

ERES 2024

Criando um App de Tarefas com Kotlin e Android

PEDRO DENARDI MINUZZI

Cronograma

Principais tópicos

- O que é Android?
- Banco de dados SQL
- Jetpack Compose
- Linguagem de programação Kotlin
- Android Studio
- Projeto

O que é Android?



Sistema operacional (SO) para dispositivos móveis, como smartphones e tablets, e outros aparelhos, como smart TVs e relógios inteligentes.

Banco de dados SQL



ENTITY

Entidades responsáveis por mapear as tabelas.

DAO

Interfaces utilizadas para acessar os dados armazenados no banco

DATABASE

Representação da classe abstrata do Banco de Dados



- Menos código
- Poderoso
- API declarativa
- Interoperabilidade
- Nativo
- Flexibilidade

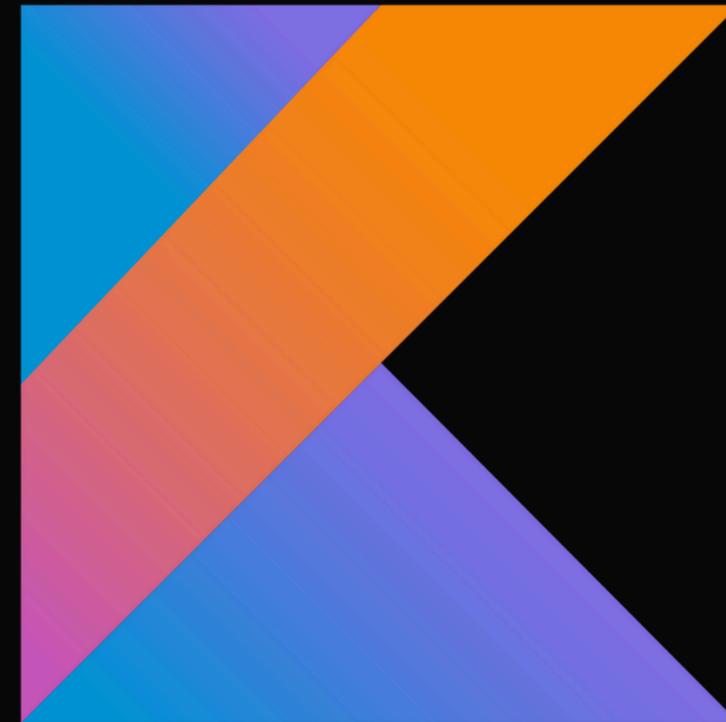


- Mais código
- Linguagem extensível
- Linguagem de marcação leve
- API não declarativa
- Menor flexibilidade
- Tempo de desenvolvimento maior

Linguagem de programação Kotlin

JETBRAINS

Kotlin é uma linguagem de programação multiplataforma, orientada a objetos e funcional, concisa e estaticamente tipada, desenvolvida em 2011 pela empresa JetBrains.



```
1 @file:Suppress(...names: "SpellCheckingInspection", "KotlinConstantConditions")
2
3 package com.software.todo
4
5 ▶ fun main() {
6     // Ao atribuir um valor, não é possível alterar esse valor
7     val myVal1: Int = 0
8
9     print("Valor: $myVal1") // Vai printar 0
10
11     // Compilador não deixa eu alterar o valor devido essa variavel ser imutavel
12     myVal1 = 1
13
14
15
16     // Ao atribuir um valor, eu posso alterar esse valor depois
17     var myVal2: Int = 0
18
19     print("Valor: $myVal2") // Vai printar 0
20
21     // Estou alterando o valor da variavel para 1
22     myVal2 = 1
23
24     print("Valor: $myVal2") // Vai printar 1
25 }
```



```
40 // Uma classe que contém métodos responsável por fazer somas e divisões
41 // dos valores passados no construtor 'Calculate(1,1)'
42 class Calculate(
43     private val a: Int,
44     private val b: Int
45 ) {
46
47     // Função responsável por multiplicar dois números do tipo [Int] e retornar a multiplicação
48     private fun multiply(a: Int, b: Int): Int { // Declaramos o tipo de retorno da função
49         return a * b // [return] palavra reservada do kotlin
50     }
51
52     // Função responsável por somar dois números do tipo [Int] e retornar a soma
53     fun plus(): Int { // Declaramos o tipo de retorno da função
54         return a + b // [return] palavra reservada do kotlin
55     }
56
57     // Função responsável por somar dois números e retornar a soma
58     fun minus(): Int { // Declaramos o tipo de retorno da função
59         return a - b // [return] palavra reservada do kotlin
60     }
61
62 }
```

```
65 ▶ fun main() {
66     // Instanciar uma [data class]
67     val numbers = Numbers(a: 1, b: 2)
68
69     // Para acessar os valores/métodos da [data class]
70     numbers.a
71     numbers.b
72
73     // Para instanciar uma [class]
74     val calculator = Calculator()
75
76     // Para acessar as funções da [class]
77     calculator.plus(numbers)
78     calculator.minus(a: 1, b: 2)
79
80     // Para instanciar uma [class] que tem parâmetros no construtor
81     val calculate = Calculate(a: 1, b: 2)
82
83     // OBS: métodos/variáveis privadas não é possível acessar fora da [class]
84     calculate.multiply(a: 1, b: 2)
85 }
```

```
64 // Classe com herança de [Calculator] com novas funcionalidades da nossa calculadora
65 // OBS: [class] sem a palavra reservada 'open' não podem ser herdadas
66 class CalculatorV2 : Calculator() {
67
68     // Função responsável por multiplicar dois números do tipo [Int] e retornar a multiplicação
69     fun multiply(a: Int, b: Int): Int { // Declaramos o tipo de retorno da função
70         return a * b // [return] palavra reservada do kotlin
71     }
72
73     // Função responsável por dividir dois números do tipo [Int] e retornar a divisão
74     fun divide(a: Int, b: Int): Int { // Declaramos o tipo de retorno da função
75         return a / b // [return] palavra reservada do kotlin
76     }
77
78 }
```

```
25 // Uma classe que contém métodos responsável por fazer somas e divisões de valores/números
26 @ ↓ v open class Calculator {
27
28     // Função responsável por somar dois números e retornar a soma
29     v fun plus(numbers: Numbers): Int { // Declaramos o tipo de retorno da função
30         | return numbers.a + numbers.b // [return] palavra reservada do kotlin
31     }
32
33     // Função responsável por somar dois números do tipo [Int] e retornar a soma
34     v fun minus(a: Int, b: Int): Int { // Declaramos o tipo de retorno da função
35         | return a - b // [return] palavra reservada do kotlin
36     }
37
38 }
```

```
64 // Classe com herança de [Calculator] com novas funcionalidades da nossa calculadora
65 // OBS: Agora é possível herdar a [class] 'Calculator' uma vez que ela é aberta
66 class CalculatorV2 : Calculator() {
67
68     // Função responsável por multiplicar dois números do tipo [Int] e retornar a multiplicação
69     fun multiply(a: Int, b: Int): Int { // Declaramos o tipo de retorno da função
70         return a * b // [return] palavra reservada do kotlin
71     }
72
73     // Função responsável por dividir dois números do tipo [Int] e retornar a divisão
74     fun divide(a: Int, b: Int): Int { // Declaramos o tipo de retorno da função
75         return a / b // [return] palavra reservada do kotlin
76     }
77
78     // Função que faz uma calculo complexo
79     fun complexCalculus(a: Int, b: Int): Int { // Declaramos o tipo de retorno da função
80         val result1 = multiply(a,b)
81         val result2 = minus(result1,b)
82         val result3 = divide(result2,a)
83         return result3 + result1 // [return] palavra reservada do kotlin
84     }
```

```
88 // Lambda retornando nada '() -> Unit';
89 // Lambda requisitando um valor '() -> Int`;
90 // Lambda retornando um valor '(result: Int) -> Unit`;
91
92 // Função que faz uma soma e tem como parâmetro um lambda '(result: Int) -> Unit'
93 √ fun calculate1(a: Int, b: Int, result: (result: Int) -> Unit) {
94     result(result: a + b) // Instanciar um lambda passando valores
95 }
96
97 // Função que faz uma soma e tem como parâmetro um lambda '() -> Int'
98 √ fun calculate2(a: Int, b: Int, result: () -> Int) {
99     result() // Instanciar um lambda sem passar valores
100 }
101
102 // Função que faz uma soma e tem como parâmetro um lambda '() -> Unit'
103 √ fun calculate3(a: Int, b: Int, result: () -> Unit) {
104     result() // Instanciar um lambda sem passar valores
105 }
```

```
107 ▶ fun main() {
108     // Função que tem como parametro um lambda '(result: Int) -> Unit'
109     calculate1(a: 1, b: 2) { result ->
110         val a = result // Aqui eu consigo acessar o resultado que obtive do calculo da função
111     }
112
113     // Função que tem como parametro um lambda '() -> Int'
114     calculate2(a: 1, b: 2) {
115         1 // No final do bloco, é necessário retornar um valor
116
117         // OBS: é possível retornar um valor assim também: 'return@calculate2 1'
118         // mas o compilar é esperto o suficiente e não precisa escrever o return
119     }
120
121     // Função que tem como parametro um lambda '() -> Unit'
122     calculate3(a: 1, b: 2) {
123         // Não retorna nada, logo, apenas executa esse trecho de código
124     }
125 }
```

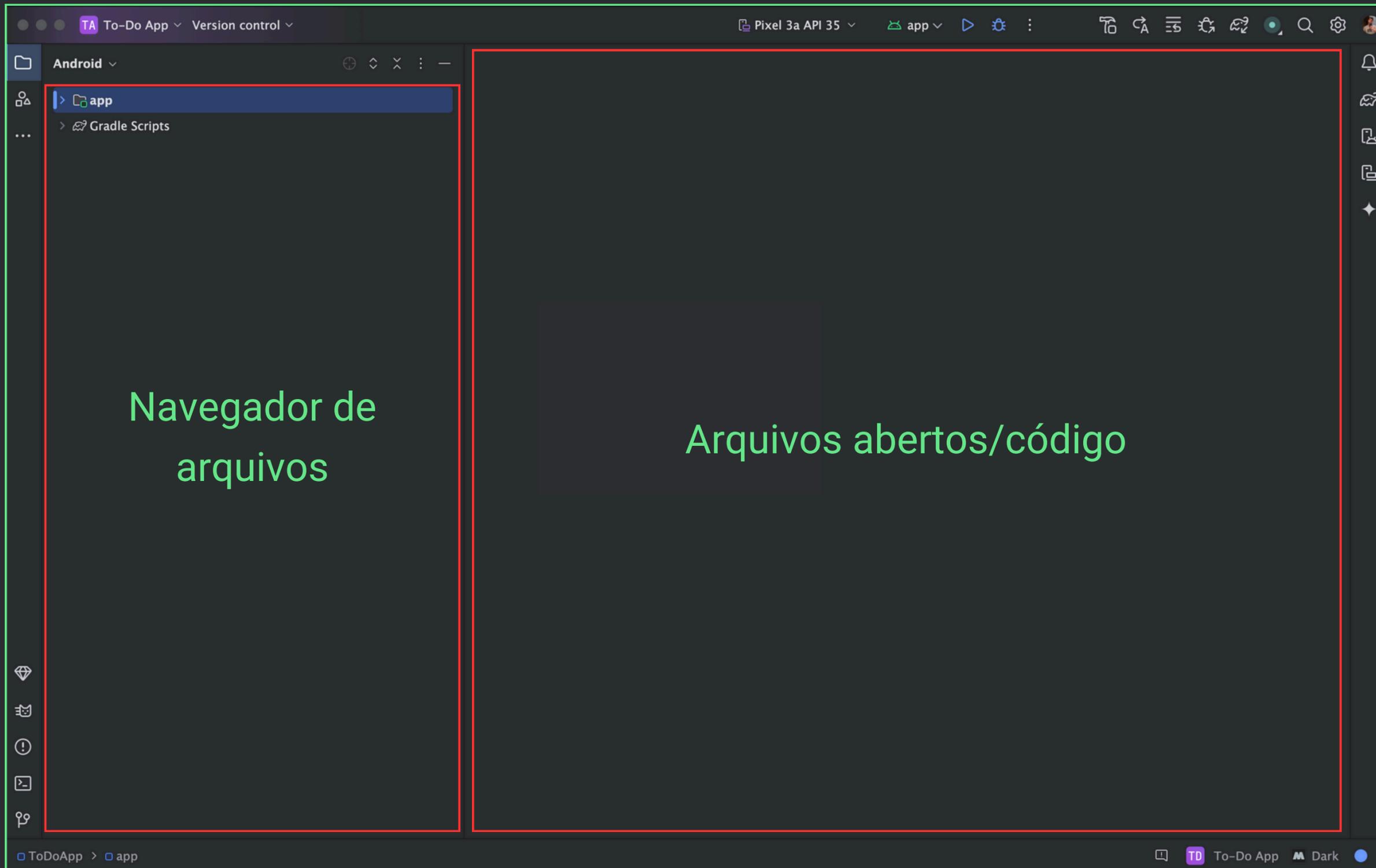
```
110 // Activity é uma [class] do Android usada para instanciar screens (telas)
111 ▶ class MyActivity : Activity() {
112
113     // Estamos sobrescrevendo a função aberta 'onCreate' (aqui é onde desenhamos o layout)
114     // OBS: override significa que vamos sobrescrever a função
115     // que já existe dentro da [class] que herdamos
116 Ⓞ↑ override fun onCreate(savedInstanceState: Bundle?) {
117     // OBS: super.nomeDaFunção significa que primeiro vamos executar o código 'onCreate'
118     // dentro da [class] 'Activity' para depois executar o nosso código
119     // que vamos escrever
120     super.onCreate(savedInstanceState)
121     // código
122     (...)
123 }
124
125 }
```

Android Studio

BY GOOGLE

Android Studio é um ambiente de desenvolvimento integrado para desenvolver para o sistema Android, baseado no software IntelliJ IDEA da empresa JetBrains





Projeto

To-Do Application

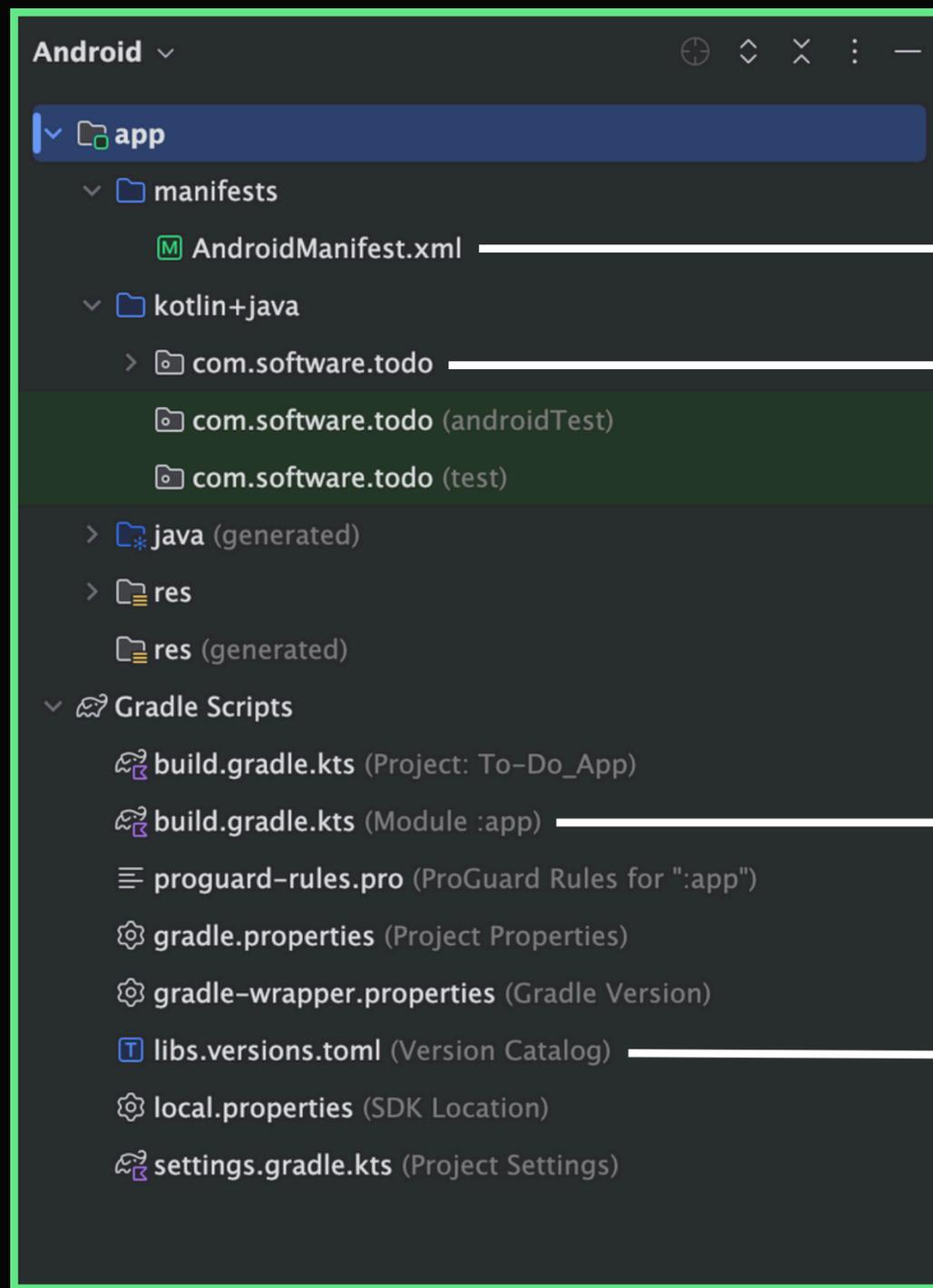


Android

- app
 - manifests
 - AndroidManifest.xml
 - kotlin+java
 - com.software.todo (androidTest)
 - com.software.todo (test)
 - com.software.todo.presentation.theme
 - Theme.kt
 - Type.kt
 - java (generated)
 - res
 - res (generated)
 - Gradle Scripts
 - build.gradle.kts (Project: To-Do_App)
 - build.gradle.kts (Module :app)
 - proguard-rules.pro (ProGuard Rules for ":app")
 - gradle.properties (Project Properties)
 - gradle-wrapper.properties (Gradle Version)
 - libs.versions.toml (Version Catalog)
 - local.properties (SDK Location)
 - settings.gradle.kts (Project Settings)

