IS INVOLVED?

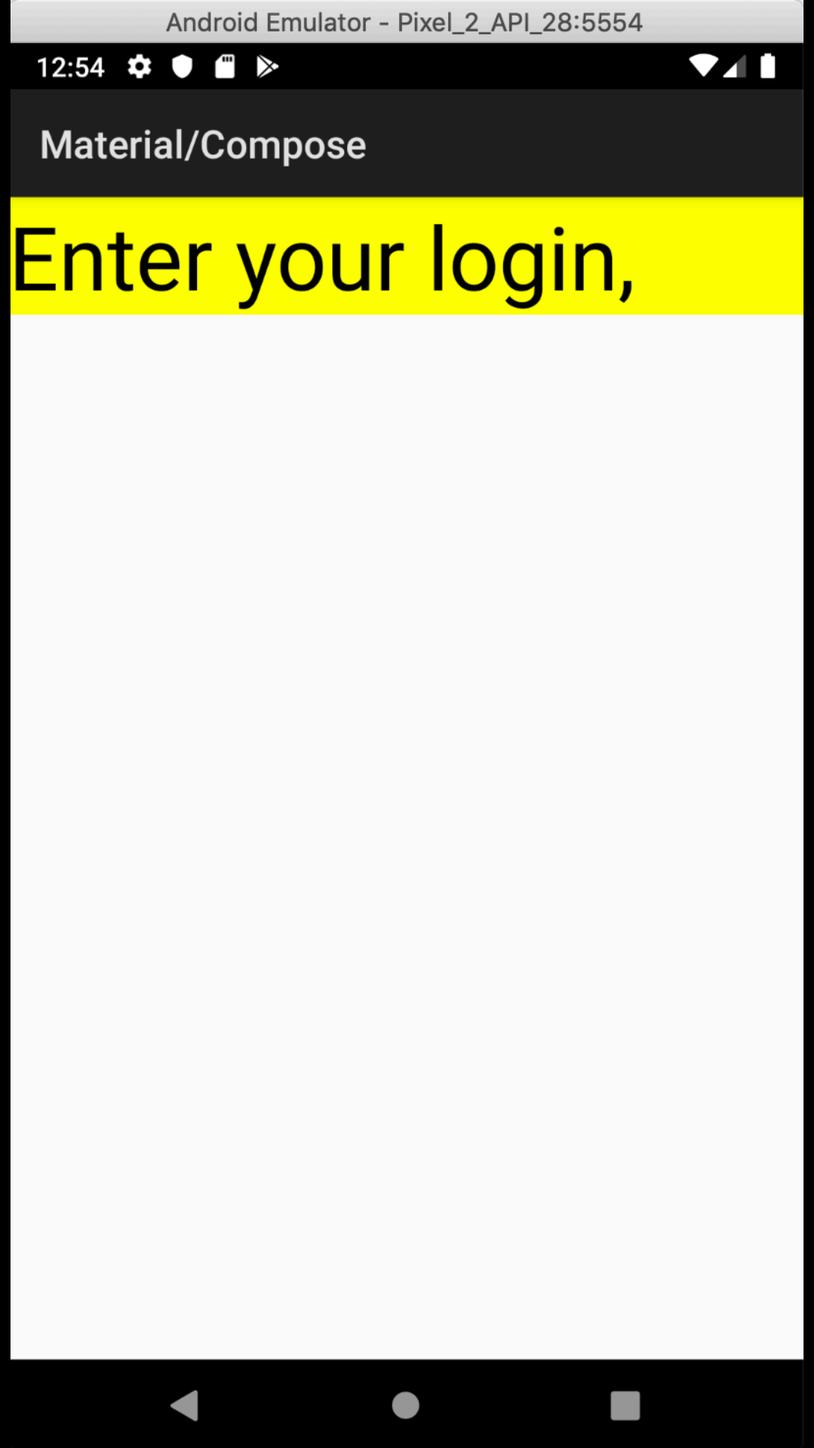
- EVERY @COMPOSABLE SHOULD ACCEPT A COMPOSER AS A LAST ARGUMENT
- THINK OF @COMPOSABLE AS A KEYWORD(LIKE SUSPEND)

```
@Composable
fun EditAndSlider() {
    Surface(color = Color(0xFFFFFF00.toInt())) {
        Text(text = "Enter your login, ")
    }
}
```

