JAVA PARA KOTLIN

PERSPECTIVA DE UM DESENVOLVEDOR

Filipe Nunes

QUEM SOU

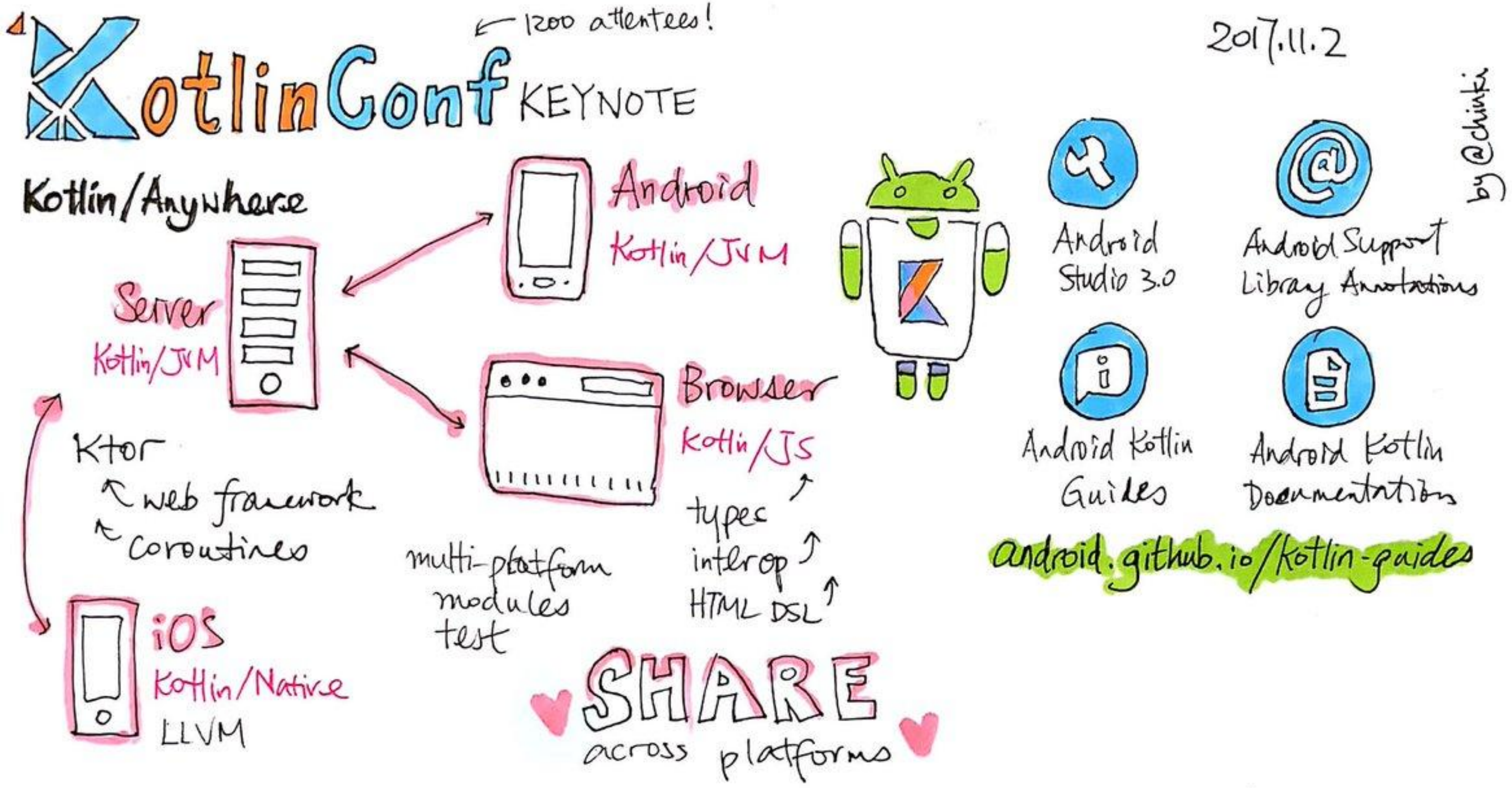
Filipe Nunes

Mobile Specialist.

