

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



COMPOSE

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



XML

- 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.




```
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 accessar 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/variaveis 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 open class Calculator {
27
28     // Função responsável por somar dois números e retornar a soma
29     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     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     }
}
```

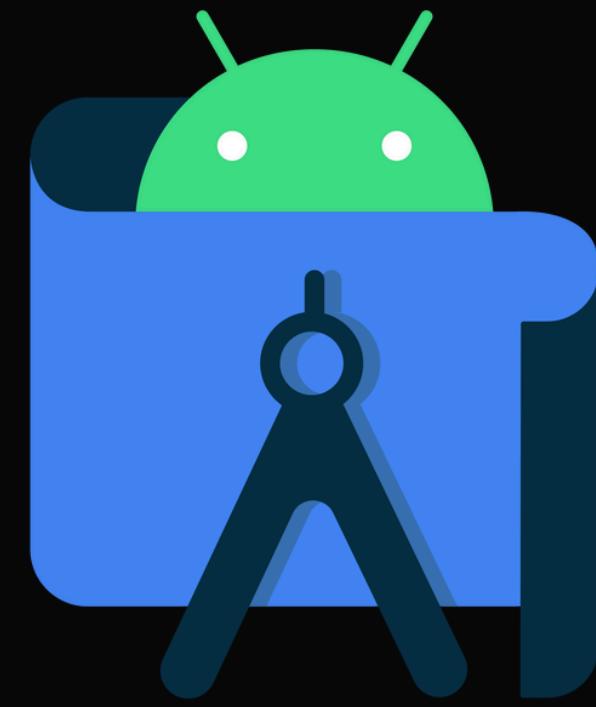
```
88 // Lambda retornando nada '() -> Unit';
89 // Lambda requisitando um valor '() -> Int`;
90 // Lambda retornando um valor '(result: Int) -> Unit`;
91
92 // Função que faz uma soma e tem como parâmetro um lambda '(result: Int) -> Unit'
93 fun calculate1(a: Int, b: Int, result: (result: Int) -> Unit) {
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 compilador 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 }
```

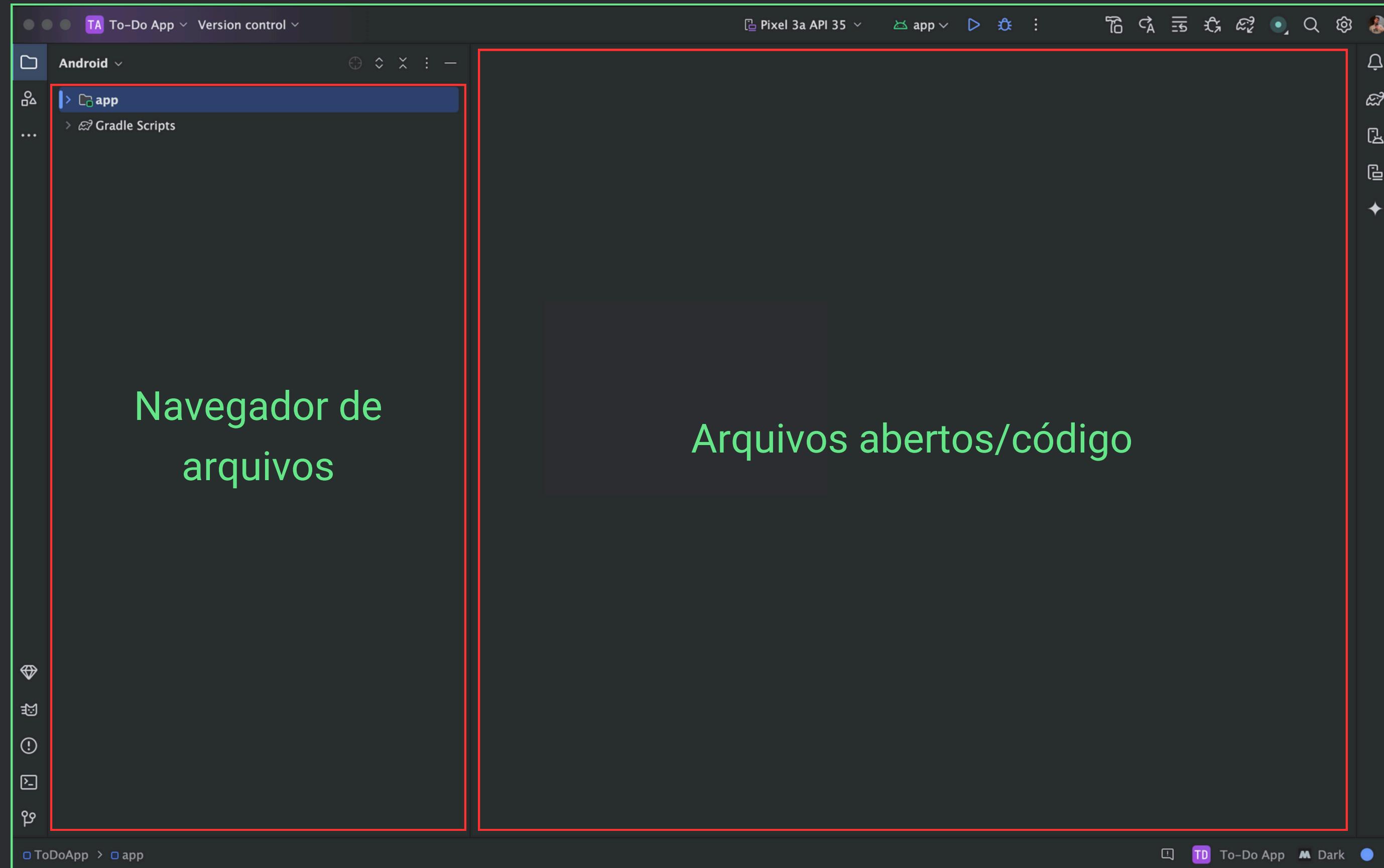
```
110 // Activity é uma [class] do Android usada para instanciar screens (telas)
111 > class MyActivity : Activity() {
112
113     // Estamos sobreescrevendo a função aberta 'onCreate' (aqui é onde desenhamos o layout)
114     // OBS: override significa que vamos sobreescrver a função
115     // que 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     (...)
```

Android Studio

BY GOOGLE

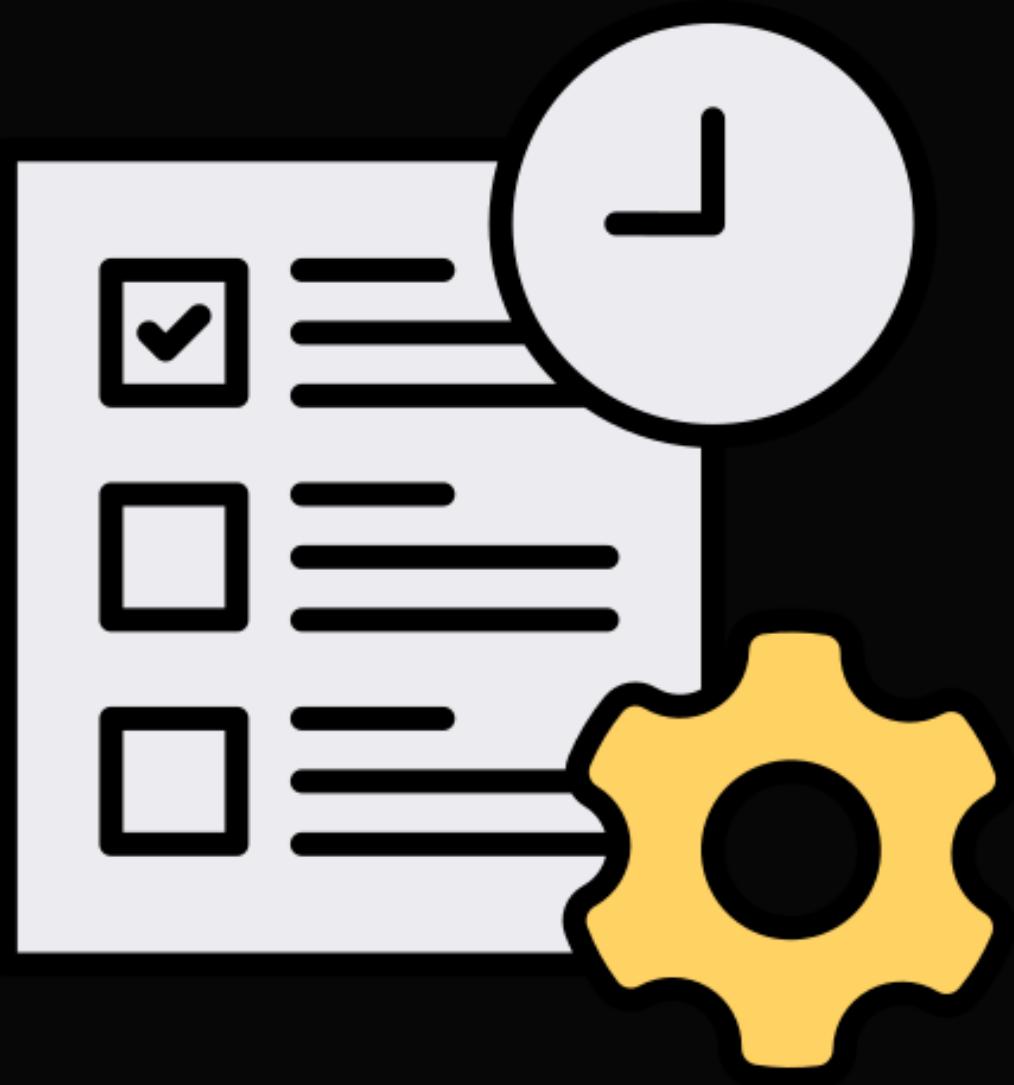


Android Studio é um ambiente de desenvolvimento integrado para desenvolver para o sistema Android, baseado no software IntelliJ IDEA da empresa JetBrains



Projeto

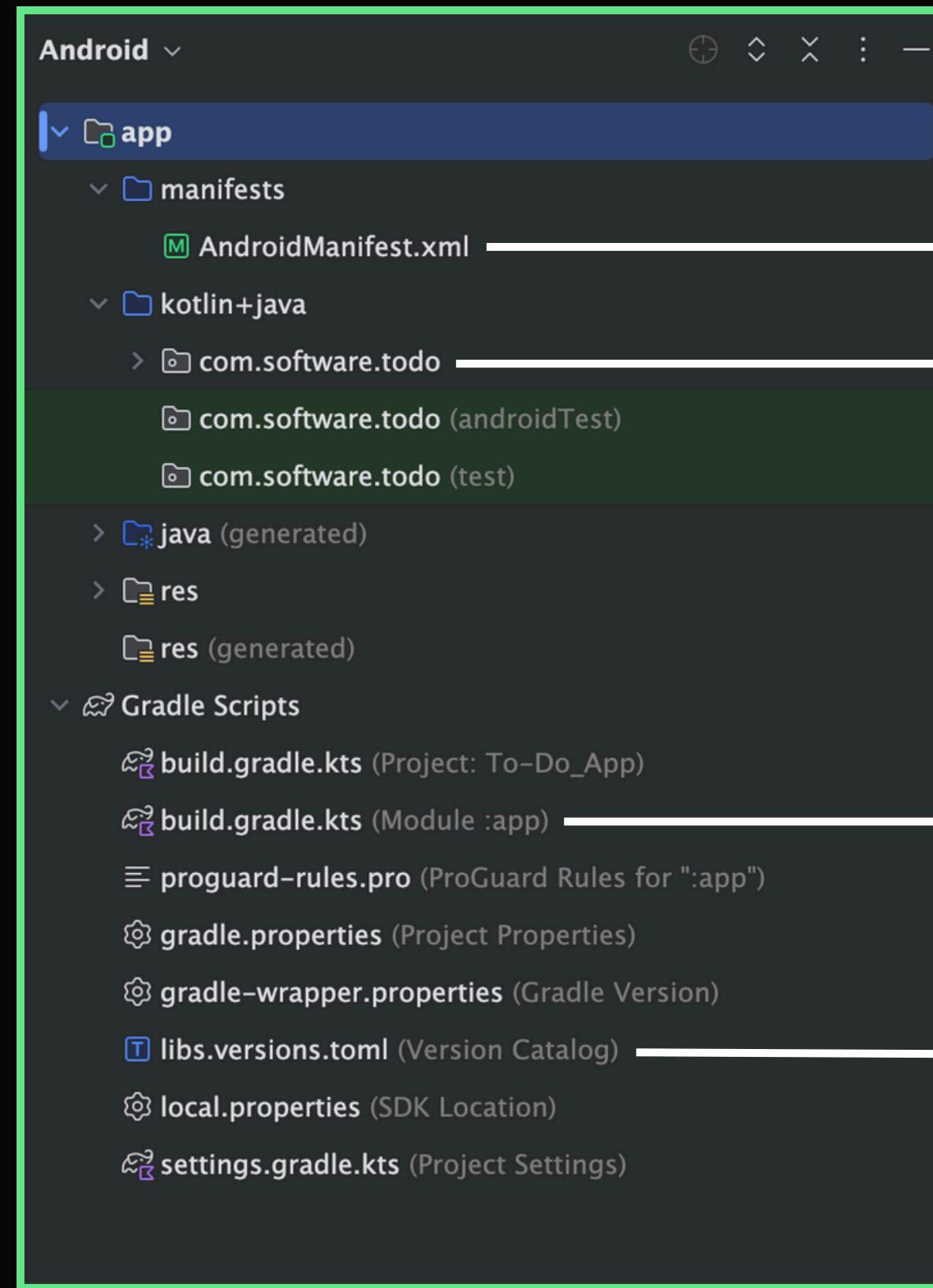
To-Do Application



The screenshot shows the Android Studio interface with the following details:

- Project Structure:** The left sidebar shows the project structure under the "Android" tab. It includes the `app` module with `manifests`, `AndroidManifest.xml`, and `kotlin+java` sections. The `kotlin+java` section contains the `com.software.todo` package, which further contains `test` and `com.software.todo.presentation.theme`. The `com.software.todo.presentation.theme` package is expanded, showing `Theme.kt` and `Type.kt`. Other sections include `java (generated)`, `res`, `res (generated)`, and `Gradle Scripts` with files like `build.gradle.kts`, `proguard-rules.pro`, and `gradle.properties`.
- Code Editor:** The right pane displays the `Theme.kt` file. The code is as follows:

