Corrutinas en Kotlin

Vamos a partir de la **App_03_01** proporcionada por el profesor.

PASOS PREVIOS

En primer lugar refactoriza el proyecto para cambiar el nombre de los paquetes. No deberían llamarse es.ivanlorenzo…., sino es.TUNOMBREYAPELLIDO

Una vez cambiado prueba que la aplicación funcione correctamente.
Revisa el código proporcionado hasta familiarizarte con él.

IMPLEMENTACIÓN

En la siguiente tarea consumiremos un servicio web para no tener que cargar los datos contenidos en la propia aplicación.
En esta tarea vamos a crear corrutinas en nuestra aplicación.

En primer lugar vamos a simular qué pasaría si trato de consumir un servicio web (para obtener los datos de los animales) desde el hilo principal sin usar corrutinas.

Para ello, en el objeto DatosAnimales añade un Thread.sleep(3000)

```
fun getDatosAnimales(): List<Animal>
{
   Thread.sleep(3000)
   return arrayListOf(
      Animal("Ballena" ....
```

Como no hemos cambiado de hilo, la experiencia de usuario será malísima si vamos a favoritos y luego volvemos a cargar la lista de animales.

Para usar corrutinas, en este mismo objeto iniciamos la corrutina con la operación **withContext** pasándole el **Dispatcher**.
También debemos cambiar el método para que sea un método **suspend** ya que son los únicos desde los que se pueden ejecutar una corrutina.

```
suspend fun getDatosAnimales(): List<Animal>
{
   return withContext(Dispatchers.IO)
   {
      Thread.sleep(3000)
      arrayListOf(
         Animal("Ballena" .....
```

Ahora el método get de la clase AnimalRepository nos dará error, ya que se llama a un método suspend. Únicamente se puede llamar a una función suspend desde otra suspend.

```
class AnimalRepository
{
   suspend fun get(): List<Animal> = DatosAnimales.getDatosAnimales()
}
```

Por último, desde AnimalViewModel debemos poner suspend o lanzar la corrutina desde un builder de corrutinas.
Desde el método cargarDatos llamaremos al builder launch de viewModelScope y meteremos dentro

```
fun cargarDatos()
{
   viewModelScope.launch {
       estaCargandoLiveData.postValue(true)
      animalesLiveData.postValue(repository.get())
      estaCargandoLiveData.postValue(false)
   }
}
```

Vemos como ahora no hay ningún bloqueo de la interfaz de usuario.