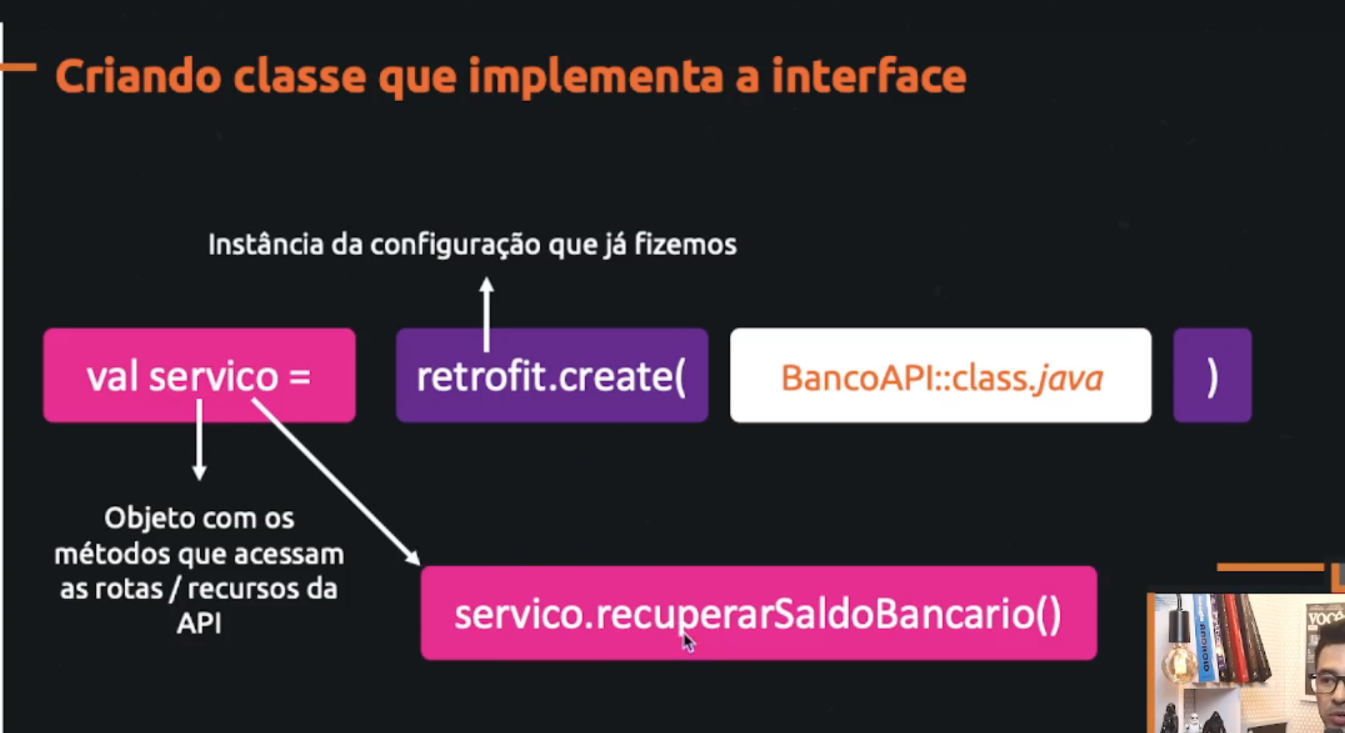
Resumo do que fizemos na aula anterior.

Dentro do método recuperarEndereco()

A Retrofit esta acessando a interface e por baixo dos panos criando uma classe para a gente.

Ela retorna essa classe para serviço

E com serviço podemos acessar os métodos dessa classe (métodos que esta dentro da interface)



Em resumo a Retrofit que vai criar a classe que implementa os métodos que vamos precisar utilizar, esses métodos estão dentro da INTERFACE

interface EnderecoAPI {  
 suspend fun recuperarEndereco()  
}

Aqui a interface/classe vai retornar alguma coisa, nesse caso vai retornar o endereço então vamos pegar esse endereço e por dentro da variável

private suspend fun recuperarEnderecoMetodo() {  
 */\*\* Explicando o codigo  
 \* O metodo create ele faz assim:  
 \* Ele recebe a interface e ele mesmo cria e retorna para voce o objeto  
 \* Sem voce precisar de uma classe para usar o metodo  
 \* ai esse objeto permite voce acessa o metodo recuperarEndereco  
 \*/* val enderecoAPI = retrofit.create( EnderecoAPI::class.*java* )  
 val retorno = enderecoAPI.recuperarEndereco() // metodo dentro da INTERFACE  
}

