Compose Compiler & Runtime:

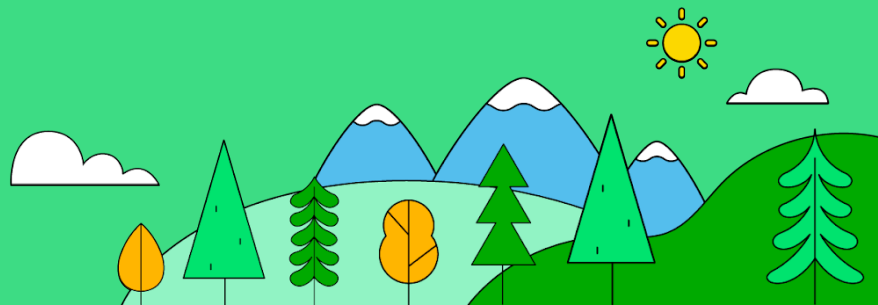
Beyond Android

android

Expositor

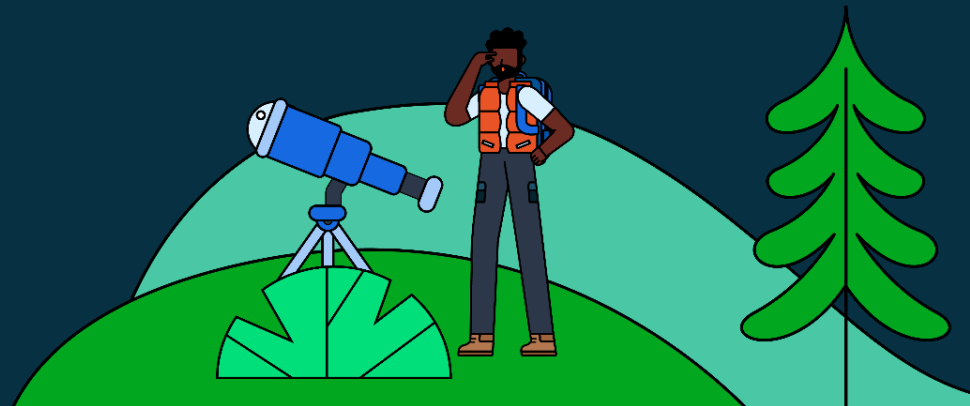
Bruno Aybar
Shopify, Senior Mobile Developer

Twitter: @brunoaybarg
Github: @Bruno125



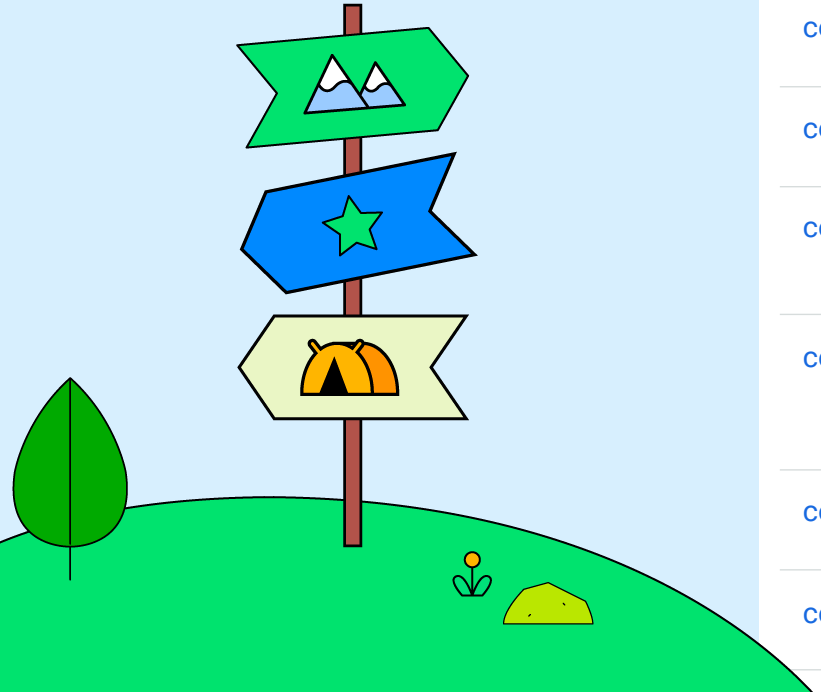
android

```
@Composable
@Preview
fun SurveyAnswer(answer: Answer) {
    Row {
        Image(answer.image)
        Text(answer.text)
        RadioButton(selected = false, onClick = { /* ... */ })
    }
}
```



android

Dependencies



Compose

Compose is combination of 7 Maven Group Ids within `androidx`. Each Group contains a targeted subset of functionality, each with it's own set of release notes.

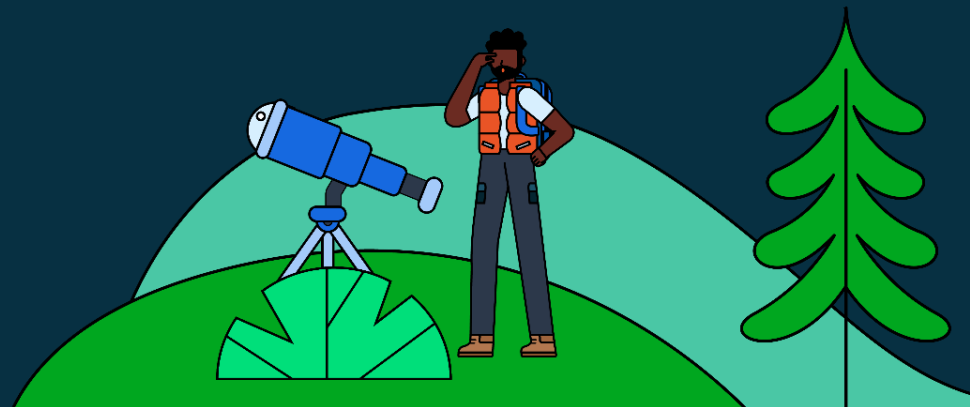
Group	Description
compose.animation	Build animations in their Jetpack Compose applications to enrich the user experience.
compose.compiler	Transform @Composable functions and enable optimizations with a Kotlin compiler plugin.
compose.foundation	Write Jetpack Compose applications with ready to use building blocks and extend foundation to build your own design system pieces.
compose.material	Build Jetpack Compose UIs with ready to use Material Design Components. This is the higher level entry point of Compose, designed to provide components that match those described at www.material.io.
compose.material3	Build Jetpack Compose UIs with Material Design 3 Components, the next evolution of Material Design. Material 3 includes updated theming and components and Material You personalization features like dynamic color, and is designed to be cohesive with the new Android 12 visual style and system UI.
compose.runtime	Fundamental building blocks of Compose's programming model and state management, and core runtime for the Compose Compiler Plugin to target.
compose.ui	Fundamental components of compose UI needed to interact with the device, including layout, drawing, and input.

```
@Composable
@Preview
fun SurveyAnswer(answer: Answer) {
    Row {
        Image(answer.image)
        Text(answer.text)
        RadioButton(selected = false, onClick = { /* ... */ })
    }
}
```

compose.foundation

compose.material
compose.material3

compose.ui



android

Compose

Compose is combination of 7 Maven Group Ids within `androidx`. Each Group contains a targeted subset of functionality, each with it's own set of release notes.

Group	Description
compose.animation	Build animations in their Jetpack Compose applications to enrich the user experience.
compose.compiler	Transform @Composable functions and enable optimizations with a Kotlin compiler plugin.
compose.foundation	Write Jetpack Compose applications with ready to use building blocks and extend foundation to build your own design system pieces.
compose.material	Build Jetpack Compose UIs with ready to use Material Design Components. This is the higher level entry point of Compose, designed to provide components that match those described at www.material.io.
compose.material3	Build Jetpack Compose UIs with Material Design 3 Components, the next evolution of Material Design. Material 3 includes updated theming and components and Material You personalization features like dynamic color, and is designed to be cohesive with the new Android 12 visual style and system UI.
compose.runtime	Fundamental building blocks of Compose's programming model and state management, and core runtime for the Compose Compiler Plugin to target.
compose.ui	Fundamental components of compose UI needed to interact with the device, including layout, drawing, and input.