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



Configurações do App

Código do App

Configurações de build

Bibliotecas

Android

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

```
[versions]
android = "8.5.2"

ksp = "2.0.0-1.0.22"
kotlin = "2.0.0"
kotlinCoroutine = "1.9.0-RC"

roomKtx = "2.6.1"
coreKtx = "1.15.0"
lifecycleKtx = "2.8.7"

composeActivity = "1.9.3"
composeBom = "2024.10.01"

composeViewModel = "2.8.7"
composeRuntime = "2.8.7"
composeIcons = "1.7.5"

[libraries]
androidx-core-ktx = { group = "androidx.core", name = "core-ktx", version.ref = "coreKtx" }
androidx-lifecycle-runtime-ktx = { group = "androidx.lifecycle", name = "lifecycle-runtime-ktx", version.ref = "lifecycleKtx" }

jetpack-room-compiler = { module = "androidx.room:room-compiler", version.ref = "roomKtx" }
jetpack-room-runtime = { module = "androidx.room:room-runtime", version.ref = "roomKtx" }
jetpack-room-kotlin = { module = "androidx.room:ktx", version.ref = "roomKtx" }

androidx-activity-compose = { group = "androidx.activity", name = "activity-compose", version.ref = "composeActivity" }
androidx-compose-bom = { group = "androidx.compose", name = "compose-bom", version.ref = "composeBom" }

androidx-ui = { group = "androidx.compose.ui", name = "ui" }
androidx-ui-graphics = { group = "androidx.compose.ui", name = "ui-graphics" }
androidx-ui-tooling = { group = "androidx.compose.ui", name = "ui-tooling" }
androidx-ui-tooling-preview = { group = "androidx.compose.ui", name = "ui-tooling-preview" }

androidx-view-model = { module = "androidx.lifecycle:lifecycle-viewmodel-compose", version.ref = "composeViewModel" }
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) highlighted
- 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

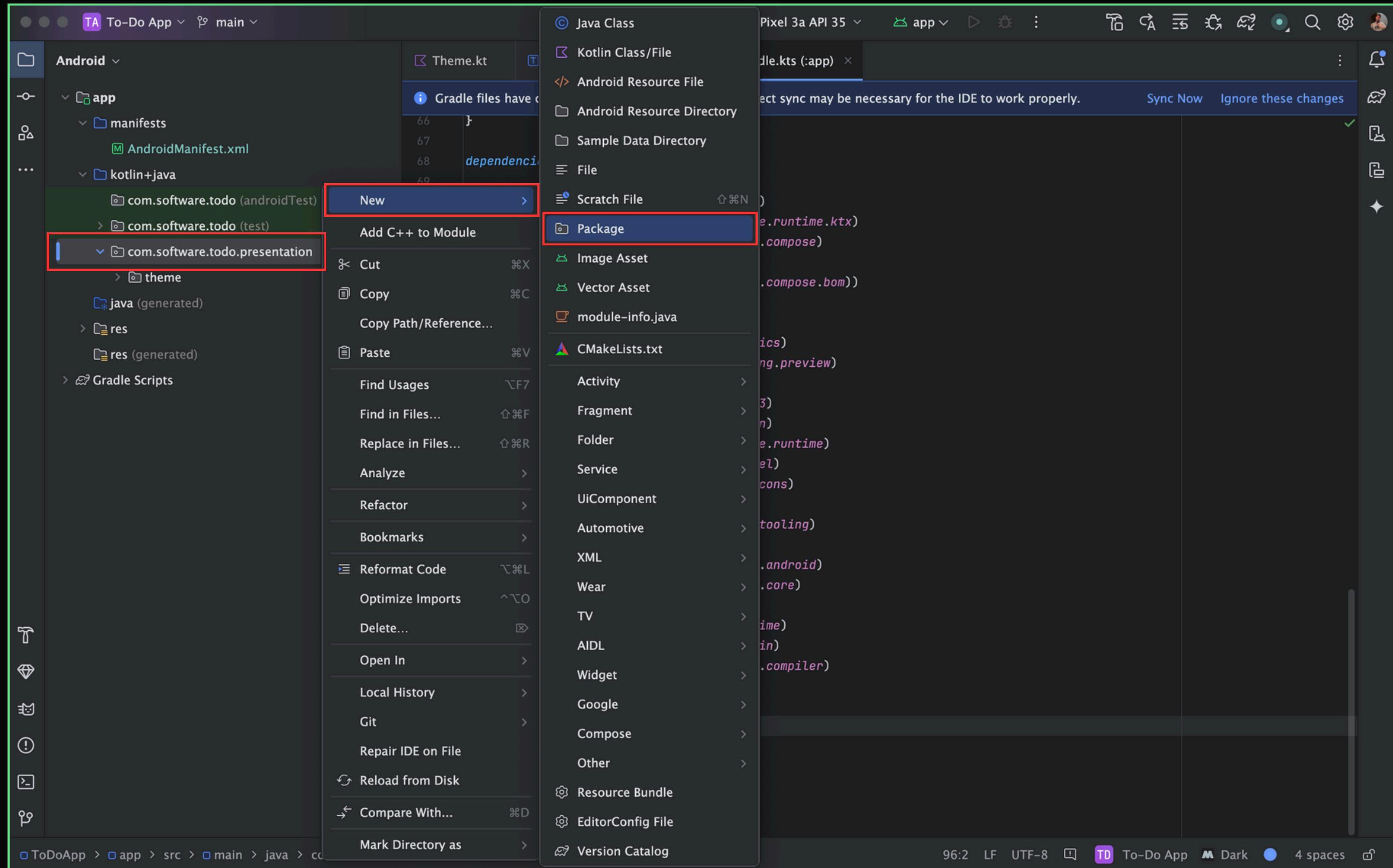
libs.versions.toml

build.gradle.kts (:app)

Gradle files have changed since last project sync. A project sync may be necessary for the IDE to work properly.

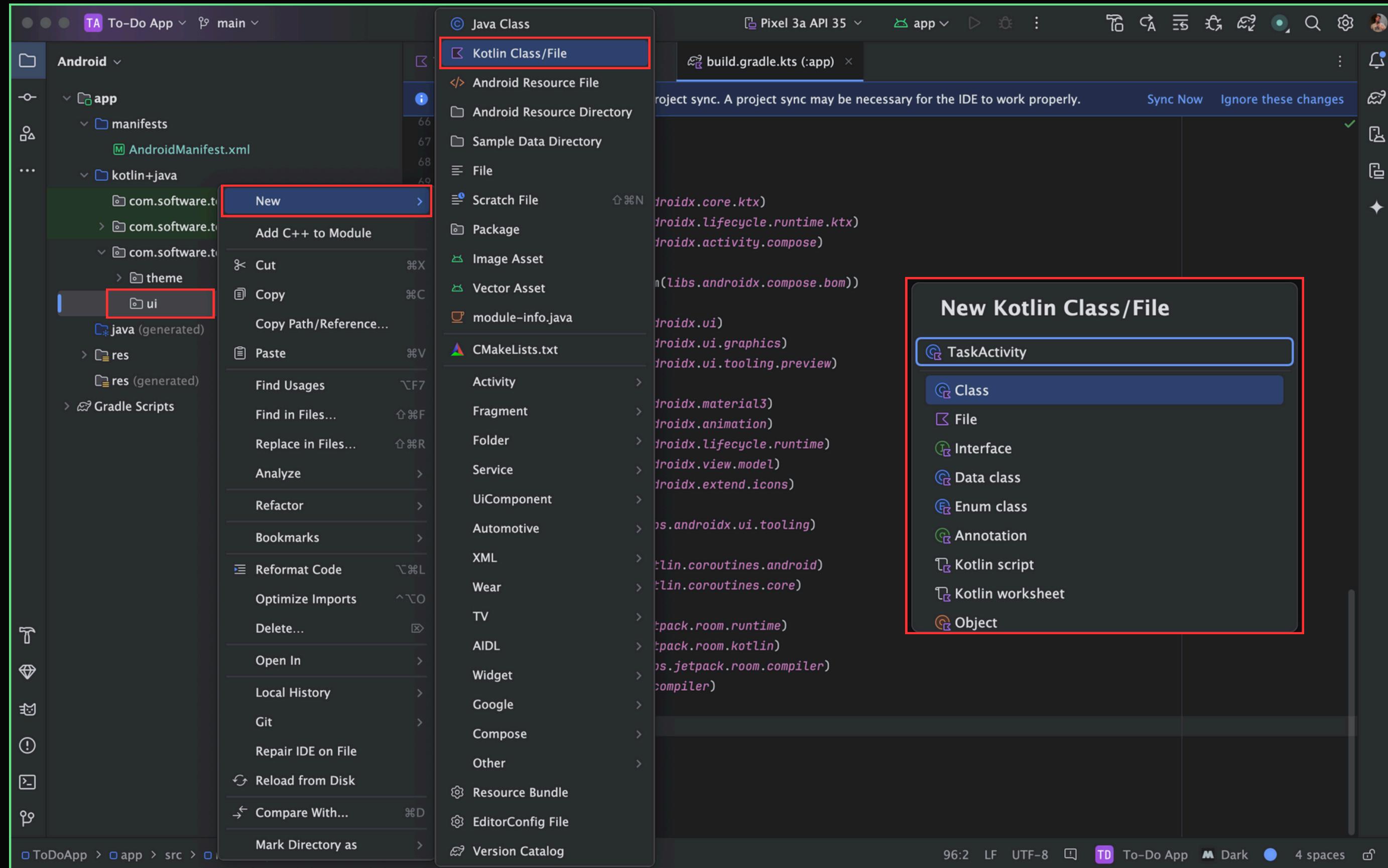
Sync Now Ignore these changes

```
dependencies {  
    implementation(libs.androidx.core.ktx)  
    implementation(libs.androidx.lifecycle.runtime.ktx)  
    implementation(libs.androidx.activity.compose)  
  
    implementation(platform(libs.androidx.compose.bom))  
  
    implementation(libs.androidx.ui)  
    implementation(libs.androidx.ui.graphics)  
    implementation(libs.androidx.ui.tooling.preview)  
  
    implementation(libs.androidx.material3)  
    implementation(libs.androidx.animation)  
    implementation(libs.androidx.lifecycle.runtime)  
    implementation(libs.androidx.viewmodel)  
    implementation(libs.androidx.extend.icons)  
  
    debugImplementation(libs.androidx.ui.tooling)  
  
    implementation(libs.kotlin.coroutines.android)  
    implementation(libs.kotlin.coroutines.core)  
  
    implementation(libs.jetpack.room.runtime)  
    implementation(libs.jetpack.room.kotlin)  
    annotationProcessor(libs.jetpack.room.compiler)  
    ksp(libs.jetpack.room.compiler)  
}
```



The screenshot shows the Android Studio interface with the project navigation bar at the top. The main window displays the build.gradle.kts file for the app module. A red box highlights a modal dialog titled "New Package" with the package name "com.software.todo.presentation.ui" entered in the input field. The code editor shows dependencies for androidx.core.ktx, lifecycle.runtime.ktx, activity.compose, compose.bom, androidx.ui, ui.graphics, ui.tooling.preview, androidx.material3, and androidx.coroutines.core. The bottom status bar indicates the file has 96 lines, is LF encoded, uses UTF-8 encoding, and is in To-Do App mode.

```
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.kotlin.coroutines.android)
82     implementation(libs.kotlin.coroutines.core)
83
84     implementation(libs.jetpack.room.runtime)
85     implementation(libs.jetpack.room.kotlin)
86     annotationProcessor(libs.jetpack.room.compiler)
87     ksp(libs.jetpack.room.compiler)
88
89
90
91
92
93
94
95
96 }
```



The screenshot shows the Android Studio interface with the following details:

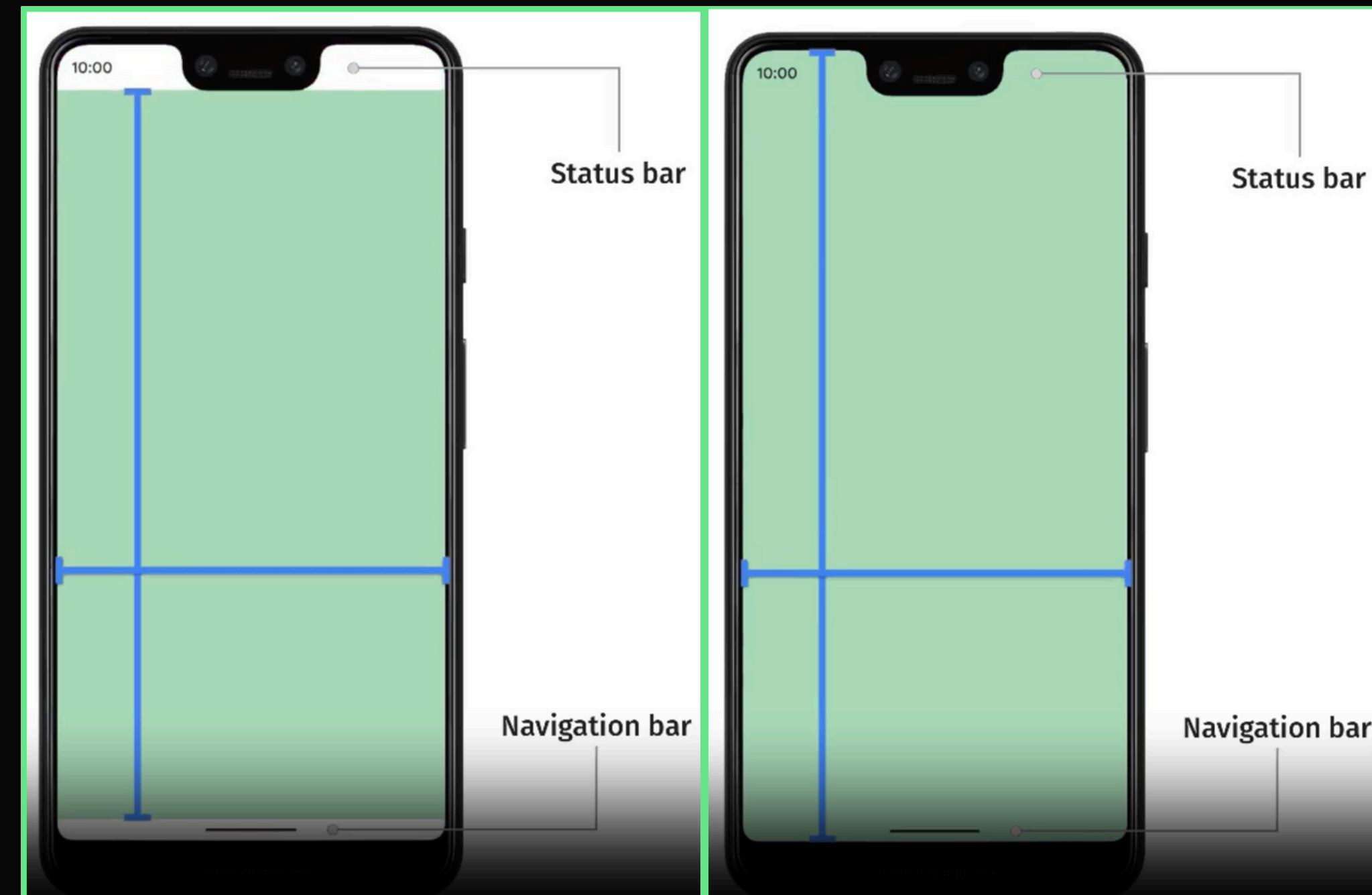
- Title Bar:** TA To-Do App - main
- Toolbar:** Pixel 3a API 35, app, navigation icons, search, settings.
- Project Navigators:** Left sidebar showing the project structure under "Android". The "ui" folder is expanded, showing "TaskActivity".
 - app
 - manifests
 - AndroidManifest.xml
 - kotlin+java
 - com.software.todo (androidTest)
 - com.software.todo (test)
 - com.software.todo.presentation
 - theme
 - ui
 - TaskActivity
- Main Editor:** TaskActivity.kt (Kotlin)

```
package com.software.todo.presentation.ui
class TaskActivity
```
- Bottom Status Bar:** ToDoApp > app > src > main > java > com > software > todo > presentation > ui > TaskActivity, 3:21 LF UTF-8, To-Do App, Dark, 4 spaces.

The screenshot shows the Android Studio interface with the following details:

- Title Bar:** TA To-Do App, develop
- Device/Emulator:** Pixel 3a API 35
- Toolbars:** Top bar with icons for navigation, search, and settings.
- Side Navigation Bar:** Shows the project structure under "Android". The "TaskActivity" file is selected in the "com.software.todo.presentation.ui" package.
- Editor Area:** Displays the code for `TaskActivity.kt`. The code includes:

```
1  @file:Suppress("SpellCheckingInspection")
2
3  package com.software.todo.presentation.ui
4
5  import ...
6
7  class TaskActivity : ComponentActivity() {
8
9      override fun onCreate(savedInstanceState: Bundle?) {
10          super.onCreate(savedInstanceState)
11          enableEdgeToEdge() // O que significa isso?
12          setContent {
13              Theme {
14                  ...
15              }
16          }
17      }
18
19  }
20
21
22
23 }
```
- Bottom Status Bar:** Shows the path "ToDoApp > app > src > main > java > com > software > todo > presentation > ui > TaskActivity", file encoding "23:2 LF UTF-8", file type "TA To-Do App", theme "Dark", and code style "4 spaces".



To-Do App main

Pixel 3a API 35 app

TaskActivity.kt AndroidManifest.xml Commit: TaskActivity.kt

Android

app manifests

AndroidManifest.xml

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

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="To-Do App"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.ToDoApp"
        tools:targetApi="31">

        <activity
            android:name=".presentation.ui.TaskActivity"
            android:exported="true"
            android:theme="@style/Theme.ToDoApp">

            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>

        </activity>
    </application>
</manifest>
```

