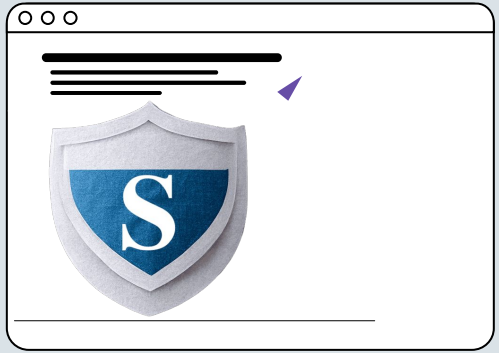
SPRING AND REACT WITH KOTLIN

Full-stack's phantom pain



ABOUT ME

Andrey Kuleshov

https://github.com/**akuleshov7**

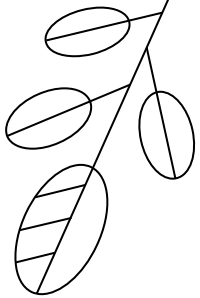
intel. Deutsche Bank HUAWEI

saveourtool/save-cloud Public

Cluster-based cloud mechanism for running SAVE framework

Kotlin 28 1

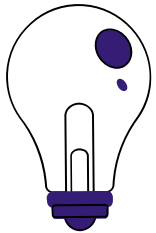
https://github.com/saveourtool

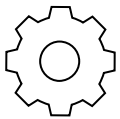


01

INTRODUCTION

MOTIVATION OF THIS TALK





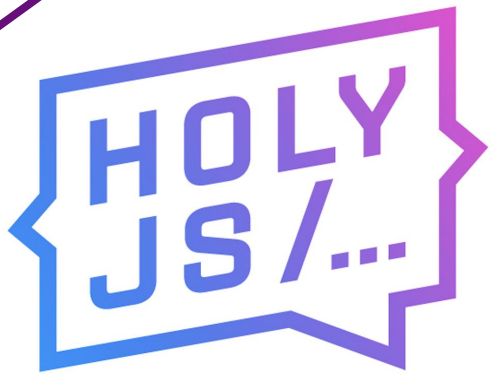

HEISENBUG
Benchmarking
and testing of code
analyzers
and compilers
Andrey
Kuleshov
Huawei

BASIC MOTIVATION OF THIS TALK

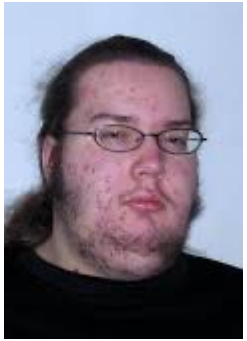
Joker<?>

BASIC MOTIVATION OF THIS TALK

Joker<?>



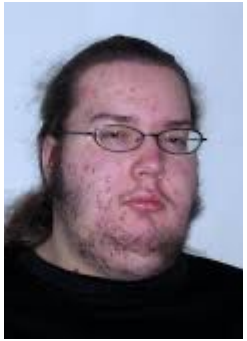
BASIC MOTIVATION OF THIS TALK



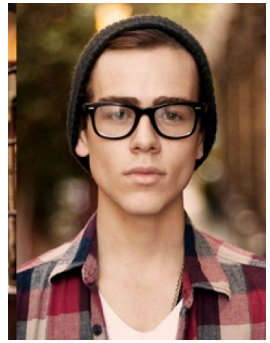
Joker<?>

HOLY JS /...

BASIC MOTIVATION OF THIS TALK



Joker<?>



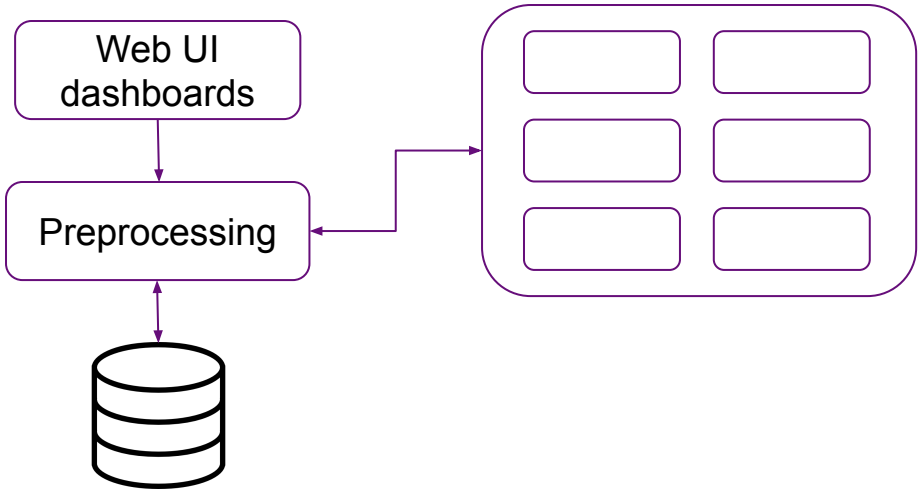
Sometimes **we** also want to make a frontend

HOLY JS /...

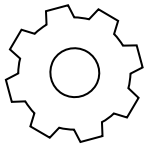
REAL PROBLEM

OUR SCOPE

Pool of **distributed** cloud executors for running batches of specific tests for code analyzers

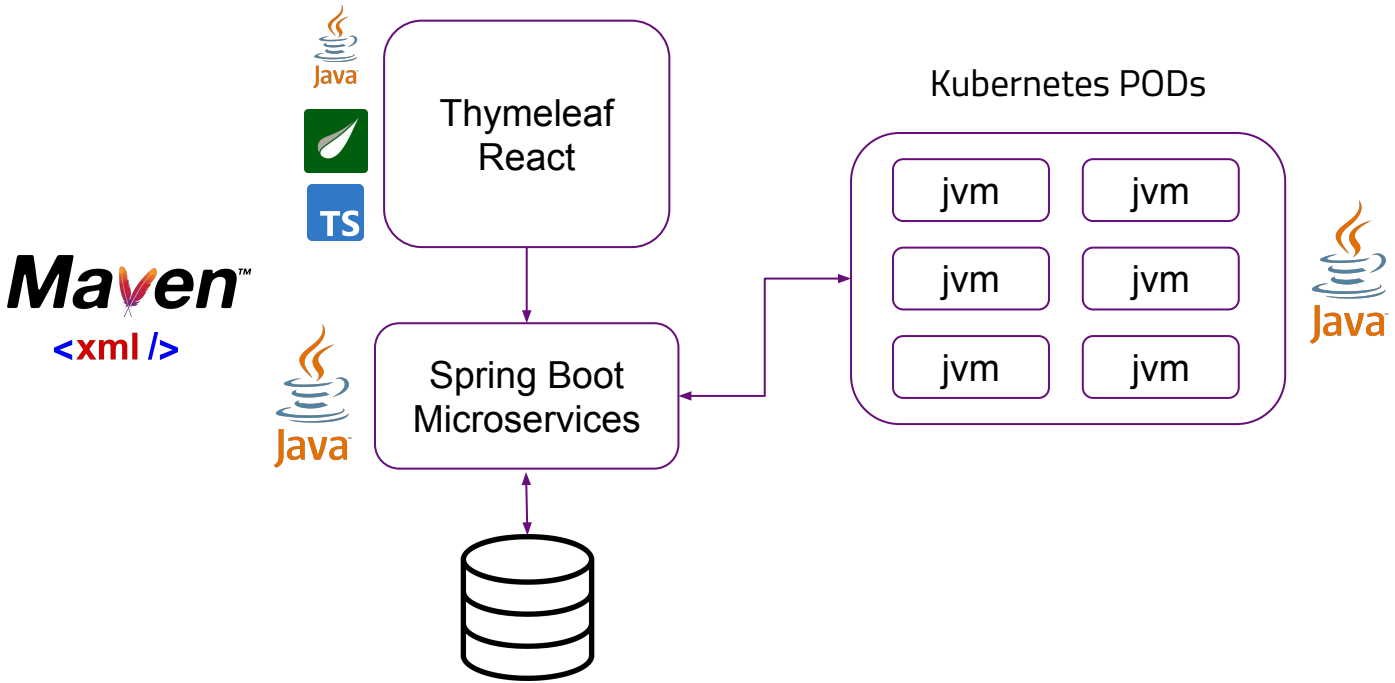


Database for historical info

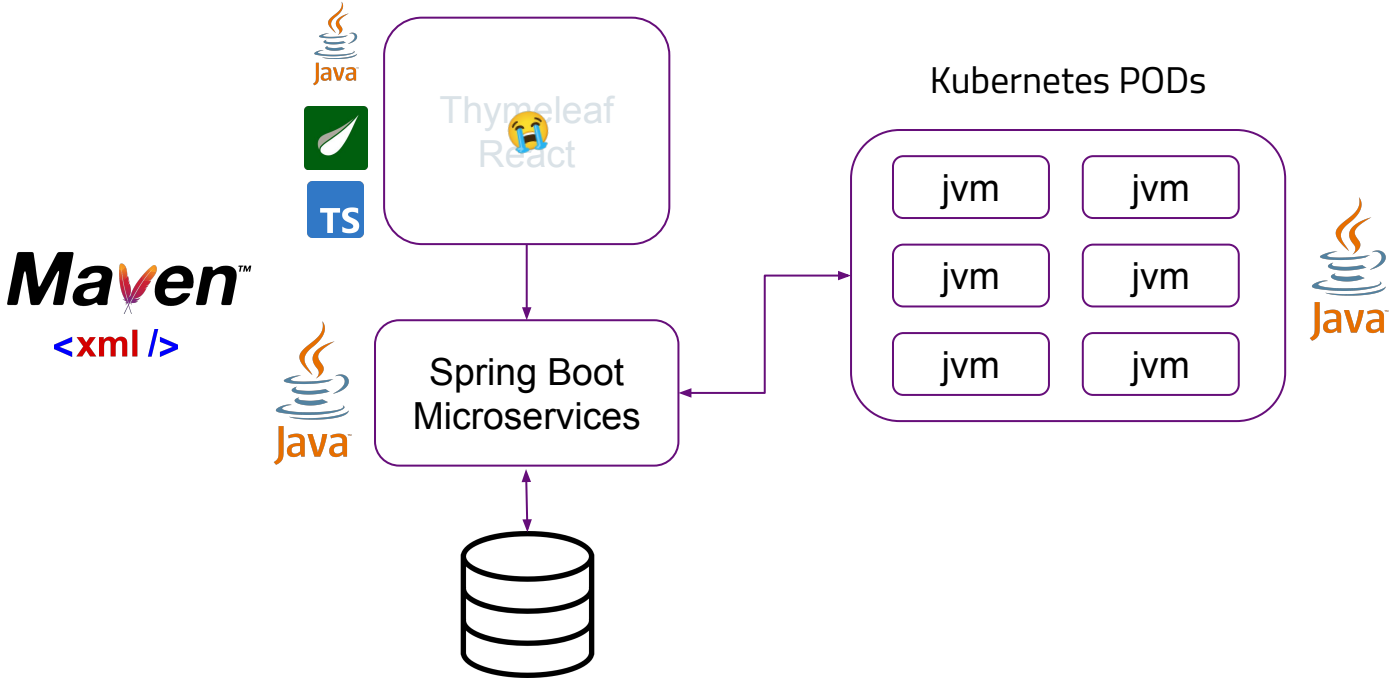



AS JAVA BACKEND DEVELOPERS HOW DO WE CAN SOLVE IT?