Organizador do GDG Porto Alegre e Leader Jam da Google.

Evangelista do open source, envolvido em projetos como FISL e HacktoberFest, Google IO Extended, Congressos de TI, TDCs e DevFests.

Participante de projetos e empresas como IBM, SAP, Warren, Grupo RBS dentre outras.


KotlinConf KEYNOTE
← 1200 attentees!
2017.11.2
by @chiuki
Kotlin/Anywhere
Server
Kotlin/JVM
Android
Kotlin/JVM
Browser
Kotlin/JS
Ktor
↖ web framework
↖ coroutines
iOS
Kotlin/Native
LLVM
multi-platform
modules
test
types
interop
HTML DSL
SHARE
across platforms
Android Studio 3.0
Android Support Library Annotations
Android Kotlin Guides
Android Kotlin Documentation
android.github.io/kotlin-guides

PECULIARIDADES

- Null Safety
- Open Source
- Trabalha totalmente com Java
- Menos código
- Linguagem primária para Android
- Inferência de tipo
- Não existe tipo primitivo
- Toda função é final
- Podemos trabalhar com Java e Kotlin

Properties / Function / Strings Templates

```
val a: Int = 1  // immediate assignment
val b = 2   // `Int` type is inferred
val c: Int  // Type required when no initializer is provided
c = 3       // deferred assignment

fun sum(a: Int, b: Int) {
   return a + b
}

fun sum(a: Int, b: Int) = a + b

var a = 1
// simple name in template:
val s1 = "a is $a"

a = 2
// arbitrary expression in template:
val s2 = "${s1.replace("is", "was")}, but now is $a

fun printSum(a: Int, b: Int) {
   println("sum of $a and $b is ${a + b}")
}
```

VAMOS ENTENDER UM POUCO MAIS

Class / Named Arguments

```
class Player(
   val name: String,
   val nickName: String,
   var level: Int,
   var xp: Double,
   var vip: Boolean
) {
         var children: MutableList<Player> = mutableListOf()

         constructor(name: String, parent: Player) : this(
            name = name,
            nickName = "",
            level = 0,
            xp = 0.0,
            vip = false
         ) {
            parent.children.add(this)
         }

         fun sumXp(xpUpdate: Double) {
            xp += xpUpdate
         }
}
```

VAMOS ENTENDER UM POUCO MAIS

Data Class

```kotlin
data class Player2(
  val name: String = "",
  val nickName: String = "",
  var xp: Double = 0.0,
  var level: Int = 0.0,
  var vip: Boolean = false
) {

  val fullName: String
     get() = "$name $level"

}

fun anonymousPlayer() : Player2 {
  val ksdrof = Player2(
     nickName = "ksdrof",
     name = "Filipe"
  )

  return ksdrof.copy(level = 2)
}
```

VAMOS ENTENDER UM POUCO MAIS

Enum

```
enum class Race {
   ELF, DWARF, HUMAN, ORC
}

enum class Race(val ability: String) {
   ELF("DEX"),
   DWARF("HEF"),
   HUMAN("GEN"),
   ORC("STR")
}
```

```
enum class Race {
   ELF {
       override fun frail(): Race {
           return ORC
       }
   },
   DWARF {
       override fun frail(): Race {
           return ELF
       }
   },
   HUMAN {
       override fun frail() = DWARF
   },
   ORC {
       override fun frail() = HUMAN
   };

   abstract fun frail(): Race

}
```

