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 extensivel
- 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.

```kotlin
@file:Suppress( ...names: "SpellCheckingInspection", "KotlinConstantConditions")

package com.software.todo

fun main() {
    // Ao atribuir um valor, não é possível alterar esse valor
    val myVal1: Int = 0

    print("Valor: $myVal1") // Vai printar 0

    // Compilador não deixa eu alterar o valor devido essa variavel ser imutavel
    myVal1 = 1



    // Ao atribuir um valor, eu posso alterar esse valor depois
    var myVal2: Int = 0

    print("Valor: $myVal2") // Vai printar 0

    // Estou alterando o valor da variavel para 1
    myVal2 = 1

    print("Valor: $myVal2") // Vai printar 1
}
```

```kotlin
// Classe com tipos de instâncias deferentes
sealed class Operation {

    // Representação de um objeto sem dados, pertencente ao Operation
    data object Idle : Operation()

    // Representação de um objeto com dados do tipo [Int], pertencente ao Operation
    data class Plus(val a: Int, val b: Int) : Operation()

    // Representação de um objeto com dados do tipo [Int], pertencente ao Operation
    data class Minus(val a: Int, val b: Int) : Operation()

}

// Representação de um objeto com dados
data class Numbers(
    val a: Int,
    val b: Int
)

// Uma classe que contém métodos responsável por fazer somas e divisões de valores/números
class Calculator {

    // Função responsável por somar dois números e retornar a soma
    fun plus(numbers: Numbers): Int { // Declaramos o tipo de retorno da função
        return numbers.a + numbers.b // [return] palavra reservada do kotlin
    }

    // Função responsável por somar dois números do tipo [Int] e retornar a soma
    fun minus(a: Int, b: Int): Int { // Declaramos o tipo de retorno da função
        return a - b // [return] palavra reservada do kotlin
    }

}
```

```kotlin
// Uma classe que contém métodos responsável por fazer somas e divisões
// dos valores passados no construtor 'Calculate(1,1)'
class Calculate(
    private val a: Int,
    private val b: Int
) {

    // Função responsável por multiplicar dois números do tipo [Int] e retornar a multiplicação
    private fun multiply(a: Int, b: Int): Int { // Declaramos o tipo de retorno da função
        return a * b // [return] palavra reservada do kotlin
    }

    // Função responsável por somar dois números do tipo [Int] e retornar a soma
    fun plus(): Int { // Declaramos o tipo de retorno da função
        return a + b // [return] palavra reservada do kotlin
    }

    // Função responsável por somar dois números e retornar a soma
    fun minus(): Int { // Declaramos o tipo de retorno da função
        return a - b // [return] palavra reservada do kotlin
    }

}
```

```kotlin
fun main() {
    // Instanciar uma [data class]
    val numbers = Numbers( a: 1, b: 2)

    // Para acessar os valores/métodos da [data class]
    numbers.a
    numbers.b

    // Para instanciar uma [class]
    val calculator = Calculator()

    // Para accessar as funções da [class]
    calculator.plus(numbers)
    calculator.minus( a: 1, b: 2)

    // Para instanciar uma [class] que tem parâmetros no construtor
    val calculate = Calculate( a: 1, b: 2)

    // OBS: métodos/variaveis privadas não é possível acessar fora da [class]
    calculate.multiply( a: 1, b: 2)
}
```

```kotlin
// Classe com herança de [Calculator] com novas funcionalidades da nossa calculadora
// OBS: [class] sem a palavra reservada 'open' não podem ser herdadas
class CalculatorV2 : Calculator() {

    // Função responsável por multiplicar dois números do tipo [Int] e retornar a multiplicação
    fun multiply(a: Int, b: Int): Int { // Declaramos o tipo de retorno da função
        return a * b // [return] palavra reservada do kotlin
    }

    // Função responsável por dividir dois números do tipo [Int] e retornar a divisão
    fun divide(a: Int, b: Int): Int { // Declaramos o tipo de retorno da função
        return a / b // [return] palavra reservada do kotlin
    }

}
```

```kotlin
// Uma classe que contém métodos responsável por fazer somas e divisões de valores/números
open class Calculator {

    // Função responsável por somar dois números e retornar a soma
    fun plus(numbers: Numbers): Int { // Declaramos o tipo de retorno da função
        return numbers.a + numbers.b // [return] palavra reservada do kotlin
    }

    // Função responsável por somar dois números do tipo [Int] e retornar a soma
    fun minus(a: Int, b: Int): Int { // Declaramos o tipo de retorno da função
        return a - b // [return] palavra reservada do kotlin
    }

}
```

```kotlin
// Classe com herança de [Calculator] com novas funcionalidades da nossa calculadora
// OBS: Agora é possível herdar a [class] 'Calculator' uma vez que ela é aberta
class CalculatorV2 : Calculator() {

    // Função responsável por multiplicar dois números do tipo [Int] e retornar a multiplicação
    fun multiply(a: Int, b: Int): Int { // Declaramos o tipo de retorno da função
        return a * b // [return] palavra reservada do kotlin
    }

    // Função responsável por dividir dois números do tipo [Int] e retornar a divisão
    fun divide(a: Int, b: Int): Int { // Declaramos o tipo de retorno da função
        return a / b // [return] palavra reservada do kotlin
    }

    // Função que faz uma calculo complexo
    fun complexCalculus(a: Int, b: Int): Int { // Declaramos o tipo de retorno da função
        val result1 = multiply(a,b)
        val result2 = minus(result1,b)
        val result3 = divide(result2,a)
        return result3 + result1 // [return] palavra reservada do kotlin
    }
```

```
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 }
```

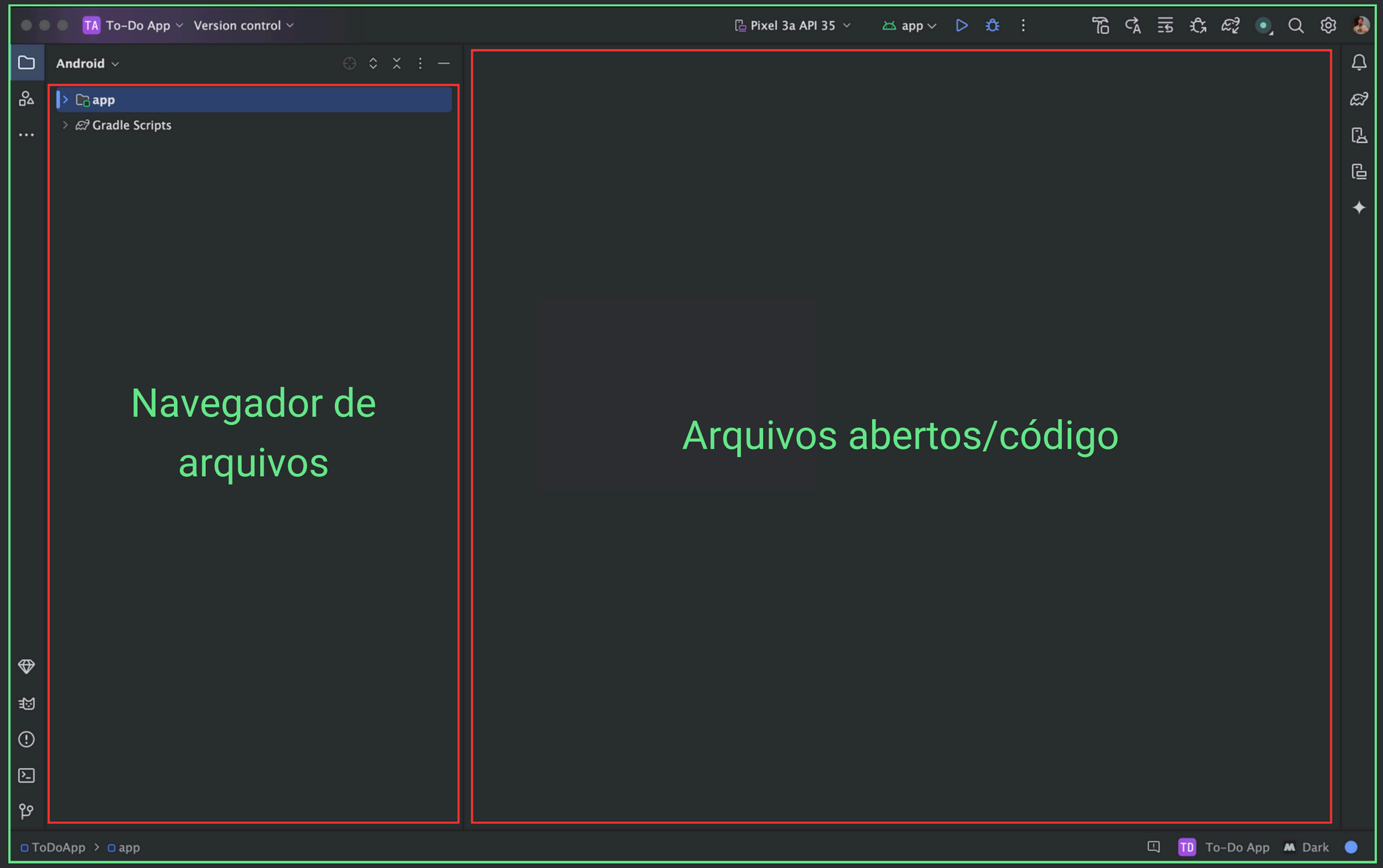
```kotlin
fun main() {
    // Função que tem como parametro um lambda '(result: Int) -> Unit'
    calculate1( a: 1, b: 2) { result ->
        val a = result // Aqui eu consigo acessar o resultado que obtive do calculo da função
    }

    // Função que tem como parametro um lambda '() -> Int'
    calculate2( a: 1, b: 2) {
        1 // No final do bloco, é necessário retornar um valor

        // OBS: é possível retornar um valor assim também: 'return@calculate2 1'
        // mas o compilar é esperto o suficiente e não precisa escrever o return
    }

    // Função que tem como parametro um lambda '() -> Unit'
    calculate3( a: 1, b: 2) {
        // Não retorna nada, logo, apenas executa esse trecho de código
    }
}
```

```kotlin
// Activity é uma [class] do Android usada para instanciar screens (telas)
class MyActivity : Activity() {

    // Estamos sobreescrevendo a função aberta 'onCreate' (aqui é onde desenhamos o layout)
    // OBS: override significa que vamos sobreescrever a função
    // que já existe dentro da [class] que herdamos
    override fun onCreate(savedInstanceState: Bundle?) {
        // OBS: super.nomeDaFunção significa que primeiro vamos executar o código 'onCreate'
        // dentro da [class] 'Activity' para depois executar o nosso código
        // que vamos escrever
        super.onCreate(savedInstanceState)
        // código
        (...)
    }

}
```

Android Studio

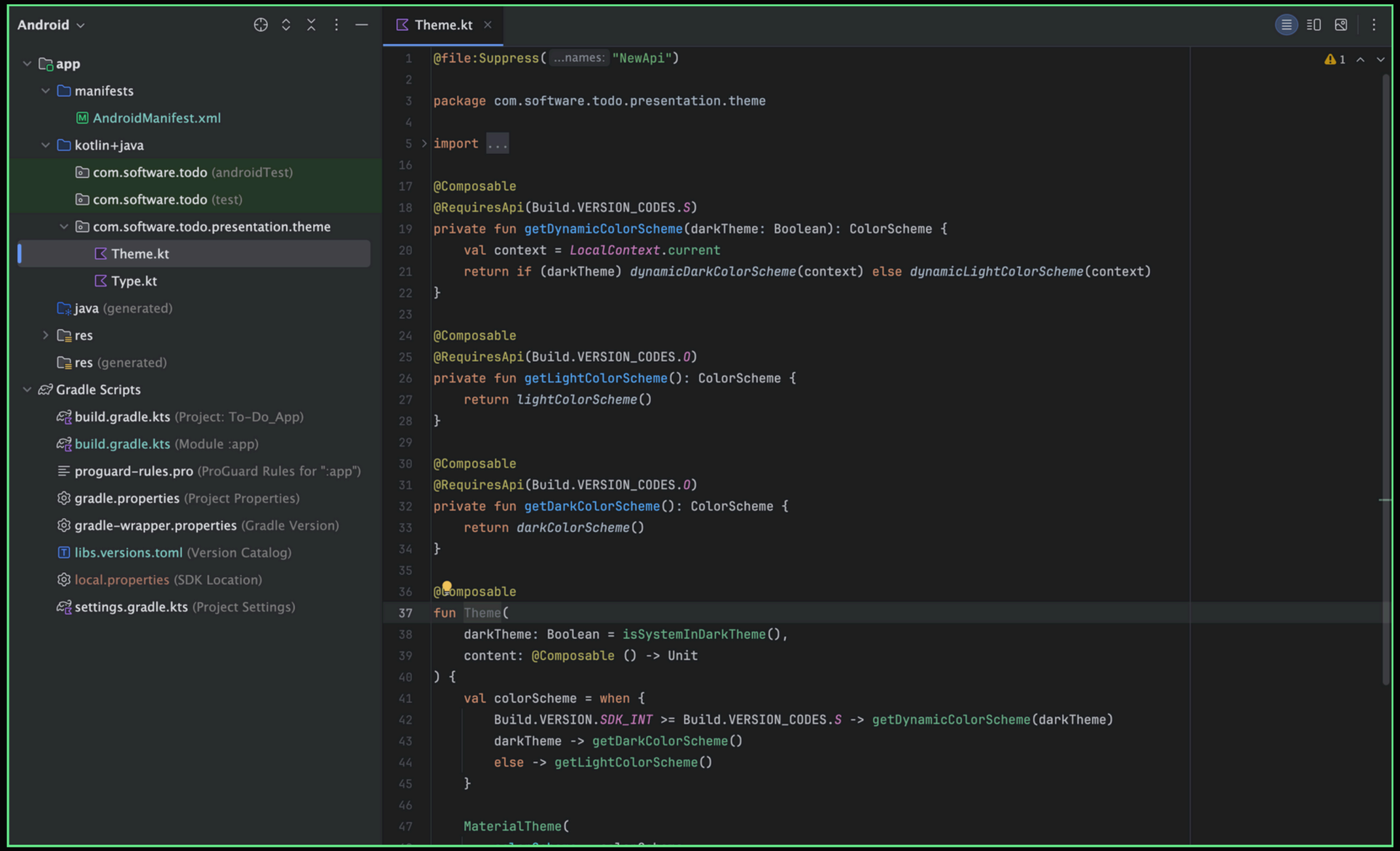
BY GOOGLE

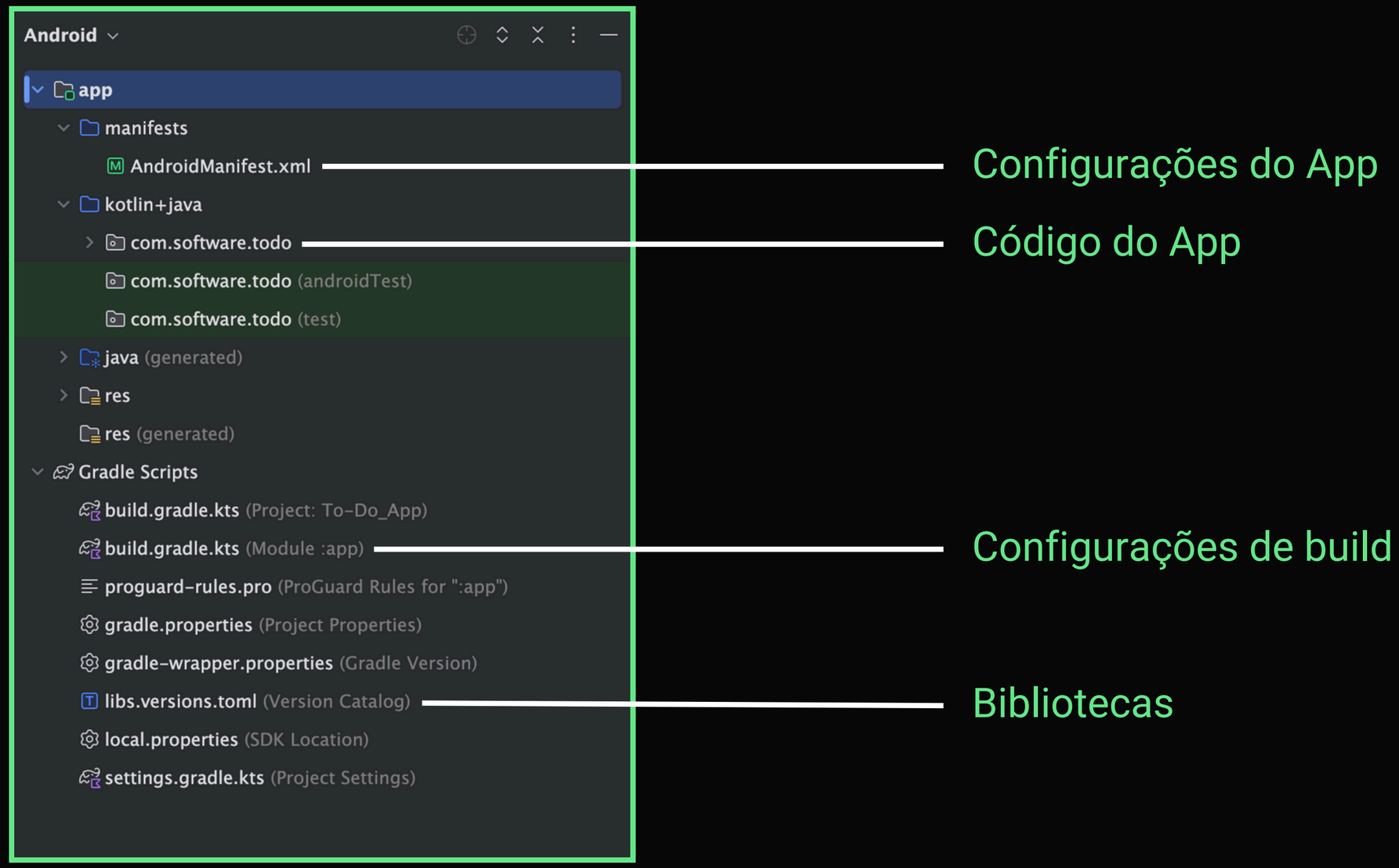
Android Studio é um ambiente de desenvolvimento integrado para desenvolver para o sistema Android, baseado no software IntelliJ IDEA da empresa JetBrains

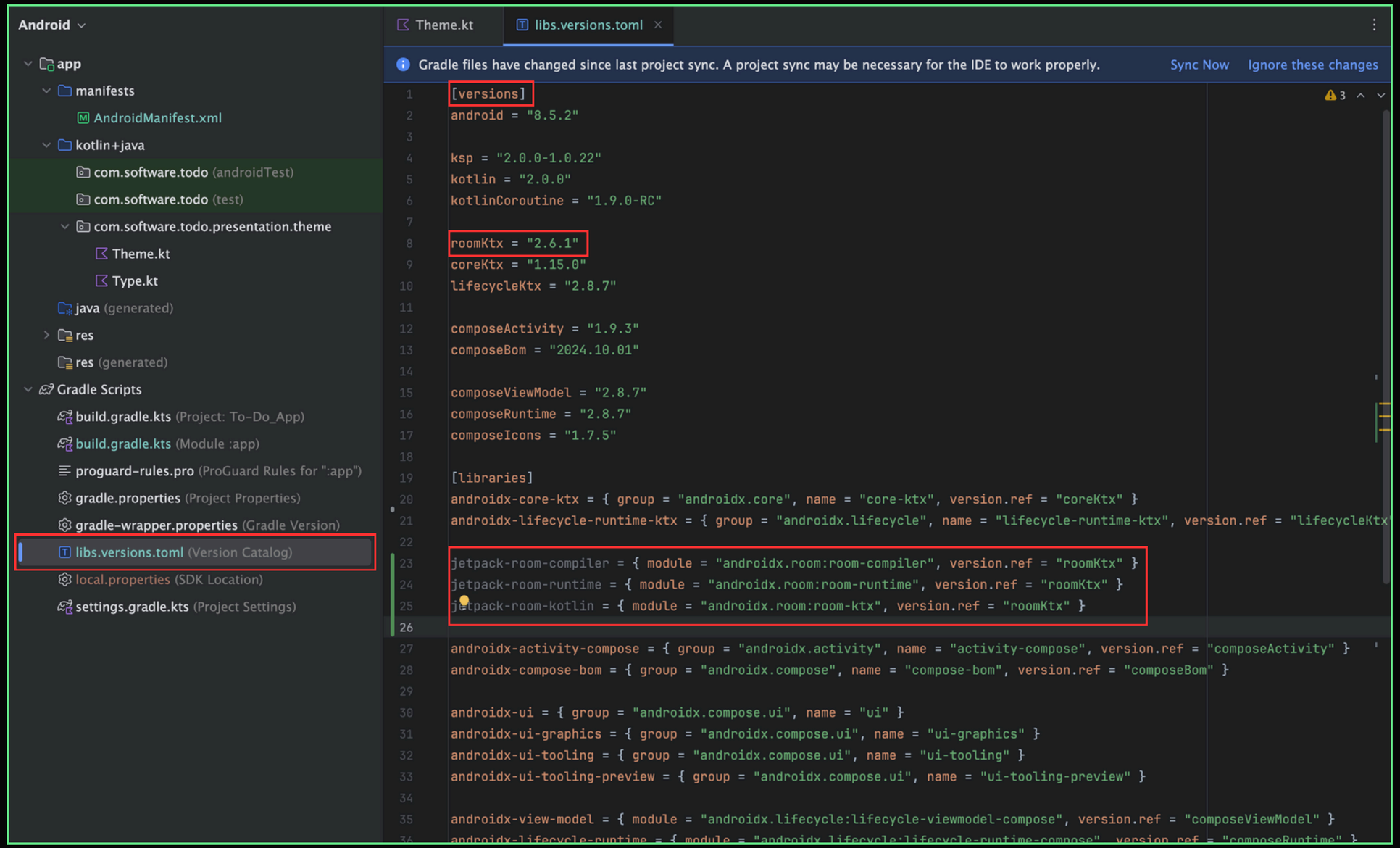

To-Do App
Version control
Pixel 3a API 35
app
Android
app
Gradle Scripts
Navegador de arquivos
Arquivos abertos/código
ToDoApp
app
To-Do App
Dark

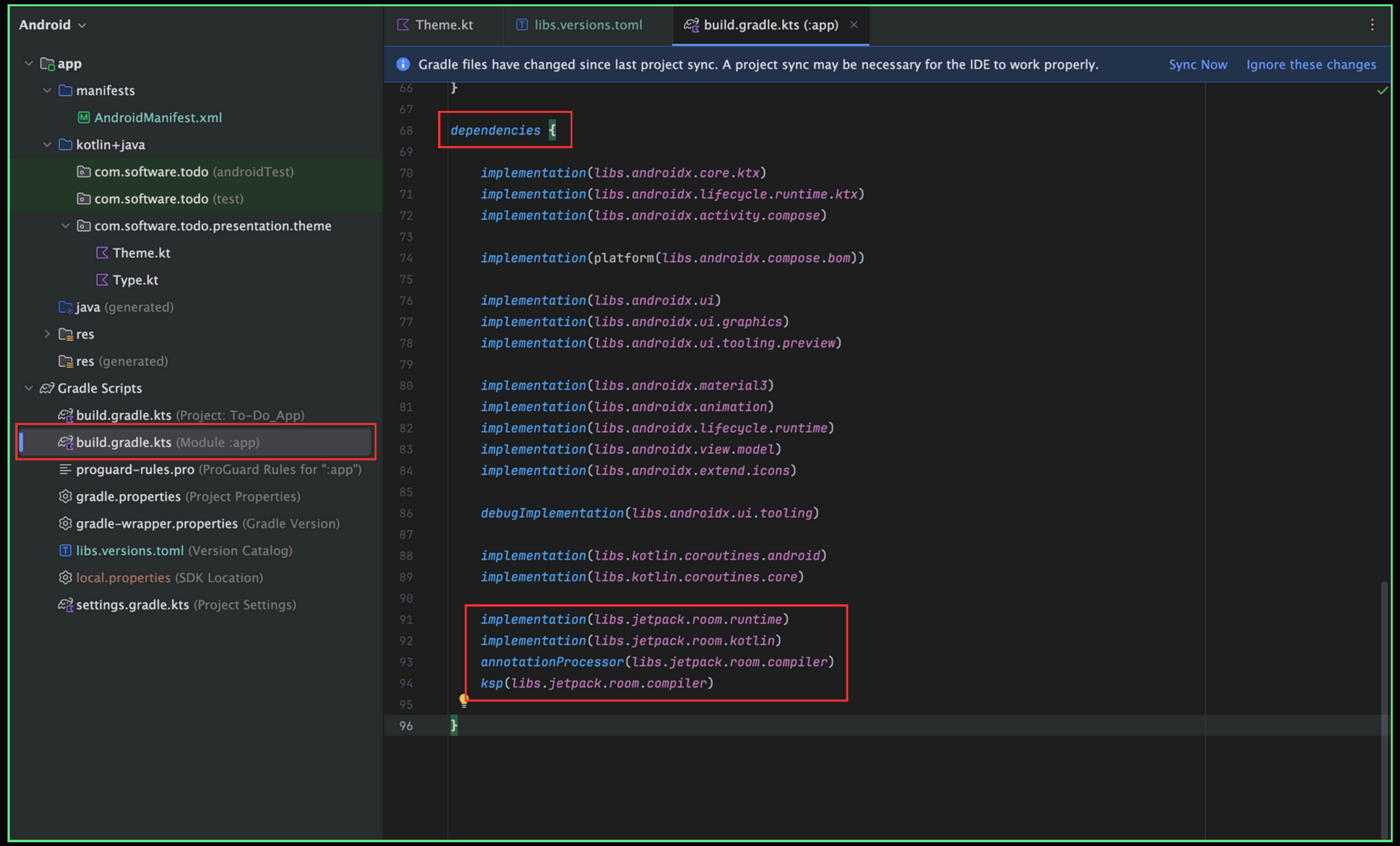
Projeto

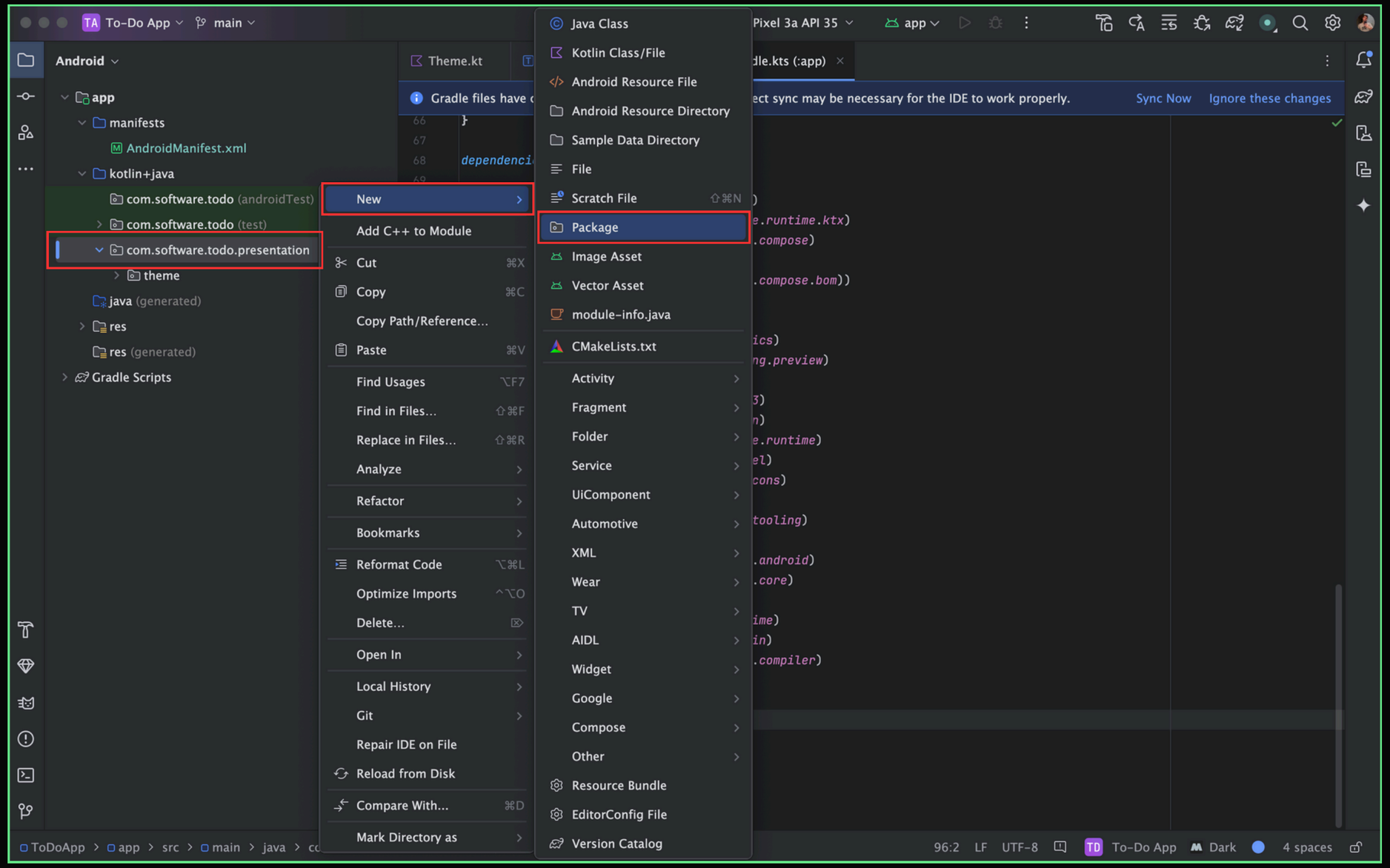
To-Do Application

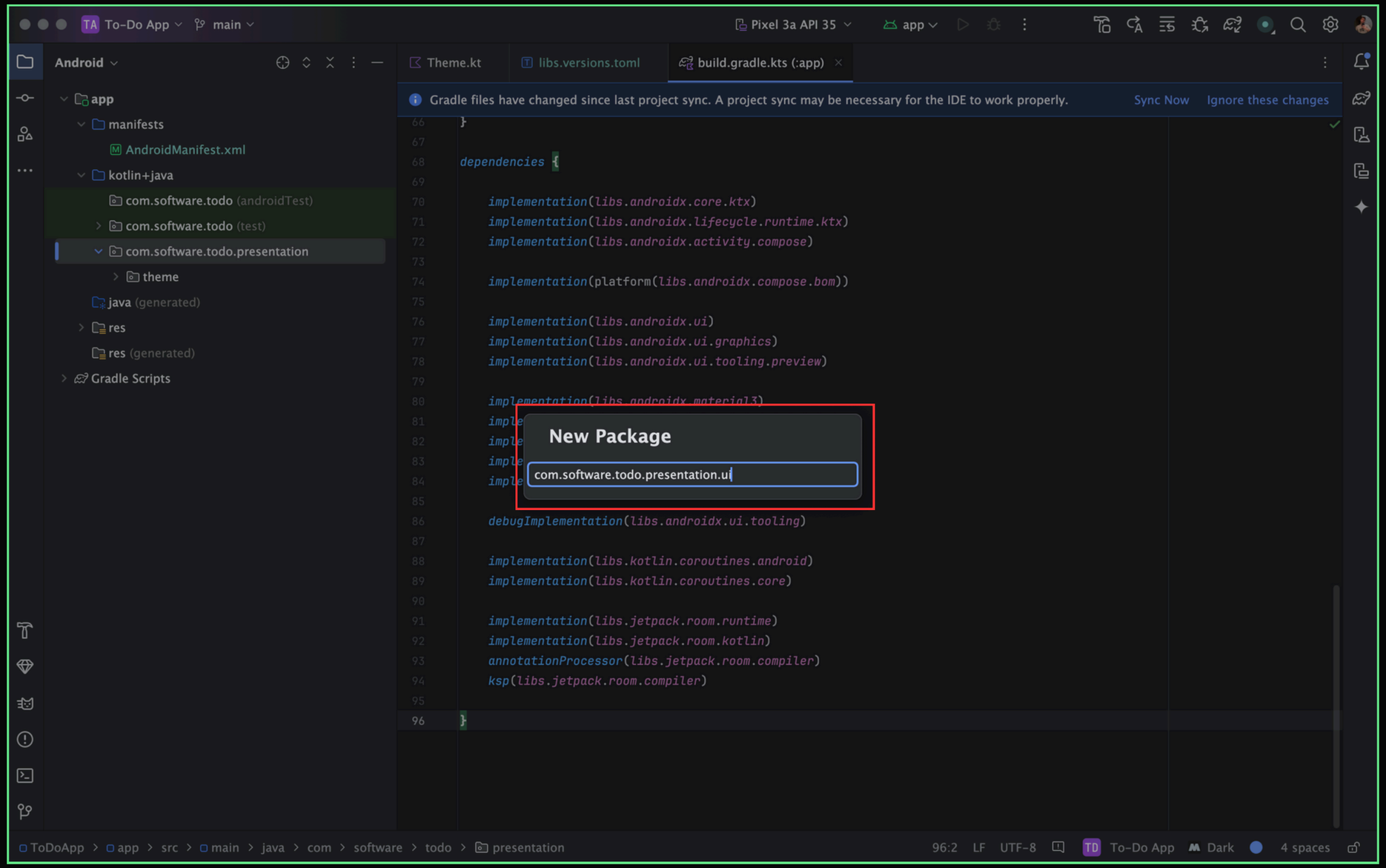

Android
app
manifests
AndroidManifest.xml
kotlin+java
com.software.todo (androidTest)
com.software.todo (test)
com.software.todo.presentation.theme
Theme.kt
Type.kt
java (generated)
res
res (generated)
Gradle Scripts
build.gradle.kts (Project: To-Do_App)
build.gradle.kts (Module :app)
proguard-rules.pro (ProGuard Rules for ":app")
gradle.properties (Project Properties)
gradle-wrapper.properties (Gradle Version)
libs.versions.toml (Version Catalog)
local.properties (SDK Location)
settings.gradle.kts (Project Settings)
Theme.kt
@file:Suppress(...names: "NewApi")
package com.software.todo.presentation.theme
import ...
@Composable
@RequiresApi(Build.VERSION_CODES.S)
private fun getDynamicColorScheme(darkTheme: Boolean): ColorScheme {
val context = LocalContext.current
return if (darkTheme) dynamicDarkColorScheme(context) else dynamicLightColorScheme(context)
}
@Composable
@RequiresApi(Build.VERSION_CODES.O)
private fun getLightColorScheme(): ColorScheme {
return lightColorScheme()
}
@Composable
@RequiresApi(Build.VERSION_CODES.O)
private fun getDarkColorScheme(): ColorScheme {
return darkColorScheme()
}
@Composable
fun Theme(
darkTheme: Boolean = isSystemInDarkTheme(),
content: @Composable () -> Unit
) {
val colorScheme = when {
Build.VERSION.SDK_INT >= Build.VERSION_CODES.S -> getDynamicColorScheme(darkTheme)
darkTheme -> getDarkColorScheme()
else -> getLightColorScheme()
}
MaterialTheme(

