Kotlin en Corda.

Corda es una plataforma de blockchain y contratos inteligentes totalmente construida en Kotlin. Como una plataforma de blockchain, Corda permite a las partes realizar transacciones directamente, con valor. Los contratos inteligentes permiten a Corda hacer esto utilizando acuerdos complejos y cualquier tipo de asset. Esta capacidad tiene amplias aplicaciones en todas las industrias, incluidas las finanzas, la cadena de producción y healthcare.

Por qué Kotlin?

Al momento de tomar esta decisión, el equipo de Corda tuvo que tomar dos decisiones principales.La primera, que plataforma utilizar. Habiendo elegido la JVM por razones conocidas dentro del espacio empresarial (tales como que es una plataforma escalable, thread-safe, con cross platform runtime, entre otras) .

Al inicial el proyecto, no se tenía en claro si él mismo terminaría siendo un producto. Corda comenzó como una colección de prototipos que exploraba una nueva serie de ideas y requerimientos. Al no sabes si estos prototipos acabarían siendo algo o únicamente servirían para inspirar a otros productos del mercado, se encontraron con una decisión complicada.
Por un lado, querían explorar algoritmos y estructuras de datos de manera rápida y que pudiera ser productiva.Por el otro, necesitaba ser algo que pudiera ser usado para construir un gran producto empresarial.

Por supuesto Java encajaba, pero su falta de conveniencias modernas reduce significativamente la productividad y, más sutilmente, la moral de los developers.
El tipado dinámico estaba fuera: La correctitud, las herramientas y los beneficios del tipado estático son demasiado grandes como para ignorarlos.
Lenguajes radicalmente diferentes del mainstream también estaban fuera, ya que querían contratar expertos de la industria financiera.
Estos requerimientos los llevó a elegir entre Kotlin, Scala y Ceylon.

Las razones por las que optaron por Kotlin son las siguientes:

- Interoperabilidad casi sin problemas con Java
- Los programas en Kotlin utilizan una versión mejorada por compilador de las colecciones del JDK. Por lo tanto, no hay problemas de interoperabilidad creados por el uso de una biblioteca de colecciones diferente. Pasar colecciones dentro y fuera de las bibliotecas de Java es algo que hacen en Corda, por lo que es importante que esto sea indoloro.
- Las clases de Kotlin producen APIs de Java de aspecto normal con métodos get / set / is según sea apropiado para el tipo. No se requieren anotaciones especiales. Debido a que Corda expone una API que está destinada a ser utilizada de forma transparente por los desarrolladores de Java, esta es una gran ventaja: el código ordinario tiende a producir APIs que no pueden diferenciarse de una API de Java, con solo algunos aspectos menores que requieren reflexión (por ejemplo, es necesario anotar manualmente los métodos para especificar que arrojan excepciones comprobadas, de lo contrario no se pueden capturar desde Java).
- El inling de funciones como cómo map / filter / fold / groupBy se realiza mediante la interfaz del compilador en lugar de confiar en la JVM para hacerlo. Desafortunadamente, los compiladores JIT de JVM, aunque en general son excelentes, no eliminan de manera confiable la sobrecarga del uso intensivo de funciones de orden superior. El uso de Kotlin soluciona esto y le permite controlar el flujo dentro de las funciones lambda. Esta es el tipo de característica que no se ve, pero que suma, ya que se utiliza código escrito de forma

funcional por todos lados y esto puede llevar a un agujero de performance si se traduce mal a código máquina.
- Debido a que el código de Kotlin se a su equivalente Java casi a la perfección, casi todas las herramientas orientadas a Java existentes funcionan de manera inmediata. Esto no siempre es así en otros idiomas, por ejemplo, Quasar se esfuerza por instrumentar el código de Scala porque quiere method annotations y Scala traduce lambdas a métodos que no pueden tener annotations.
- Su excelente documentación y su pequeña librería estándar hacen que sea un lenguaje que se aprende rápido.

Si no hubiera existido Kotlin al momento de iniciar el proyecto, el lenguaje que seguramente hubieran utilizada es Scala, siendo que Kotlin tiene gran inspiración en él y son ambos grandes lenguajes.

Experiencias luego de un año de uso.

Lo más destacado es escuchar cómo disfrutan los programadores trabajando con el lenguaje. Muchas veces pedirle a alguien que aprenda un nuevo lenguaje de programación apenas son contratados en una apuesta que puede hacer que dicha persona odie el lenguaje y baje su productividad en vez de incrementarla. Esto no fue para nada un inconveniente en el equipo de corda.
Uno de los inconvenientes que pudieron llegar a notar fué la aceptación del consumidor. Al iniciar por supuesto hubo miradas poco convencidas al escuchar el lenguaje elegido. Por suerte, con el pasar del año la pregunta “¿Por qué Kotlin?” se fue esfumando poco a poco mientras que la gente iba notando que no era tan riesgoso como generalmente los nuevos lenguajes son.
Otra duda que tenían era si iba a tener el soporte comercial necesario. Es común que nuevos lenguajes no se mantengan por mucho o tomen un camino distinto al que necesita el producto. Viendo que JetBrains, que ha estado en el mercado por más de 15 años, implementa Kotlin en el core de sus principales productos, el lenguaje corre pocos riesgos de no ser mantenido.

Como conclusión, el equipo sostiene que esta para nada arrepentido de haber elegido este lenguaje. Fué un riesgo, pero uno controlado. Ha funcionado bien y lo volverían a elegir si tuvieran que hacerlo.