Theme.kt

```
1 @file:Suppress(...names: "NewApi")
2
3 package com.software.todo.presentation.theme
4
5 > import ...
6
7
8
9
10
11
12
13
14
15
16
17 @Composable
18 @RequiresApi(Build.VERSION_CODES.S)
19 private fun getDynamicColorScheme(darkTheme: Boolean): ColorScheme {
20     val context = LocalContext.current
21     return if (darkTheme) dynamicDarkColorScheme(context) else dynamicLightColorScheme(context)
22 }
23
24 @Composable
25 @RequiresApi(Build.VERSION_CODES.O)
26 private fun getLightColorScheme(): ColorScheme {
27     return lightColorScheme()
28 }
29
30 @Composable
31 @RequiresApi(Build.VERSION_CODES.O)
32 private fun getDarkColorScheme(): ColorScheme {
33     return darkColorScheme()
34 }
35
36 @Composable
37 fun Theme(
38     darkTheme: Boolean = isSystemInDarkTheme(),
39     content: @Composable () -> Unit
40 ) {
41     val colorScheme = when {
42         Build.VERSION.SDK_INT >= Build.VERSION_CODES.S -> getDynamicColorScheme(darkTheme)
43         darkTheme -> getDarkColorScheme()
44         else -> getLightColorScheme()
45     }
46
47     MaterialTheme(
```



Configurações do App

Código do App

Configurações de build

Bibliotecas

Android ▾

- app
 - manifests
 - AndroidManifest.xml
 - kotlin+java
 - com.software.todo (androidTest)
 - com.software.todo (test)
 - com.software.todo.presentation.theme
 - Theme.kt
 - Type.kt
 - java (generated)
 - res
 - res (generated)
 - Gradle Scripts
 - build.gradle.kts (Project: To-Do_App)
 - build.gradle.kts (Module :app)
 - proguard-rules.pro (ProGuard Rules for ":app")
 - gradle.properties (Project Properties)
 - gradle-wrapper.properties (Gradle Version)
 - libs.versions.toml (Version Catalog)
 - local.properties (SDK Location)
 - settings.gradle.kts (Project Settings)

Theme.kt | libs.versions.toml ×

Gradle files have changed since last project sync. A project sync may be necessary for the IDE to work properly. Sync Now Ignore these changes

```
1 [versions]
2 android = "8.5.2"
3
4 ksp = "2.0.0-1.0.22"
5 kotlin = "2.0.0"
6 kotlinCoroutine = "1.9.0-RC"
7
8 roomKtx = "2.6.1"
9 coreKtx = "1.15.0"
10 lifecycleKtx = "2.8.7"
11
12 composeActivity = "1.9.3"
13 composeBom = "2024.10.01"
14
15 composeViewModel = "2.8.7"
16 composeRuntime = "2.8.7"
17 composeIcons = "1.7.5"
18
19 [libraries]
20 androidx-core-ktx = { group = "androidx.core", name = "core-ktx", version.ref = "coreKtx" }
21 androidx-lifecycle-runtime-ktx = { group = "androidx.lifecycle", name = "lifecycle-runtime-ktx", version.ref = "lifecycleKtx" }
22
23 jetpack-room-compiler = { module = "androidx.room:room-compiler", version.ref = "roomKtx" }
24 jetpack-room-runtime = { module = "androidx.room:room-runtime", version.ref = "roomKtx" }
25 jetpack-room-kotlin = { module = "androidx.room:room-ktx", version.ref = "roomKtx" }
26
27 androidx-activity-compose = { group = "androidx.activity", name = "activity-compose", version.ref = "composeActivity" }
28 androidx-compose-bom = { group = "androidx.compose", name = "compose-bom", version.ref = "composeBom" }
29
30 androidx-ui = { group = "androidx.compose.ui", name = "ui" }
31 androidx-ui-graphics = { group = "androidx.compose.ui", name = "ui-graphics" }
32 androidx-ui-tooling = { group = "androidx.compose.ui", name = "ui-tooling" }
33 androidx-ui-tooling-preview = { group = "androidx.compose.ui", name = "ui-tooling-preview" }
34
35 androidx-view-model = { module = "androidx.lifecycle:lifecycle-viewmodel-compose", version.ref = "composeViewModel" }
36 androidx-lifecycle-runtime = { module = "androidx.lifecycle:lifecycle-runtime-compose", version.ref = "composeRuntime" }
```

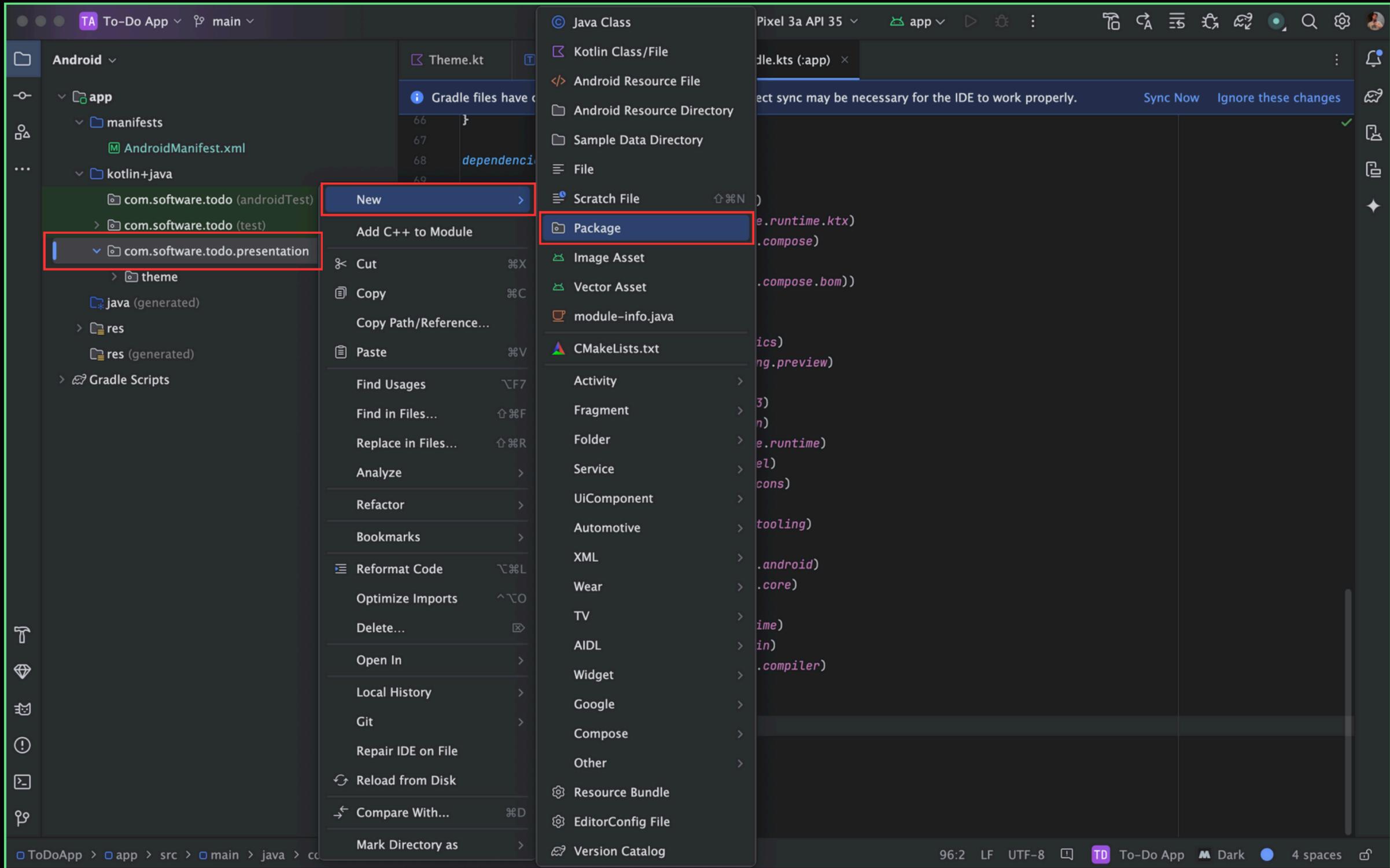
Android ▾

- app
 - manifests
 - AndroidManifest.xml
 - kotlin+java
 - com.software.todo (androidTest)
 - com.software.todo (test)
 - com.software.todo.presentation.theme
 - Theme.kt
 - Type.kt
 - java (generated)
 - res
 - res (generated)
 - Gradle Scripts
 - build.gradle.kts (Project: To-Do_App)
 - build.gradle.kts (Module :app)**
 - proguard-rules.pro (ProGuard Rules for ":app")
 - gradle.properties (Project Properties)
 - gradle-wrapper.properties (Gradle Version)
 - libs.versions.toml (Version Catalog)
 - local.properties (SDK Location)
 - settings.gradle.kts (Project Settings)

Theme.kt | libs.versions.toml | build.gradle.kts (:app) ×

Gradle files have changed since last project sync. A project sync may be necessary for the IDE to work properly. Sync Now Ignore these changes

```
66 }
67
68 dependencies {
69
70     implementation(libs.androidx.core.ktx)
71     implementation(libs.androidx.lifecycle.runtime.ktx)
72     implementation(libs.androidx.activity.compose)
73
74     implementation(platform(libs.androidx.compose.bom))
75
76     implementation(libs.androidx.ui)
77     implementation(libs.androidx.ui.graphics)
78     implementation(libs.androidx.ui.tooling.preview)
79
80     implementation(libs.androidx.material3)
81     implementation(libs.androidx.animation)
82     implementation(libs.androidx.lifecycle.runtime)
83     implementation(libs.androidx.view.model)
84     implementation(libs.androidx.extend.icons)
85
86     debugImplementation(libs.androidx.ui.tooling)
87
88     implementation(libs.kotlin.coroutines.android)
89     implementation(libs.kotlin.coroutines.core)
90
91     implementation(libs.jetpack.room.runtime)
92     implementation(libs.jetpack.room.kotlin)
93     annotationProcessor(libs.jetpack.room.compiler)
94     ksp(libs.jetpack.room.compiler)
95
96 }
```



New

Add C++ to Module

Cut

Copy

Copy Path/Reference...

Paste

Find Usages

Find in Files...

Replace in Files...

Analyze

Refactor

Bookmarks

Reformat Code

Optimize Imports

Delete...

Open In

Local History

Git

Repair IDE on File

Reload from Disk

Compare With...

Mark Directory as

- Java Class
- Kotlin Class/File
- Android Resource File
- Android Resource Directory
- Sample Data Directory
- File
- Scratch File
- Package
- Image Asset
- Vector Asset
- module-info.java
- CMakeLists.txt
- Activity
- Fragment
- Folder
- Service
- UiComponent
- Automotive
- XML
- Wear
- TV
- AIDL
- Widget
- Google
- Compose
- Other
- Resource Bundle
- EditorConfig File
- Version Catalog

```
Pixel 3a API 35
app
Sync Now Ignore these changes
e.runtime.ktx
.compose)
.compose.bom))
ics)
ng.preview)
3)
n)
e.runtime)
el)
cons)
tooling)
.android)
.core)
ime)
in)
.compiler)
```

Android

- app
 - manifests
 - AndroidManifest.xml
 - kotlin+java
 - com.software.todo (androidTest)
 - com.software.todo (test)
 - com.software.todo.presentation
 - theme
 - java (generated)
 - res
 - res (generated)
 - Gradle Scripts

Theme.kt | libs.versions.toml | build.gradle.kts (:app)

Gradle files have changed since last project sync. A project sync may be necessary for the IDE to work properly. Sync Now Ignore these changes

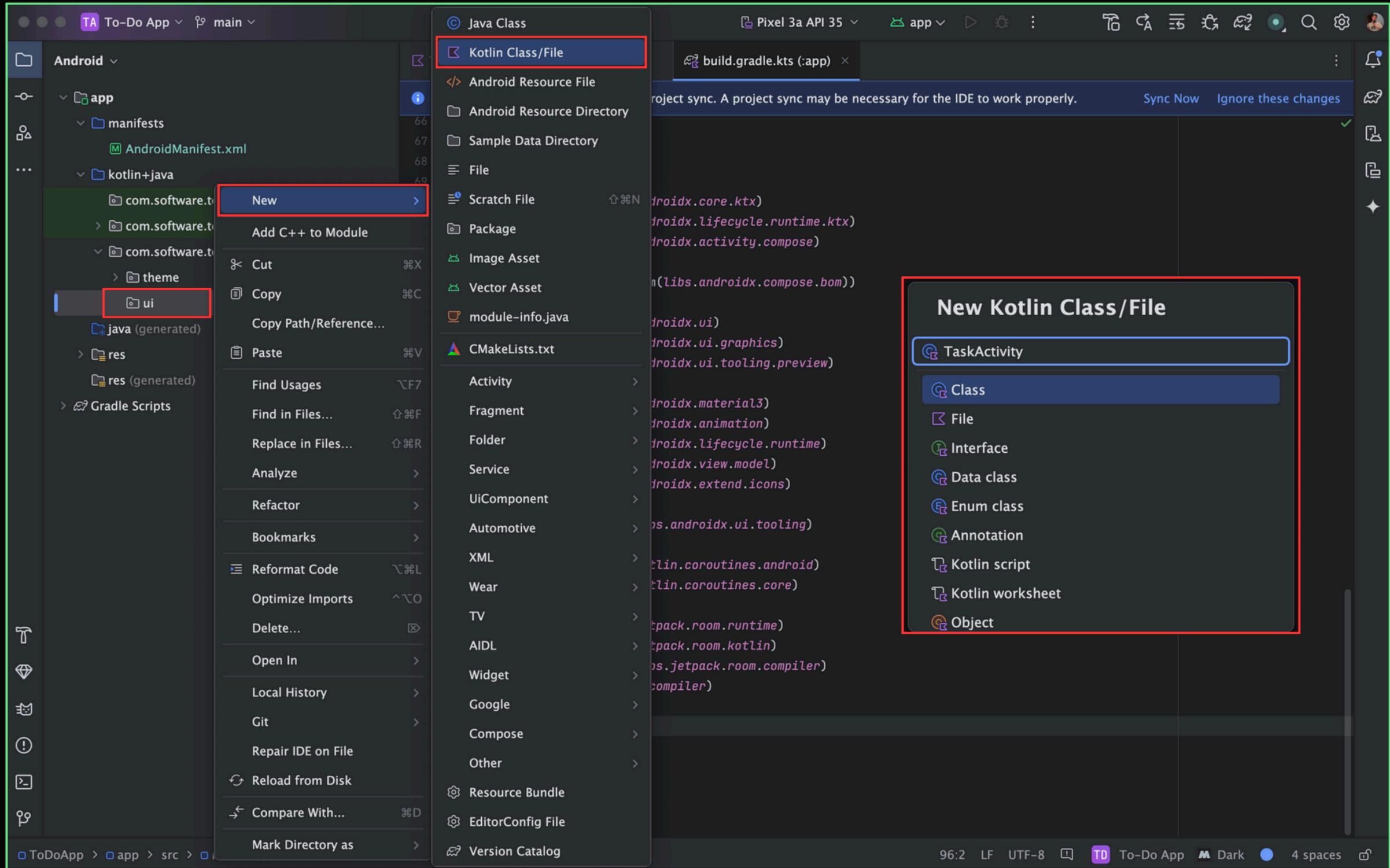
```

66 }
67
68 dependencies {
69
70     implementation(libs.androidx.core.ktx)
71     implementation(libs.androidx.lifecycle.runtime.ktx)
72     implementation(libs.androidx.activity.compose)
73
74     implementation(platform(libs.androidx.compose.bom))
75
76     implementation(libs.androidx.ui)
77     implementation(libs.androidx.ui.graphics)
78     implementation(libs.androidx.ui.tooling.preview)
79
80     implementation(libs.androidx.material3)
81     implementation(libs.androidx.navigation.compose)
82     implementation(libs.androidx.navigation.fragment.ktx)
83     implementation(libs.androidx.navigation.ui.ktx)
84     implementation(libs.androidx.appcompat)
85
86     debugImplementation(libs.androidx.ui.tooling)
87
88     implementation(libs.kotlin.coroutines.android)
89     implementation(libs.kotlin.coroutines.core)
90
91     implementation(libs.jetpack.room.runtime)
92     implementation(libs.jetpack.room.kotlin)
93     annotationProcessor(libs.jetpack.room.compiler)
94     ksp(libs.jetpack.room.compiler)
95
96 }