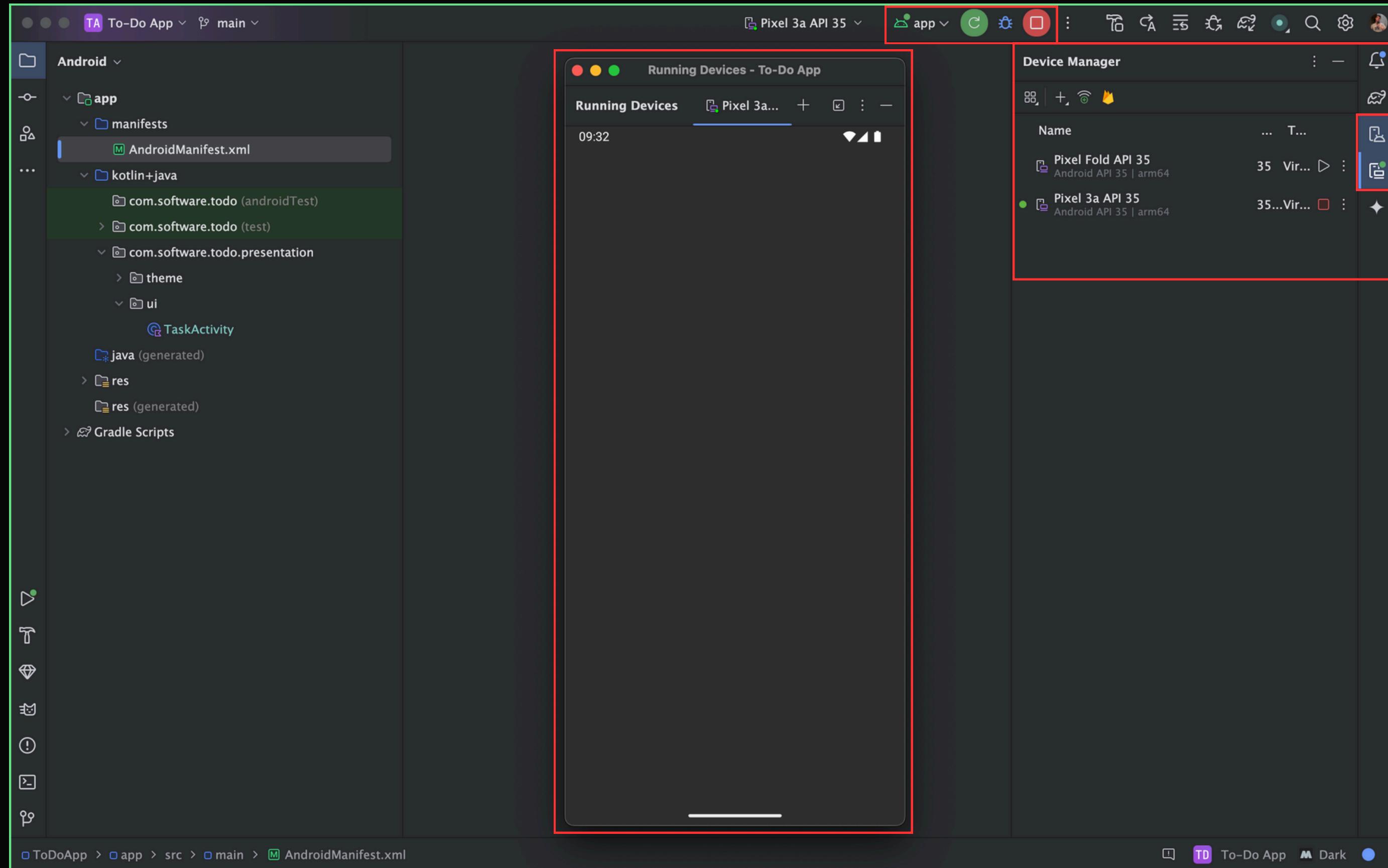
manifest > application

Text Merged Manifest

ToDoApp > app > src > main > AndroidManifest.xml

29:19 LF UTF-8 ! To-Do App M Dark ● 4 spaces ⌂

The screenshot shows the Android Studio interface with the 'AndroidManifest.xml' file open. The 'TaskActivity' section is highlighted with a red box. The 'application' and 'activity' sections are also highlighted with a red box.



The screenshot shows the Android Studio interface with the following details:

- Title Bar:** TA To-Do App, main
- Toolbar:** Pixel 3a API 35, app, navigation icons.
- File Structure (Left Sidebar):**
 - 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
- Code Editor (Right Side):** 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 )
```

The file structure sidebar shows the `Task` file under the `domain` package highlighted with a red box. The code editor shows the `Task.kt` file with its Kotlin code. The bottom status bar shows the path `ToDoApp > app > src > main > java > com > software > todo > domain > Task`, file encoding LF, character set UTF-8, and other settings like Dark mode and 4 spaces.

The screenshot shows the Android Studio interface with the following details:

- Title Bar:** TA To-Do App, main
- Toolbar:** Pixel 3a API 35, app, navigation icons.
- Side Navigation:** Shows the project structure under "Android". The "app" module is selected. Inside "app", there are "manifests", "kotlin+java", and "com.software.todo". "com.software.todo" contains "data", "domain", "Task", "presentation", and "com.software.todo" (androidTest and test). "domain" is highlighted with a red border.
- Code Editor:** The file "TaskDao.kt" is open. The code defines an interface "TaskDao" with methods for querying, inserting, updating, and deleting tasks from a database table named "task".

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

The screenshot shows the Android Studio interface with the following details:

- Title Bar:** TA To-Do App - main
- Toolbar:** Pixel 3a API 35, app, navigation icons.
- Left Sidebar (Project Structure):**
 - 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
- Central Editor:** TaskDatabase.kt
- Code Content:**

```
1  @file:Suppress("SpellCheckingInspection")
2
3  package com.software.todo.data
4
5  import ...
6
7  // @Database define esta classe como um banco de dados Room com a entidade `Task` e a versão 1
8  @Database(entities = [Task::class], version = 1)
9  abstract class TaskDatabase : RoomDatabase() {
10
11     // Fornece acesso ao DAO `TaskDao` para operações no banco de dados
12     abstract fun dao(): TaskDao
13
14     // No Kotlin, o companion object é uma maneira de definir membros estáticos dentro
15     // de uma classe. Isso significa que os métodos e propriedades definidos
16     // dentro do companion object pertencem à classe em si,
17     // e não a uma instância específica dessa classe.
18     companion object {
19
20         // Função para obter uma instância de `TaskDao` utilizando um singleton do banco de dados
21         fun getInstance(context: Context): TaskDao {
22             return Room.databaseBuilder(context, TaskDatabase::class.java, "tasks")
23                 .fallbackToDestructiveMigration() // Permite migrações destrutivas
24                 .build() // Constrói o banco de dados
25                 .dao() // Retorna o DAO `TaskDao` para interagir com o banco de dados
26         }
27     }
28
29
30
31
32
33
34 }
```
- Bottom Status Bar:** ToDoApp > app > src > main > java > com > software > todo > domain, 34:2 LF UTF-8, TA To-Do App, Dark, 4 spaces.

The screenshot shows the Android Studio interface with the following details:

- Title Bar:** TA To-Do App, main
- Toolbar:** Pixel 3a API 35, app, navigation icons.
- Side Navigation:** Shows the project structure under "Android". A red box highlights the "TaskViewModel" file in the "ui" directory.
- Code Editor:** Displays the `TaskViewModel.kt` file content. The code implements a ViewModel for managing tasks using a DAO.

```
TaskViewModel.kt
```

```
1  @file:Suppress("SpellCheckingInspection")
2
3  package com.software.todo.presentation.ui
4
5  import ...
6
7  // Ao herdar a classe ViewModel, estamos criando uma classe especializada para manter e gerenciar
8  // os dados de UI de maneira isolada das Activities
9  class TaskViewModel(
10     private val dao: TaskDao // Injeta uma instância do DAO para acessar as operações de bd
11 ) : ViewModel() {
12
13     // Executa a inserção da tarefa em um escopo de `viewModelScope` para que a coroutine
14     // seja cancelada ao final do ciclo de vida do `ViewModel`
15     fun insert(task: Task) = viewModelScope.launch {
16         withContext(Dispatchers.IO) { // Executa a atualização da tarefa de forma assíncrona
17             dao.insert(task) // Insere uma nova tarefa no banco de dados
18         }
19     }
20
21     // Executa a atualização da tarefa em um escopo de `viewModelScope` para que a coroutine
22     // seja cancelada ao final do ciclo de vida do `ViewModel`
23     fun update(task: Task) = viewModelScope.launch {
24         withContext(Dispatchers.IO) { // Executa a atualização da tarefa de forma assíncrona
25             dao.update(task) // Atualiza a tarefa no banco de dados
26         }
27     }
28
29     // Executa a exclusão da tarefa em um escopo de `viewModelScope` para que a coroutine
30     // seja cancelada ao final do ciclo de vida do `ViewModel`
31     fun delete(task: Task) = viewModelScope.launch {
32         withContext(Dispatchers.IO) { // Executa a atualização da tarefa de forma assíncrona
33             dao.delete(task) // Deleta a tarefa do banco de dados
34         }
35     }
36
37     // Executa a exclusão da tarefa em um escopo de `viewModelScope` para que a coroutine
38     // seja cancelada ao final do ciclo de vida do `ViewModel`
39     fun deleteAll(tasks: List<Task>) = viewModelScope.launch {
40         withContext(Dispatchers.IO) { // Executa a exclusão de todas as tarefas de forma assíncrona
41             dao.deleteAll(tasks)
42         }
43     }
}
```

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

The screenshot shows the Android Studio interface with the following details:

- Title Bar:** TA To-Do App, main
- Toolbar:** Pixel 3a API 35, app, navigation icons.
- Side Navigation:** Shows the project structure under "Android". The "TaskViewModel" file is selected and highlighted with a red box.
- Code Editor:** Displays the `TaskViewModel.kt` file content. A red box highlights the code block from line 23 to line 35, which defines the `tasks` variable and its implementation.
- Bottom Status Bar:** ToDoApp > app > src > main > java > com > software > todo > presentation > ui > TaskViewModel, 38:1 LF UTF-8, TA To-Do App, Dark, 4 spaces.

```
1  @file:Suppress("SpellCheckingInspection")
2
3  package com.software.todo.presentation.ui
4
5  import ...
6
7  // Ao herdar a classe ViewModel, estamos criando uma classe especializada para manter e gerenciar
8  // os dados de UI de maneira isolada das Activities
9  class TaskViewModel(
10     private val dao: TaskDao // Injeta uma instância do DAO para acessar as operações de bd
11 ) : ViewModel() {
12
13     // Variável pública que obtém as tarefas cadastradas banco de dados
14     val tasks: StateFlow<List<Task>> = dao
15         .getAllTasks()
16         .asStateFlow() // Converte o `Flow` em um `StateFlow`
17
18     /* ----- */
19
20     // Extensão para converter um `Flow` em `StateFlow` com um
21     // escopo de `viewModelScope`. `stateIn` inicia o fluxo de forma preguiçosa
22     // e usa uma lista vazia como valor inicial
23     private fun <T> Flow<List<T>>.asStateFlow(): StateFlow<List<T>> {
24         return stateIn(viewModelScope, SharingStarted.Lazily, emptyList())
25     }
26
27     /* ----- */
28
29     // Executa a inserção da tarefa em um escopo de `viewModelScope` para que a coroutine
30     // seja cancelada ao final do ciclo de vida do `ViewModel`
31     fun insert(task: Task) = viewModelScope.launch {
32         withContext(Dispatchers.IO) { // Executa a atualização da tarefa de forma assíncrona
33             dao.insert(task) // Insere uma nova tarefa no banco de dados
34         }
35     }
36
37     /* ----- */
38
39     // Executa a atualização da tarefa em um escopo de `viewModelScope` para que a coroutine
40     // seja cancelada ao final do ciclo de vida do `ViewModel`
41     fun update(task: Task) = viewModelScope.launch {
42         withContext(Dispatchers.IO) { // Executa a atualização da tarefa de forma assíncrona
43             dao.update(task) // Atualiza a tarefa existente no banco de dados
44         }
45     }
46
47     // Executa a exclusão da tarefa em um escopo de `viewModelScope` para que a coroutine
48     // seja cancelada ao final do ciclo de vida do `ViewModel`
49     fun delete(task: Task) = viewModelScope.launch {
50         withContext(Dispatchers.IO) { // Executa a exclusão da tarefa de forma assíncrona
51             dao.delete(task) // Exclui a tarefa do banco de dados
52         }
53     }
54 }
```

TA To-Do App main

Pixel 3a API 35 app

TaskActivity.kt

```
package com.software.todo.presentation.ui

import androidx.lifecycle.ViewModel
import androidx.lifecycle.ViewModelProvider

class TaskActivity : ComponentActivity() {

    // Instancia o ViewModel utilizando a função viewModels, que permite
    // a criação de ViewModel com um ViewModelProvider
    // personalizado.
    //
    // ViewModelFactory fornece uma maneira de inicializar o
    // TaskViewModel com a instância do
    // TaskDatabase.

    private val viewModel by viewModels<TaskViewModel> {
        viewModelFactory { // Bloco para criar a inicialização do ViewModel
            initializer { // Bloco para inicializar o ViewModel
                TaskViewModel( // Instância do ViewModel (igual uma class)
                    TaskDatabase.getInstance(applicationContext) // Instância do banco de dados
                )
            }
        }
    }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {
            Theme {
            }
        }
    }
}
```

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

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

The screenshot shows the Android Studio interface with the following details:

- Title Bar:** TA To-Do App - main
- Device/Emulator:** Pixel 3a API 35
- Toolbars:** Top bar with various icons for navigation and settings.
- Left Sidebar:** Shows the project structure under "Android". The "app" module is selected, containing "manifests", "kotlin+java", and "ui". The "ui" folder contains "TaskActivity" and "TaskViewModel". Other modules like "com.software.todo" and "com.software.todo (test)" are also listed.
- Right Sidebar:** Includes icons for file operations like copy, paste, and delete, along with a bell icon for notifications.
- Code Editor:** Displays the `TaskActivity.kt` file. The code defines a `ComponentActivity` named `TaskActivity`. It includes logic for initializing the `viewModel` and `viewModelFactory`. The `onCreate` method sets the content theme. A red box highlights the beginning of a new function definition at line 63: `@Composable private fun TaskItem(task: Task) {`.
- Bottom Status Bar:** Shows the file path (ToDoApp > app > src > main > java > com > software > todo > presentation > ui > TaskActivity > TaskItem), encoding (65:6 LF), character set (UTF-8), and other settings like "Dark" mode and "4 spaces".

The screenshot shows the Android Studio interface with the project structure on the left and the code editor on the right.

Project Structure:

- Project: To-Do App
- Module: main
- 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

Code Editor (TaskActivity.kt):

```
23 class TaskActivity : ComponentActivity() {  
42     override fun onCreate(savedInstanceState: Bundle?) {  
43         super.onCreate(savedInstanceState)  
44         enableEdgeToEdge()  
45         setContent {  
46             Theme {  
47                 /* ----- */  
48             }  
49         }  
50     }  
51  
52     /* ----- */  
53  
54     @Composable // Indica para o compilador que essa função é um layout  
55     private fun TaskListScreen(items: List<Task>, padding: PaddingValues) {  
56         LazyColumn(contentPadding = padding) { // Visualização de items (layout) na vertical  
57             items( // Scopo para obter os items e transformar em Composable (layout)  
58                 items = items,  
59                 key = { task -> task.id }  
60             ) { task -> // Lambda contendo o objeto tarefa  
61                 TaskItem(task = task) // Nossa tarefa. Para cada item na lista, essa função é chamada  
62             }  
63         }  
64     }  
65  
66     @Composable  
67     private fun TaskEmptyScreen(padding: PaddingValues) {...}  
68  
69     @Composable  
70     private fun TaskItem(task: Task) {...}  
71  
72  
73  
74  
75  
76  
77 }
```

A red box highlights the following section of the code:

```
54     @Composable // Indica para o compilador que essa função é um layout  
55     private fun TaskListScreen(items: List<Task>, padding: PaddingValues) {  
56         LazyColumn(contentPadding = padding) { // Visualização de items (layout) na vertical  
57             items( // Scopo para obter os items e transformar em Composable (layout)  
58                 items = items,  
59                 key = { task -> task.id }  
60             ) { task -> // Lambda contendo o objeto tarefa  
61                 TaskItem(task = task) // Nossa tarefa. Para cada item na lista, essa função é chamada  
62             }  
63         }  
64     }  
65
```

Bottom Status Bar:

- ToDoApp > app > src > main > java > com > software > todo > presentation > ui > TaskActivity
- 77:2 LF UTF-8
- TA To-Do App
- Dark
- 4 spaces

To-Do App main

Pixel 3a API 35 app : To CA ES Fx 🔍 ⚙️

Android

app

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

TaskActivity.kt