# Configurando as rotas

Agora dentro da interface, vamos colocar as rotas, já temos o endereço base configurado la dentro da classe

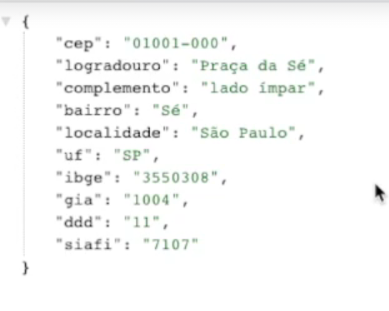
class RetrofitHelper {  
 // Queremos acessar os recursos sem precisar instanciar a classe  
 companion object {  
 val retrofit = Retrofit.Builder()  
 .baseUrl("https://viacep.com.br/")  
 .addConverterFactory(GsonConverterFactory.create()) // Conversor para JSON || XML  
 .build()  
 }  
}

Dentro da interface vamos apenas colocar as rotas

Tambem vamos precisar que o recuperarEndereco retorne algo

E para isso ele vai retornar uma Response = Resposta

Que vai ser nosso JSON



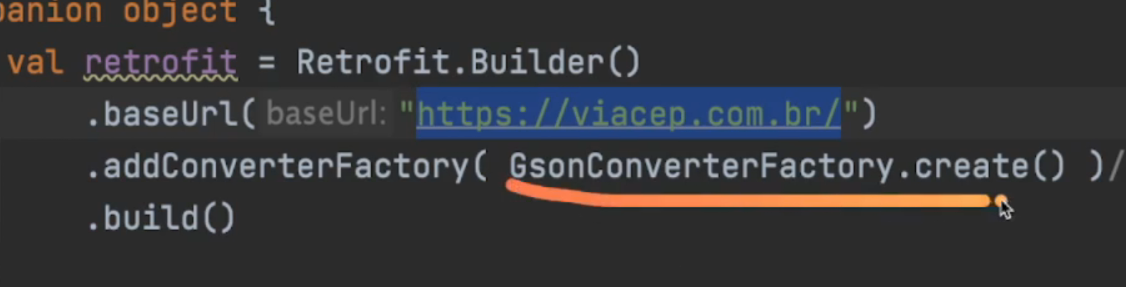
E com isso vamos associar esses dados a uma classe que vamos criar.

E ai o conversor vai pegar esses dados e vai converter em OBJETOS.

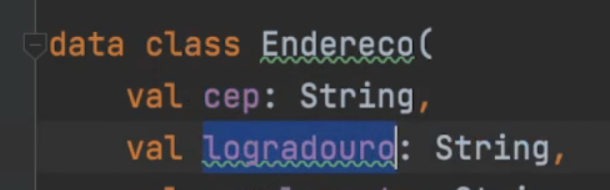
interface EnderecoAPI {  
  
 /\* Aqui vamos passar somente as rotas, ja temos o BASE URL definido  
 \* https://viacep.com.br/ + ws/01001000/json/  
 \*/  
 @GET("ws/01001000/json/")  
 suspend fun recuperarEndereco() : Response<Endereco>  
}

Endereco = aqui temos uma classe que ele vai retornar, mas oque vem do servidor é um objeto JSON.