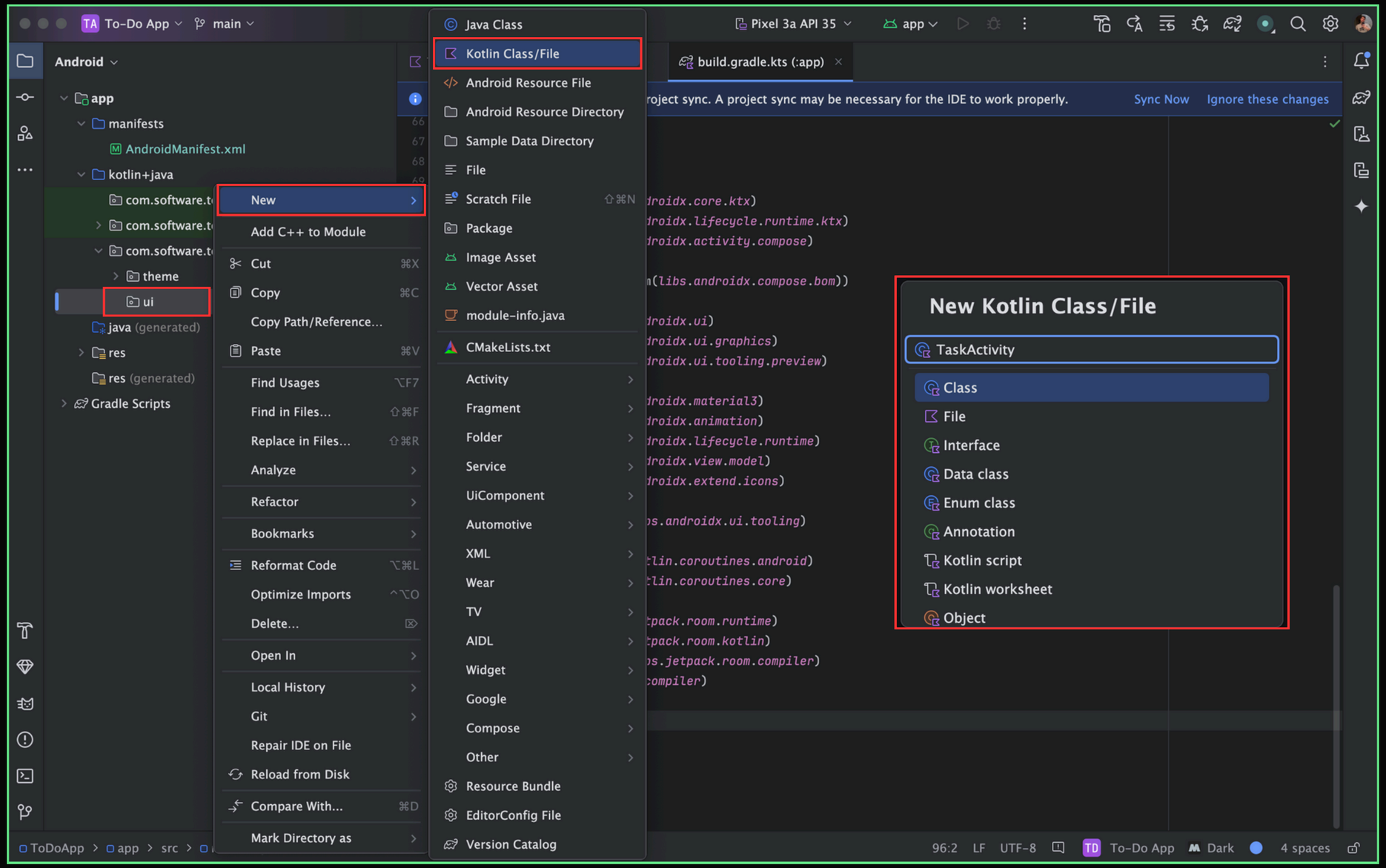

Android
app
manifests
AndroidManifest.xml
kotlin+java
com.software.todo
com.software.todo (androidTest)
com.software.todo (test)
java (generated)
res
res (generated)
Gradle Scripts
build.gradle.kts (Project: To-Do_App)
build.gradle.kts (Module :app)
proguard-rules.pro (ProGuard Rules for ":app")
gradle.properties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

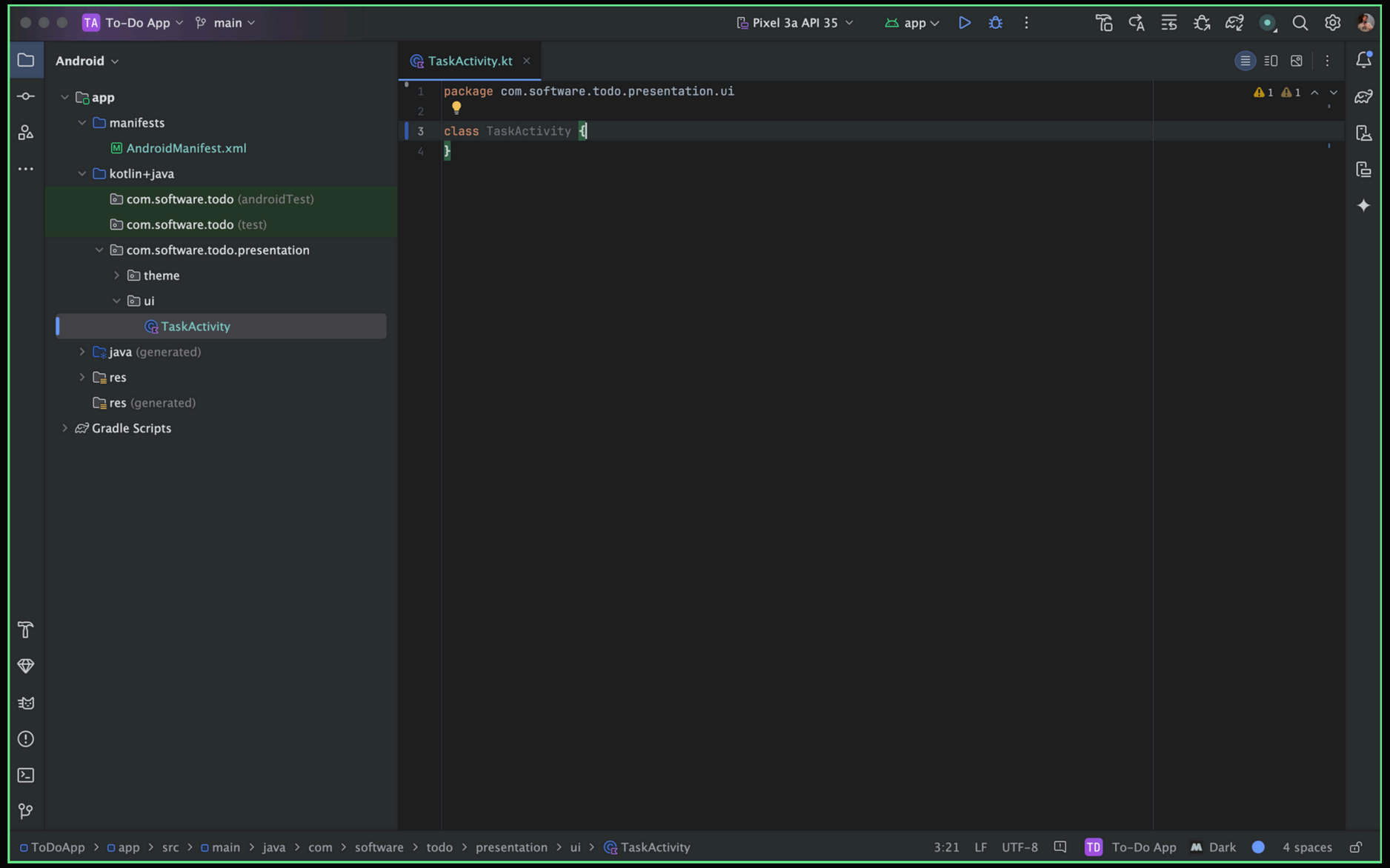

Android
app
manifests
AndroidManifest.xml
kotlin+java
com.software.todo (androidTest)
com.software.todo (test)
com.software.todo.presentation.theme
Theme.kt
Type.kt
java (generated)
res
res (generated)
Gradle Scripts
build.gradle.kts (Project: To-Do_App)
build.gradle.kts (Module :app)
proguard-rules.pro (ProGuard Rules for ":app")
gradle.properties (Project Properties)
gradle-wrapper.properties (Gradle Version)
libs.versions.toml (Version Catalog)
local.properties (SDK Location)
settings.gradle.kts (Project Settings)
Theme.kt
libs.versions.toml
Gradle files have changed since last project sync. A project sync may be necessary for the IDE to work properly.
Sync Now
Ignore these changes
[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: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" }

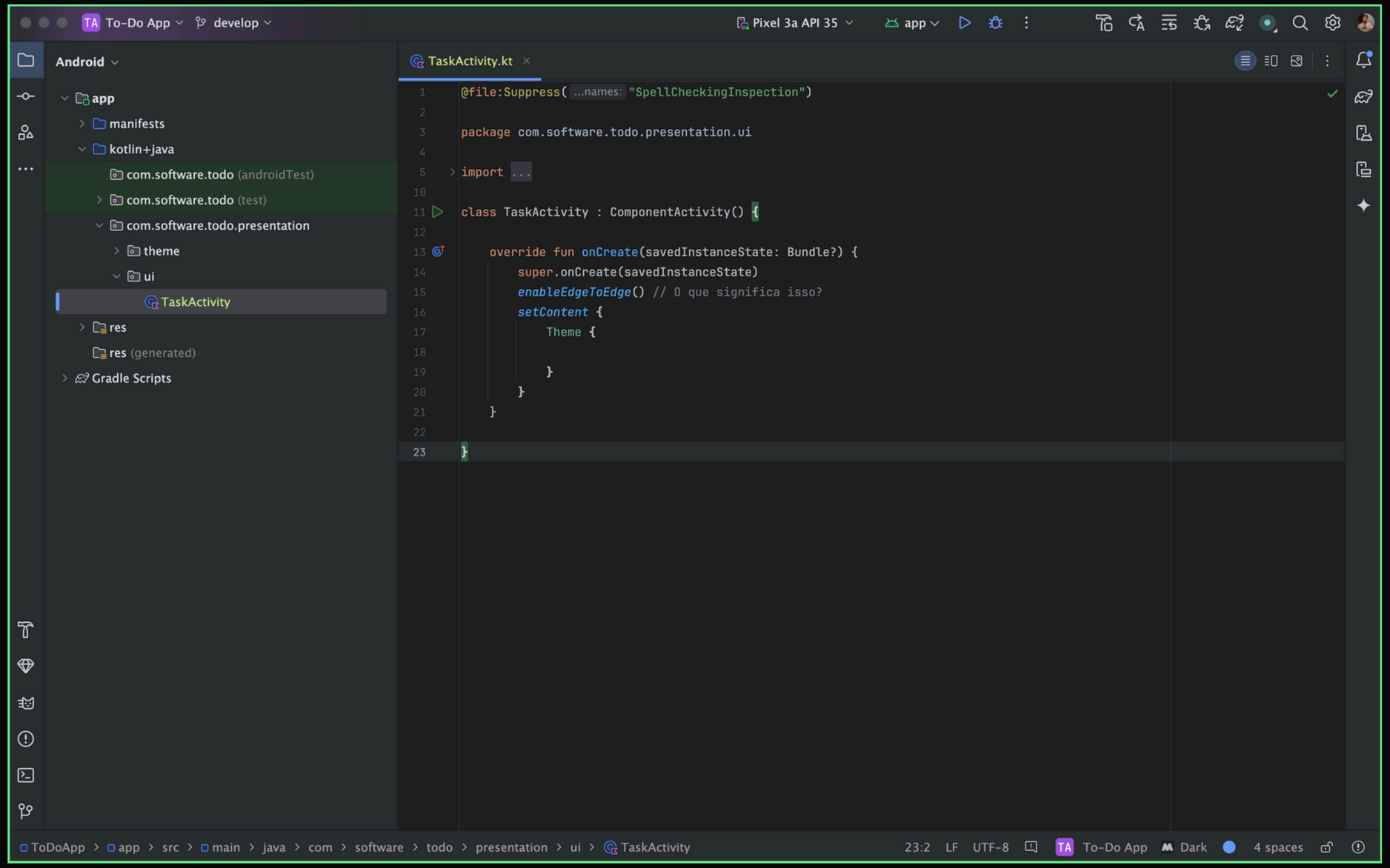

Android
app
manifests
AndroidManifest.xml
kotlin+java
com.software.todo (androidTest)
com.software.todo (test)
com.software.todo.presentation.theme
Theme.kt
Type.kt
java (generated)
res
res (generated)
Gradle Scripts
build.gradle.kts (Project: To-Do_App)
build.gradle.kts (Module :app)
proguard-rules.pro (ProGuard Rules for ":app")
gradle.properties (Project Properties)
gradle-wrapper.properties (Gradle Version)
libs.versions.toml (Version Catalog)
local.properties (SDK Location)
settings.gradle.kts (Project Settings)
Theme.kt
libs.versions.toml
build.gradle.kts (:app)
Gradle files have changed since last project sync. A project sync may be necessary for the IDE to work properly.
Sync Now
Ignore these changes
}
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.view.model)
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)
}

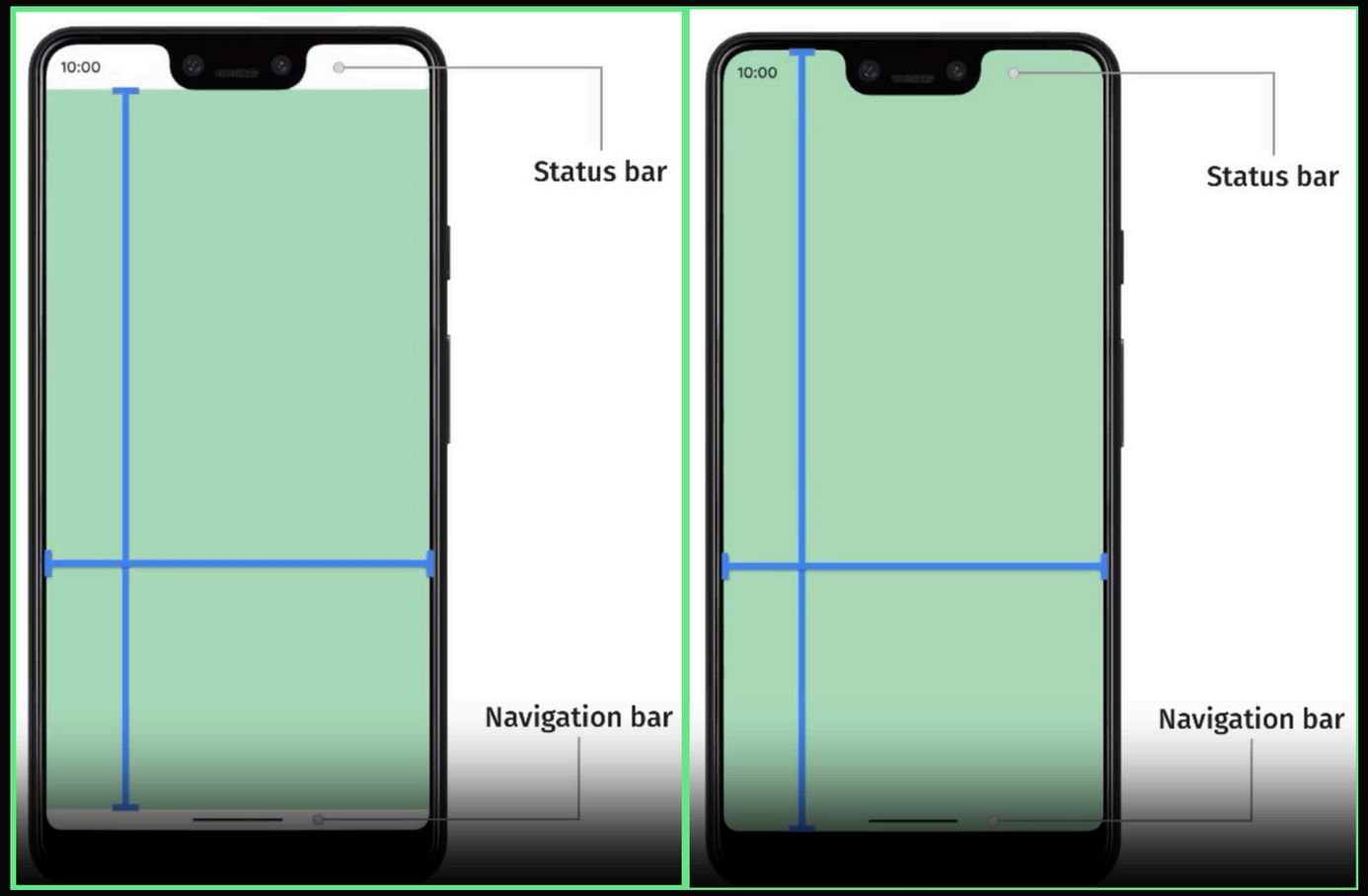

To-Do App
main
Pixel 3a API 35
app
Android
Theme.kt
app
manifests
AndroidManifest.xml
kotlin+java
com.software.todo (androidTest)
com.software.todo (test)
com.software.todo.presentation
theme
java (generated)
res
res (generated)
Gradle Scripts
Gradle files have
ect sync may be necessary for the IDE to work properly.
Sync Now
Ignore these changes
New
Add C++ to Module
Cut ⌘X
Copy ⌘C
Copy Path/Reference...
Paste ⌘V
Find Usages ⌥F7
Find in Files... ⇧⌘F
Replace in Files... ⇧⌘R
Analyze
Refactor
Bookmarks
Reformat Code ⌥⌘L
Optimize Imports ^⌥O
Delete...
Open In
Local History
Git
Repair IDE on File
Reload from Disk
Compare With... ⌘D
Mark Directory as
Java Class
Kotlin Class/File
Android Resource File
Android Resource Directory
Sample Data Directory
File
Scratch File ⇧⌘N
Package
Image Asset
Vector Asset
module-info.java
CMakeLists.txt
Activity
Fragment
Folder
Service
UiComponent
Automotive
XML
Wear
TV
AIDL
Widget
Google
Compose
Other
Resource Bundle
EditorConfig File
Version Catalog
ToDoApp
app
src
main
java
96:2
LF
UTF-8
To-Do App
Dark
4 spaces

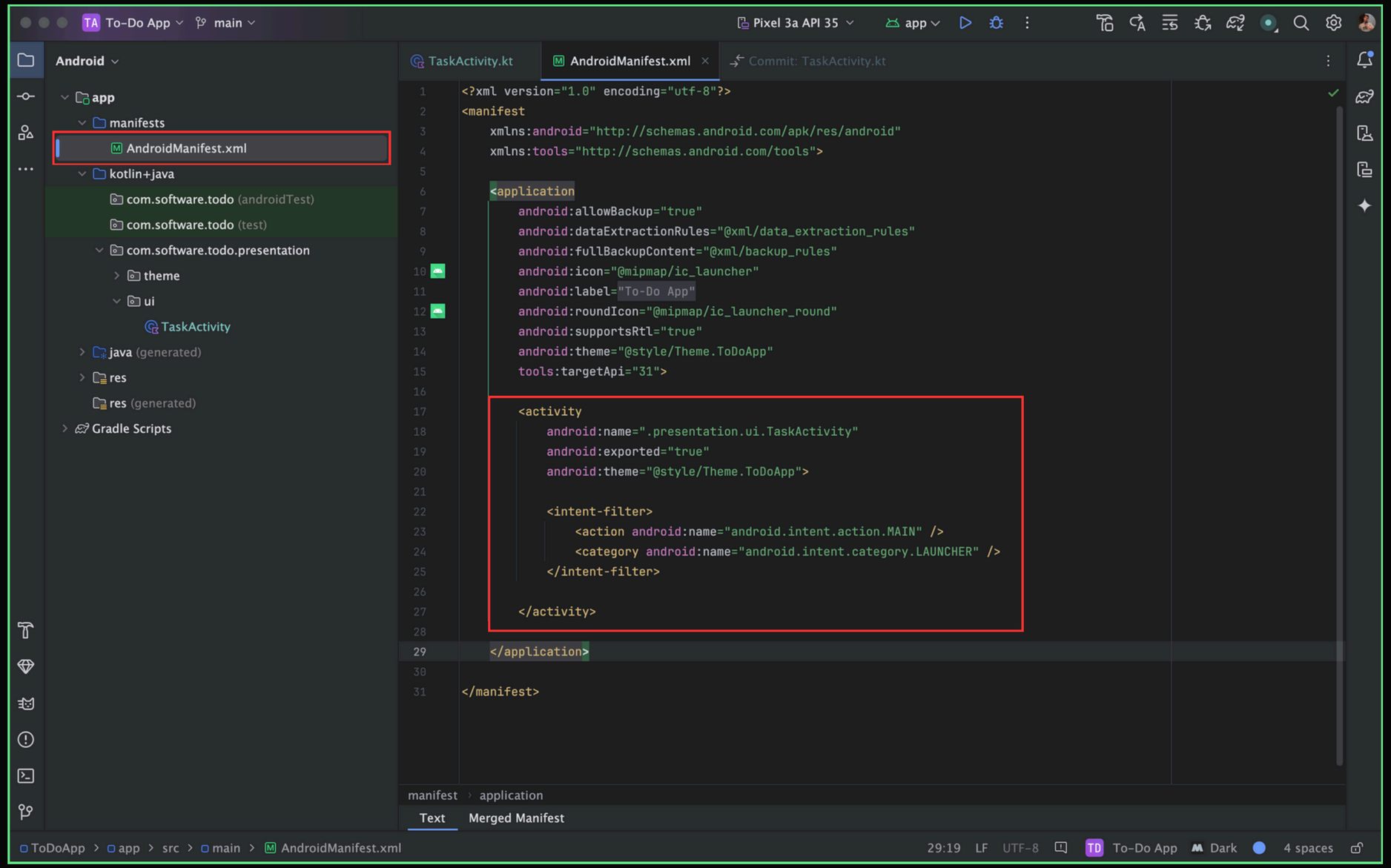

To-Do App
main
Pixel 3a API 35
app
Android
app
manifests
AndroidManifest.xml
kotlin+java
com.software.todo (androidTest)
com.software.todo (test)
com.software.todo.presentation
theme
java (generated)
res
res (generated)
Gradle Scripts
Theme.kt
libs.versions.toml
build.gradle.kts (:app)
Gradle files have changed since last project sync. A project sync may be necessary for the IDE to work properly.
Sync Now
Ignore these changes
}
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)
New Package
com.software.todo.presentation.ui
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)
}
ToDoApp
app
src
main
java
com
software
todo
presentation
96:2
LF
UTF-8
To-Do App
Dark
4 spaces

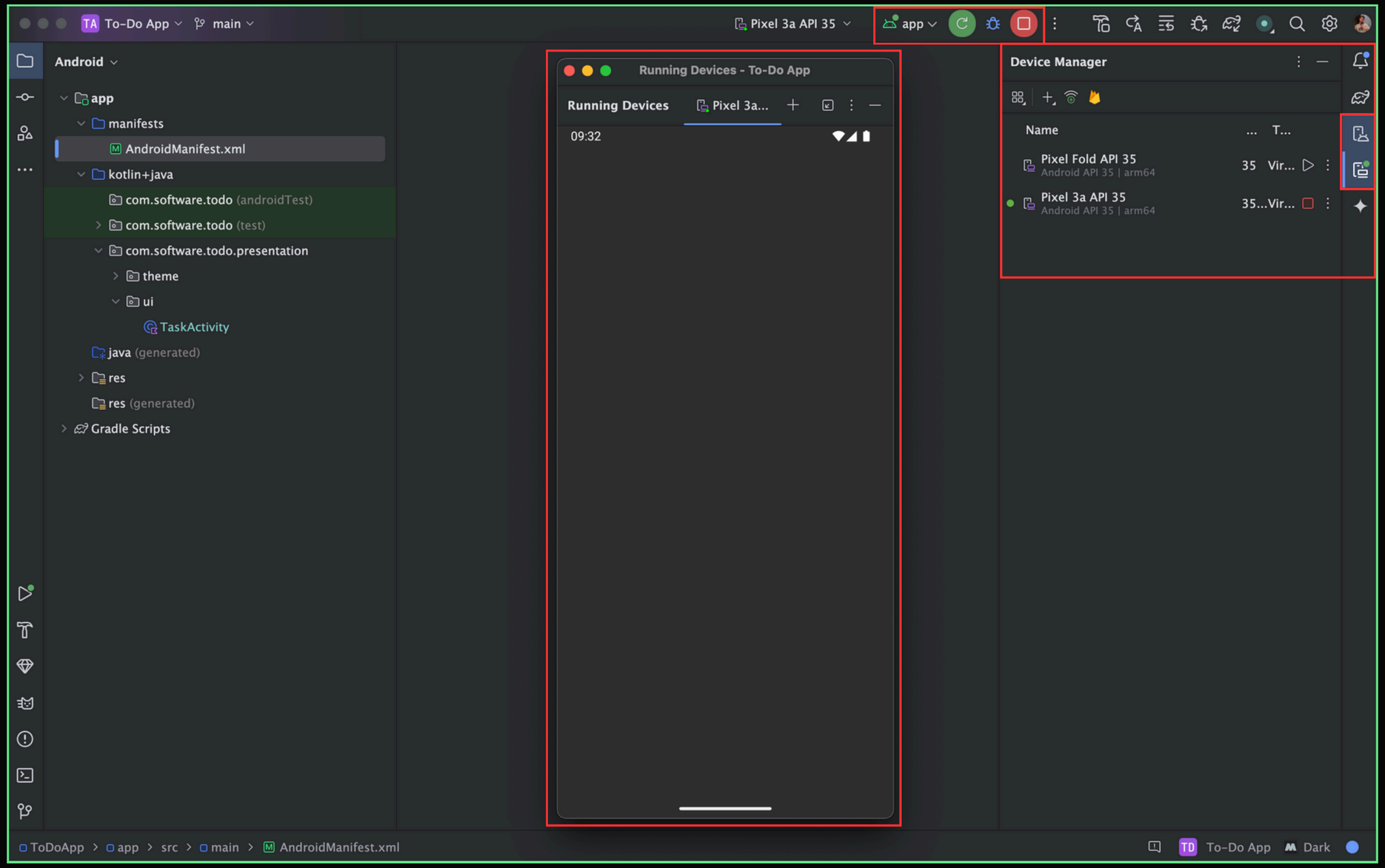

To-Do App
main
Pixel 3a API 35
app
Android
app
manifests
AndroidManifest.xml
kotlin+java
com.software.t
com.software.t
com.software.t
theme
ui
java (generated)
res
res (generated)
Gradle Scripts
build.gradle.kts (:app)
roject sync. A project sync may be necessary for the IDE to work properly.
Sync Now
Ignore these changes
droidx.core.ktx)
droidx.lifecycle.runtime.ktx)
droidx.activity.compose)
n(libs.androidx.compose.bom))
droidx.ui)
droidx.ui.graphics)
droidx.ui.tooling.preview)
droidx.material3)
droidx.animation)
droidx.lifecycle.runtime)
droidx.view.model)
droidx.extend.icons)
s.androidx.ui.tooling)
tlin.coroutines.android)
tlin.coroutines.core)
tpack.room.runtime)
tpack.room.kotlin)
s.jetpack.room.compiler)
compiler)
New
Add C++ to Module
Cut ⌘X
Copy ⌘C
Copy Path/Reference...
Paste ⌘V
Find Usages ⌥F7
Find in Files... ⇧⌘F
Replace in Files... ⇧⌘R
Analyze
Refactor
Bookmarks
Reformat Code ⌥⌘L
Optimize Imports ^⌥O
Delete...
Open In
Local History
Git
Repair IDE on File
Reload from Disk
Compare With... ⌘D
Mark Directory as
Java Class
Kotlin Class/File
Android Resource File
Android Resource Directory
Sample Data Directory
File
Scratch File ⇧⌘N
Package
Image Asset
Vector Asset
module-info.java
CMakeLists.txt
Activity
Fragment
Folder
Service
UiComponent
Automotive
XML
Wear
TV
AIDL
Widget
Google
Compose
Other
Resource Bundle
EditorConfig File
Version Catalog
New Kotlin Class/File
TaskActivity
Class
File
Interface
Data class
Enum class
Annotation
Kotlin script
Kotlin worksheet
Object
ToDoApp
app
src
96:2
LF
UTF-8
To-Do App
Dark
4 spaces

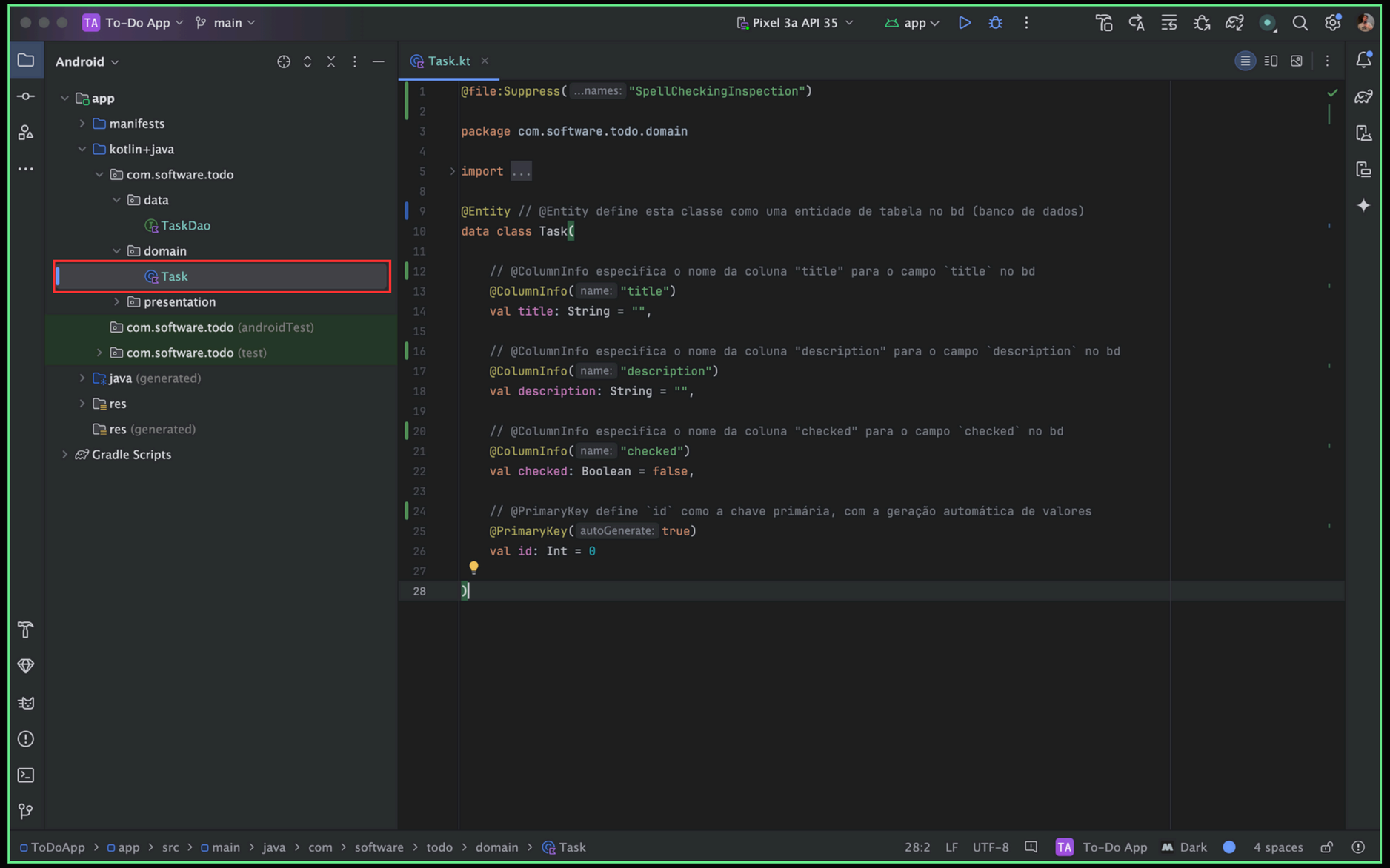

To-Do App
main
Pixel 3a API 35
app
Android
app
manifests
AndroidManifest.xml
kotlin+java
com.software.todo (androidTest)
com.software.todo (test)
com.software.todo.presentation
theme
ui
TaskActivity
java (generated)
res
res (generated)
Gradle Scripts
TaskActivity.kt
1 package com.software.todo.presentation.ui
2
3 class TaskActivity {
4 }
ToDoApp › app › src › main › java › com › software › todo › presentation › ui › TaskActivity
3:21
LF
UTF-8
To-Do App
Dark
4 spaces

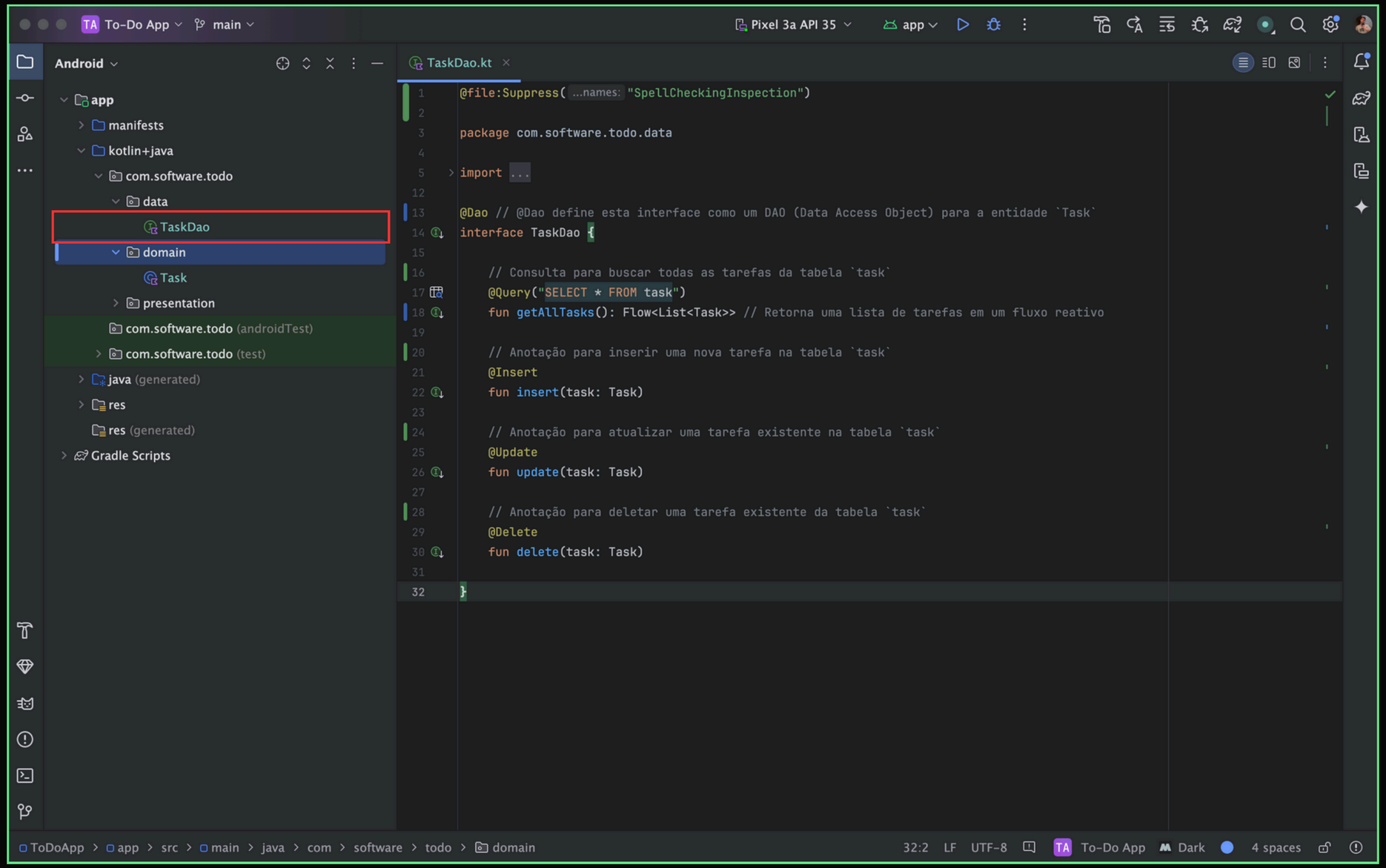

To-Do App
develop
Pixel 3a API 35
app
Android
app
manifests
kotlin+java
com.software.todo (androidTest)
com.software.todo (test)
com.software.todo.presentation
theme
ui
TaskActivity
res
res (generated)
Gradle Scripts
TaskActivity.kt
@file:Suppress(...names: "SpellCheckingInspection")
package com.software.todo.presentation.ui
import ...
class TaskActivity : ComponentActivity() {
override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
enableEdgeToEdge() // O que significa isso?
setContent {
Theme {
}
}
}
}
ToDoApp › app › src › main › java › com › software › todo › presentation › ui › TaskActivity
23:2
LF
UTF-8
To-Do App
Dark
4 spaces