VAMOS ENTENDER UM POUCO MAIS

Interface

```
interface RaceAttack {
  val nameAttack: String

  fun modifier(mod: Any) : Any
}


class Race(
override val nameAttack: String) :RaceAttack
{
  override fun modifier(mod: Any): Any {
     return "$nameAttack $mod"
  }
}
```

```
enum class Race : RaceAttack {
  ELF {
     override val nameAttack: String
        get() = "Arrow Flame"

     override fun modifier(mod: Any): Any {
         return "nameAttack $mod"
     }
  }, DWARF {
     override val nameAttack: String
        get() = "Double Axe Flame"

     override fun modifier(mod: Any): Any {
        return mod as Int + 5
     }
  }, HUMAN {
     override val nameAttack: String
        get() = "Magic Long Sword"

     override fun modifier(mod: Any) = nameAttack
  }, ORC {
     override val nameAttack: String
        get() = "Fury"

     override fun modifier(mod: Any) = nameAttack
  }
}
```

VAMOS ENTENDER UM POUCO MAIS

Sealed Class / Smart Cast / Open

```kotlin
sealed class Boss
data class Necromancer(val name: String, val str: Double) : Boss()
data class Dragon(val babyDragon: Boss, val name: String, val str: Double) : Boss()


fun verifyBossYourRace(boss: Boss) {
   when (boss) {
       is Necromancer -> {
           sendToCave()
       }
       is Dragon -> {
           sendToDragonSlayerCity()
       }
   }
}


open class subBoss( val summon : String)
sealed class Boss : subBoss("Kachucia")

fun verifyBossYourRace(boss: Boss) {
   boss.summon
}
```

VAMOS ENTENDER UM POUCO MAIS

Null Safety

```
val monsterName = "Cthulhu"
monsterName = null
```

```
var monsterName : String? = "Cthulhu"
monsterName = null
```

```
if (monsterName.isNullOrBlank()) monsterName?.length else -1

println(monsterName?.length)

monster?.head?.race?frail()

ctuchlu as? Monster
```

VAMOS ENTENDER UM POUCO MAIS

elvis operator ?:

```
monsterName?.length ?: -1
```

Scope Functions Context

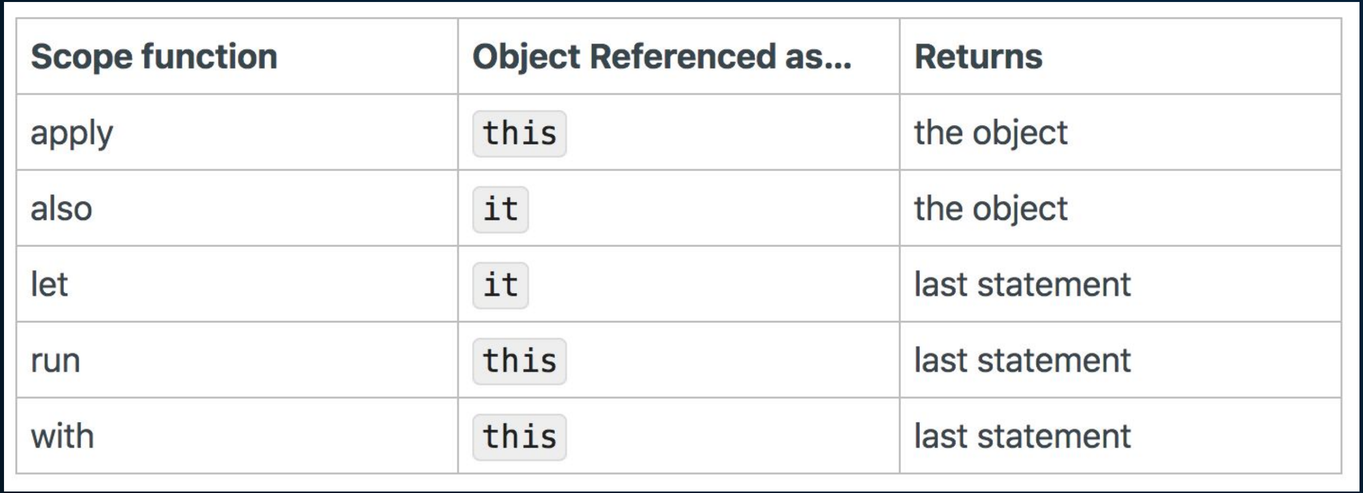
```
val line = PoetryGenerator.obtain().run {
   style = "Emily Dickinson"
   style += "Lucille Clifton"
   lines = 1
   generate()
}

val paint = Paint().apply {
   color = Color.MAGENTA
   style = Paint.Style.STROKE
   textSize = textHeadlinePx
}

inner class ViewHolder(parent: ViewGroup) :
RecyclerView.ViewHolder(parent.inflate(R.layout.item_delegate_article_container)) {

   fun bind(container: AreaContainerUiModel) = with(itemView) {
      recycler.adapter = AreaContainerAdapter(container.items, articleListener)
      recycler.layoutManager = GridLayoutManager(context, 2)
      recycler.isNestedScrollingEnabled = false
   }
}
```