Então isso precisa ser convertido e quem vai converter isso é nosso conversor

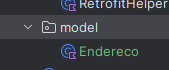


E assim ele vai associar os dados, o CEP que esta no JSON ele vai colocar dentro do atributo CEP Endereco



# Criando nossa classe para receber os dados

Criamos uma data class porque ela vai servi apenas para receber os dados.



package com.allephnogueira.configuracaoretrofit.model  
  
data class Endereco (  
 val cep : String,  
 val logradouro: String,  
 val complemento: String,  
 val bairro: String,  
 val localidade: String,  
 val uf: String,  
 val ibge: String,  
 val gia: String,  
 val ddd: String,  
 val siafi: String  
)

# Usando o método para recuperar o endereço

Usamos um try e catch porque é um dado que estamos tentando e não temos certeza se vai esta tudo certo com o servidor que vai servir os dados para a gente

private suspend fun recuperarEnderecoMetodo() {  
  
  
 try {  
 */\*\* Explicando o codigo  
 \* O metodo create ele faz assim:  
 \* Ele recebe a interface e ele mesmo cria e retorna para voce o objeto  
 \* Sem voce precisar de uma classe para usar o metodo  
 \* ai esse objeto permite voce acessa o metodo recuperarEndereco  
 \*/* val enderecoAPI = retrofit.create( EnderecoAPI::class.*java* )  
 val retorno = enderecoAPI.recuperarEndereco() // metodo dentro da INTERFACE  
 }catch (e: Exception){  
 e.printStackTrace()  
 Log.i("info\_endereco", "Erro ao recuperar")  
 }  
  
}

Aqui nesse código temos a instancia da retrofit que já tínhamos configurado la em cima como atributo

O método create vai criar um objeto e ai passamos oque queremos que ele crie, no nosso caso estamos usando uma interface

Que já esta configurado a nossa rota e nosso método

E por baixo dos panos ele vai criar uma classe com esse método para a gente poder utilizar

E ai o retorno vai receber essa “classe” com esse método que esta configurado para receber uma response<Endereco>

\*\* Isso significa que: na response os dados vai ser convertido pelo conversor de JSON para Objeto \*\*\*

E ai sim podemos utilizar esse objeto em qualquer local

# Configurando o método 2

Sabemos que aqui vamos ter um retorno de response(resposta) Endereco

Aqui esta toda a configuração e comentada

private suspend fun recuperarEnderecoMetodo() {  
  
 /\* Retorno: estamos pegando o objeto convertido de JSON para nossa CLASSE  
  
 \*/  
 var retorno: Response<Endereco>? = null  
  
 try {  
 */\*\* Explicando o codigo  
 \* O metodo create ele faz assim:  
 \* Ele recebe a interface e ele mesmo cria e retorna para voce o objeto  
 \* Sem voce precisar de uma classe para usar o metodo  
 \* ai esse objeto permite voce acessa o metodo recuperarEndereco  
 \*/* val enderecoAPI = retrofit.create( EnderecoAPI::class.*java* )  
 retorno = enderecoAPI.recuperarEndereco() // metodo dentro da INTERFACE  
  
  
 }catch (e: Exception){  
 e.printStackTrace()  
 Log.i("info\_endereco", "Erro ao recuperar")  
 }  
  
 /\* Agora vamos verificar se ese retorno é nulo, se ele conseguiu ou nao trazer os dados  
 \* Não precisamos fazer o else, porque ele só vai cair aqui se por acaso nao retornar o catch \*/  
  
 if (retorno != null) {  
 /\* Se ele conseguiu trazer os dados e ele é diferente de nulo \*/  
 if (retorno.*isSuccessful*) {  
 /\* Esse metodo ele retorna sucesso se ele conseguiu fazer a requisição \*/  
 /\* Retorno é um objeto response e para capturar usamos o body(corpo)  
 \* O retorno é igual ao objeto = ENDERECO  
 \*  
 \* Porque nao precisamos instanciar? porque a propria retrofit  
 \* Ja fez isso pra gente, quando passamos la nossa classe  
 \* Ela mesmo ja converteu todos os objetos para dentro da classe que criamos.\*/  
  
 val endereco = retorno.body()  
 val rua = endereco?.logradouro  
 val bairro = endereco?.bairro  
 Log.i("info\_endereco", "Endereco: $bairro")  
  
 }  
  
 }  
  
  
}

# Permissão de internet para acessar a API

Já que estamos acessando a internet, precisamos de permissão para usar.

<?xml version="1.0" encoding="utf-8"?>  
<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools">  
  
 <uses-permission android:name="android.permission.INTERNET" />  
  
 <application