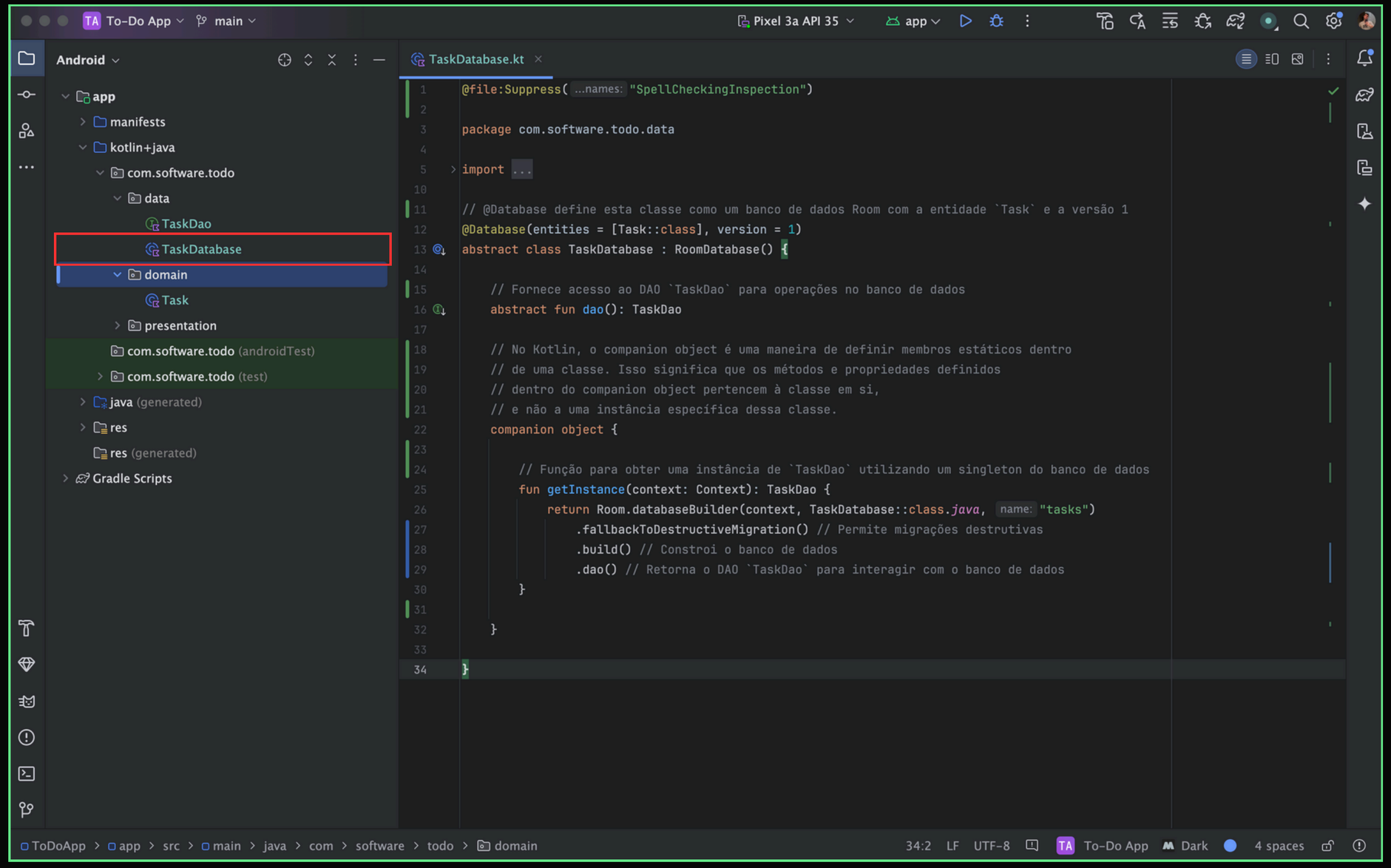

10:00
Status bar
Navigation bar
10:00
Status bar
Navigation bar

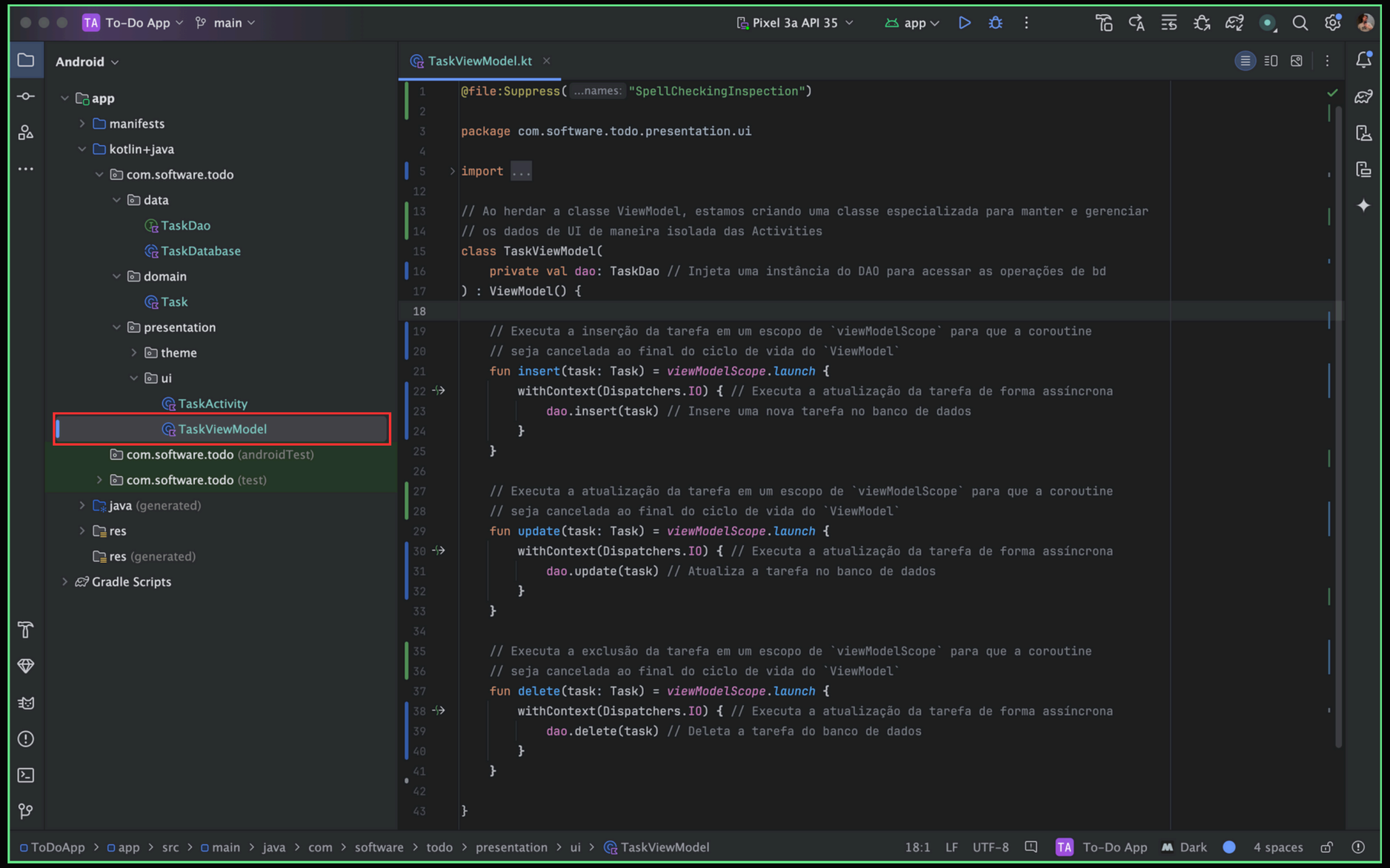

To-Do App
main
Pixel 3a API 35
app
Android
app
manifests
AndroidManifest.xml
kotlin+java
com.software.todo (androidTest)
com.software.todo (test)
com.software.todo.presentation
theme
ui
TaskActivity
java (generated)
res
res (generated)
Gradle Scripts
TaskActivity.kt
AndroidManifest.xml
Commit: TaskActivity.kt
<?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
Dark
4 spaces

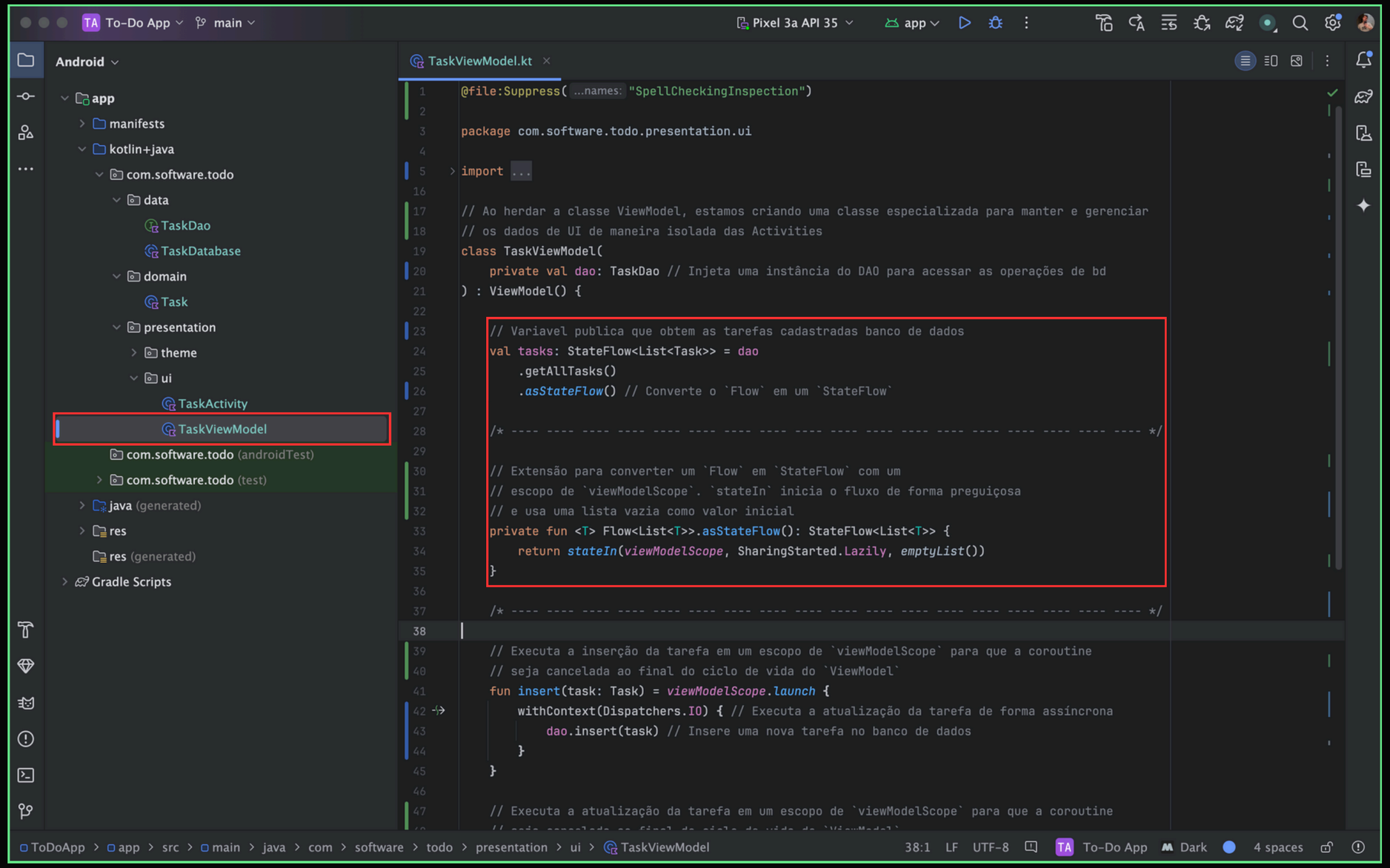

To-Do App
main
Pixel 3a API 35
app
Android
app
manifests
AndroidManifest.xml
kotlin+java
com.software.todo (androidTest)
com.software.todo (test)
com.software.todo.presentation
theme
ui
TaskActivity
java (generated)
res
res (generated)
Gradle Scripts
Running Devices - To-Do App
Running Devices
Pixel 3a...
09:32
Device Manager
Name
...
T...
Pixel Fold API 35
Android API 35 | arm64
35
Vir...
Pixel 3a API 35
Android API 35 | arm64
35...
Vir...
ToDoApp
app
src
main
AndroidManifest.xml
To-Do App
Dark

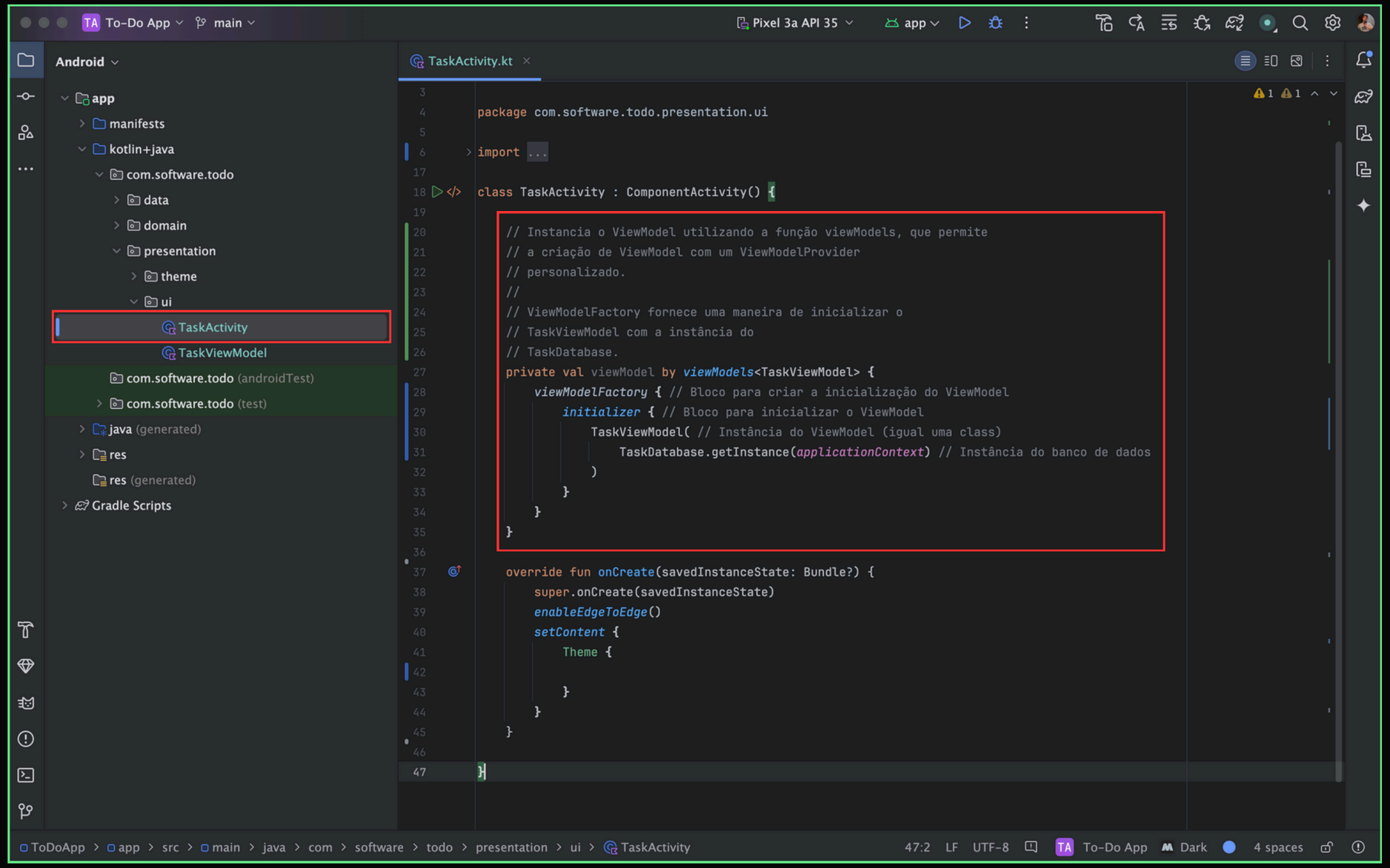

To-Do App
main
Pixel 3a API 35
app
Android
app
manifests
kotlin+java
com.software.todo
data
TaskDao
domain
Task
presentation
com.software.todo (androidTest)
com.software.todo (test)
java (generated)
res
res (generated)
Gradle Scripts
Task.kt
@file:Suppress(...names: "SpellCheckingInspection")
package com.software.todo.domain
import ...
@Entity // @Entity define esta classe como uma entidade de tabela no bd (banco de dados)
data class Task(
// @ColumnInfo especifica o nome da coluna "title" para o campo `title` no bd
@ColumnInfo(name: "title")
val title: String = "",
// @ColumnInfo especifica o nome da coluna "description" para o campo `description` no bd
@ColumnInfo(name: "description")
val description: String = "",
// @ColumnInfo especifica o nome da coluna "checked" para o campo `checked` no bd
@ColumnInfo(name: "checked")
val checked: Boolean = false,
// @PrimaryKey define `id` como a chave primária, com a geração automática de valores
@PrimaryKey(autoGenerate: true)
val id: Int = 0
)
ToDoApp
app
src
main
java
com
software
todo
domain
Task
28:2
LF
UTF-8
To-Do App
Dark
4 spaces

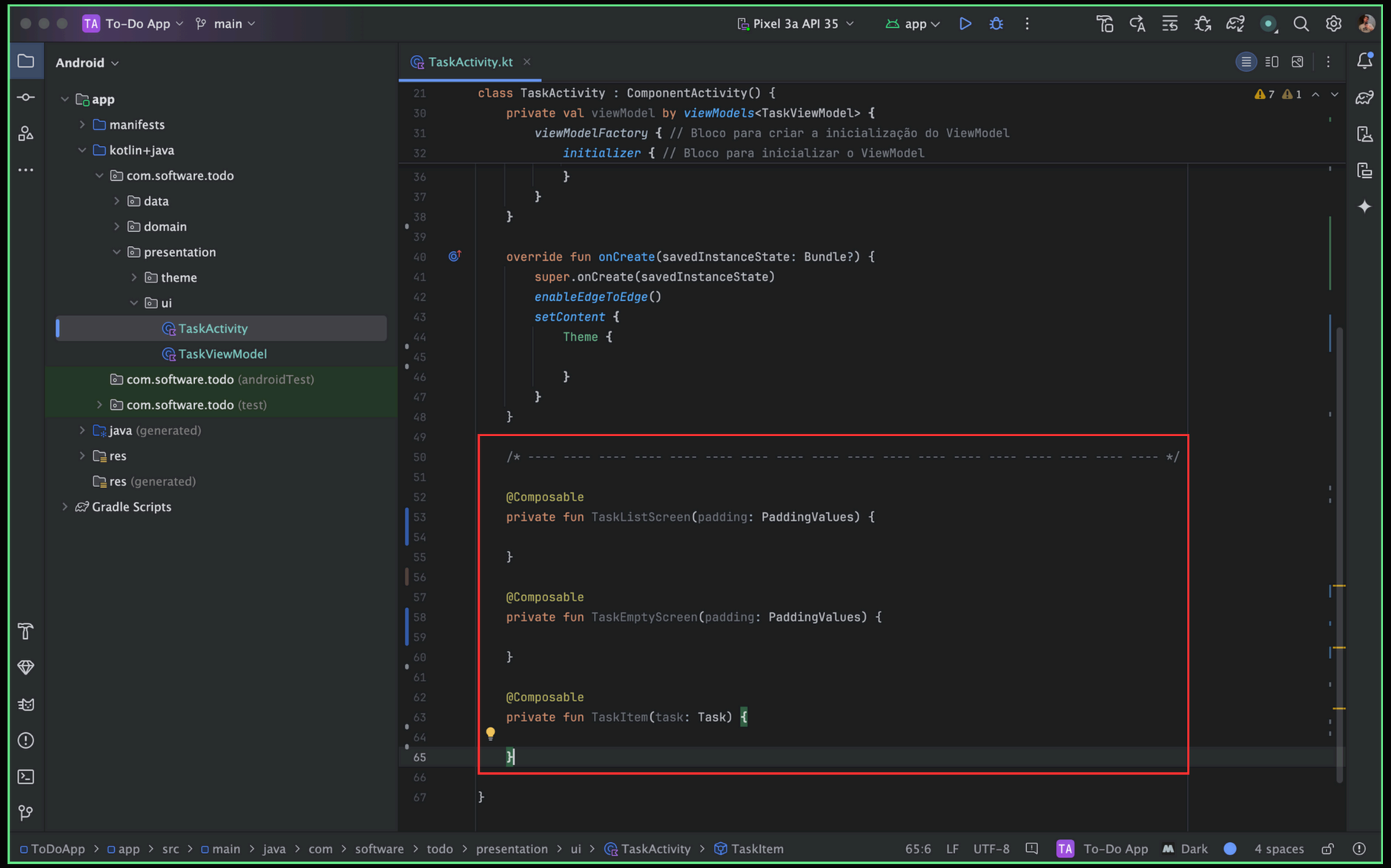

To-Do App
main
Pixel 3a API 35
app
Android
app
manifests
kotlin+java
com.software.todo
data
TaskDao
domain
Task
presentation
com.software.todo (androidTest)
com.software.todo (test)
java (generated)
res
res (generated)
Gradle Scripts
TaskDao.kt
@file:Suppress(...names: "SpellCheckingInspection")
package com.software.todo.data
import ...
@Dao // @Dao define esta interface como um DAO (Data Access Object) para a entidade `Task`
interface TaskDao {
// Consulta para buscar todas as tarefas da tabela `task`
@Query("SELECT * FROM task")
fun getAllTasks(): Flow<List<Task>> // Retorna uma lista de tarefas em um fluxo reativo
// Anotação para inserir uma nova tarefa na tabela `task`
@Insert
fun insert(task: Task)
// Anotação para atualizar uma tarefa existente na tabela `task`
@Update
fun update(task: Task)
// Anotação para deletar uma tarefa existente da tabela `task`
@Delete
fun delete(task: Task)
}
ToDoApp
app
src
main
java
com
software
todo
domain
32:2
LF
UTF-8
To-Do App
Dark
4 spaces

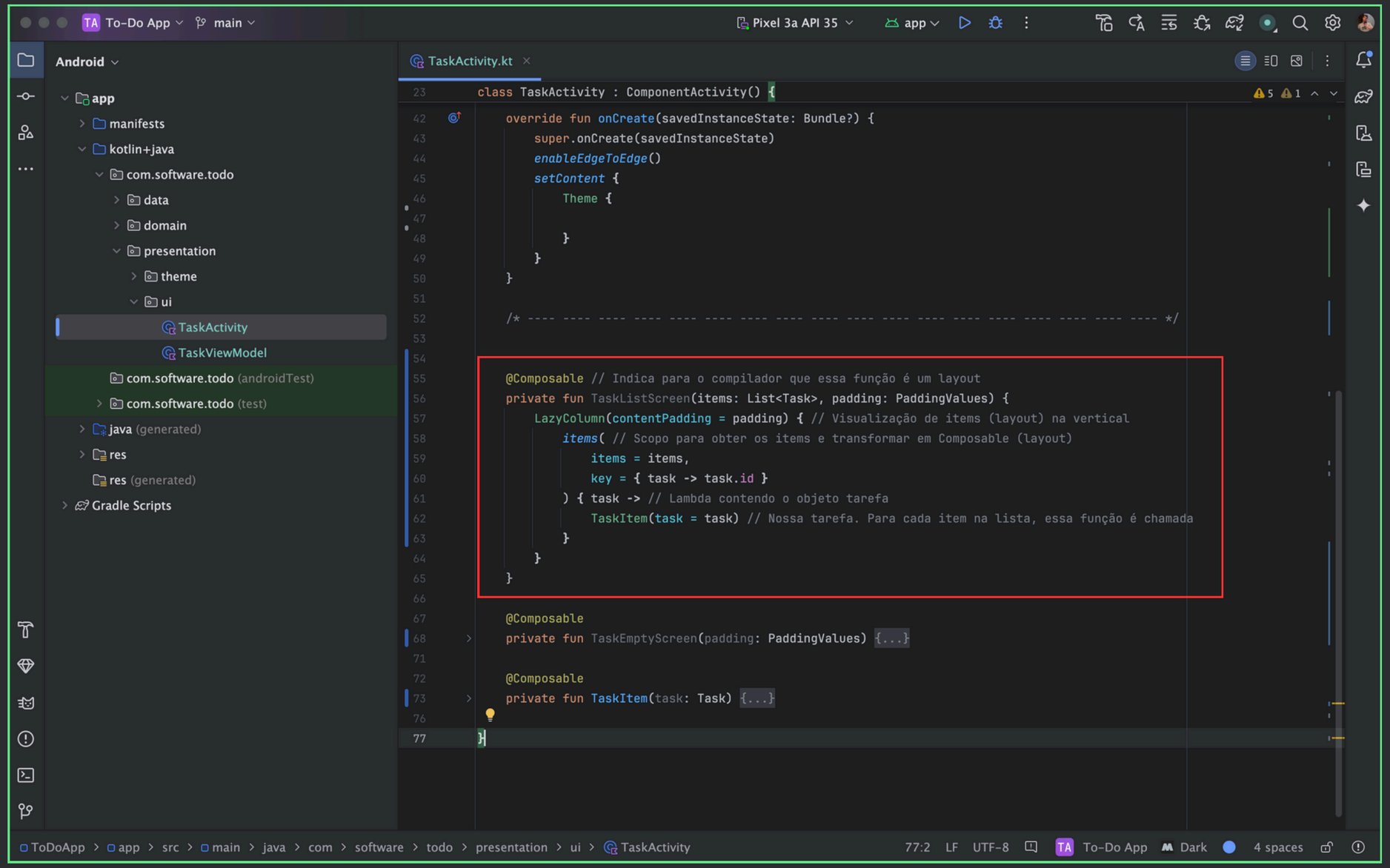

To-Do App
main
Pixel 3a API 35
app
Android
app
manifests
kotlin+java
com.software.todo
data
TaskDao
TaskDatabase
domain
Task
presentation
com.software.todo (androidTest)
com.software.todo (test)
java (generated)
res
res (generated)
Gradle Scripts
TaskDatabase.kt
@file:Suppress(...names: "SpellCheckingInspection")
package com.software.todo.data
import ...
// @Database define esta classe como um banco de dados Room com a entidade `Task` e a versão 1
@Database(entities = [Task::class], version = 1)
abstract class TaskDatabase : RoomDatabase() {
// Fornece acesso ao DAO `TaskDao` para operações no banco de dados
abstract fun dao(): TaskDao
// No Kotlin, o companion object é uma maneira de definir membros estáticos dentro
// de uma classe. Isso significa que os métodos e propriedades definidos
// dentro do companion object pertencem à classe em si,
// e não a uma instância específica dessa classe.
companion object {
// Função para obter uma instância de `TaskDao` utilizando um singleton do banco de dados
fun getInstance(context: Context): TaskDao {
return Room.databaseBuilder(context, TaskDatabase::class.java, name: "tasks")
.fallbackToDestructiveMigration() // Permite migrações destrutivas
.build() // Constroi o banco de dados
.dao() // Retorna o DAO `TaskDao` para interagir com o banco de dados
}
}
}
ToDoApp
app
src
main
java
com
software
todo
domain
34:2
LF
UTF-8
To-Do App
Dark
4 spaces

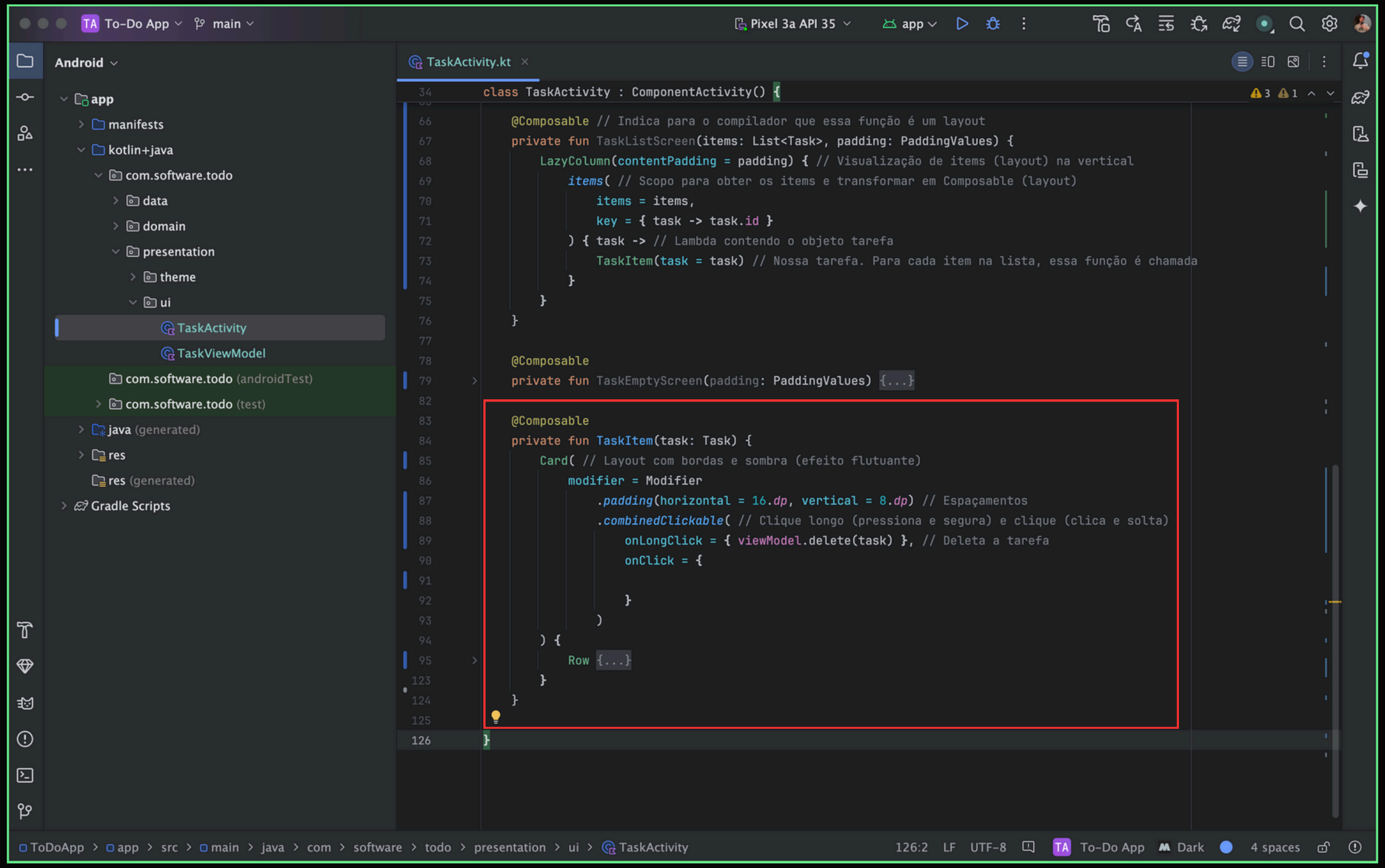

To-Do App
main
Pixel 3a API 35
app
Android
app
manifests
kotlin+java
com.software.todo
data
TaskDao
TaskDatabase
domain
Task
presentation
theme
ui
TaskActivity
TaskViewModel
com.software.todo (androidTest)
com.software.todo (test)
java (generated)
res
res (generated)
Gradle Scripts
TaskViewModel.kt
@file:Suppress(...names: "SpellCheckingInspection")
package com.software.todo.presentation.ui
import ...
// Ao herdar a classe ViewModel, estamos criando uma classe especializada para manter e gerenciar
// os dados de UI de maneira isolada das Activities
class TaskViewModel(
private val dao: TaskDao // Injeta uma instância do DAO para acessar as operações de bd
) : ViewModel() {
// Executa a inserção da tarefa em um escopo de `viewModelScope` para que a coroutine
// seja cancelada ao final do ciclo de vida do `ViewModel`
fun insert(task: Task) = viewModelScope.launch {
withContext(Dispatchers.IO) { // Executa a atualização da tarefa de forma assíncrona
dao.insert(task) // Insere uma nova tarefa no banco de dados
}
}
// Executa a atualização da tarefa em um escopo de `viewModelScope` para que a coroutine
// seja cancelada ao final do ciclo de vida do `ViewModel`
fun update(task: Task) = viewModelScope.launch {
withContext(Dispatchers.IO) { // Executa a atualização da tarefa de forma assíncrona
dao.update(task) // Atualiza a tarefa no banco de dados
}
}
// Executa a exclusão da tarefa em um escopo de `viewModelScope` para que a coroutine
// seja cancelada ao final do ciclo de vida do `ViewModel`
fun delete(task: Task) = viewModelScope.launch {
withContext(Dispatchers.IO) { // Executa a atualização da tarefa de forma assíncrona
dao.delete(task) // Deleta a tarefa do banco de dados
}
}
}
ToDoApp › app › src › main › java › com › software › todo › presentation › ui › TaskViewModel
18:1 LF UTF-8 To-Do App Dark 4 spaces