```

New Package

com.software.todo.presentation.ui



TA To-Do App main

Pixel 3a API 35 app

Android

- app
 - manifests
 - AndroidManifest.xml
 - kotlin+java
 - com.software.todo (androidTest)
 - com.software.todo (test)
 - com.software.todo.presentation
 - theme
 - ui
 - TaskActivity
 - java (generated)
 - res
 - res (generated)
 - Gradle Scripts

TaskActivity.kt

```
1 package com.software.todo.presentation.ui
2
3 class TaskActivity {}
4
```

ToDoApp > app > src > main > java > com > software > todo > presentation > ui > TaskActivity

3:21 LF UTF-8 To-Do App Dark 4 spaces

TA To-Do App develop

Pixel 3a API 35 app

Android

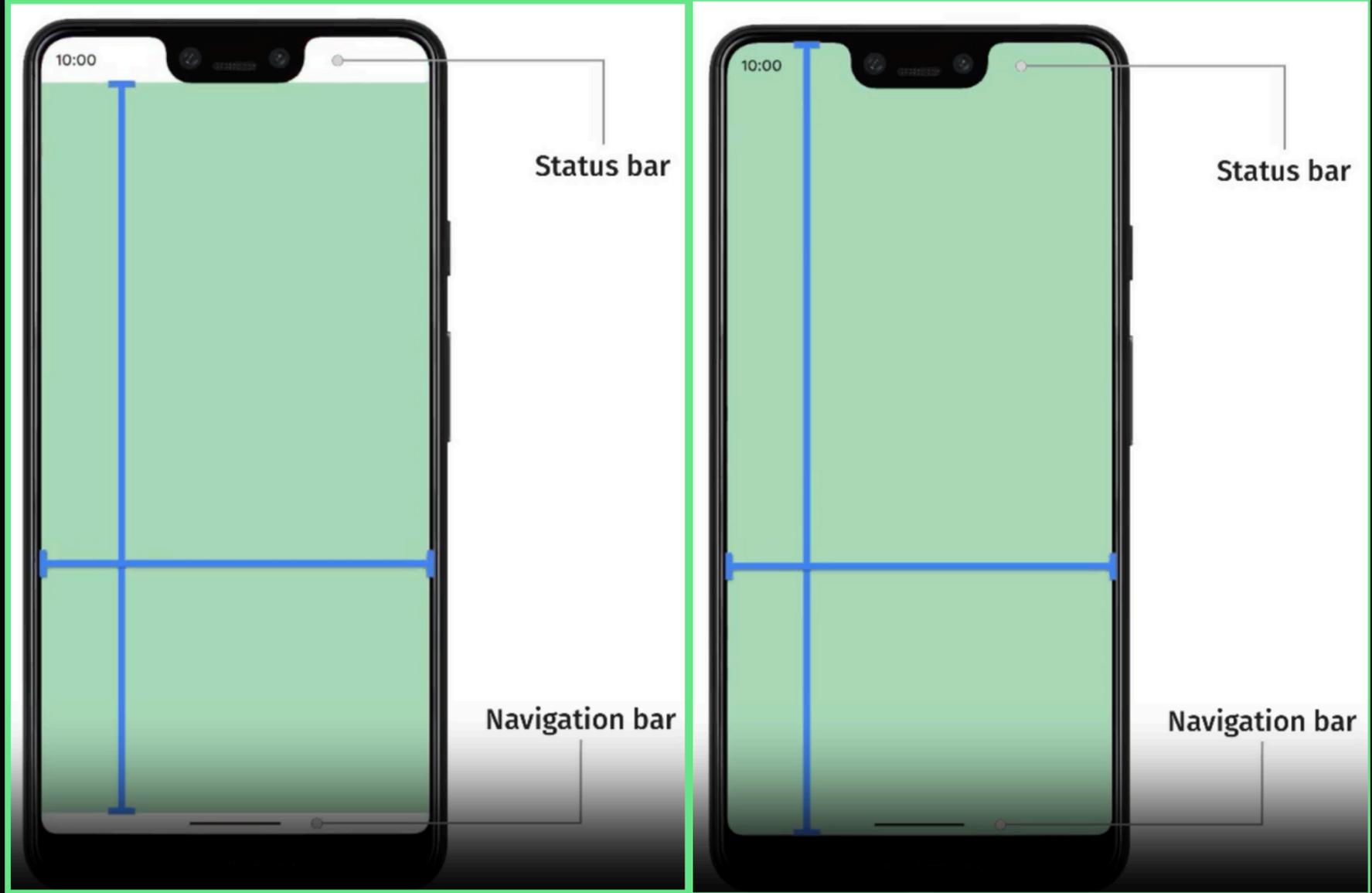
- app
 - manifests
 - kotlin+java
 - com.software.todo (androidTest)
 - com.software.todo (test)
 - com.software.todo.presentation
 - theme
 - ui
 - TaskActivity
 - res
 - res (generated)
 - Gradle Scripts

TaskActivity.kt

```
1 @file:Suppress( ...names: "SpellCheckingInspection")
2
3 package com.software.todo.presentation.ui
4
5 > import ...
10
11 class TaskActivity : ComponentActivity() {
12
13     override fun onCreate(savedInstanceState: Bundle?) {
14         super.onCreate(savedInstanceState)
15         enableEdgeToEdge() // 0 que significa isso?
16         setContent {
17             Theme {
18
19             }
20         }
21     }
22
23 }
```

ToDoApp > app > src > main > java > com > software > todo > presentation > ui > TaskActivity

23:2 LF UTF-8 TA To-Do App Dark 4 spaces



TA To-Do App main Pixel 3a API 35 app

Android

- app
 - manifests
 - AndroidManifest.xml
 - kotlin+java
 - com.software.todo (androidTest)
 - com.software.todo (test)
 - com.software.todo.presentation
 - theme
 - ui
 - TaskActivity
 - java (generated)
 - res
 - res (generated)
 - Gradle Scripts

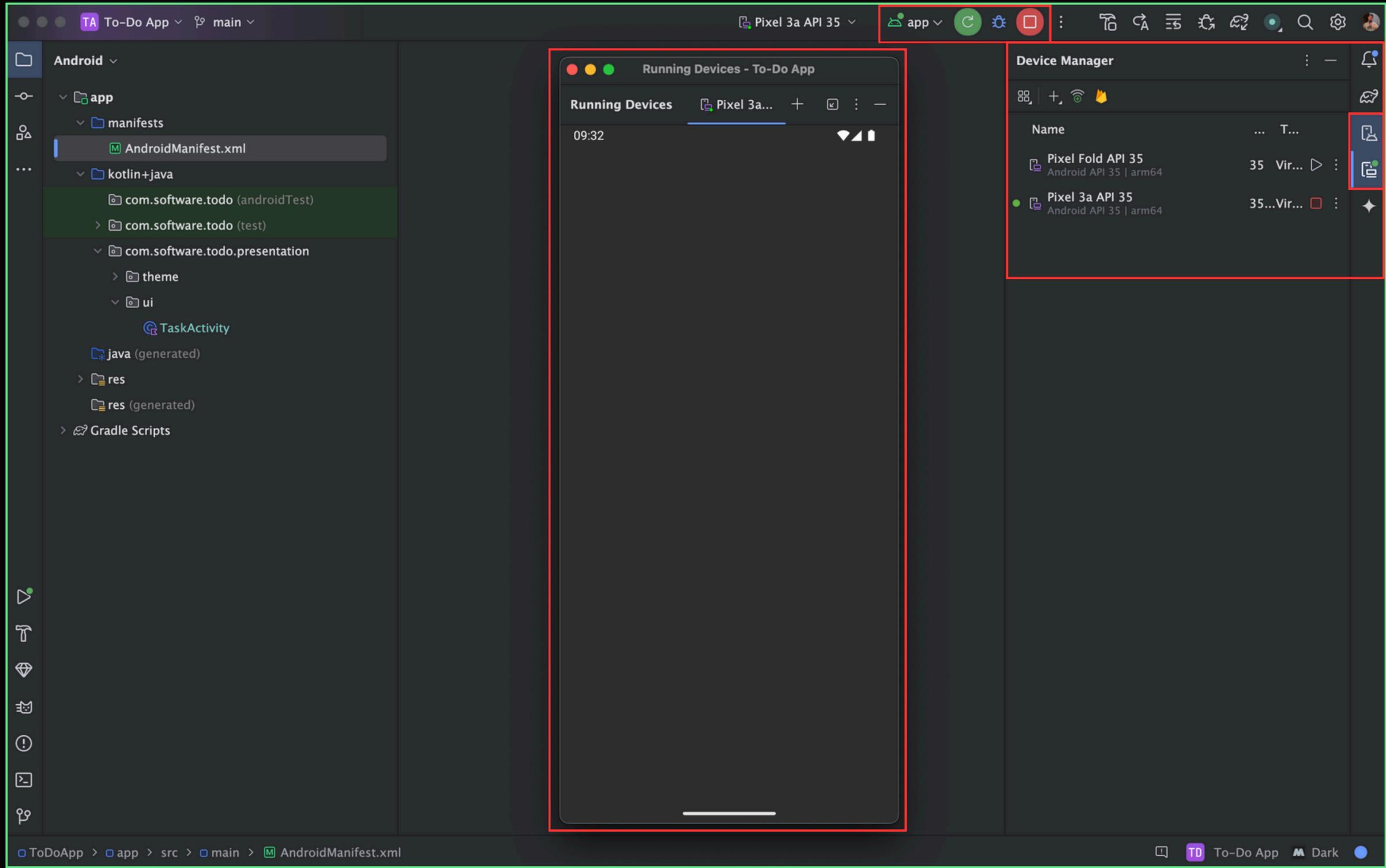
TaskActivity.kt AndroidManifest.xml Commit: TaskActivity.kt

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest
3   xmlns:android="http://schemas.android.com/apk/res/android"
4   xmlns:tools="http://schemas.android.com/tools">
5
6   <application
7     android:allowBackup="true"
8     android:dataExtractionRules="@xml/data_extraction_rules"
9     android:fullBackupContent="@xml/backup_rules"
10    android:icon="@mipmap/ic_launcher"
11    android:label="To-Do App"
12    android:roundIcon="@mipmap/ic_launcher_round"
13    android:supportsRtl="true"
14    android:theme="@style/Theme.ToDoApp"
15    tools:targetApi="31">
16
17    <activity
18      android:name=".presentation.ui.TaskActivity"
19      android:exported="true"
20      android:theme="@style/Theme.ToDoApp">
21
22      <intent-filter>
23        <action android:name="android.intent.action.MAIN" />
24        <category android:name="android.intent.category.LAUNCHER" />
25      </intent-filter>
26
27    </activity>
28
29  </application>
30
31 </manifest>
```

manifest > application

Text Merged Manifest

ToDoApp > app > src > main > AndroidManifest.xml 29:19 LF UTF-8 To-Do App Dark 4 spaces



TA To-Do App main

Pixel 3a API 35 app

Android

- app
 - manifests
 - kotlin+java
 - com.software.todo
 - data
 - TaskDao
 - domain
 - Task**
 - presentation
 - com.software.todo (androidTest)
 - com.software.todo (test)
 - java (generated)
 - res
 - res (generated)
 - Gradle Scripts

Task.kt

```
1 @file:Suppress( ...names: "SpellCheckingInspection")
2
3 package com.software.todo.domain
4
5 > import ...
6
7
8
9 @Entity // @Entity define esta classe como uma entidade de tabela no bd (banco de dados)
10 data class Task(
11
12     // @ColumnInfo especifica o nome da coluna "title" para o campo `title` no bd
13     @ColumnInfo( name: "title")
14     val title: String = "",
15
16     // @ColumnInfo especifica o nome da coluna "description" para o campo `description` no bd
17     @ColumnInfo( name: "description")
18     val description: String = "",
19
20     // @ColumnInfo especifica o nome da coluna "checked" para o campo `checked` no bd
21     @ColumnInfo( name: "checked")
22     val checked: Boolean = false,
23
24     // @PrimaryKey define `id` como a chave primária, com a geração automática de valores
25     @PrimaryKey( autoGenerate: true)
26     val id: Int = 0
27
28 )
```

ToDoApp > app > src > main > java > com > software > todo > domain > Task

28:2 LF UTF-8 TA To-Do App Dark 4 spaces

Android Studio interface showing the development of a To-Do App. The left sidebar displays the project structure, with the `TaskDao` class highlighted in the `data` package. The main editor displays the Kotlin code for `TaskDao.kt`.

```
1 @file:Suppress( ...names: "SpellCheckingInspection")
2
3 package com.software.todo.data
4
5 > import ...
6
7
8
9
10
11
12
13 @Dao // @Dao define esta interface como um DAO (Data Access Object) para a entidade `Task`
14 interface TaskDao {
15
16     // Consulta para buscar todas as tarefas da tabela `task`
17     @Query("SELECT * FROM task")
18     fun getAllTasks(): Flow<List<Task>> // Retorna uma lista de tarefas em um fluxo reativo
19
20     // Anotação para inserir uma nova tarefa na tabela `task`
21     @Insert
22     fun insert(task: Task)
23
24     // Anotação para atualizar uma tarefa existente na tabela `task`
25     @Update
26     fun update(task: Task)
27
28     // Anotação para deletar uma tarefa existente da tabela `task`
29     @Delete
30     fun delete(task: Task)
31
32 }
```

The bottom status bar shows the file path: `ToDoApp > app > src > main > java > com > software > todo > domain`. The status bar also displays the current line and column (32:2), encoding (LF UTF-8), and other settings like "Dark" theme and "4 spaces" indentation.

TA To-Do App main Pixel 3a API 35 app

Android app manifests kotlin+java com.software.todo data TaskDao TaskDatabase domain Task presentation com.software.todo (androidTest) com.software.todo (test) java (generated) res res (generated) Gradle Scripts

TaskDatabase.kt

```
1 @file:Suppress(...names: "SpellCheckingInspection")
2
3 package com.software.todo.data
4
5 > import ...
10
11 // @Database define esta classe como um banco de dados Room com a entidade `Task` e a versão 1
12 @Database(entities = [Task::class], version = 1)
13 abstract class TaskDatabase : RoomDatabase() {
14
15     // Fornece acesso ao DAO `TaskDao` para operações no banco de dados
16     abstract fun dao(): TaskDao
17
18     // No Kotlin, o companion object é uma maneira de definir membros estáticos dentro
19     // de uma classe. Isso significa que os métodos e propriedades definidos
20     // dentro do companion object pertencem à classe em si,
21     // e não a uma instância específica dessa classe.
22     companion object {
23
24         // Função para obter uma instância de `TaskDao` utilizando um singleton do banco de dados
25         fun getInstance(context: Context): TaskDao {
26             return Room.databaseBuilder(context, TaskDatabase::class.java, name: "tasks")
27                 .fallbackToDestructiveMigration() // Permite migrações destrutivas
28                 .build() // Constroi o banco de dados
29                 .dao() // Retorna o DAO `TaskDao` para interagir com o banco de dados
30         }
31     }
32 }
33
34 }
```

ToDoApp > app > src > main > java > com > software > todo > domain 34:2 LF UTF-8 TA To-Do App Dark 4 spaces

TA To-Do App main Pixel 3a API 35 app

Android

- app
 - manifests
 - kotlin+java
 - com.software.todo
 - data
 - TaskDao
 - TaskDatabase
 - domain
 - Task
 - presentation
 - theme
 - ui
 - TaskActivity
 - TaskViewModel**
 - com.software.todo (androidTest)
 - com.software.todo (test)
 - java (generated)
 - res
 - res (generated)
 - Gradle Scripts

TaskViewModel.kt

