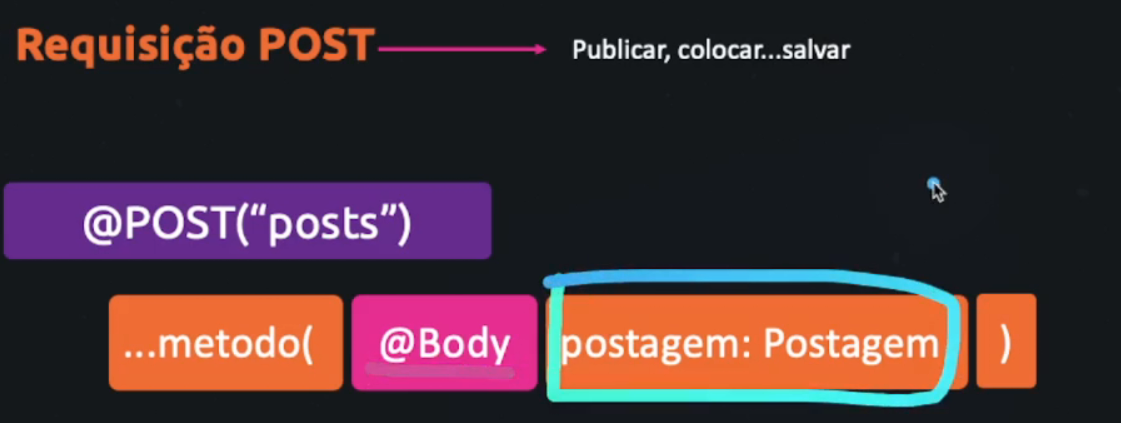
Imagina nosso aplicativo o usuario quer salvar um novo posto no banco de dados, para isso ele vai fazer uma requisição do tipo POST

Aqui vamos usar a anotação do tipo Post

A rota esta com o nome de post, mas poderia ser qualquer nome.



Vamos ter o método

Objeto postagem, podemos passar ele inteiro

Body indica o corpo do que vai ser salvo

# Criando nova interface para o usuário poder postar algo.

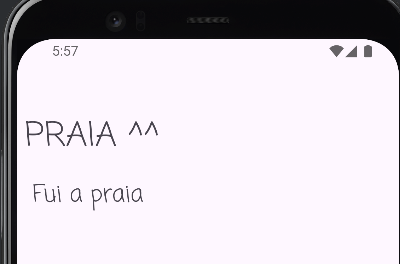
*/\*\* Quando o usuario clicar em publicar algo, vamos para uma nova activity para evitar sobrecarregar aqui \*/*binding.btnNovaPostagem.setOnClickListener **{** val intent = Intent(this, PublicarActivity::class.*java*)  
 startActivity(intent)  
**}**

# Criando requisição do tipo Post

@POST("posts/")  
/\* Body = o corpo do que vamos salvar  
Nesse caso estamos salvando uma postagem.  
 \*/  
suspend fun salvarPostagem(@Body postagem: Postagem) : Response<Postagem>

# Criando a postagem

class PublicarActivity : AppCompatActivity() {  
  
 private val binding by *lazy* **{** ActivityPublicarBinding.inflate(*layoutInflater*) **}** private val retrofit by *lazy* **{** RetrofitHelper.retrofit  
 **}** override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 *enableEdgeToEdge*()  
 setContentView(binding.*root*)  
 ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.*main*)) **{** v, insets **->** val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())  
 v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom)  
 insets  
 **}** *CoroutineScope*(Dispatchers.IO).*launch* **{** salvarPostagem()  
 **}** }  
  
 private suspend fun salvarPostagem() {  
 var retorno : Response<Postagem>? = null  
  
 /\* Aqui vamos passar os valores simulando que o usuario digitou para publicar  
 \* O id é gerado de forma automatica pela regra de negocio, então um padrao para ele sempre sera o -1 \*/  
  
 val novaPostagem = Postagem(  
 "Fui a praia",  
 -1,  
 "PRAIA ^^",  
 1090  
 )  
  
 try {  
 val postagemAPI = retrofit.create(PostagemAPI::class.*java*)  
 retorno = postagemAPI.salvarPostagem(novaPostagem)  
  
 }catch (e: Exception) {  
 e.printStackTrace()  
 Log.i("info\_postagem", "ERRO: Postagem não enviada.")  
 }  
  
 if (retorno != null && retorno.*isSuccessful*) {  
 val postagemCriada = retorno.body()  
 /\* Objetos retornados pela postagem \*/  
 val titulo = postagemCriada?.title  
 val descricao = postagemCriada?.descricao  
  
 /\* Adicionando itens na tela  
  
 \*/  
  
 withContext(Dispatchers.Main) **{** binding.textTitulo.*text* = titulo  
 binding.textPostagem.*text* = descricao  
 **}** }  
  
  
 }  
}



Outro detalhe, qual código vem da API? 201 = sucesso?