No es necesario

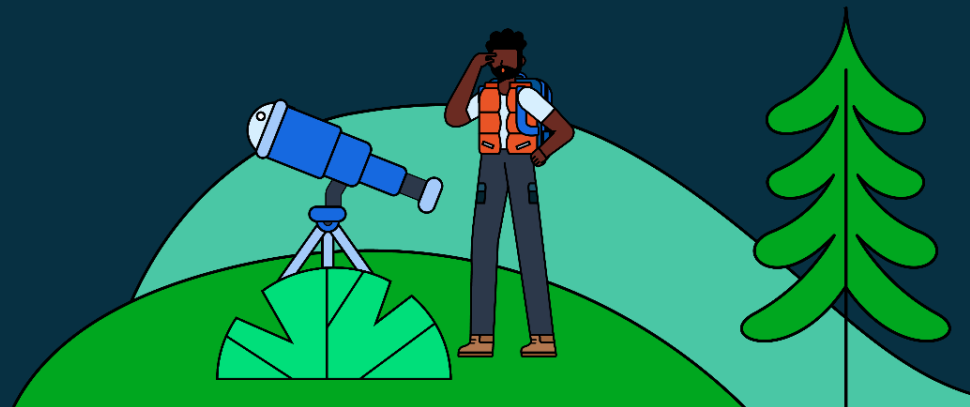
entender a fondo el Compiler / Runtime para usar Compose!

```
@Composable
@Preview
fun SurveyAnswer(answer: Answer) {
    Row {
        Image(answer.image)
        Text(answer.text)
        RadioButton(selected = false, onClick = { /* ... */ })
    }
}
```

compose.foundation

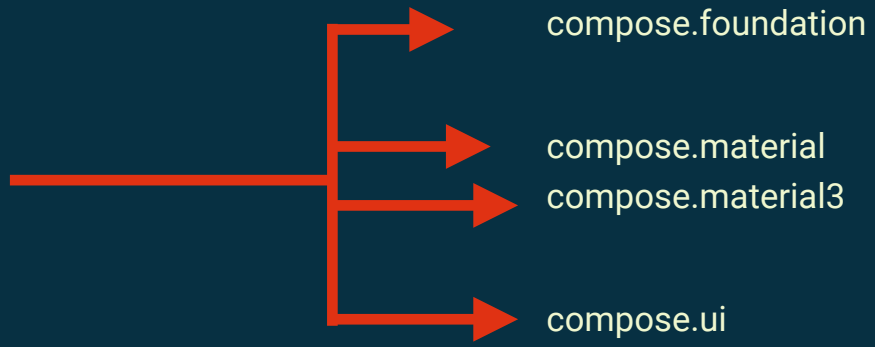
compose.material
compose.material3

compose.ui



android

Entienden / conocen los elementos de UI



android

compose.ui

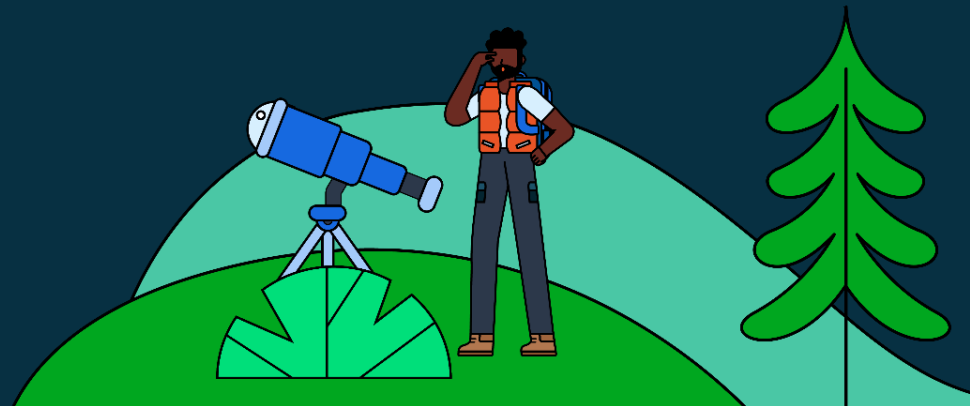
Modifier
Layout
LayoutNode
FocusManager
Alpha / Blur / Scale
...

compose.foundation

Box
Column
Row
Size
Padding
...

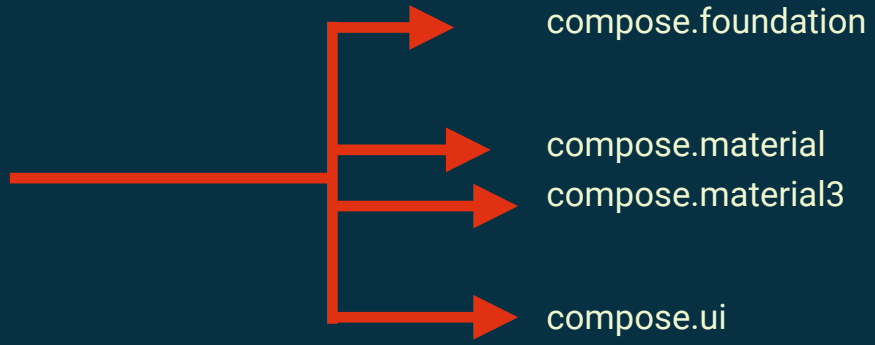
compose.material

Text
Button
Snackbar
Card
Scaffold
...



android

Saben cómo renderizar la UI



¿cuándo?

~~cómo~~

Saben
renderizar la UI

compose.foundation

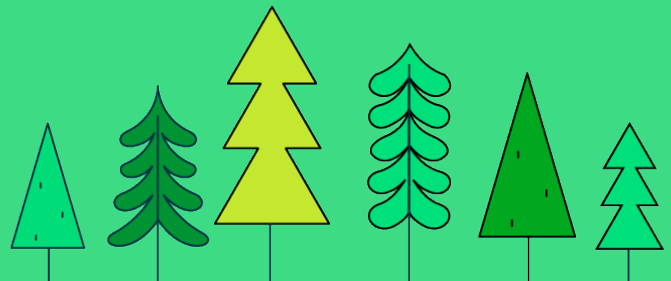
compose.material
compose.material3

compose.ui

android

Recomposición

Los elementos de UI necesitan volver a renderizarse bajo distintas circunstancias



¿cuándo?