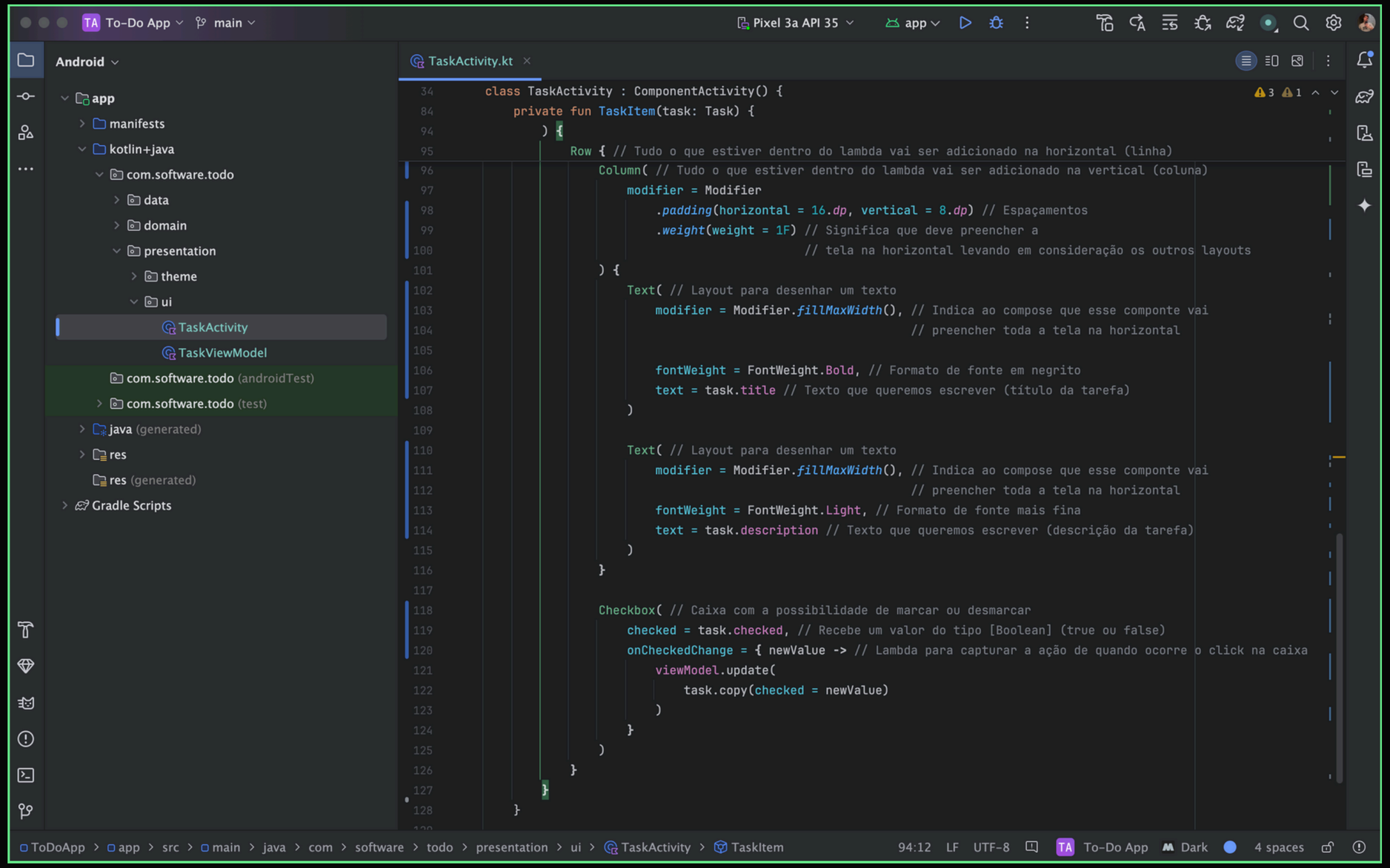

To-Do App
main
Pixel 3a API 35
app
Android
app
manifests
kotlin+java
com.software.todo
data
TaskDao
TaskDatabase
domain
Task
presentation
theme
ui
TaskActivity
TaskViewModel
com.software.todo (androidTest)
com.software.todo (test)
java (generated)
res
res (generated)
Gradle Scripts
TaskViewModel.kt
@file:Suppress(...names: "SpellCheckingInspection")
package com.software.todo.presentation.ui
import ...
// Ao herdar a classe ViewModel, estamos criando uma classe especializada para manter e gerenciar
// os dados de UI de maneira isolada das Activities
class TaskViewModel(
private val dao: TaskDao // Injeta uma instância do DAO para acessar as operações de bd
) : ViewModel() {
// Variavel publica que obtem as tarefas cadastradas banco de dados
val tasks: StateFlow<List<Task>> = dao
.getAllTasks()
.asStateFlow() // Converte o `Flow` em um `StateFlow`
/* ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- */
// Extensão para converter um `Flow` em `StateFlow` com um
// escopo de `viewModelScope`. `stateIn` inicia o fluxo de forma preguiçosa
// e usa uma lista vazia como valor inicial
private fun <T> Flow<List<T>>.asStateFlow(): StateFlow<List<T>> {
return stateIn(viewModelScope, SharingStarted.Lazily, emptyList())
}
/* ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- */
// Executa a inserção da tarefa em um escopo de `viewModelScope` para que a coroutine
// seja cancelada ao final do ciclo de vida do `ViewModel`
fun insert(task: Task) = viewModelScope.launch {
withContext(Dispatchers.IO) { // Executa a atualização da tarefa de forma assincrona
dao.insert(task) // Insere uma nova tarefa no banco de dados
}
}
// Executa a atualização da tarefa em um escopo de `viewModelScope` para que a coroutine
ToDoApp
app
src
main
java
com
software
todo
presentation
ui
TaskViewModel
38:1
LF
UTF-8
To-Do App
Dark
4 spaces

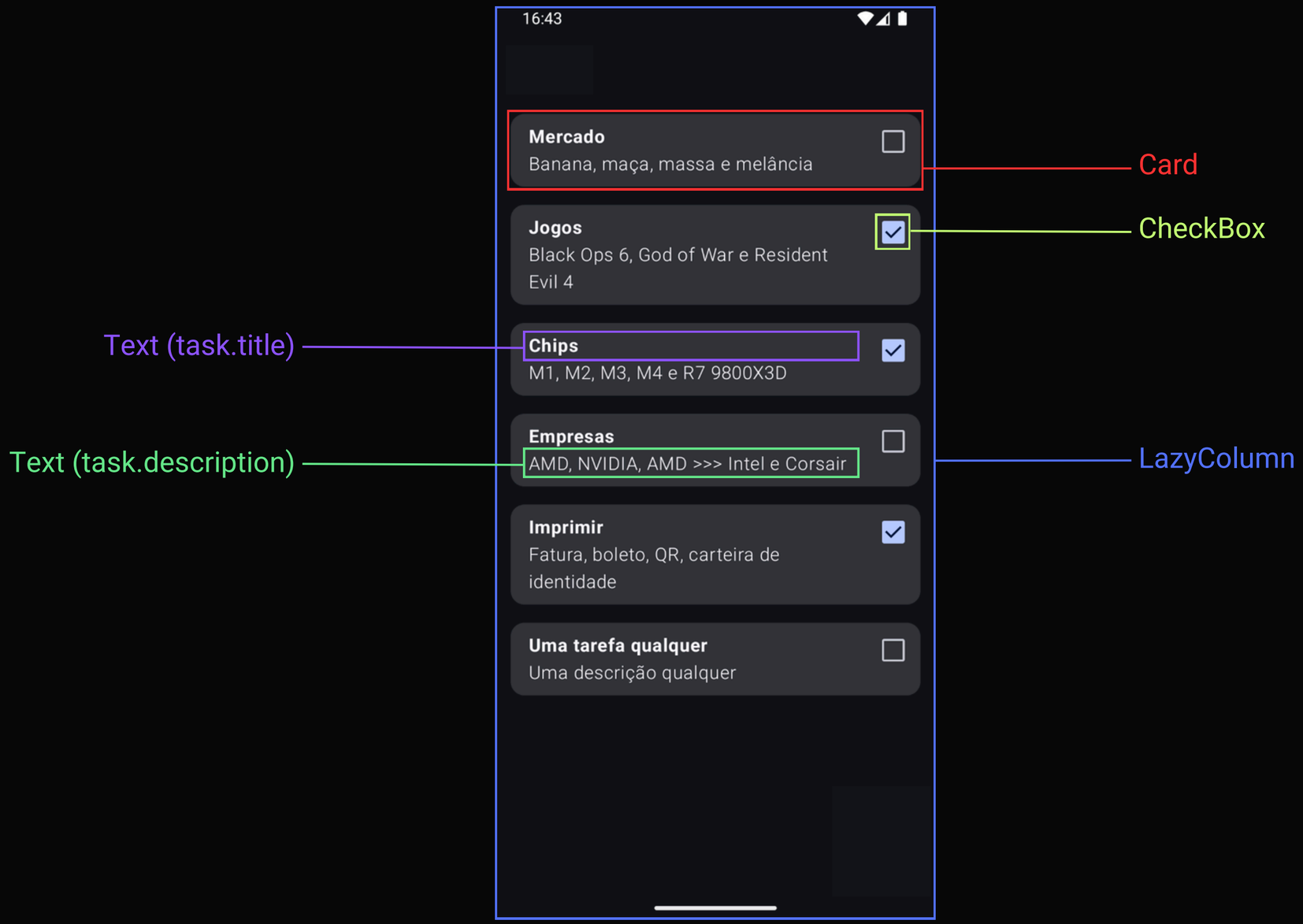

To-Do App
main
Pixel 3a API 35
app
Android
app
manifests
kotlin+java
com.software.todo
data
domain
presentation
theme
ui
TaskActivity
TaskViewModel
com.software.todo (androidTest)
com.software.todo (test)
java (generated)
res
res (generated)
Gradle Scripts
TaskActivity.kt
package com.software.todo.presentation.ui
import ...
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
To-Do App
Dark
4 spaces

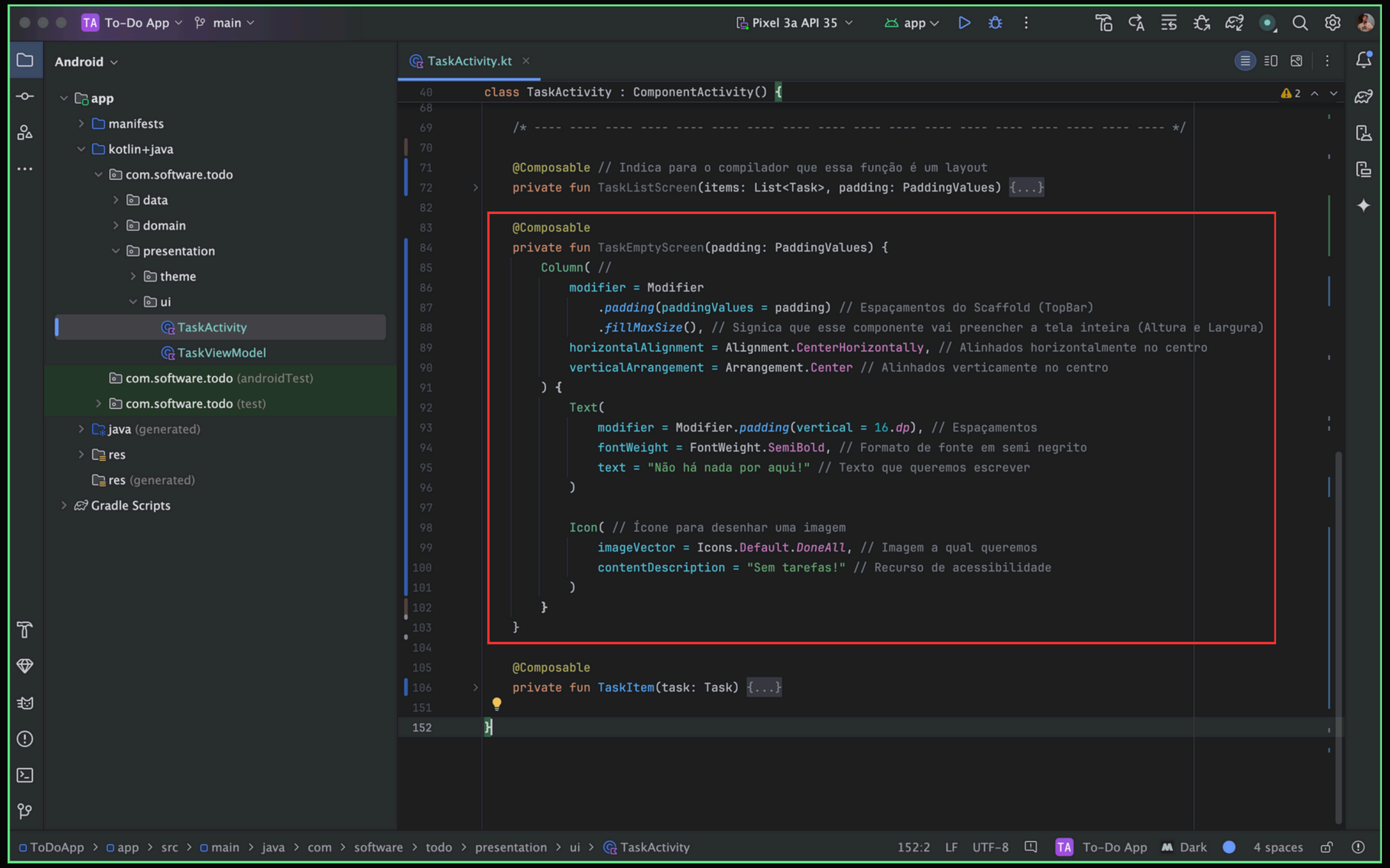

To-Do App
main
Pixel 3a API 35
app
Android
app
manifests
kotlin+java
com.software.todo
data
domain
presentation
theme
ui
TaskActivity
TaskViewModel
com.software.todo (androidTest)
com.software.todo (test)
java (generated)
res
res (generated)
Gradle Scripts
TaskActivity.kt
class TaskActivity : ComponentActivity() {
private val viewModel by viewModels<TaskViewModel> {
viewModelFactory { // Bloco para criar a inicialização do ViewModel
initializer { // Bloco para inicializar o ViewModel
}
}
}
override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
enableEdgeToEdge()
setContent {
Theme {
}
}
}
/* ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- */
@Composable
private fun TaskListScreen(padding: PaddingValues) {
}
@Composable
private fun TaskEmptyScreen(padding: PaddingValues) {
}
@Composable
private fun TaskItem(task: Task) {
}
}
ToDoApp
app
src
main
java
com
software
todo
presentation
ui
TaskActivity
TaskItem
65:6
LF
UTF-8
To-Do App
Dark
4 spaces

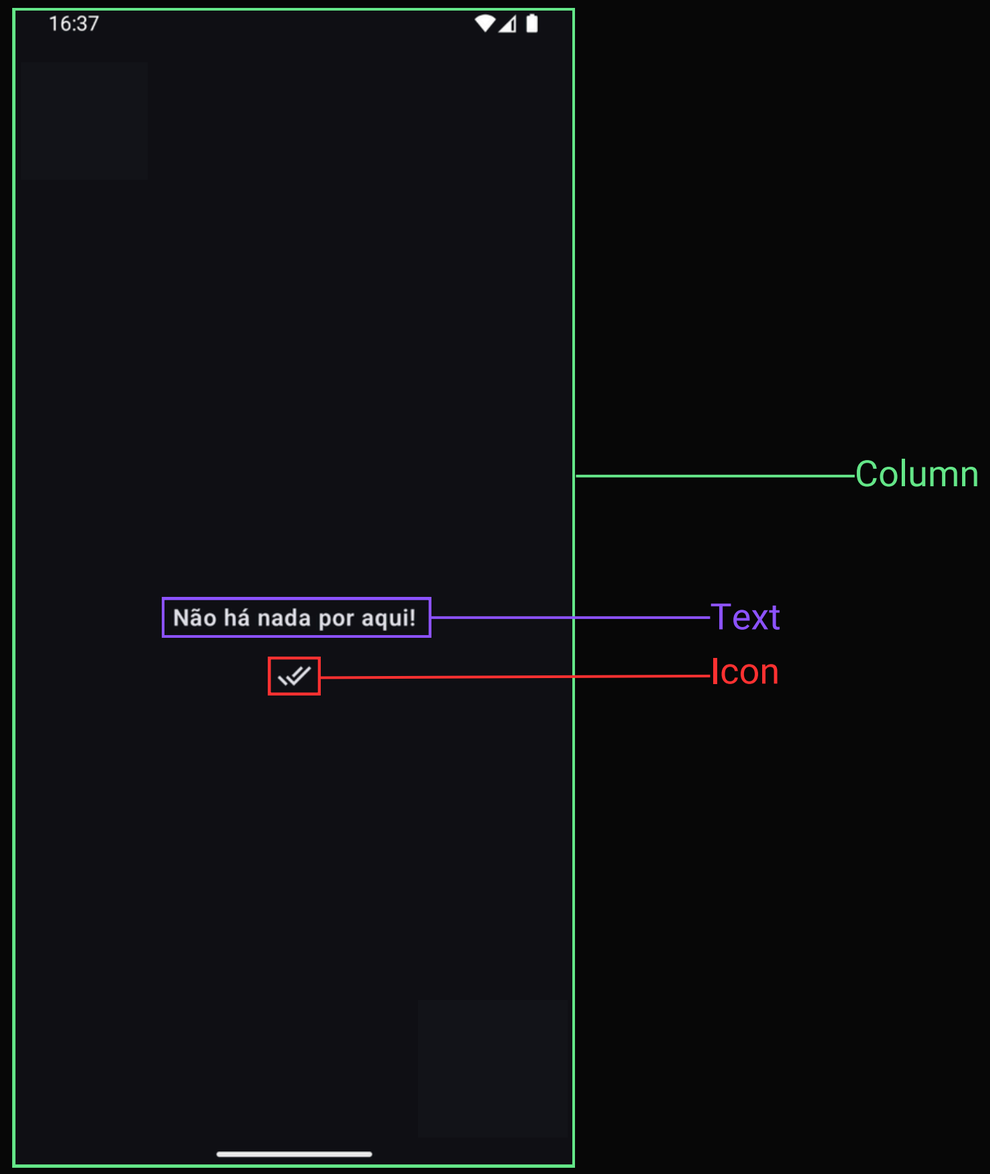

To-Do App
main
Pixel 3a API 35
app
Android
app
manifests
kotlin+java
com.software.todo
data
domain
presentation
theme
ui
TaskActivity
TaskViewModel
com.software.todo (androidTest)
com.software.todo (test)
java (generated)
res
res (generated)
Gradle Scripts
TaskActivity.kt
class TaskActivity : ComponentActivity() {
override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
enableEdgeToEdge()
setContent {
Theme {
}
}
}
/* ---- */
@Composable // Indica para o compilador que essa função é um layout
private fun TaskListScreen(items: List<Task>, padding: PaddingValues) {
LazyColumn(contentPadding = padding) { // Visualização de items (layout) na vertical
items(// Scopo para obter os items e transformar em Composable (layout)
items = items,
key = { task -> task.id }
) { task -> // Lambda contendo o objeto tarefa
TaskItem(task = task) // Nossa tarefa. Para cada item na lista, essa função é chamada
}
}
}
@Composable
private fun TaskEmptyScreen(padding: PaddingValues) {...}
@Composable
private fun TaskItem(task: Task) {...}
}
ToDoApp › app › src › main › java › com › software › todo › presentation › ui › TaskActivity
77:2 LF UTF-8 To-Do App Dark 4 spaces

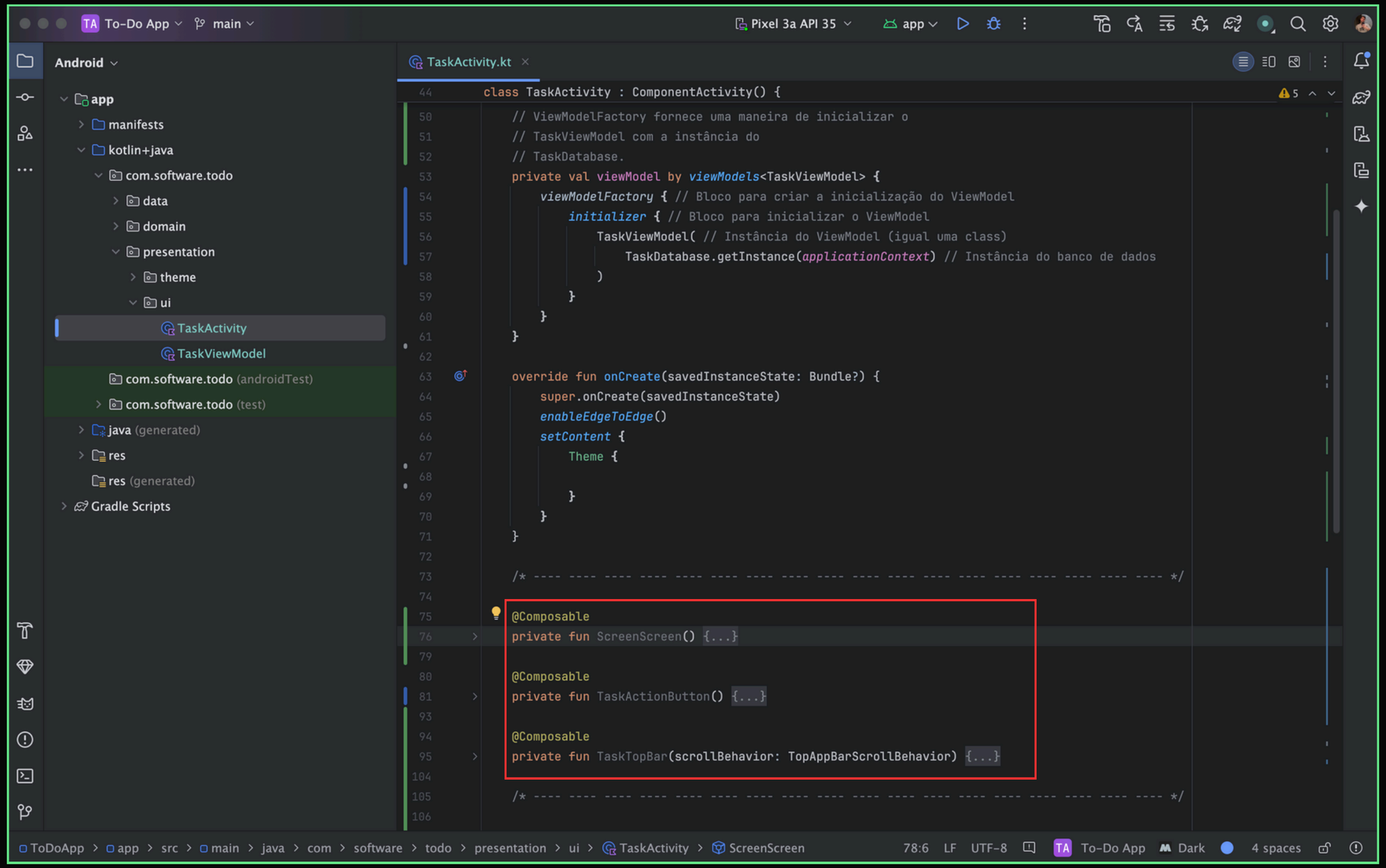

To-Do App
main
Pixel 3a API 35
app
Android
app
manifests
kotlin+java
com.software.todo
data
domain
presentation
theme
ui
TaskActivity
TaskViewModel
com.software.todo (androidTest)
com.software.todo (test)
java (generated)
res
res (generated)
Gradle Scripts
TaskActivity.kt
class TaskActivity : ComponentActivity() {
@Composable // Indica para o compilador que essa função é um layout
private fun TaskListScreen(items: List<Task>, padding: PaddingValues) {
LazyColumn(contentPadding = padding) { // Visualização de items (layout) na vertical
items(// Scopo para obter os items e transformar em Composable (layout)
items = items,
key = { task -> task.id }
) { task -> // Lambda contendo o objeto tarefa
TaskItem(task = task) // Nossa tarefa. Para cada item na lista, essa função é chamada
}
}
}
@Composable
private fun TaskEmptyScreen(padding: PaddingValues) {...}
@Composable
private fun TaskItem(task: Task) {
Card(// Layout com bordas e sombra (efeito flutuante)
modifier = Modifier
.padding(horizontal = 16.dp, vertical = 8.dp) // Espaçamentos
.combinedClickable(// Clique longo (pressiona e segura) e clique (clica e solta)
onLongClick = { viewModel.delete(task) }, // Deleta a tarefa
onClick = {
}
)
) {
Row {...}
}
}
}
ToDoApp
app
src
main
java
com
software
todo
presentation
ui
TaskActivity
126:2
LF
UTF-8
To-Do App
Dark
4 spaces

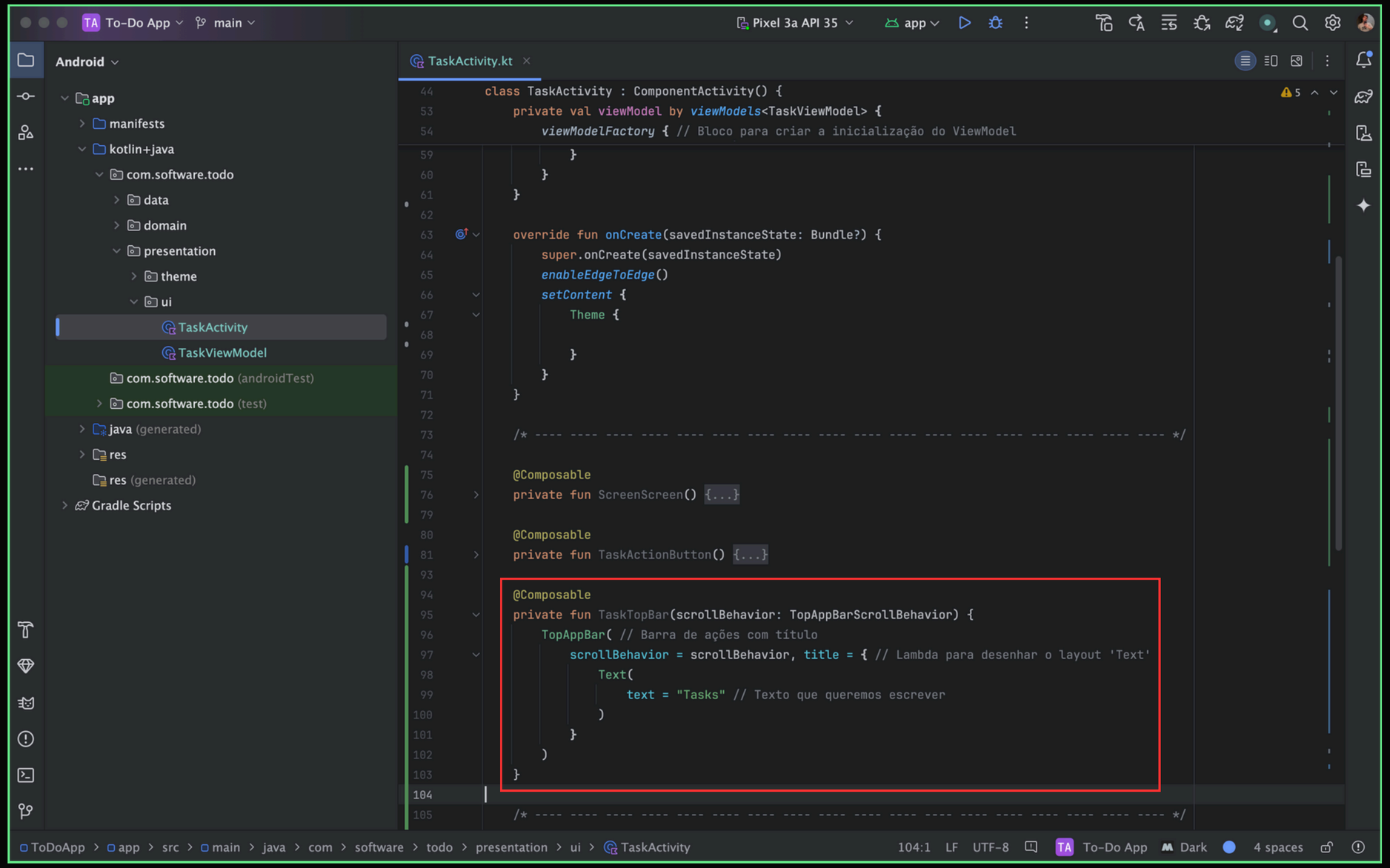

To-Do App
main
Pixel 3a API 35
app
Android
app
manifests
kotlin+java
com.software.todo
data
domain
presentation
theme
ui
TaskActivity
TaskViewModel
com.software.todo (androidTest)
com.software.todo (test)
java (generated)
res
res (generated)
Gradle Scripts
TaskActivity.kt
class TaskActivity : ComponentActivity() {
private fun TaskItem(task: Task) {
) {
Row { // Tudo o que estiver dentro do lambda vai ser adicionado na horizontal (linha)
Column(// Tudo o que estiver dentro do lambda vai ser adicionado na vertical (coluna)
modifier = Modifier
.padding(horizontal = 16.dp, vertical = 8.dp) // Espaçamentos
.weight(weight = 1F) // Significa que deve preencher a
// tela na horizontal levando em consideração os outros layouts
) {
Text(// Layout para desenhar um texto
modifier = Modifier.fillMaxWidth(), // Indica ao compose que esse componte vai
// preencher toda a tela na horizontal
fontWeight = FontWeight.Bold, // Formato de fonte em negrito
text = task.title // Texto que queremos escrever (título da tarefa)
)
Text(// Layout para desenhar um texto
modifier = Modifier.fillMaxWidth(), // Indica ao compose que esse componte vai
// preencher toda a tela na horizontal
fontWeight = FontWeight.Light, // Formato de fonte mais fina
text = task.description // Texto que queremos escrever (descrição da tarefa)
)
}
Checkbox(// Caixa com a possibilidade de marcar ou desmarcar
checked = task.checked, // Recebe um valor do tipo [Boolean] (true ou false)
onCheckedChange = { newValue -> // Lambda para capturar a ação de quando ocorre o click na caixa
viewModel.update(
task.copy(checked = newValue)
)
}
)
}
}
}
ToDoApp
app
src
main
java
com
software
todo
presentation
ui
TaskActivity
TaskItem
94:12
LF
UTF-8
To-Do App
Dark
4 spaces


16:43
Mercado
Banana, maça, massa e melância
Jogos
Black Ops 6, God of War e Resident Evil 4
Chips
M1, M2, M3, M4 e R7 9800X3D
Empresas
AMD, NVIDIA, AMD >>> Intel e Corsair
Imprimir
Fatura, boleto, QR, carteira de identidade
Uma tarefa qualquer
Uma descrição qualquer
Card
CheckBox
Text (task.title)
Text (task.description)
LazyColumn

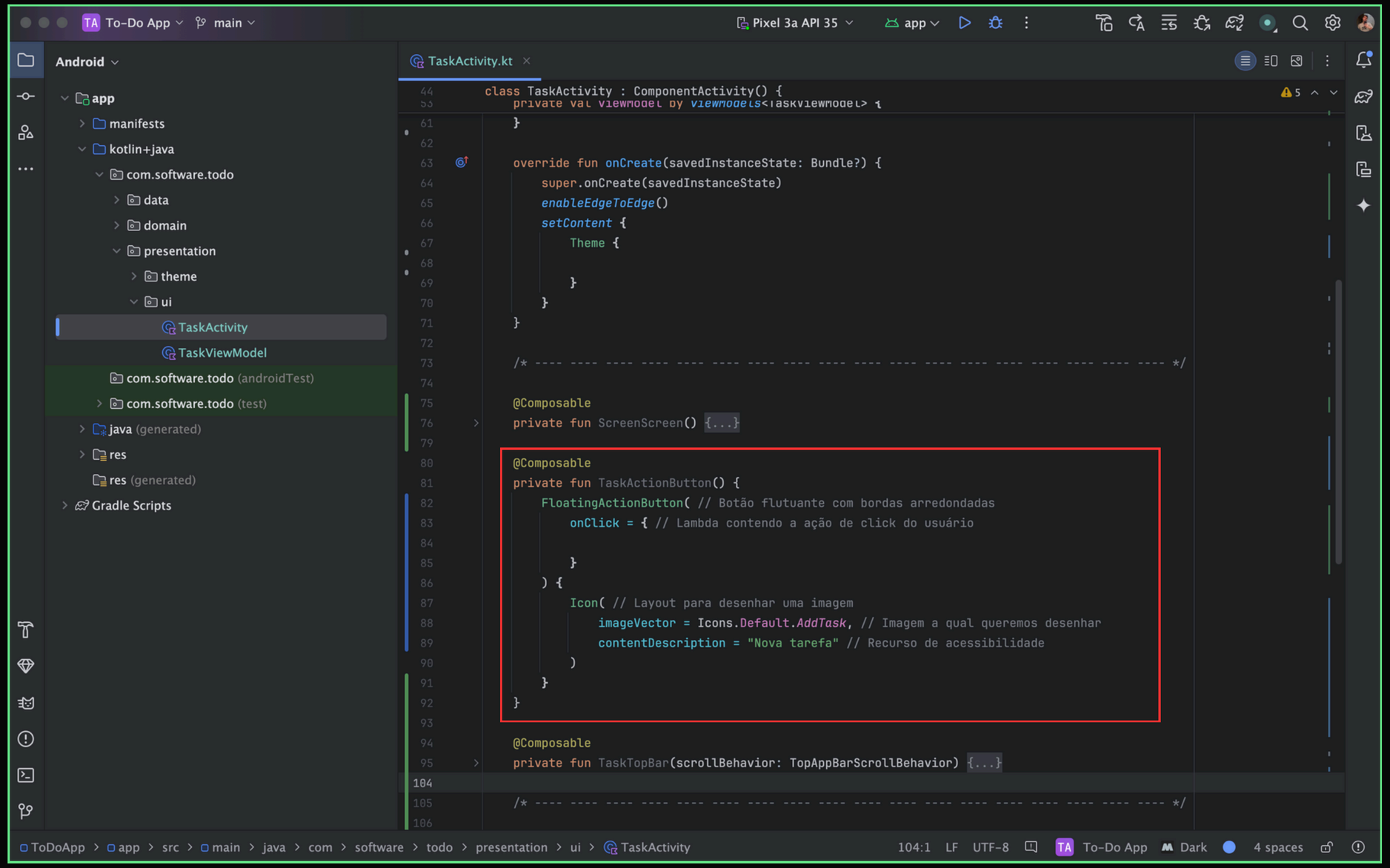

To-Do App
main
Pixel 3a API 35
app
Android
app
manifests
kotlin+java
com.software.todo
data
domain
presentation
theme
ui
TaskActivity
TaskViewModel
com.software.todo (androidTest)
com.software.todo (test)
java (generated)
res
res (generated)
Gradle Scripts
TaskActivity.kt
class TaskActivity : ComponentActivity() {
/* ---- */
@Composable // Indica para o compilador que essa função é um layout
private fun TaskListScreen(items: List<Task>, padding: PaddingValues) {...}
@Composable
private fun TaskEmptyScreen(padding: PaddingValues) {
Column(//
modifier = Modifier
.padding(paddingValues = padding) // Espaçamentos do Scaffold (TopBar)
.fillMaxSize(), // Signica que esse componente vai preencher a tela inteira (Altura e Largura)
horizontalAlignment = Alignment.CenterHorizontally, // Alinhados horizontalmente no centro
verticalArrangement = Arrangement.Center // Alinhados verticamente no centro
) {
Text(
modifier = Modifier.padding(vertical = 16.dp), // Espaçamentos
fontWeight = FontWeight.SemiBold, // Formato de fonte em semi negrito
text = "Não há nada por aqui!" // Texto que queremos escrever
)
Icon(// Ícone para desenhar uma imagem
imageVector = Icons.Default.DoneAll, // Imagem a qual queremos
contentDescription = "Sem tarefas!" // Recurso de acessibilidade
)
}
}
@Composable
private fun TaskItem(task: Task) {...}
}
ToDoApp
app
src
main
java
com
software
todo
presentation
ui
TaskActivity
152:2
LF
UTF-8
To-Do App
Dark
4 spaces