```
34 class TaskActivity : ComponentActivity() {  
35     @Composable // Indica para o compilador que essa função é um layout  
36     private fun TaskListScreen(items: List<Task>, padding: PaddingValues) {  
37         LazyColumn(contentPadding = padding) { // Visualização de items (layout) na vertical  
38             items( // Scopo para obter os items e transformar em Composable (layout)  
39                 items = items,  
40                 key = { task -> task.id }  
41             ) { task -> // Lambda contendo o objeto tarefa  
42                 TaskItem(task = task) // Nossa tarefa. Para cada item na lista, essa função é chamada  
43             }  
44         }  
45     }  
46     @Composable  
47     private fun TaskEmptyScreen(padding: PaddingValues) {...}  
48     @Composable  
49     private fun TaskItem(task: Task) {  
50         Card( // Layout com bordas e sombra (efeito flutuante)  
51             modifier = Modifier  
52                 .padding(horizontal = 16.dp, vertical = 8.dp) // Espaçamentos  
53                 .combinedClickable( // Clique longo (pressiona e segura) e clique (clica e solta)  
54                     onLongClick = { viewModel.delete(task) }, // Deleta a tarefa  
55                     onClick = {  
56                         //  
57                     }  
58                 )  
59             ) {  
60                 Row { ... }  
61             }  
62         }  
63     }  
64 }
```

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

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

To-Do App main

Pixel 3a API 35 app : To CA ES F A S Q G

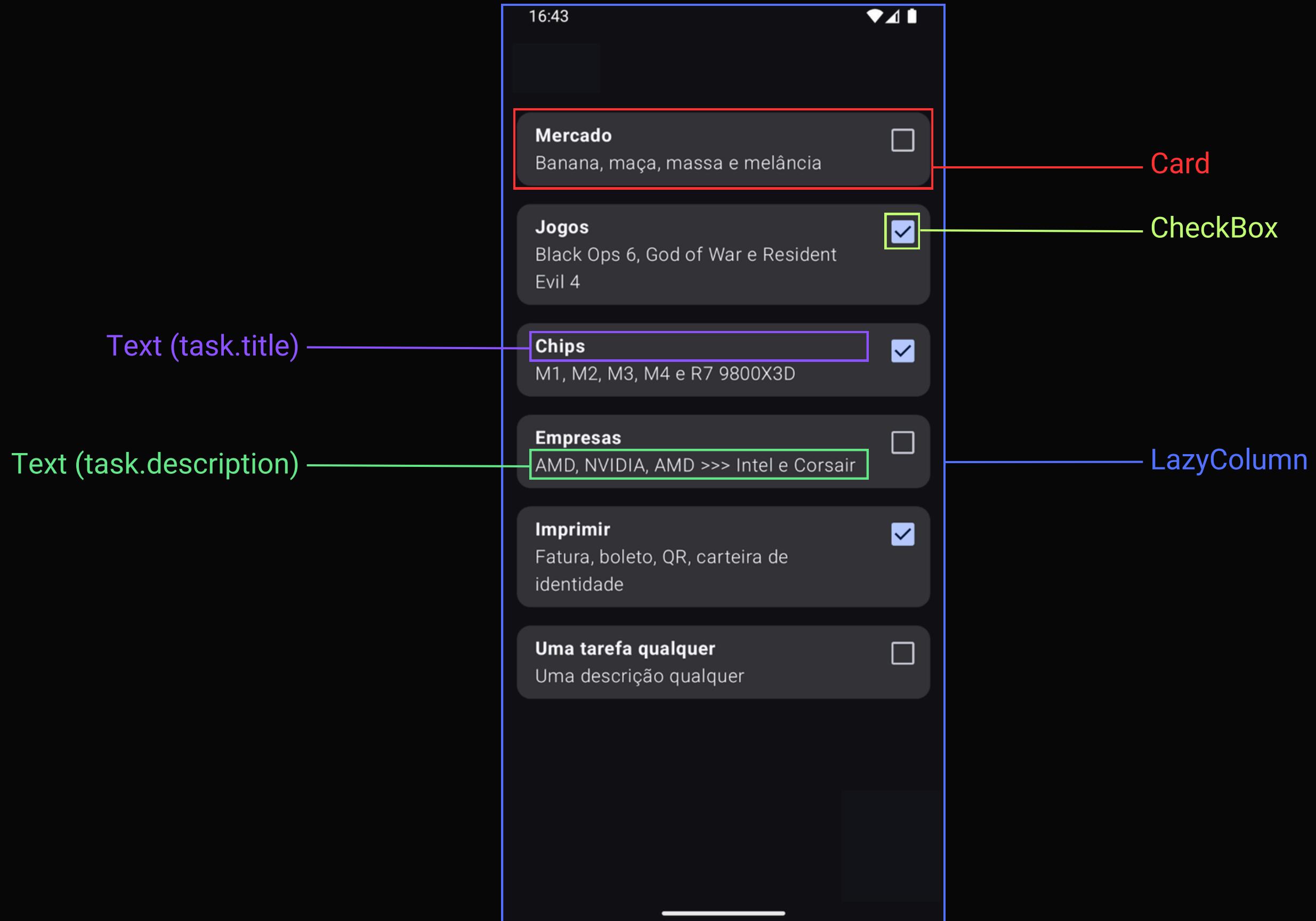
Android

TaskActivity.kt

```
34     class TaskActivity : ComponentActivity() {
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 componente 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
111                     Text( // Layout para desenhar um texto
112                         modifier = Modifier.fillMaxWidth(), // Indica ao compose que esse componente vai
113                                // preencher toda a tela na horizontal
114                         fontWeight = FontWeight.Light, // Formato de fonte mais fina
115                         text = task.description // Texto que queremos escrever (descrição da tarefa)
116
117                 )
118
119                 Checkbox( // Caixa com a possibilidade de marcar ou desmarcar
120                     checked = task.checked, // Recebe um valor do tipo [Boolean] (true ou false)
121                     onCheckedChange = { newValue -> // Lambda para capturar a ação de quando ocorre o click na caixa
122                         viewModel.update(
123                             task.copy(checked = newValue)
124                         )
125                     }
126
127                 )
128             }
129         }
130     }
```

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

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



TA To-Do App main

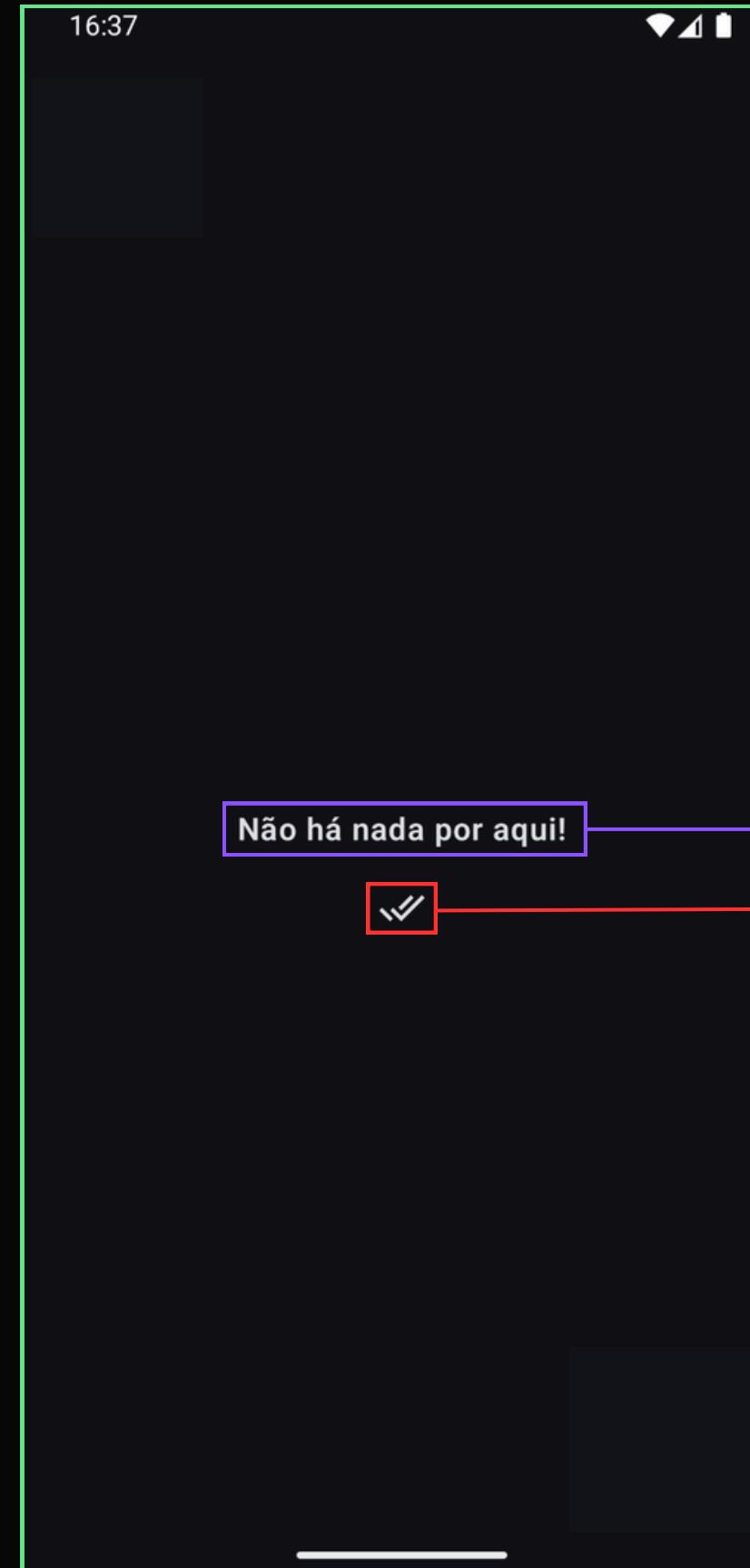
Pixel 3a API 35 app

TaskActivity.kt

```
40 class TaskActivity : ComponentActivity() {  
41     /* ----- */  
42     @Composable // Indica para o compilador que essa função é um layout  
43     private fun TaskListScreen(items: List<Task>, padding: PaddingValues) {...}  
44  
45     @Composable  
46     private fun TaskEmptyScreen(padding: PaddingValues) {  
47         Column( //  
48             modifier = Modifier  
49                 .padding(paddingValues = padding) // Espaçamentos do Scaffold (TopBar)  
50                 .fillMaxSize(), // Significa que esse componente vai preencher a tela inteira (Altura e Largura)  
51                 horizontalAlignment = Alignment.CenterHorizontally, // Alinhados horizontalmente no centro  
52                 verticalArrangement = Arrangement.Center // Alinhados verticalmente no centro  
53         ) {  
54             Text(  
55                 modifier = Modifier.padding(vertical = 16.dp), // Espaçamentos  
56                 fontWeight = FontWeight.SemiBold, // Formato de fonte em semi negrito  
57                 text = "Não há nada por aqui!" // Texto que queremos escrever  
58             )  
59  
60             Icon( // ícone para desenhar uma imagem  
61                 imageVector = Icons.Default.DoneAll, // Imagem a qual queremos  
62                 contentDescription = "Sem tarefas!" // Recurso de acessibilidade  
63             )  
64         }  
65     }  
66  
67     @Composable  
68     private fun TaskItem(task: Task) {...}  
69 }
```

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

152:2 LF UTF-8 TA To-Do App M Dark 4 spaces



Column

Text

Icon

The screenshot shows the Android Studio interface with the project navigation bar at the top. The main area displays the `TaskActivity.kt` file. A red box highlights three `@Composable` functions:

```
44 class TaskActivity : ComponentActivity() {  
50     // ViewModelFactory fornece uma maneira de inicializar o  
51     // TaskViewModel com a instância do  
52     // TaskDatabase.  
53     private val viewModel by viewModels<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     @Composable  
75     private fun ScreenScreen() {...}  
76  
77     @Composable  
78     private fun TaskActionButton() {...}  
79  
80     @Composable  
81     private fun TaskTopBar(scrollBehavior: TopAppBarScrollBehavior) {...}  
82  
83     /* ----- */  
84  
85     @Composable  
86     private fun TaskList(scrollBehavior: TopAppBarScrollBehavior) {...}  
87  
88     /* ----- */  
89  
90     @Composable  
91     private fun TaskDetail(scrollBehavior: TopAppBarScrollBehavior) {...}  
92  
93     /* ----- */  
94  
95     @Composable  
96     private fun TaskEdit(scrollBehavior: TopAppBarScrollBehavior) {...}  
97  
98     /* ----- */  
99  
100    @Composable  
101    private fun TaskDelete(scrollBehavior: TopAppBarScrollBehavior) {...}  
102  
103    /* ----- */  
104  
105    @Composable  
106    private fun TaskSearch(scrollBehavior: TopAppBarScrollBehavior) {...}
```

The screenshot shows the Android Studio interface with the following details:

- Title Bar:** TA To-Do App, main
- Toolbar:** Pixel 3a API 35, app, navigation icons
- Side Navigation:** Android, app, manifests, kotlin+java, com.software.todo (selected), data, domain, presentation, theme, ui.
- Code Editor:** TaskActivity.kt

```
44     class TaskActivity : ComponentActivity() {
45         private val viewModel by viewModels<TaskViewModel> {
46             viewModelFactory // Bloco para criar a inicialização do ViewModel
47         }
48     }
49
50     override fun onCreate(savedInstanceState: Bundle?) {
51         super.onCreate(savedInstanceState)
52         enableEdgeToEdge()
53         setContent {
54             Theme {
55                 ...
56             }
57         }
58     }
59
60     /* ----- */
61
62     @Composable
63     private fun ScreenScreen() {...}
64
65     @Composable
66     private fun TaskActionButton() {...}
67
68     @Composable
69     private fun TaskTopBar(scrollBehavior: TopAppBarScrollBehavior) {
70         TopAppBar( // Barra de ações com título
71             scrollBehavior = scrollBehavior, title = { // Lambda para desenhar o layout 'Text'
72                 Text(
73                     text = "Tasks" // Texto que queremos escrever
74                 )
75             }
76         )
77     }
78
79     /* ----- */
80
81     /* ----- */
82
83     /* ----- */
84
85     /* ----- */
86
87     /* ----- */
88
89     /* ----- */
90
91     /* ----- */
92
93     /* ----- */
94
95     /* ----- */
96
97     /* ----- */
98
99     /* ----- */
100
101
102
103
104     /* ----- */
105
106     /* ----- */
```

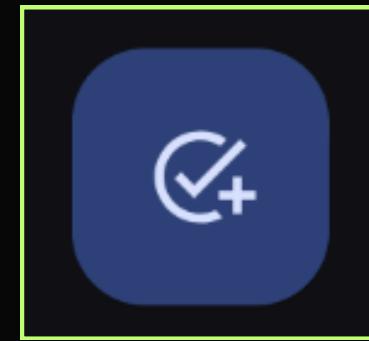
- Bottom Status Bar:** ToDoApp > app > src > main > java > com > software > todo > presentation > ui > TaskActivity, 104:1 LF UTF-8, TA To-Do App, Dark, 4 spaces, !

Tarefas

TopBar

```
44     class TaskActivity : ComponentActivity() {
45         private val viewmodel by viewmodels<TaskViewModel>()
46
47         override fun onCreate(savedInstanceState: Bundle?) {
48             super.onCreate(savedInstanceState)
49             enableEdgeToEdge()
50             setContent {
51                 Theme {
52                     ...
53                 }
54             }
55         }
56
57         /* ----- */
58
59         @Composable
60         private fun ScreenScreen() {...}
61
62         @Composable
63         private fun TaskActionButton() {
64             FloatingActionButton( // Botão flutuante com bordas arredondadas
65                 onClick = { // Lambda contendo a ação de click do usuário
66                     ...
67                 }
68             ) {
69                 Icon( // Layout para desenhar uma imagem
70                     imageVector = Icons.Default.AddTask, // Imagem a qual queremos desenhar
71                     contentDescription = "Nova tarefa" // Recurso de acessibilidade
72                 )
73             }
74         }
75
76         /* ----- */
77
78         @Composable
79         private fun TaskTopBar(scrollBehavior: TopAppBarScrollBehavior) {...}
80
81         /* ----- */
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
```

The screenshot shows the Android Studio interface with the project navigation bar at the top. The left sidebar displays the project structure under the 'Android' tab, with 'TaskActivity' selected. The main editor window shows the code for 'TaskActivity.kt'. A red box highlights the section of code defining the 'TaskActionButton' Composable function. This function creates a floating action button (FAB) with an 'Add Task' icon and a descriptive content description. The code uses Jetpack Compose's @Composable annotation and FloatingActionButton component.



FloatingActionButton

The screenshot shows the Android Studio interface with the project navigation bar at the top. The main area displays the `TaskActivity.kt` file. A red box highlights the `Scaffold` block in the code.