```
1 @file:Suppress( ...names: "SpellCheckingInspection")
2
3 package com.software.todo.presentation.ui
4
5 > import ...
12
13 // Ao herdar a classe ViewModel, estamos criando uma classe especializada para manter e gerenciar
14 // os dados de UI de maneira isolada das Activities
15 class TaskViewModel(
16     private val dao: TaskDao // Injeta uma instância do DAO para acessar as operações de bd
17 ) : ViewModel() {
18
19     // Executa a inserção da tarefa em um escopo de `viewModelScope` para que a coroutine
20     // seja cancelada ao final do ciclo de vida do `ViewModel`
21     fun insert(task: Task) = viewModelScope.launch {
22         withContext(Dispatchers.IO) { // Executa a atualização da tarefa de forma assíncrona
23             dao.insert(task) // Insere uma nova tarefa no banco de dados
24         }
25     }
26
27     // Executa a atualização da tarefa em um escopo de `viewModelScope` para que a coroutine
28     // seja cancelada ao final do ciclo de vida do `ViewModel`
29     fun update(task: Task) = viewModelScope.launch {
30         withContext(Dispatchers.IO) { // Executa a atualização da tarefa de forma assíncrona
31             dao.update(task) // Atualiza a tarefa no banco de dados
32         }
33     }
34
35     // Executa a exclusão da tarefa em um escopo de `viewModelScope` para que a coroutine
36     // seja cancelada ao final do ciclo de vida do `ViewModel`
37     fun delete(task: Task) = viewModelScope.launch {
38         withContext(Dispatchers.IO) { // Executa a atualização da tarefa de forma assíncrona
39             dao.delete(task) // Deleta a tarefa do banco de dados
40         }
41     }
42
43 }
```

ToDoApp > app > src > main > java > com > software > todo > presentation > ui > TaskViewModel 18:1 LF UTF-8 TA To-Do App Dark 4 spaces

TA To-Do App main Pixel 3a API 35 app

Android

- app
 - manifests
 - kotlin+java
 - com.software.todo
 - data
 - TaskDao
 - TaskDatabase
 - domain
 - Task
 - presentation
 - theme
 - ui
 - TaskActivity
 - TaskViewModel**
 - com.software.todo (androidTest)
 - com.software.todo (test)
 - java (generated)
 - res
 - res (generated)
 - Gradle Scripts

TaskViewModel.kt

```
1 @file:Suppress(...names: "SpellCheckingInspection")
2
3 package com.software.todo.presentation.ui
4
5 > import ...
16
17 // Ao herdar a classe ViewModel, estamos criando uma classe especializada para manter e gerenciar
18 // os dados de UI de maneira isolada das Activities
19 class TaskViewModel(
20     private val dao: TaskDao // Injeta uma instância do DAO para acessar as operações de bd
21 ) : ViewModel() {
22
23     // Variável pública que obtém as tarefas cadastradas banco de dados
24     val tasks: StateFlow<List<Task>> = dao
25         .getAllTasks()
26         .asStateFlow() // Converte o `Flow` em um `StateFlow`
27
28     /* ----- */
29
30     // Extensão para converter um `Flow` em `StateFlow` com um
31     // escopo de `viewModelScope`. `stateIn` inicia o fluxo de forma preguiçosa
32     // e usa uma lista vazia como valor inicial
33     private fun <T> Flow<List<T>>.asStateFlow(): StateFlow<List<T>> {
34         return stateIn(viewModelScope, SharingStarted.Lazily, emptyList())
35     }
36
37     /* ----- */
38
39     // Executa a inserção da tarefa em um escopo de `viewModelScope` para que a coroutine
40     // seja cancelada ao final do ciclo de vida do `ViewModel`
41     fun insert(task: Task) = viewModelScope.launch {
42         withContext(Dispatchers.IO) { // Executa a atualização da tarefa de forma assíncrona
43             dao.insert(task) // Insere uma nova tarefa no banco de dados
44         }
45     }
46
47     // Executa a atualização da tarefa em um escopo de `viewModelScope` para que a coroutine
48     // seja cancelada ao final do ciclo de vida do `ViewModel`
```

ToDoApp > app > src > main > java > com > software > todo > presentation > ui > TaskViewModel 38:1 LF UTF-8 TA To-Do App Dark 4 spaces

TA To-Do App main

Pixel 3a API 35 app

Android

- app
 - manifests
 - kotlin+java
 - com.software.todo
 - data
 - domain
 - presentation
 - theme
 - ui
 - TaskActivity**
 - TaskViewModel

- com.software.todo (androidTest)
- com.software.todo (test)
- java (generated)
- res
- res (generated)
- Gradle Scripts

TaskActivity.kt

```
3
4 package com.software.todo.presentation.ui
5
6 > import ...
7
8
9
10
11
12
13
14
15
16
17
18 <> class TaskActivity : ComponentActivity() {
19
20     // Instancia o ViewModel utilizando a função viewModels, que permite
21     // a criação de ViewModel com um ViewModelProvider
22     // personalizado.
23     //
24     // ViewModelFactory fornece uma maneira de inicializar o
25     // TaskViewModel com a instância do
26     // TaskDatabase.
27     private val viewModel by viewModels<TaskViewModel> {
28         viewModelFactory { // Bloco para criar a inicialização do ViewModel
29             initializer { // Bloco para inicializar o ViewModel
30                 TaskViewModel( // Instância do ViewModel (igual uma class)
31                     TaskDatabase.getInstance(applicationContext) // Instância do banco de dados
32                 )
33             }
34         }
35     }
36
37     override fun onCreate(savedInstanceState: Bundle?) {
38         super.onCreate(savedInstanceState)
39         enableEdgeToEdge()
40         setContent {
41             Theme {
42
43             }
44         }
45     }
46
47 }
```

ToDoApp > app > src > main > java > com > software > todo > presentation > ui > TaskActivity

47:2 LF UTF-8 TA To-Do App Dark 4 spaces

TA To-Do App main

Pixel 3a API 35 app

Android

- app
 - manifests
 - kotlin+java
 - com.software.todo
 - data
 - domain
 - presentation
 - theme
 - ui
 - TaskActivity
 - TaskViewModel
 - com.software.todo (androidTest)
 - com.software.todo (test)
 - java (generated)
 - res
 - res (generated)
 - Gradle Scripts

TaskActivity.kt

```
34 class TaskActivity : AppCompatActivity() {
66     @Composable // Indica para o compilador que essa função é um layout
67     private fun TaskListScreen(items: List<Task>, padding: PaddingValues) {
68         LazyColumn(contentPadding = padding) { // Visualização de itens (layout) na vertical
69             items( // Scopo para obter os itens e transformar em Composable (layout)
70                 items = items,
71                 key = { task -> task.id }
72             ) { task -> // Lambda contendo o objeto tarefa
73                 TaskItem(task = task) // Nossa tarefa. Para cada item na lista, essa função é chamada
74             }
75         }
76     }
77
78     @Composable
79     private fun TaskEmptyScreen(padding: PaddingValues) {...}
82
83     @Composable
84     private fun TaskItem(task: Task) {
85         Card( // Layout com bordas e sombra (efeito flutuante)
86             modifier = Modifier
87                 .padding(horizontal = 16.dp, vertical = 8.dp) // Espaçamentos
88                 .combinedClickable( // Clique longo (pressiona e segura) e clique (clica e solta)
89                     onLongClick = { viewModel.delete(task) }, // Deleta a tarefa
90                     onClick = {
91
92                 }
93             )
94         ) {
95             Row {...}
96         }
97     }
123
124
125
126 }
```

ToDoApp > app > src > main > java > com > software > todo > presentation > ui > TaskActivity

126:2 LF UTF-8 TA To-Do App Dark 4 spaces

TA To-Do App main

Pixel 3a API 35 app

Android

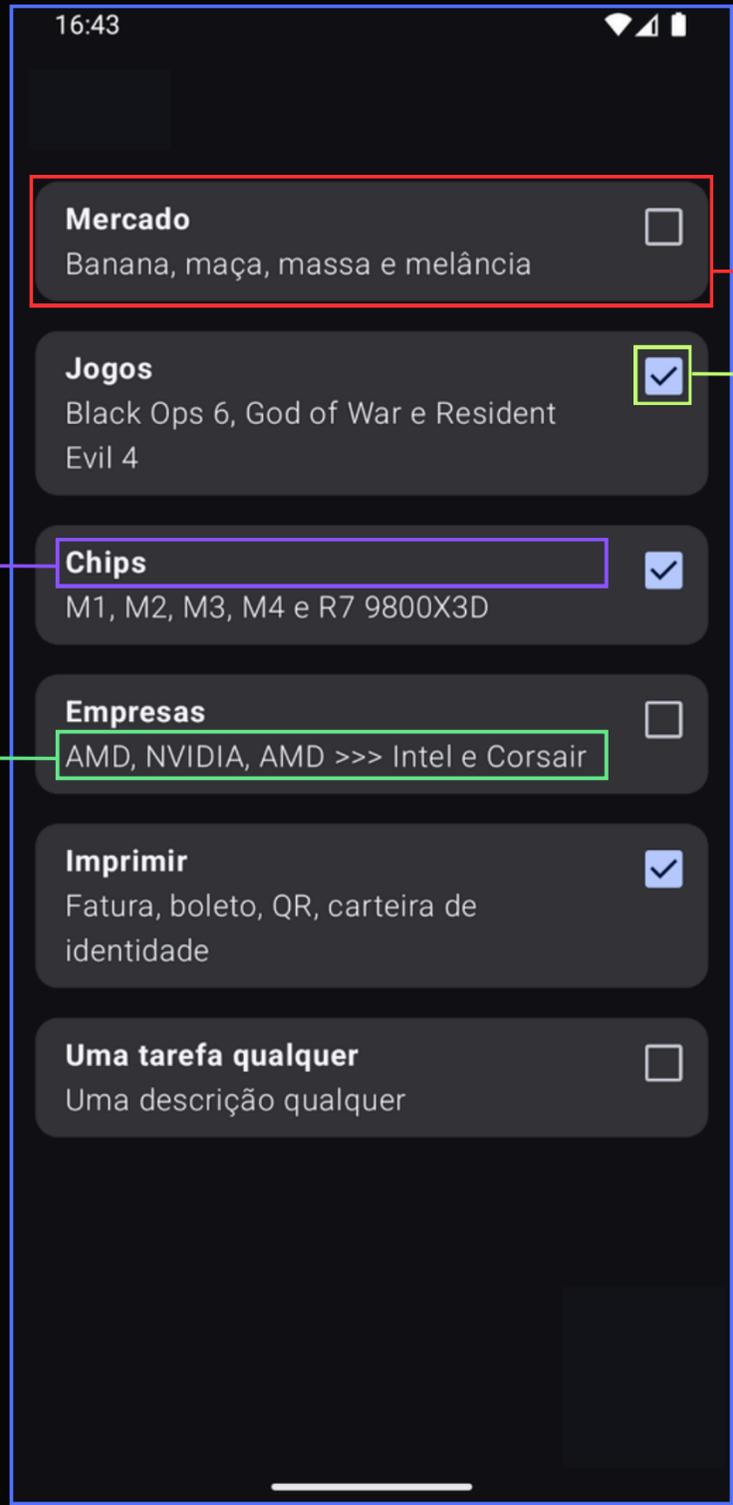
- app
 - manifests
 - kotlin+java
 - com.software.todo
 - data
 - domain
 - presentation
 - theme
 - ui
 - TaskActivity
 - TaskViewModel
 - com.software.todo (androidTest)
 - com.software.todo (test)
 - java (generated)
 - res
 - res (generated)
 - Gradle Scripts

TaskActivity.kt

```
34 class TaskActivity : AppCompatActivity() {
84     private fun TaskItem(task: Task) {
94     }
95     Row { // Tudo o que estiver dentro do lambda vai ser adicionado na horizontal (linha)
96     Column( // Tudo o que estiver dentro do lambda vai ser adicionado na vertical (coluna)
97         modifier = Modifier
98             .padding(horizontal = 16.dp, vertical = 8.dp) // Espaçamentos
99             .weight(weight = 1F) // Significa que deve preencher a
100                 // tela na horizontal levando em consideração os outros layouts
101     ) {
102     Text( // Layout para desenhar um texto
103         modifier = Modifier.fillMaxWidth(), // Indica ao compose que esse componte vai
104             // preencher toda a tela na horizontal
105
106         fontWeight = FontWeight.Bold, // Formato de fonte em negrito
107         text = task.title // Texto que queremos escrever (título da tarefa)
108     )
109
110     Text( // Layout para desenhar um texto
111         modifier = Modifier.fillMaxWidth(), // Indica ao compose que esse componte vai
112             // preencher toda a tela na horizontal
113
114         fontWeight = FontWeight.Light, // Formato de fonte mais fina
115         text = task.description // Texto que queremos escrever (descrição da tarefa)
116     )
117
118     Checkbox( // Caixa com a possibilidade de marcar ou desmarcar
119         checked = task.checked, // Recebe um valor do tipo [Boolean] (true ou false)
120         onCheckedChange = { newValue -> // Lambda para capturar a ação de quando ocorre o click na caixa
121             viewModel.update(
122                 task.copy(checked = newValue)
123             )
124         }
125     )
126     }
127     }
128 }
```

ToDoApp > app > src > main > java > com > software > todo > presentation > ui > TaskActivity > TaskItem

94:12 LF UTF-8 TA To-Do App Dark 4 spaces



Card

CheckBox

Text (task.title)

Text (task.description)

LazyColumn

TA To-Do App main Pixel 3a API 35 app

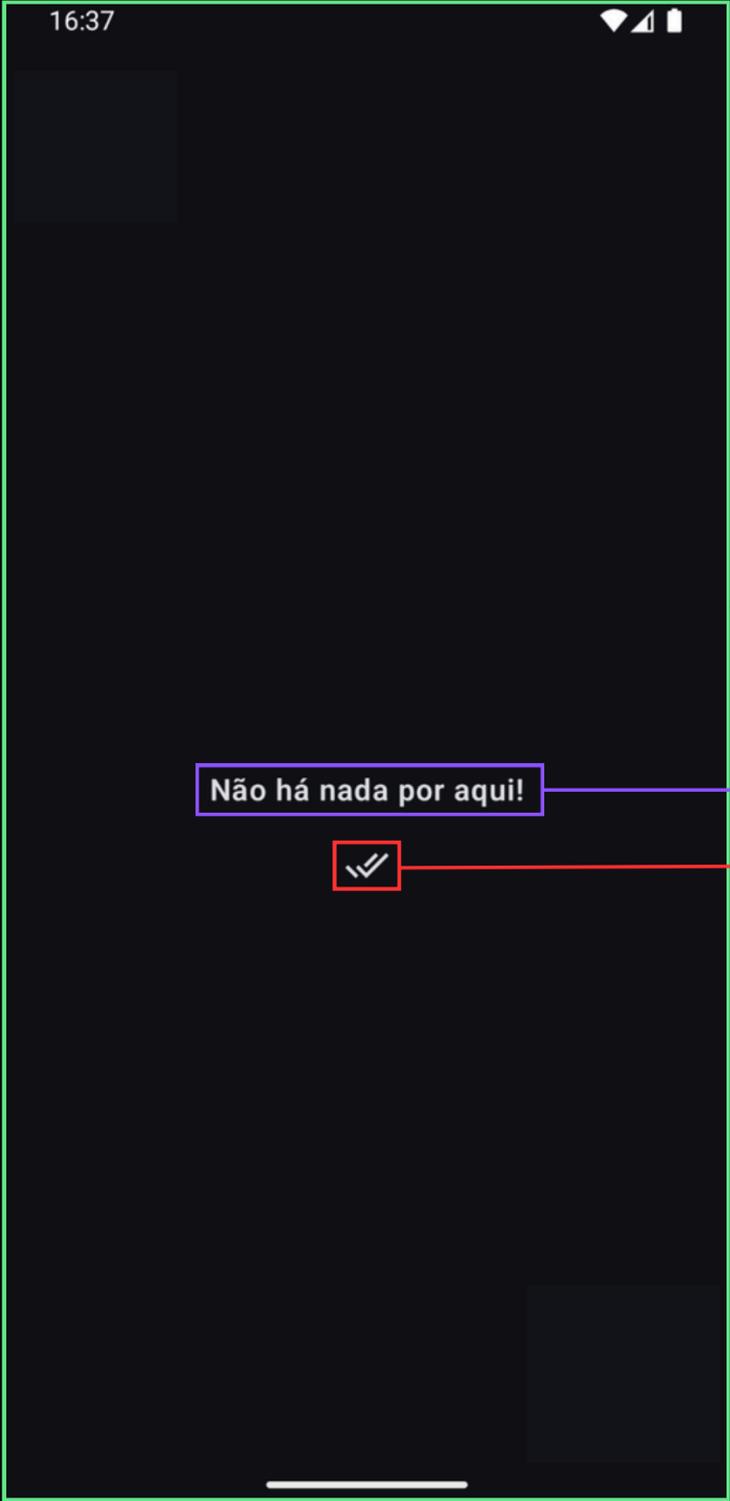
Android

- app
 - manifests
 - kotlin+java
 - com.software.todo
 - data
 - domain
 - presentation
 - theme
 - ui
 - TaskActivity
 - TaskViewModel

- com.software.todo (androidTest)
- com.software.todo (test)
- java (generated)
- res
- res (generated)
- Gradle Scripts