16:37
Não há nada por aqui!
Column
Text
Icon

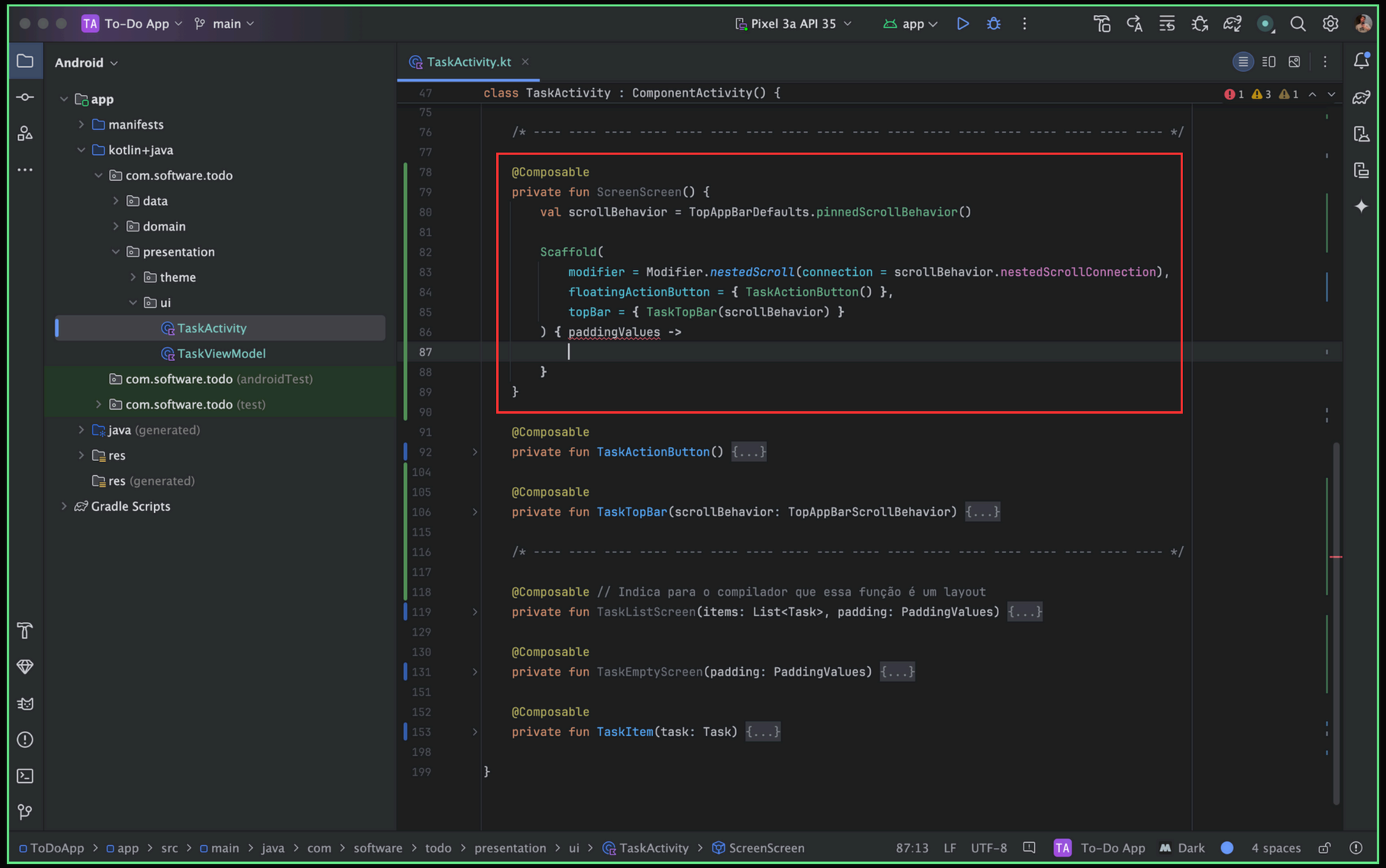

To-Do App
main
Pixel 3a API 35
app
Android
app
manifests
kotlin+java
com.software.todo
data
domain
presentation
theme
ui
TaskActivity
TaskViewModel
com.software.todo (androidTest)
com.software.todo (test)
java (generated)
res
res (generated)
Gradle Scripts
TaskActivity.kt
class TaskActivity : ComponentActivity() {
// 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 {
}
}
}
/* ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- */
@Composable
private fun ScreenScreen() {...}
@Composable
private fun TaskActionButton() {...}
@Composable
private fun TaskTopBar(scrollBehavior: TopAppBarScrollBehavior) {...}
/* ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- */
ToDoApp › app › src › main › java › com › software › todo › presentation › ui › TaskActivity › ScreenScreen
78:6
LF
UTF-8
To-Do App
Dark
4 spaces

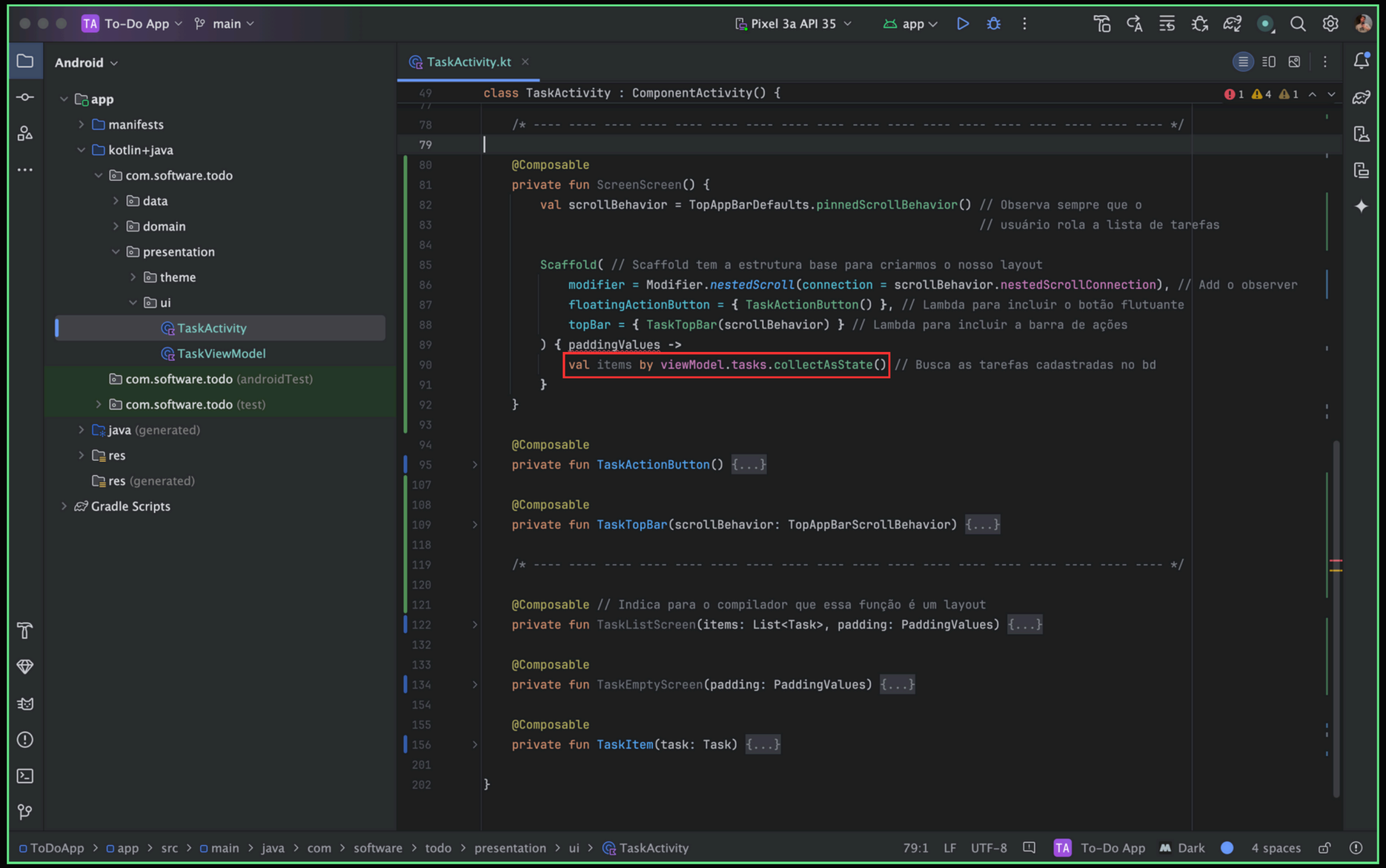

To-Do App
main
Pixel 3a API 35
app
Android
app
manifests
kotlin+java
com.software.todo
data
domain
presentation
theme
ui
TaskActivity
TaskViewModel
com.software.todo (androidTest)
com.software.todo (test)
java (generated)
res
res (generated)
Gradle Scripts
TaskActivity.kt
class TaskActivity : ComponentActivity() {
private val viewModel by viewModels<TaskViewModel> {
viewModelFactory { // Bloco para criar a inicialização do ViewModel
}
}
}
override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
enableEdgeToEdge()
setContent {
Theme {
}
}
}
/* ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- */
@Composable
private fun ScreenScreen() {...}
@Composable
private fun TaskActionButton() {...}
@Composable
private fun TaskTopBar(scrollBehavior: TopAppBarScrollBehavior) {
TopAppBar(// Barra de ações com título
scrollBehavior = scrollBehavior, title = { // Lambda para desenhar o layout 'Text'
Text(
text = "Tasks" // Texto que queremos escrever
)
}
)
}
/* ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- */
ToDoApp › app › src › main › java › com › software › todo › presentation › ui › TaskActivity
104:1 LF UTF-8
To-Do App
Dark
4 spaces

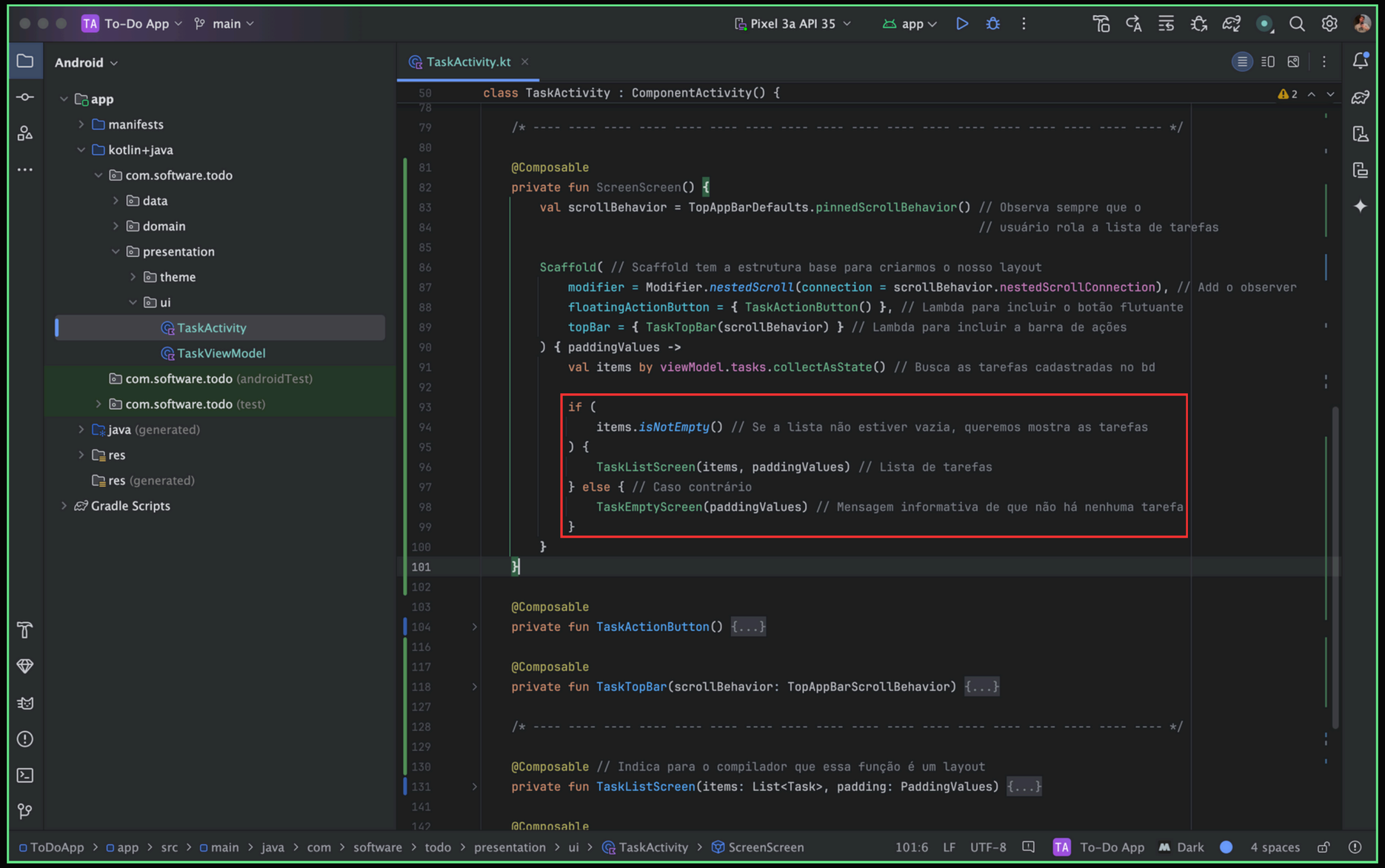

Tarefas
TopBar

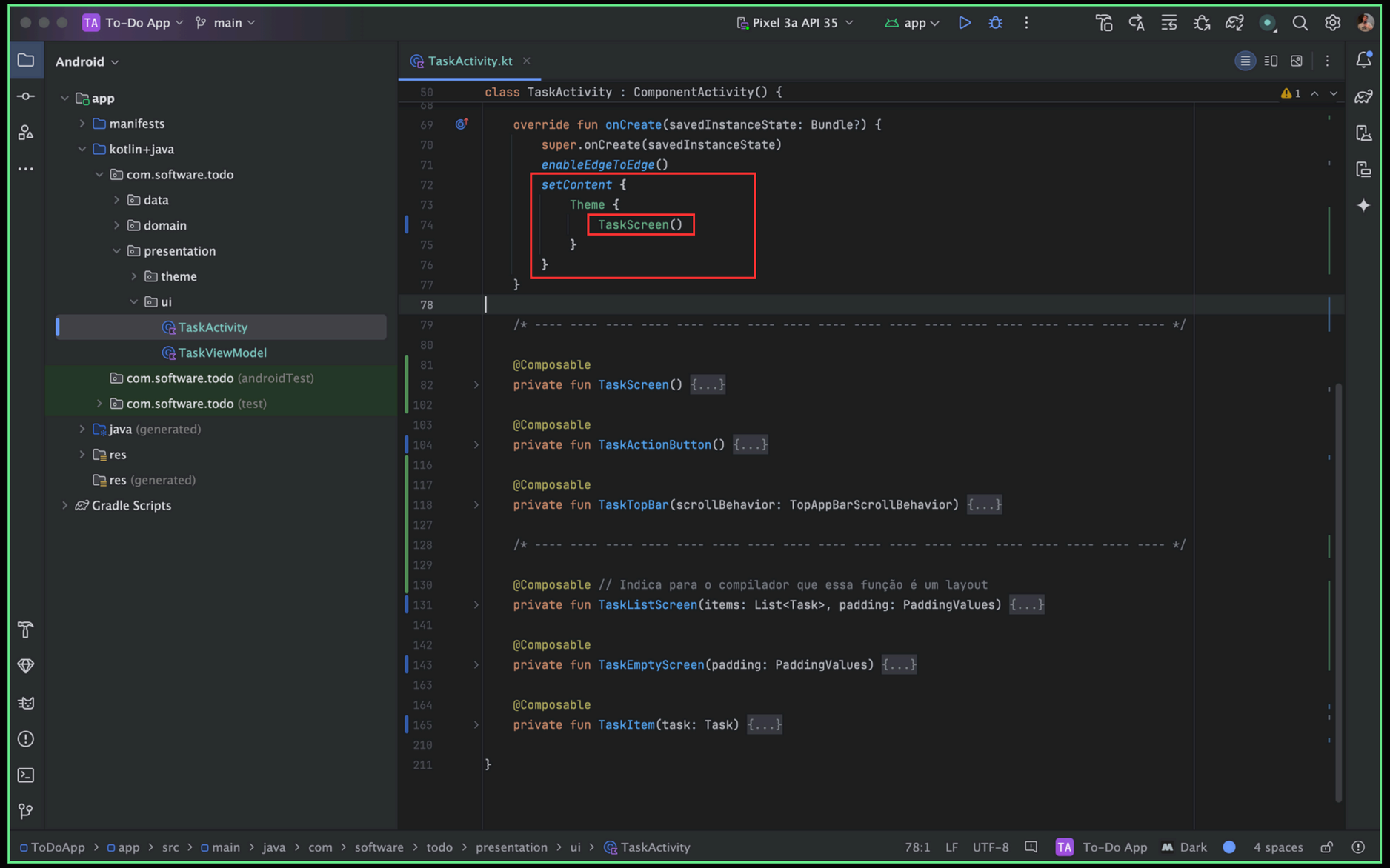

To-Do App
main
Pixel 3a API 35
app
Android
app
manifests
kotlin+java
com.software.todo
data
domain
presentation
theme
ui
TaskActivity
TaskViewModel
com.software.todo (androidTest)
com.software.todo (test)
java (generated)
res
res (generated)
Gradle Scripts
TaskActivity.kt
class TaskActivity : ComponentActivity() {
private val viewModel by viewModels<TaskViewModel> {
}
override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
enableEdgeToEdge()
setContent {
Theme {
}
}
}
/* ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- */
@Composable
private fun ScreenScreen() {...}
@Composable
private fun TaskActionButton() {
FloatingActionButton(// Botão flutuante com bordas arredondadas
onClick = { // Lambda contendo a ação de click do usuário
}
) {
Icon(// Layout para desenhar uma imagem
imageVector = Icons.Default.AddTask, // Imagem a qual queremos desenhar
contentDescription = "Nova tarefa" // Recurso de acessibilidade
)
}
}
@Composable
private fun TaskTopBar(scrollBehavior: TopAppBarScrollBehavior) {...}
/* ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- */
ToDoApp › app › src › main › java › com › software › todo › presentation › ui › TaskActivity
104:1 LF UTF-8 To-Do App Dark 4 spaces

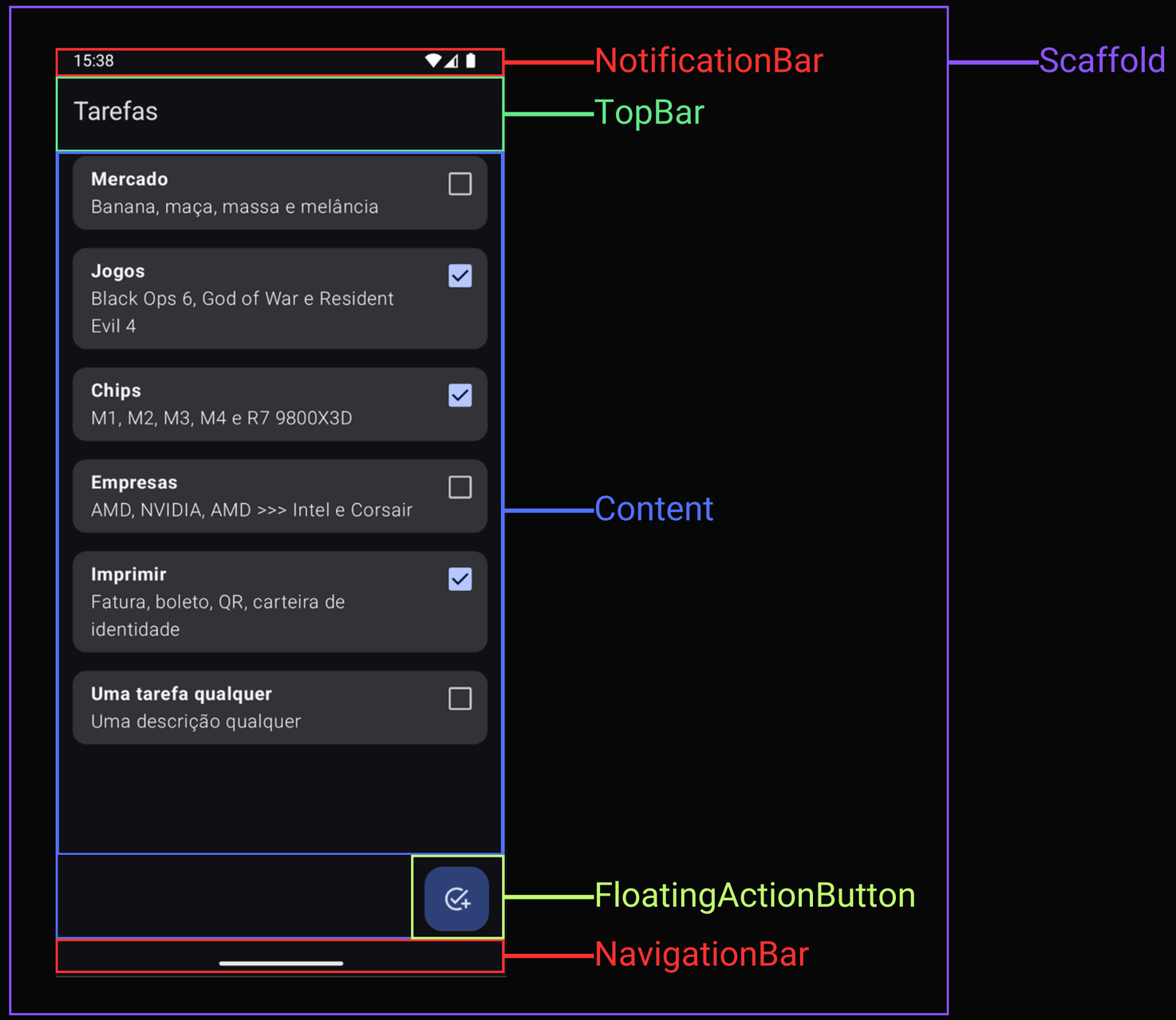

FloatingActionButton

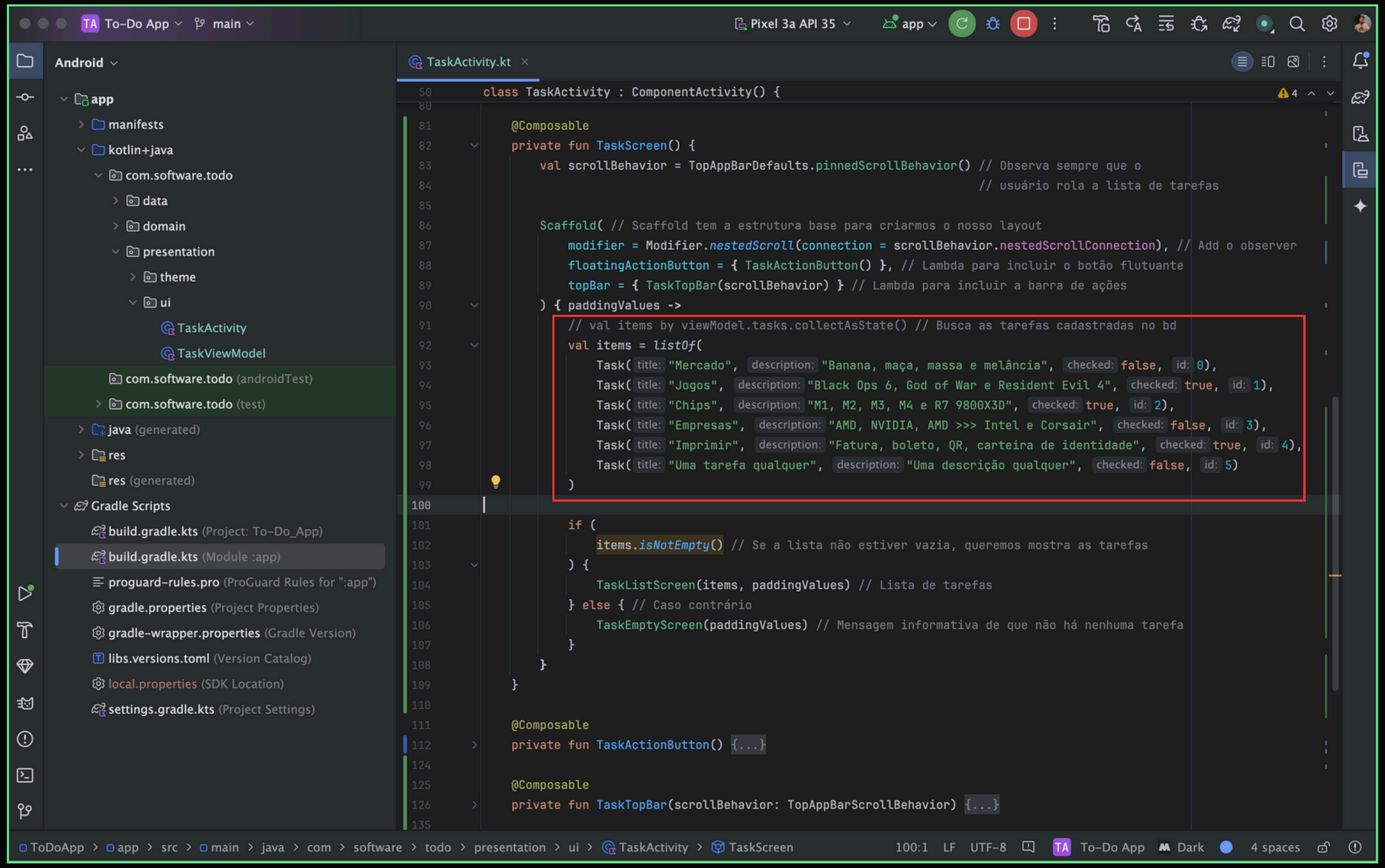

To-Do App
main
Pixel 3a API 35
app
Android
app
manifests
kotlin+java
com.software.todo
data
domain
presentation
theme
ui
TaskActivity
TaskViewModel
com.software.todo (androidTest)
com.software.todo (test)
java (generated)
res
res (generated)
Gradle Scripts
TaskActivity.kt
class TaskActivity : ComponentActivity() {
/* ---- */
@Composable
private fun ScreenScreen() {
val scrollBehavior = TopAppBarDefaults.pinnedScrollBehavior()
Scaffold(
modifier = Modifier.nestedScroll(connection = scrollBehavior.nestedScrollConnection),
floatingActionButton = { TaskActionButton() },
topBar = { TaskTopBar(scrollBehavior) }
) { paddingValues ->
}
}
@Composable
private fun TaskActionButton() {...}
@Composable
private fun TaskTopBar(scrollBehavior: TopAppBarScrollBehavior) {...}
/* ---- */
@Composable // Indica para o compilador que essa função é um layout
private fun TaskListScreen(items: List<Task>, padding: PaddingValues) {...}
@Composable
private fun TaskEmptyScreen(padding: PaddingValues) {...}
@Composable
private fun TaskItem(task: Task) {...}
}
ToDoApp › app › src › main › java › com › software › todo › presentation › ui › TaskActivity › ScreenScreen
87:13
LF
UTF-8
To-Do App
Dark
4 spaces

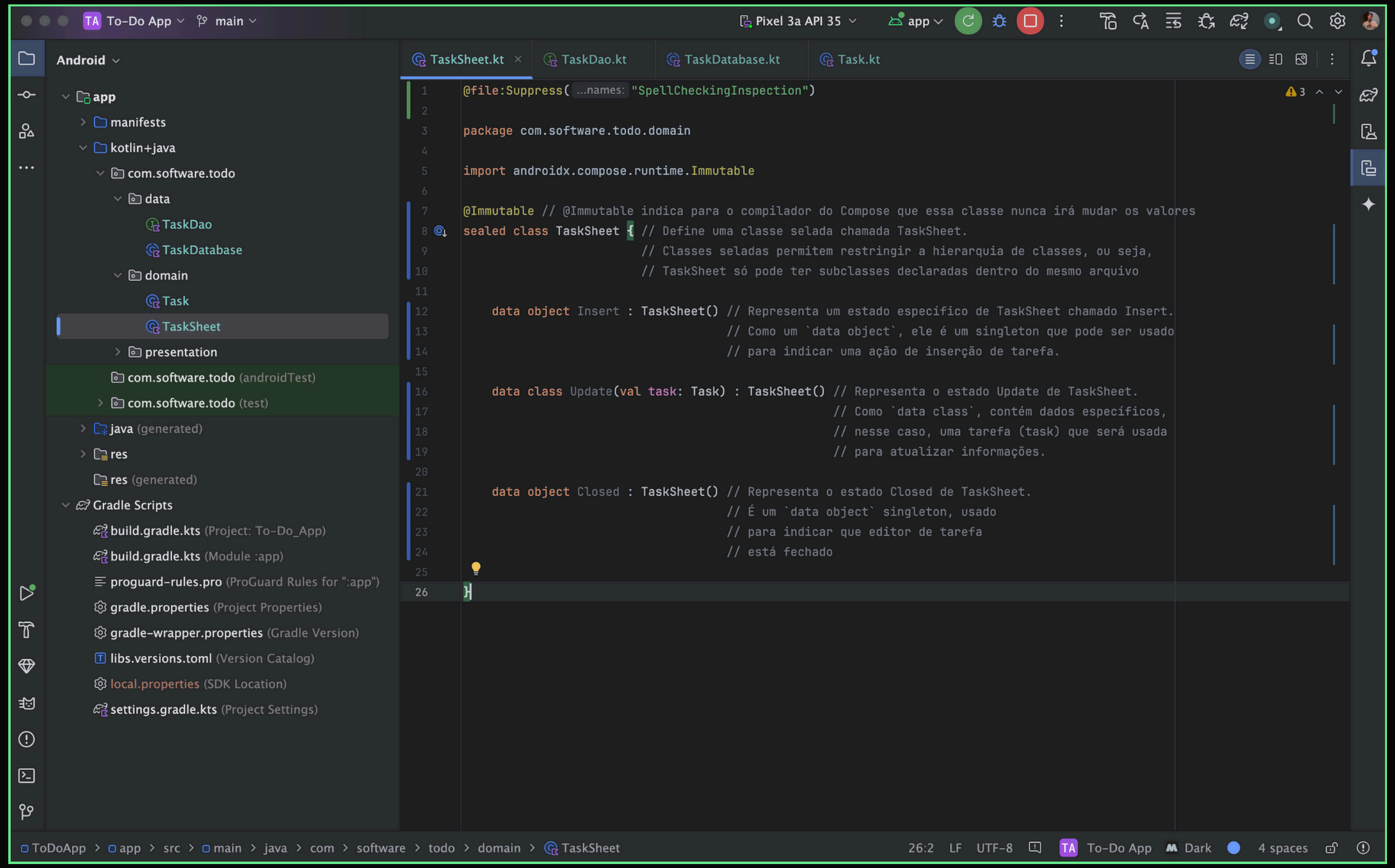

To-Do App
main
Pixel 3a API 35
app
Android
app
manifests
kotlin+java
com.software.todo
data
domain
presentation
theme
ui
TaskActivity
TaskViewModel
com.software.todo (androidTest)
com.software.todo (test)
java (generated)
res
res (generated)
Gradle Scripts
TaskActivity.kt
class TaskActivity : ComponentActivity() {
/* ---- */
@Composable
private fun ScreenScreen() {
val scrollBehavior = TopAppBarDefaults.pinnedScrollBehavior() // Observa sempre que o
// usuário rola a lista de tarefas
Scaffold(// Scaffold tem a estrutura base para criarmos o nosso layout
modifier = Modifier.nestedScroll(connection = scrollBehavior.nestedScrollConnection), // Add o observer
floatingActionButton = { TaskActionButton() }, // Lambda para incluir o botão flutuante
topBar = { TaskTopBar(scrollBehavior) } // Lambda para incluir a barra de ações
) { paddingValues ->
val items by viewModel.tasks.collectAsState() // Busca as tarefas cadastradas no bd
}
}
@Composable
private fun TaskActionButton() {...}
@Composable
private fun TaskTopBar(scrollBehavior: TopAppBarScrollBehavior) {...}
/* ---- */
@Composable // Indica para o compilador que essa função é um layout
private fun TaskListScreen(items: List<Task>, padding: PaddingValues) {...}
@Composable
private fun TaskEmptyScreen(padding: PaddingValues) {...}
@Composable
private fun TaskItem(task: Task) {...}
}
ToDoApp › app › src › main › java › com › software › todo › presentation › ui › TaskActivity
79:1 LF UTF-8 To-Do App Dark 4 spaces

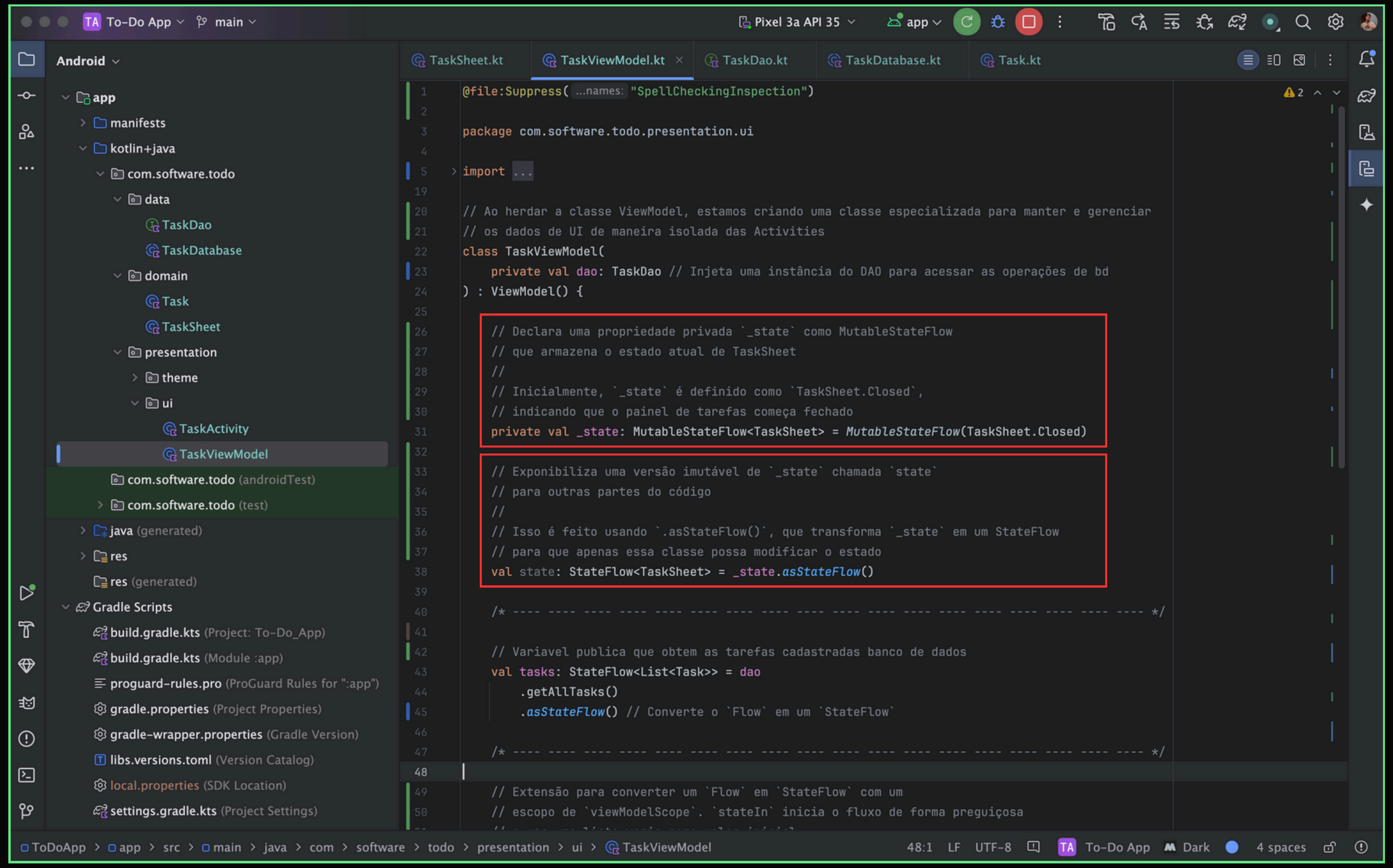

To-Do App
main
Pixel 3a API 35
app
Android
app
manifests
kotlin+java
com.software.todo
data
domain
presentation
theme
ui
TaskActivity
TaskViewModel
com.software.todo (androidTest)
com.software.todo (test)
java (generated)
res
res (generated)
Gradle Scripts
TaskActivity.kt
class TaskActivity : ComponentActivity() {
/* ---- */
@Composable
private fun ScreenScreen() {
val scrollBehavior = TopAppBarDefaults.pinnedScrollBehavior() // Observa sempre que o
// usuário rola a lista de tarefas
Scaffold(// Scaffold tem a estrutura base para criarmos o nosso layout
modifier = Modifier.nestedScroll(connection = scrollBehavior.nestedScrollConnection), // Add o observer
floatingActionButton = { TaskActionButton() }, // Lambda para incluir o botão flutuante
topBar = { TaskTopBar(scrollBehavior) } // Lambda para incluir a barra de ações
) { paddingValues ->
val items by viewModel.tasks.collectAsState() // Busca as tarefas cadastradas no bd
if (
items.isNotEmpty() // Se a lista não estiver vazia, queremos mostra as tarefas
) {
TaskListScreen(items, paddingValues) // Lista de tarefas
} else { // Caso contrário
TaskEmptyScreen(paddingValues) // Mensagem informativa de que não há nenhuma tarefa
}
}
}
@Composable
private fun TaskActionButton() {...}
@Composable
private fun TaskTopBar(scrollBehavior: TopAppBarScrollBehavior) {...}
/* ---- */
@Composable // Indica para o compilador que essa função é um layout
private fun TaskListScreen(items: List<Task>, padding: PaddingValues) {...}
ToDoApp
app
src
main
java
com
software
todo
presentation
ui
TaskActivity
ScreenScreen
101:6
LF
UTF-8
To-Do App
Dark
4 spaces

