Para saber todas as permissoes podemos acessar o link

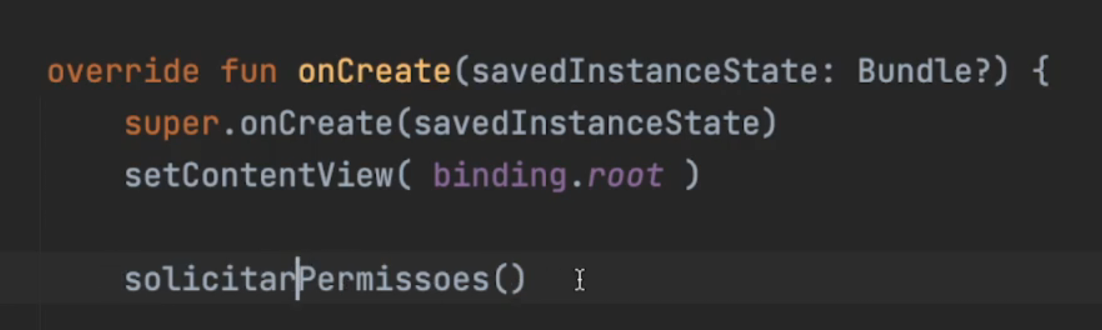
https://developer.android.com/reference/android/Manifest.permission

E todas que estiverem escrito dangerous = perigosas

Ai devemos solicitar do usuário no tempo de execução, na hora que o aplicativo estiver rodando.

# Mão na massa

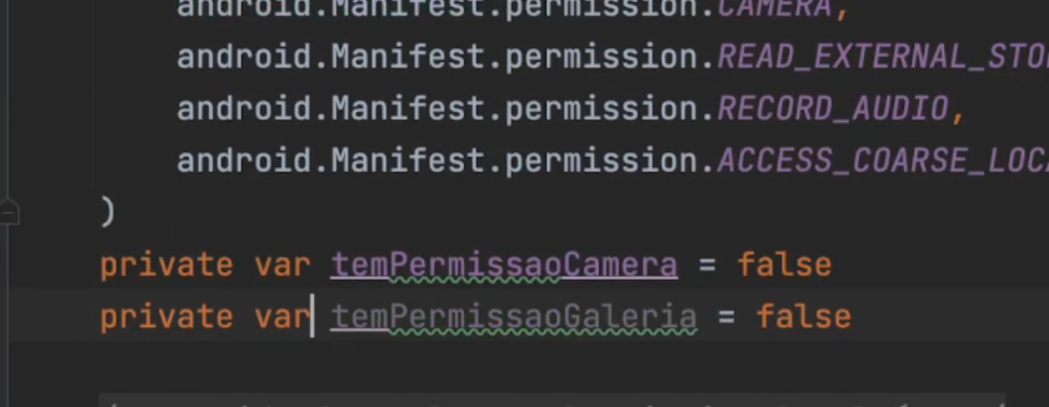
Primeiro vamos criar o método solicitarPermissoes()



private fun solicitarPermissoes() {  
  
  
 val gerenciadorDePermissoes = registerForActivityResult(  
 /\* Vamos agora definir um contrato aqui dentro . Requisitar Multiplas Permissao \*/  
 ActivityResultContracts.RequestMultiplePermissions()  
 ) **{** permissoes: Map<String, Boolean> **->** /\* Aqui essa funçao lambda vai retornar um Map<String> e um boolean  
 ex: camera - true ||||| Se voce estiver pedindo a permisao da camera OU falso se a permisao for negada..  
 \*/  
 // Diferente do outro metodo e que nao vamos precisar de um metodo para capturar o retorno das permissoes, podemos fazer aqui dentro mesmo  
 // Essa funçao lambda que criamos aqui vai servi exatamente para isso.  
 // Aqui dentro vamos capturar as permissoes que foram dadas  
 Log.i("novas\_permissoes", "Permissoes: $permissoes")  
 **}** // executando as permissoes  
 // Lembrar que as permissoes manda entrar os dados tipo Array, mas nosso permissoes e do tipo List  
 // Por isso estamos convertendo para Array com o toTypeArray  
 gerenciadorDePermissoes.launch(permissoes.*toTypedArray*())  
  
  
  
}

# Maneira completa e refatorada

Criamos 2 atributos para as permissões e vamos solicitar elas



Já estamos deixando como falso...

Aqui esta tudo explicado, se o usuário não der permissão, vamos pedir novamente, e se ele continuar não dando permissão, vamos ocultar a câmera por exemplo

temPermissaoGaleria = ContextCompat.checkSelfPermission(  
 this, Manifest.permission.*CAMERA* // Aqui estamos pedindo uma permissao especifica.  
 /\* Aqui vamos testar com checkSelfPermission e vamos ver se existe a permissao para essa camera.  
 \* Se existir vamos ter a permissao como TRUE \*/  
 ) == PackageManager.*PERMISSION\_GRANTED* // Agora vamos testar, se NAO tiver permissao de camera, vamos adicionar ela dentro da lista  
 // Agora com essas permissoes controladas, vamos poder exibir ou esconder algo se nao tiver permissao  
 // Supondo que o usuario nao der permissao para a camera, então podemos esconder o seu botao.  
  