```
40 class TaskActivity : AppCompatActivity() {
68
69     /* ----- */
70
71     @Composable // Indica para o compilador que essa função é um layout
72     private fun TaskListScreen(items: List<Task>, padding: PaddingValues) {...}
82
83     @Composable
84     private fun TaskEmptyScreen(padding: PaddingValues) {
85         Column( //
86             modifier = Modifier
87                 .padding(paddingValues = padding) // Espaçamentos do Scaffold (AppBar)
88                 .fillMaxSize(), // Signica que esse componente vai preencher a tela inteira (Altura e Largura)
89             horizontalAlignment = Alignment.CenterHorizontally, // Alinhados horizontalmente no centro
90             verticalArrangement = Arrangement.Center // Alinhados verticalmente no centro
91         ) {
92             Text(
93                 modifier = Modifier.padding(vertical = 16.dp), // Espaçamentos
94                 fontWeight = FontWeight.SemiBold, // Formato de fonte em semi negrito
95                 text = "Não há nada por aqui!" // Texto que queremos escrever
96             )
97
98             Icon( // Ícone para desenhar uma imagem
99                 imageVector = Icons.Default.DoneAll, // Imagem a qual queremos
100                 contentDescription = "Sem tarefas!" // Recurso de acessibilidade
101             )
102         }
103     }
104
105     @Composable
106     private fun TaskItem(task: Task) {...}
151
152 }
```

ToDoApp > app > src > main > java > com > software > todo > presentation > ui > TaskActivity 152:2 LF UTF-8 TA To-Do App Dark 4 spaces



16:37



Não há nada por aqui!



Column

Text

Icon

TA To-Do App main

Pixel 3a API 35 app

Android

- app
 - manifests
 - kotlin+java
 - com.software.todo
 - data
 - domain
 - presentation
 - theme
 - ui
 - TaskActivity
 - TaskViewModel
 - com.software.todo (androidTest)
 - com.software.todo (test)
 - java (generated)
 - res
 - res (generated)
 - Gradle Scripts

TaskActivity.kt

```
44 class TaskActivity : AppCompatActivity() {
50     // ViewModelFactory fornece uma maneira de inicializar o
51     // TaskViewModel com a instância do
52     // TaskDatabase.
53     private val viewModel by viewModel<TaskViewModel> {
54         viewModelFactory { // Bloco para criar a inicialização do ViewModel
55             initializer { // Bloco para inicializar o ViewModel
56                 TaskViewModel( // Instância do ViewModel (igual uma class)
57                     TaskDatabase.getInstance(applicationContext) // Instância do banco de dados
58                 )
59             }
60         }
61     }
62
63     override fun onCreate(savedInstanceState: Bundle?) {
64         super.onCreate(savedInstanceState)
65         enableEdgeToEdge()
66         setContent {
67             Theme {
68
69             }
70         }
71     }
72
73     /* ----- */
74
75     @Composable
76     private fun ScreenScreen() {...}
77
78
79
80
81     @Composable
82     private fun TaskActionButton() {...}
83
84
85
86     @Composable
87     private fun TaskTopBar(scrollBehavior: TopAppBarScrollBehavior) {...}
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105     /* ----- */
106 }
```

ToDoApp > app > src > main > java > com > software > todo > presentation > ui > TaskActivity > ScreenScreen 78:6 LF UTF-8 TA To-Do App Dark 4 spaces

TA To-Do App main Pixel 3a API 35 app

Android

- app
 - manifests
 - kotlin+java
 - com.software.todo
 - data
 - domain
 - presentation
 - theme
 - ui
 - TaskActivity
 - TaskViewModel

```
44 class TaskActivity : AppCompatActivity() {
53     private val viewModel by viewModels<TaskViewModel> {
54         viewModelFactory { // Bloco para criar a inicialização do ViewModel
59             }
60     }
61 }
62
63 override fun onCreate(savedInstanceState: Bundle?) {
64     super.onCreate(savedInstanceState)
65     enableEdgeToEdge()
66     setContent {
67         Theme {
68             }
69     }
70 }
71
72
73 /* ----- */
74
75 @Composable
76 private fun ScreenScreen() {...}
79
80 @Composable
81 private fun TaskActionButton() {...}
93
94 @Composable
95 private fun TaskTopBar(scrollBehavior: TopAppBarScrollBehavior) {
96     TopAppBar( // Barra de ações com título
97         scrollBehavior = scrollBehavior, title = { // Lambda para desenhar o layout 'Text'
98             Text(
99                 text = "Tasks" // Texto que queremos escrever
100             )
101         }
102     )
103 }
104
105 /* ----- */
```

ToDoApp > app > src > main > java > com > software > todo > presentation > ui > TaskActivity 104:1 LF UTF-8 TA To-Do App Dark 4 spaces

Tarefas

TopBar

A diagram on a black background. On the left, there is a dark gray rectangular box with a thin red border. Inside the box, the word "Tarefas" is written in white. To the right of the box, the word "TopBar" is written in red. A thin red horizontal line connects the right side of the box to the "T" in "TopBar".

TA To-Do App main

Pixel 3a API 35 app

Android

- app
 - manifests
 - kotlin+java
 - com.software.todo
 - data
 - domain
 - presentation
 - theme
 - ui
 - TaskActivity
 - TaskViewModel

com.software.todo (androidTest)

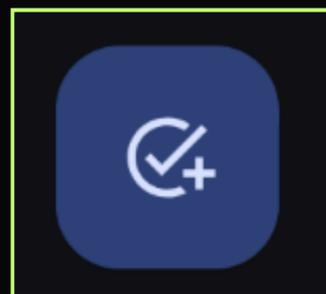
- com.software.todo (test)
- java (generated)
- res
- res (generated)
- Gradle Scripts

TaskActivity.kt

```
44 class TaskActivity : AppCompatActivity() {
55     private val viewModel by viewModel<TaskViewModel>()
61 }
62
63 override fun onCreate(savedInstanceState: Bundle?) {
64     super.onCreate(savedInstanceState)
65     enableEdgeToEdge()
66     setContent {
67         Theme {
68
69         }
70     }
71 }
72
73 /* ----- */
74
75 @Composable
76 private fun ScreenScreen() {...}
77
78
79
80 @Composable
81 private fun TaskActionButton() {
82     FloatingActionButton( // Botão flutuante com bordas arredondadas
83         onClick = { // Lambda contendo a ação de click do usuário
84
85         }
86     ) {
87         Icon( // Layout para desenhar uma imagem
88             imageVector = Icons.Default.AddTask, // Imagem a qual queremos desenhar
89             contentDescription = "Nova tarefa" // Recurso de acessibilidade
90         )
91     }
92 }
93
94 @Composable
95 private fun TaskTopBar(scrollBehavior: TopAppBarScrollBehavior) {...}
104
105 /* ----- */
106
```

ToDoApp > app > src > main > java > com > software > todo > presentation > ui > TaskActivity

104:1 LF UTF-8 TA To-Do App Dark 4 spaces



FloatingActionButton

TA To-Do App main

Pixel 3a API 35 app

Android

- app
 - manifests
 - kotlin+java
 - com.software.todo
 - data
 - domain
 - presentation
 - theme
 - ui
 - TaskActivity
 - TaskViewModel
 - com.software.todo (androidTest)
 - com.software.todo (test)
 - java (generated)
 - res
 - res (generated)
 - Gradle Scripts

TaskActivity.kt

```
47 class TaskActivity : AppCompatActivity() {
75
76     /* ----- */
77
78     @Composable
79     private fun ScreenScreen() {
80         val scrollBehavior = TopAppBarDefaults.pinnedScrollBehavior()
81
82         Scaffold(
83             modifier = Modifier.nestedScroll(connection = scrollBehavior.nestedScrollConnection),
84             floatingActionButton = { TaskActionButton() },
85             topBar = { TaskTopBar(scrollBehavior) }
86         ) { paddingValues ->
87             |
88         }
89     }
90
91     @Composable
92     private fun TaskActionButton() {...}
93
94
95     @Composable
96     private fun TaskTopBar(scrollBehavior: TopAppBarScrollBehavior) {...}
97
98     /* ----- */
99
100     @Composable // Indica para o compilador que essa função é um layout
101     private fun TaskListScreen(items: List<Task>, padding: PaddingValues) {...}
102
103
104     @Composable
105     private fun TaskEmptyScreen(padding: PaddingValues) {...}
106
107
108     @Composable
109     private fun TaskItem(task: Task) {...}
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199 }
```

ToDoApp > app > src > main > java > com > software > todo > presentation > ui > TaskActivity > ScreenScreen 87:13 LF UTF-8 TA To-Do App Dark 4 spaces

TA To-Do App main

Pixel 3a API 35 app

Android

- app
 - manifests
 - kotlin+java
 - com.software.todo
 - data
 - domain
 - presentation
 - theme
 - ui
 - TaskActivity
 - TaskViewModel
 - com.software.todo (androidTest)
 - com.software.todo (test)
 - java (generated)
 - res
 - res (generated)
 - Gradle Scripts

TaskActivity.kt

```
49 class TaskActivity : AppCompatActivity() {
78     /* ----- */
79
80     @Composable
81     private fun ScreenScreen() {
82         val scrollBehavior = TopAppBarDefaults.pinnedScrollBehavior() // Observa sempre que o
83                                     // usuário rola a lista de tarefas
84
85         Scaffold( // Scaffold tem a estrutura base para criarmos o nosso layout
86             modifier = Modifier.nestedScroll(connection = scrollBehavior.nestedScrollConnection), // Add o observer
87             floatingActionButton = { TaskActionButton() }, // Lambda para incluir o botão flutuante
88             topBar = { TaskTopBar(scrollBehavior) } // Lambda para incluir a barra de ações
89         ) { paddingValues ->
90             val items by viewModel.tasks.collectAsState() // Busca as tarefas cadastradas no bd
91         }
92     }
93
94     @Composable
95     private fun TaskActionButton() {...}
96
97
98
99
100
101
102
103
104
105
106
107
108     @Composable
109     private fun TaskTopBar(scrollBehavior: TopAppBarScrollBehavior) {...}
110
111
112
113
114
115
116
117
118     /* ----- */
119
120
121     @Composable // Indica para o compilador que essa função é um layout
122     private fun TaskListScreen(items: List<Task>, padding: PaddingValues) {...}
123
124
125
126
127
128
129
130
131     @Composable
132     private fun TaskEmptyScreen(padding: PaddingValues) {...}
133
134
135
136
137
138
139
140
141
142
143
144
145     @Composable
146     private fun TaskItem(task: Task) {...}
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202 }
```

ToDoApp > app > src > main > java > com > software > todo > presentation > ui > TaskActivity

79:1 LF UTF-8 TA To-Do App Dark 4 spaces

TA To-Do App main

Pixel 3a API 35 app

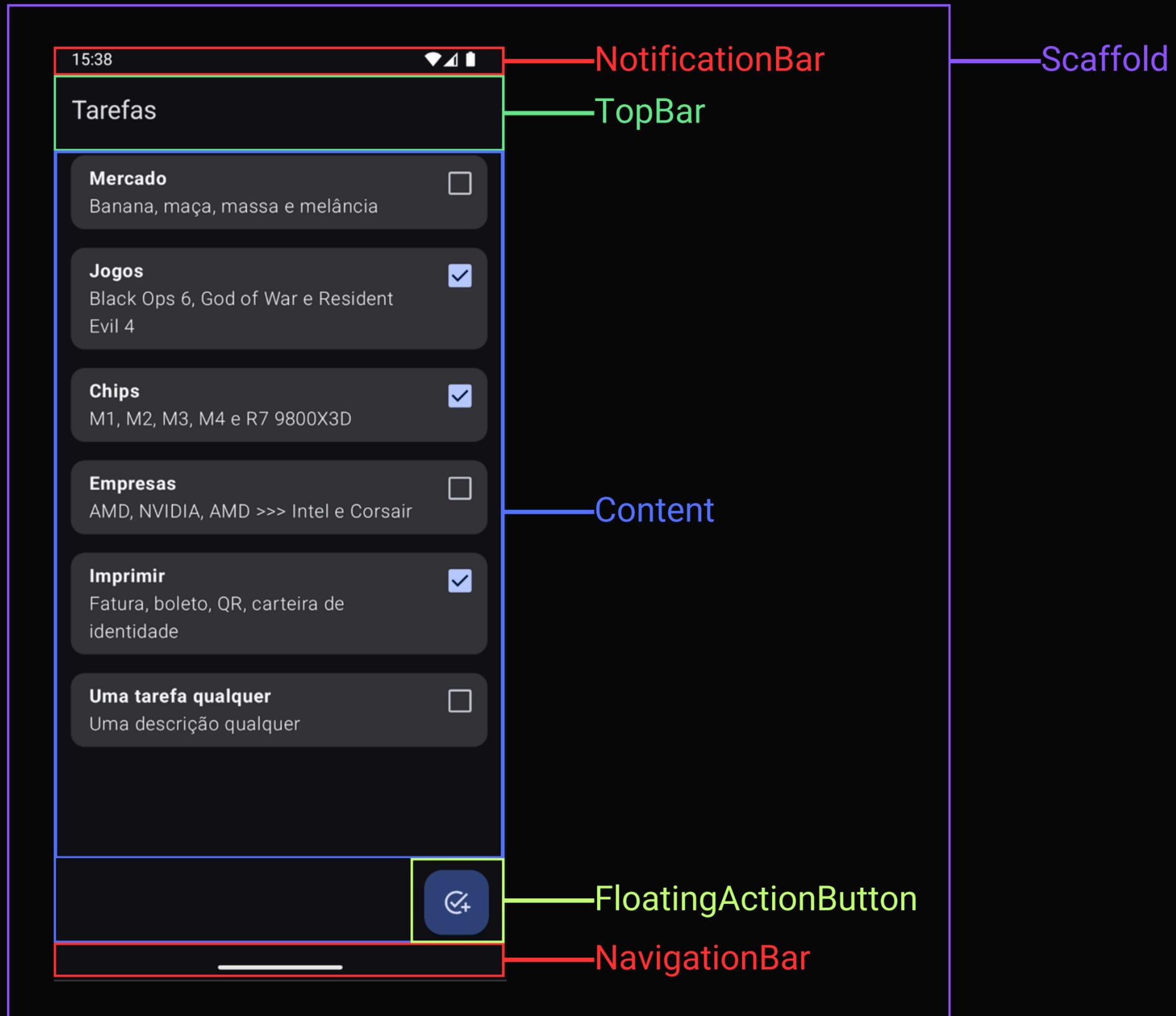
Android

- app
 - manifests
 - kotlin+java
 - com.software.todo
 - data
 - domain
 - presentation
 - theme
 - ui
 - TaskActivity
 - TaskViewModel
 - com.software.todo (androidTest)
 - com.software.todo (test)
 - java (generated)
 - res
 - res (generated)
 - Gradle Scripts

TaskActivity.kt

```
50 class TaskActivity : AppCompatActivity() {
78
79     /* ----- */
80
81     @Composable
82     private fun ScreenScreen() {
83         val scrollBehavior = TopAppBarDefaults.pinnedScrollBehavior() // Observa sempre que o
84                                     // usuário rola a lista de tarefas
85
86         Scaffold( // Scaffold tem a estrutura base para criarmos o nosso layout
87             modifier = Modifier.nestedScroll(connection = scrollBehavior.nestedScrollConnection), // Add o observer
88             floatingActionButton = { TaskActionButton() }, // Lambda para incluir o botão flutuante
89             topBar = { TaskTopBar(scrollBehavior) } // Lambda para incluir a barra de ações
90         ) { paddingValues ->
91             val items by viewModel.tasks.collectAsState() // Busca as tarefas cadastradas no bd
92
93             if (
94                 items.isNotEmpty() // Se a lista não estiver vazia, queremos mostra as tarefas
95             ) {
96                 TaskListScreen(items, paddingValues) // Lista de tarefas
97             } else { // Caso contrário
98                 TaskEmptyScreen(paddingValues) // Mensagem informativa de que não há nenhuma tarefa
99             }
100         }
101     }
102
103     @Composable
104     private fun TaskActionButton() {...}
105
106
107     @Composable
108     private fun TaskTopBar(scrollBehavior: TopAppBarScrollBehavior) {...}
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128     /* ----- */
129
130     @Composable // Indica para o compilador que essa função é um layout
131     private fun TaskListScreen(items: List<Task>, padding: PaddingValues) {...}
132
133
134
135
136
137
138
139
140
141
142     @Composable
```

ToDoApp > app > src > main > java > com > software > todo > presentation > ui > TaskActivity > ScreenScreen 101:6 LF UTF-8 TA To-Do App Dark 4 spaces



Android Studio interface showing the code for `TaskActivity.kt` in a To-Do App project. The code defines a `TaskActivity` class that inherits from `ComponentActivity` and implements a `TaskScreen` composable function. The `TaskScreen` function uses `Scaffold` to create a layout with a `TaskTopBar` and a `TaskListScreen` (or `TaskEmptyScreen` if the list is empty). The `TaskListScreen` displays a list of tasks retrieved from a database.