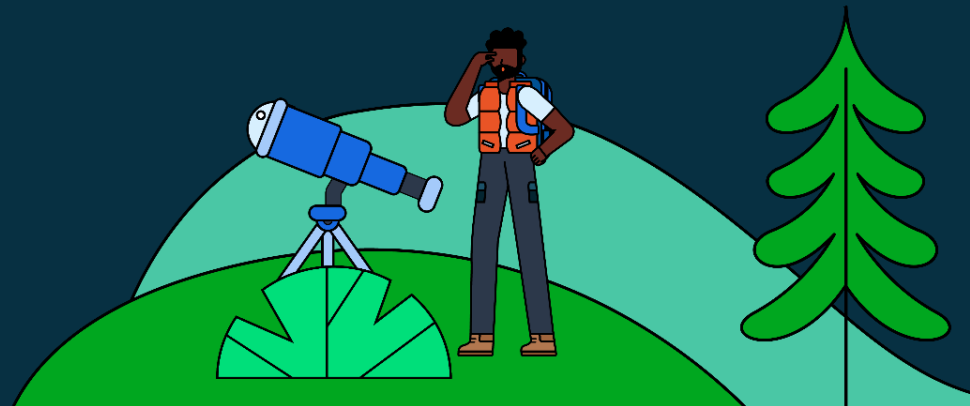
~~cómo~~

Saben

renderizar la UI

compose.runtime

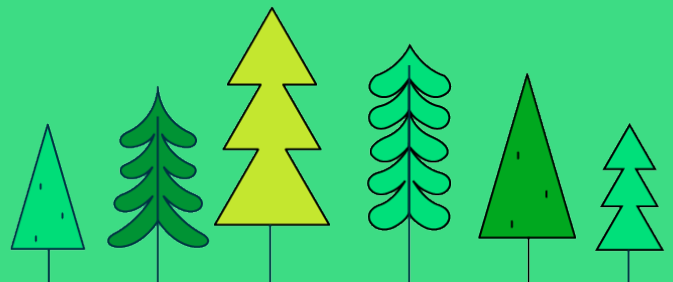
compose.compiler



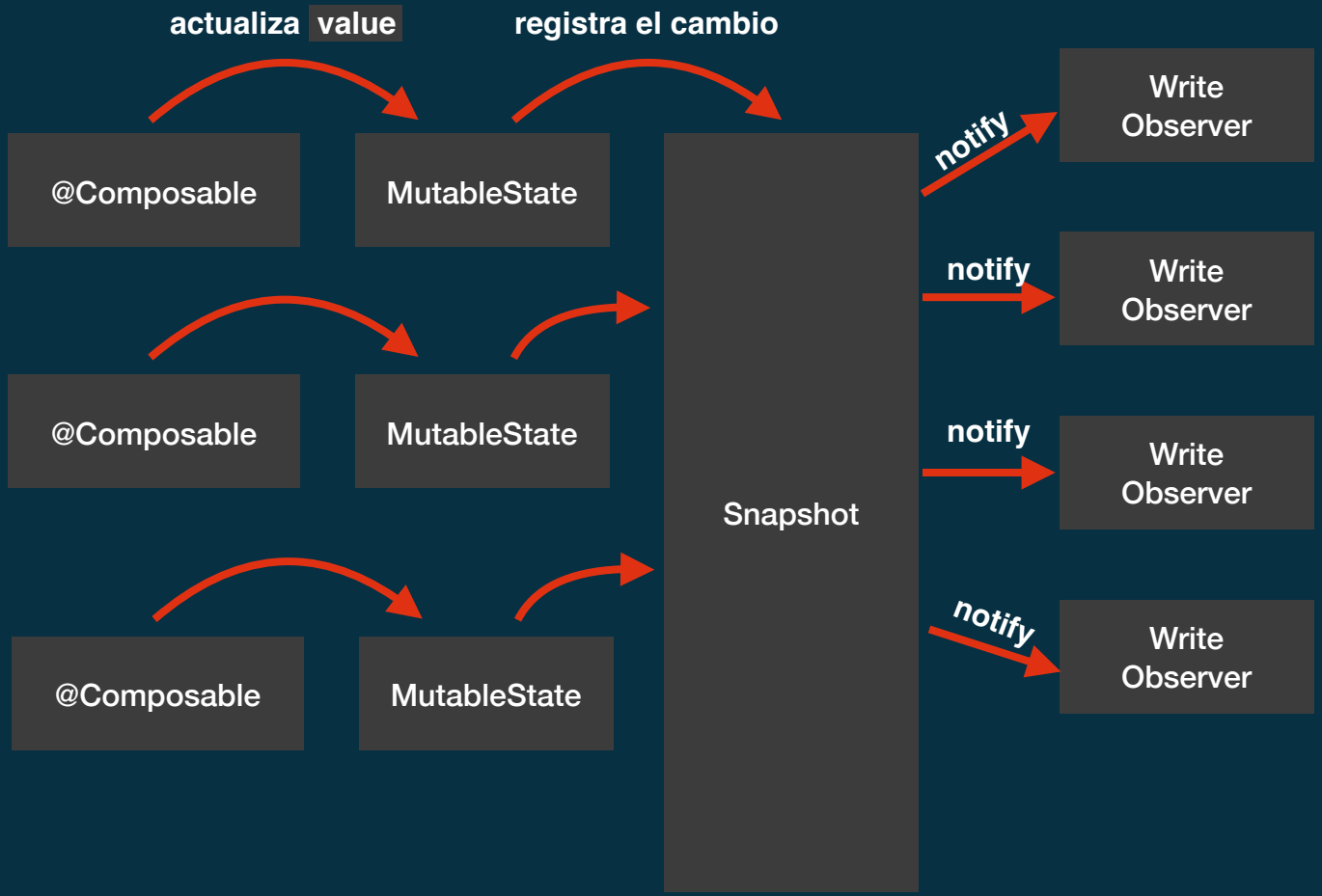
Recomposición

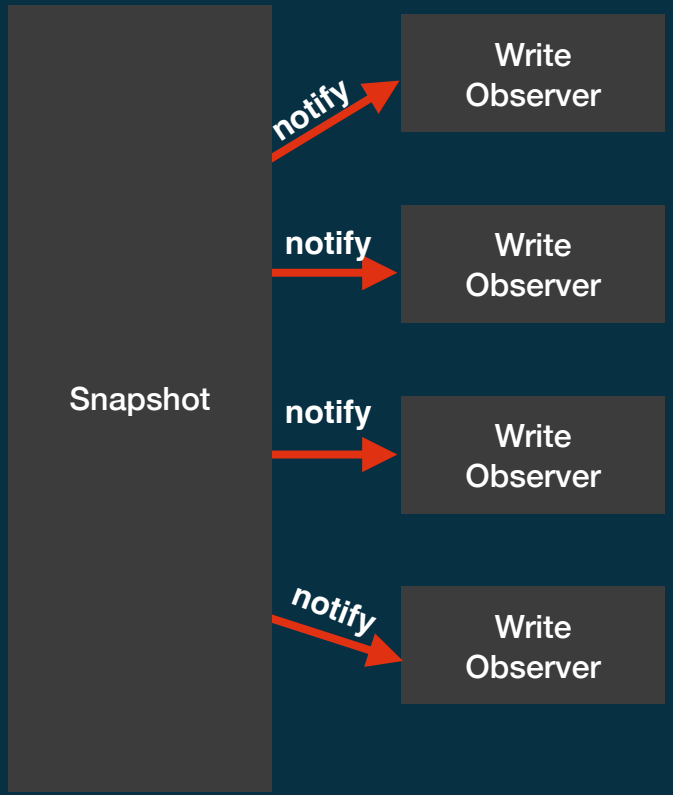
Los elementos de UI necesitan volver a renderizarse bajo distintas circunstancias

(ej. cuando los estados mutan)

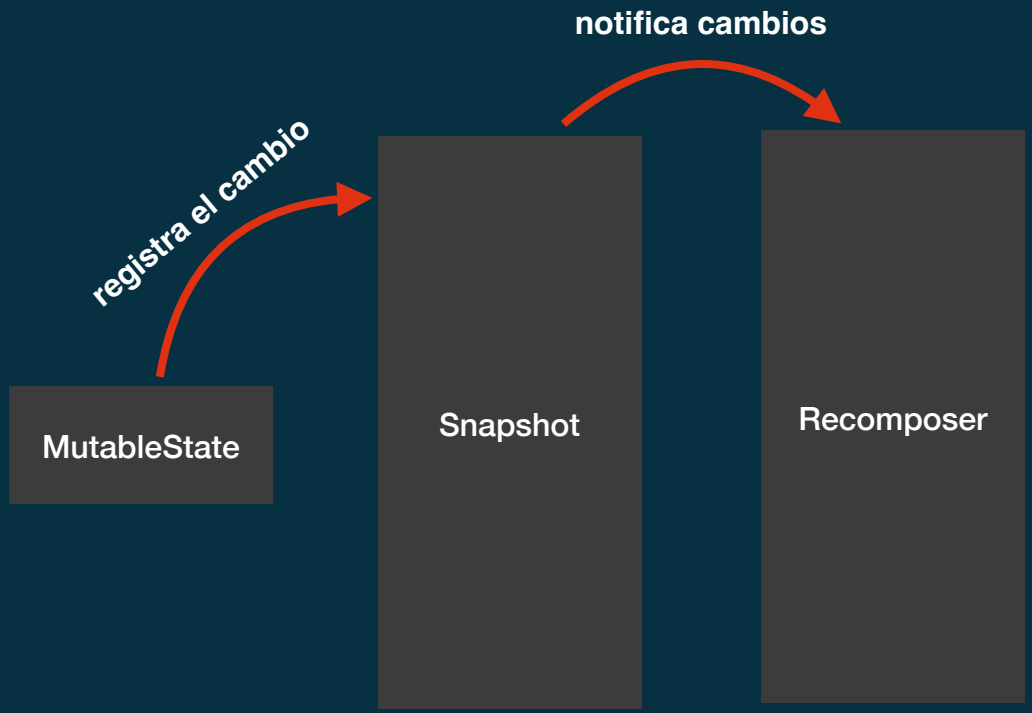


```
@Composable
fun Counter() {
    val count = remember { mutableStateOf(0) }
    Button(
        onClick = { count.value += 1 },
        content = { Text("value: ${count.value}") }
    )
}
```


actualiza value
registra el cambio
@Composable
MutableState
@Composable
MutableState
@Composable
MutableState
Snapshot
notify
notify
notify
notify
Write Observer
Write Observer
Write Observer
Write Observer