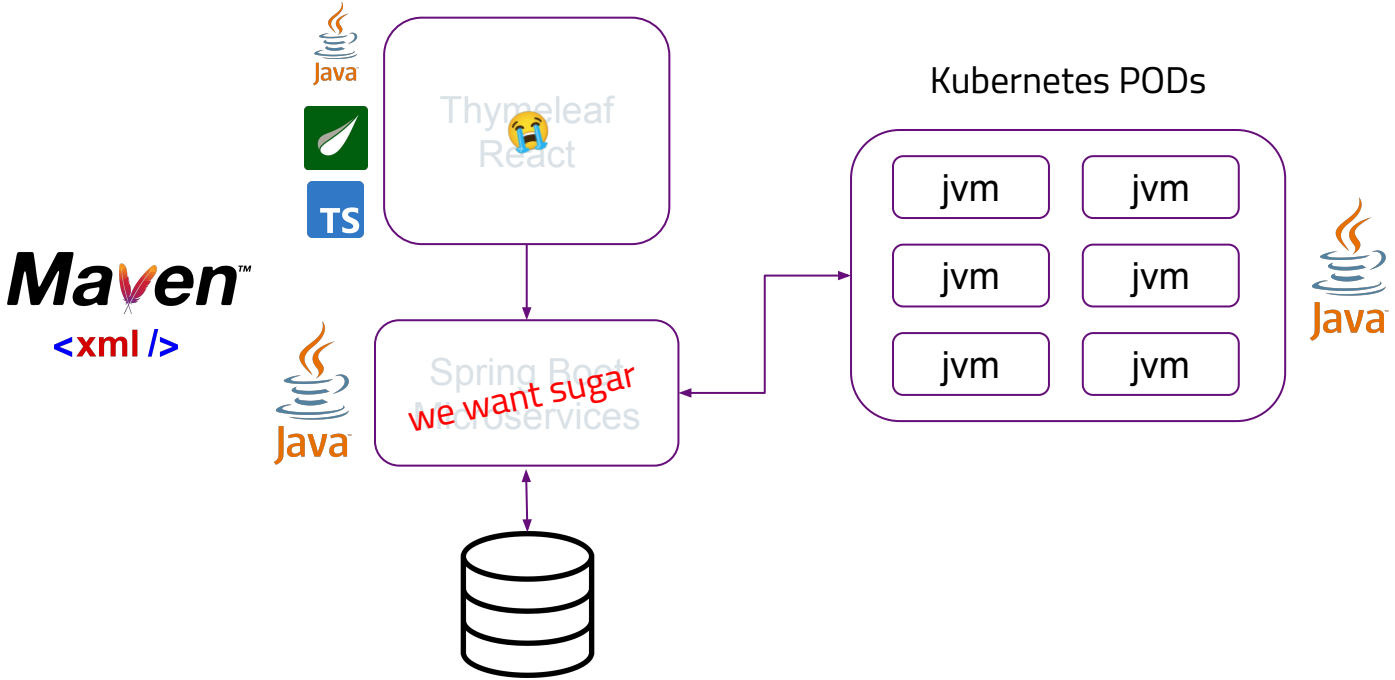

FAMILIAR ENTERPRISE APPROACH



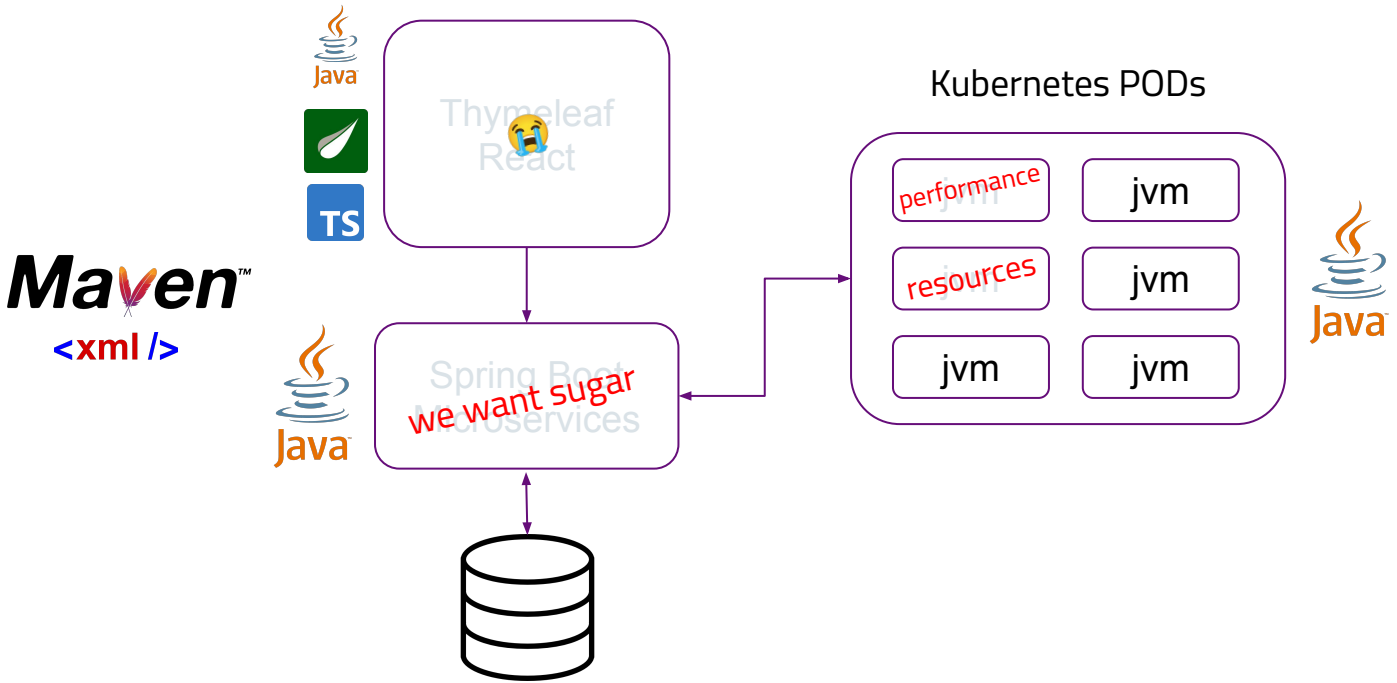
FAMILIAR ENTERPRISE APPROACH



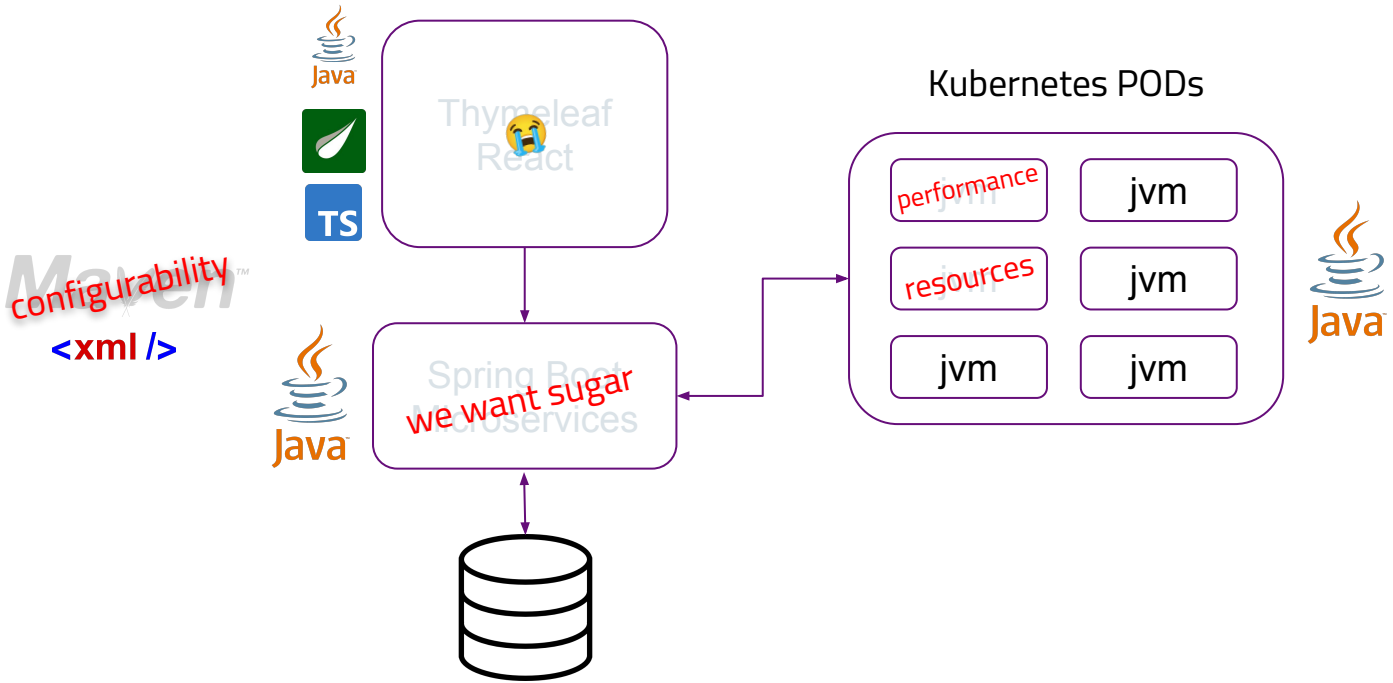
FAMILIAR ENTERPRISE APPROACH



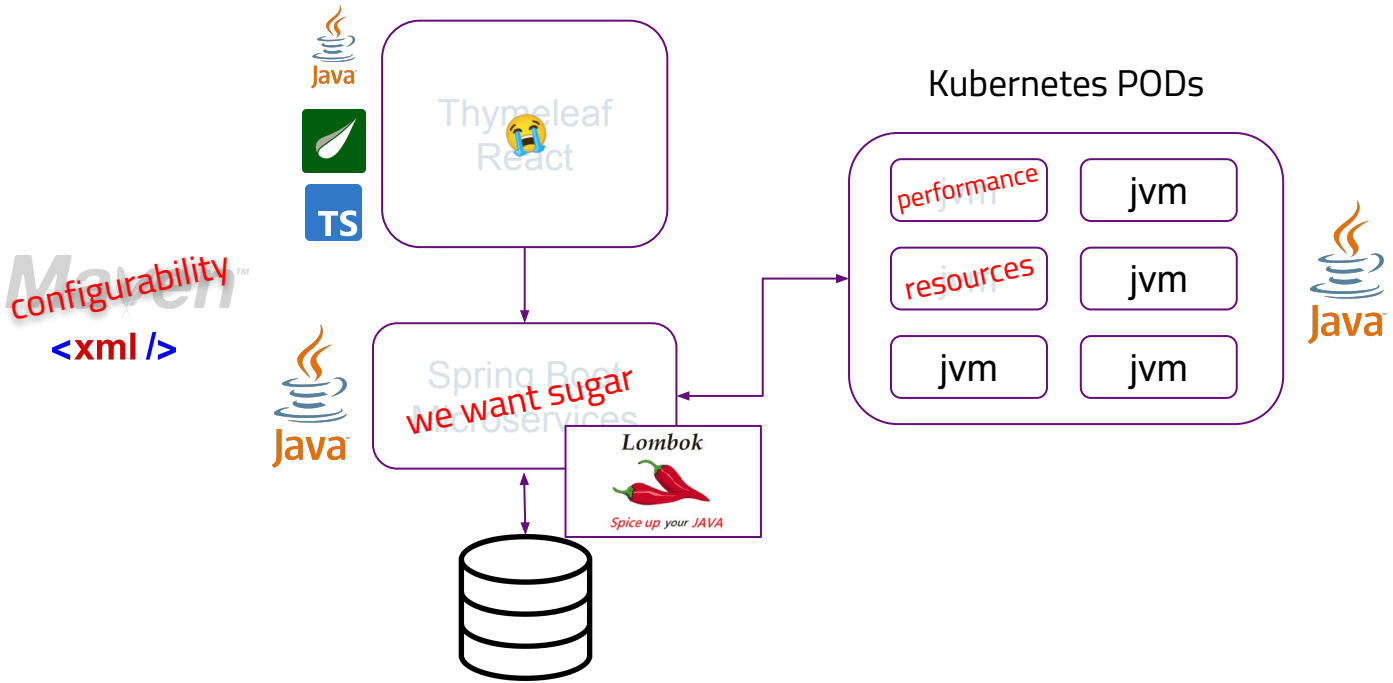
FAMILIAR ENTERPRISE APPROACH



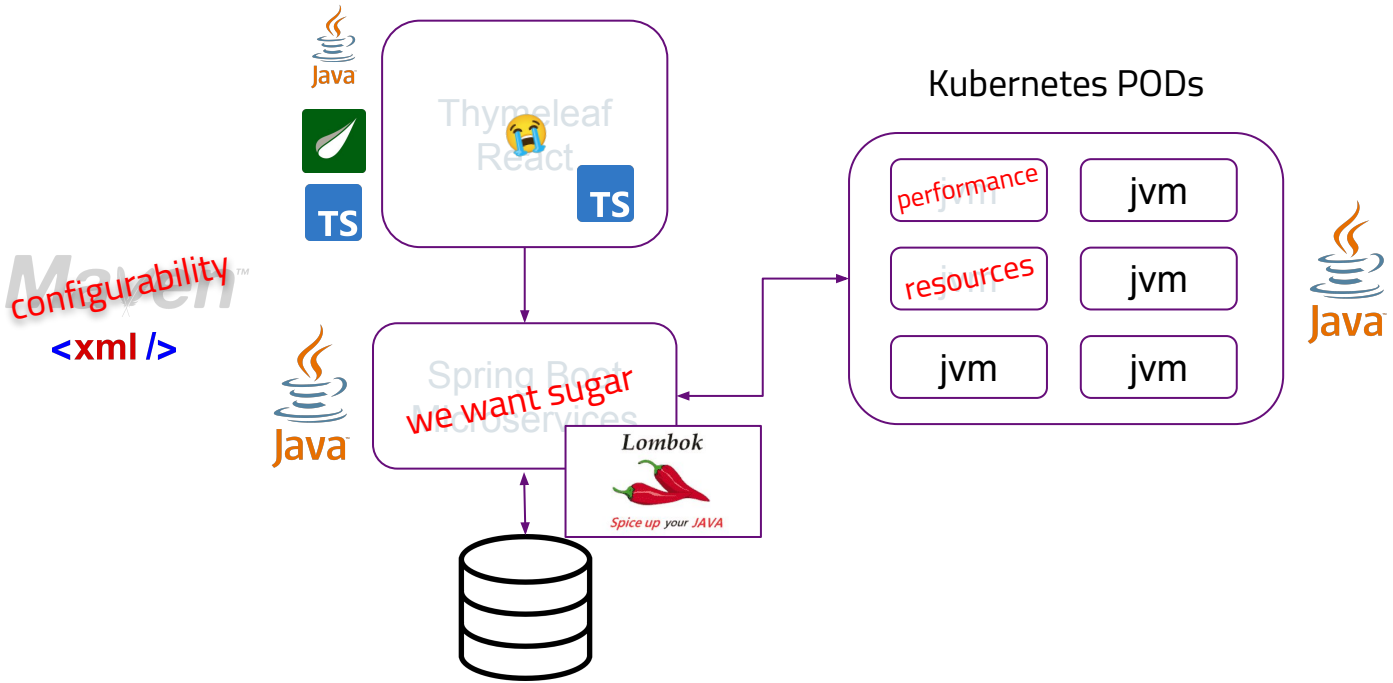
FAMILIAR ENTERPRISE APPROACH



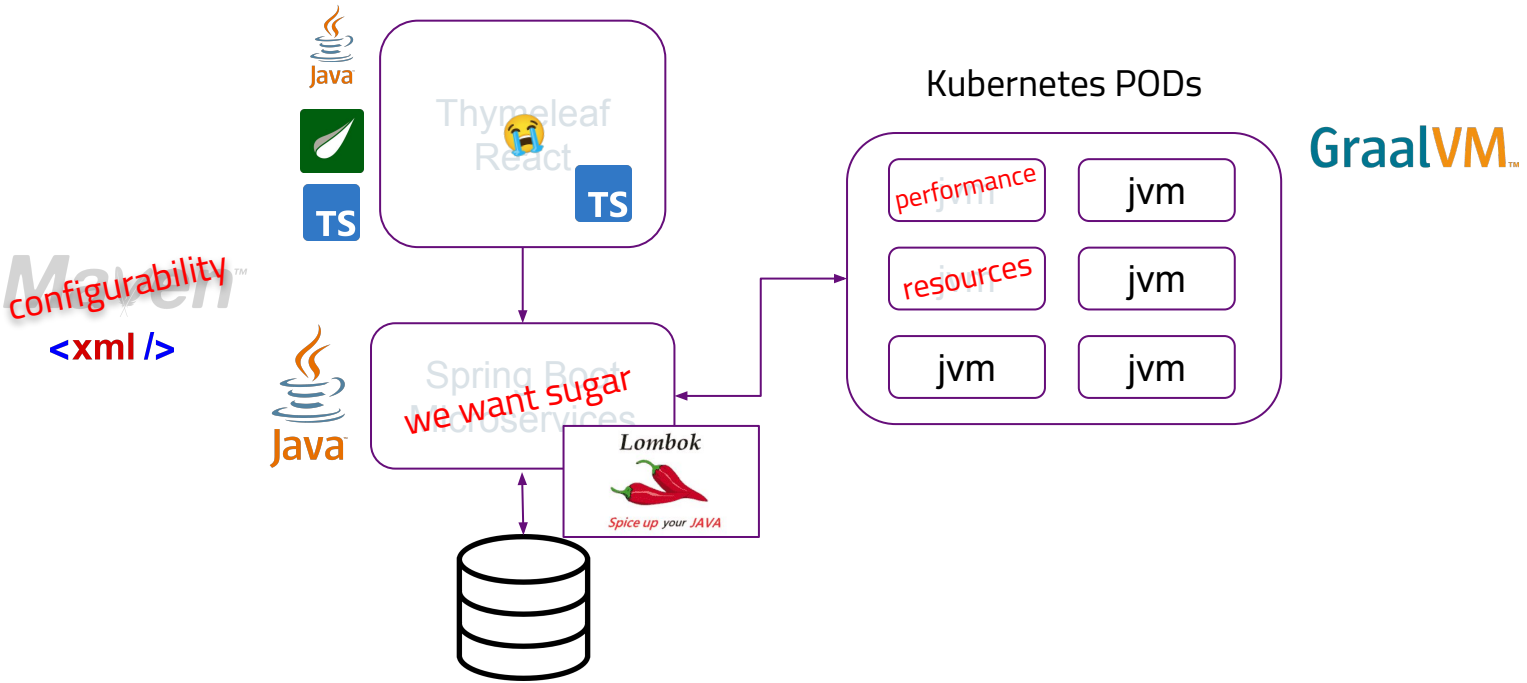
FAMILIAR ENTERPRISE APPROACH



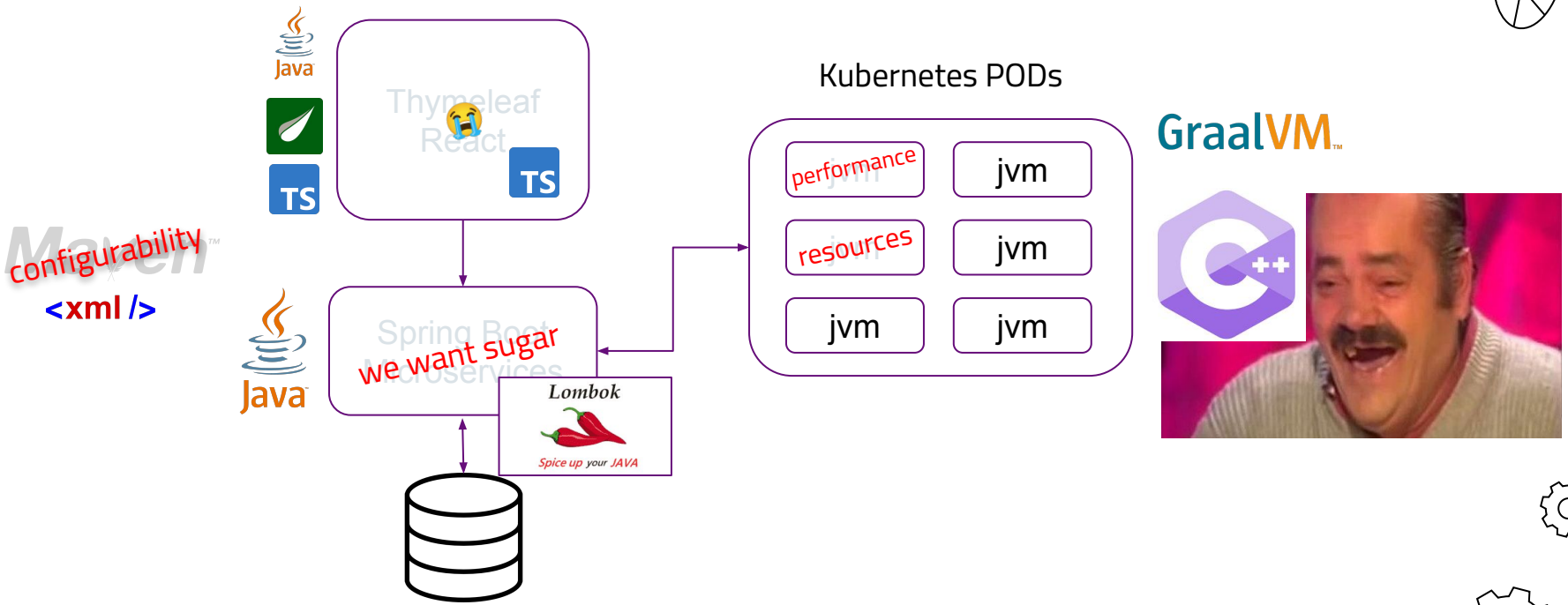
FAMILIAR ENTERPRISE APPROACH



FAMILIAR ENTERPRISE APPROACH



FAMILIAR ENTERPRISE APPROACH