```
50 class TaskActivity : ComponentActivity() {
80
81     @Composable
82     private fun TaskScreen() {
83         val scrollBehavior = TopAppBarDefaults.pinnedScrollBehavior() // Observa sempre que o
84                                     // usuário rola a lista de tarefas
85
86         Scaffold( // Scaffold tem a estrutura base para criarmos o nosso layout
87             modifier = Modifier.nestedScroll(connection = scrollBehavior.nestedScrollConnection), // Add o observer
88             floatingActionButton = { TaskActionButton() }, // Lambda para incluir o botão flutuante
89             topBar = { TaskTopBar(scrollBehavior) } // Lambda para incluir a barra de ações
90         ) { paddingValues ->
91             // val items by viewModel.tasks.collectAsState() // Busca as tarefas cadastradas no bd
92             val items = listOf(
93                 Task( title: "Mercado", description: "Banana, maçã, massa e melância", checked: false, id: 0),
94                 Task( title: "Jogos", description: "Black Ops 6, God of War e Resident Evil 4", checked: true, id: 1),
95                 Task( title: "Chips", description: "M1, M2, M3, M4 e R7 9800X3D", checked: true, id: 2),
96                 Task( title: "Empresas", description: "AMD, NVIDIA, AMD >>> Intel e Corsair", checked: false, id: 3),
97                 Task( title: "Imprimir", description: "Fatura, boleto, QR, carteira de identidade", checked: true, id: 4),
98                 Task( title: "Uma tarefa qualquer", description: "Uma descrição qualquer", checked: false, id: 5)
99             )
100
101             if (
102                 items.isNotEmpty() // Se a lista não estiver vazia, queremos mostra as tarefas
103             ) {
104                 TaskListScreen(items, paddingValues) // Lista de tarefas
105             } else { // Caso contrário
106                 TaskEmptyScreen(paddingValues) // Mensagem informativa de que não há nenhuma tarefa
107             }
108         }
109     }
110
111     @Composable
112     private fun TaskActionButton() {...}
113
114
115     @Composable
116     private fun TaskTopBar(scrollBehavior: TopAppBarScrollBehavior) {...}
117
118 }
```

The code is written in Kotlin and uses Compose for UI. The `Task` data class is used to represent tasks with attributes like `title`, `description`, `checked`, and `id`. The `TaskScreen` function uses `Scaffold` to create a layout with a `TaskTopBar` and a `TaskListScreen` (or `TaskEmptyScreen` if the list is empty). The `TaskListScreen` displays a list of tasks retrieved from a database.

TA To-Do App main

Pixel 3a API 35 app

Android

- app
 - manifests
 - kotlin+java
 - com.software.todo
 - data
 - TaskDao
 - TaskDatabase
 - domain
 - Task
 - TaskSheet**
 - presentation
 - com.software.todo (androidTest)
 - com.software.todo (test)
 - java (generated)
 - res
 - res (generated)
 - Gradle Scripts
 - build.gradle.kts (Project: To-Do_App)
 - build.gradle.kts (Module :app)
 - proguard-rules.pro (ProGuard Rules for ":app")
 - gradle.properties (Project Properties)
 - gradle-wrapper.properties (Gradle Version)
 - libs.versions.toml (Version Catalog)
 - local.properties (SDK Location)
 - settings.gradle.kts (Project Settings)

TaskSheet.kt TaskDao.kt TaskDatabase.kt Task.kt

```
1 @file:Suppress( ...names: "SpellCheckingInspection")
2
3 package com.software.todo.domain
4
5 import androidx.compose.runtime.Immutable
6
7 @Immutable // @Immutable indica para o compilador do Compose que essa classe nunca irá mudar os valores
8 sealed class TaskSheet { // Define uma classe selada chamada TaskSheet.
9     // Classes seladas permitem restringir a hierarquia de classes, ou seja,
10     // TaskSheet só pode ter subclasses declaradas dentro do mesmo arquivo
11
12     data object Insert : TaskSheet() // Representa um estado específico de TaskSheet chamado Insert.
13         // Como um `data object`, ele é um singleton que pode ser usado
14         // para indicar uma ação de inserção de tarefa.
15
16     data class Update(val task: Task) : TaskSheet() // Representa o estado Update de TaskSheet.
17         // Como `data class`, contém dados específicos,
18         // nesse caso, uma tarefa (task) que será usada
19         // para atualizar informações.
20
21     data object Closed : TaskSheet() // Representa o estado Closed de TaskSheet.
22         // É um `data object` singleton, usado
23         // para indicar que editor de tarefa
24         // está fechado
25
26 }
```

TA To-Do App main Pixel 3a API 35 app

Android

- app
 - manifests
 - kotlin+java
 - com.software.todo
 - data
 - TaskDao
 - TaskDatabase
 - domain
 - Task
 - TaskSheet
 - presentation
 - theme
 - ui
 - TaskActivity
 - TaskViewModel**

TaskSheet.kt TaskViewModel.kt TaskDao.kt TaskDatabase.kt Task.kt

```
1 @file:Suppress( ...names: "SpellCheckingInspection")
2
3 package com.software.todo.presentation.ui
4
5 > import ...
19
20 // Ao herdar a classe ViewModel, estamos criando uma classe especializada para manter e gerenciar
21 // os dados de UI de maneira isolada das Activities
22 class TaskViewModel(
23     private val dao: TaskDao // Injeta uma instância do DAO para acessar as operações de bd
24 ) : ViewModel() {
25
26     // Declara uma propriedade privada `_state` como MutableStateFlow
27     // que armazena o estado atual de TaskSheet
28     //
29     // Inicialmente, `_state` é definido como `TaskSheet.Closed`,
30     // indicando que o painel de tarefas começa fechado
31     private val _state: MutableStateFlow<TaskSheet> = MutableStateFlow(TaskSheet.Closed)
32
33     // Exponibiliza uma versão imutável de `_state` chamada `state`
34     // para outras partes do código
35     //
36     // Isso é feito usando `.asStateFlow()`, que transforma `_state` em um StateFlow
37     // para que apenas essa classe possa modificar o estado
38     val state: StateFlow<TaskSheet> = _state.asStateFlow()
39
40     /* ----- */
41
42     // Variavel publica que obtem as tarefas cadastradas banco de dados
43     val tasks: StateFlow<List<Task>> = dao
44         .getAllTasks()
45         .asStateFlow() // Converte o `Flow` em um `StateFlow`
46
47     /* ----- */
48
49     // Extensão para converter um `Flow` em `StateFlow` com um
50     // escopo de `viewModelScope`. `stateIn` inicia o fluxo de forma preguiçosa
```

ToDoApp > app > src > main > java > com > software > todo > presentation > ui > TaskViewModel 48:1 LF UTF-8 TA To-Do App Dark 4 spaces

TA To-Do App main Pixel 3a API 35 app

Android

- app
 - manifests
 - kotlin+java
 - com.software.todo
 - data
 - TaskDao
 - TaskDatabase
 - domain
 - Task
 - TaskSheet
 - presentation
 - theme
 - ui
 - TaskActivity
 - TaskViewModel**
 - com.software.todo (androidTest)
 - com.software.todo (test)
 - java (generated)
 - res
 - res (generated)
 - Gradle Scripts
 - build.gradle.kts (Project: To-Do_App)
 - build.gradle.kts (Module :app)
 - proguard-rules.pro (ProGuard Rules for ":app")
 - gradle.properties (Project Properties)
 - gradle-wrapper.properties (Gradle Version)
 - libs.versions.toml (Version Catalog)
 - local.properties (SDK Location)
 - settings.gradle.kts (Project Settings)

TaskViewModel.kt

```
23 class TaskViewModel(  
46     .asStateFlow() // Converte o `Flow` em um `StateFlow`  
47  
48     /* -----  
49  
50     // Extensão para converter um `Flow` em `StateFlow` com um  
51     // escopo de `viewModelScope`. `stateIn` inicia o fluxo de forma preguiçosa  
52     // e usa uma lista vazia como valor inicial  
53     private fun <T> Flow<List<T>>.asStateFlow(): StateFlow<List<T>> {...}  
54  
55     /* -----  
56  
57     // Executa a inserção da tarefa em um escopo de `viewModelScope` para que a coroutine  
58     // seja cancelada ao final do ciclo de vida do `ViewModel`  
59     fun insert(task: Task) = viewModelScope.launch {...}  
60  
61     // Executa a atualização da tarefa em um escopo de `viewModelScope` para que a coroutine  
62     // seja cancelada ao final do ciclo de vida do `ViewModel`  
63     fun update(task: Task) = viewModelScope.launch {...}  
64  
65     // Executa a exclusão da tarefa em um escopo de `viewModelScope` para que a coroutine  
66     // seja cancelada ao final do ciclo de vida do `ViewModel`  
67     fun delete(task: Task) = viewModelScope.launch {...}  
68  
69     /* -----  
70  
71     // Função responsável por atualizar o status do nosso ModalBottomSheet  
72     fun update(update: (TaskSheet) -> TaskSheet) {  
73         _state.update(update)  
74     }  
75  
76  
77  
78  
79  
80
```

To-DoApp > app > src > main > java > com > software > todo > presentation > ui > TaskViewModel 90:2 LF UTF-8 TA To-Do App Dark 4 spaces

TA To-Do App main

Pixel 3a API 35 app

Android

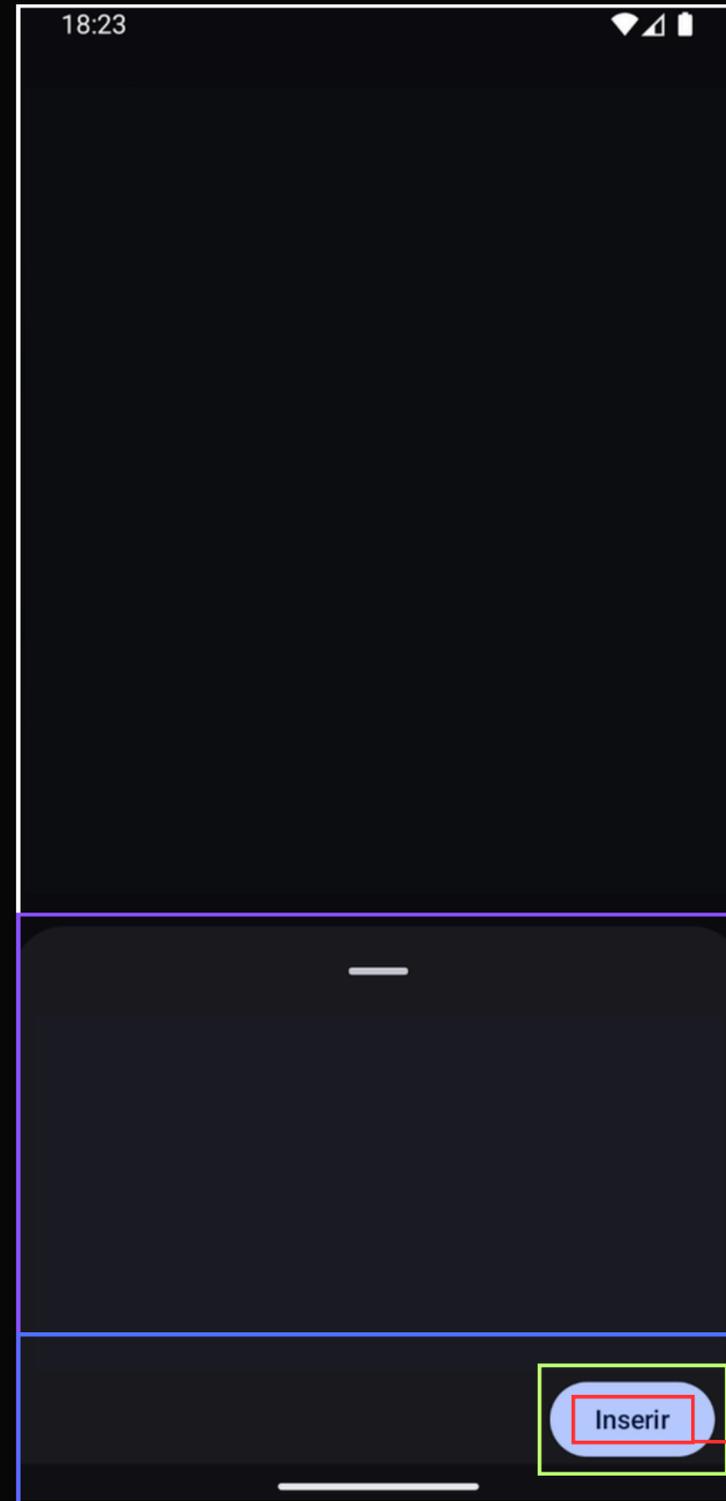
- app
 - manifests
 - kotlin+java
 - com.software.todo
 - data
 - domain
 - presentation
 - sheet
 - ModalBottomSheet.kt
 - theme
 - ui
 - com.software.todo (androidTest)
 - com.software.todo (test)
 - java (generated)
 - res
 - res (generated)
 - Gradle Scripts
 - build.gradle.kts (Project: To-Do_App)
 - build.gradle.kts (Module :app)
 - proguard-rules.pro (ProGuard Rules for ":app")
 - gradle.properties (Project Properties)
 - gradle-wrapper.properties (Gradle Version)
 - libs.versions.toml (Version Catalog)
 - local.properties (SDK Location)
 - settings.gradle.kts (Project Settings)

ModalBottomSheet.kt

```
1 @file:OptIn(ExperimentalMaterial3Api::class)
2
3 package com.software.todo.presentation.sheet
4
5 > import ...
6
7
8 @Composable
9 private fun ModalOutlinedTextField(text: String, value: String, onValueChanged: (String) -> Unit) {
10
11 }
12
13 @Composable
14 private fun ModalCheckBox(value: Boolean, onValueChanged: (Boolean) -> Unit) {
15
16 }
17
18 @Composable
19 private fun ModalButton(text: String, onClick: () -> Unit) {
20
21 }
```

ToDoApp > app > src > main > java > com > software > todo > presentation > sheet > ModalBottomSheet.kt

21:2 LF UTF-8 TA To-Do App Dark 4 spaces



ModalBottomSheet

Button

Text

Row

TA To-Do App main Pixel 3a API 35 app

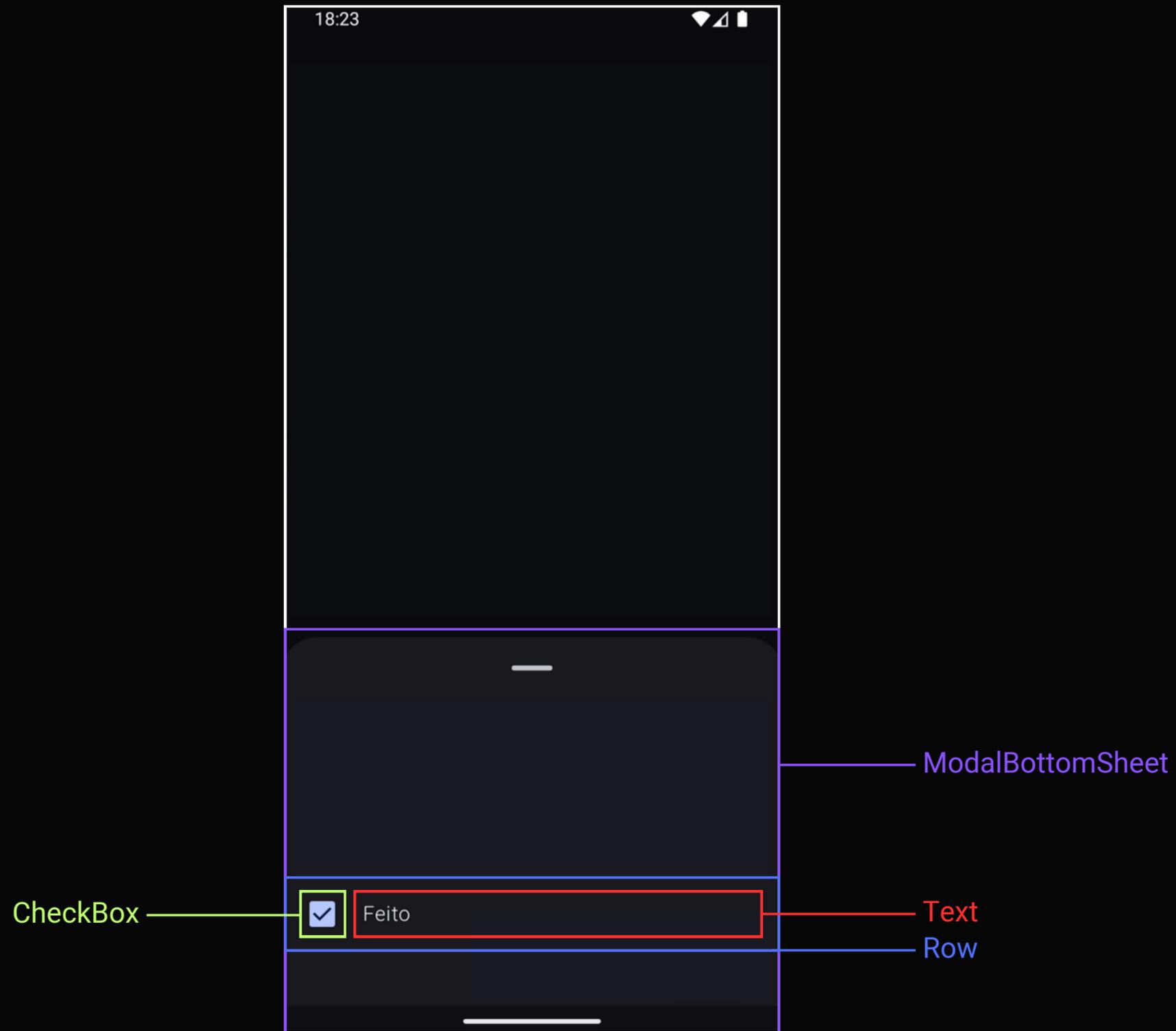
Android

- app
 - manifests
 - kotlin+java
 - com.software.todo
 - data
 - domain
 - presentation
 - sheet
 - ModalBottomSheet.kt
- com.software.todo (androidTest)
 - com.software.todo (test)
- java (generated)
- res
 - res (generated)
- Gradle Scripts
 - build.gradle.kts (Project: To-Do_App)
 - build.gradle.kts (Module :app)
 - proguard-rules.pro (ProGuard Rules for ":app")
 - gradle.properties (Project Properties)
 - gradle-wrapper.properties (Gradle Version)
 - libs.versions.toml (Version Catalog)
 - local.properties (SDK Location)
 - settings.gradle.kts (Project Settings)

ModalBottomSheet.kt