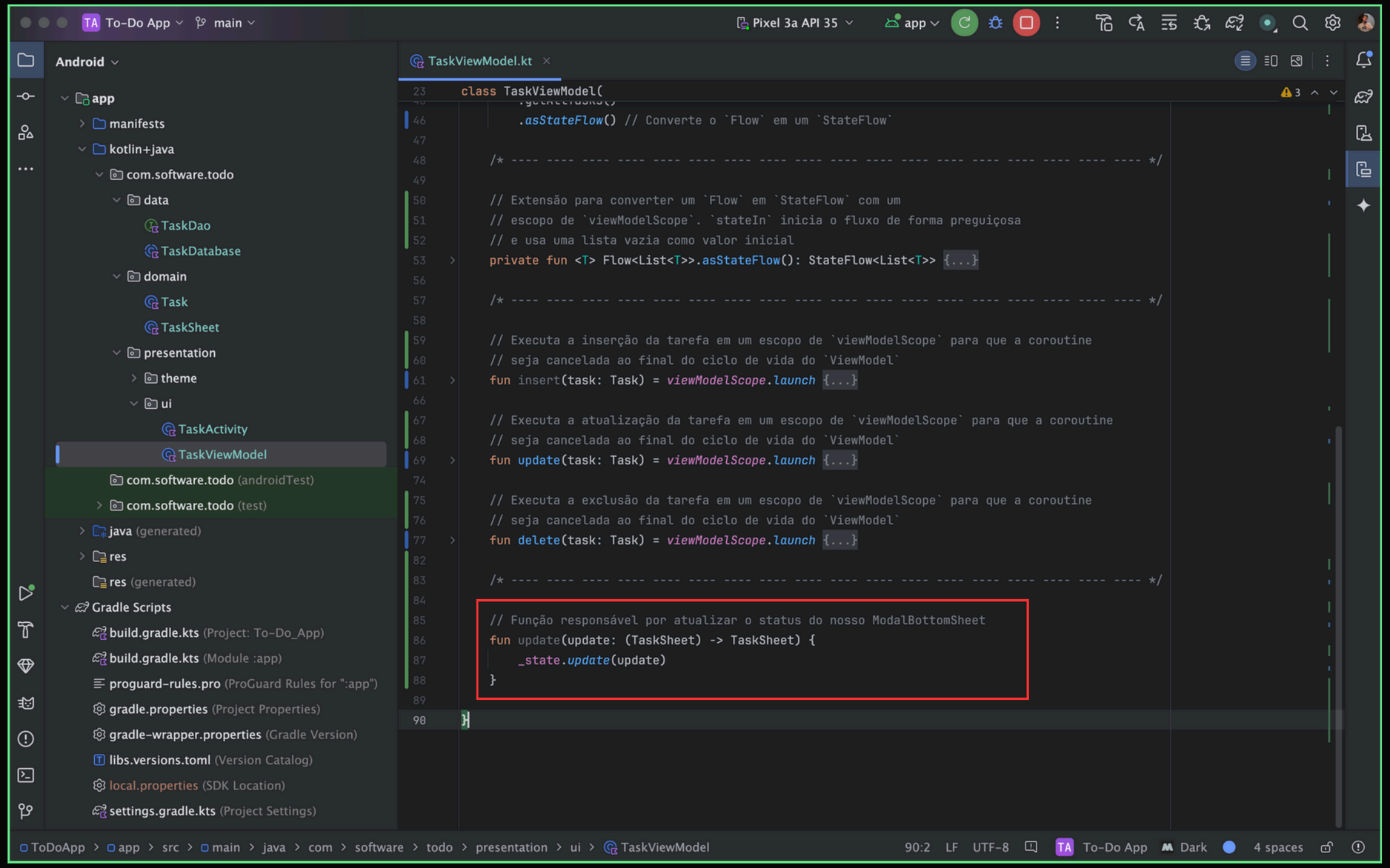

To-Do App
main
Pixel 3a API 35
app
Android
app
manifests
kotlin+java
com.software.todo
data
domain
presentation
theme
ui
TaskActivity
TaskViewModel
com.software.todo (androidTest)
com.software.todo (test)
java (generated)
res
res (generated)
Gradle Scripts
TaskActivity.kt
class TaskActivity : ComponentActivity() {
override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
enableEdgeToEdge()
setContent {
Theme {
TaskScreen()
}
}
}
/* ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- */
@Composable
private fun TaskScreen() {...}
@Composable
private fun TaskActionButton() {...}
@Composable
private fun TaskTopBar(scrollBehavior: TopAppBarScrollBehavior) {...}
/* ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- */
@Composable // Indica para o compilador que essa função é um layout
private fun TaskListScreen(items: List<Task>, padding: PaddingValues) {...}
@Composable
private fun TaskEmptyScreen(padding: PaddingValues) {...}
@Composable
private fun TaskItem(task: Task) {...}
}
ToDoApp
app
src
main
java
com
software
todo
presentation
ui
TaskActivity
78:1
LF
UTF-8
To-Do App
Dark
4 spaces

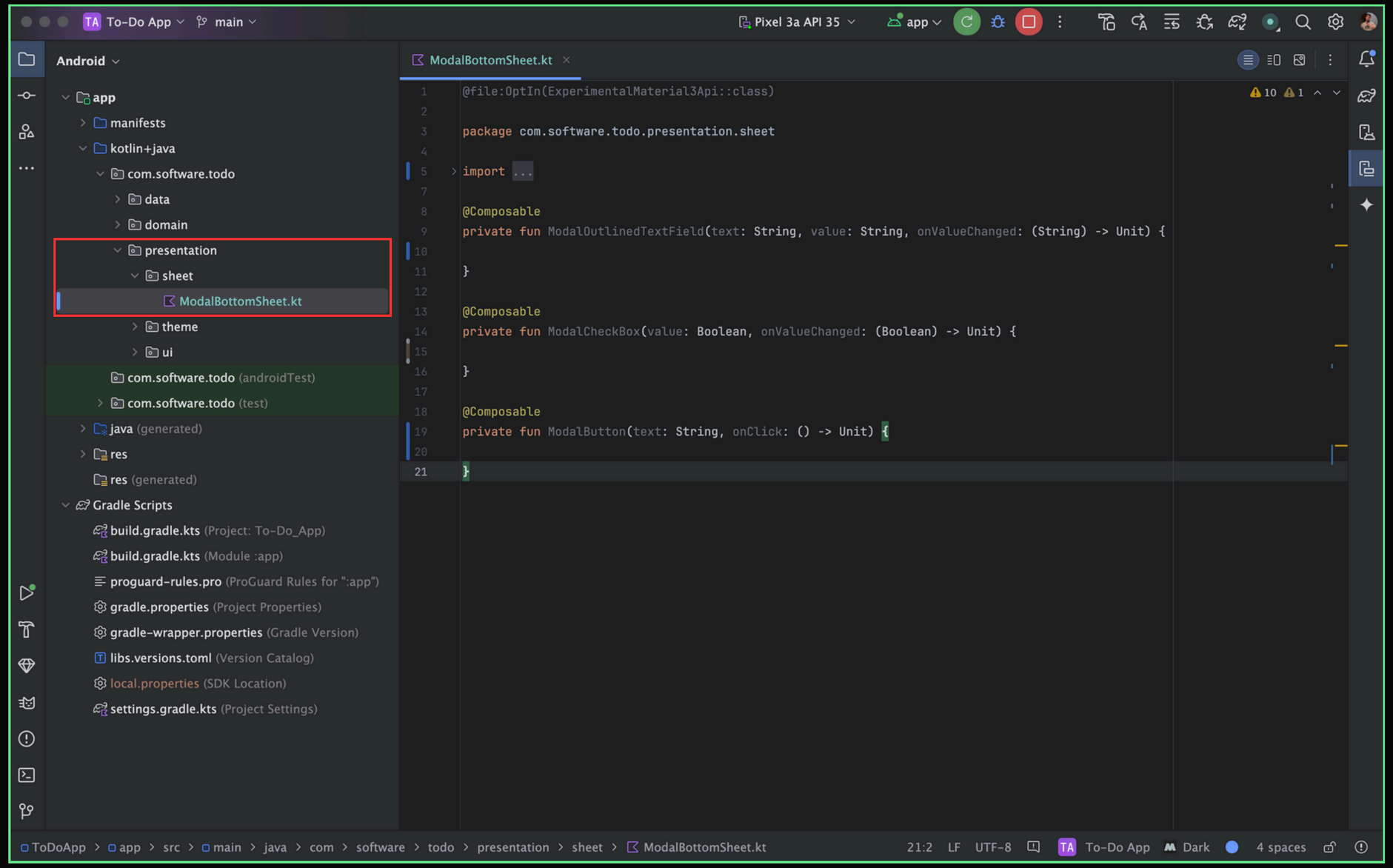

15:38
Tarefas
Mercado
Banana, maça, massa e melância
Jogos
Black Ops 6, God of War e Resident Evil 4
Chips
M1, M2, M3, M4 e R7 9800X3D
Empresas
AMD, NVIDIA, AMD >>> Intel e Corsair
Imprimir
Fatura, boleto, QR, carteira de identidade
Uma tarefa qualquer
Uma descrição qualquer
NotificationBar
TopBar
Content
FloatingActionButton
NavigationBar
Scaffold

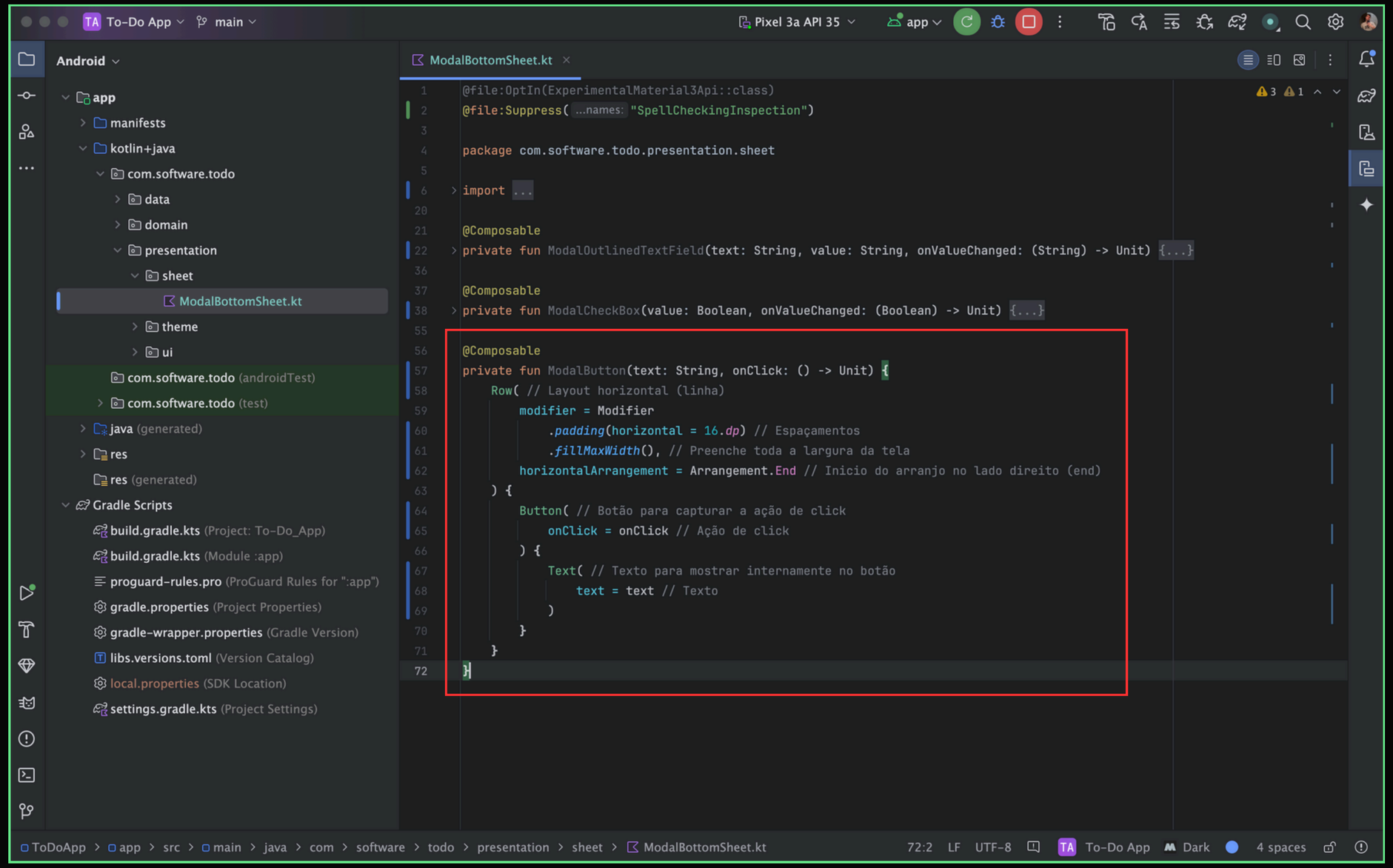

To-Do App
main
Pixel 3a API 35
app
Android
app
manifests
kotlin+java
com.software.todo
data
domain
presentation
theme
ui
TaskActivity
TaskViewModel
com.software.todo (androidTest)
com.software.todo (test)
java (generated)
res
res (generated)
Gradle Scripts
build.gradle.kts (Project: To-Do_App)
build.gradle.kts (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
class TaskActivity : ComponentActivity() {
@Composable
private fun TaskScreen() {
val scrollBehavior = TopAppBarDefaults.pinnedScrollBehavior() // Observa sempre que o
// usuário rola a lista de tarefas
Scaffold(// Scaffold tem a estrutura base para criarmos o nosso layout
modifier = Modifier.nestedScroll(connection = scrollBehavior.nestedScrollConnection), // Add o observer
floatingActionButton = { TaskActionButton() }, // Lambda para incluir o botão flutuante
topBar = { TaskTopBar(scrollBehavior) } // Lambda para incluir a barra de ações
) { paddingValues ->
// val items by viewModel.tasks.collectAsState() // Busca as tarefas cadastradas no bd
val items = listOf(
Task(title: "Mercado", description: "Banana, maça, 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)
)
if (
items.isNotEmpty() // Se a lista não estiver vazia, queremos mostra as tarefas
) {
TaskListScreen(items, paddingValues) // Lista de tarefas
} else { // Caso contrário
TaskEmptyScreen(paddingValues) // Mensagem informativa de que não há nenhuma tarefa
}
}
}
@Composable
private fun TaskActionButton() {...}
@Composable
private fun TaskTopBar(scrollBehavior: TopAppBarScrollBehavior) {...}
ToDoApp › app › src › main › java › com › software › todo › presentation › ui › TaskActivity › TaskScreen
100:1
LF
UTF-8
To-Do App
Dark
4 spaces

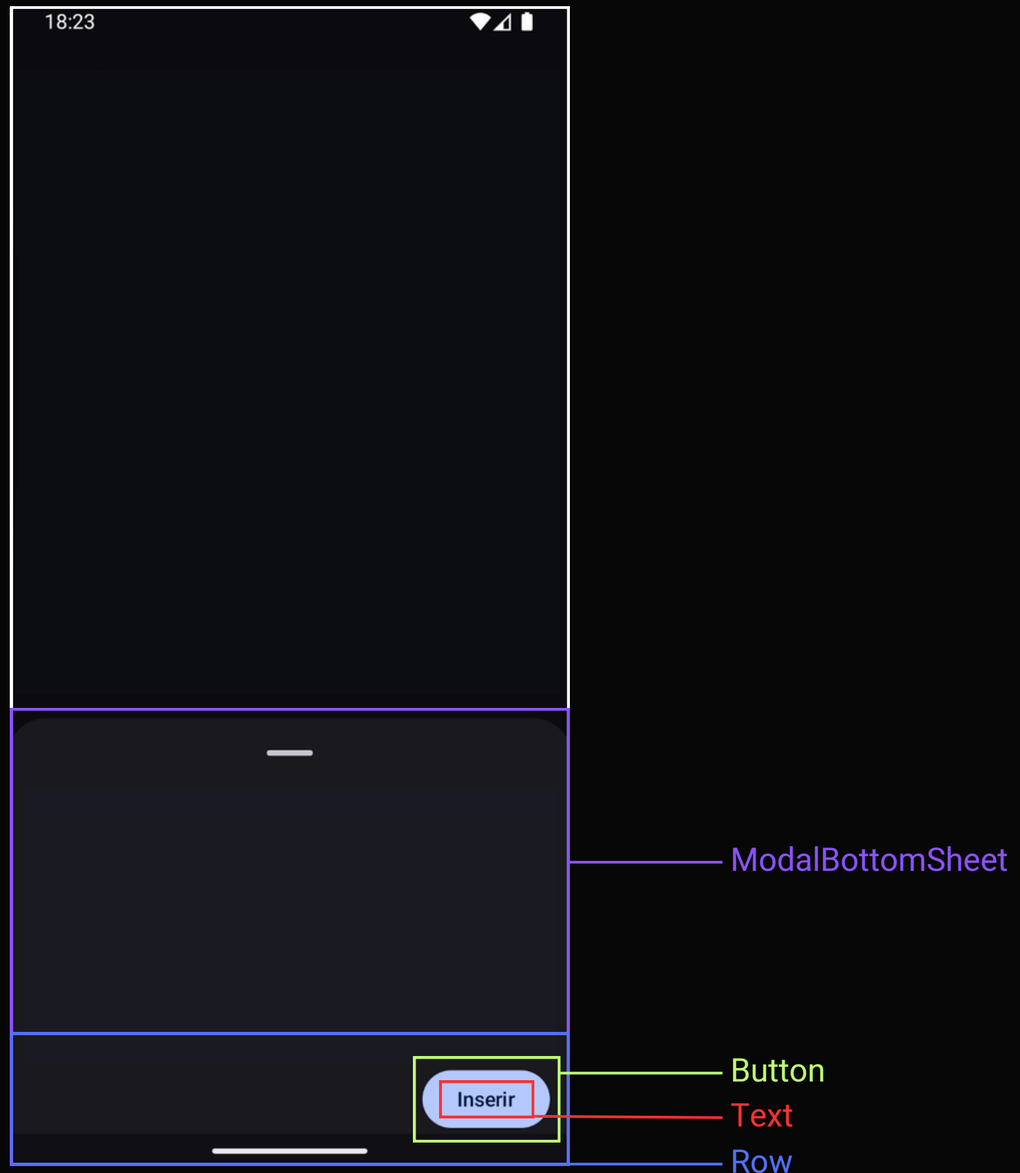

To-Do App
main
Pixel 3a API 35
app
Android
app
manifests
kotlin+java
com.software.todo
data
TaskDao
TaskDatabase
domain
Task
TaskSheet
presentation
com.software.todo (androidTest)
com.software.todo (test)
java (generated)
res
res (generated)
Gradle Scripts
build.gradle.kts (Project: To-Do_App)
build.gradle.kts (Module :app)
proguard-rules.pro (ProGuard Rules for ":app")
gradle.properties (Project Properties)
gradle-wrapper.properties (Gradle Version)
libs.versions.toml (Version Catalog)
local.properties (SDK Location)
settings.gradle.kts (Project Settings)
TaskSheet.kt
TaskDao.kt
TaskDatabase.kt
Task.kt
@file:Suppress(...names: "SpellCheckingInspection")
package com.software.todo.domain
import androidx.compose.runtime.Immutable
@Immutable // @Immutable indica para o compilador do Compose que essa classe nunca irá mudar os valores
sealed class TaskSheet { // Define uma classe selada chamada TaskSheet.
// Classes seladas permitem restringir a hierarquia de classes, ou seja,
// TaskSheet só pode ter subclasses declaradas dentro do mesmo arquivo
data object Insert : TaskSheet() // Representa um estado específico de TaskSheet chamado Insert.
// Como um `data object`, ele é um singleton que pode ser usado
// para indicar uma ação de inserção de tarefa.
data class Update(val task: Task) : TaskSheet() // Representa o estado Update de TaskSheet.
// Como `data class`, contém dados específicos,
// nesse caso, uma tarefa (task) que será usada
// para atualizar informações.
data object Closed : TaskSheet() // Representa o estado Closed de TaskSheet.
// É um `data object` singleton, usado
// para indicar que editor de tarefa
// está fechado
}
ToDoApp › app › src › main › java › com › software › todo › domain › TaskSheet
26:2 LF UTF-8 To-Do App Dark 4 spaces

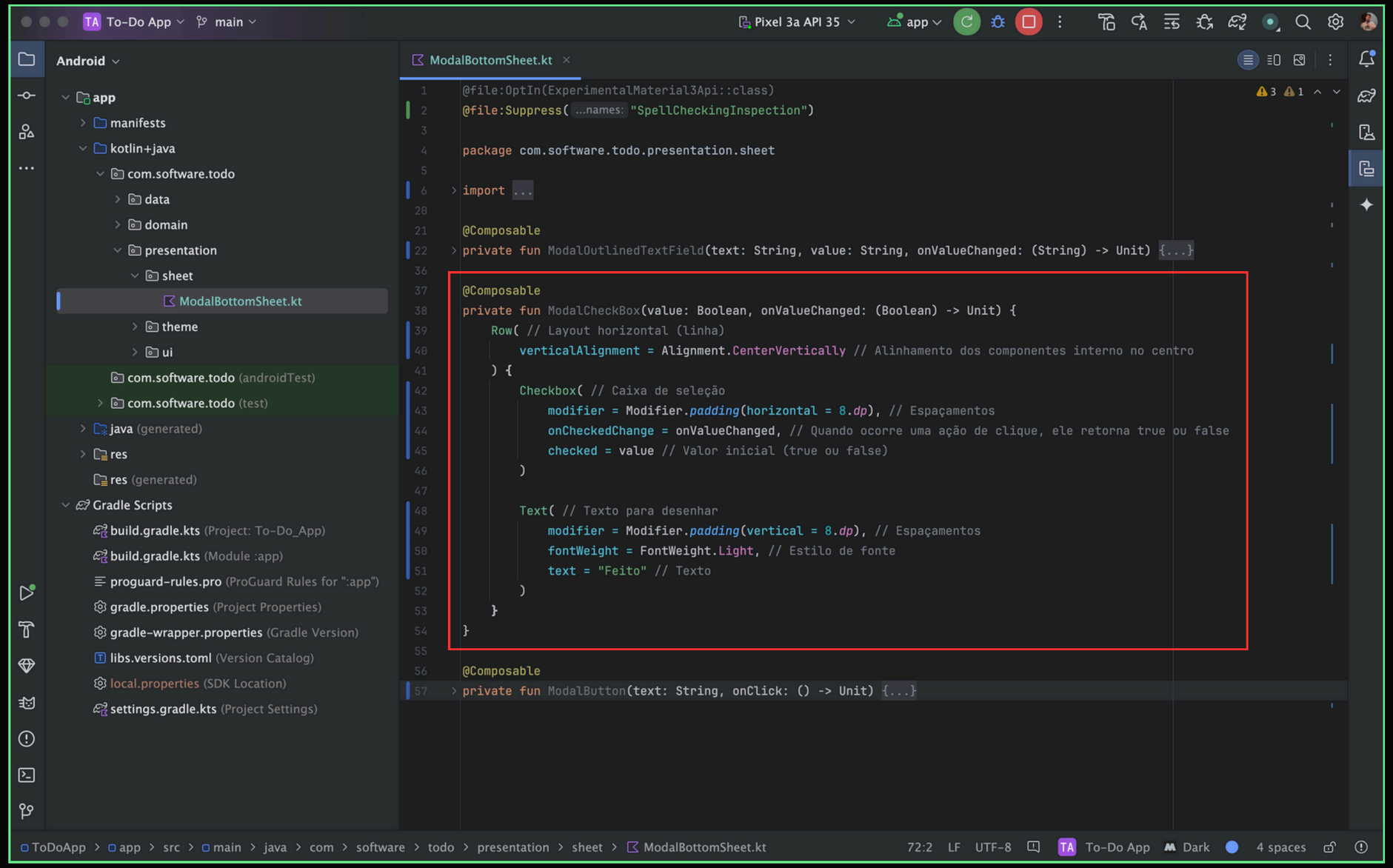

To-Do App
main
Pixel 3a API 35
app
Android
app
manifests
kotlin+java
com.software.todo
data
TaskDao
TaskDatabase
domain
Task
TaskSheet
presentation
theme
ui
TaskActivity
TaskViewModel
com.software.todo (androidTest)
com.software.todo (test)
java (generated)
res
res (generated)
Gradle Scripts
build.gradle.kts (Project: To-Do_App)
build.gradle.kts (Module :app)
proguard-rules.pro (ProGuard Rules for ":app")
gradle.properties (Project Properties)
gradle-wrapper.properties (Gradle Version)
libs.versions.toml (Version Catalog)
local.properties (SDK Location)
settings.gradle.kts (Project Settings)
TaskSheet.kt
TaskViewModel.kt
TaskDao.kt
TaskDatabase.kt
Task.kt
@file:Suppress(...names: "SpellCheckingInspection")
package com.software.todo.presentation.ui
import ...
// Ao herdar a classe ViewModel, estamos criando uma classe especializada para manter e gerenciar
// os dados de UI de maneira isolada das Activities
class TaskViewModel(
private val dao: TaskDao // Injeta uma instância do DAO para acessar as operações de bd
) : ViewModel() {
// Declara uma propriedade privada `_state` como MutableStateFlow
// que armazena o estado atual de TaskSheet
//
// Inicialmente, `_state` é definido como `TaskSheet.Closed`,
// indicando que o painel de tarefas começa fechado
private val _state: MutableStateFlow<TaskSheet> = MutableStateFlow(TaskSheet.Closed)
// Exponibiliza uma versão imutável de `_state` chamada `state`
// para outras partes do código
//
// Isso é feito usando `.asStateFlow()`, que transforma `_state` em um StateFlow
// para que apenas essa classe possa modificar o estado
val state: StateFlow<TaskSheet> = _state.asStateFlow()
/* ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- */
// Variavel publica que obtem as tarefas cadastradas banco de dados
val tasks: StateFlow<List<Task>> = dao
.getAllTasks()
.asStateFlow() // Converte o `Flow` em um `StateFlow`
/* ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- */
// Extensão para converter um `Flow` em `StateFlow` com um
// escopo de `viewModelScope`. `stateIn` inicia o fluxo de forma preguiçosa
ToDoApp
app
src
main
java
com
software
todo
presentation
ui
TaskViewModel
48:1
LF
UTF-8
To-Do App
Dark
4 spaces

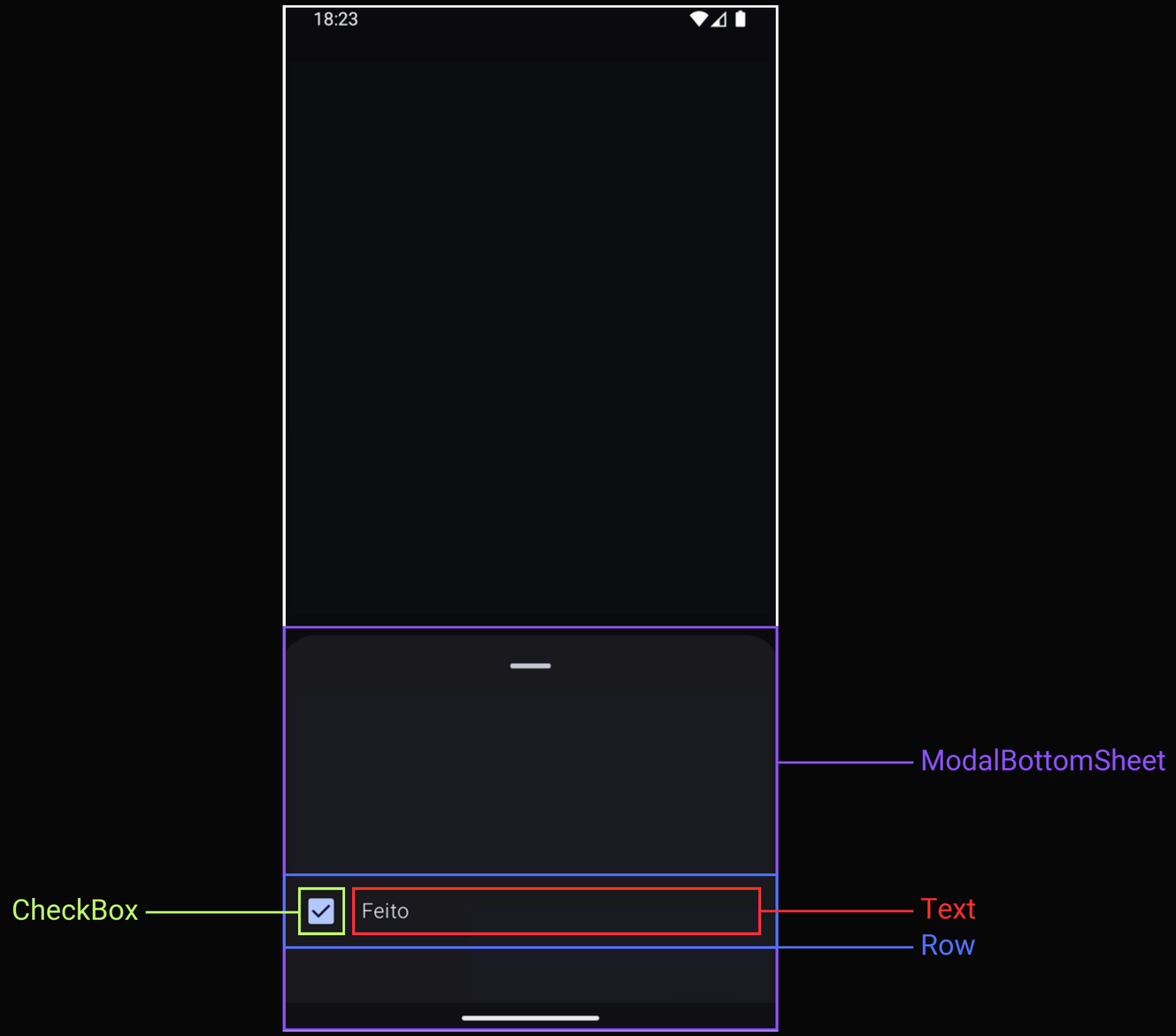

To-Do App
main
Pixel 3a API 35
app
Android
app
manifests
kotlin+java
com.software.todo
data
TaskDao
TaskDatabase
domain
Task
TaskSheet
presentation
theme
ui
TaskActivity
TaskViewModel
com.software.todo (androidTest)
com.software.todo (test)
java (generated)
res
res (generated)
Gradle Scripts
build.gradle.kts (Project: To-Do_App)
build.gradle.kts (Module :app)
proguard-rules.pro (ProGuard Rules for ":app")
gradle.properties (Project Properties)
gradle-wrapper.properties (Gradle Version)
libs.versions.toml (Version Catalog)
local.properties (SDK Location)
settings.gradle.kts (Project Settings)
TaskViewModel.kt
class TaskViewModel(
.asStateFlow() // Converte o `Flow` em um `StateFlow`
/* ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- */
// Extensão para converter um `Flow` em `StateFlow` com um
// escopo de `viewModelScope`. `stateIn` inicia o fluxo de forma preguiçosa
// e usa uma lista vazia como valor inicial
private fun <T> Flow<List<T>>.asStateFlow(): StateFlow<List<T>> {...}
/* ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- */
// Executa a inserção da tarefa em um escopo de `viewModelScope` para que a coroutine
// seja cancelada ao final do ciclo de vida do `ViewModel`
fun insert(task: Task) = viewModelScope.launch {...}
// Executa a atualização da tarefa em um escopo de `viewModelScope` para que a coroutine
// seja cancelada ao final do ciclo de vida do `ViewModel`
fun update(task: Task) = viewModelScope.launch {...}
// Executa a exclusão da tarefa em um escopo de `viewModelScope` para que a coroutine
// seja cancelada ao final do ciclo de vida do `ViewModel`
fun delete(task: Task) = viewModelScope.launch {...}
/* ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- */
// Função responsável por atualizar o status do nosso ModalBottomSheet
fun update(update: (TaskSheet) -> TaskSheet) {
_state.update(update)
}
}
ToDoApp > app > src > main > java > com > software > todo > presentation > ui > TaskViewModel
90:2
LF
UTF-8
To-Do App
Dark
4 spaces