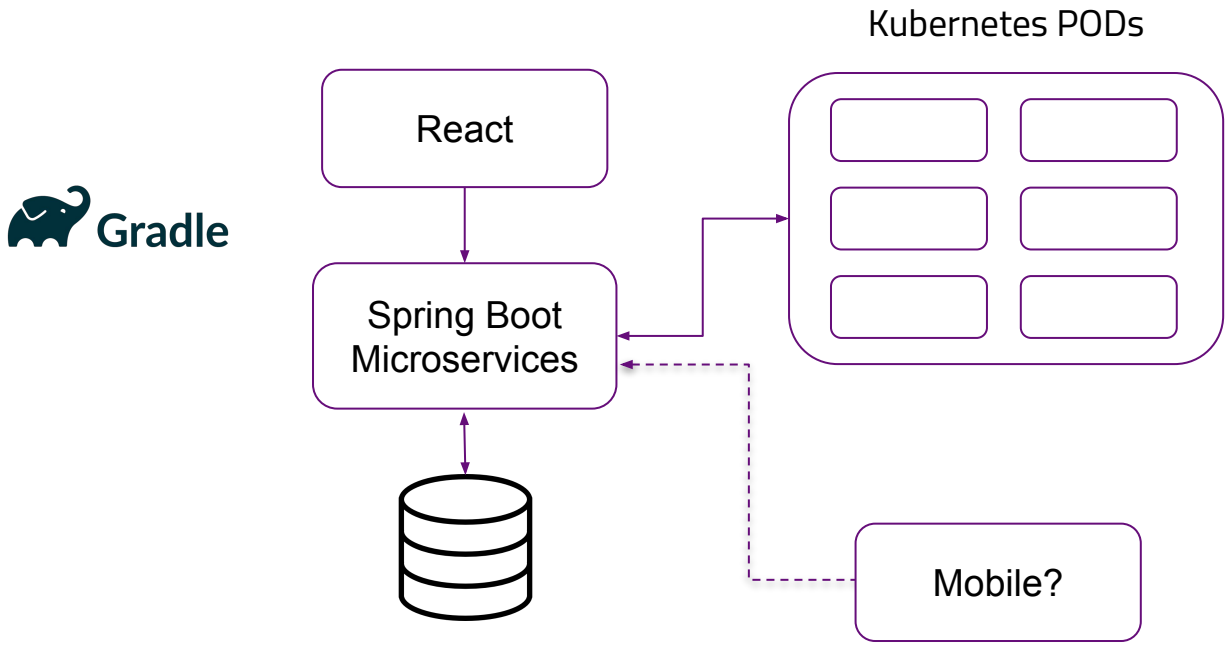


AND HERE IS WHERE THE STORY BEGINS

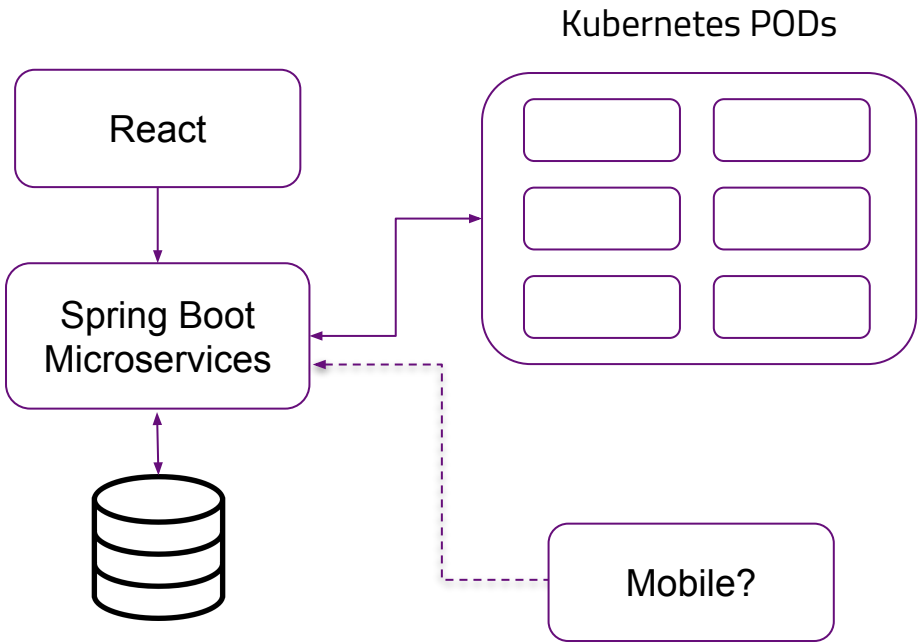
02

MULTI PLATFORM PROGRAMMING

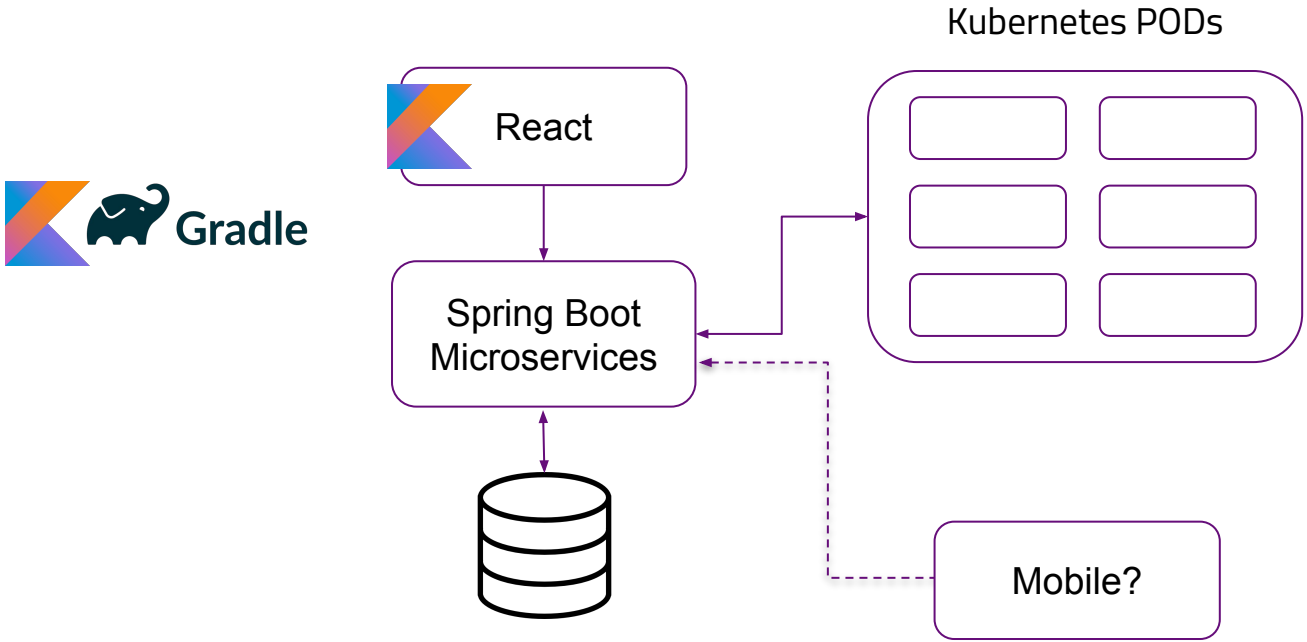
MPP SOLUTION



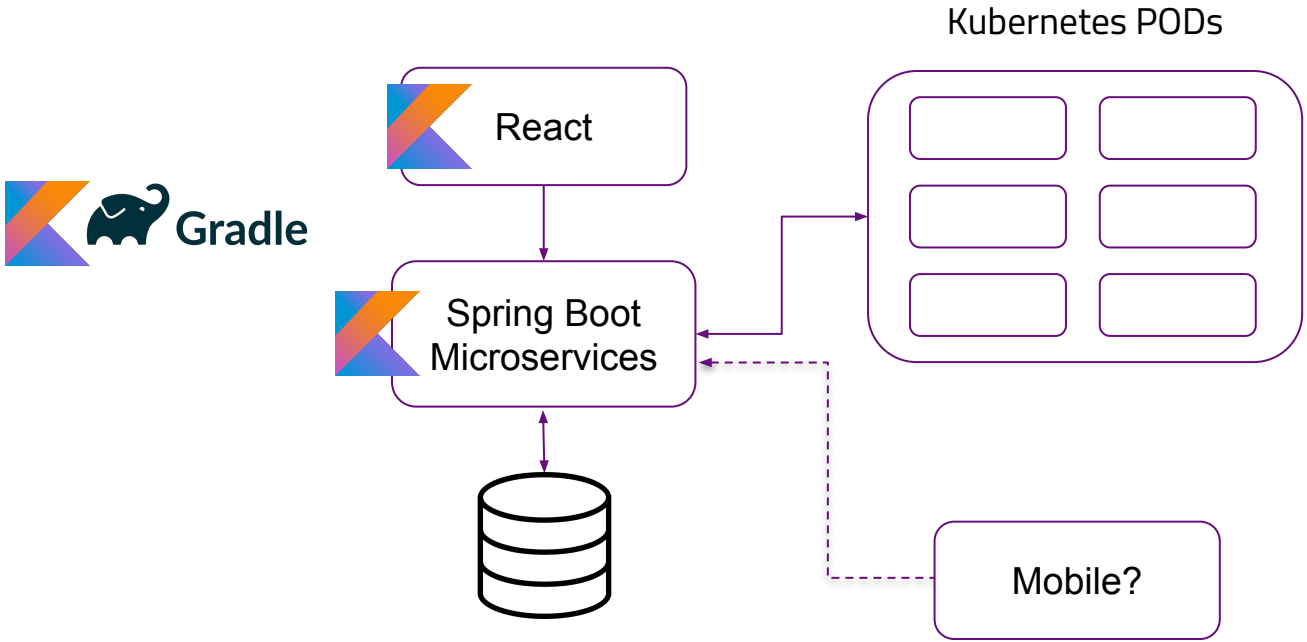
MPP SOLUTION



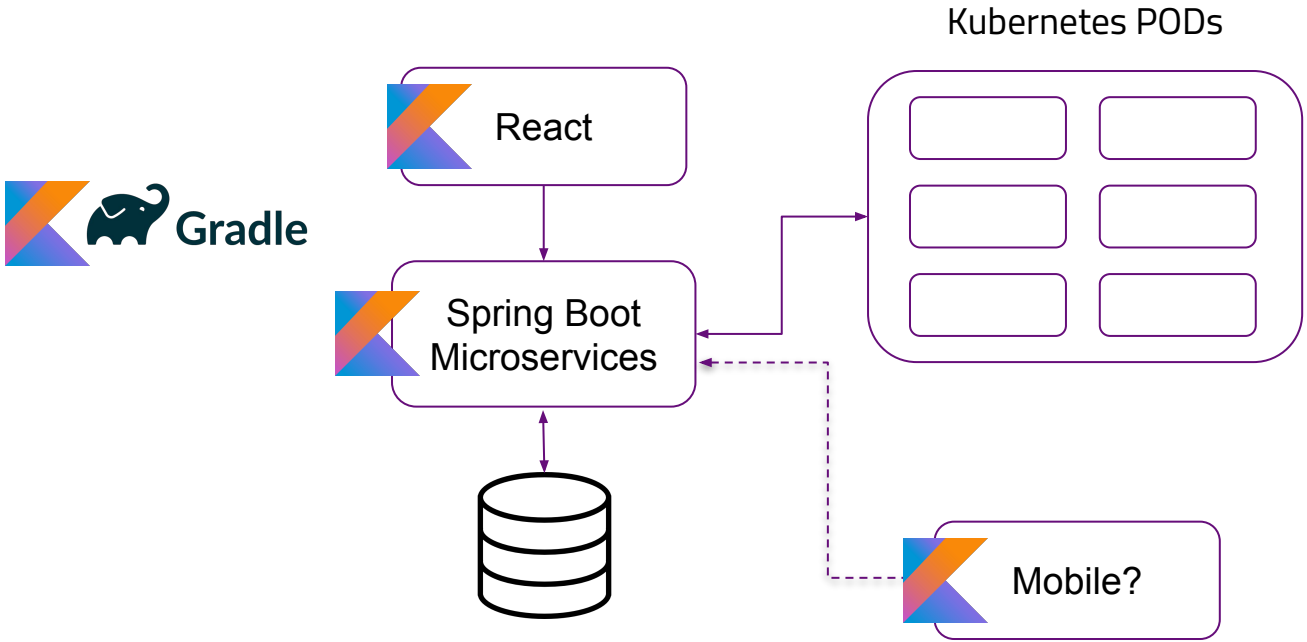
MPP SOLUTION



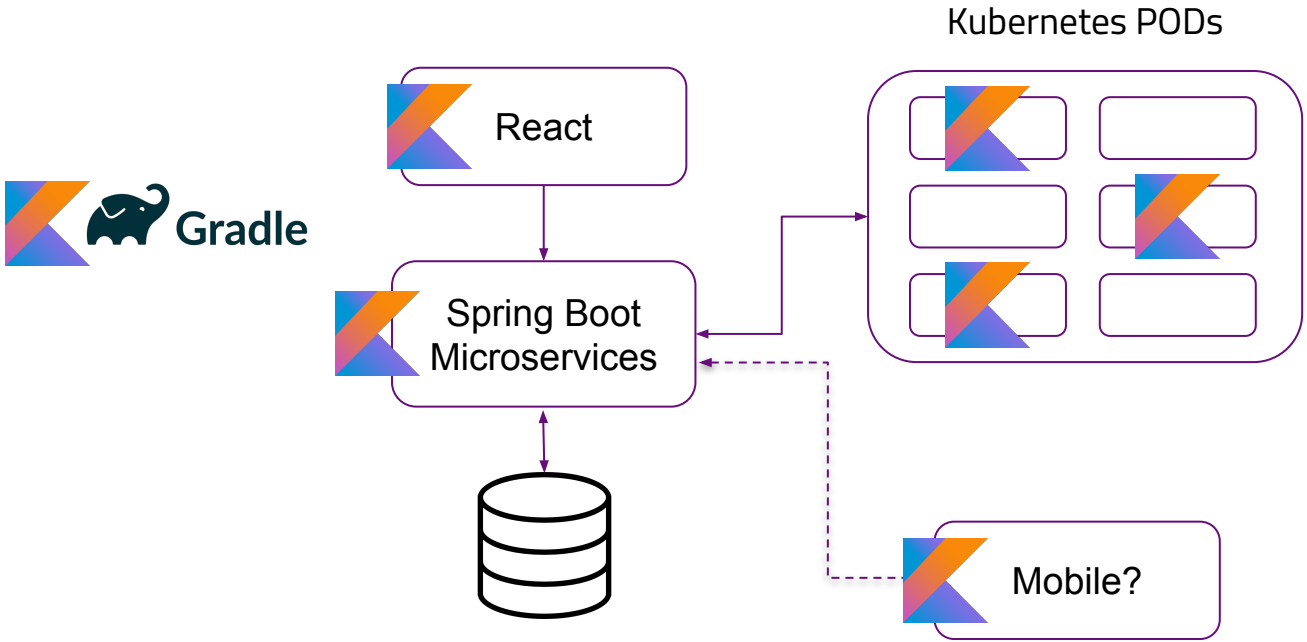
MPP SOLUTION



MPP SOLUTION

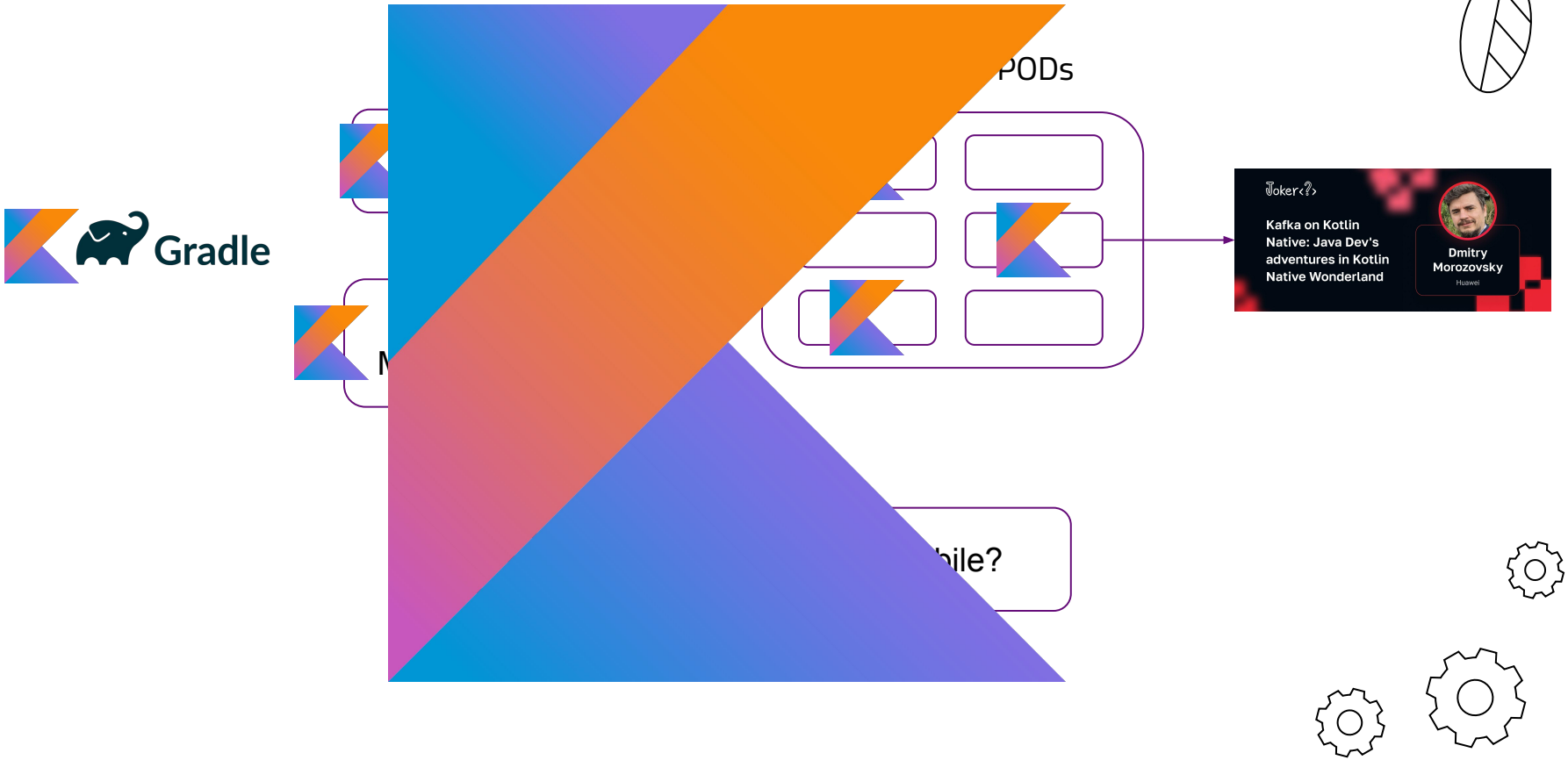


MPP SOLUTION



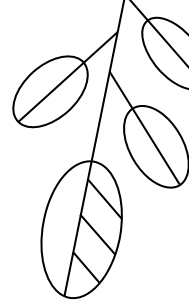
MPP SOLUTION

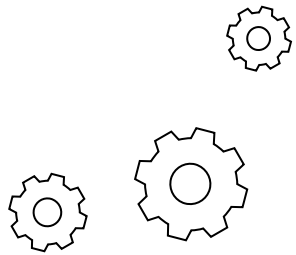
MPP SOLUTION



WHY KOTLIN?

