Teoría del Lenguaje

Informe Individual

Alumno: Tomás Nocetti

Profesor: Leandro Ferrigno

Padrón: 100853

Fecha de Entrega: 03/12/2018

Lenguaje: Kotlin

Introducción

Blue Apron es una compañía Americana dedicada a la venta de recetas y kits de cocina. Opera exclusivamente en los Estados Unidos y su funcionamiento es a través de suscripciones mensuales. Poseen una App por la cual los usuarios operan su membresía y obtienen distintos tipos de contenido digital complementario.

El objetivo de este informe es, basándose en el post hecho por Prem Nirmal - Software Engineer (*The First Step to Kotlin-izing our Android App*), proveer al lector una idea del proceso migratorio y de adaptación de Kotlin en una empresa de tamaño pequeño-mediano.

El principio de la migración

Cuando Kotlin se anunció como lenguaje oficial de desarrollo para Android, la compañía Blue Apron decidió comenzar un proceso de integración en su 'codebase', lo cual obviamente no fue/es una tarea rápida.
En el momento de tomar la decisión se encontraron con múltiples opciones en su código base para comenzar. Una de las cosas que hace hincapié Prem Nirmal en su nota, es que necesitaban una transición suave para alivianar el proceso de aprendizaje y lo que conlleva la incorporación de un nuevo lenguaje. Es entonces cuando el 'network layer' es lo primero que se decide migrar.

Como fue el proceso ?

Para la migración se utilizó parte del tooling provisto por Android Studio que convierte código Java a Kotlin. Sin embargo el 'network layer' que necesitaban migrar estaba compuesto por múltiples POJOs ("sigla creada por Martin Fowler, Rebecca Parsons y Josh MacKenzie en septiembre de 2000 y utilizada por programadores Java para enfatizar el uso de clases simples y que no dependen de un framework en especial") para lo cual las Data Classes provistas por Kotlin le brindaban funcionalidades ya mencionadas en el informe grupal como equals, hashCode y toString. El problema fue que la conversión automática no pasaba las clases de Java a Data Classes, por lo que el proceso tuvo partes manuales.

Las instancias de PJOS utilizadas originalmente eran parseadas de un Json utilizando la la librería Moshi. La adaptación de esto al sistema fue sencilla al principio, debido a que ofrece soporte directo para Kotlin bajo la notación `@Json`.

Para poder obtener los beneficios de las Data Classes, las propiedades deben ser declaradas en el constructor. A continuación se presenta el ejemplo provisto en la nota sobre una clase llamada `NutritionalInfoNet`:

```
import com.squareup.moshi.Json

/**
 * Class representing a recipe's nutritional information.
 * The [calories_per_serving] field does not include the correct
annotation for Moshi to be able to parse it.
 */
data class NutritionalInfoNet(
        @JvmField
        var id: String,
        @JvmField
        var recipe_id: String,
        @JvmField
        @Json(name = "calories-per-serving") // FAIL!
        var calories_per_serving: String,
        @JvmField
        var media: List<AssetNet>)
```

- Recordemos que Kotlin genera automáticamente getters y setters. La notación `@JvmField` le indica al compilador que no le haga para poder acceder el código desde Java como ‘instance fields’ y no como ‘properties’.
- Debido a que las propiedades se encuentran en el constructor de la clase aparece una ambigüedad a la hora de utilizar Moshi de si el parámetro es un ‘instance field’ o un parámetro del constructor. La solución a esto la encontraron ‘por las malas’ luego de darse cuenta que no se estaban generando las instancias correctamente. En definitiva consistió en cambiar la notación a `@field:Json`

Conclusiones

Esto es solamente un ejemplo de algunos ‘case scenarios’ a los cuales el equipo de Android de Blue Apron se tuvo que enfrentar. Luego de terminar la primera etapa el codigo quedo en proporción distribuido de la siguiente manera:

El ‘network layer’ fue la primera fase de la migración. Al principio notaron un aumento en los tiempos de compilación pero luego de Kotlin 1.2, en la cual se incorporó el build cache al build process de Android, se mejoró circunstancialmente.
Hasta aquí se puede hacer un recupero de que las razones fundamentales iniciales para la adopción del lenguaje van más por el lado de la adopción de Google como lenguaje de desarrollo oficial y que los contratiempos encontrados son característicos de la incorporación de un nuevo lenguaje al stack. Si se hace hincapié en las ventajas de un feature como el Data Class que en definitiva terminó achicando los archivos.

Bibliografía:

- https://blog.blueapron.io/java-to-kt-the-first-step-to-modernizing-our-android-app-288670836e85
- https://kotlinlang.org/docs/reference/java-to-kotlin-interop.html
- https://en.wikipedia.org/wiki/Blue_Apron