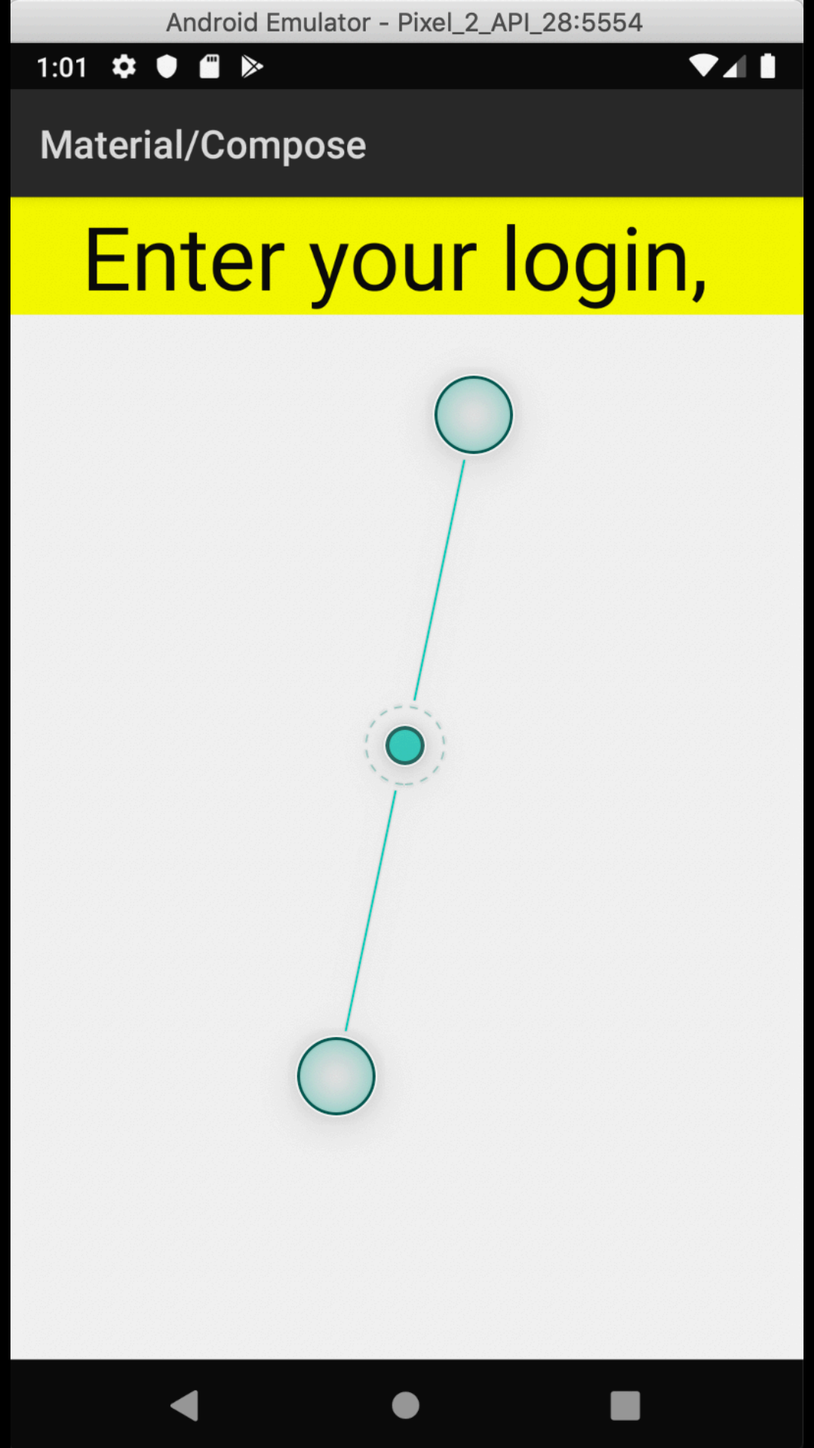
```
@Composable
fun EditAndSlider() {
    Surface(color =
    Color(0xFFFFFF00.toInt())) {
        Text(text = "Enter your login, ")
    }
}
```



```
Text(
    text = "Enter your login, ",
    style = TextStyle(fontSize = 120f)
)
```

