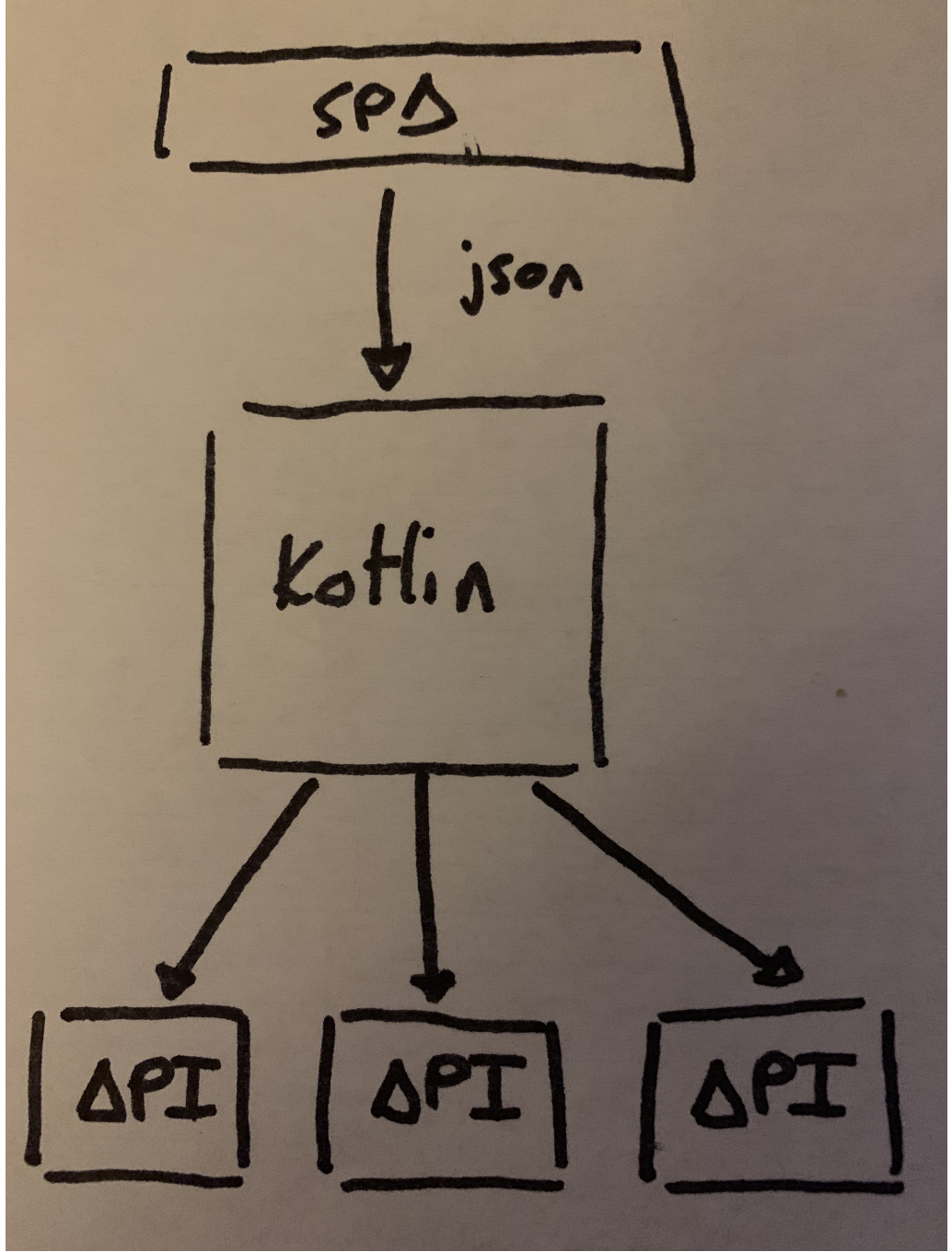
What we learnt building a microservice in Kotlin

Mario Fernandez

Lead Developer

ThoughtWorks

Context


SPA
json
Kotlin
API
API
API


spring
boot

Why switch?

- Immutability: Be sure of what your data looks like

Why switch?

- Immutability: Be sure of what your data looks like
- Null safety: Less unexpected exceptions

Why switch?

- Immutability: Be sure of what your data looks like
- Null safety: Less unexpected exceptions
- Compactness: More bang for the buck

Let's see some examples

Pseudo DSLs

```
object CountryLanguageMatrix {
    val locales = mapOf(
            Country("de") to listOf(Language("de_DE")),
            Country("us") to listOf(Language("en_US")),
            Country("ca") to listOf(Language("en_US"), Language("fr_FR")
            Country("cn") to listOf(Language("zh_CN"), Language("en_GB")
    )
}

fun Country.isDefaultLanguage(language: Language): Boolean {
    return CountryLanguageMatrix.locales[this]?.run {
       this[0] == language
    } ?: false
}
```

Extension functions as a replacement for helper functions

```
package my.utils

object Utils

fun String.asStream(): InputStream {
    return Utils.javaClass.classLoader.getResourceAsStream(this)
}

fun InputStream.readTextAndClose(charset: Charset = Charsets.UTF_8): Str
    return this.bufferedReader(charset).use { it.readText() }
}
```

```
import my.utils.asStream
import my.utils.readTextAndClose

val jwt = "jwt".asStream().readTextAndClose()
```

JSON

```
data class UserId(private val value: String) {
    companion object {
        @JvmStatic
        @JsonCreator
        fun create(value: String) = UserId(value.toLowerCase())
    }

    @JsonValue
    override fun toString() = value
}
```

```
class User(
  id: UserId,
  name: String
)
```

```
{
  "id": "33242123",
  "name": "The Dude"
}
```

Testing


JUnit
5


MockK


ATRIUM

Meaningful test names

```
@Test
fun `is case insensitive`() {
}
```

Fluent declarations

```
val car = mockk<Car>()

every { car.drive(Direction.NORTH) } returns Outcome.OK
car.drive(Direction.NORTH) // returns OK

verify { car.drive(Direction.NORTH) }
```

Capturing arguments

```
@Test
fun `injects header into the request and passes it to the filter`() {
    filter.doFilter(request, response, filterChain)

    slot<ServletRequest>().let { slot ->
        verify { filterChain.doFilter(capture(slot), response) }

        expect(slot.captured).isA<HttpServletRequestWrapper> {
            expect(subject.getHeader(Headers.EXTRA_HEADER))
              .toBe("value")
        }
    }
}
```

Null Safety pitfalls

```
override fun preHandle(
  request: HttpServletRequest,
  response: HttpServletResponse,
  handler: Any): Boolean {
    val value: String = request.getHeader("Header")

    // value is technically of type String!
    // It can actually be null
}
```

Conclusion

Quick transition

Productivity boost

Links

- https://hceris.com/painless-json-with-kotlin-and-jackson/
- https://hceris.com/mock-verification-in-kotlin/