VAMOS ENTENDER UM POUCO MAIS

Scope Functions Context

```
fun initSDK(context: Context, appId: AppId) {
   INSTANCE ?: synchronized(this) {
      INSTANCE ?: AdManager(context, appId).also { INSTANCE = it }
   }
}
val it: AdManager get() = it()

private fun it(): AdManager {
   INSTANCE?.let {
      return it
   }
   throw NullPointerException("AdManager must to be initialized!")
}
```

VAMOS ENTENDER UM POUCO MAIS



Scope function	Object Referenced as...	Returns
apply	`this`	the object
also	`it`	the object
let	`it`	last statement
run	`this`	last statement
with	`this`	last statement

Generics

```
object PreferenceHelper {
  fun defaultPrefs(context: Context): SharedPreferences =
      PreferenceManager.getDefaultSharedPreferences(context)
  fun customPrefs(context: Context, name: String): SharedPreferences =
      context.getSharedPreferences(name, Context.MODE_PRIVATE)

  inline fun SharedPreferences.edit(operation: (SharedPreferences.Editor) -> Unit) {
      val editor = this.edit()
      operation(editor)
      editor.apply()
  }

  fun SharedPreferences.setValue(key: String, value: Any?) {
      when (value) {
          is String? -> edit({ it.putString(key, value) })
          is Int -> edit({ it.putInt(key, value) })
          is Boolean -> edit({ it.putBoolean(key, value) })
          is Float -> edit({ it.putFloat(key, value) })
          is Long -> edit({ it.putLong(key, value) })
          else -> throw UnsupportedOperationException("Not yet implemented")
      }
  }
```

```
object PreferenceHelper {
  fun defaultPrefs(context: Context): SharedPreferences =
      PreferenceManager.getDefaultSharedPreferences(context)
  fun customPrefs(context: Context, name: String): SharedPreferences =
      context.getSharedPreferences(name, Context.MODE_PRIVATE)
  inline fun SharedPreferences.edit(operation: (SharedPreferences.Editor) -> Unit) {
      val editor = this.edit()
      operation(editor)
      editor.apply()
  }
  fun SharedPreferences.setValue(key: String, value: Any?) {
      when (value) {
          is String? -> edit({ it.putString(key, value) })
          is Int -> edit({ it.putInt(key, value) })
          is Boolean -> edit({ it.putBoolean(key, value) })
          is Float -> edit({ it.putFloat(key, value) })
          is Long -> edit({ it.putLong(key, value) })
          else -> throw UnsupportedOperationException("Not yet implemented")
      }
  }
  inline fun <reified T : Any> SharedPreferences.get(key: String, defaultValue: T? = null): T? {
      return when (T::class) {
          String::class -> getString(key, defaultValue as? String) as T?
          Int::class -> getInt(key, defaultValue as? Int ?: -1) as T?
          Boolean::class -> getBoolean(key, defaultValue as? Boolean ?: false) as T?
          Float::class -> getFloat(key, defaultValue as? Float ?: -1f) as T?
          Long::class -> getLong(key, defaultValue as? Long ?: -1) as T?
          else -> throw UnsupportedOperationException("Not yet implemented")
      }
  }
}
```