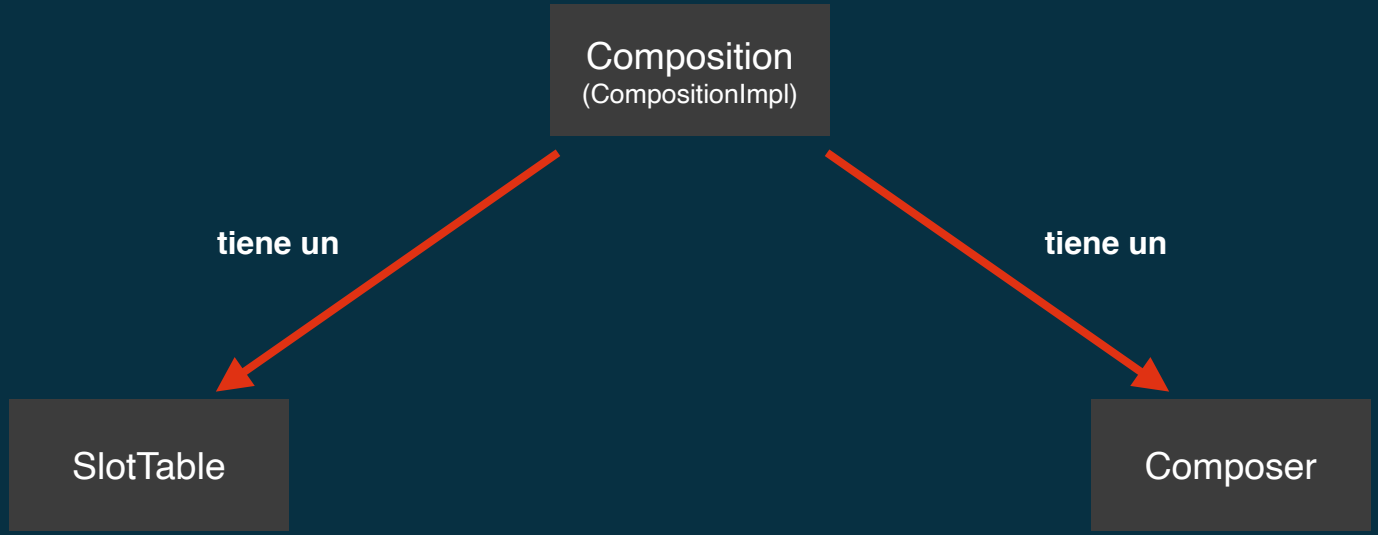

Snapshot
notify
notify
notify
notify
Write Observer
Write Observer
Write Observer
Write Observer

```
class Recomposer(...) {

  ...
  suspend fun recomposeAndApplyChanges(...) {
    ...
    val observer = Snapshot.registerApplyObserver { changed, _ ->
      appliedChanges.offer(changed)
    }
  }
  ...

}
```

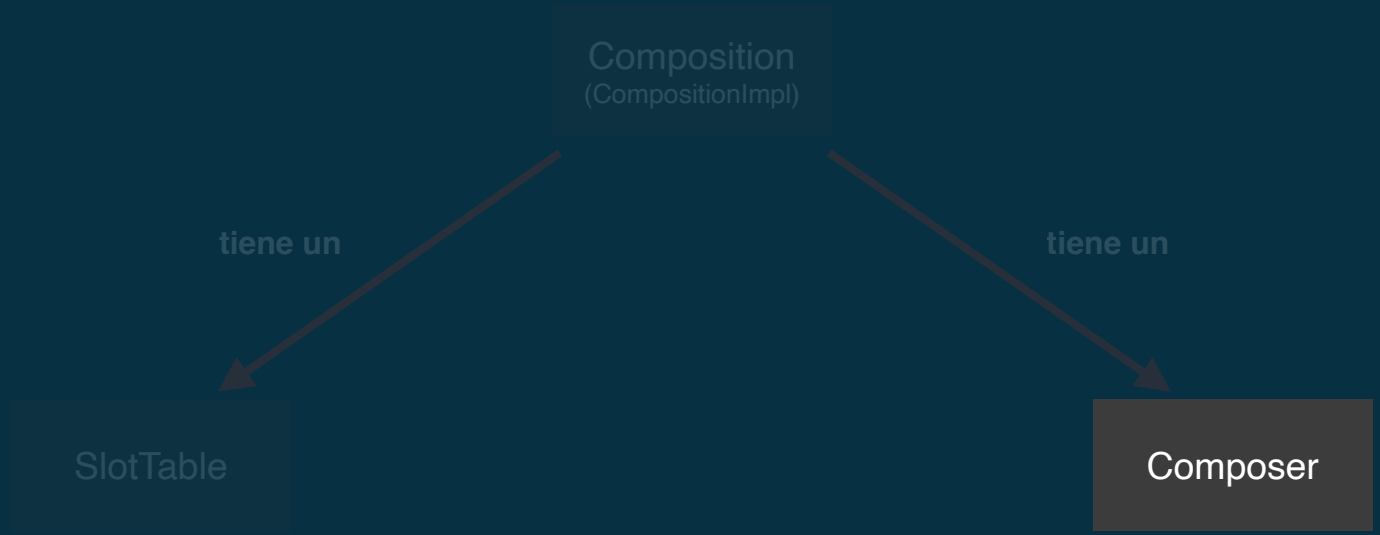

notifica cambios
registra el cambio
MutableState
Snapshot
Recomposer


setContent { ...}
crea (o accede a) un
se adhiere al padre
AndroidOwner
(AndroidComposeView)
crea (o accede a) un
Composition