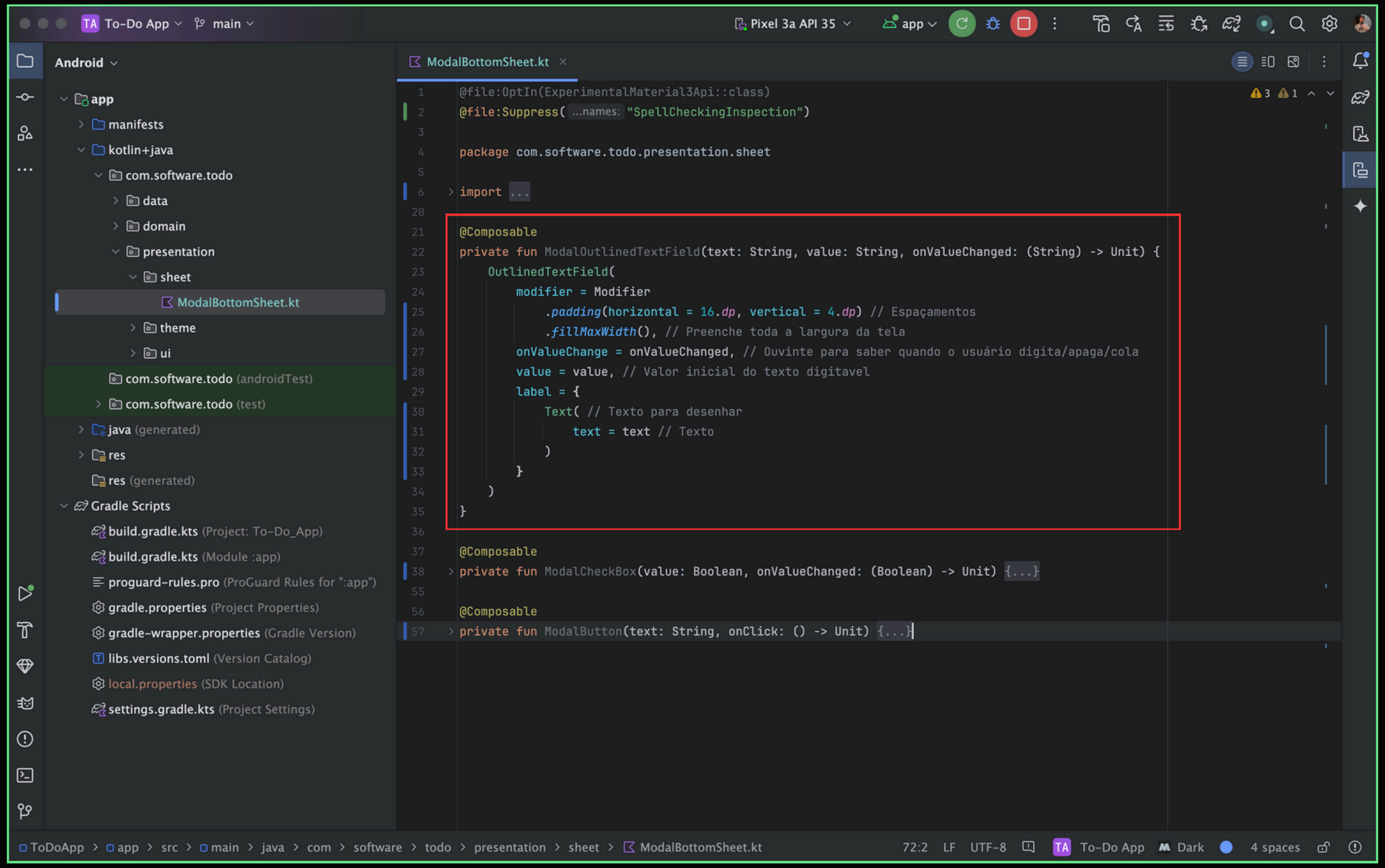

To-Do App
main
Pixel 3a API 35
app
Android
app
manifests
kotlin+java
com.software.todo
data
domain
presentation
sheet
ModalBottomSheet.kt
theme
ui
com.software.todo (androidTest)
com.software.todo (test)
java (generated)
res
res (generated)
Gradle Scripts
build.gradle.kts (Project: To-Do_App)
build.gradle.kts (Module :app)
proguard-rules.pro (ProGuard Rules for ":app")
gradle.properties (Project Properties)
gradle-wrapper.properties (Gradle Version)
libs.versions.toml (Version Catalog)
local.properties (SDK Location)
settings.gradle.kts (Project Settings)
ModalBottomSheet.kt
@file:OptIn(ExperimentalMaterial3Api::class)
package com.software.todo.presentation.sheet
import ...
@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) {
}
10
1
ToDoApp
app
src
main
java
com
software
todo
presentation
sheet
ModalBottomSheet.kt
21:2
LF
UTF-8
To-Do App
Dark
4 spaces

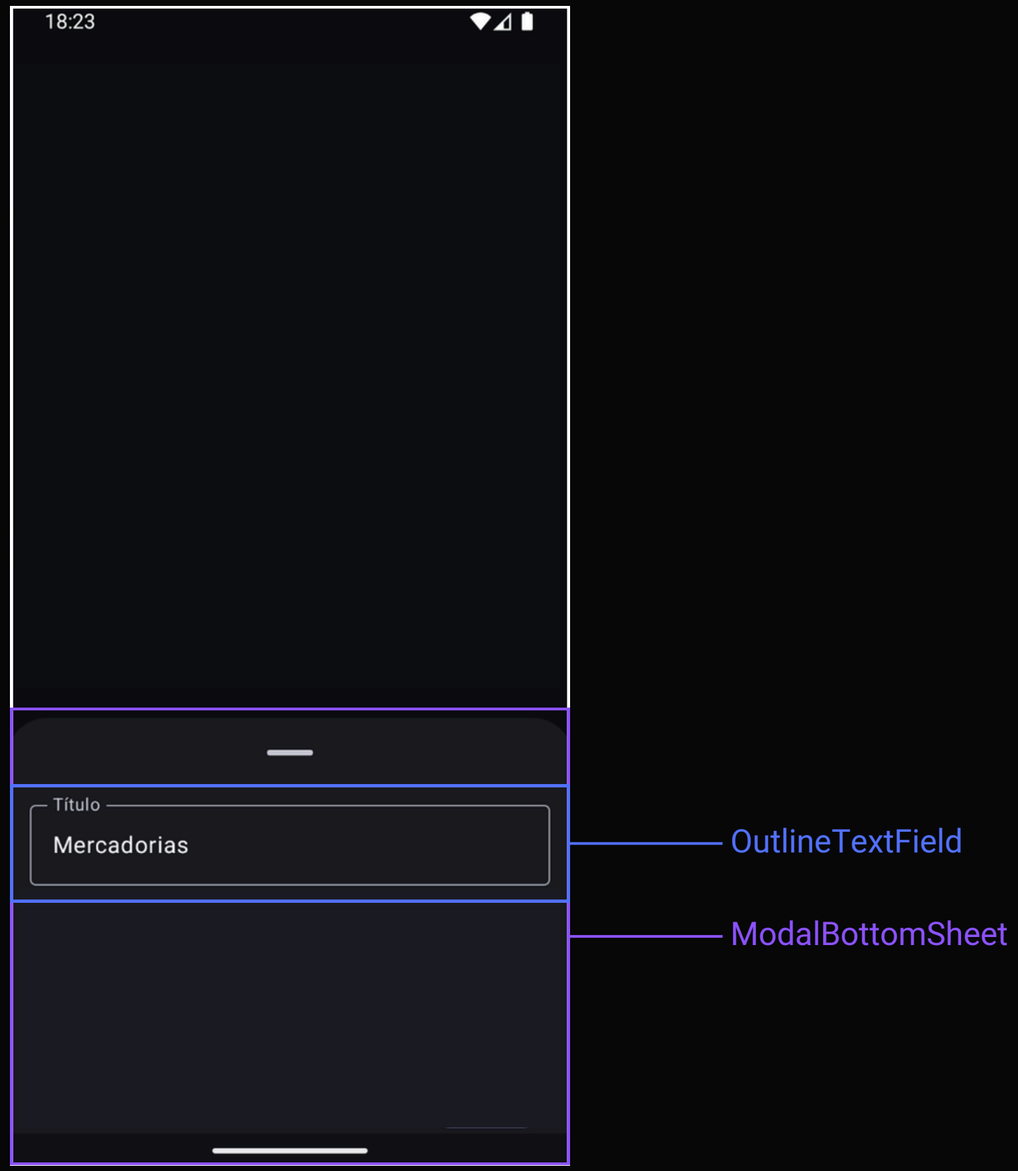

To-Do App
main
Pixel 3a API 35
app
Android
app
manifests
kotlin+java
com.software.todo
data
domain
presentation
sheet
ModalBottomSheet.kt
theme
ui
com.software.todo (androidTest)
com.software.todo (test)
java (generated)
res
res (generated)
Gradle Scripts
build.gradle.kts (Project: To-Do_App)
build.gradle.kts (Module :app)
proguard-rules.pro (ProGuard Rules for ":app")
gradle.properties (Project Properties)
gradle-wrapper.properties (Gradle Version)
libs.versions.toml (Version Catalog)
local.properties (SDK Location)
settings.gradle.kts (Project Settings)
ModalBottomSheet.kt
@file:OptIn(ExperimentalMaterial3Api::class)
@file:Suppress(...names: "SpellCheckingInspection")
package com.software.todo.presentation.sheet
import ...
@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) {
Row(// Layout horizontal (linha)
modifier = Modifier
.padding(horizontal = 16.dp) // Espaçamentos
.fillMaxWidth(), // Preenche toda a largura da tela
horizontalArrangement = Arrangement.End // Inicio do arranjo no lado direito (end)
) {
Button(// Botão para capturar a ação de click
onClick = onClick // Ação de click
) {
Text(// Texto para mostrar internamente no botão
text = text // Texto
)
}
}
}
ToDoApp
app
src
main
java
com
software
todo
presentation
sheet
ModalBottomSheet.kt
72:2
LF
UTF-8
To-Do App
Dark
4 spaces

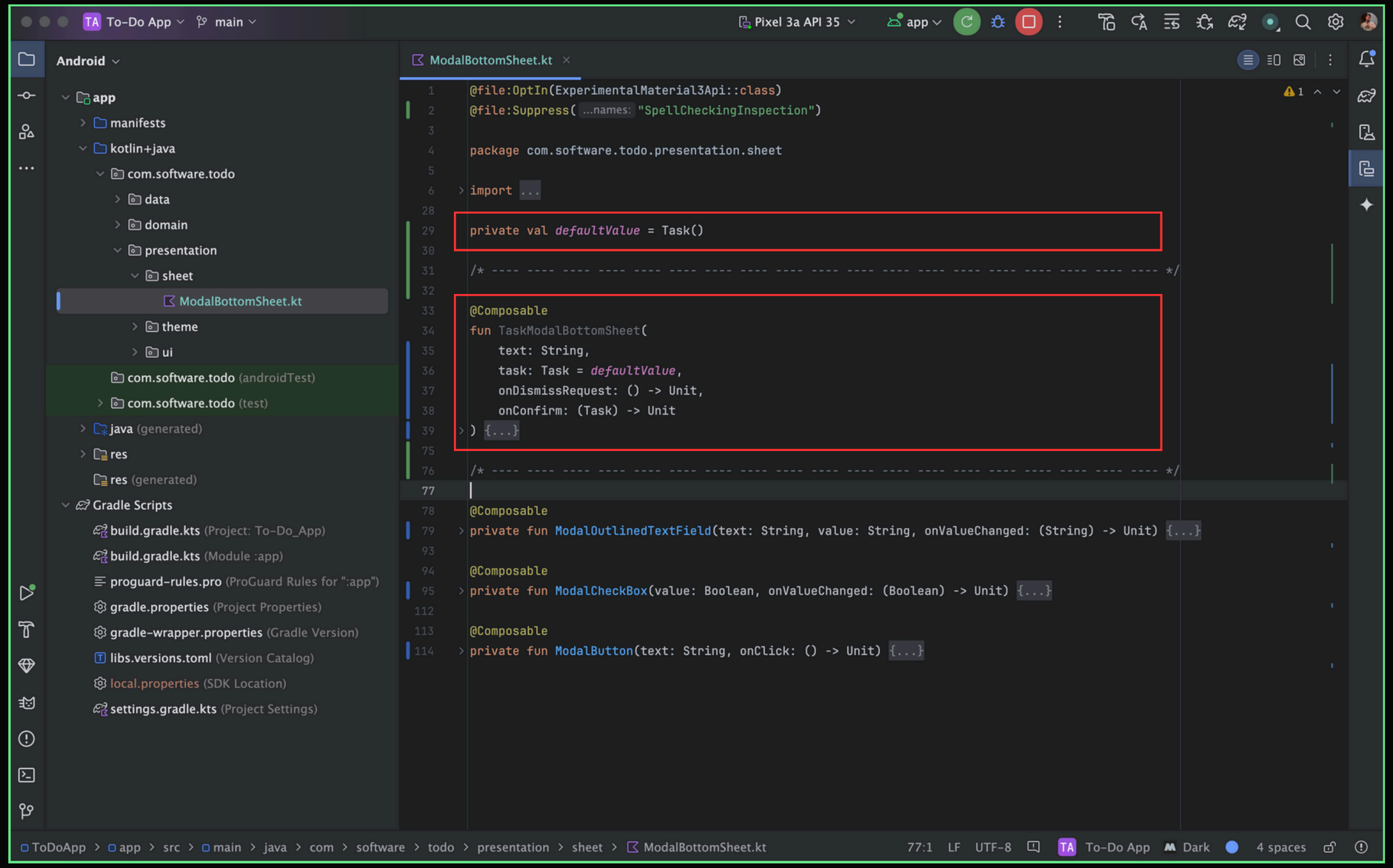

18:23
Inserir
ModalBottomSheet
Button
Text
Row

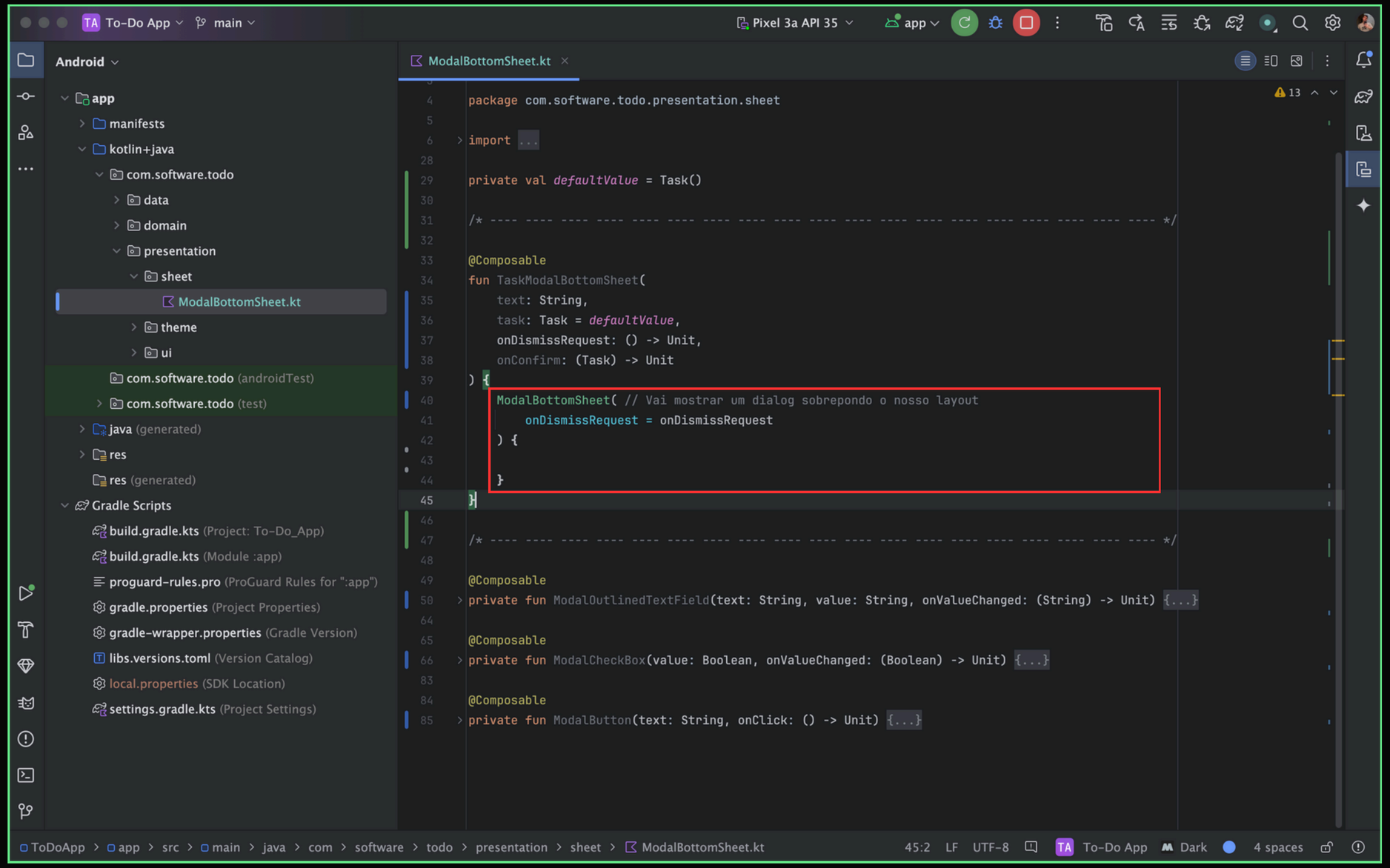

To-Do App
main
Pixel 3a API 35
app
Android
app
manifests
kotlin+java
com.software.todo
data
domain
presentation
sheet
ModalBottomSheet.kt
theme
ui
com.software.todo (androidTest)
com.software.todo (test)
java (generated)
res
res (generated)
Gradle Scripts
build.gradle.kts (Project: To-Do_App)
build.gradle.kts (Module :app)
proguard-rules.pro (ProGuard Rules for ":app")
gradle.properties (Project Properties)
gradle-wrapper.properties (Gradle Version)
libs.versions.toml (Version Catalog)
local.properties (SDK Location)
settings.gradle.kts (Project Settings)
ModalBottomSheet.kt
@file:OptIn(ExperimentalMaterial3Api::class)
@file:Suppress(...names: "SpellCheckingInspection")
package com.software.todo.presentation.sheet
import ...
@Composable
private fun ModalOutlinedTextField(text: String, value: String, onValueChanged: (String) -> Unit) {...}
@Composable
private fun ModalCheckBox(value: Boolean, onValueChanged: (Boolean) -> Unit) {
Row(// Layout horizontal (linha)
verticalAlignment = Alignment.CenterVertically // Alinhamento dos componentes interno no centro
) {
Checkbox(// Caixa de seleção
modifier = Modifier.padding(horizontal = 8.dp), // Espaçamentos
onCheckedChange = onValueChanged, // Quando ocorre uma ação de clique, ele retorna true ou false
checked = value // Valor inicial (true ou false)
)
Text(// Texto para desenhar
modifier = Modifier.padding(vertical = 8.dp), // Espaçamentos
fontWeight = FontWeight.Light, // Estilo de fonte
text = "Feito" // Texto
)
}
}
@Composable
private fun ModalButton(text: String, onClick: () -> Unit) {...}
ToDoApp
app
src
main
java
com
software
todo
presentation
sheet
ModalBottomSheet.kt
72:2
LF
UTF-8
To-Do App
Dark
4 spaces

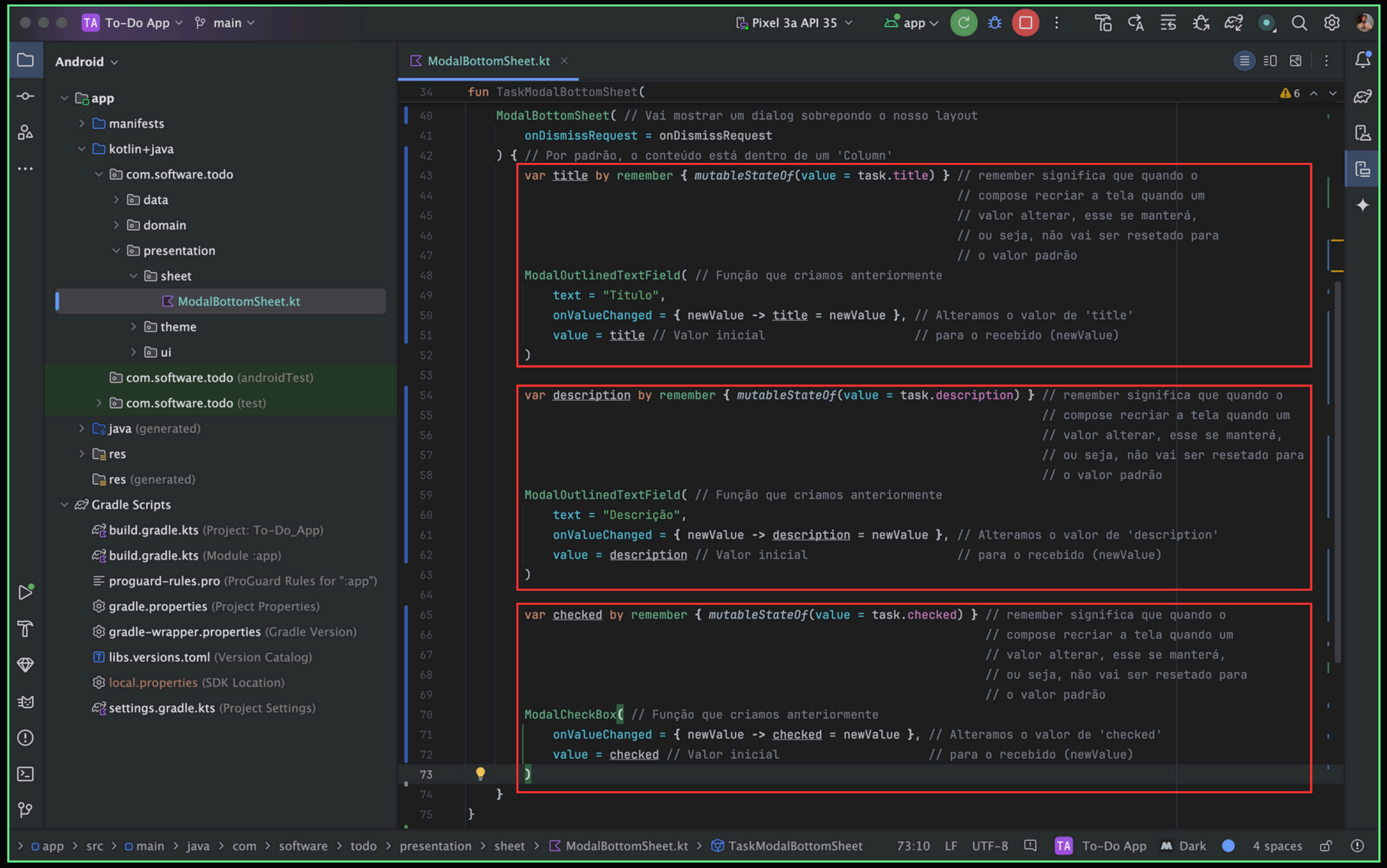

18:23
ModalBottomSheet
CheckBox
Feito
Text
Row

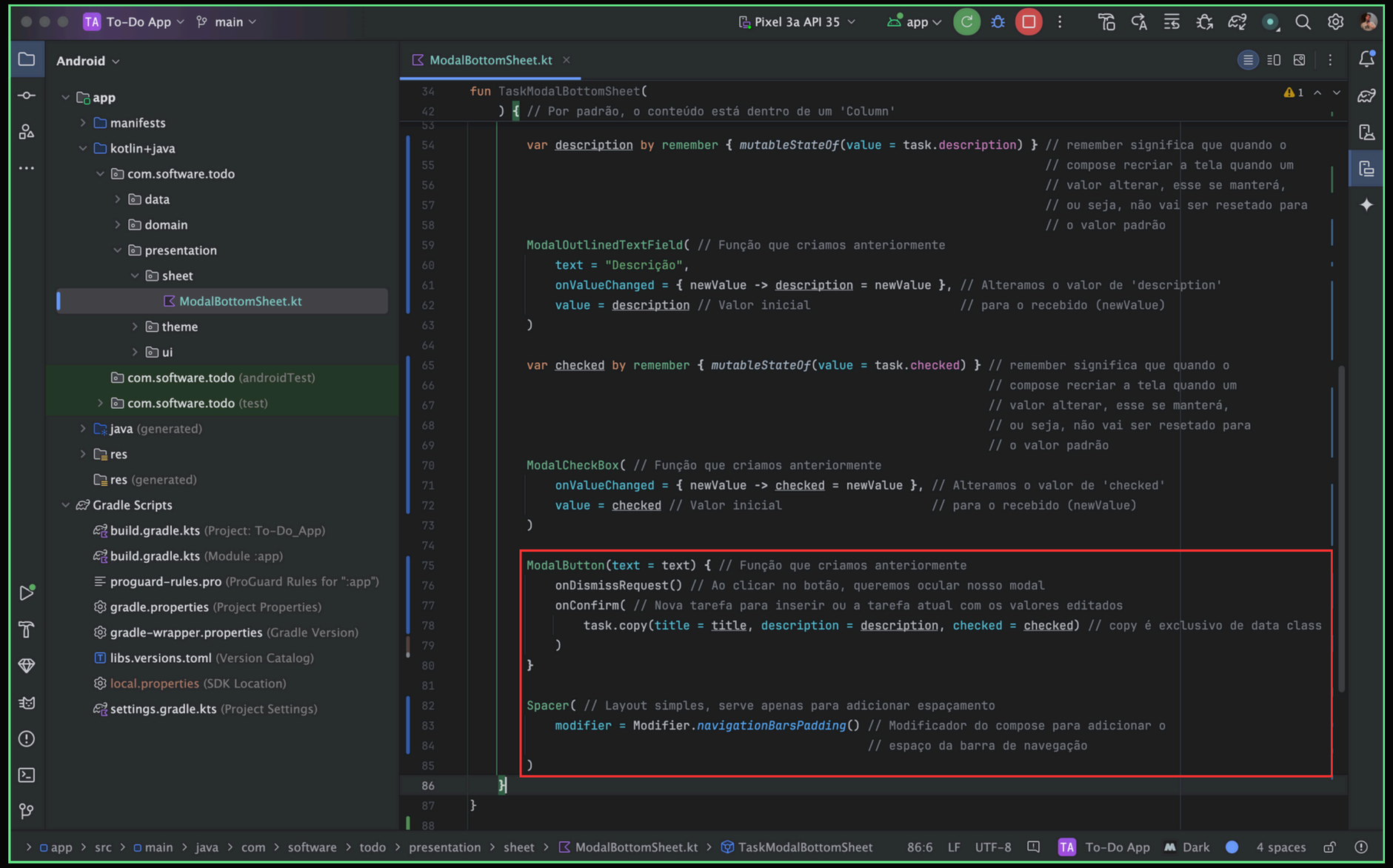

To-Do App
main
Pixel 3a API 35
app
Android
app
manifests
kotlin+java
com.software.todo
data
domain
presentation
sheet
ModalBottomSheet.kt
theme
ui
com.software.todo (androidTest)
com.software.todo (test)
java (generated)
res
res (generated)
Gradle Scripts
build.gradle.kts (Project: To-Do_App)
build.gradle.kts (Module :app)
proguard-rules.pro (ProGuard Rules for ":app")
gradle.properties (Project Properties)
gradle-wrapper.properties (Gradle Version)
libs.versions.toml (Version Catalog)
local.properties (SDK Location)
settings.gradle.kts (Project Settings)
ModalBottomSheet.kt
@file:OptIn(ExperimentalMaterial3Api::class)
@file:Suppress(...names: "SpellCheckingInspection")
package com.software.todo.presentation.sheet
import ...
@Composable
private fun ModalOutlinedTextField(text: String, value: String, onValueChanged: (String) -> Unit) {
OutlinedTextField(
modifier = Modifier
.padding(horizontal = 16.dp, vertical = 4.dp) // Espaçamentos
.fillMaxWidth(), // Preenche toda a largura da tela
onValueChange = onValueChanged, // Ouvinte para saber quando o usuário digita/apaga/cola
value = value, // Valor inicial do texto digitavel
label = {
Text(// Texto para desenhar
text = text // Texto
)
}
)
}
@Composable
private fun ModalCheckBox(value: Boolean, onValueChanged: (Boolean) -> Unit) {...}
@Composable
private fun ModalButton(text: String, onClick: () -> Unit) {...}
ToDoApp
app
src
main
java
com
software
todo
presentation
sheet
ModalBottomSheet.kt
72:2
LF
UTF-8
To-Do App
Dark
4 spaces

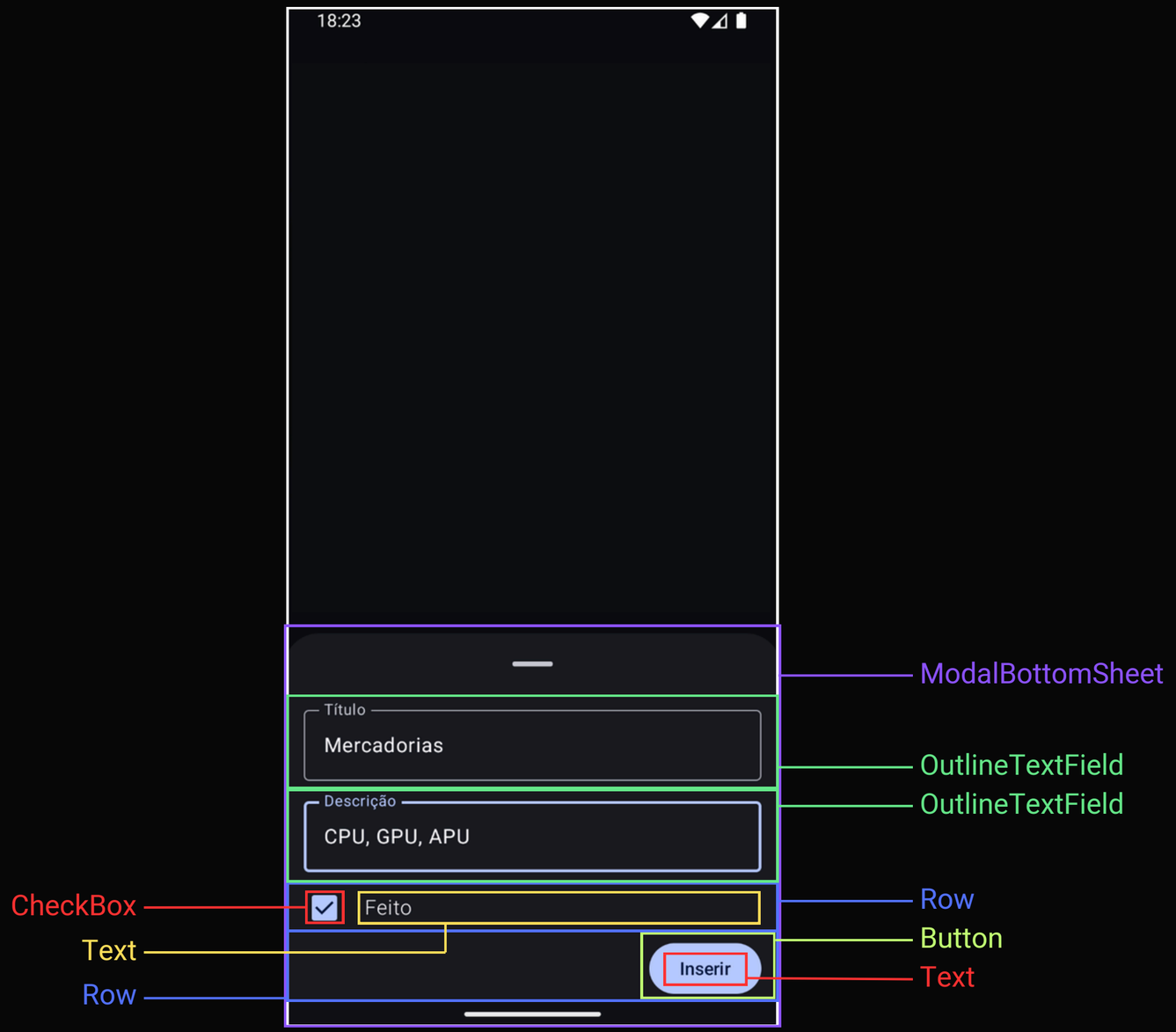

18:23
Título
Mercadorias
OutlineTextField
ModalBottomSheet

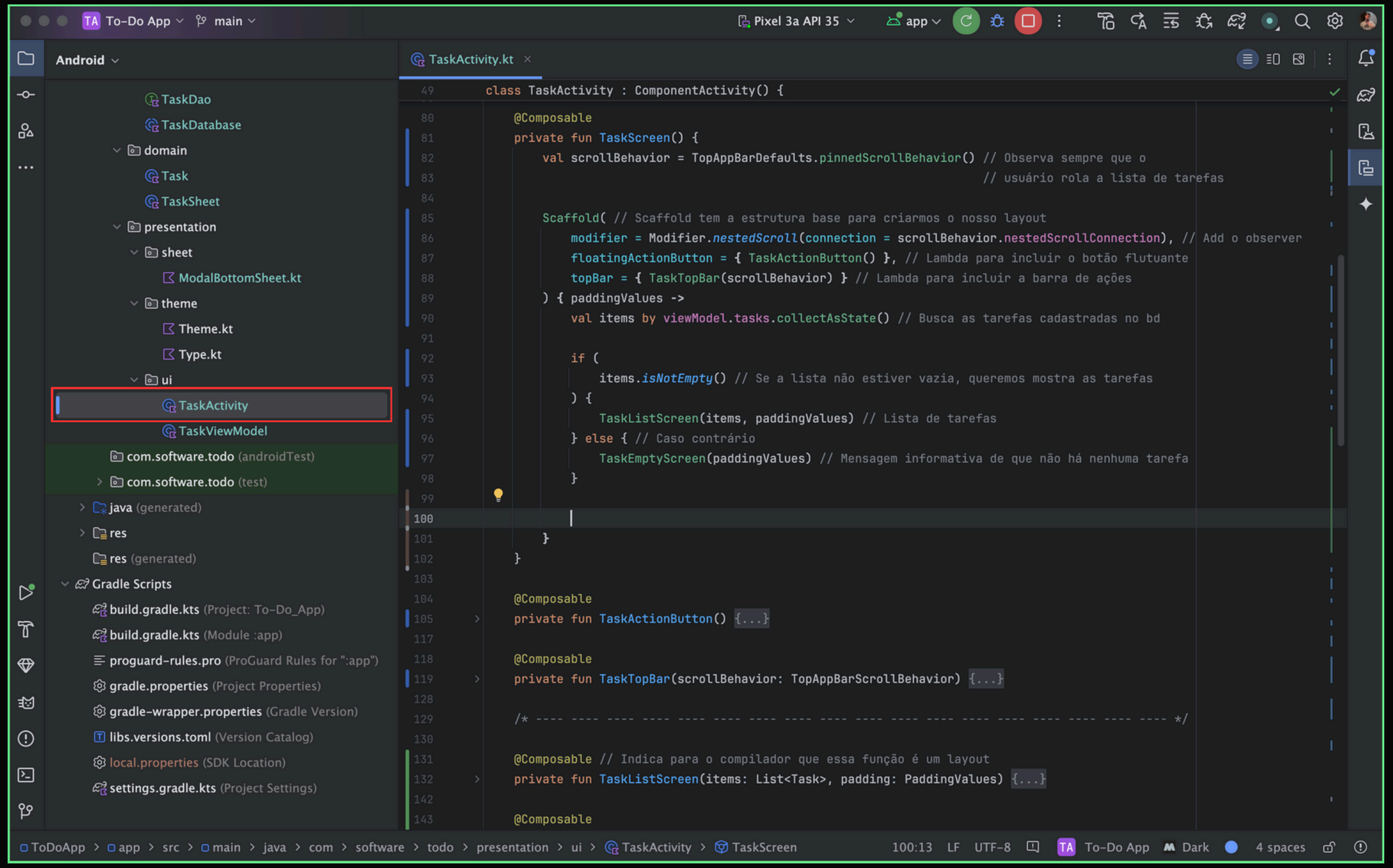

To-Do App
main
Pixel 3a API 35
app
Android
ModalBottomSheet.kt
app
manifests
kotlin+java
com.software.todo
data
domain
presentation
sheet
ModalBottomSheet.kt
theme
ui
com.software.todo (androidTest)
com.software.todo (test)
java (generated)
res
res (generated)
Gradle Scripts
build.gradle.kts (Project: To-Do_App)
build.gradle.kts (Module :app)
proguard-rules.pro (ProGuard Rules for ":app")
gradle.properties (Project Properties)
gradle-wrapper.properties (Gradle Version)
libs.versions.toml (Version Catalog)
local.properties (SDK Location)
settings.gradle.kts (Project Settings)
@file:OptIn(ExperimentalMaterial3Api::class)
@file:Suppress(...names: "SpellCheckingInspection")
package com.software.todo.presentation.sheet
import ...
private val defaultValue = Task()
/* ---- */
@Composable
fun TaskModalBottomSheet(
text: String,
task: Task = defaultValue,
onDismissRequest: () -> Unit,
onConfirm: (Task) -> Unit
) {...}
/* ---- */
@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) {...}
ToDoApp
app
src
main
java
com
software
todo
presentation
sheet
ModalBottomSheet.kt
77:1
LF
UTF-8
To-Do App
Dark
4 spaces