VAMOS ENTENDER UM POUCO MAIS

Higher-Order

```
fun <T> ArrayList<T>.filterOnCondition(condition: (T) -> Boolean): ArrayList<T> {
   val result = arrayListOf<T>()
   for (item in this) {
      if (condition(item)) {
         result.add(item)
      }
   }

   return result
}


fun isMultipleOf(number: Int, multipleOf: Int): Boolean {
   return number % multipleOf == 0
}

fun multiples() {
   var list = arrayListOf<Int>()
   for (number in 1..10) {
      list.add(number)
   }
   var resultList = list.filterOnCondition { isMultipleOf(it, 5) }
}
```

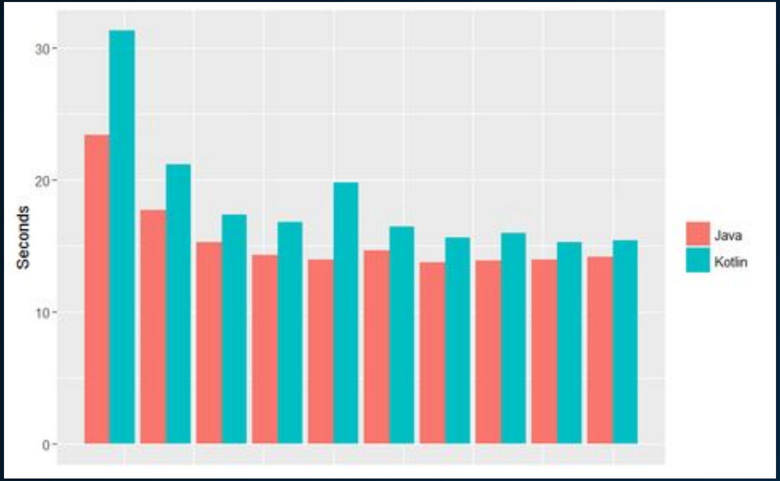
VAMOS ENTENDER UM POUCO MAIS

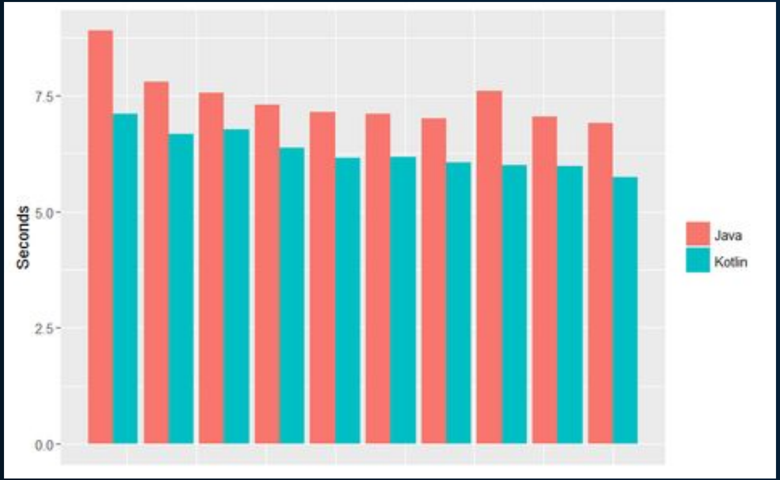
```
fun existCharacter() {
   var listOfStr = arrayListOf<String>()
   listOfStr.add("The Butcher")
   listOfStr.add("Belial")
   listOfStr.add("Azmodan")
   listOfStr.add("Diablo")

   var modifiedList = listOfStr.filterOnCondition { it.contains("e")
}
}
```