```
1 @file:OptIn(ExperimentalMaterial3Api::class)
2 @file:Suppress(...names: "SpellCheckingInspection")
3
4 package com.software.todo.presentation.sheet
5
6 > import ...
20
21 @Composable
22 > private fun ModalOutlinedTextField(text: String, value: String, onValueChanged: (String) -> Unit) {...}
36
37 @Composable
38 private fun ModalCheckBox(value: Boolean, onValueChanged: (Boolean) -> Unit) {
39     Row( // Layout horizontal (linha)
40         verticalAlignment = Alignment.CenterVertically // Alinhamento dos componentes interno no centro
41     ) {
42         Checkbox( // Caixa de seleção
43             modifier = Modifier.padding(horizontal = 8.dp), // Espaçamentos
44             onCheckedChange = onValueChanged, // Quando ocorre uma ação de clique, ele retorna true ou false
45             checked = value // Valor inicial (true ou false)
46         )
47
48         Text( // Texto para desenhar
49             modifier = Modifier.padding(vertical = 8.dp), // Espaçamentos
50             fontWeight = FontWeight.Light, // Estilo de fonte
51             text = "Feito" // Texto
52         )
53     }
54 }
55
56 @Composable
57 > private fun ModalButton(text: String, onClick: () -> Unit) {...}
```

ToDoApp > app > src > main > java > com > software > todo > presentation > sheet > ModalBottomSheet.kt 72:2 LF UTF-8 TA To-Do App Dark 4 spaces



TA To-Do App main

Pixel 3a API 35 app

Android

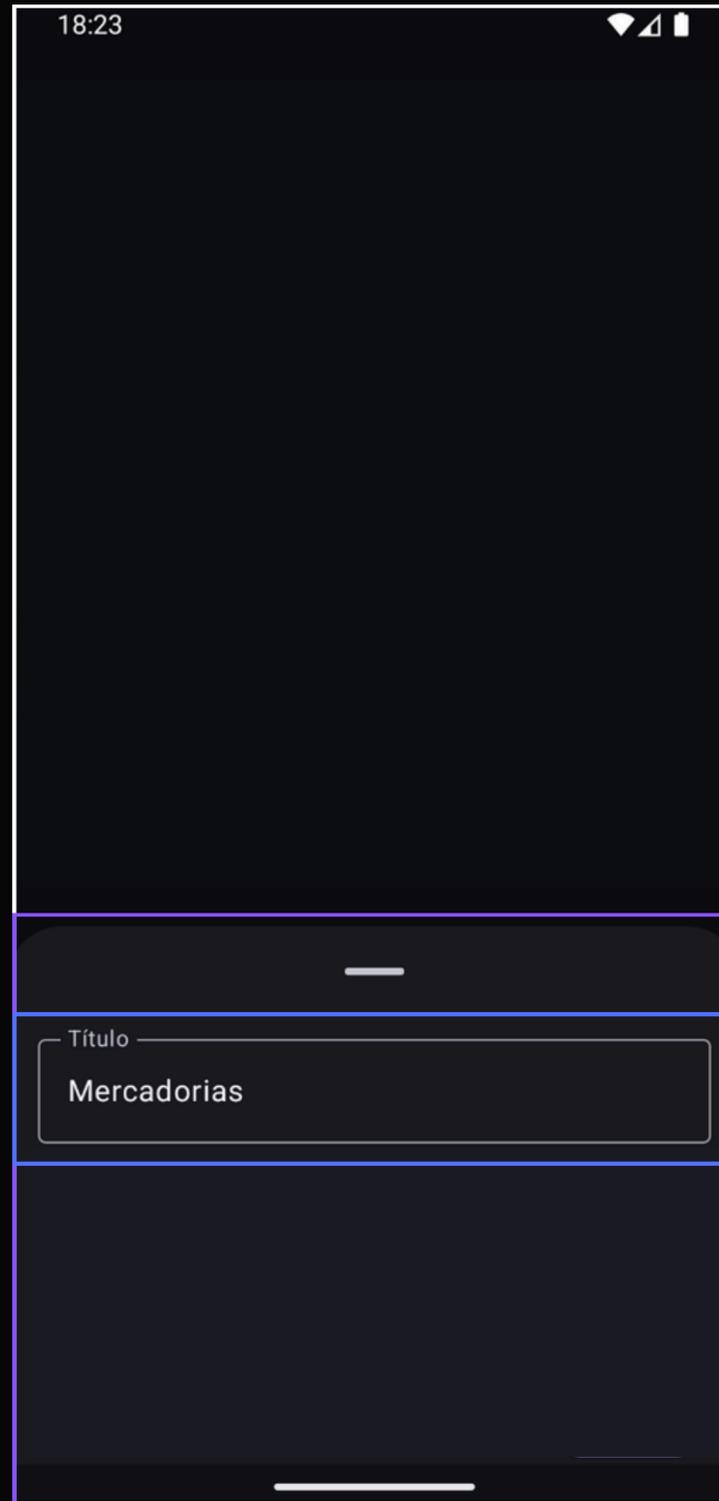
- app
 - manifests
 - kotlin+java
 - com.software.todo
 - data
 - domain
 - presentation
 - sheet
 - ModalBottomSheet.kt
 - theme
 - ui
 - com.software.todo (androidTest)
 - com.software.todo (test)
 - java (generated)
 - res
 - res (generated)
 - Gradle Scripts
 - build.gradle.kts (Project: To-Do_App)
 - build.gradle.kts (Module :app)
 - proguard-rules.pro (ProGuard Rules for ":app")
 - gradle.properties (Project Properties)
 - gradle-wrapper.properties (Gradle Version)
 - libs.versions.toml (Version Catalog)
 - local.properties (SDK Location)
 - settings.gradle.kts (Project Settings)

ModalBottomSheet.kt

```
1 @file:OptIn(ExperimentalMaterial3Api::class)
2 @file:Suppress(...names: "SpellCheckingInspection")
3
4 package com.software.todo.presentation.sheet
5
6 > import ...
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21 @Composable
22 private fun ModalOutlinedTextField(text: String, value: String, onValueChanged: (String) -> Unit) {
23     OutlinedTextField(
24         modifier = Modifier
25             .padding(horizontal = 16.dp, vertical = 4.dp) // Espaçamentos
26             .fillMaxWidth(), // Preenche toda a largura da tela
27         onValueChange = onValueChanged, // Ouvinte para saber quando o usuário digita/apaga/cola
28         value = value, // Valor inicial do texto digitável
29         label = {
30             Text( // Texto para desenhar
31                 text = text // Texto
32             )
33         }
34     )
35 }
36
37 @Composable
38 > private fun ModalCheckBox(value: Boolean, onValueChanged: (Boolean) -> Unit) {...}
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56 @Composable
57 > private fun ModalButton(text: String, onClick: () -> Unit) {...}
```

ToDoApp > app > src > main > java > com > software > todo > presentation > sheet > ModalBottomSheet.kt

72:2 LF UTF-8 TA To-Do App Dark 4 spaces



OutlineTextField

ModalBottomSheet

TA To-Do App main Pixel 3a API 35 app

Android

- app
 - manifests
 - kotlin+java
 - com.software.todo
 - data
 - domain
 - presentation
 - sheet
 - ModalBottomSheet.kt

com.software.todo (androidTest)

- com.software.todo (test)

java (generated)

res

res (generated)

Gradle Scripts

- build.gradle.kts (Project: To-Do_App)
- build.gradle.kts (Module :app)
- proguard-rules.pro (ProGuard Rules for ":app")
- gradle.properties (Project Properties)
- gradle-wrapper.properties (Gradle Version)
- libs.versions.toml (Version Catalog)
- local.properties (SDK Location)
- settings.gradle.kts (Project Settings)

ModalBottomSheet.kt

```
1 @file:OptIn(ExperimentalMaterial3Api::class)
2 @file:Suppress(...names: "SpellCheckingInspection")
3
4 package com.software.todo.presentation.sheet
5
6 > import ...
7
28
29 private val defaultValue = Task()
30
31 /* ----- */
32
33 @Composable
34 fun TaskModalBottomSheet(
35     text: String,
36     task: Task = defaultValue,
37     onDismissRequest: () -> Unit,
38     onConfirm: (Task) -> Unit
39 ) {...}
40
75
76 /* ----- */
77
78 @Composable
79 > private fun ModalOutlinedTextField(text: String, value: String, onValueChanged: (String) -> Unit) {...}
80
93
94 @Composable
95 > private fun ModalCheckBox(value: Boolean, onValueChanged: (Boolean) -> Unit) {...}
96
112
113 @Composable
114 > private fun ModalButton(text: String, onClick: () -> Unit) {...}
115
```

ToDoApp > app > src > main > java > com > software > todo > presentation > sheet > ModalBottomSheet.kt 77:1 LF UTF-8 TA To-Do App Dark 4 spaces

TA To-Do App main Pixel 3a API 35 app

Android

- app
 - manifests
 - kotlin+java
 - com.software.todo
 - data
 - domain
 - presentation
 - sheet
 - ModalBottomSheet.kt
 - theme
 - ui
- com.software.todo (androidTest)
 - com.software.todo (test)
- java (generated)
- res
 - res (generated)
- Gradle Scripts
 - build.gradle.kts (Project: To-Do_App)
 - build.gradle.kts (Module :app)
 - proguard-rules.pro (ProGuard Rules for ":app")
 - gradle.properties (Project Properties)
 - gradle-wrapper.properties (Gradle Version)
 - libs.versions.toml (Version Catalog)
 - local.properties (SDK Location)
 - settings.gradle.kts (Project Settings)

ModalBottomSheet.kt

```
4 package com.software.todo.presentation.sheet
5
6 > import ...
28
29 private val defaultValue = Task()
30
31 /* ----- */
32
33 @Composable
34 fun TaskModalBottomSheet(
35     text: String,
36     task: Task = defaultValue,
37     onDismissRequest: () -> Unit,
38     onConfirm: (Task) -> Unit
39 ) {
40     ModalBottomSheet( // Vai mostrar um dialog sobrepondo o nosso layout
41         onDismissRequest = onDismissRequest
42     ) {
43
44     }
45 }
46
47 /* ----- */
48
49 @Composable
50 > private fun ModalOutlinedTextField(text: String, value: String, onValueChanged: (String) -> Unit) {...}
64
65 @Composable
66 > private fun ModalCheckBox(value: Boolean, onValueChanged: (Boolean) -> Unit) {...}
83
84 @Composable
85 > private fun ModalButton(text: String, onClick: () -> Unit) {...}
```

ToDoApp > app > src > main > java > com > software > todo > presentation > sheet > ModalBottomSheet.kt 45:2 LF UTF-8 TA To-Do App Dark 4 spaces

TA To-Do App main Pixel 3a API 35 app

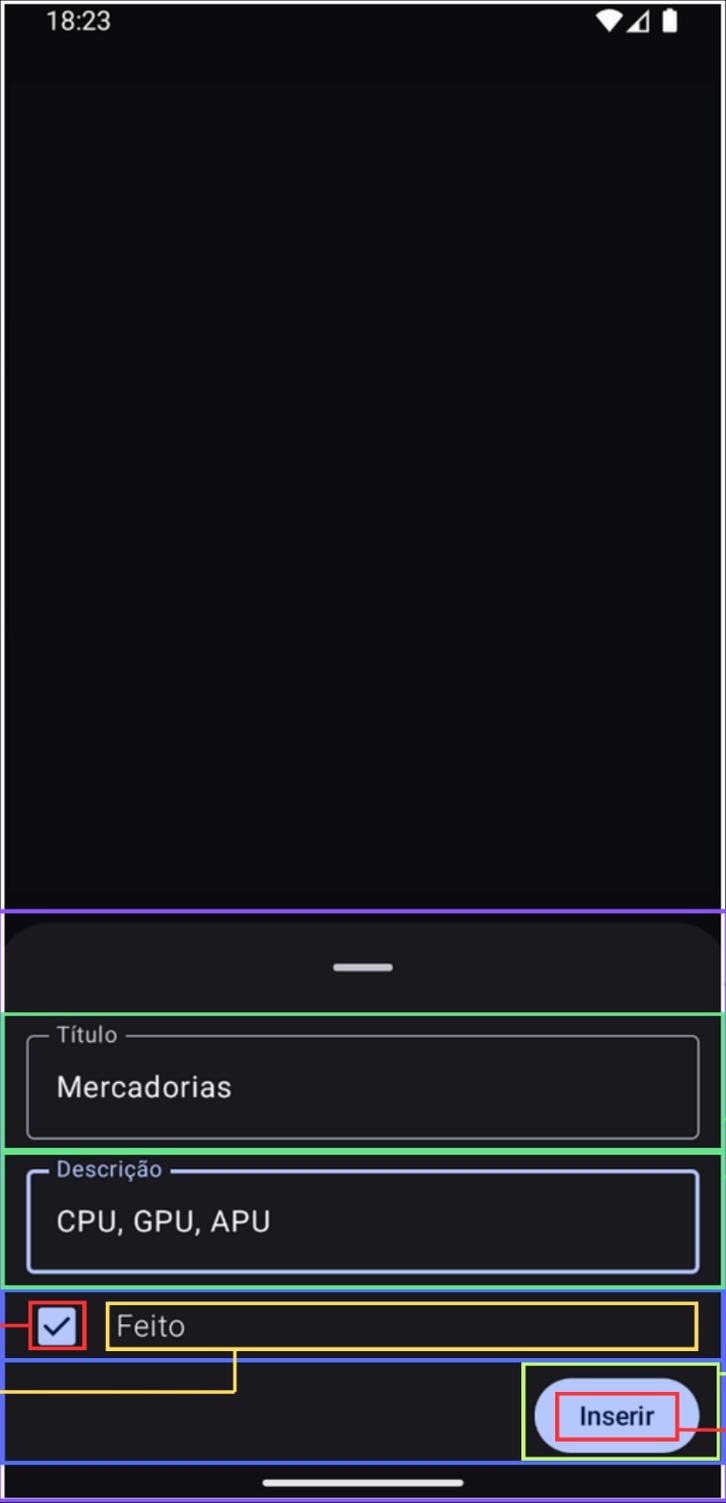
Android

- app
 - manifests
 - kotlin+java
 - com.software.todo
 - data
 - domain
 - presentation
 - sheet
 - ModalBottomSheet.kt
 - theme
 - ui
 - com.software.todo (androidTest)
 - com.software.todo (test)
 - java (generated)
 - res
 - res (generated)
 - Gradle Scripts
 - build.gradle.kts (Project: To-Do_App)
 - build.gradle.kts (Module :app)
 - proguard-rules.pro (ProGuard Rules for ":app")
 - gradle.properties (Project Properties)
 - gradle-wrapper.properties (Gradle Version)
 - libs.versions.toml (Version Catalog)
 - local.properties (SDK Location)
 - settings.gradle.kts (Project Settings)

ModalBottomSheet.kt