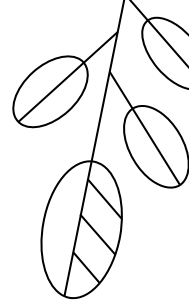
- We love **Java** and we use **Spring**

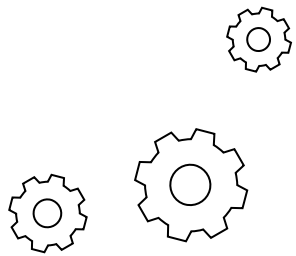




WHY KOTLIN?

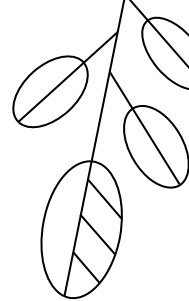
- We love **Java** and we use **Spring**
- But it will be great to have nice syntax sugar

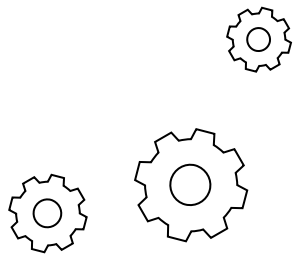




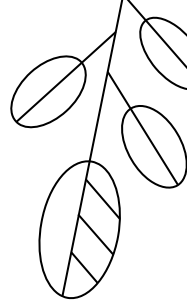
WHY KOTLIN?

- We love **Java** and we use **Spring**
- But it will be great to have nice syntax sugar
- We want to learn how to write FE with familiar languages

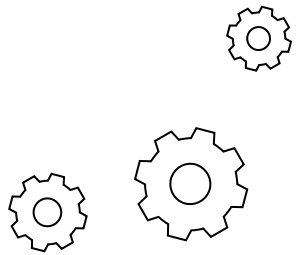




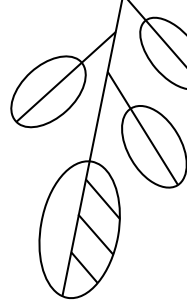
WHY KOTLIN?



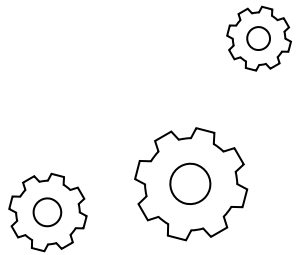
- We love **Java** and we use **Spring**
- But it will be great to have nice syntax sugar
- We want to learn how to write FE with familiar languages
- We want **one** programming language for everything



WHY KOTLIN?



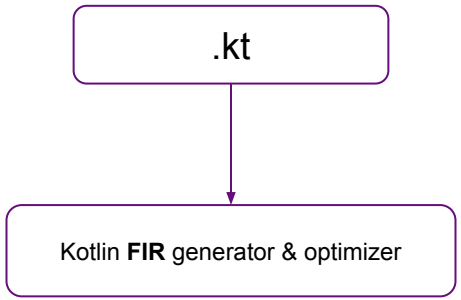
- We love **Java** and we use **Spring**
- But it will be great to have nice syntax sugar
- We want to learn how to write FE with familiar languages
- We want **one** programming language for everything
- We want to stay in a **mainstream**



WHY KOTLIN?