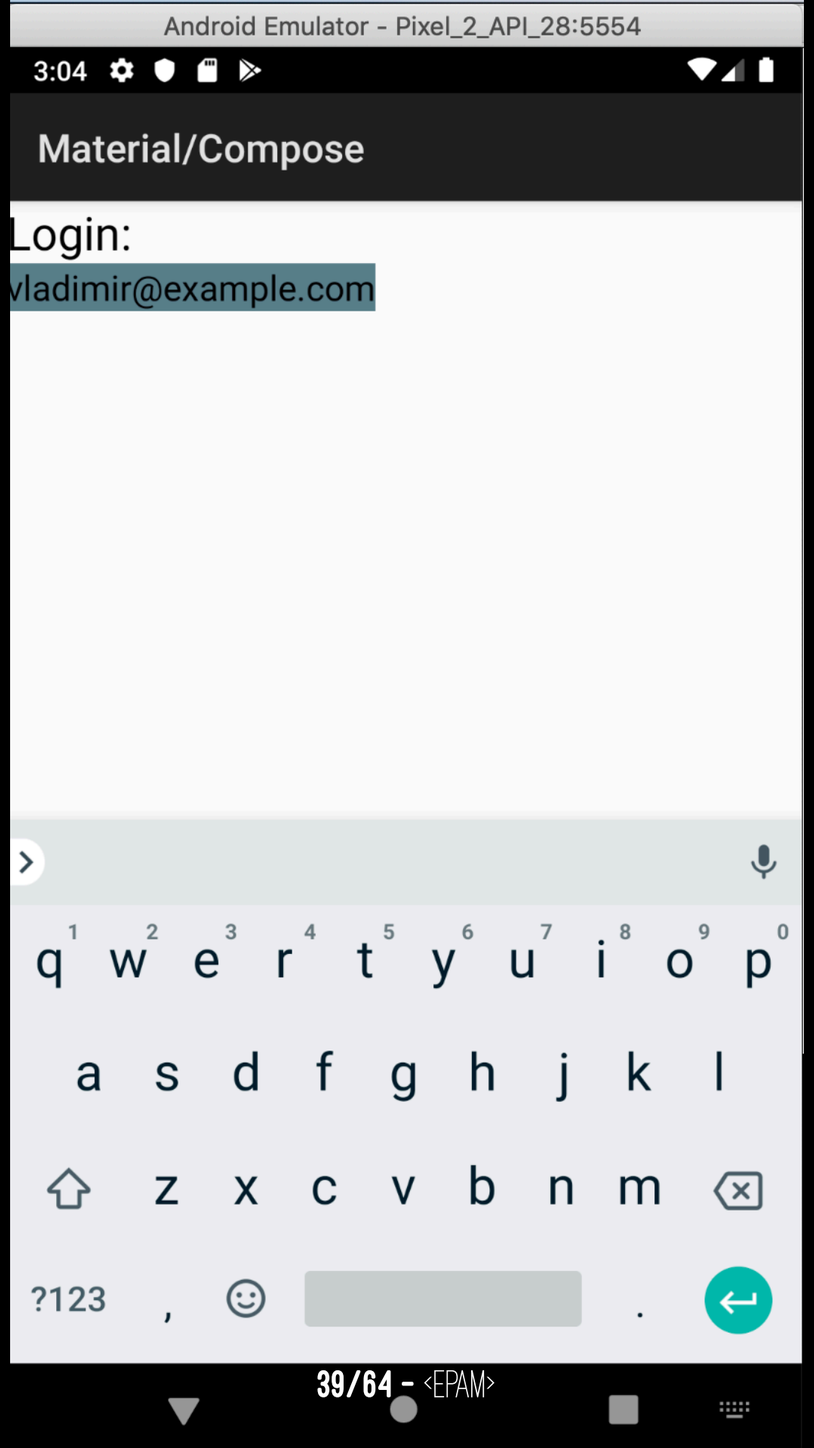


```
Text(
  text = "Enter your login, ",
  style = TextStyle(fontSize = 120f),
  textAlign = TextAlign.Center
)
```



```
val progressState = +state { EditorState(text = "vladimir@example.com") }

EditableText(
    value = progressState.value,
    editorStyle = EditorStyle(TextStyle(fontSize = 48.toFloat())),
    onValueChange = { progressState.value = it }
)
```

```
val progressState = +state { EditorState(text = "vladimir@example.com") }

EditableText(
    value = progressState.value,
    editorStyle = EditorStyle(TextStyle(fontSize = 48.toFloat())),
    onValueChange = { progressState.value = it }
)
```