Extensions

```kotlin
fun String.appendCharactersPasswordForLoginLegacy() = this.padEnd(6, 'A').toUpperCase()

fun String.isValidEmail(): Boolean = this.isNotEmpty() &&
   Patterns.EMAIL_ADDRESS.matcher(this).matches()

fun EditText.toText(): String = Mask.replaceChars(this.text.toString())

fun EditText.addMask(mask: String) {
   this.addTextChangedListener(Mask.mask(mask, this))
}

fun Context.ToastKT(message: String) {
   Toast.makeText(this, message, Toast.LENGTH_LONG).show()
}

fun Context.validEmpty(layout: TextInputLayout, edit: TextInputEditText, error: Int): Boolean {
   layout.error = null
   if (edit.text.isNullOrBlank()) {
      layout.error = getString(error)
      return true
   }
   layout.error = null
   return false
}
fun Activity.urlLost() {
    NossaApplication.remoteConfig?.fetchAndActivate()?.addOnCompleteListener(this) {
       NossaApplication.remoteConfig?.getString(LoginActivity.URL_LOST_EMAIL)?.let { url ->
             openWebView(url.replace("/nossa/", "/" + NossaApplication.product.slug + "/"))
       }
   }
```

VAMOS ENTENDER UM POUCO MAIS





NO DIA A DIA

Intel Core i7–6700 running at 3.4 GHz, with 32GiB of DDR4 memory and a Samsung 850 Pro SSD. The source code was built with Gradle 2.14.1.

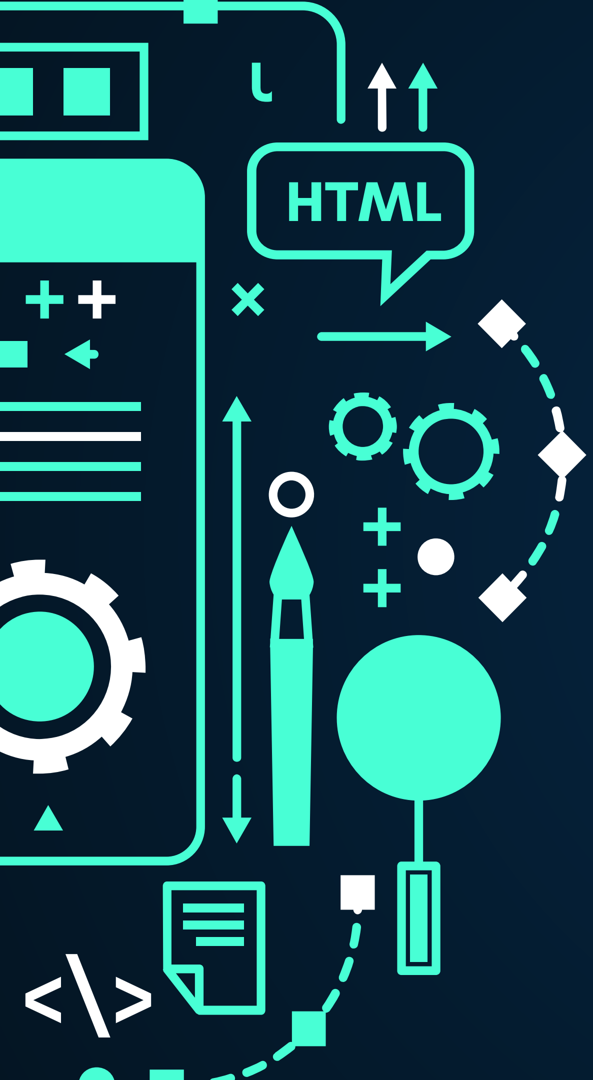
Before the transition, App Lock's Java codebase was 5,491 methods and 12,371 lines of code. After the rewrite, those numbers dropped down to was 4,987 methods and 8,564 lines of *Kotlin* code

https://github.com/ksdrof500/design-patterns-with-rpg

Coroutines

VAMOS ENTENDER UM POUCO MAIS

HTTPS://WWW.JETBRAINS.COM/PT-PT/LP/DEVECOSYSTEM-2019/KOTLIN/



THANKS!

filipenunes.developer@gmail.com

FilipeFNunes

@ksdrof500

bagu.al/1OH