```
47 class TaskActivity : ComponentActivity() {  
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     @Composable  
95     private fun TaskTopBar(scrollBehavior: TopAppBarScrollBehavior) {...}  
96  
97     /* ----- */  
98  
99     @Composable // Indica para o compilador que essa função é um layout  
100    private fun TaskListScreen(items: List<Task>, padding: PaddingValues) {...}  
101  
102    @Composable  
103    private fun TaskEmptyScreen(padding: PaddingValues) {...}  
104  
105    @Composable  
106    private fun TaskItem(task: Task) {...}  
107  
108    /* ----- */  
109  
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 }
```

The bottom status bar shows the path: `ToDoApp > app > src > main > java > com > software > todo > presentation > ui > TaskActivity > ScreenScreen`. It also displays the time as 87:13, encoding as LF, character set as UTF-8, and other settings like Dark mode and 4 spaces.

To-Do App main

Pixel 3a API 35 app : To CA ES Fx 🔍 ⚙️

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 : ComponentActivity() {
78     /* ----- */
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    @Composable
107    private fun TaskTopBar(scrollBehavior: TopAppBarScrollBehavior) {...}
108
109    /* ----- */
110
111    @Composable // Indica para o compilador que essa função é um layout
112    private fun TaskListScreen(items: List<Task>, padding: PaddingValues) {...}
113
114    @Composable
```

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

101:6 LF UTF-8 ☰ TA To-Do App M Dark ● 4 spaces ☰ ⓘ

TA To-Do App main

Pixel 3a API 35 app : To CA ES F A S Q G

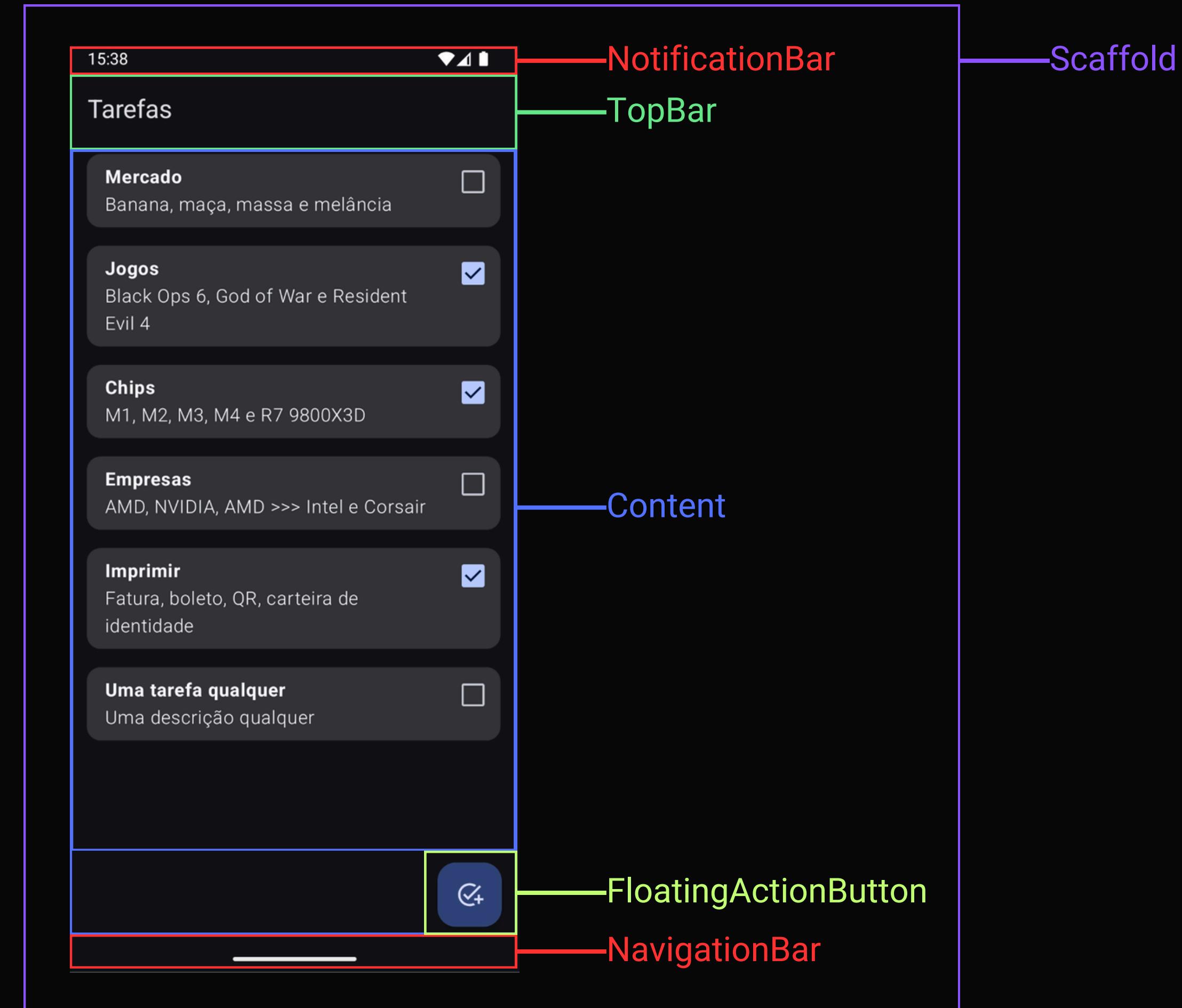
Android

TaskActivity.kt

```
50 class TaskActivity : ComponentActivity() {  
68     override fun onCreate(savedInstanceState: Bundle?) {  
69         super.onCreate(savedInstanceState)  
70         enableEdgeToEdge()  
71         setContent {  
72             Theme {  
73                 TaskScreen()  
74             }  
75         }  
76     }  
77 }  
78 /* ----- */  
80  
81 @Composable  
82 private fun TaskScreen() {...}  
83  
84 @Composable  
85 private fun TaskActionButton() {...}  
86  
87 @Composable  
88 private fun TaskTopBar(scrollBehavior: TopAppBarScrollBehavior) {...}  
89  
90 /* ----- */  
92  
93 @Composable // Indica para o compilador que essa função é um layout  
94 private fun TaskListScreen(items: List<Task>, padding: PaddingValues) {...}  
95  
96 @Composable  
97 private fun TaskEmptyScreen(padding: PaddingValues) {...}  
98  
99 @Composable  
100 private fun TaskItem(task: Task) {...}  
101  
102 }  
103  
104  
105  
106  
107  
108  
109  
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  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211 }
```

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

78:1 LF UTF-8 TA To-Do App M Dark 4 spaces



The screenshot shows the Android Studio interface with the project 'To-Do App' open. The left sidebar displays the project structure under 'Android'. The main editor window shows the file 'TaskActivity.kt'. A red box highlights the following code block:

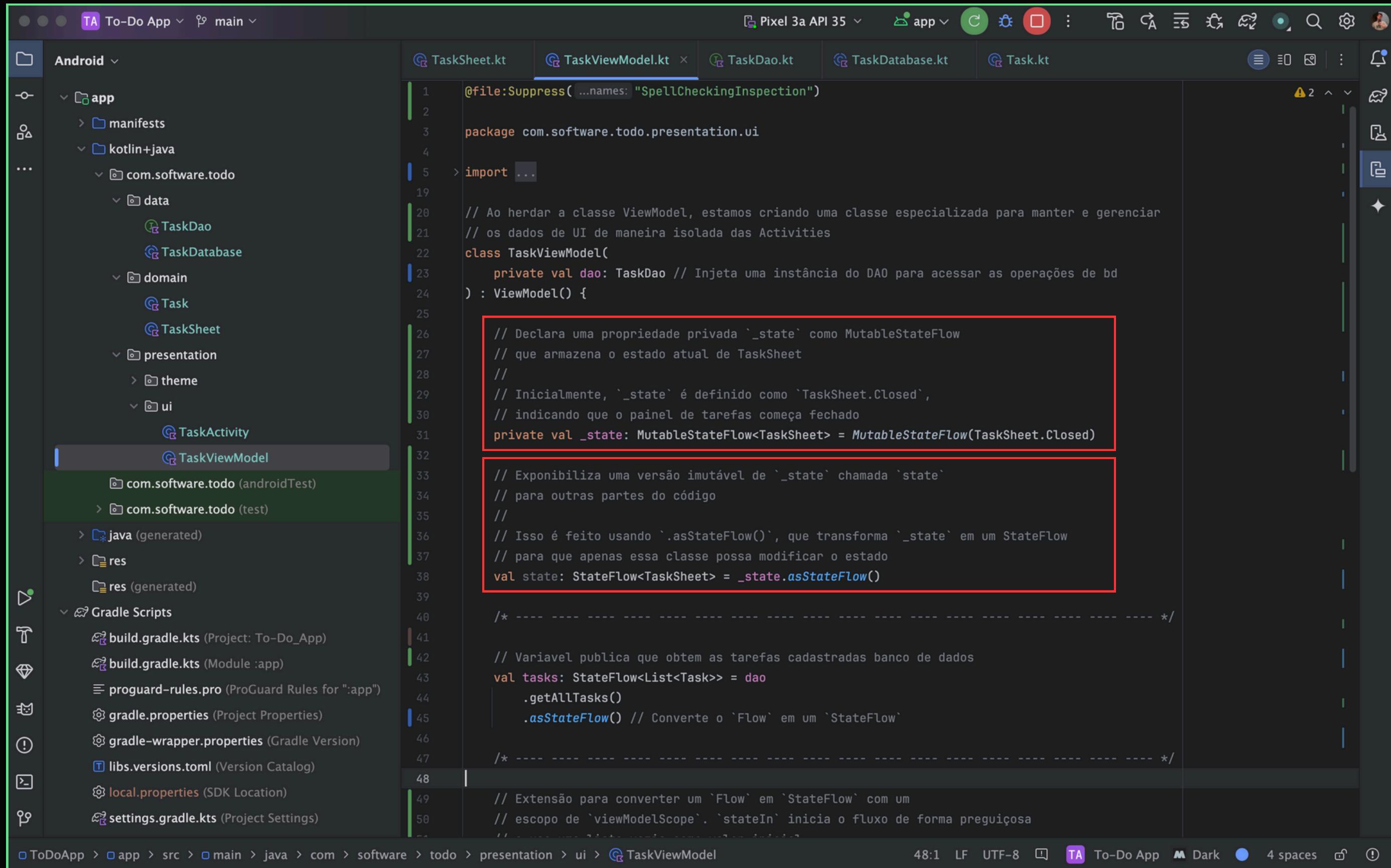
```
    // val items by viewModel.tasks.collectAsState() // Busca as tarefas cadastradas no bd
    val items = listOf(
        Task(title: "Mercado", description: "Banana, maçã, massa e melância", checked: false, id: 0),
        Task(title: "Jogos", description: "Black Ops 6, God of War e Resident Evil 4", checked: true, id: 1),
        Task(title: "Chips", description: "M1, M2, M3, M4 e R7 9800X3D", checked: true, id: 2),
        Task(title: "Empresas", description: "AMD, NVIDIA, AMD >> Intel e Corsair", checked: false, id: 3),
        Task(title: "Imprimir", description: "Fatura, boleto, QR, carteira de identidade", checked: true, id: 4),
        Task(title: "Uma tarefa qualquer", description: "Uma descrição qualquer", checked: false, id: 5)
    )
```

The code defines a list of tasks using the `Task` class. The `Task` class has properties: title, description, checked (a boolean), and id (an integer). The list contains six tasks with specific details.

The screenshot shows the Android Studio interface with the project navigation bar at the top. The project tree on the left shows the structure of the 'To-Do App' project, specifically the 'app' module. The 'TaskSheet.kt' file is open in the main editor window. The code defines a sealed class 'TaskSheet' with three data objects: 'Insert', 'Update', and 'Closed'. The code includes annotations like '@file:Suppress("SpellCheckingInspection")' and '@Immutable'.

```
1 @file:Suppress("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 }
```

The bottom status bar shows the current file path: 'ToDoApp > app > src > main > java > com > software > todo > domain > TaskSheet'. It also displays the encoding as 'UTF-8' and the font size as '26:2'. The status bar also indicates the theme is 'Dark' and uses '4 spaces' for indentation.