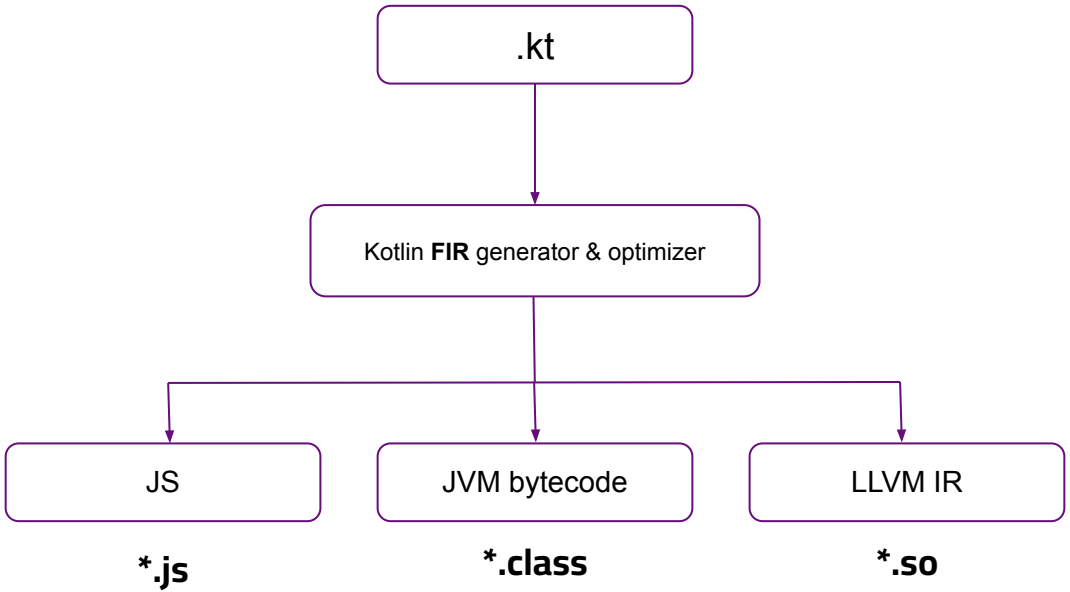
- We love **Java** and we use **Spring**
- But it will be great to have nice syntax sugar
- We want to learn how to write FE with familiar languages
- We want **one** programming language for everything
- We want to stay in a **mainstream**

HOW MPP WORKS (SIMPLIFIED)



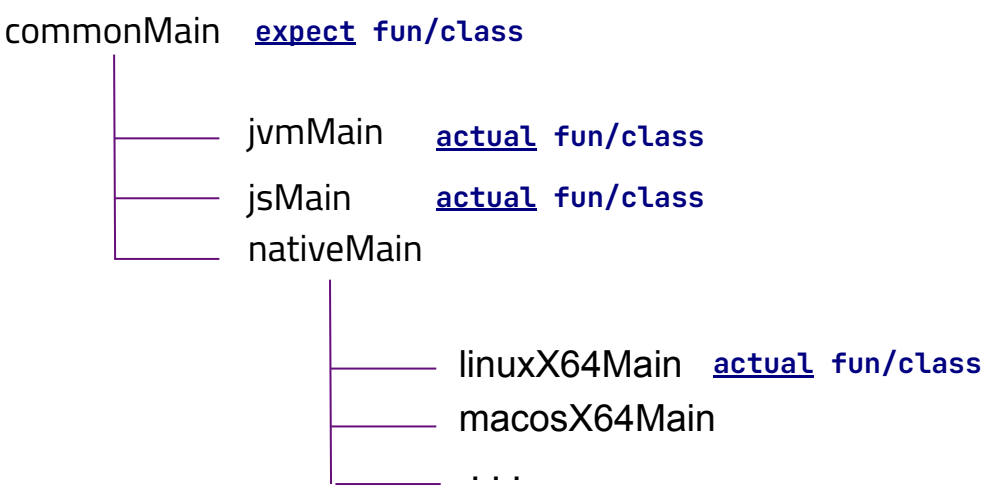
*perfect model from the future with K2

HOW MPP WORKS (SIMPLIFIED)

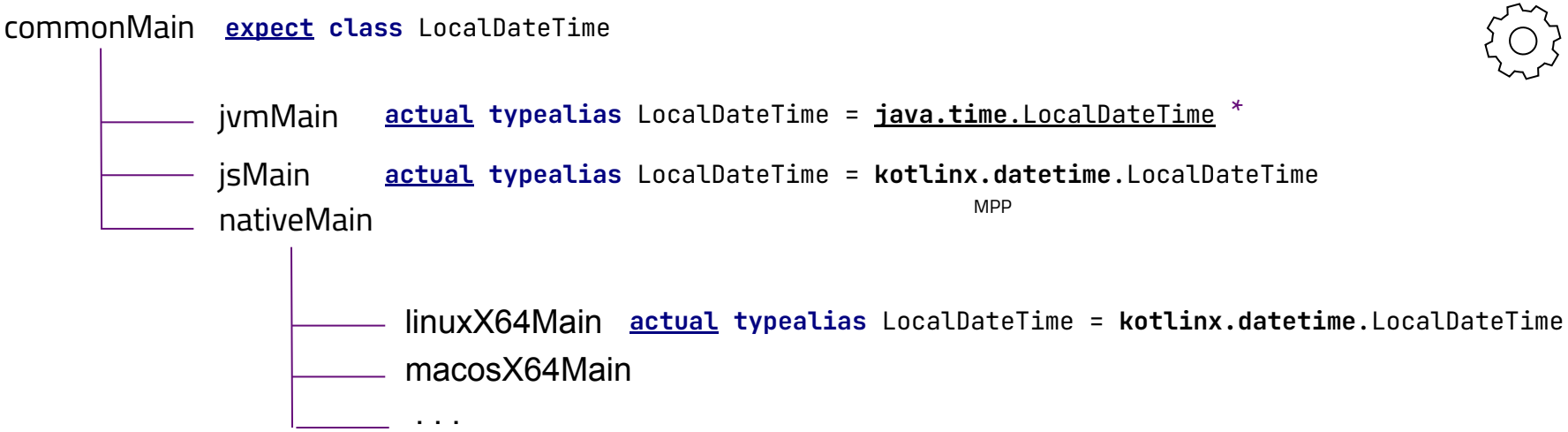


*perfect model from the future with K2

HOW MPP WORKS (SIMPLIFIED): EXPECT/ACTUAL

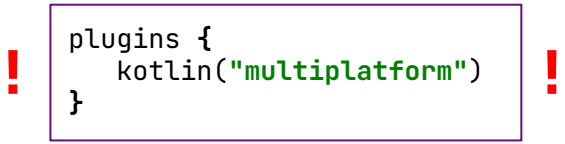


HOW MPP WORKS (SIMPLIFIED): EXPECT/ACTUAL



* (?) interview question

HOW MPP WORKS (SIMPLIFIED)



master ▾ save-cloud / save-cloud-common / src /

akuleshov7 Temp removal of auth service dependencies (#1401) ...

..

commonMain/kotlin/com/saveourtool/save	Move authorization logic into separate module (#1359)
jsMain/kotlin/com/saveourtool/save	Minor TestSuitesSourcesDisplayer improvements (#1380)
jvmMain/kotlin/com/saveourtool/save	Temp removal of auth service dependencies (#1401)
jvmTest/kotlin/com/saveourtool/save	Create LnkExecutionTestSuite table (#1286)
linuxX64Main/kotlin/com/saveourtool/save	Moving to saveourtool domain (omg) (#794)

MAIN THING THAT WE EXPECTED

COMMON, SHARED CODE

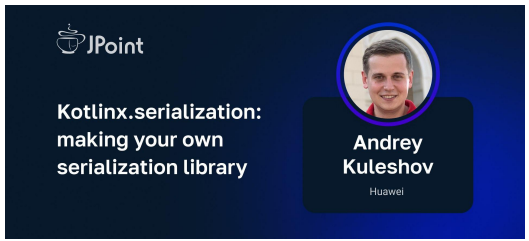
commonMain/

```
@Serializable
data class TestDto(
    . . .
)
```

COMMON, SHARED CODE

commonMain/

```
@Serializable
data class TestDto(
    . . .
)
```

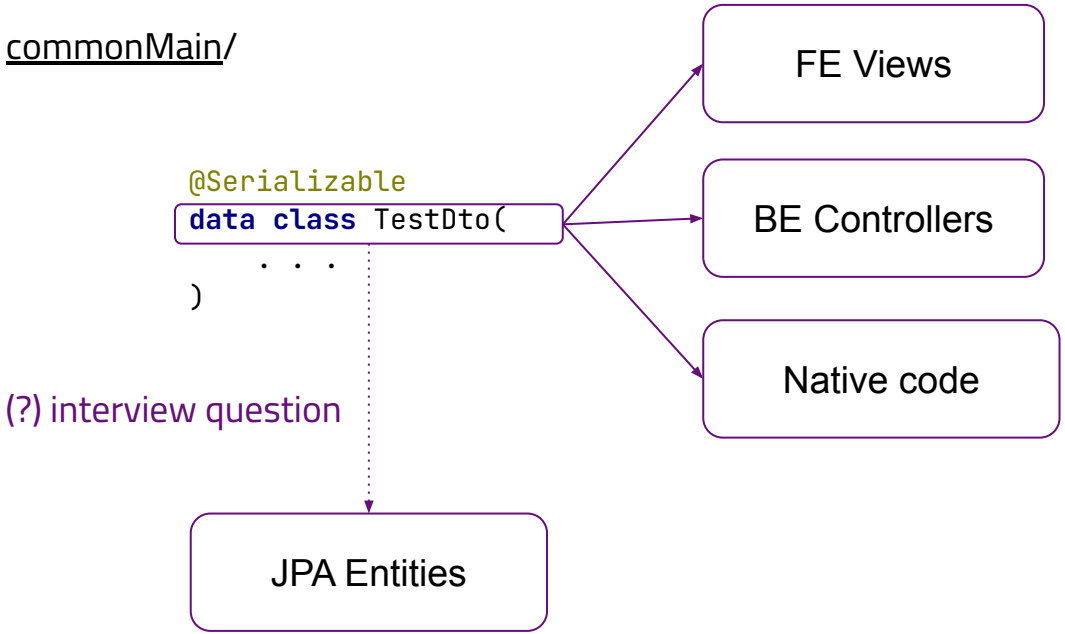


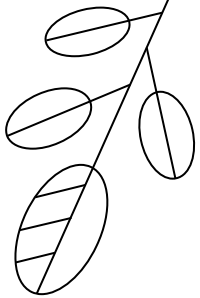
COMMON, SHARED CODE

commonMain/

```
@Serializable
data class TestDto(
    . . .
)
```

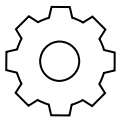
(?) interview question

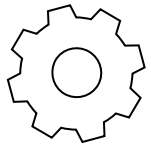




03 SPRING

HINTS AND TRICKS



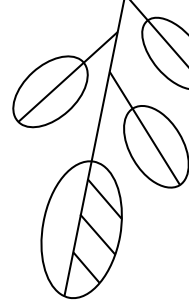



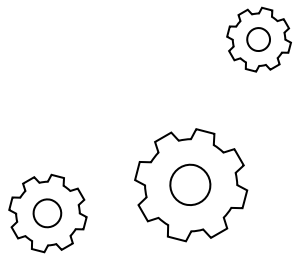
“JAVA IS THE OBVIOUS DEFAULT CHOICE,
BUT KOTLIN IS AN INCREASINGLY POPULAR ALTERNATIVE”