Android Emulator - Pixel_2_API_28:5554
3:04
Material/Compose
Login:
vladimir@example.com
q w e r t y u i o p
a s d f g h j k l
z x c v b n m
?123

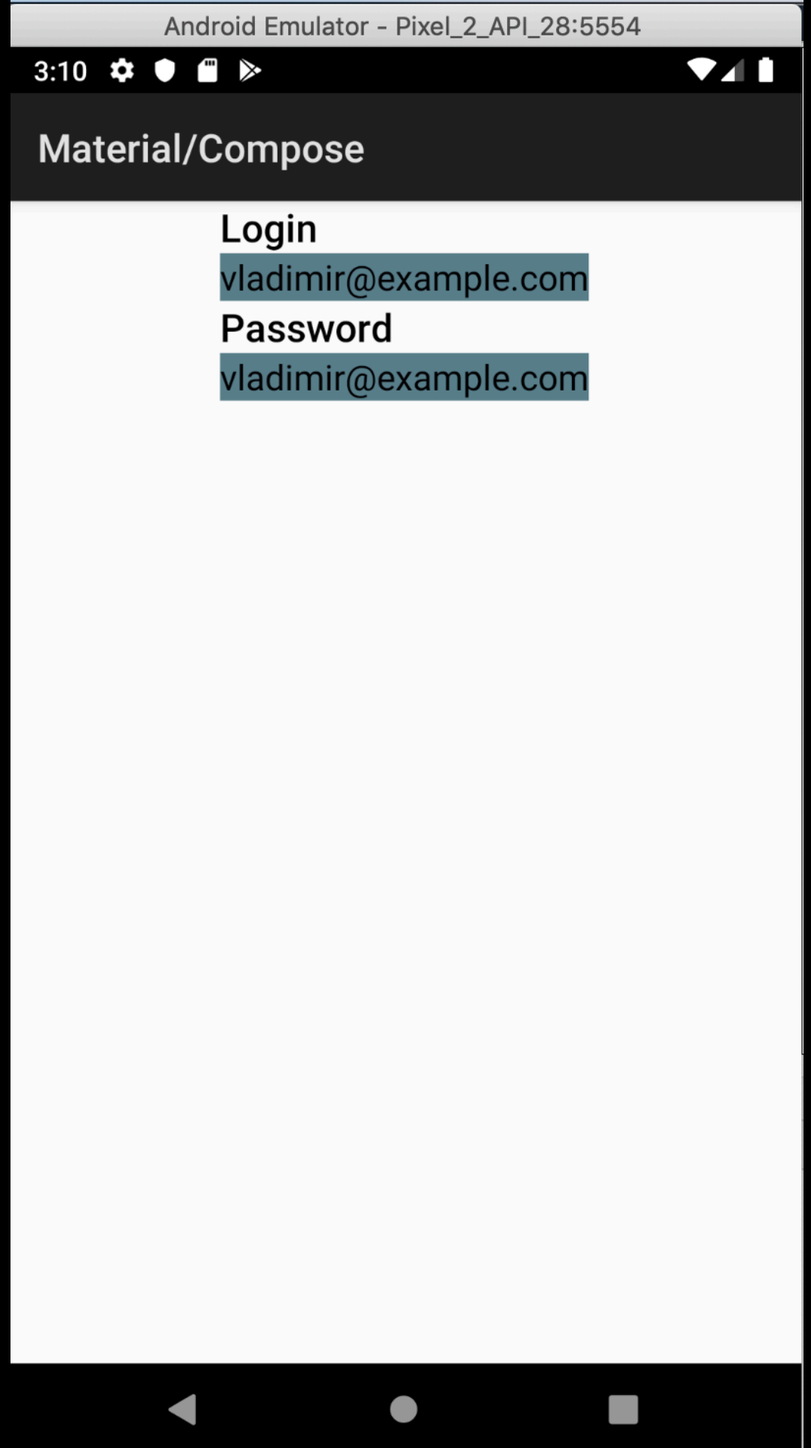
REUSABILITY

```
@Composable
fun InputWithCaption(caption: String) {

    val labelStyle = +themeTextStyle { h6 }
    val progressState = +state { EditorState(text = "vladimir@example.com") }

    Column(crossAxisAlignment = CrossAxisAlignment.Start) {
        Text(caption, style = labelStyle)
        Surface(color = customColor3) {
            EditableText(
                    value = progressState.value,
                    editorStyle = EditorStyle(TextStyle(fontSize = 48.toFloat())),
                    onValueChange = { progressState.value = it }
            )
        }
    }
}
```

```
@Composable
fun InputWithCaption(caption: String) {

    val labelStyle = +themeTextStyle { h6 }
    val progressState = +state { EditorState(text = "vladimir@example.com") }

    Column(crossAxisAlignment = CrossAxisAlignment.Start) {
        Text(caption, style = labelStyle)
        Surface(color = customColor3) {
            EditableText(
                    value = progressState.value,
                    editorStyle = EditorStyle(TextStyle(fontSize = 48.toFloat())),
                    onValueChange = { progressState.value = it }
            )
        }
    }
}
```

```
@Composable
fun InputWithCaption(caption: String) {

    val labelStyle = +themeTextStyle { h6 }
    val progressState = +state { EditorState(text = "vladimir@example.com") }

    Column(crossAxisAlignment = CrossAxisAlignment.Start) {
        Text(caption, style = labelStyle)
        Surface(color = customColor3) {
            EditableText(
                    value = progressState.value,
                    editorStyle = EditorStyle(TextStyle(fontSize = 48.toFloat())),
                    onValueChange = { progressState.value = it }
            )
        }
    }
}
```

```kotlin
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContent {
        CraneWrapper {
            MaterialTheme {
                Column {
                    InputWithCaption("Login")
                    InputWithCaption("Password")
                }
            }
        }
    }
}
```