The screenshot shows the Android Studio interface with the project structure on the left and the code editor on the right. The code editor displays the `TaskViewModel.kt` file.

```
1  @file:Suppress("SpellCheckingInspection")
2
3  package com.software.todo.presentation.ui
4
5  import ...
6
7  // Ao herdar a classe ViewModel, estamos criando uma classe especializada para manter e gerenciar
8  // os dados de UI de maneira isolada das Activities
9  class TaskViewModel(
10     private val dao: TaskDao // Injeta uma instância do DAO para acessar as operações de bd
11 ) : ViewModel() {
12
13     // Declara uma propriedade privada `_state` como MutableStateFlow
14     // que armazena o estado atual de TaskSheet
15     //
16     // Inicialmente, `_state` é definido como `TaskSheet.Closed` ,
17     // indicando que o painel de tarefas começa fechado
18     private val _state: MutableStateFlow<TaskSheet> = MutableStateFlow(TaskSheet.Closed)
19
20     // Exponibiliza uma versão imutável de `_state` chamada `state`
21     // para outras partes do código
22     //
23     // Isso é feito usando `asStateFlow()`, que transforma `_state` em um StateFlow
24     // para que apenas essa classe possa modificar o estado
25     val state: StateFlow<TaskSheet> = _state.asStateFlow()
26
27     /* ----- */
28
29     // Variavel publica que obtém as tarefas cadastradas banco de dados
30     val tasks: StateFlow<List<Task>> = dao
31         .getAllTasks()
32         .asStateFlow() // Converte o `Flow` em um `StateFlow`
33
34     /* ----- */
35
36     // Extensão para converter um `Flow` em `StateFlow` com um
37     // escopo de `viewModelScope`. `stateIn` inicia o fluxo de forma preguiçosa
38
39     /*
40      */
41
42     // Variavel publica que obtém 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
51
52     /*
53      */
54
55     /*
56      */
57
58     /*
59      */
60
61     /*
62      */
63
64     /*
65      */
66
67     /*
68      */
69
69
70     /*
71      */
72
72
73     /*
74      */
75
75
76     /*
77      */
78
78
79     /*
80      */
80
80
81     /*
82      */
82
82
83     /*
84      */
84
84
85     /*
86      */
86
86
87     /*
88      */
88
88
89     /*
90      */
90
90
91     /*
92      */
92
92
93     /*
94      */
94
94
95     /*
96      */
96
96
97     /*
98      */
98
98
99     /*
100     */
100
100
101     /*
102     */
102
102
103     /*
104     */
104
104
105     /*
106     */
106
106
107     /*
108     */
108
108
109     /*
109     */
109
109
110     /*
110     */
110
110
111     /*
111     */
111
111
112     /*
112     */
112
112
113     /*
113     */
113
113
114     /*
114     */
114
114
115     /*
115     */
115
115
116     /*
116     */
116
116
117     /*
117     */
117
117
118     /*
118     */
118
118
119     /*
119     */
119
119
120     /*
120     */
120
120
121     /*
121     */
121
121
122     /*
122     */
122
122
123     /*
123     */
123
123
124     /*
124     */
124
124
125     /*
125     */
125
125
126     /*
126     */
126
126
127     /*
127     */
127
127
128     /*
128     */
128
128
129     /*
129     */
129
129
130     /*
130     */
130
130
131     /*
131     */
131
131
132     /*
132     */
132
132
133     /*
133     */
133
133
134     /*
134     */
134
134
135     /*
135     */
135
135
136     /*
136     */
136
136
137     /*
137     */
137
137
138     /*
138     */
138
138
139     /*
139     */
139
139
140     /*
140     */
140
140
141     /*
141     */
141
141
142     /*
142     */
142
142
143     /*
143     */
143
143
144     /*
144     */
144
144
145     /*
145     */
145
145
146     /*
146     */
146
146
147     /*
147     */
147
147
148     /*
148     */
148
148
149     /*
149     */
149
149
150     /*
150     */
150
150
151     /*
151     */
151
151
152     /*
152     */
152
152
153     /*
153     */
153
153
154     /*
154     */
154
154
155     /*
155     */
155
155
156     /*
156     */
156
156
157     /*
157     */
157
157
158     /*
158     */
158
158
159     /*
159     */
159
159
160     /*
160     */
160
160
161     /*
161     */
161
161
162     /*
162     */
162
162
163     /*
163     */
163
163
164     /*
164     */
164
164
165     /*
165     */
165
165
166     /*
166     */
166
166
167     /*
167     */
167
167
168     /*
168     */
168
168
169     /*
169     */
169
169
170     /*
170     */
170
170
171     /*
171     */
171
171
172     /*
172     */
172
172
173     /*
173     */
173
173
174     /*
174     */
174
174
175     /*
175     */
175
175
176     /*
176     */
176
176
177     /*
177     */
177
177
178     /*
178     */
178
178
179     /*
179     */
179
179
180     /*
180     */
180
180
181     /*
181     */
181
181
182     /*
182     */
182
182
183     /*
183     */
183
183
184     /*
184     */
184
184
185     /*
185     */
185
185
186     /*
186     */
186
186
187     /*
187     */
187
187
188     /*
188     */
188
188
189     /*
189     */
189
189
190     /*
190     */
190
190
191     /*
191     */
191
191
192     /*
192     */
192
192
193     /*
193     */
193
193
194     /*
194     */
194
194
195     /*
195     */
195
195
196     /*
196     */
196
196
197     /*
197     */
197
197
198     /*
198     */
198
198
199     /*
199     */
199
199
200     /*
200     */
200
200
201     /*
201     */
201
201
202     /*
202     */
202
202
203     /*
203     */
203
203
204     /*
204     */
204
204
205     /*
205     */
205
205
206     /*
206     */
206
206
207     /*
207     */
207
207
208     /*
208     */
208
208
209     /*
209     */
209
209
210     /*
210     */
210
210
211     /*
211     */
211
211
212     /*
212     */
212
212
213     /*
213     */
213
213
214     /*
214     */
214
214
215     /*
215     */
215
215
216     /*
216     */
216
216
217     /*
217     */
217
217
218     /*
218     */
218
218
219     /*
219     */
219
219
220     /*
220     */
220
220
221     /*
221     */
221
221
222     /*
222     */
222
222
223     /*
223     */
223
223
224     /*
224     */
224
224
225     /*
225     */
225
225
226     /*
226     */
226
226
227     /*
227     */
227
227
228     /*
228     */
228
228
229     /*
229     */
229
229
230     /*
230     */
230
230
231     /*
231     */
231
231
232     /*
232     */
232
232
233     /*
233     */
233
233
234     /*
234     */
234
234
235     /*
235     */
235
235
236     /*
236     */
236
236
237     /*
237     */
237
237
238     /*
238     */
238
238
239     /*
239     */
239
239
240     /*
240     */
240
240
241     /*
241     */
241
241
242     /*
242     */
242
242
243     /*
243     */
243
243
244     /*
244     */
244
244
245     /*
245     */
245
245
246     /*
246     */
246
246
247     /*
247     */
247
247
248     /*
248     */
248
248
249     /*
249     */
249
249
250     /*
250     */
250
250
251     /*
251     */
251
251
252     /*
252     */
252
252
253     /*
253     */
253
253
254     /*
254     */
254
254
255     /*
255     */
255
255
256     /*
256     */
256
256
257     /*
257     */
257
257
258     /*
258     */
258
258
259     /*
259     */
259
259
260     /*
260     */
260
260
261     /*
261     */
261
261
262     /*
262     */
262
262
263     /*
263     */
263
263
264     /*
264     */
264
264
265     /*
265     */
265
265
266     /*
266     */
266
266
267     /*
267     */
267
267
268     /*
268     */
268
268
269     /*
269     */
269
269
270     /*
270     */
270
270
271     /*
271     */
271
271
272     /*
272     */
272
272
273     /*
273     */
273
273
274     /*
274     */
274
274
275     /*
275     */
275
275
276     /*
276     */
276
276
277     /*
277     */
277
277
278     /*
278     */
278
278
279     /*
279     */
279
279
280     /*
280     */
280
280
281     /*
281     */
281
281
282     /*
282     */
282
282
283     /*
283     */
283
283
284     /*
284     */
284
284
285     /*
285     */
285
285
286     /*
286     */
286
286
287     /*
287     */
287
287
288     /*
288     */
288
288
289     /*
289     */
289
289
290     /*
290     */
290
290
291     /*
291     */
291
291
292     /*
292     */
292
292
293     /*
293     */
293
293
294     /*
294     */
294
294
295     /*
295     */
295
295
296     /*
296     */
296
296
297     /*
297     */
297
297
298     /*
298     */
298
298
299     /*
299     */
299
299
300     /*
300     */
300
300
301     /*
301     */
301
301
302     /*
302     */
302
302
303     /*
303     */
303
303
304     /*
304     */
304
304
305     /*
305     */
305
305
306     /*
306     */
306
306
307     /*
307     */
307
307
308     /*
308     */
308
308
309     /*
309     */
309
309
310     /*
310     */
310
310
311     /*
311     */
311
311
312     /*
312     */
312
312
313     /*
313     */
313
313
314     /*
314     */
314
314
315     /*
315     */
315
315
316     /*
316     */
316
316
317     /*
317     */
317
317
318     /*
318     */
318
318
319     /*
319     */
319
319
320     /*
320     */
320
320
321     /*
321     */
321
321
322     /*
322     */
322
322
323     /*
323     */
323
323
324     /*
324     */
324
324
325     /*
325     */
325
325
326     /*
326     */
326
326
327     /*
327     */
327
327
328     /*
328     */
328
328
329     /*
329     */
329
329
330     /*
330     */
330
330
331     /*
331     */
331
331
332     /*
332     */
332
332
333     /*
333     */
333
333
334     /*
334     */
334
334
335     /*
335     */
335
335
336     /*
336     */
336
336
337     /*
337     */
337
337
338     /*
338     */
338
338
339     /*
339     */
339
339
340     /*
340     */
340
340
341     /*
341     */
341
341
342     /*
342     */
342
342
343     /*
343     */
343
343
344     /*
344     */
344
344
345     /*
345     */
345
345
346     /*
346     */
346
346
347     /*
347     */
347
347
348     /*
348     */
348
348
349     /*
349     */
349
349
350     /*
350     */
350
350
351     /*
351     */
351
351
352     /*
352     */
352
352
353     /*
353     */
353
353
354     /*
354     */
354
354
355     /*
355     */
355
355
356     /*
356     */
356
356
357     /*
357     */
357
357
358     /*
358     */
358
358
359     /*
359     */
359
359
360     /*
360     */
360
360
361     /*
361     */
361
361
362     /*
362     */
362
362
363     /*
363     */
363
363
364     /*
364     */
364
364
365     /*
365     */
365
365
366     /*
366     */
366
366
367     /*
367     */
367
367
368     /*
368     */
368
368
369     /*
369     */
369
369
370     /*
370     */
370
370
371     /*
371     */
371
371
372     /*
372     */
372
372
373     /*
373     */
373
373
374     /*
374     */
374
374
375     /*
375     */
375
375
376     /*
376     */
376
376
377     /*
377     */
377
377
378     /*
378     */
378
378
379     /*
379     */
379
379
380     /*
380     */
380
380
381     /*
381     */
381
381
382     /*
382     */
382
382
383     /*
383     */
383
383
384     /*
384     */
384
384
385     /*
385     */
385
385
386     /*
386     */
386
386
387     /*
387     */
387
387
388     /*
388     */
388
388
389     /*
389     */
389
389
390     /*
390     */
390
390
391     /*
391     */
391
391
392     /*
392     */
392
392
393     /*
393     */
393
393
394     /*
394     */
394
394
395     /*
395     */
395
395
396     /*
396     */
396
396
397     /*
397     */
397
397
398     /*
398     */
398
398
399     /*
399     */
399
399
400     /*
400     */
400
400
401     /*
401     */
401
401
402     /*
402     */
402
402
403     /*
403     */
403
403
404     /*
404     */
404
404
405     /*
405     */
405
405
406     /*
406     */
406
406
407     /*
407     */
407
407
408     /*
408     */
408
408
409     /*
409     */
409
409
410     /*
410     */
410
410
411     /*
411     */
411
411
412     /*
412     */
412
412
413     /*
413     */
413
413
414     /*
414     */
414
414
415     /*
415     */
415
415
416     /*
416     */
416
416
417     /*
417     */
417
417
418     /*
418     */
418
418
419     /*
419     */
419
419
420     /*
420     */
420
420
421     /*
421     */
421
421
422     /*
422     */
422
422
423     /*
423     */
423
423
424     /*
424     */
424
424
425     /*
425     */
425
425
426     /*
426     */
426
426
427     /*
427     */
427
427
428     /*
428     */
428
428
429     /*
429     */
429
429
430     /*
430     */
430
430
431     /*
431     */
431
431
432     /*
432     */
432
432
433     /*
433     */
433
433
434     /*
434     */
434
434
435     /*
435     */
435
435
436     /*
436     */
436
436
437     /*
437     */
437
437
438     /*
438     */
438
438
439     /*
439     */
439
439
440     /*
440     */
440
440
441     /*
441     */
441
441
442     /*
442     */
442
442
443     /*
443     */
443
443
444     /*
444     */
444
444
445     /*
445     */
445
445
446     /*
446     */
446
446
447     /*
447     */
447
447
448     /*
448     */
448
448
449     /*
449     */
449
449
450     /*
450     */
450
450
451     /*
451     */
451
451
452     /*
452     */
452
452
453     /*
453     */
453
453
454     /*
454     */
454
454
455     /*
455     */
455
455
456     /*
456     */
456
456
457     /*
457     */
457
457
458     /*
458     */
458
458
459     /*
459     */
459
459
460     /*
460     */
460
460
461     /*
461     */
461
461
462     /*
462     */
462
462
463     /*
463     */
463
463
464     /*
464     */
464
464
465     /*
465     */
465
465
466     /*
466     */
466
466
467     /*
467     */
467
467
468     /*
468     */
468
468
469     /*
469     */
469
469
470     /*
470     */
470
470
471     /*
471     */
471
471
472     /*
472     */
472
472
473     /*
473     */
473
473
474     /*
474     */
474
474
475     /*
475     */
475
475
476     /*
476     */
476
476
477     /*
477     */
477
477
478     /*
478     */
478
47
```


The screenshot shows the Android Studio interface with the project **To-Do App** open. The **Android** tab is selected in the left navigation bar. The **app** module is expanded, showing the **src/main/java/com/software/todo/presentation/sheet** directory. Inside this directory, the file **ModalBottomSheet.kt** is selected and highlighted with a red box. The code editor displays the following Kotlin code:

```
1  @file:OptIn(ExperimentalMaterial3Api::class)
2
3  package com.software.todo.presentation.sheet
4
5  > import ...
6
7  @Composable
8  private fun ModalOutlinedTextField(text: String, value: String, onValueChanged: (String) -> Unit) {
9
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 }
```

The status bar at the bottom shows the path **ToDoApp > app > src > main > java > com > software > todo > presentation > sheet > ModalBottomSheet.kt**, the time **21:2**, and other system information.

The screenshot shows the Android Studio interface with the project structure on the left and the code editor on the right.

Project Structure:

- Project: To-Do App
- Module: main
- Android
- app
 - manifests
 - kotlin+java
 - com.software.todo
 - data
 - domain
 - presentation
 - sheet
 - ...
 - 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)

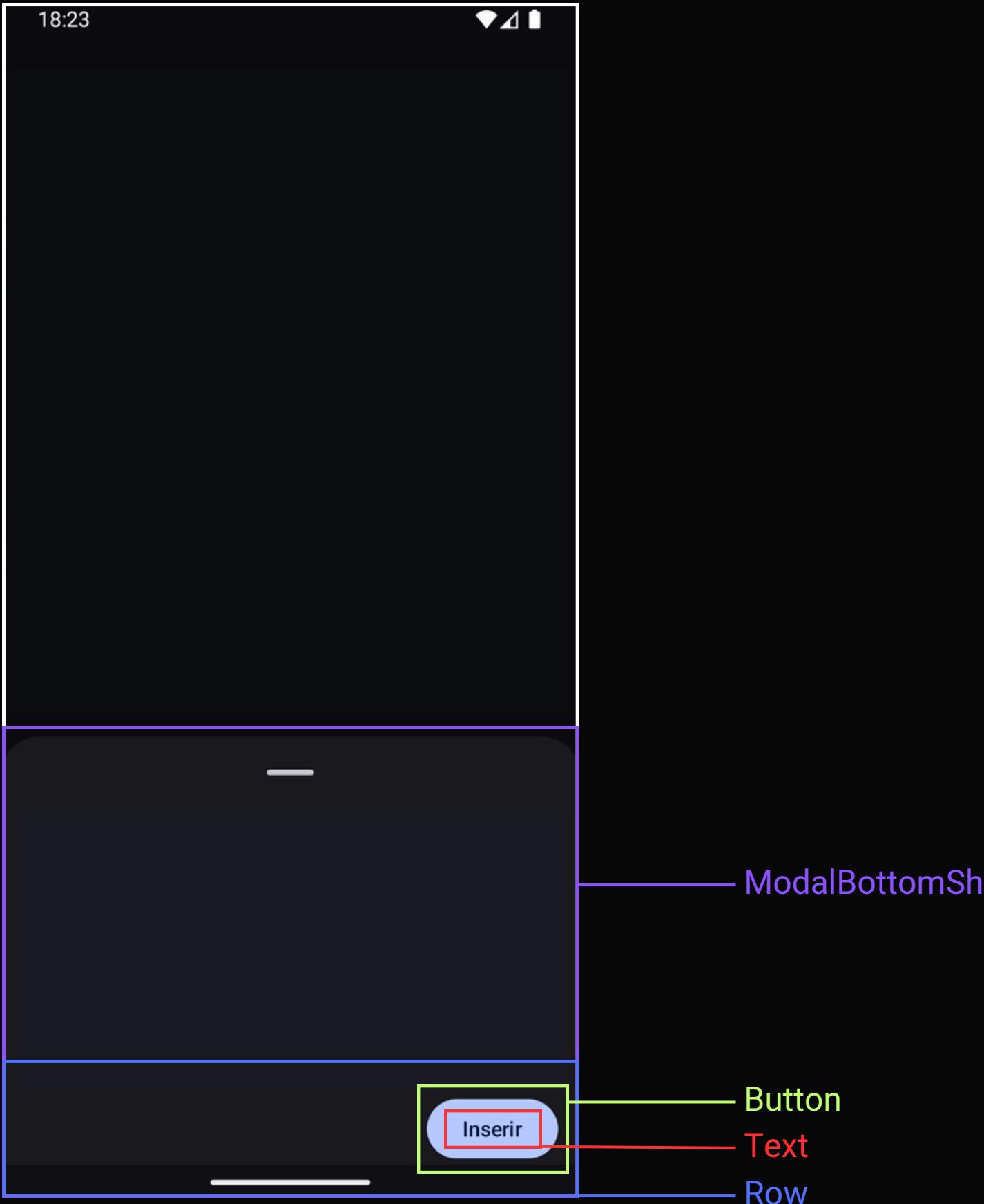
Code Editor:

```
1  @file:OptIn(ExperimentalMaterial3Api::class)
2  @file:Suppress(...names: "SpellCheckingInspection")
3
4  package com.software.todo.presentation.sheet
5
6  import ...
7
8  @Composable
9  private fun ModalOutlinedTextField(text: String, value: String, onValueChanged: (String) -> Unit) {...}
10
11 @Composable
12 private fun ModalCheckBox(value: Boolean, onValueChanged: (Boolean) -> Unit) {...}
13
14 @Composable
15 private fun ModalButton(text: String, onClick: () -> Unit) {
16     Row( // Layout horizontal (linha)
17         modifier = Modifier
18             .padding(horizontal = 16.dp) // Espaçamentos
19             .fillMaxWidth(), // Preenche toda a largura da tela
20             horizontalArrangement = Arrangement.End // Inicio do arranjo no lado direito (end)
21     ) {
22         Button( // Botão para capturar a ação de click
23             onClick = onClick // Ação de click
24         ) {
25             Text( // Texto para mostrar internamente no botão
26                 text = text // Texto
27             )
28         }
29     }
30 }
```

A red box highlights the implementation of the `ModalButton` function, specifically the `Row` and `Button` blocks.