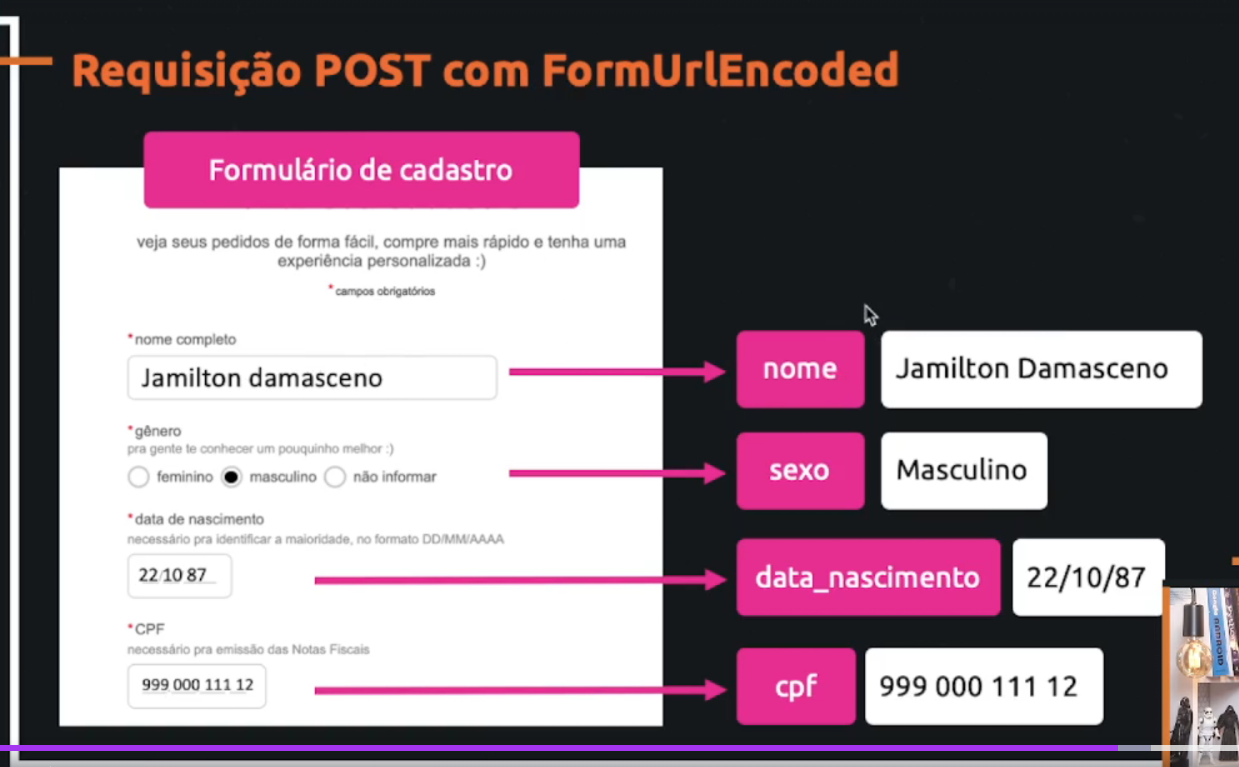
404 ? caminho não encontrado?