APPLY KOTLIN CONSTRUCTIONS

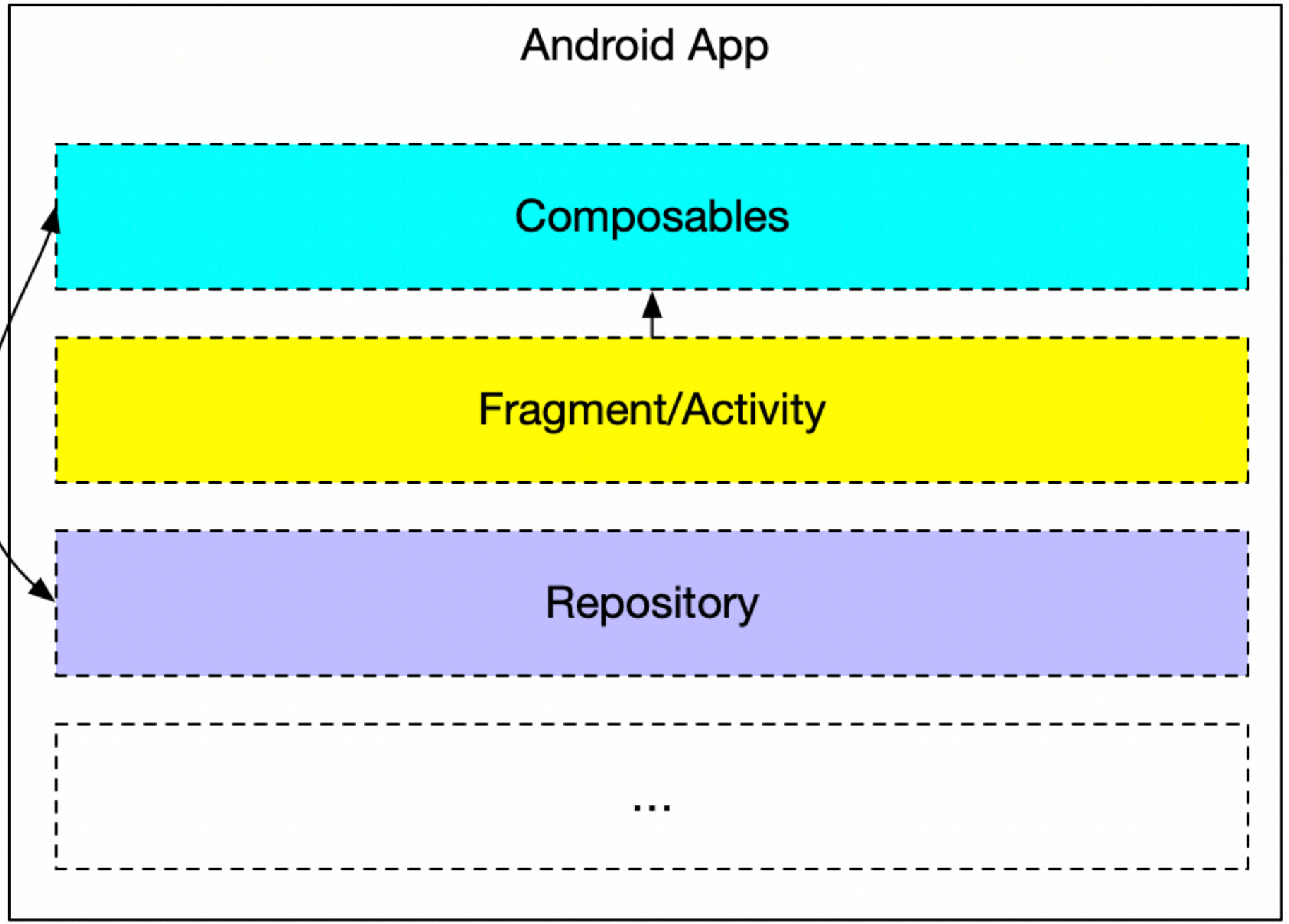
```
setContent {
    CraneWrapper {
        MaterialTheme {
            Column {
                for (1..10) {
                    InputWithCaption("$it")
                }
            }
        }
    }
}
```