Status Bar:

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

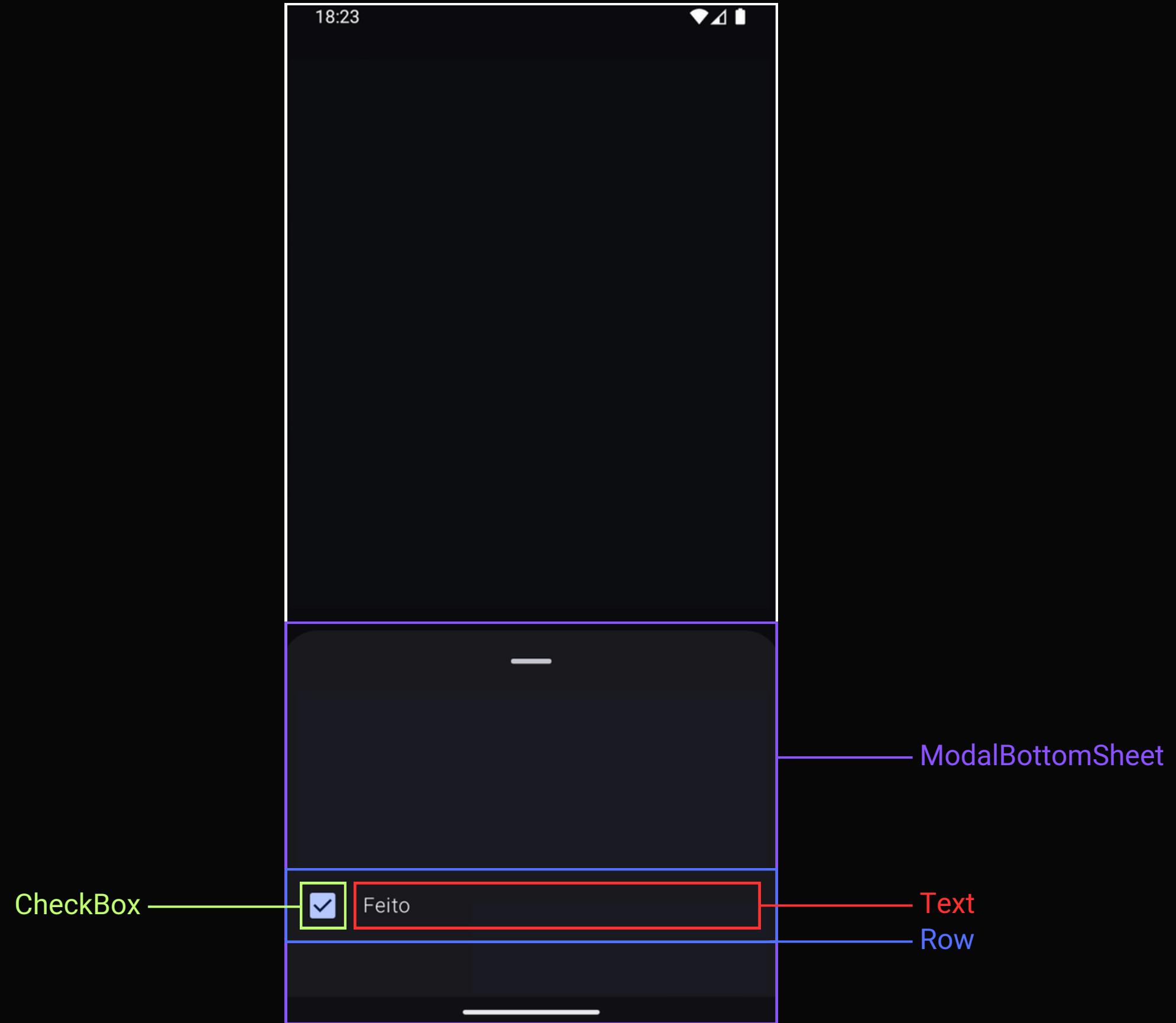


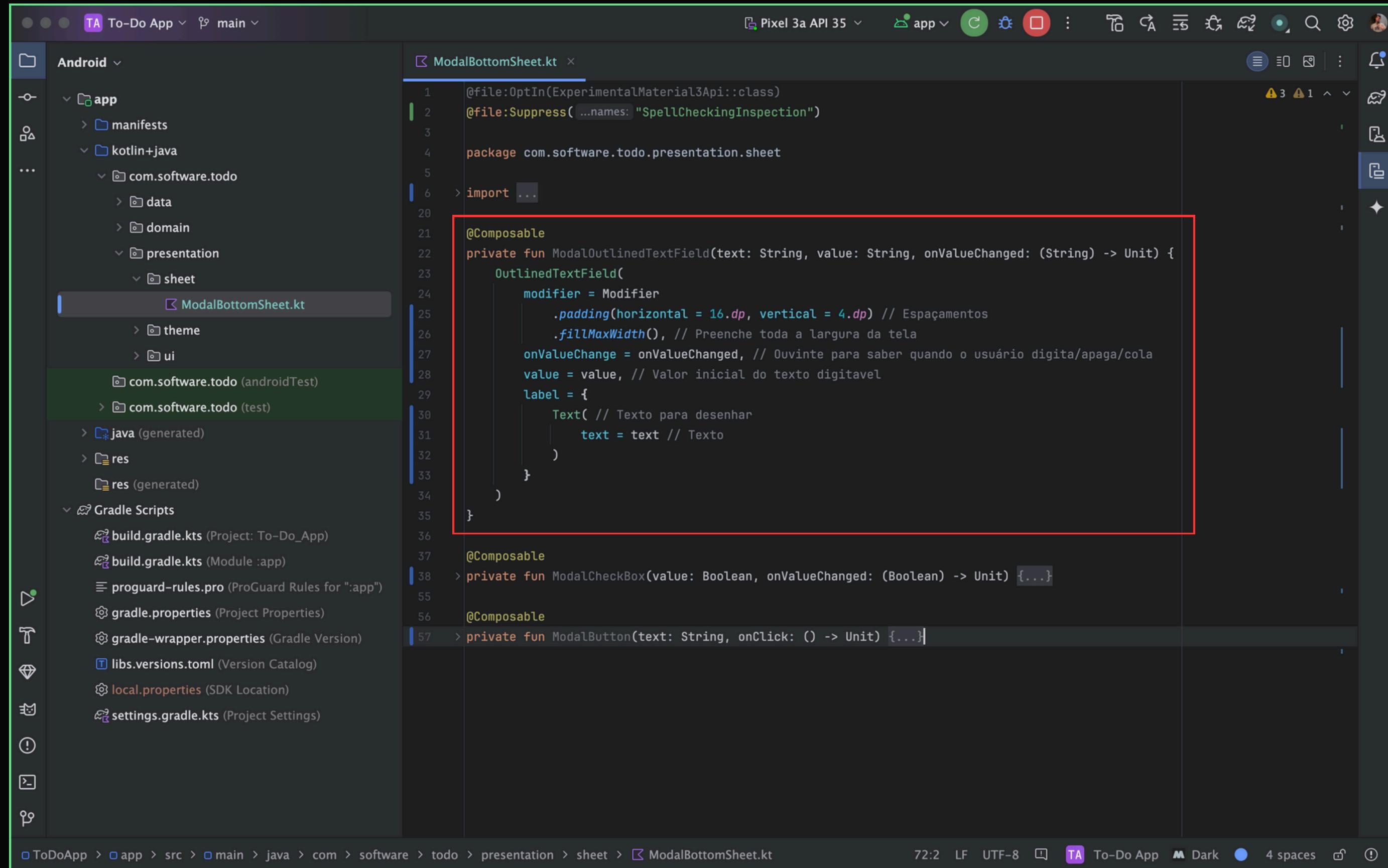
The screenshot shows the Android Studio interface with the project structure on the left and the code editor on the right. The code editor displays the file `ModalBottomSheet.kt`. A red box highlights the implementation of the `ModalCheckBox` function.

```
1  @file:OptIn(ExperimentalMaterial3Api::class)
2  @file:Suppress(...names: "SpellCheckingInspection")
3
4  package com.software.todo.presentation.sheet
5
6  import ...
7
8  @Composable
9  > private fun ModalOutlinedTextField(text: String, value: String, onValueChanged: (String) -> Unit) {...}
10
11
12  @Composable
13  private fun ModalCheckBox(value: Boolean, onValueChanged: (Boolean) -> Unit) {
14      Row( // Layout horizontal (linha)
15          verticalAlignment = Alignment.CenterVertically // Alinhamento dos componentes interno no centro
16      ) {
17          Checkbox( // Caixa de seleção
18              modifier = Modifier.padding(horizontal = 8.dp), // Espaçamentos
19              onCheckedChange = onValueChanged, // Quando ocorre uma ação de clique, ele retorna true ou false
20              checked = value // Valor inicial (true ou false)
21          )
22
23          Text( // Texto para desenhar
24              modifier = Modifier.padding(vertical = 8.dp), // Espaçamentos
25              fontWeight = FontWeight.Light, // Estilo de fonte
26              text = "Feito" // Texto
27          )
28      }
29  }
30
31
32  @Composable
33  > private fun ModalButton(text: String, onClick: () -> Unit) {...}
```

The project structure on the left shows the following directory tree:

- Android
- app
 - manifests
 - kotlin+java
 - com.software.todo
 - data
 - domain
 - presentation
 - sheet
- 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)

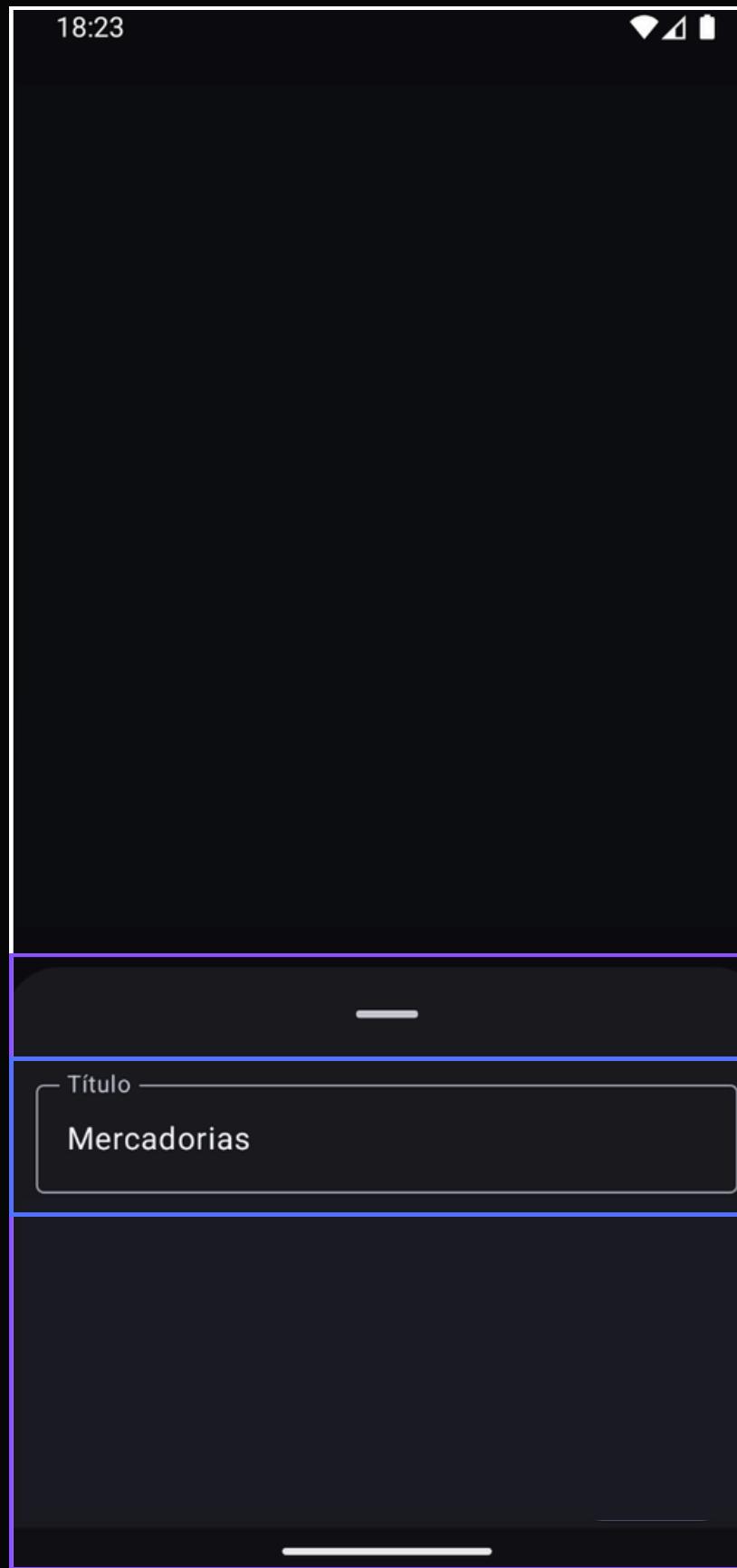




The screenshot shows the Android Studio interface with the project structure on the left and the code editor on the right. The code editor displays the file `ModalBottomSheet.kt` which contains the following code:

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

The code defines three composables: `ModalOutlinedTextField`, `ModalCheckBox`, and `ModalButton`. The first two are highlighted with a red box. The `ModalOutlinedTextField` function creates an `OutlinedTextField` with specific padding and a `fillMaxWidth` modifier. It also handles text changes and provides an initial value. The `ModalCheckBox` and `ModalButton` functions are partially visible below it.



OutlineTextField

ModalBottomSheet

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  private val defaultValue = Task()
10
11  /* ----- */
12
13  @Composable
14  fun TaskModalBottomSheet(
15      text: String,
16      task: Task = defaultValue,
17      onDismissRequest: () -> Unit,
18      onConfirm: (Task) -> Unit
19  ) {...}
20
21  /* ----- */
22
23  @Composable
24  > private fun ModalOutlinedTextField(text: String, value: String, onValueChanged: (String) -> Unit) {...}
25
26  @Composable
27  > private fun ModalCheckBox(value: Boolean, onValueChanged: (Boolean) -> Unit) {...}
28
29  @Composable
30  > private fun ModalButton(text: String, onClick: () -> Unit) {...}
```

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

The screenshot shows the Android Studio interface with the project 'To-Do App' open. The left sidebar displays the project structure under 'Android'. The main editor window shows the file 'ModalBottomSheet.kt'.

Project Structure:

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

ModalBottomSheet.kt File Content:

```
package com.software.todo.presentation.sheet

import ...

private val defaultValue = Task()

/*
 * -----
 */

@Composable
fun TaskModalBottomSheet(
    text: String,
    task: Task = defaultValue,
    onDismissRequest: () -> Unit,
    onConfirm: (Task) -> Unit
) {
    ModalBottomSheet( // Vai mostrar um dialog sobrepondo o nosso layout
        onDismissRequest = onDismissRequest
    )
}

/*
 * -----
 */

@Composable
private fun ModalOutlinedTextField(text: String, value: String, onValueChanged: (String) -> Unit) {...}

@Composable
private fun ModalCheckBox(value: Boolean, onValueChanged: (Boolean) -> Unit) {...}