© SPRING.IO BLOGS, SÉBASTIEN DELEUZE


SYNTACTIC SUGAR

```
// even when we have records in Java 16/17
data class User(
    val name: String,
    val age: Int
)
```

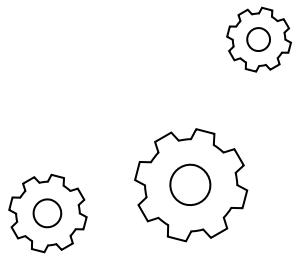




SYNTACTIC SUGAR

```
// even when we have records in Java 16/17
data class User(
    val name: String,
    val age: Int
)
```

```
@GetMapping(path = ["/api/$v1/execution"])
// nullability
fun getExecution(
   @RequestParam id: Long,
   authentication: Authentication?
)
```

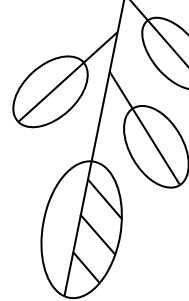


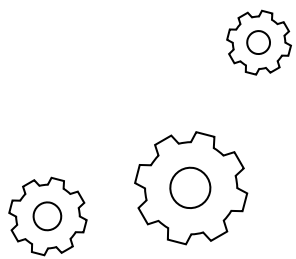
SYNTACTIC SUGAR

```
// even when we have records in Java 16/17
data class User(
    val name: String,
    val age: Int
)
```

```
@RestController
@RequestMapping(path = ["/api/$v1/contests"])
// constructor DI looks more convenient
public class ContestController(
   private val contestService: ContestService
)
```

```
@GetMapping(path = ["/api/$v1/execution"])
// nullability
fun getExecution(
   @RequestParam id: Long,
   authentication: Authentication?
)
```





SYNTACTIC SUGAR

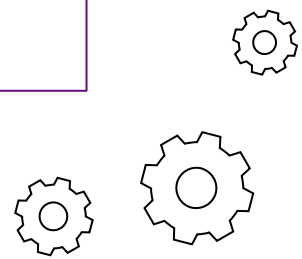
```
// even when we have records in Java 16/17
data class User(
    val name: String,
    val age: Int
)
```

```
@RestController
@RequestMapping(path = ["/api/$v1/contests"])
// constructor DI looks more convenient
public class ContestController(
   private val contestService: ContestService
)
```

```
@GetMapping(path = ["/api/$v1/execution"])
// nullability
fun getExecution(
   @RequestParam id: Long,
   authentication: Authentication?
)
```

```
@ConstructorBinding
@ConfigurationProperties(prefix = "gateway")
// configuration properties
data class ConfigurationProperties(
   val backend: Backend,
   val knownActuatorConsumers: String?,
)
```

And much more language-related features...



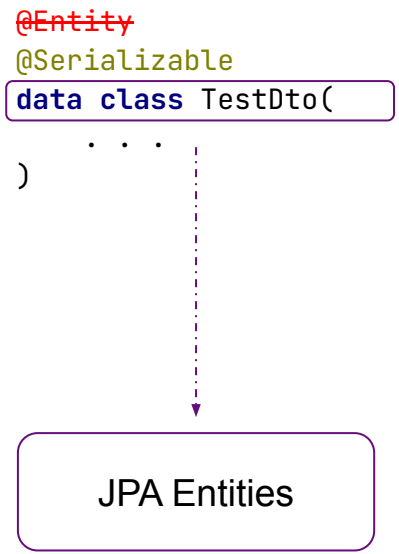
PROBLEMS: ALL OPEN PLUGIN

- Kotlin has classes and their members ***final*** by default
- But we know, that, for example, @Entity create proxy classes

SO:

```
plugins {
   kotlin("plugin.allopen")
}

allOpen {
   annotation("javax.persistence.Entity")
   . . .
}
```

PROBLEMS: ENTITY AND DATA CLASS



PROBLEMS: ENTITY AND DATA CLASS

```
@Entity
@Serializable
data class TestDto(
    . . .
)
```

JPA Entities

"Here we don't use data classes with val properties because JPA is not designed to work with immutable classes or the methods generated automatically by data classes."

https://github.com/spring-guides/tut-spring-boot-kotlin#persistence-with-jpa

PROBLEMS: ENTITY AND DATA CLASS

```
@Entity
@Serializable
data class TestDto(
    . . .
)
```

JPA Entities

"*Here we don't use data classes with val properties because JPA is not designed to work with immutable classes or the methods generated automatically by data classes.*"

https://github.com/spring-guides/tut-spring-boot-kotlin#persistence-with-jpa

All **@ToMany** associations are **lazy** by default

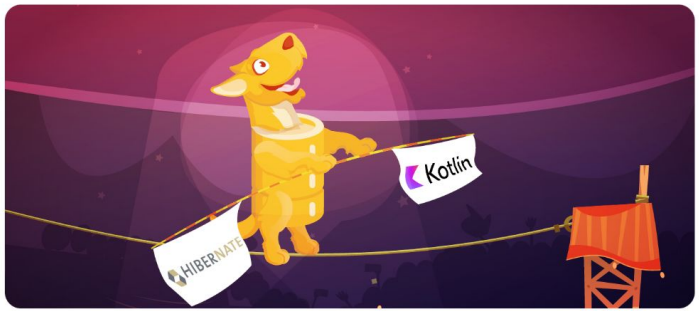
=> unwanted requests to the DB or a LazyInitializationException due to **generated methods**

jpabuddy

Get Features Documentation Blog Contact us

Andrey Oganesyan

Share:



JOIN US ON DISCORD

Subscribe

Facebook

Twitter

Youtube

24 June 2021

Best Practices and Common Pitfalls of Using JPA (Hibernate) with Kotlin

https://www.jpa-buddy.com/blog/best-practices-and-common-pitfalls/

SO SIMPLY CREATE BOTH DTO AND ENTITY

PROBLEM: JVM-ONLY CODE IN MP PROJECT

- But even we have a simple data class without any relations
- We have a new limitations, as @Entity is **not** present for example in JS

commonMain/

```
@Serializable
@Entity
data class TestDto(
    var i: Int
)
```

commonMain/

```
expect annotation class Entity()
```

jvmMain/

```
actual typealias Entity = Entity
```

jsMain/

(?)

PROBLEM: JVM-ONLY CODE IN MP PROJECT

- But even we have a simple data class without any relations
- We have a new limitations, as @Entity is **not** present for example in JS

commonMain/

```
@Serializable
@Entity
data class TestDto(
    var i: Int
)
```

commonMain/

```
@OptionalExpectation
expect annotation class Entity()
```

jvmMain/

```
actual typealias Entity = Entity
```

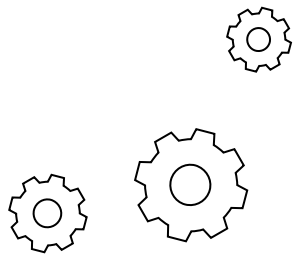
jsMain/

X

WEBFLUX AND COROUTINES

```
// controllers with coroutines and flow
@RestController
class UserController(private val userRepository: UserRepository) {
    @GetMapping("/")
    fun findAll(): Flow<User> =
        userRepository.findAll()

    @GetMapping("/{id}")
    suspend fun findOne(@PathVariable id: String): User? =
        userRepository.findOne(id) ?:
        throw CustomException("This user does not exist")
}
```

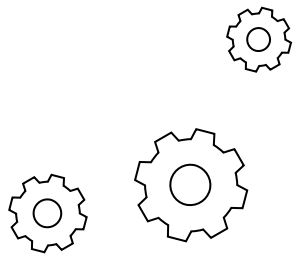
* We are using bare WebFlux with Mono and Flux, because we are idiots, don't repeat our mistakes, coroutines are 🔥



WEBFLUX AND COROUTINES

```
// controllers with coroutines and flow
@RestController
class UserController(private val userRepository: UserRepository) {
    @GetMapping("/")
    fun findAll(): Flow<User> =
        userRepository.findAll()

    @GetMapping("/{id}")
    suspend fun findOne(@PathVariable id: String): User? =
        userRepository.findOne(id) ?:
        throw CustomException("This user does not exist")
}
```

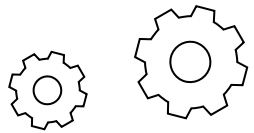
* We are using bare WebFlux with Mono and Flux, because we are idiots, don't repeat our mistakes, coroutines are 🔥



WEBFLUX AND COROUTINES

```
// controllers with coroutines and flow
@RestController
class UserController(private val userRepository: UserRepository) {
    @GetMapping("/")
    fun findAll(): Flow<User> =
        userRepository.findAll()

    @GetMapping("/{id}")
    suspend fun findOne(@PathVariable id: String): User? =
        userRepository.findOne(id) ?:
        throw CustomException("This user does not exist")
}
```

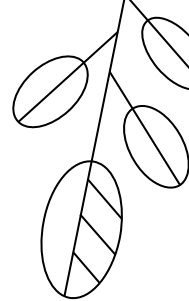
```
class UserRepository(
    private val client: DatabaseClient // R2DBC
) {
       fun findAll(): Flow<User> = client
              .select()
              .from("users")
              .asType<User>()
              .fetch()
              .flow() // asFlow()
}
```



* We are using bare WebFlux with Mono and Flux, because we are idiots, don't repeat our mistakes, coroutines are 🔥

WEBFLUX AND COROUTINES

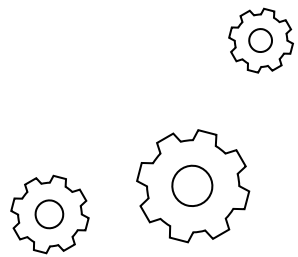
```
// WebClient.create(url)
client.get()
    .uri("/test")
    .accept(MediaType.APPLICATION_JSON)
    .retrieve()
    .bodyToFlow<String>()
```



WEBFLUX AND COROUTINES

```
// WebClient.create(url)
client.get()
   .uri("/test")
   .accept(MediaType.APPLICATION_JSON)
   .retrieve()
   .bodyToFlow<String>()
```

```
inline fun <reified T : Any> WebClient.ResponseSpec.bodyToFlow(): Flow<T> =
     bodyToFlux<T>().asFlow()
```

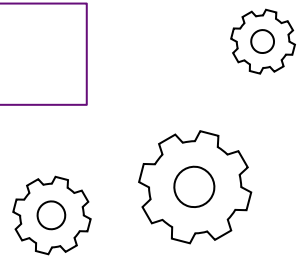


WEBFLUX AND COROUTINES

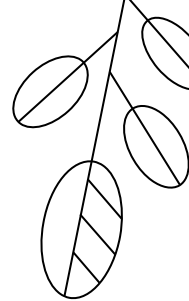
```
// WebClient.create(url)
client.get()
   .uri("/test")
   .accept(MediaType.APPLICATION_JSON)
   .retrieve()
   .bodyToFlow<String>()
```

```
inline fun <reified T : Any> WebClient.ResponseSpec.bodyToFlow(): Flow<T> =
     bodyToFlux<T>().asFlow()
```

```
public fun <T : Any> Publisher<T>.asFlow(): Flow<T> =
   PublisherAsFlow(this)
```