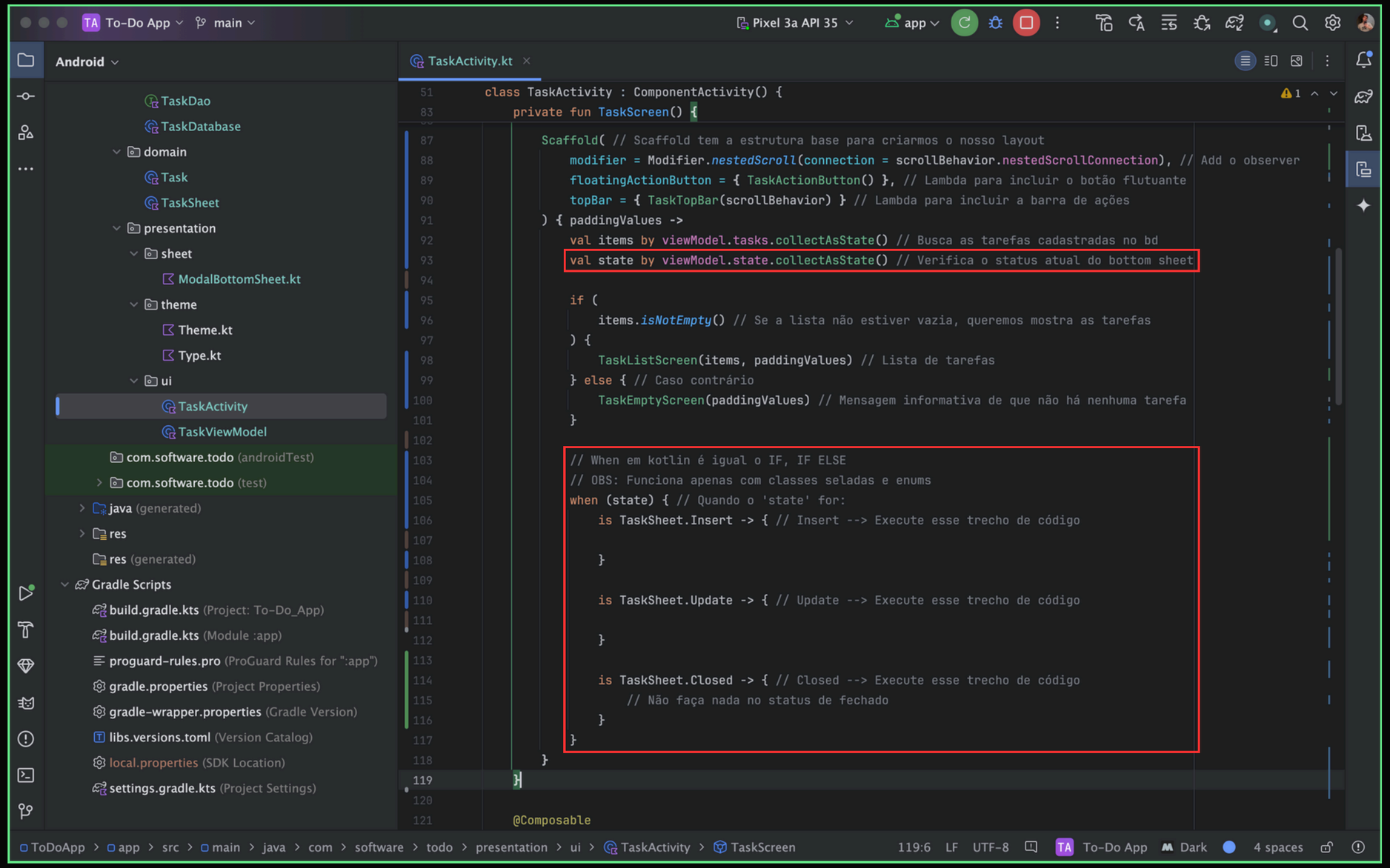

To-Do App
main
Pixel 3a API 35
app
Android
ModalBottomSheet.kt
app
manifests
kotlin+java
com.software.todo
data
domain
presentation
sheet
ModalBottomSheet.kt
theme
ui
com.software.todo (androidTest)
com.software.todo (test)
java (generated)
res
res (generated)
Gradle Scripts
build.gradle.kts (Project: To-Do_App)
build.gradle.kts (Module :app)
proguard-rules.pro (ProGuard Rules for ":app")
gradle.properties (Project Properties)
gradle-wrapper.properties (Gradle Version)
libs.versions.toml (Version Catalog)
local.properties (SDK Location)
settings.gradle.kts (Project Settings)
4 package com.software.todo.presentation.sheet
6 import ...
29 private val defaultValue = Task()
31 /* ---- */
33 @Composable
34 fun TaskModalBottomSheet(
35 text: String,
36 task: Task = defaultValue,
37 onDismissRequest: () -> Unit,
38 onConfirm: (Task) -> Unit
39) {
40 ModalBottomSheet(// Vai mostrar um dialog sobrepondo o nosso layout
41 onDismissRequest = onDismissRequest
42) {
44 }
45 }
47 /* ---- */
49 @Composable
50 private fun ModalOutlinedTextField(text: String, value: String, onValueChanged: (String) -> Unit) {...}
65 @Composable
66 private fun ModalCheckBox(value: Boolean, onValueChanged: (Boolean) -> Unit) {...}
84 @Composable
85 private fun ModalButton(text: String, onClick: () -> Unit) {...}
13
ToDoApp > app > src > main > java > com > software > todo > presentation > sheet > ModalBottomSheet.kt
45:2 LF UTF-8 To-Do App Dark 4 spaces

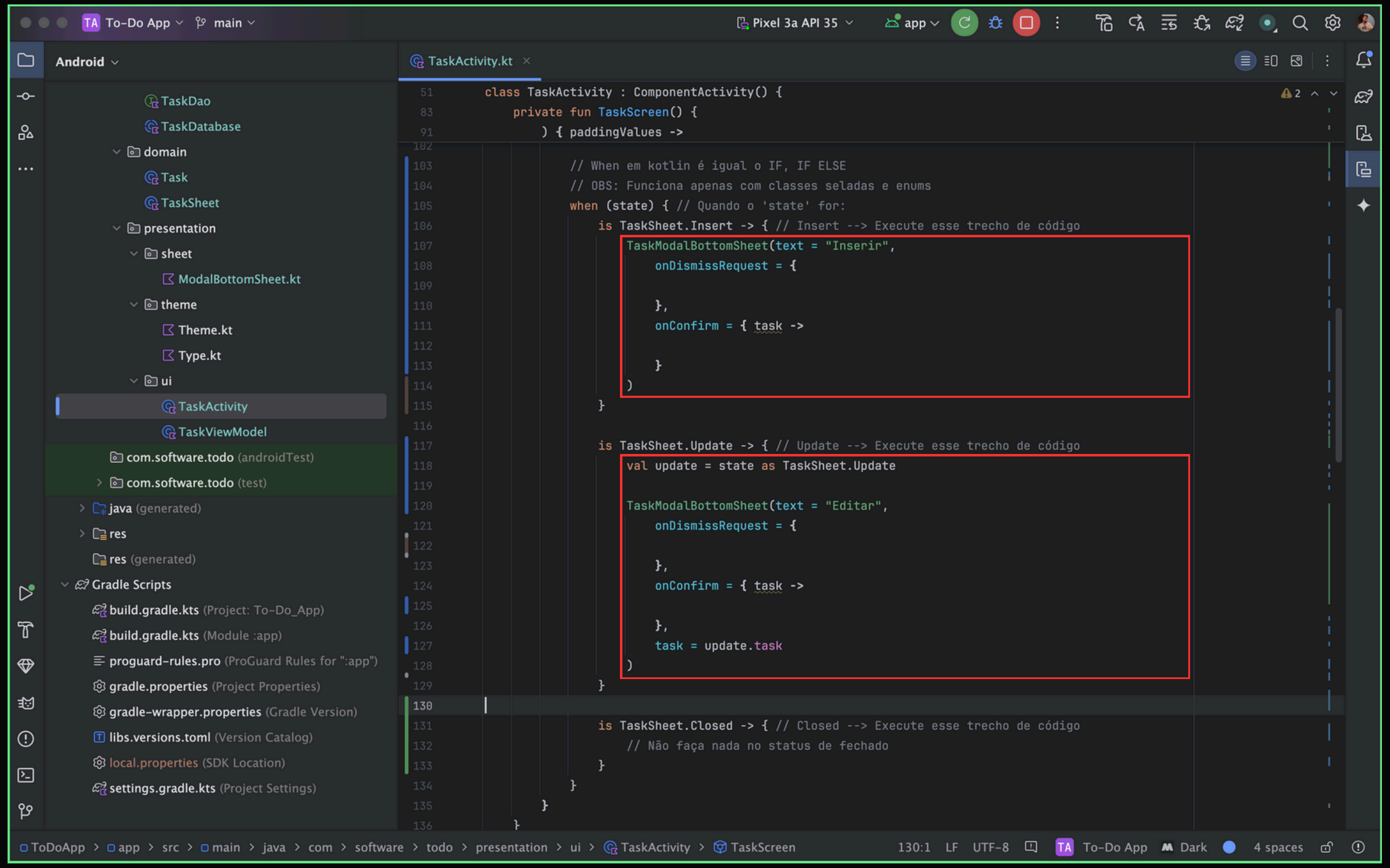

To-Do App
main
Pixel 3a API 35
app
Android
app
manifests
kotlin+java
com.software.todo
data
domain
presentation
sheet
ModalBottomSheet.kt
theme
ui
com.software.todo (androidTest)
com.software.todo (test)
java (generated)
res
res (generated)
Gradle Scripts
build.gradle.kts (Project: To-Do_App)
build.gradle.kts (Module :app)
proguard-rules.pro (ProGuard Rules for ":app")
gradle.properties (Project Properties)
gradle-wrapper.properties (Gradle Version)
libs.versions.toml (Version Catalog)
local.properties (SDK Location)
settings.gradle.kts (Project Settings)
ModalBottomSheet.kt
fun TaskModalBottomSheet(
ModalBottomSheet(// Vai mostrar um dialog sobrepondo o nosso layout
onDismissRequest = onDismissRequest
) { // Por padrão, o conteúdo está dentro de um 'Column'
var title by remember { mutableStateOf(value = task.title) } // remember significa que quando o
// compose recriar a tela quando um
// valor alterar, esse se manterá,
// ou seja, não vai ser resetado para
// o valor padrão
ModalOutlinedTextField(// Função que criamos anteriormente
text = "Título",
onValueChanged = { newValue -> title = newValue }, // Alteramos o valor de 'title'
value = title // Valor inicial // para o recebido (newValue)
)
var description by remember { mutableStateOf(value = task.description) } // remember significa que quando o
// compose recriar a tela quando um
// 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
// compose recriar a tela quando um
// 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)
)
}
}
app > src > main > java > com > software > todo > presentation > sheet > ModalBottomSheet.kt > TaskModalBottomSheet
73:10 LF UTF-8 To-Do App Dark 4 spaces

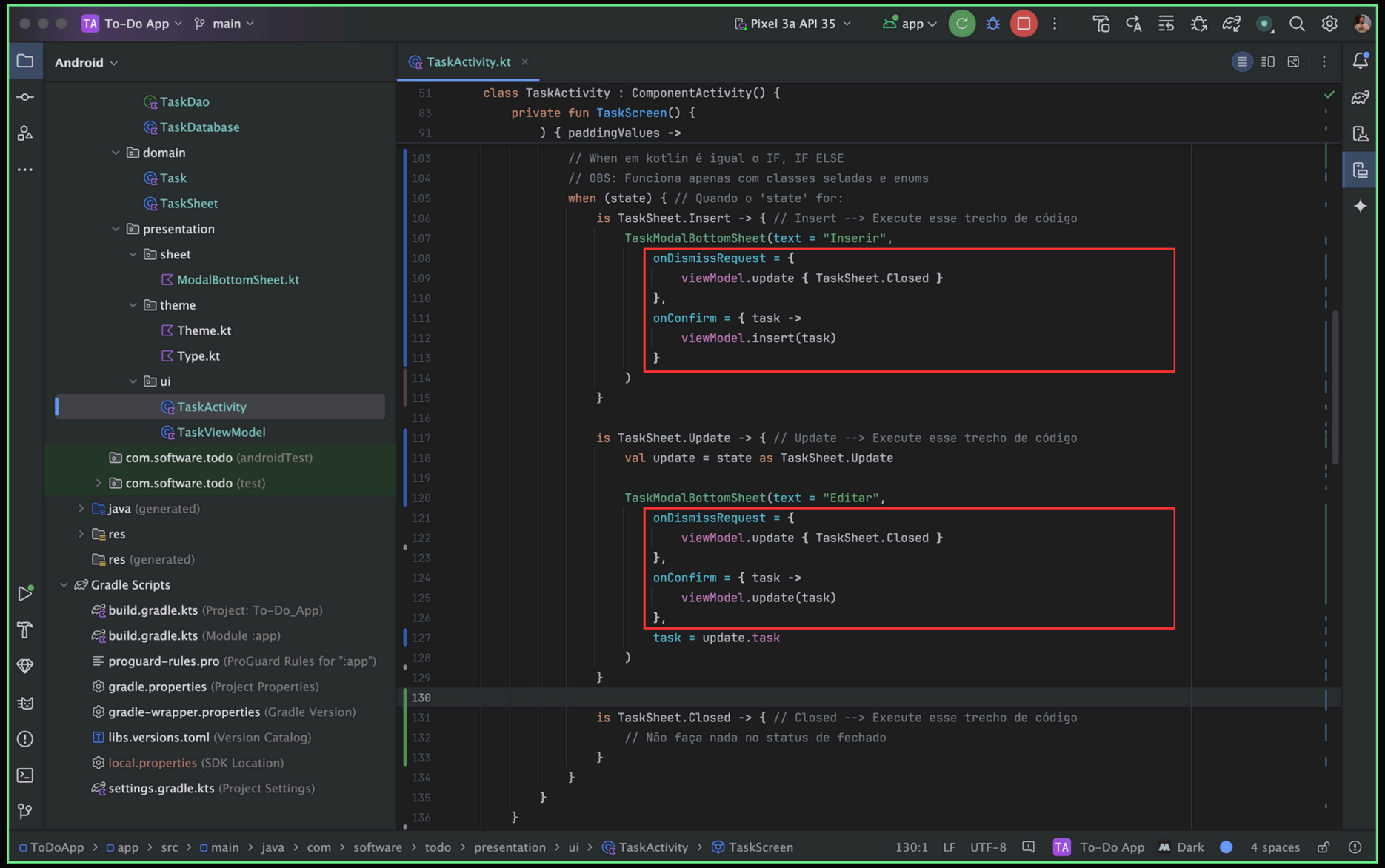

To-Do App
main
Pixel 3a API 35
app
Android
app
manifests
kotlin+java
com.software.todo
data
domain
presentation
sheet
ModalBottomSheet.kt
theme
ui
com.software.todo (androidTest)
com.software.todo (test)
java (generated)
res
res (generated)
Gradle Scripts
build.gradle.kts (Project: To-Do_App)
build.gradle.kts (Module :app)
proguard-rules.pro (ProGuard Rules for ":app")
gradle.properties (Project Properties)
gradle-wrapper.properties (Gradle Version)
libs.versions.toml (Version Catalog)
local.properties (SDK Location)
settings.gradle.kts (Project Settings)
ModalBottomSheet.kt
fun TaskModalBottomSheet(
) { // Por padrão, o conteúdo está dentro de um 'Column'
var description by remember { mutableStateOf(value = task.description) } // remember significa que quando o
// compose recriar a tela quando um
// 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
// compose recriar a tela quando um
// 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 ocular 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
)
}
}
app > src > main > java > com > software > todo > presentation > sheet > ModalBottomSheet.kt > TaskModalBottomSheet
86:6 LF UTF-8 To-Do App Dark 4 spaces

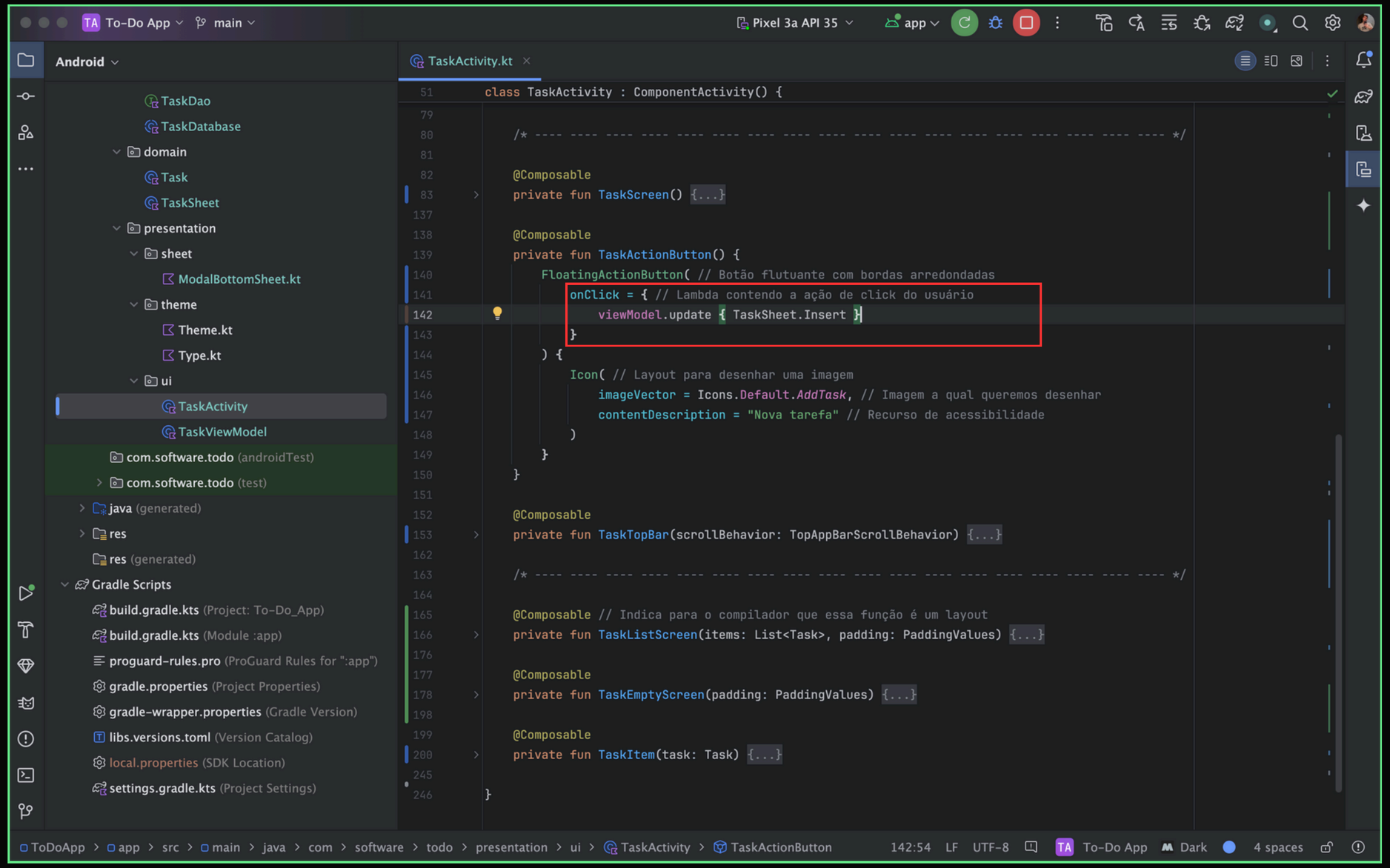

18:23
Título
Mercadorias
Descrição
CPU, GPU, APU
Feito
Inserir
ModalBottomSheet
OutlineTextField
OutlineTextField
Row
Button
Text
CheckBox
Text
Row

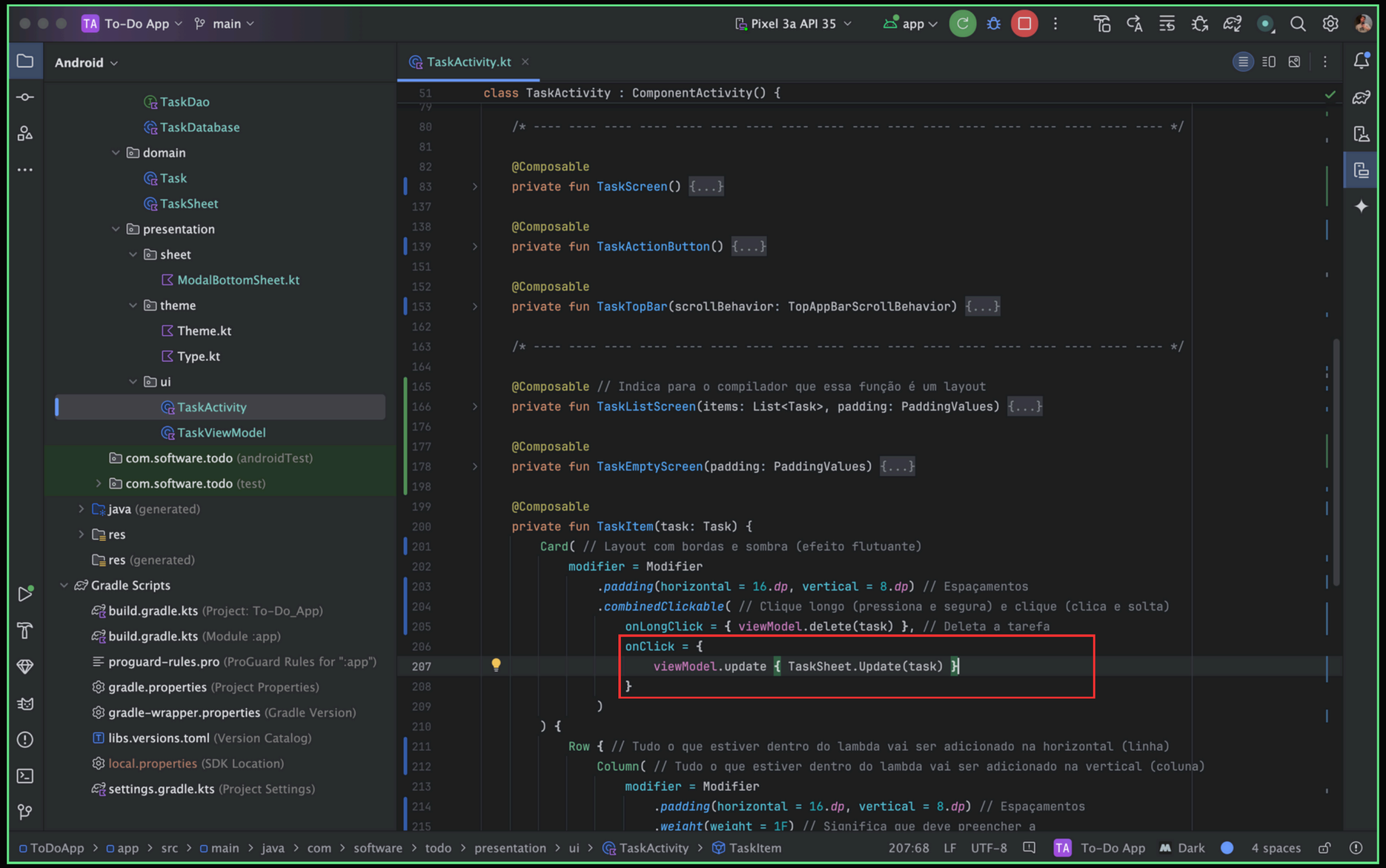

To-Do App
main
Pixel 3a API 35
app
Android
TaskDao
TaskDatabase
domain
Task
TaskSheet
presentation
sheet
ModalBottomSheet.kt
theme
Theme.kt
Type.kt
ui
TaskActivity
TaskViewModel
com.software.todo (androidTest)
com.software.todo (test)
java (generated)
res
res (generated)
Gradle Scripts
build.gradle.kts (Project: To-Do_App)
build.gradle.kts (Module :app)
proguard-rules.pro (ProGuard Rules for ":app")
gradle.properties (Project Properties)
gradle-wrapper.properties (Gradle Version)
libs.versions.toml (Version Catalog)
local.properties (SDK Location)
settings.gradle.kts (Project Settings)
TaskActivity.kt
class TaskActivity : ComponentActivity() {
@Composable
private fun TaskScreen() {
val scrollBehavior = TopAppBarDefaults.pinnedScrollBehavior() // Observa sempre que o
// usuário rola a lista de tarefas
Scaffold(// Scaffold tem a estrutura base para criarmos o nosso layout
modifier = Modifier.nestedScroll(connection = scrollBehavior.nestedScrollConnection), // Add o observer
floatingActionButton = { TaskActionButton() }, // Lambda para incluir o botão flutuante
topBar = { TaskTopBar(scrollBehavior) } // Lambda para incluir a barra de ações
) { paddingValues ->
val items by viewModel.tasks.collectAsState() // Busca as tarefas cadastradas no bd
if (
items.isNotEmpty() // Se a lista não estiver vazia, queremos mostra as tarefas
) {
TaskListScreen(items, paddingValues) // Lista de tarefas
} else { // Caso contrário
TaskEmptyScreen(paddingValues) // Mensagem informativa de que não há nenhuma tarefa
}
}
}
@Composable
private fun TaskActionButton() {...}
@Composable
private fun TaskTopBar(scrollBehavior: TopAppBarScrollBehavior) {...}
/* ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- */
@Composable // Indica para o compilador que essa função é um layout
private fun TaskListScreen(items: List<Task>, padding: PaddingValues) {...}
@Composable
ToDoApp › app › src › main › java › com › software › todo › presentation › ui › TaskActivity › TaskScreen
100:13 LF UTF-8 To-Do App Dark 4 spaces


To-Do App
main
Pixel 3a API 35
app
Android
TaskActivity.kt
TaskDao
TaskDatabase
domain
Task
TaskSheet
presentation
sheet
ModalBottomSheet.kt
theme
Theme.kt
Type.kt
ui
TaskActivity
TaskViewModel
com.software.todo (androidTest)
com.software.todo (test)
java (generated)
res
res (generated)
Gradle Scripts
build.gradle.kts (Project: To-Do_App)
build.gradle.kts (Module :app)
proguard-rules.pro (ProGuard Rules for ":app")
gradle.properties (Project Properties)
gradle-wrapper.properties (Gradle Version)
libs.versions.toml (Version Catalog)
local.properties (SDK Location)
settings.gradle.kts (Project Settings)
class TaskActivity : ComponentActivity() {
private fun TaskScreen() {
Scaffold(// Scaffold tem a estrutura base para criarmos o nosso layout
modifier = Modifier.nestedScroll(connection = scrollBehavior.nestedScrollConnection), // Add o observer
floatingActionButton = { TaskActionButton() }, // Lambda para incluir o botão flutuante
topBar = { TaskTopBar(scrollBehavior) } // Lambda para incluir a barra de ações
) { paddingValues ->
val items by viewModel.tasks.collectAsState() // Busca as tarefas cadastradas no bd
val state by viewModel.state.collectAsState() // Verifica o status atual do bottom sheet
if (
items.isNotEmpty() // Se a lista não estiver vazia, queremos mostra as tarefas
) {
TaskListScreen(items, paddingValues) // Lista de tarefas
} else { // Caso contrário
TaskEmptyScreen(paddingValues) // Mensagem informativa de que não há nenhuma tarefa
}
// When em kotlin é igual o IF, IF ELSE
// OBS: Funciona apenas com classes seladas e enums
when (state) { // Quando o 'state' for:
is TaskSheet.Insert -> { // Insert --> Execute esse trecho de código
}
is TaskSheet.Update -> { // Update --> Execute esse trecho de código
}
is TaskSheet.Closed -> { // Closed --> Execute esse trecho de código
// Não faça nada no status de fechado
}
}
}
}
@Composable
ToDoApp › app › src › main › java › com › software › todo › presentation › ui › TaskActivity › TaskScreen
119:6 LF UTF-8 To-Do App Dark 4 spaces


To-Do App
main
Pixel 3a API 35
app
Android
TaskDao
TaskDatabase
domain
Task
TaskSheet
presentation
sheet
ModalBottomSheet.kt
theme
Theme.kt
Type.kt
ui
TaskActivity
TaskViewModel
com.software.todo (androidTest)
com.software.todo (test)
java (generated)
res
res (generated)
Gradle Scripts
build.gradle.kts (Project: To-Do_App)
build.gradle.kts (Module :app)
proguard-rules.pro (ProGuard Rules for ":app")
gradle.properties (Project Properties)
gradle-wrapper.properties (Gradle Version)
libs.versions.toml (Version Catalog)
local.properties (SDK Location)
settings.gradle.kts (Project Settings)
TaskActivity.kt
class TaskActivity : ComponentActivity() {
private fun TaskScreen() {
) { paddingValues ->
// When em kotlin é igual o IF, IF ELSE
// OBS: Funciona apenas com classes seladas e enums
when (state) { // Quando o 'state' for:
is TaskSheet.Insert -> { // Insert --> Execute esse trecho de código
TaskModalBottomSheet(text = "Inserir",
onDismissRequest = {
},
onConfirm = { task ->
}
)
}
is TaskSheet.Update -> { // Update --> Execute esse trecho de código
val update = state as TaskSheet.Update
TaskModalBottomSheet(text = "Editar",
onDismissRequest = {
},
onConfirm = { task ->
},
task = update.task
)
}
is TaskSheet.Closed -> { // Closed --> Execute esse trecho de código
// Não faça nada no status de fechado
}
}
}
}
ToDoApp › app › src › main › java › com › software › todo › presentation › ui › TaskActivity › TaskScreen
130:1 LF UTF-8 To-Do App Dark 4 spaces


To-Do App
main
Pixel 3a API 35
app
Android
TaskDao
TaskDatabase
domain
Task
TaskSheet
presentation
sheet
ModalBottomSheet.kt
theme
Theme.kt
Type.kt
ui
TaskActivity
TaskViewModel
com.software.todo (androidTest)
com.software.todo (test)
java (generated)
res
res (generated)
Gradle Scripts
build.gradle.kts (Project: To-Do_App)
build.gradle.kts (Module :app)
proguard-rules.pro (ProGuard Rules for ":app")
gradle.properties (Project Properties)
gradle-wrapper.properties (Gradle Version)
libs.versions.toml (Version Catalog)
local.properties (SDK Location)
settings.gradle.kts (Project Settings)
TaskActivity.kt
class TaskActivity : ComponentActivity() {
private fun TaskScreen() {
) { paddingValues ->
// When em kotlin é igual o IF, IF ELSE
// OBS: Funciona apenas com classes seladas e enums
when (state) { // Quando o 'state' for:
is TaskSheet.Insert -> { // Insert --> Execute esse trecho de código
TaskModalBottomSheet(text = "Inserir",
onDismissRequest = {
viewModel.update { TaskSheet.Closed }
},
onConfirm = { task ->
viewModel.insert(task)
}
)
}
is TaskSheet.Update -> { // Update --> Execute esse trecho de código
val update = state as TaskSheet.Update
TaskModalBottomSheet(text = "Editar",
onDismissRequest = {
viewModel.update { TaskSheet.Closed }
},
onConfirm = { task ->
viewModel.update(task)
},
task = update.task
)
}
is TaskSheet.Closed -> { // Closed --> Execute esse trecho de código
// Não faça nada no status de fechado
}
}
}
}
ToDoApp > app > src > main > java > com > software > todo > presentation > ui > TaskActivity > TaskScreen
130:1 LF UTF-8 To-Do App Dark 4 spaces


To-Do App
main
Pixel 3a API 35
app
Android
TaskDao
TaskDatabase
domain
Task
TaskSheet
presentation
sheet
ModalBottomSheet.kt
theme
Theme.kt
Type.kt
ui
TaskActivity
TaskViewModel
com.software.todo (androidTest)
com.software.todo (test)
java (generated)
res
res (generated)
Gradle Scripts
build.gradle.kts (Project: To-Do_App)
build.gradle.kts (Module :app)
proguard-rules.pro (ProGuard Rules for ":app")
gradle.properties (Project Properties)
gradle-wrapper.properties (Gradle Version)
libs.versions.toml (Version Catalog)
local.properties (SDK Location)
settings.gradle.kts (Project Settings)
TaskActivity.kt
class TaskActivity : ComponentActivity() {
/* ---- */
@Composable
private fun TaskScreen() {...}
@Composable
private fun TaskActionButton() {
FloatingActionButton(// Botão flutuante com bordas arredondadas
onClick = { // Lambda contendo a ação de click do usuário
viewModel.update { TaskSheet.Insert }
}
) {
Icon(// Layout para desenhar uma imagem
imageVector = Icons.Default.AddTask, // Imagem a qual queremos desenhar
contentDescription = "Nova tarefa" // Recurso de acessibilidade
)
}
}
@Composable
private fun TaskTopBar(scrollBehavior: TopAppBarScrollBehavior) {...}
/* ---- */
@Composable // Indica para o compilador que essa função é um layout
private fun TaskListScreen(items: List<Task>, padding: PaddingValues) {...}
@Composable
private fun TaskEmptyScreen(padding: PaddingValues) {...}
@Composable
private fun TaskItem(task: Task) {...}
}
ToDoApp > app > src > main > java > com > software > todo > presentation > ui > TaskActivity > TaskActionButton
142:54
LF
UTF-8
To-Do App
Dark
4 spaces


To-Do App
main
Pixel 3a API 35
app
Android
TaskActivity.kt
TaskDao
TaskDatabase
domain
Task
TaskSheet
presentation
sheet
ModalBottomSheet.kt
theme
Theme.kt
Type.kt
ui
TaskActivity
TaskViewModel
com.software.todo (androidTest)
com.software.todo (test)
java (generated)
res
res (generated)
Gradle Scripts
build.gradle.kts (Project: To-Do_App)
build.gradle.kts (Module :app)
proguard-rules.pro (ProGuard Rules for ":app")
gradle.properties (Project Properties)
gradle-wrapper.properties (Gradle Version)
libs.versions.toml (Version Catalog)
local.properties (SDK Location)
settings.gradle.kts (Project Settings)
class TaskActivity : ComponentActivity() {
/* ---- */
@Composable
private fun TaskScreen() {...}
@Composable
private fun TaskActionButton() {...}
@Composable
private fun TaskTopBar(scrollBehavior: TopAppBarScrollBehavior) {...}
/* ---- */
@Composable // Indica para o compilador que essa função é um layout
private fun TaskListScreen(items: List<Task>, padding: PaddingValues) {...}
@Composable
private fun TaskEmptyScreen(padding: PaddingValues) {...}
@Composable
private fun TaskItem(task: Task) {
Card(// Layout com bordas e sombra (efeito flutuante)
modifier = Modifier
.padding(horizontal = 16.dp, vertical = 8.dp) // Espaçamentos
.combinedClickable(// Clique longo (pressiona e segura) e clique (clica e solta)
onLongClick = { viewModel.delete(task) }, // Deleta a tarefa
onClick = {
viewModel.update { TaskSheet.Update(task) }
}
)
) {
Row { // Tudo o que estiver dentro do lambda vai ser adicionado na horizontal (linha)
Column(// Tudo o que estiver dentro do lambda vai ser adicionado na vertical (coluna)
modifier = Modifier
.padding(horizontal = 16.dp, vertical = 8.dp) // Espaçamentos
.weight(weight = 1F) // Significa que deve preencher a
ToDoApp > app > src > main > java > com > software > todo > presentation > ui > TaskActivity > TaskItem
207:68
LF
UTF-8
To-Do App
Dark
4 spaces


18:59
Tarefas
Não há nada por aqui!


18:58
Tarefas
BR
Não há nada por aqui!
Título
Mercadorias
Descrição
CPU, GPU, APU
Feito
Inserir


18:59
Tarefas
Mercadorias
CPU, GPU, APU

Obrigado!

Baixe o projeto completo em:
https://github.com/pedrindenard/task-app