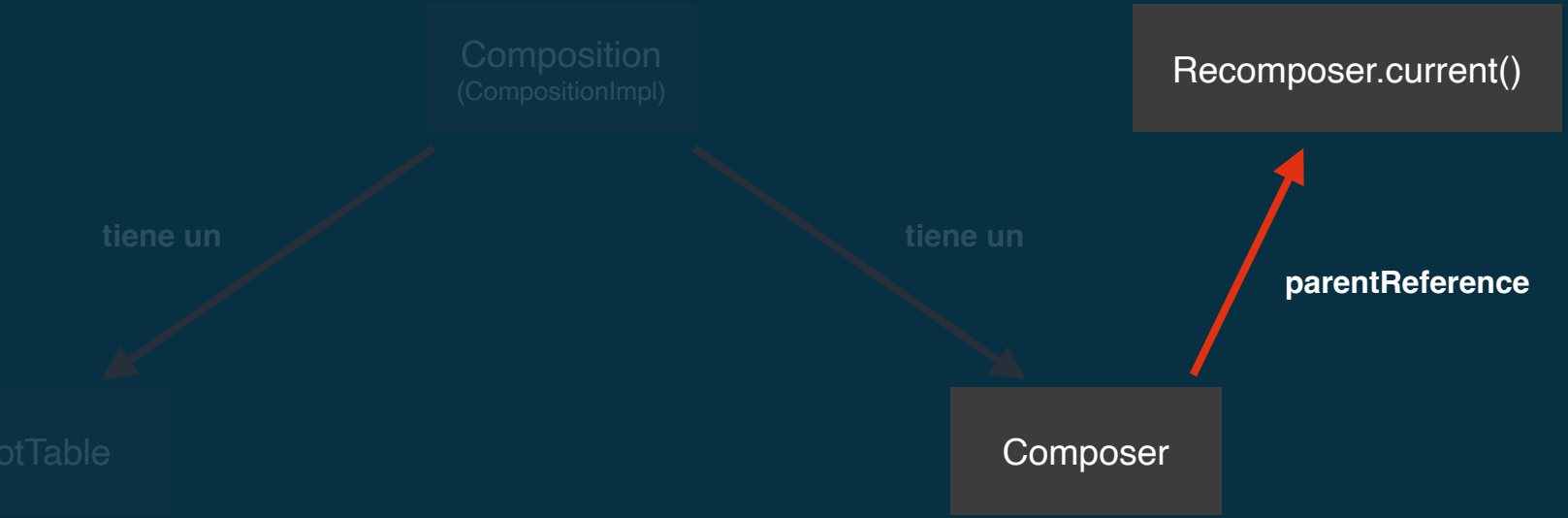
Composition

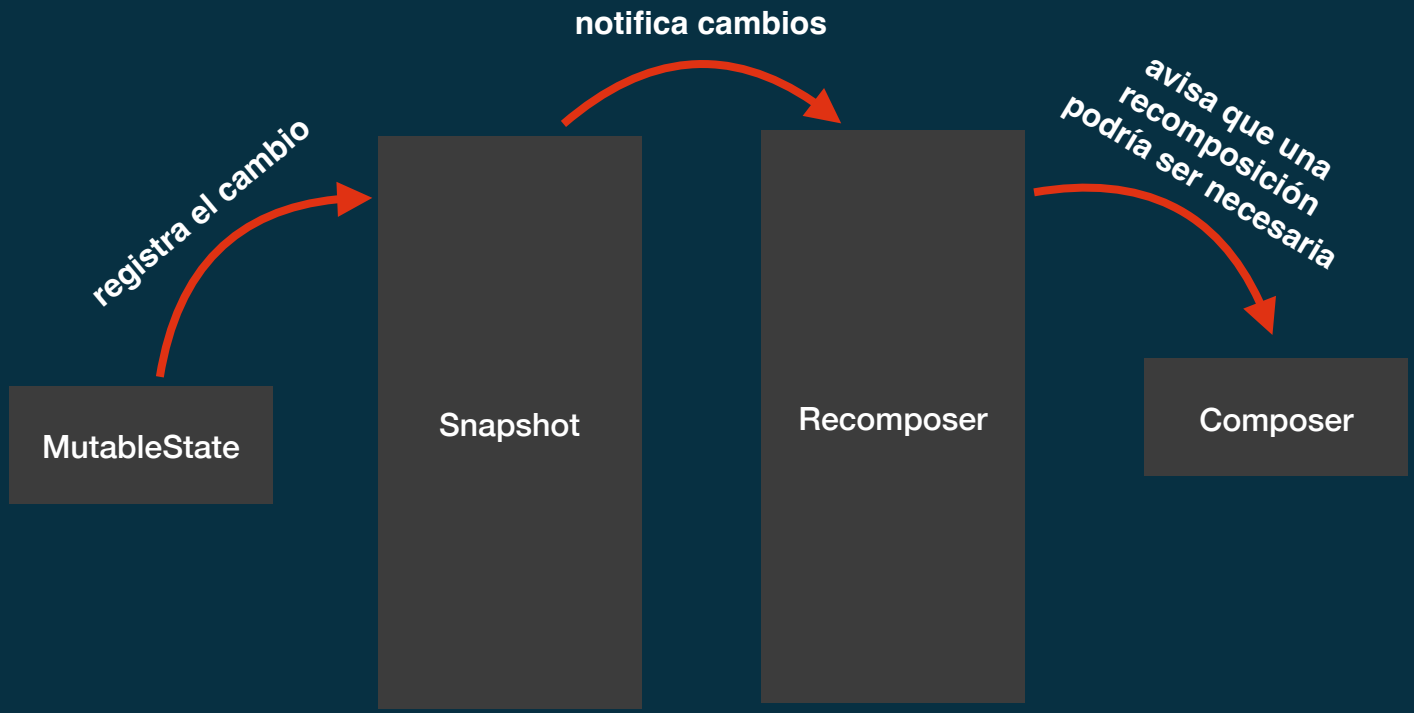

Composition
(CompositionImpl)
tiene un
tiene un
SlotTable
Composer

1. Realiza cambios (basados en positional memoization) sobre el SlotTable.
2. Coordina el "ciclo de vida” de un Composable (onEnter / onLeave)
3. Decide cómo actuar en base a cambios ocurridos en algún otro lugar.



1. Realiza cambios (basados en positional memoization) sobre el SlotTable.
2. Coordina el "ciclo de vida" de un Composable (onEnter / onLeave)
3. Decide cómo actuar en base a cambios ocurridos en algún otro lugar.


Composition
(CompositionImpl)
Recomposer.current()
tiene un
tiene un
parentReference
otTable
Composer

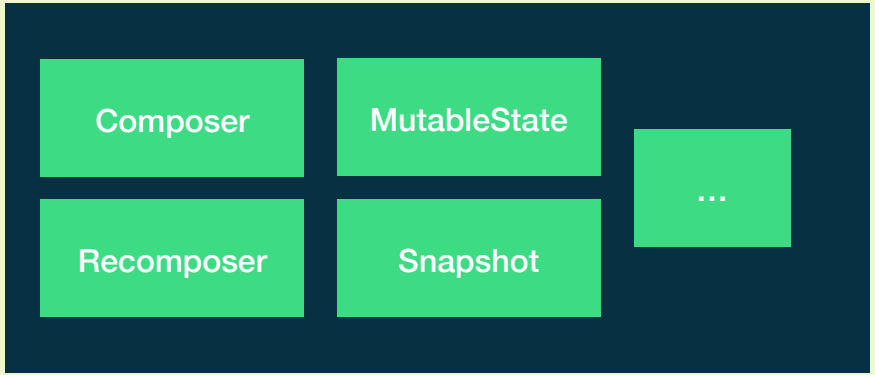

notifica cambios
registra el cambio
avisa que una recomposición podría ser necesaria
MutableState
Snapshot
Recomposer
Composer

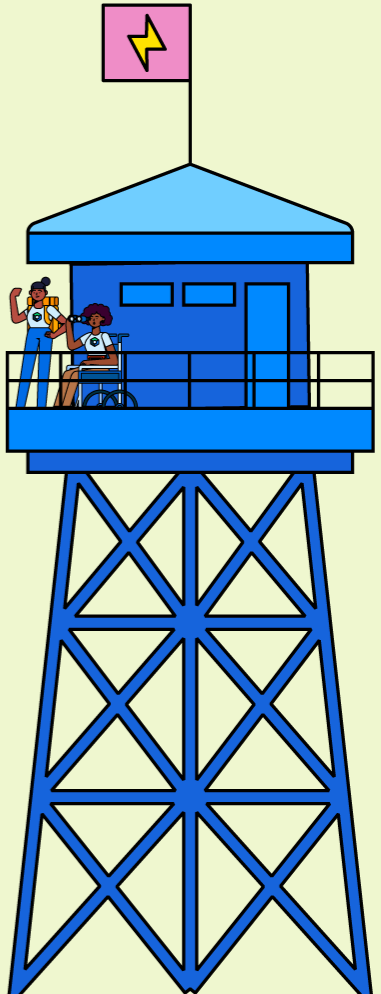
[mDevConf 2020]

#TeamNative

Native Stage - Day 3 - mDevConf 2020

Compose Runtime





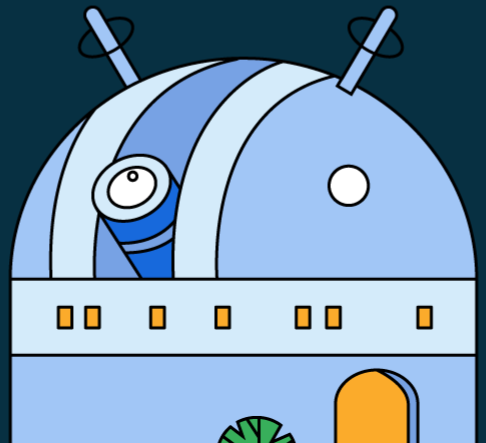
android

Compose Runtime

1. La recomposición no es ✨magia✨
2. **Registra** los cambios de estado en tiempo de ejecución (Snapshot)
3. **Ejecuta** las recomposiciones que hagan falta, en base a los cambios pendientes (Composer)

android

KAPT != Compiler Plugin != KSP



¿cuándo?

