Sintaxe de comandos em Android com Kotlin

Funções da Activity

```
<objeto View> = findViewById<Tipo>( id : Int )
setContentView( view: View )
setContentView( idLayout : Int )
runOnUIThread { … }
```

Objetos do tipo View

```
val <campo Texto> : EditText
val <objeto botao> : Button
val <label> : TextView
```

Acesso as propriedades de Texto e registro de função em objeto clicável

```
<objeto Editable> : Editable =
      <objeto do tipo EditText>.text
<texto> : String  = <objeto Editable>.toString()
<Objeto clicável>.setOnClickListener {
  ... codigo ...
}
```

Kotlin Básico

```
class <nome> : <superclasse>() { }
for (<nome Obj> in <lista de Tipo do Obejto>) { }
val <contem> : boolean = <texto>.contains( <outro texto> )

[override] fun <nome funcao> ( [ val <nomeParametro1> : <Tipo
Parametro1>, ...  ,val <nomeParametroN> : <Tipo ParametroN>]) : <Tipo
de Retorno> {
  … codigo …
}
data class <nome da classe>(<var ou val> <nome do atributo1> :
<tipo>[=<valor padrão>] [, ... <var ou val> <nome do atributoN> :
<tipo>[=<valor padrão>]]) { [<corpo>] }

val texto : String = String.valueOf( <double> )
val numero : Double = "10.0".toDouble()
Log.i(<tag>, <texto> );
val lista = ArrayList<Tipo>()
public get<Nome>() : <Tipo>{
   return this.<nome>
}
public set<Nome>( val <valor> : <Tipo>) : Unit {
   this.<nome> = <valor>
}
```

Acesso ao Backend

```
val <objeto OkHttpClient> = OkHttpClient()
<objeto OkHttpClient>.newCall( <objeto
Request> ).enqueue( <objeto Response> )
```

Objeto Request

```
val <objeto Request> = Request.Builder()
.post( <objeto RequestBody> )
.get()
.url( <caminho URL>).build()
```

Objeto Response

```
val <objeto Response> = object : Callback {
  override fun onResponse(call : Call, response:
Response) { … }
  override fun onFailure(call : Call, e :
IOException) { … }
}
val <objeto RequestBody> =
<texto>.toRequestBody( <objeto MediaType> )
val <objeto MediaType> =
<texto>.toMediaType()
```

GSon

```
val <texto JSON> = <objeto Gson>.toJson( <objeto> )
val <objeto> : <tipo> = <objeto Gson>.fromJson( <texto JSON>,
<tipo::class.java>)
val <objeto Gson> = Gson()
val <tipo> = object : TypeToken<HashMap<String, Object>>() {}.type
```

LocalDate

```
val <objeto localDate> = LocalDate.now() // Data Atual
val <objeto DateTimeFormatter> =
DateTimeFormatter.ofPattern("dd/MM/yyyy")
val <texto> = <objeto localDate>.format(<objeto DateTimeFormatter>)
val <objeto localDate> = LocalDate.parse(<texto>, <objeto
DateTimeFormatter>)
```