@Composable
private fun ModalButton(text: String, onClick: () -> Unit) {...}
```

A red box highlights the code block from line 40 to line 50, specifically the call to `ModalBottomSheet` with its parameters.

Bottom Bar:

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

45:2 LF UTF-8 TA To-Do App M Dark 4 spaces

The screenshot shows the Android Studio interface with the project structure on the left and the code editor on the right. The code editor displays the file `ModalBottomSheet.kt` containing the function `TaskModalBottomSheet`. The code uses `remember` to track state and `ModalOutlinedTextField` and `ModalCheckBox` components.

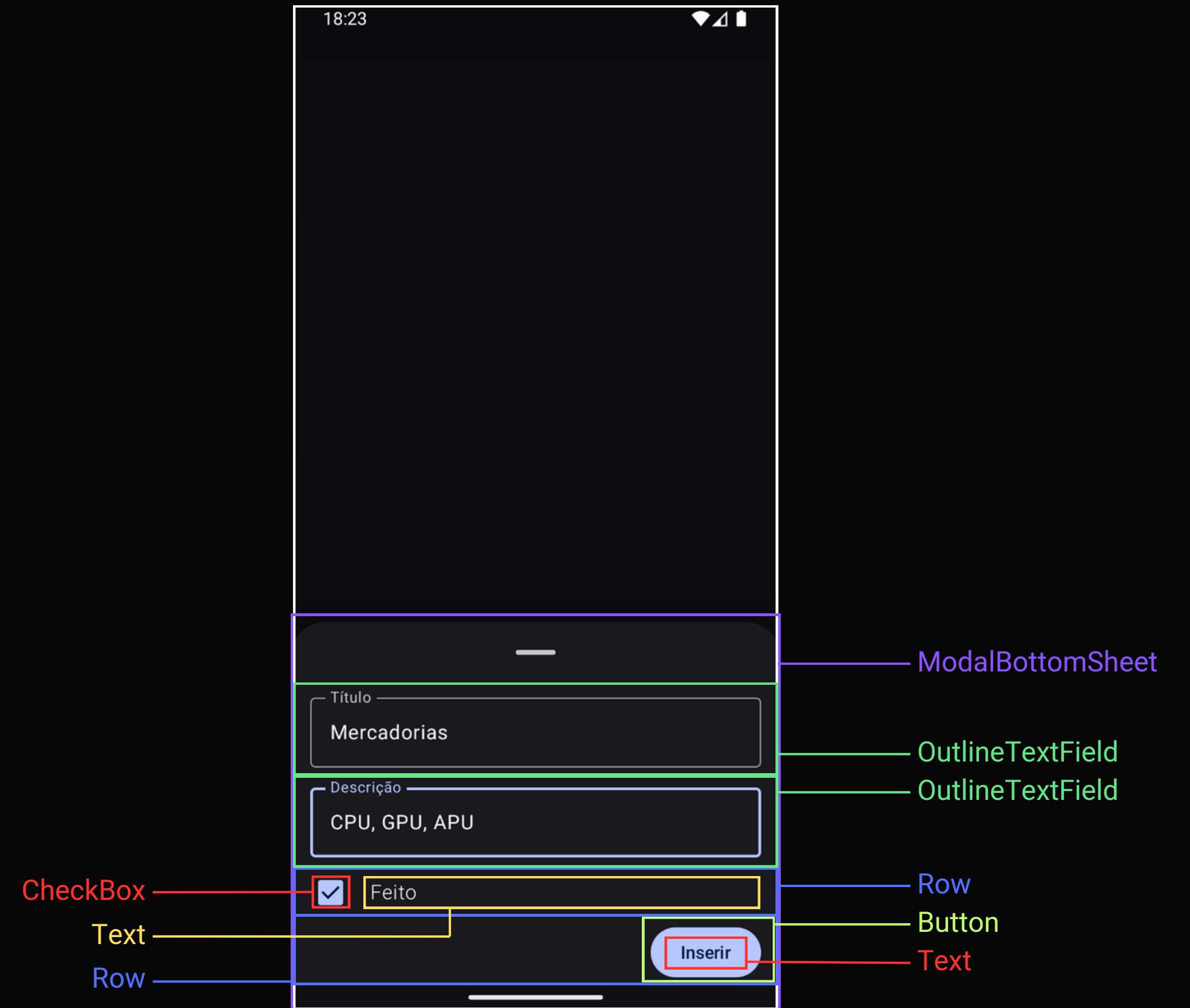
```
34     fun TaskModalBottomSheet(
35         ModalBottomSheet( // Vai mostrar um dialog sobrepondo o nosso layout
36             onDismissRequest = onDismissRequest
37         ) { // Por padrão, o conteúdo está dentro de um 'Column'
38             var title by remember { mutableStateOf(value = task.title)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

The code editor has three sections of code highlighted with red boxes:

- The first section handles the `title` state, using `remember` to track the initial value from `task.title` and updating it via `onValueChanged`.
- The second section handles the `description` state, using `remember` to track the initial value from `task.description` and updating it via `onValueChanged`.
- The third section handles the `checked` state, using `remember` to track the initial value from `task.checked` and updating it via `onValueChanged`.

The screenshot shows the Android Studio interface with the project structure on the left and the code editor on the right. The code editor displays the file `ModalBottomSheet.kt` which contains the following Kotlin code:

```
fun TaskModalBottomSheet() { // Por padrão, o conteúdo está dentro de um 'Column'  
    var description by remember { mutableStateOf(value = task.description) } // remember significa que quando o  
    // valor alterar, esse se manterá,  
    // ou seja, não vai ser resetado para  
    // o valor padrão  
  
    ModalOutlinedTextField( // Função que criamos anteriormente  
        text = "Descrição",  
        onValueChanged = { newValue -> description = newValue }, // Alteramos o valor de 'description'  
        value = description // Valor inicial // para o recebido (newValue)  
    )  
  
    var checked by remember { mutableStateOf(value = task.checked) } // remember significa que quando o  
    // valor alterar, esse se manterá,  
    // ou seja, não vai ser resetado para  
    // o valor padrão  
  
    ModalCheckBox( // Função que criamos anteriormente  
        onValueChanged = { newValue -> checked = newValue }, // Alteramos o valor de 'checked'  
        value = checked // Valor inicial // para o recebido (newValue)  
    )  
  
    ModalButton(text = text) { // Função que criamos anteriormente  
        onDismissRequest() // Ao clicar no botão, queremos ocultar nosso modal  
        onConfirm() // Nova tarefa para inserir ou a tarefa atual com os valores editados  
            task.copy(title = title, description = description, checked = checked) // copy é exclusivo de data class  
        )  
    }  
  
    Spacer( // Layout simples, serve apenas para adicionar espaçamento  
        modifier = Modifier.navigationBarsPadding() // Modificador do compose para adicionar o  
        // espaço da barra de navegação  
    )  
}
```



```
49  class TaskActivity : ComponentActivity() {  
50  
51      @Composable  
52      private fun TaskScreen() {  
53          val scrollBehavior = TopAppBarDefaults.pinnedScrollBehavior() // Observa sempre que o  
54                                         // usuário rola a lista de tarefas  
55  
56          Scaffold( // Scaffold tem a estrutura base para criarmos o nosso layout  
57              modifier = Modifier.nestedScroll(connection = scrollBehavior.nestedScrollConnection), // Add o observer  
58              floatingActionButton = { TaskActionButton() }, // Lambda para incluir o botão flutuante  
59              topBar = { TaskTopBar(scrollBehavior) } // Lambda para incluir a barra de ações  
60          ) { paddingValues ->  
61              val items by viewModel.tasks.collectAsState() // Busca as tarefas cadastradas no bd  
62  
63              if (  
64                  items.isNotEmpty() // Se a lista não estiver vazia, queremos mostra as tarefas  
65              ) {  
66                  TaskListScreen(items, paddingValues) // Lista de tarefas  
67              } else { // Caso contrário  
68                  TaskEmptyScreen(paddingValues) // Mensagem informativa de que não há nenhuma tarefa  
69              }  
70          }  
71      }  
72  
73      @Composable  
74      private fun TaskActionButton() {...}  
75  
76      @Composable  
77      private fun TaskTopBar(scrollBehavior: TopAppBarScrollBehavior) {...}  
78  
79      /* ----- */  
80  
81      @Composable // Indica para o compilador que essa função é um layout  
82      private fun TaskListScreen(items: List<Task>, padding: PaddingValues) {...}  
83  
84      @Composable
```

The screenshot shows the Android Studio interface with the project structure on the left and the code editor on the right.

Project Structure:

- Project: To-Do App
- Module: main
- Source Sets:
 - Android
 - Java
 - res
 - generated
- Gradle Scripts:
 - build.gradle.kts (Project: To-Do_App)
 - build.gradle.kts (Module :app)
 - proguard-rules.pro
 - gradle.properties
 - gradle-wrapper.properties
 - libs.versions.toml
 - local.properties
 - settings.gradle.kts

Code Editor (TaskActivity.kt):

```
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
```

Two sections of the code are highlighted with red boxes:

- The first section highlights the declaration of `val state by viewModel.state.collectAsState()` and its usage in the `if` statement.
- The second section highlights the `when (state)` block, specifically the `is TaskSheet.Insert ->`, `is TaskSheet.Update ->`, and `is TaskSheet.Closed ->` branches.

To-Do App main Pixel 3a API 35 app C A E F K Q S

Android

TaskDao
TaskDatabase
domain
Task
TaskSheet
presentation
sheet
ModalBottomSheet.kt
theme
Theme.kt
Type.kt
ui
TaskActivity
TaskViewModel
comsoftware.todo (androidTest)
comsoftware.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

```
51     class TaskActivity : ComponentActivity() {
83         private fun TaskScreen() {
91             ) { paddingValues ->
102                 // Quando em kotlin é igual o IF, IF ELSE
103                 // 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                                 onConfirm = { task ->
111                                     }
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                         onConfirm = { task ->
124                             },
125                             task = update.task
126                         )
127                     }
128                 }
129             }
130             is TaskSheet.Closed -> { // Closed --> Execute esse trecho de código
131                 // Não faça nada no status de fechado
132             }
133             }
134             }
135             }
136             }
```

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

130:1 LF UTF-8 TA To-Do App M Dark 4 spaces

To-Do App main

Pixel 3a API 35 app C A E F M Q

Android TaskActivity.kt

```
51     class TaskActivity : ComponentActivity() {
83         private fun TaskScreen() {
91             ) { paddingValues ->
103             // Quando 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             is TaskSheet.Closed -> { // Closed --> Execute esse trecho de código
131                 // Não faça nada no status de fechado
132             }
133         }
134     }
135 }
```

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)

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

130:1 LF UTF-8 TA To-Do App M Dark 4 spaces

The screenshot shows the Android Studio interface with the project structure on the left and the code editor on the right.

Project Structure:

- Project: To-Do App
- Module: main
- Source Sets:
 - Android
 - 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)

Code Editor (TaskActivity.kt):

```
51 class TaskActivity : ComponentActivity() {  
79     /* ----- */  
80     @Composable  
81     private fun TaskScreen() {...}  
82     @Composable  
83     private fun TaskActionButton() {  
84         FloatingActionButton( // Botão flutuante com bordas arredondadas  
85             onClick = { // Lambda contendo a ação de click do usuário  
86                 viewModel.update { TaskSheet.Insert }  
87             }  
88         )  
89     }  
90     /* ----- */  
91     @Composable  
92     private fun TaskTopBar(scrollBehavior: TopAppBarScrollBehavior) {...}  
93     /* ----- */  
94     @Composable // Indica para o compilador que essa função é um layout  
95     private fun TaskListScreen(items: List<Task>, padding: PaddingValues) {...}  
96     @Composable  
97     private fun TaskEmptyScreen(padding: PaddingValues) {...}  
98     @Composable  
99     private fun TaskItem(task: Task) {...}  
100 }
```

A red box highlights the lambda expression `onClick = { viewModel.update { TaskSheet.Insert } }` at line 142, which triggers the insertion of a new task into the database.

Bottom Status Bar:

- ToDoApp > app > src > main > java > com > software > todo > presentation > ui > TaskActivity > TaskActionButton
- 142:54 LF UTF-8
- TA To-Do App
- Dark
- 4 spaces

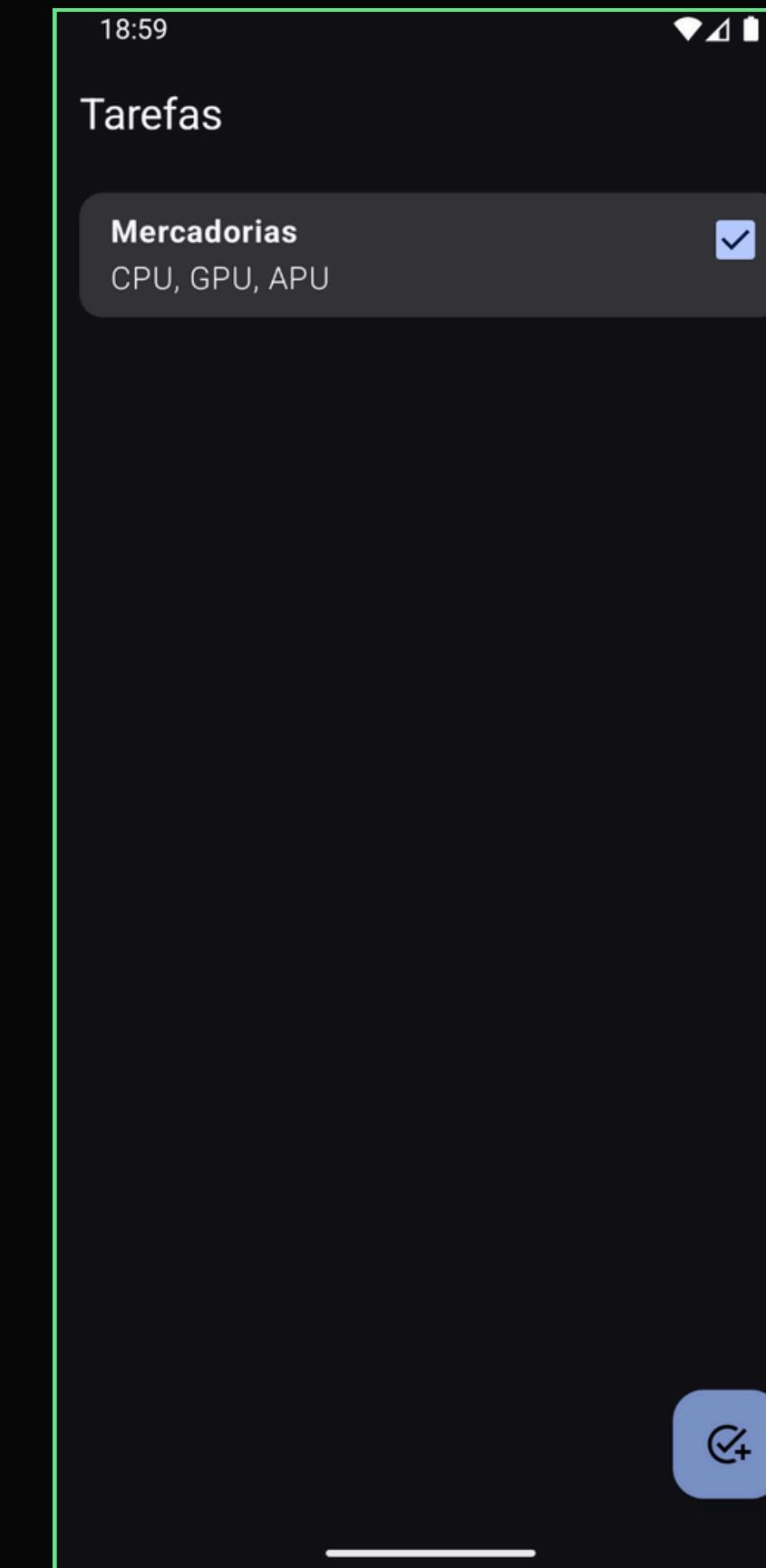
To-Do App main Pixel 3a API 35 app C A E F M Q

Android TaskActivity.kt

```
51 class TaskActivity : ComponentActivity() {
79     /* ----- */
80     @Composable
81     private fun TaskScreen() {...}
82
83     @Composable
84     private fun TaskActionButton() {...}
85
86     @Composable
87     private fun TaskTopBar(scrollBehavior: TopAppBarScrollBehavior) {...}
88
89     /* ----- */
90
91     @Composable // Indica para o compilador que essa função é um layout
92     private fun TaskListScreen(items: List<Task>, padding: PaddingValues) {...}
93
94     @Composable
95     private fun TaskEmptyScreen(padding: PaddingValues) {...}
96
97     @Composable
98     private fun TaskItem(task: Task) {
99         Card( // Layout com bordas e sombra (efeito flutuante)
100            modifier = Modifier
101               .padding(horizontal = 16.dp, vertical = 8.dp) // Espaçamentos
102               .combinedClickable( // Clique longo (pressiona e segura) e clique (clica e solta)
103                  onLongClick = { viewModel.delete(task) }, // Deleta a tarefa
104                  onClick = {
105                      viewModel.update { TaskSheet.Update(task) }
106                  }
107        )
108
109        Row { // Tudo o que estiver dentro do lambda vai ser adicionado na horizontal (linha)
110            Column( // Tudo o que estiver dentro do lambda vai ser adicionado na vertical (coluna)
111                modifier = Modifier
112                   .padding(horizontal = 16.dp, vertical = 8.dp) // Espaçamentos
113                   .weight(weight = 1F) // Significa que deve preencher a
114            )
115        }
116    }
117
118    Row { // Tudo o que estiver dentro do lambda vai ser adicionado na horizontal (linha)
119        Column( // Tudo o que estiver dentro do lambda vai ser adicionado na vertical (coluna)
120            modifier = Modifier
121               .padding(horizontal = 16.dp, vertical = 8.dp) // Espaçamentos
122               .weight(weight = 1F) // Significa que deve preencher a
123        )
124    }
125}
```

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

207:68 LF UTF-8 TA To-Do App M Dark 4 spaces !



Obrigado!

Baxe o projeto completo em:

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