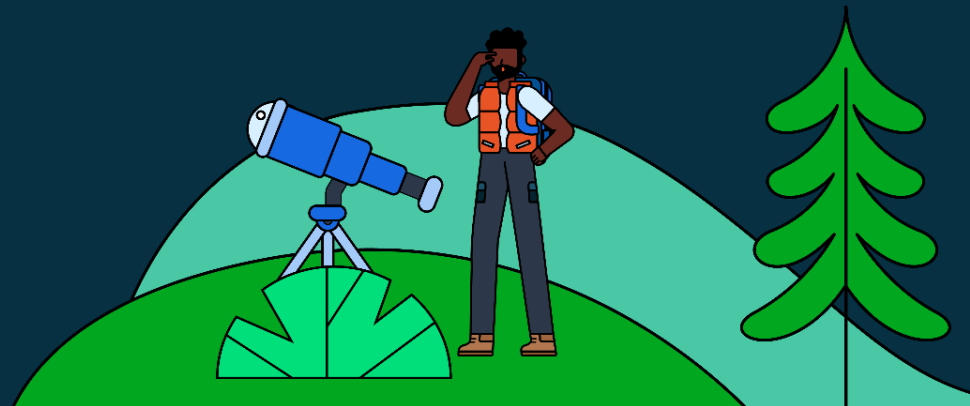
~~cómo~~

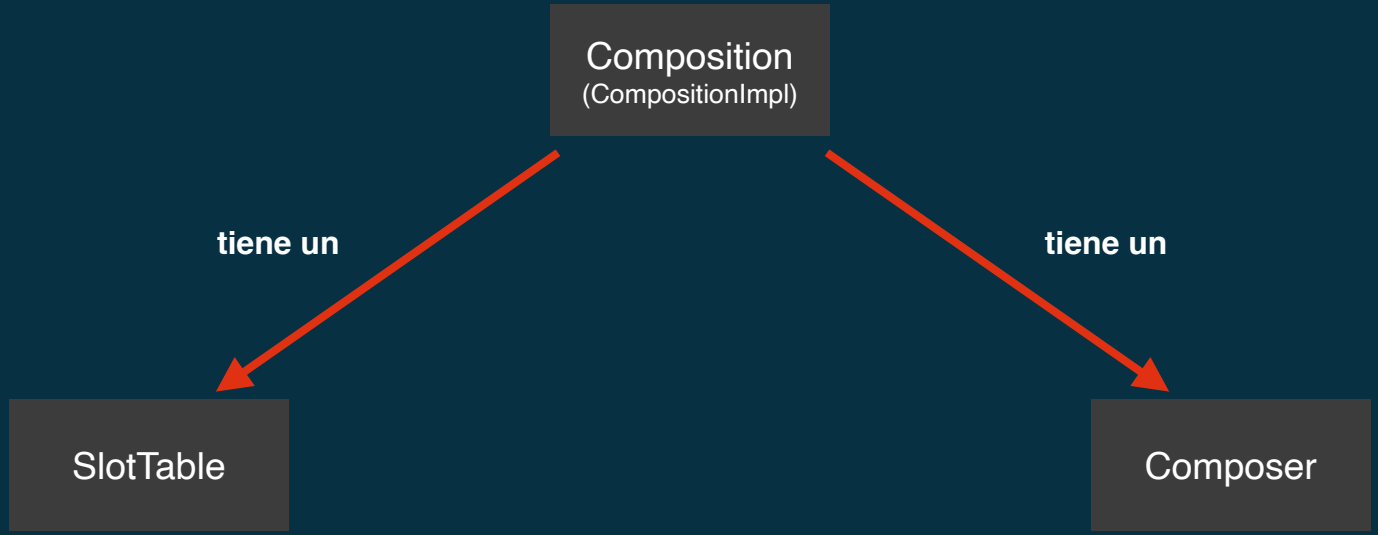
Saben

renderizar la UI

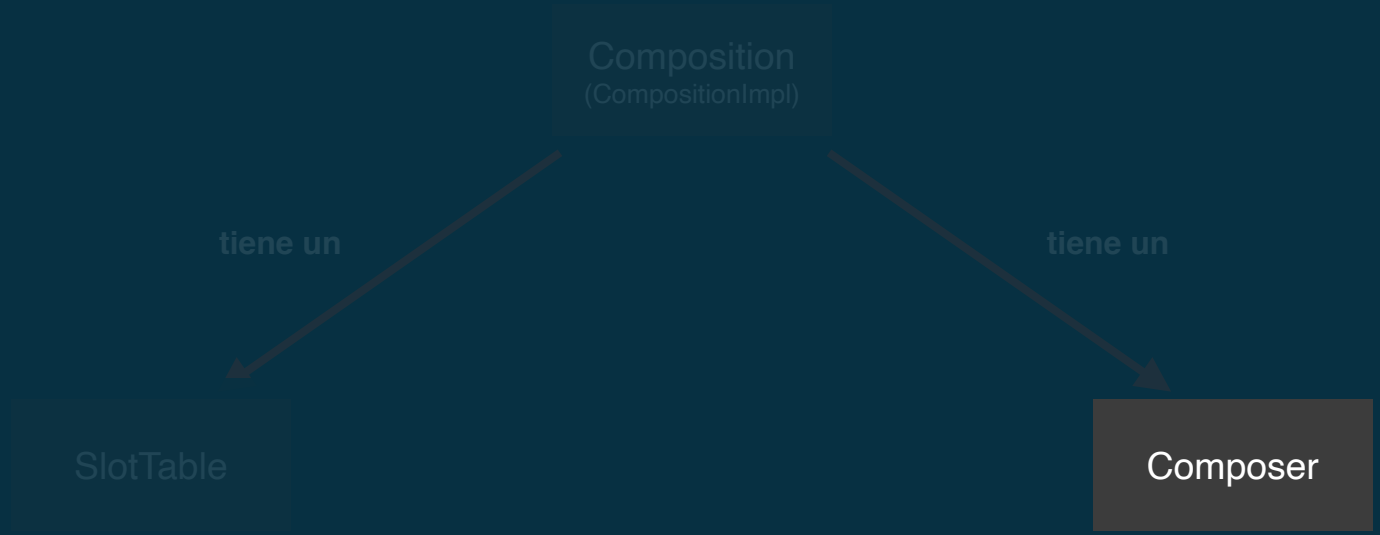
compose.runtime

compose.compiler

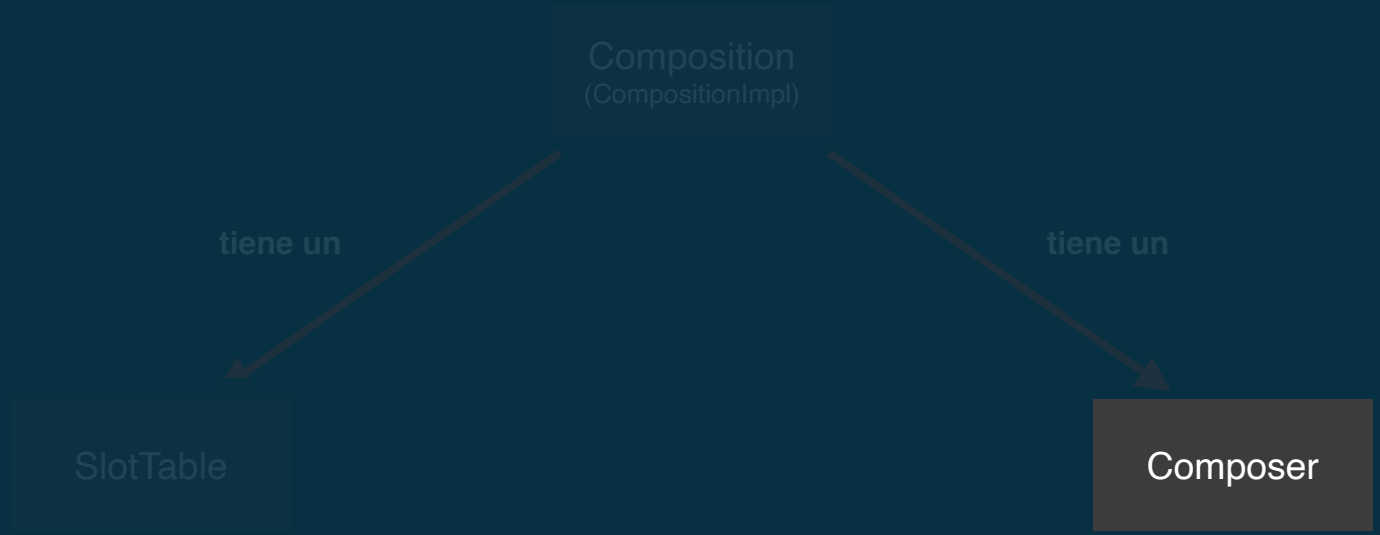




1. Realiza cambios (basados en positional memoization) sobre el SlotTable.
2. Coordina el "ciclo de vida" de un Composable (onEnter / onLeave)
3. Decide cómo actuar en base a cambios ocurridos en algún otro lugar.



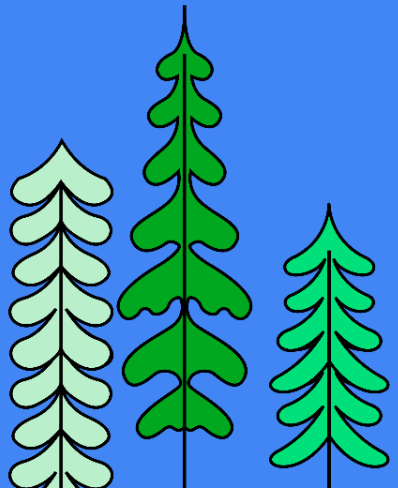
El Composer necesitar guardar referencias a cada @Composable

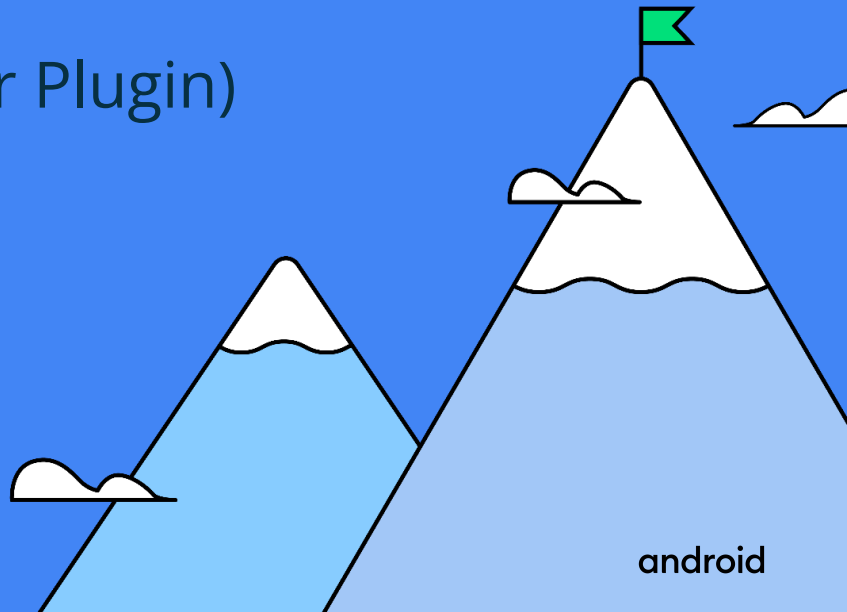


¿... pero cómo lo hace?

