Metodologías de Diseño y Programación Avanzadas

Ignacio Slater Muñoz
reachme@ravenhill.cl

Contents

Iniciar un proyecto requiere considerar varios *aspectos críticos*, incluyendo la selección de tecnologías y la planificación del tiempo y estructura del proyecto. Sin embargo, un factor frecuentemente subestimado es el cambio. El cambio es inevitable y debe considerarse desde el inicio para evitar que el proyecto se vuelva obsoleto antes de su finalización.

Esta consideración es igualmente crucial al desarrollar un curso. Si no anticipamos los cambios, el contenido del curso podría quedar desactualizado antes de que esté completo.

Por esta razón, hemos diseñado este curso para ser modular y fácil de actualizar. Cada unidad se compone de secciones independientes que pueden ser modificadas sin afectar el resto del contenido. De esta forma, podemos mantener el curso actualizado y relevante para lxs estudiantes.

Al hablar de la preparación de un curso, es esencial elegir las tecnologías adecuadas. Para este proyecto, hemos seleccionado Vue 3 y Vite para el desarrollo del sitio web, Cloudflare Pages para el alojamiento, y Kotlin como lenguaje de programación.

El sitio web puede ser encontrado en https://ravenhill.pages.dev.

1. Unidad 1: Introducción

1.1. Kotlin

Kotlin es un lenguaje de programación multiplataforma, desarrollado por JetBrains, que integra características de la programación orientada a objetos y funcional. Es conocido por su sintaxis concisa y capacidad para compilar no solo en JavaScript (JS) y WebAssembly (WASM) para ejecución en navegadores, sino también en Java Virtual Machine (JVM) para servidores y aplicaciones Android, así como en LLVM para aplicaciones de escritorio y sistemas embebidos.

En este curso, nos centraremos en la programación en Kotlin para la JVM, que es la plataforma más utilizada para este lenguaje. Sin embargo, los conceptos y técnicas que aprenderás son ampliamente aplicables a otras plataformas que soporta Kotlin y pueden ser útiles incluso en el aprendizaje de otros lenguajes de programación modernos.

1.1.1. A Taste of Kotlin

A continuación, te presentamos un ejemplo simple de Kotlin para darte una idea de cómo se ve y se siente el lenguaje.

```kotlin
data class Person(        // (1)
    val name: String,
    val age: Int? = null  // (2)
)

fun main() {
    val persons = listOf( // (3)
        Person("Harrier Du Bois"),
        Person("Kim Kitsuragi", age = 43) // (4)
    )
    val youngest = persons.minByOrNull { it.age ?: Int.MAX_VALUE }  // (5)
    println("The youngest is: $youngest") // (6)
}
// Output: The youngest is: Person(name=Kim Kitsuragi, age=43)
```

1. Declara una clase de datos `Person` con dos propiedades: `name` de tipo `String` y `age` de tipo `Int` opcional.
2. La propiedad `age` tiene un valor predeterminado de `null`.
3. Declara una lista inmutable de personas con dos elementos.
4. El segundo elemento de la lista tiene un valor de edad nombrado de 43.
5. Encuentra la persona más joven en la lista utilizando `minByOrNull` y el operador de elvis `?:`.
6. Interpola la variable `youngest` en una cadena y la imprime en la consola.

1.2. Lo básico

1.2.1. Expresiones vs. Declaraciones

En programación, es crucial distinguir entre **expresiones** y **declaraciones**, ya que cada una juega un papel diferente en la estructura y ejecución de los programas. A continuación, se explican estos conceptos en el contexto de Kotlin, aunque es importante notar que en otros lenguajes de programación, como Scala o Rust, los ciclos pueden ser expresiones o los condicionales pueden ser declaraciones.

Definición 1.2.1.1 (Expresiones): Las expresiones son fragmentos de código que producen un valor y pueden ser compuestas con otras expresiones. En Kotlin, las expresiones pueden ser tan simples como una constante o tan complejas como una función anónima. Ejemplos comunes de expresiones incluyen operaciones aritméticas, operadores lógicos y llamadas a funciones.

Definición 1.2.1.2 (Declaraciones): Las declaraciones son fragmentos de código que realizan una acción pero no retornan un valor. En Kotlin, las declaraciones no pueden ser compuestas con otras declaraciones, lo que significa que no pueden ser anidadas. Ejemplos comunes de declaraciones incluyen la asignación de variables, la ejecución de ciclos y la definición de funciones.