PASS LAMBDAS

```
fun onSubmit() {
  Toast.makeText(this, "Hello!", LENGTH_LONG).show()
}

setContent {
    CraneWrapper {
        MaterialTheme {
            InputWithCaption("Login", onSubmit)
        }
    }
}
```

REUSE STYLES

```
val h6: TextStyle = TextStyle(
    fontFamily = FontFamily("Roboto"),
    fontWeight = FontWeight.w500,
    fontSize = 20f),
```


Android App
Composables
Fragment/Activity
Repository
...

OPENED QUESTIONS

LISTS SUPPORT

LISTS SUPPORT

- RECYCLERVIEW ANALOG IN DEVELOPMENT

UNIT TESTS

```
fun testRendersTheTitleAsPartOfTheRallyAppBarComponent() {
    val title = "Any title"

    val component = RallyApp(title)

    assertEquals(title, component.title.text)
}

fun testRendersTheRallyAppBarComponent() {
    val component = RallyApp(title )

    component.matchWithSnapshot()
}

@Composable
fun RallyAppBar(title: String) {
     Row {
         Text(text = title, style = +themeTextStyle { h4 })
     }
}
```

HOWEVER

IMO THIS IS A BIT OF A JUMP IN LOGIC. THE ASSUMPTION HERE IS THAT ASSERTING ON THE RETURNED "VIRTUAL DOM" OF COMPONENTS IS THE RIGHT AND ONLY WAY TO TEST THOSE COMPONENTS. I DISAGREE ON BOTH COUNTS. WITH THE RIGHT APIS EXPOSED, COMPOSE WILL BE AS TESTABLE AS ANYTHING ELSE AND WE WILL PUSH YOU INTO THE "PIT OF SUCCESS" OF TESTING THE PUBLIC API OF COMPONENTS INSTEAD OF THEIR IMPLEMENTATION DETAILS.