Compose Compiler

(Kotlin Compiler Plugin)



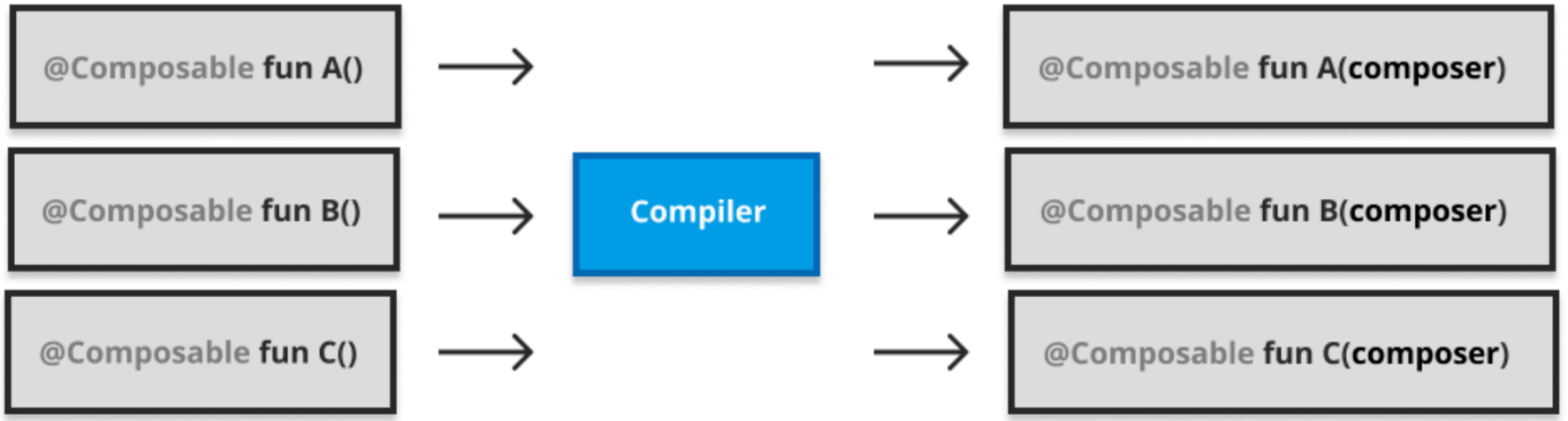


android

Kotlin Compiler Plugins pueden...

1. Modificar la representación interna (IR) del código
2. Generar nuevas clases
3. Realizar static-analysis del código

android



Source: "Jetpack Compose Internals", Chapter 2.
Author: Jorge Castillo

android

```
@Composable
fun Greetings(name: String) {
    Text("Hello ${name}")
}
```

```
@Composable
fun Greetings(name: String, $composer) {
    $composer.start(123)
    Text("Hello ${name}", $composer)
    $composer.end()
}
```

```
class ComposableFunctionBodyTransformer(
    context: IrPluginContext,
    ...
) : {

}
```

Android Code Search

```
/**
 * This IR Transform is responsible for the main transformations of the body of a composable
 * function.
 *
 * 1. Control-Flow Group Generation
 * 2. Default arguments
 * 3. Composable Function Skipping
 * 4. Comparison Propagation
 * 5. Recomposability
 * 6. Source location information (when enabled)
 */
class ComposableFunctionBodyTransformer(
    context: IrPluginContext,
    ...
) : {

}
```

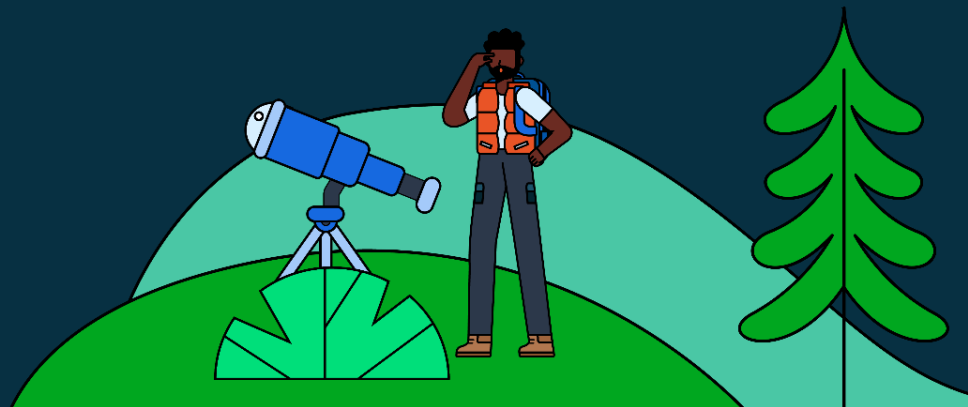
Compiler Frontend

Static-analysis checks before compile time (i.e. warning suppressions)

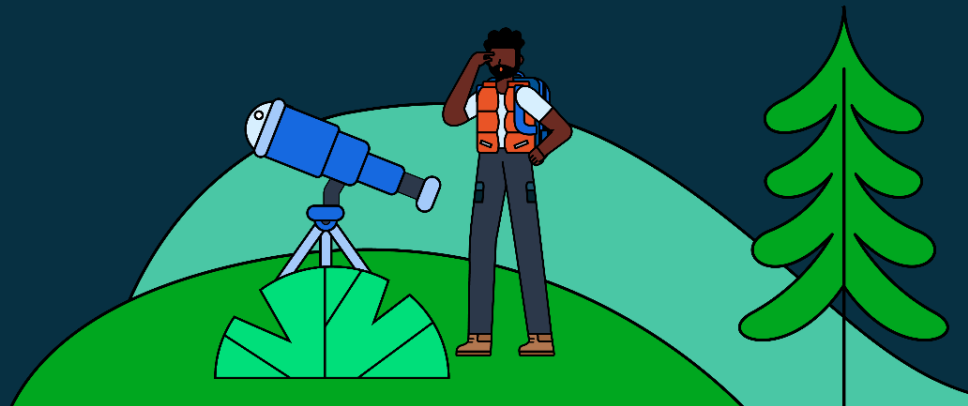
Compiler Backend

```
@Composable
fun CustomComponent(formatter: String.(suffix: String)->String) {
    "Hello".formatter(suffix = “world")
}
```

Named arguments are not allowed for function types



```
@Composable
fun CustomComponent(@Composable formatter: String.(suffix: String)->String) {
    "Hello".formatter(suffix = “world")
}
```



android

```
open class ComposeDiagnosticSuppressor : DiagnosticSuppressor {

    override fun isSuppressed(
        diagnostic: Diagnostic,
        bindingContext: BindingContext?
    ): Boolean {
        if (diagnostic.factory == Errors.NAMED_ARGUMENTS_NOT_ALLOWED) {
            val call = getResolvedCall(bindingContext)
            return call.isComposableInvocation
        }
        return false
    }

}
```

Android Code Search

Compose Runtime

+

Compose Compiler

Compose UI

(Android)

android

Compose Runtime

+

Compose Compiler

Compose UI

(iOS / Desktop / Web)

Compose Runtime

1. La recomposición no es ✨magia✨
2. **Registra** los cambios de estado en tiempo de ejecución (Snapshot)
3. **Ejecuta** las recomposiciones que hagan falta, en base a los cambios pendientes (Composer)

android

```
/**
 * An Applier is responsible for applying the tree-based
 * operations that get emitted during a composition.

 * Every [Composer] has an [Applier] which it
 * uses to emit a [ComposeNode].
 *
*/
 interface Applier<N> { ... }
```