```
34 fun TaskModalBottomSheet(  
40     ModalBottomSheet( // Vai mostrar um dialog sobrepondo o nosso layout  
41         onDismissRequest = onDismissRequest  
42     ) { // Por padrão, o conteúdo está dentro de um 'Column'  
43         var title by remember { mutableStateOf(value = task.title) } // remember significa que quando o  
44             // compose recriar a tela quando um  
45             // valor alterar, esse se manterá,  
46             // ou seja, não vai ser resetado para  
47             // o valor padrão  
48         ModalOutlinedTextField( // Função que criamos anteriormente  
49             text = "Título",  
50             onChange = { newValue -> title = newValue }, // Alteramos o valor de 'title'  
51             value = title // Valor inicial // para o recebido (newValue)  
52         )  
53  
54         var description by remember { mutableStateOf(value = task.description) } // remember significa que quando o  
55             // compose recriar a tela quando um  
56             // valor alterar, esse se manterá,  
57             // ou seja, não vai ser resetado para  
58             // o valor padrão  
59         ModalOutlinedTextField( // Função que criamos anteriormente  
60             text = "Descrição",  
61             onChange = { newValue -> description = newValue }, // Alteramos o valor de 'description'  
62             value = description // Valor inicial // para o recebido (newValue)  
63         )  
64  
65         var checked by remember { mutableStateOf(value = task.checked) } // remember significa que quando o  
66             // compose recriar a tela quando um  
67             // valor alterar, esse se manterá,  
68             // ou seja, não vai ser resetado para  
69             // o valor padrão  
70         ModalCheckBox( // Função que criamos anteriormente  
71             onChange = { newValue -> checked = newValue }, // Alteramos o valor de 'checked'  
72             value = checked // Valor inicial // para o recebido (newValue)  
73     )  
74 }  
75 }
```

app > src > main > java > com > software > todo > presentation > sheet > ModalBottomSheet.kt > TaskModalBottomSheet 73:10 LF UTF-8 TA To-Do App Dark 4 spaces



18:23



ModalBottomSheet

OutlineTextField

OutlineTextField

Row

Button

Text

CheckBox

Text

Row

TA To-Do App main

Pixel 3a API 35 app

Android

- TaskDao
- TaskDatabase
- domain
 - Task
 - TaskSheet
- presentation
 - sheet
 - ModalBottomSheet.kt
 - theme
 - Theme.kt
 - Type.kt
 - ui
 - TaskActivity**
 - TaskViewModel
- com.software.todo (androidTest)
 - com.software.todo (test)
- java (generated)
- res
 - res (generated)
- Gradle Scripts
 - build.gradle.kts (Project: To-Do_App)
 - build.gradle.kts (Module :app)
 - proguard-rules.pro (ProGuard Rules for ":app")
 - gradle.properties (Project Properties)
 - gradle-wrapper.properties (Gradle Version)
 - libs.versions.toml (Version Catalog)
 - local.properties (SDK Location)
 - settings.gradle.kts (Project Settings)

TaskActivity.kt

```
49 class TaskActivity : AppCompatActivity() {
80     @Composable
81     private fun TaskScreen() {
82         val scrollBehavior = TopAppBarDefaults.pinnedScrollBehavior() // Observa sempre que o
83                                     // usuário rola a lista de tarefas
84
85         Scaffold( // Scaffold tem a estrutura base para criarmos o nosso layout
86             modifier = Modifier.nestedScroll(connection = scrollBehavior.nestedScrollConnection), // Add o observer
87             floatingActionButton = { TaskActionButton() }, // Lambda para incluir o botão flutuante
88             topBar = { TaskTopBar(scrollBehavior) } // Lambda para incluir a barra de ações
89         ) { paddingValues ->
90             val items by viewModel.tasks.collectAsState() // Busca as tarefas cadastradas no bd
91
92             if (
93                 items.isNotEmpty() // Se a lista não estiver vazia, queremos mostra as tarefas
94             ) {
95                 TaskListScreen(items, paddingValues) // Lista de tarefas
96             } else { // Caso contrário
97                 TaskEmptyScreen(paddingValues) // Mensagem informativa de que não há nenhuma tarefa
98             }
99
100         }
101     }
102 }
103
104 @Composable
105 private fun TaskActionButton() {...}
106
107
108 @Composable
109 private fun TaskTopBar(scrollBehavior: TopAppBarScrollBehavior) {...}
110
111
112 /* ----- */
113
114 @Composable // Indica para o compilador que essa função é um layout
115 private fun TaskListScreen(items: List<Task>, padding: PaddingValues) {...}
116
117
118 @Composable
```

ToDoApp > app > src > main > java > com > software > todo > presentation > ui > TaskActivity > TaskScreen 100:13 LF UTF-8 TA To-Do App Dark 4 spaces

Android Studio interface showing the code for `TaskActivity.kt`. The code defines a `TaskActivity` class that inherits from `ComponentActivity` and implements a `TaskScreen` function. The `TaskScreen` function uses `Scaffold` to create a layout with a `TaskActionButton`, `TaskTopBar`, and a `TaskListScreen` or `TaskEmptyScreen` based on the state of the tasks. The `when` statement handles the state of the tasks, including `TaskSheet.Insert`, `TaskSheet.Update`, and `TaskSheet.Closed`.

```
51 class TaskActivity : ComponentActivity() {
83     private fun TaskScreen() {
87         Scaffold( // Scaffold tem a estrutura base para criarmos o nosso layout
88             modifier = Modifier.nestedScroll(connection = scrollBehavior.nestedScrollConnection), // Add o observer
89             floatingActionButton = { TaskActionButton() }, // Lambda para incluir o botão flutuante
90             topBar = { TaskTopBar(scrollBehavior) } // Lambda para incluir a barra de ações
91         ) { paddingValues ->
92             val items by viewModel.tasks.collectAsState() // Busca as tarefas cadastradas no bd
93             val state by viewModel.state.collectAsState() // Verifica o status atual do bottom sheet
94
95             if (
96                 items.isNotEmpty() // Se a lista não estiver vazia, queremos mostra as tarefas
97             ) {
98                 TaskListScreen(items, paddingValues) // Lista de tarefas
99             } else { // Caso contrário
100                 TaskEmptyScreen(paddingValues) // Mensagem informativa de que não há nenhuma tarefa
101             }
102
103             // When em kotlin é igual o IF, IF ELSE
104             // OBS: Funciona apenas com classes seladas e enums
105             when (state) { // Quando o 'state' for:
106                 is TaskSheet.Insert -> { // Insert --> Execute esse trecho de código
107
108                 }
109
110                 is TaskSheet.Update -> { // Update --> Execute esse trecho de código
111
112                 }
113
114                 is TaskSheet.Closed -> { // Closed --> Execute esse trecho de código
115                     // Não faça nada no status de fechado
116                 }
117             }
118         }
119     }
120
121     @Composable
```

The interface also shows the project structure on the left, including the `TaskActivity` class and the `TaskScreen` function. The bottom status bar shows the file path: `To-Do App > app > src > main > java > com > software > todo > presentation > ui > TaskActivity > TaskScreen`.

Android Studio interface showing the code for `TaskActivity.kt`. The code defines a `TaskActivity` class that inherits from `ComponentActivity` and implements a `TaskScreen` interface. The code is annotated with comments in Portuguese explaining the logic for handling different states of a `TaskSheet`.

```
51 class TaskActivity : ComponentActivity() {
83     private fun TaskScreen() {
91         ) { paddingValues ->
102
103         // When em kotlin é igual o IF, IF ELSE
104         // OBS: Funciona apenas com classes seladas e enums
105         when (state) { // Quando o 'state' for:
106             is TaskSheet.Insert -> { // Insert --> Execute esse trecho de código
107                 TaskModalBottomSheet(text = "Inserir",
108                     onDismissRequest = {
109
110                     },
111                     onConfirm = { task ->
112
113                     }
114                 )
115             }
116
117             is TaskSheet.Update -> { // Update --> Execute esse trecho de código
118                 val update = state as TaskSheet.Update
119
120                 TaskModalBottomSheet(text = "Editar",
121                     onDismissRequest = {
122
123                     },
124                     onConfirm = { task ->
125
126                     },
127                     task = update.task
128                 )
129             }
130
131             is TaskSheet.Closed -> { // Closed --> Execute esse trecho de código
132                 // Não faça nada no status de fechado
133             }
134         }
135     }
136 }
```

The code is annotated with comments in Portuguese explaining the logic for handling different states of a `TaskSheet`:

- `is TaskSheet.Insert`: When the state is `Insert`, a `TaskModalBottomSheet` is shown with the text "Inserir". The `onDismissRequest` is empty, and the `onConfirm` block is empty.
- `is TaskSheet.Update`: When the state is `Update`, a `TaskModalBottomSheet` is shown with the text "Editar". The `onDismissRequest` is empty, the `onConfirm` block is empty, and the `task` property is set to `update.task`.
- `is TaskSheet.Closed`: When the state is `Closed`, no action is taken.

The bottom status bar shows the file path: `ToDoApp > app > src > main > java > com > software > todo > presentation > ui > TaskActivity > TaskScreen`. The status bar also displays the line number (130:1), line format (LF), encoding (UTF-8), and other settings.

Android Studio interface showing the code for `TaskActivity.kt`. The code defines a `TaskActivity` class that inherits from `ComponentActivity` and implements a `TaskScreen` function. The code uses a `when` statement to handle different states of a `TaskSheet`.

```
51 class TaskActivity : ComponentActivity() {
83     private fun TaskScreen() {
91         paddingValues ->
103         // When em kotlin é igual o IF, IF ELSE
104         // OBS: Funciona apenas com classes seladas e enums
105         when (state) { // Quando o 'state' for:
106             is TaskSheet.Insert -> { // Insert --> Execute esse trecho de código
107                 TaskModalBottomSheet(text = "Inserir",
108                     onDismissRequest = {
109                         viewModel.update { TaskSheet.Closed }
110                     },
111                     onConfirm = { task ->
112                         viewModel.insert(task)
113                     }
114             )
115         }
116
117         is TaskSheet.Update -> { // Update --> Execute esse trecho de código
118             val update = state as TaskSheet.Update
119
120             TaskModalBottomSheet(text = "Editar",
121                 onDismissRequest = {
122                     viewModel.update { TaskSheet.Closed }
123                 },
124                 onConfirm = { task ->
125                     viewModel.update(task)
126                 },
127                 task = update.task
128             )
129         }
130
131         is TaskSheet.Closed -> { // Closed --> Execute esse trecho de código
132             // Não faça nada no status de fechado
133         }
134     }
135 }
136 }
```

The code is displayed in a dark theme. The left sidebar shows the project structure, including the `TaskActivity` class. The bottom status bar shows the file path: `To-DoApp > app > src > main > java > com > software > todo > presentation > ui > TaskActivity > TaskScreen`. The status bar also displays the current line (130:1), encoding (LF UTF-8), and other settings like "Dark" theme and "4 spaces" indentation.

Android Studio interface showing the code for `TaskActivity.kt`. The code defines a class `TaskActivity` that inherits from `ComponentActivity`. It includes several Composable functions for handling tasks.

```
51 class TaskActivity : ComponentActivity() {
79
80     /* ----- */
81
82     @Composable
83     private fun TaskScreen() {...}
137
138     @Composable
139     private fun TaskActionButton() {
140         FloatingActionButton( // Botão flutuante com bordas arredondadas
141             onClick = { // Lambda contendo a ação de click do usuário
142                 viewModel.update { TaskSheet.Insert }
143             }
144         ) {
145             Icon( // Layout para desenhar uma imagem
146                 imageVector = Icons.Default.AddTask, // Imagem a qual queremos desenhar
147                 contentDescription = "Nova tarefa" // Recurso de acessibilidade
148             )
149         }
150     }
151
152     @Composable
153     private fun TaskTopBar(scrollBehavior: TopAppBarScrollBehavior) {...}
162
163     /* ----- */
164
165     @Composable // Indica para o compilador que essa função é um layout
166     private fun TaskListScreen(items: List<Task>, padding: PaddingValues) {...}
176
177     @Composable
178     private fun TaskEmptyScreen(padding: PaddingValues) {...}
198
199     @Composable
200     private fun TaskItem(task: Task) {...}
245
246 }
```

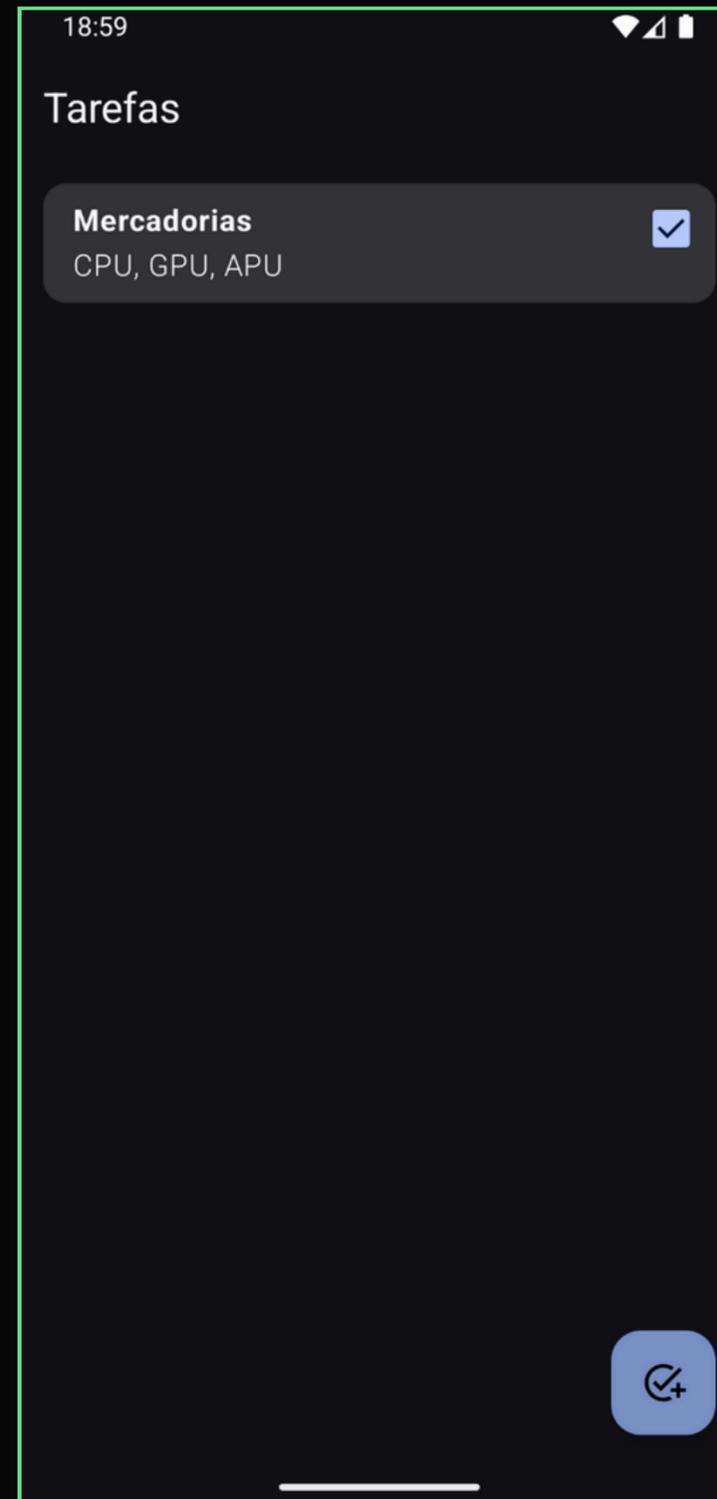
The code is displayed in a dark theme. The left sidebar shows the project structure, including the `TaskActivity` file. The bottom status bar shows the file path: `ToDoApp > app > src > main > java > com > software > todo > presentation > ui > TaskActivity > TaskActionButton`. The status bar also displays the line number `142:54`, the file encoding `LF UTF-8`, and the IDE settings `TA To-Do App Dark 4 spaces`.

Android Studio interface showing the code for `TaskActivity.kt` in a To-Do App project. The left sidebar displays the project structure, including packages like `domain`, `presentation`, `sheet`, `theme`, and `ui`. The main editor shows the Kotlin code for `TaskActivity`, which is a `ComponentActivity` implementing a Composable-based UI.

```
51 class TaskActivity : ComponentActivity() {
79
80     /* ----- */
81
82     @Composable
83     private fun TaskScreen() {...}
137
138     @Composable
139     private fun TaskActionButton() {...}
151
152     @Composable
153     private fun TaskTopBar(scrollBehavior: TopAppBarScrollBehavior) {...}
162
163     /* ----- */
164
165     @Composable // Indica para o compilador que essa função é um layout
166     private fun TaskListScreen(items: List<Task>, padding: PaddingValues) {...}
176
177     @Composable
178     private fun TaskEmptyScreen(padding: PaddingValues) {...}
198
199     @Composable
200     private fun TaskItem(task: Task) {
201         Card( // Layout com bordas e sombra (efeito flutuante)
202             modifier = Modifier
203                 .padding(horizontal = 16.dp, vertical = 8.dp) // Espaçamentos
204                 .combinedClickable( // Clique longo (pressiona e segura) e clique (clica e solta)
205                     onLongClick = { viewModel.delete(task) }, // Deleta a tarefa
206                     onClick = {
207                         viewModel.update { TaskSheet.Update(task) }
208                     }
209                 )
210         ) {
211             Row { // Tudo o que estiver dentro do lambda vai ser adicionado na horizontal (linha)
212                 Column( // Tudo o que estiver dentro do lambda vai ser adicionado na vertical (coluna)
213                     modifier = Modifier
214                         .padding(horizontal = 16.dp, vertical = 8.dp) // Espaçamentos
215                         .weight(1F) // Significa que deve preencher a
```

The code defines several Composable functions: `TaskScreen`, `TaskActionButton`, `TaskTopBar`, `TaskListScreen`, `TaskEmptyScreen`, and `TaskItem`. The `TaskItem` function uses a `Card` widget with padding and a `combinedClickable` modifier. The `onClick` event handler calls `viewModel.update { TaskSheet.Update(task) }`, which is highlighted with a red box in the image.

Bottom status bar: ToDoApp > app > src > main > java > com > software > todo > presentation > ui > TaskActivity > TaskItem | 207:68 LF UTF-8 | TA To-Do App | Dark | 4 spaces



Obrigado!

Baixe o projeto completo em:

<https://github.com/pedrindenard/task-app>