-- LELAND RICHARDSON

STATE MANAGEMENT

STATE MANAGEMENT

- REDUX IS FAR IN THE FUTURE

MULTITHREADING

MULTITHREADING

- WOULD BE AWESOME TO ACCEPT SUSPEND FUNCTIONS. BUT CURRENT KOTLIN-COMPOSE COMPILER PLUGIN DOESN'T SUPPORT IT

MULTITHREADING

- WOULD BE AWESOME TO ACCEPT SUSPEND FUNCTIONS. BUT CURRENT KOTLIN-COMPOSE COMPILER PLUGIN DOESN'T SUPPORT IT
- YOU'RE POSSIBLE TO CHANGE THE +STATE OBJECT FROM ANY THREAD SAFELY

MULTITHREADING

- WOULD BE AWESOME TO ACCEPT SUSPEND FUNCTIONS, BUT CURRENT KOTLIN-COMPOSE COMPILER PLUGIN DOESN'T SUPPORT IT
- YOU'RE POSSIBLE TO CHANGE THE +STATE OBJECT FROM ANY THREAD SAFELY
  - IF YOU HAVE A FRAME OBJECT IN CURRENT THREAD

MULTITHREADING

- WOULD BE AWESOME TO ACCEPT SUSPEND FUNCTIONS. BUT CURRENT KOTLIN-COMPOSE COMPILER PLUGIN DOESN'T SUPPORT IT
- YOU'RE POSSIBLE TO CHANGE THE +STATE OBJECT FROM ANY THREAD SAFELY
  - IF YOU HAVE A FRAME OBJECT IN CURRENT THREAD
    - FRAME EXISTS IN UI THREAD

MULTITHREADING

- WOULD BE AWESOME TO ACCEPT SUSPEND FUNCTIONS, BUT CURRENT KOTLIN-COMPOSE COMPILER PLUGIN DOESN'T SUPPORT IT
- YOU'RE POSSIBLE TO CHANGE THE +STATE OBJECT FROM ANY THREAD SAFELY
  - IF YOU HAVE A FRAME OBJECT IN CURRENT THREAD
    - FRAME EXISTS IN UI THREAD
- FRAMES ARE IN-MEMORY OBSERVABLE NON-DURABLE TRANSACTIONS

MULTITHREADING

- WOULD BE AWESOME TO ACCEPT SUSPEND FUNCTIONS, BUT CURRENT KOTLIN-COMPOSE COMPILER PLUGIN DOESN'T SUPPORT IT
- YOU'RE POSSIBLE TO CHANGE THE +STATE OBJECT FROM ANY THREAD SAFELY
  - IF YOU HAVE A FRAME OBJECT IN CURRENT THREAD
    - FRAME EXISTS IN UI THREAD
- FRAMES ARE IN-MEMORY OBSERVABLE NON-DURABLE TRANSACTIONS
  - FRAME TO BE RENAMED TO TRANSACTION

MULTITHREADING

- WOULD BE AWESOME TO ACCEPT SUSPEND FUNCTIONS, BUT CURRENT KOTLIN-COMPOSE COMPILER PLUGIN DOESN'T SUPPORT IT
- YOU'RE POSSIBLE TO CHANGE THE +STATE OBJECT FROM ANY THREAD SAFELY
  - IF YOU HAVE A FRAME OBJECT IN CURRENT THREAD
    - FRAME EXISTS IN UI THREAD
- FRAMES ARE IN-MEMORY OBSERVABLE NON-DURABLE TRANSACTIONS
  - FRAME TO BE RENAMED TO TRANSACTION
  - BATCHES ALREADY SUPPORTED