 if (!temPermissaoCamera) {  
 permissoesNegadas.add(Manifest.permission.*CAMERA*)  
 }  
  
 if (!temPermissaoGaleria) {  
 permissoesNegadas.add(Manifest.permission.*READ\_EXTERNAL\_STORAGE*)  
 }  
  
 // OU seja se nao tiver permissao, vamos adicionar na permissao negadas  
 // e aqui em baixo é onde vamos solicitar essas permissoes.  
  
  
 if (permissoesNegadas.*isNotEmpty*()) {  
  
 // Solicitar permissoes  
 val gerenciadorDePermissoes = registerForActivityResult(  
 /\* Vamos agora definir um contrato aqui dentro . Requisitar Multiplas Permissao \*/  
 ActivityResultContracts.RequestMultiplePermissions()  
 ) **{** permissoes: Map<String, Boolean> **->** /\* Aqui essa funçao lambda vai retornar um Map<String> e um boolean  
 ex: camera - true ||||| Se voce estiver pedindo a permisao da camera OU falso se a permisao for negada..  
 \*/  
 // Diferente do outro metodo e que nao vamos precisar de um metodo para capturar o retorno das permissoes, podemos fazer aqui dentro mesmo  
 // Essa funçao lambda que criamos aqui vai servi exatamente para isso.  
 // Aqui dentro vamos capturar as permissoes que foram dadas  
 Log.i("novas\_permissoes", "Permissoes: $permissoes")  
  
  
 // AQUI VAMOS PEGAR AS PERMISSOES QUE FORAM DADAS PELO USUARIO!  
 // SERIA IGUAL O onRequestPermissionsResult  
 // Sabemos que essas permissoes tem a chave que sao o nome delas e tem o valor T - F  
 // para acessar elas.  
 // O Indice dele é o proprio nome da permissao  
 // ex: android.permission.Camera  
 // Entao vamos acessar o indice dele  
 // Agora o temPermissaoCamera é uma variavel do tipo Boolean  
 // E com isso o resultado pode ser, verdadeiro, falso ou nulo  
 // Devemos retornar somente verdadeiro ou falso  
 // Se for verdadeiro retornamos, se for falso pegamos o proprio resultado dela, que seria falso  
 // Dessa forma que esta em baixo.  
 // Segue a baixo como seria, se retornar falso...  
 // Esse ?: significa: Se for null faça isso, e estamos fazendo oque? Retornando um false  
 // pq null = falso  
  
  
 temPermissaoCamera = permissoes[Manifest.permission.*CAMERA*] ?: temPermissaoCamera  
  
 temPermissaoGaleria = permissoes[Manifest.permission.*READ\_EXTERNAL\_STORAGE*] ?: false  
  
  
 **}** // executando as permissoes  
 // Lembrar que as permissoes manda entrar os dados tipo Array, mas nosso permissoes e do tipo List  
 // Por isso estamos convertendo para Array com o toTypeArray  
 gerenciadorDePermissoes.launch(permissoesNegadas.*toTypedArray*()) // Agora vamos pedir as permissoes que estao negadas.  
 }  
  
  
}

Ocultando um item se o usuário não der a permissao

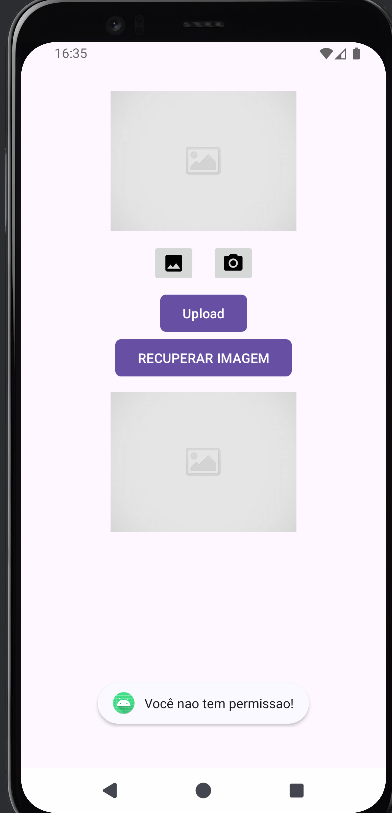
# Verificando se o usuário deu permissão e desativando caso não deu

Pedimos de forma individual a permissão da câmera, se o usuário não deu, vamos desativar o ícone

## Usuario com permissão OK



## Usuario não deu permissão



## Codigo

binding.btnCamera.setOnClickListener **{** if (temPermissaoCamera) {  
 /\* Aqui como usamos o StartActivityForResult precisamos passar uma intent  
 \* Ja na galeria informamos o tipo de dados que queremos pegar.  
 \*  
 \* MediaStore = vai acessar varios recursos que podemos usar, nesse caso aqui vamos acessar uma ação e essa ação vai ser capturar imagem...\*/  
 val intent = Intent(MediaStore.*ACTION\_IMAGE\_CAPTURE*)  
 abrirCamera.launch(intent)  
 }else {  
 Toast.makeText(this, "Você nao tem permissao!", Toast.*LENGTH\_SHORT*).show()  
 }  
  
  
**}**

Lembrar que toda vez que você executar o app ele vai pedir novamente as permissões para o usuário.

Outra maneira seria usando o onRequestPermissionsResult

Mas devemos estudar mais ela...