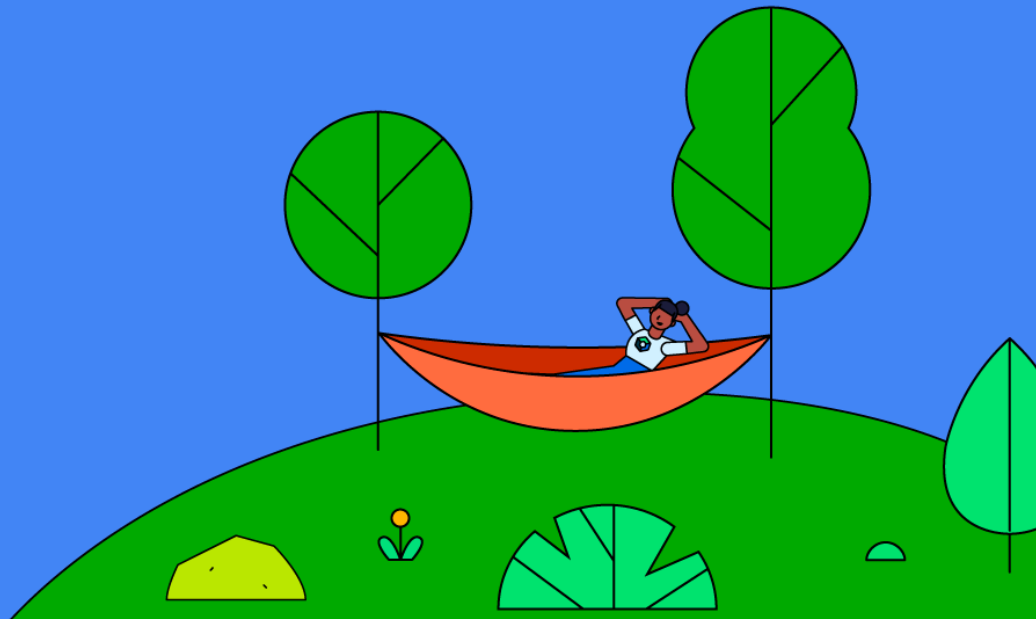
Compose UI

Applier

Compose UI

(iOS / Desktop / Web)

Recursos Adicionales



android

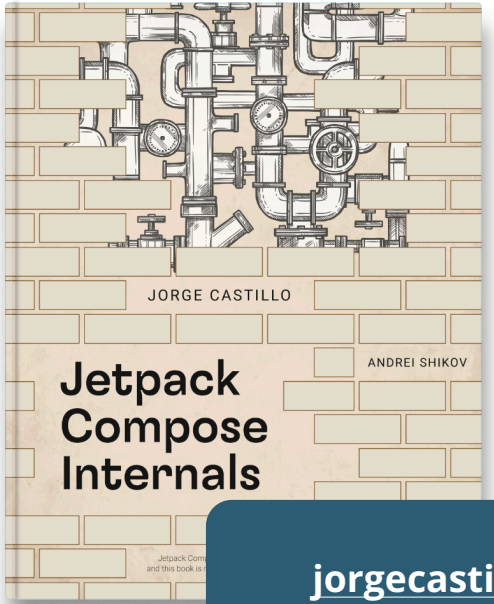
Jorge Castillo HOME ABOUT BOOK 📖 JETPACK COMPOSE INTERNALS 🚀

Jetpack Compose internals 📖

Do you wonder how Jetpack Compose works internally, or how the compiler or the runtime work together? Are you curious about other use cases for Compose? Did you ever think about how Composable functions communicate with the compiler and the runtime?

It's your lucky day 🙌 **Jetpack Compose internals** is your chance to go one step further and learn the guts of what will become the new standard of Android UI.

Get the book | Follow on Twitter

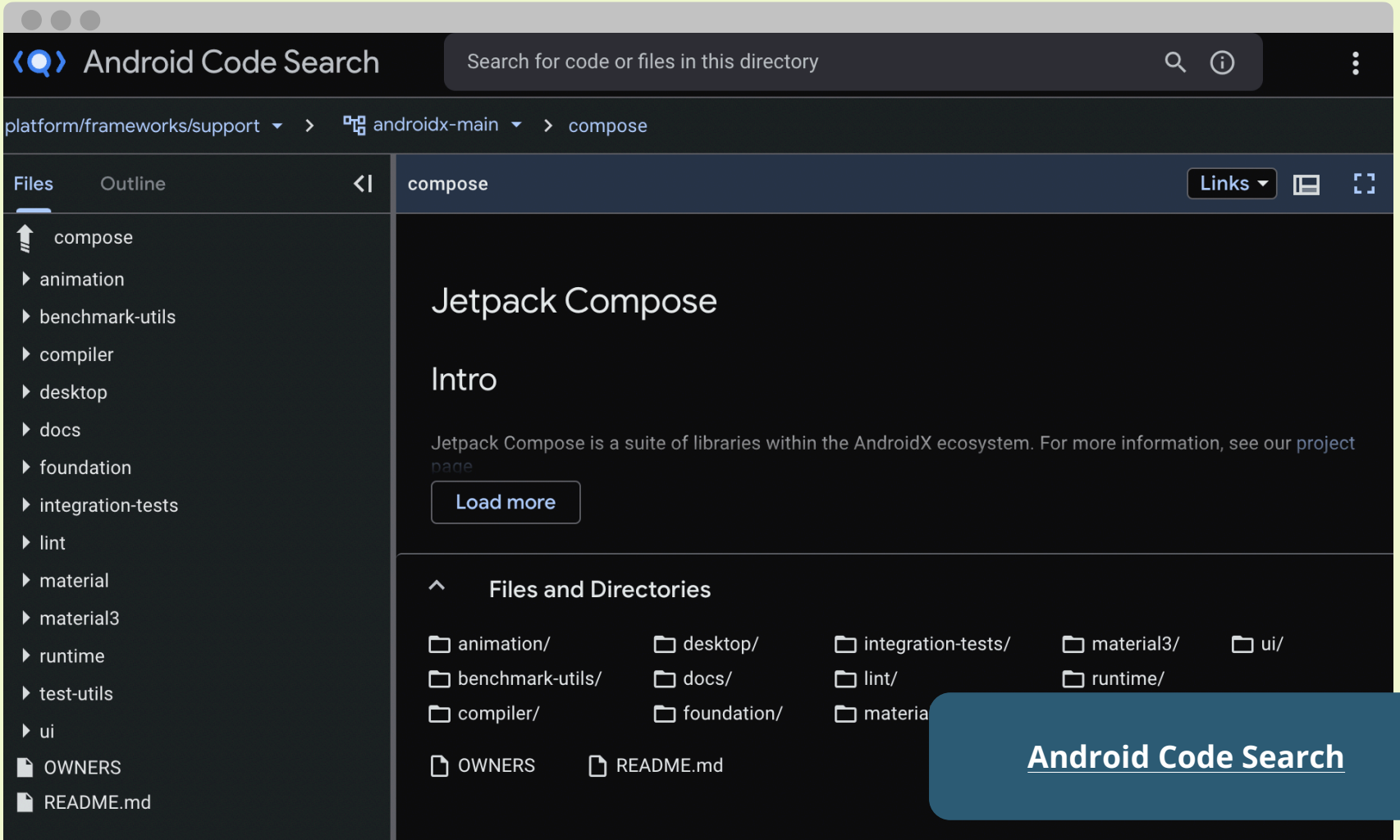


jorgecastillo.dev/book

[mDevConf 2020]

#TeamNative

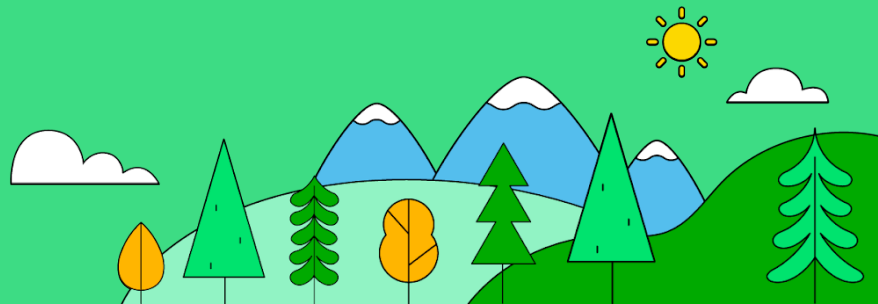
Native Stage - Day 3 - mDevConf 2020


Android Code Search
Search for code or files in this directory
platform/frameworks/support
androidx-main
compose
Files
Outline
compose
animation
benchmark-utils
compiler
desktop
docs
foundation
integration-tests
lint
material
material3
runtime
test-utils
ui
OWNERS
README.md
compose
Links
Jetpack Compose
Intro
Jetpack Compose is a suite of libraries within the AndroidX ecosystem. For more information, see our project page
Load more
Files and Directories
animation/
benchmark-utils/
compiler/
desktop/
docs/
foundation/
integration-tests/
lint/
material3/
runtime/
ui/
OWNERS
README.md
Android Code Search

Gracias!

Bruno Aybar
Shopify, Senior Mobile Developer

Twitter: @brunoaybarg
Github: @Bruno125



android