FEEDBACK LOOP

FEEDBACK LOOP

- NO PREVIEW

FEEDBACK LOOP

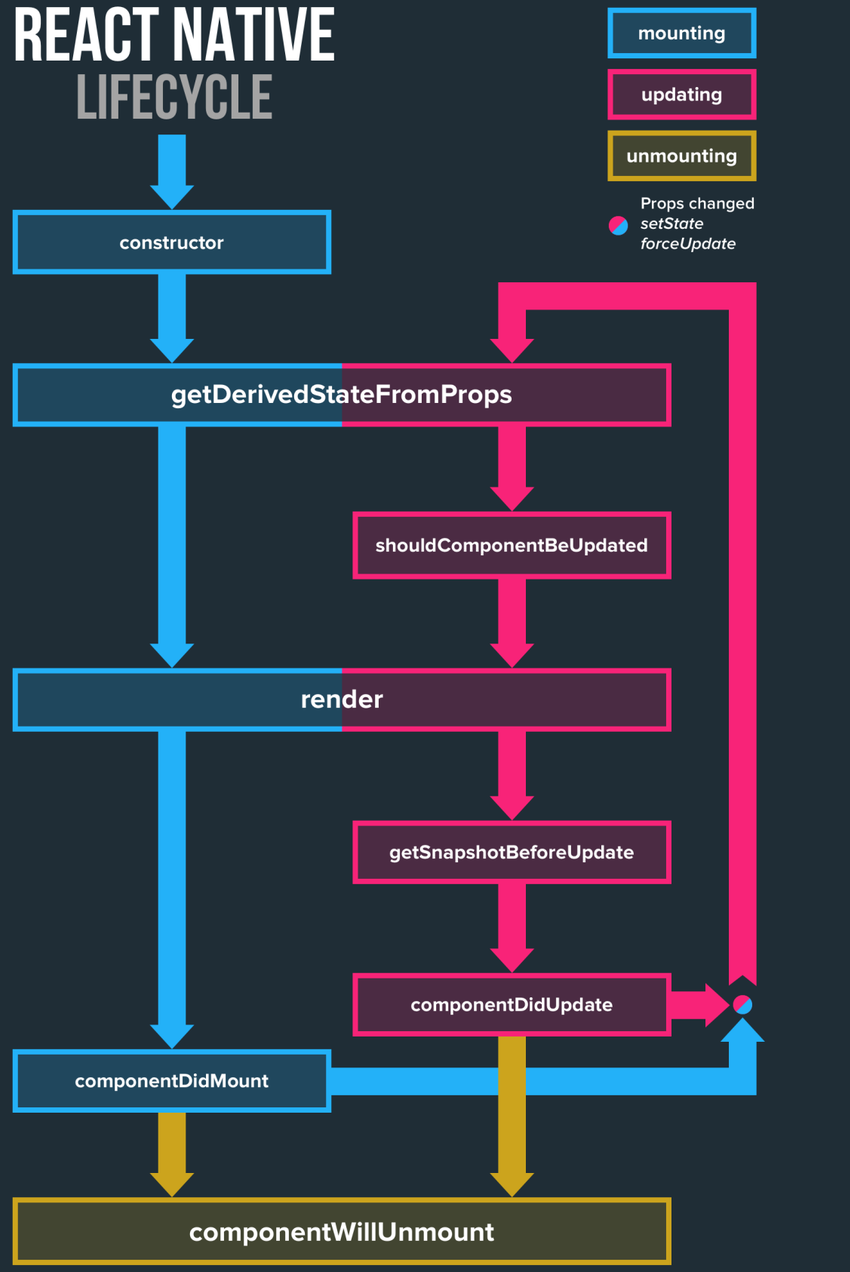
- NO PREVIEW
- HOT RELOADING IS WARM AT THE MOMENT(~30 SECONDS ON MY MACHINE)

FEEDBACK LOOP

- NO PREVIEW
- HOT RELOADING IS WARM AT THE MOMENT(~30 SECONDS ON MY MACHINE)
- FASTER, TENANT HOT RELOADING TO COME

LIFECYCLE

REACT NATIVE LIFECYCLE



ADDRESSED BY CALLBACKS

ADDRESSED BY CALLBACKS

- `onActive`

ADDRESSED BY CALLBACKS

- onActive
- onDispose

ADDRESSED BY CALLBACKS

- onActive
- onDispose
- onCommit

```
@Composable
fun UserProfile(userId: Int) {
    val user = +state<User?>(userId) { null }
    +onCommit(userId) {
        val cancellationToken = UserAPI.get(userId) {
            user.value = it
        }
        onDispose {
            UserAPI.cancel(cancellationToken)
        }
    }
    if (user == null)  {
      Loading()
      return
    }
   Text(text=user.name)
   Image(src=user.photo)
}
```

COMPATIBILITY

COMPATIBILITY

- YOU CAN USE @COMPOSABLE FUNCTIONS AS VIEWS

COMPATIBILITY

- YOU CAN USE @COMPOSABLE FUNCTIONS AS VIEWS
  - YOU CAN USE VIEWS INSIDE @COMPOSABLE

> D.ANDROID.COM/JETPACK/COMPOSE

- D.ANDROID.COM/JETPACK/COMPOSE
  - KOTLINLANG.SLACK.COM

- D.ANDROID.COM/JETPACK/COMPOSE
- KOTLINLANG.SLACK.COM
- HTTPS://BLOG.KARUMI.COM/ANDROID-JETPACK-COMPOSE-REVIEW/


THE END


<EPAM>