Para saber qual o código podemos utilizar o

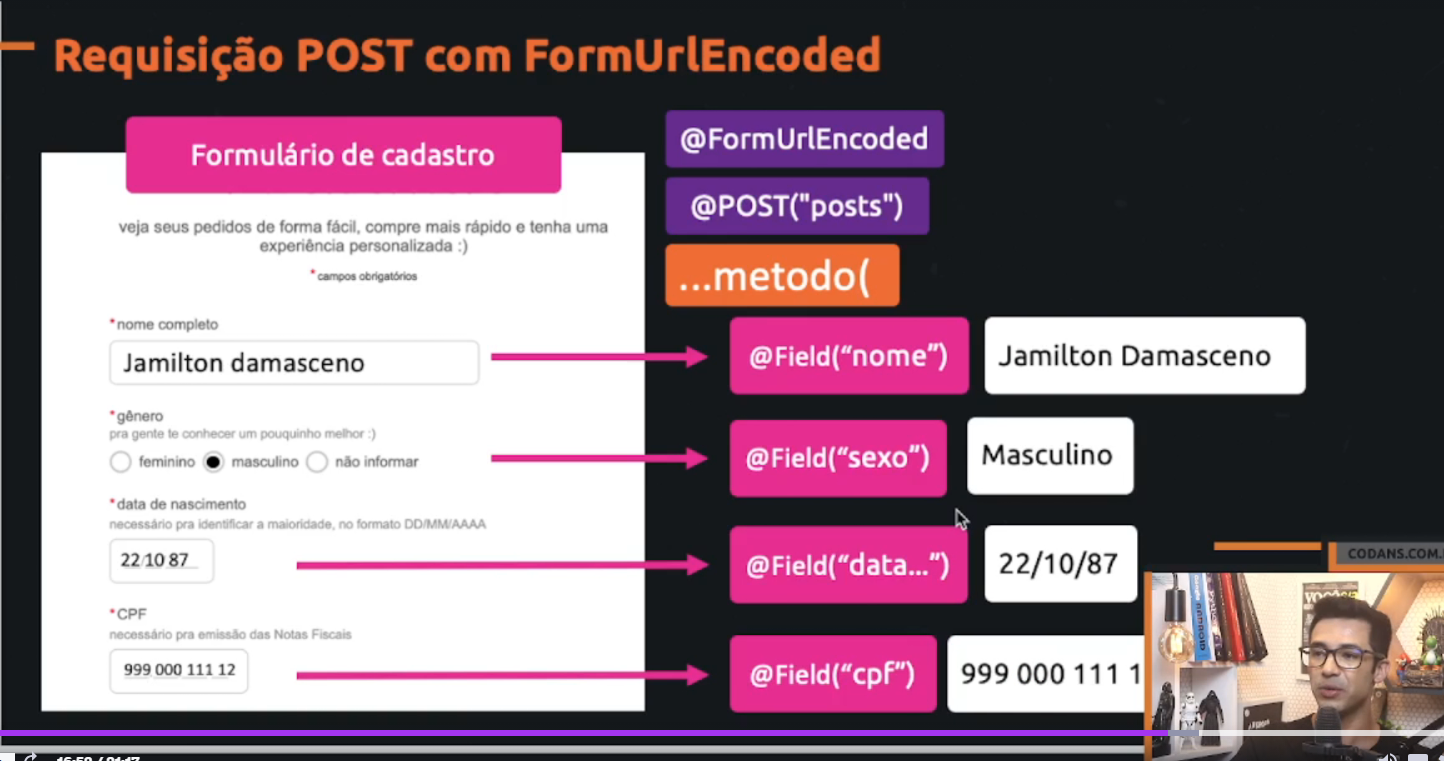
}else {  
 Log.i("info\_postagem", retorno?.code().*toString*())  
}

# Outra maneira de fazer usando o FormUrlEncoded

Quando preenchemos um formulário na internet ele usa essa estratégia



Aqui a gente vai fazer igual.



## Criando nosso método na interface

@FormUrlEncoded  
@POST("posts")  
suspend fun salvarPostagemPorFomulario(  
 /\* Precisamos utilizar os mesmos nomes que estao na api \*/  
 @Field("userID") userID : Int,  
 @Field("id") id : Int,  
 @Field("title") title : String,  
 @Field("body") body : String  
) {  
}

## Método para salvar a postagem

Precisamos passar os dados de forma separada

Mudou praticamente só isso, da ultima vez passávamos o corpo do objeto inteiro.

private suspend fun salvarPostagemPorFormulario() {  
 var retorno : Response<Postagem>? = null  
  
 /\* Aqui vamos passar os dados que queremos salvar \*/  
  
  
 val id = -1  
 val userId = 1090  
 val title = "Praia ^^"  
 val descricao = "Fui a praia"  
  
 try {  
 val postagemAPI = retrofit.create(PostagemAPI::class.*java*)  
 retorno = postagemAPI.salvarPostagemPorFomulario(  
 userId, id, title, descricao  
 )  
  
 }catch (e: Exception) {  
 e.printStackTrace()  
 Log.i("info\_postagem", "ERRO: Postagem não enviada.")  
 }  
  
 if (retorno != null && retorno.*isSuccessful*) {  
 val postagemCriada = retorno.body()  
 /\* Objetos retornados pela postagem \*/  
 val titulo = postagemCriada?.title  
 val descricao = postagemCriada?.descricao  
  
 /\* Adicionando itens na tela  
  
 \*/  
  
 withContext(Dispatchers.Main) **{** binding.textTitulo.*text* = titulo  
 binding.textPostagem.*text* = descricao  
 **}** }else {  
 Log.i("info\_postagem", retorno?.code().*toString*())  
 }  
  
  
 }  
}

Resumindo: vamos passar os itens de forma separada, vai simular como se tivéssemos preenchendo um formulário de internet.