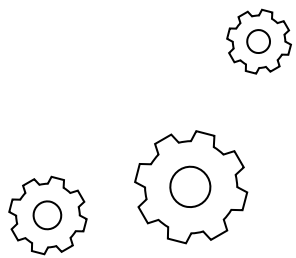


PROBLEMS: WEBFLUX AND COROUTINES

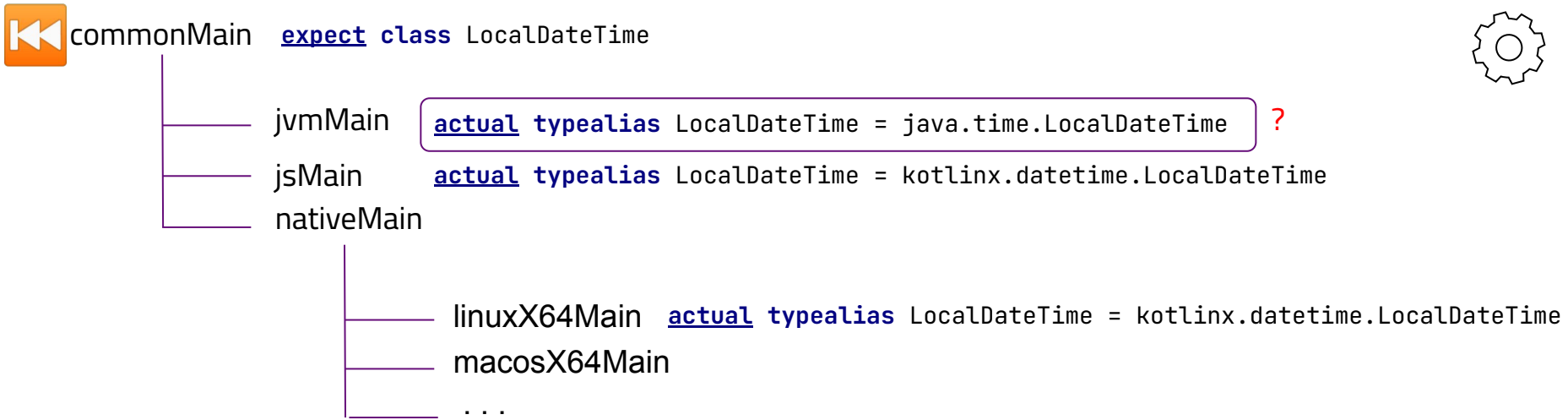


! org.jetbrains.kotlinx:kotlinx-coroutines-core
org.jetbrains.kotlinx:**kotlinx-coroutines-reactor**

Caused by: java.lang.**NoClassDefFoundError**: kotlinx/coroutines/reactor/**MonoKt**
at org.springframework.core.CoroutinesUtils.invokeSuspendingFunction(CoroutinesUtils.java:78)
at org.springframework.web.reactive.result.method.InvocableHandlerMethod.lambda$invoke$0(InvocableHandlerMethod.java:141)
at reactor.core.publisher.MonoFlatMap$FlatMapMain.onNext(MonoFlatMap.java:125)
at reactor.core.publisher.Operators$MonoSubscriber.complete(Operators.java:1816)
at reactor.core.publisher.MonoZip$ZipCoordinator.signal(MonoZip.java:251)



PROBLEMS: JVM VS COMMON



PROBLEMS: JVM VS COMMON

Hibernate knows only particular types and classes and has particular mappings, like `java.LocalDateTime`

Table 3.2. BasicTypes added by hibernate-java8

Hibernate type (org.hibernate.type package)	JDBC type	Java type	BasicTypeRegistry key(s)
DurationType	BIGINT	java.time.Duration	Duration, java.time.Duration
InstantType	TIMESTAMP	java.time.Instant	Instant, java.time.Instant
LocalDateTimeType	TIMESTAMP	java.time.LocalDateTime	LocalDateTime, java.time.LocalDateTime
LocalDateType	DATE	java.time.LocalDate	LocalDate, java.time.LocalDate
LocalTimeType	TIME	java.time.LocalTime	LocalTime, java.time.LocalTime
OffsetDateTimeType	TIMESTAMP	java.time.OffsetDateTime	OffsetDateTime, java.time.OffsetDateTime
OffsetTimeType	TIME	java.time.OffsetTime	OffsetTime, java.time.OffsetTime
OffsetTimeType	TIMESTAMP	java.time.ZonedDateTime	ZonedDateTime, java.time.ZonedDateTime

PROBLEMS: JVM VS COMMON

```
public actual class LocalDateTime
      internal constructor(internal val value: jtLocalDateTime) : Comparable<LocalDateTime> {

       public actual constructor(date: LocalDate, time: LocalTime) :
         this(jtLocalDateTime.of(date.value, time.value))

       public actual val minute: Int get() = value.minute
}
```

So, instead of making a custom mapping

Just use `expect/actual` with a typealias and use

TO SUM UP

Basically, in Kotlin MPP, JVM target is good enough for normal programming, we can treat it as Java with syntax sugar

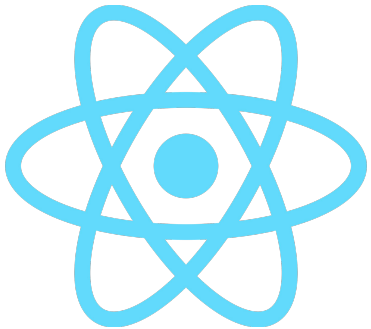
Be careful with **data classes**, use enhancements, like coroutines

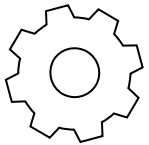
Treat it as **Java 30**

And this is mostly everything you need to know

04 REACT

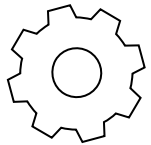
HINTS AND TRICKS



WHY KOTLIN AND NOT JAVA?





Because with Kotlin you can write a **Web UI**

Don’t say that with Java you can also make it

You can even copy-paste html and IDEA will generate KotlinJS code


STARTING WITH JS: IDEAL CASE

```
plugins {
    kotlin("js")
}

kotlin {
     js(LEGACY | IR | BOTH) {
         useCommonJs() // commonJS modules
     }

     // explicitly instructs the Kotlin compiler to emit executable .js files
     // default for LEGACY
     binaries.executable()
}
```

STARTING WITH JS: IDEAL CASE

```
plugins {
    kotlin("js")
}

kotlin {
    js(LEGACY | IR | BOTH) {
        useCommonJs() // commonJS modules
    }

    // explicitly instructs the Kotlin compiler to emit executable .js files
    // default for LEGACY
    binaries.executable()
}
```

STARTING WITH JS: IDEAL CASE

```
plugins {
    kotlin("js")
}

kotlin {
     js(LEGACY | IR | BOTH) {
         useCommonJs() // commonJS modules
     }

    // explicitly instructs the Kotlin compiler to emit executable .js files
    // default for LEGACY
    binaries.executable()
}
```

STARTING WITH JS: IDEAL CASE

```
plugins {
    kotlin("js")
}

kotlin {
     js(LEGACY | IR | BOTH) {
         useCommonJs() // commonJS modules
     }

     // explicitly instructs the Kotlin compiler to emit executable .js files
     // default for LEGACY
     binaries.executable()
}
```

```
$ ./gradlew run --continuous
```

BUT OF COURSE...

```
private suspend fun <T> getAndUpdateState(url: String, setState: (Set<T>) → Unit)
{
   it.decodeFromJsonString<List<T>>()
}
```

cause_th0jdv$_0: null

message_8yp7un$_0: **"Captured type paramerer T from generic non-reified function.
Such functionality cannot be supported as T is erased, either specif…"**

name: "IllegalStateException"

stack: "captureStack@↵Exception@↵RuntimeException@↵
IllegalStateException@↵IllegalStateException_init_0@↵
kclass@↵serializerByKTypeImpl…"

IllegalStateException Prototype

BUT OF COURSE...

https://github.com/Kotlin/kotlinx.serialization/issues/1448

sandwwraith commented on May 4, 2021 Member

`serializer<T>()` function doesn't work in LEGACY JS in many cases, and legacy backend would be eventually deprecated anyway

BUT OF COURSE...

https://github.com/Kotlin/kotlinx.serialization/issues/1448

sandwwraith commented on May 4, 2021 — Member

`serializer<T>()` function doesn't work in LEGACY JS in many cases, and legacy backend would be eventually deprecated anyway

BTW:

> ⚠ The Kotlin/JS IR compiler is in Beta. It is almost stable, but migration steps may be required in the future. We'll do our best to minimize any changes you have to make.

DEPENDENCIES

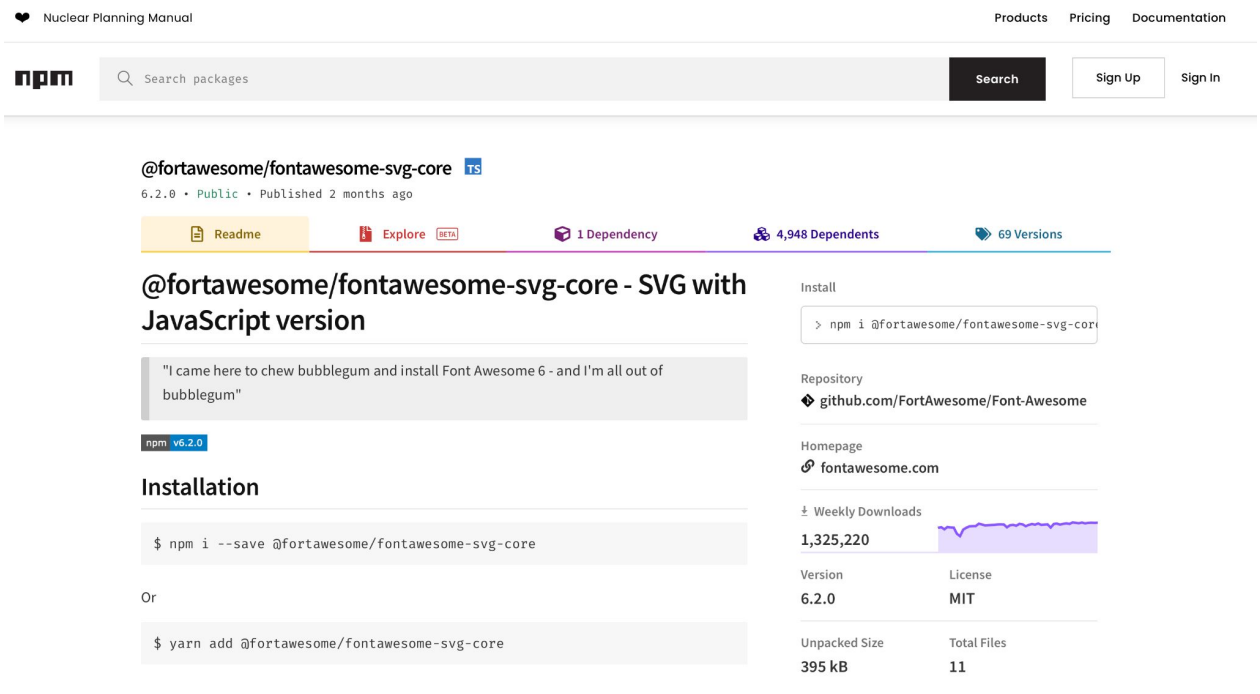
devNpm optionalNpm peerNpm

```
implementation(npm("@fortawesome/fontawesome-svg-core", "^1.2.36"))
```

DEPENDENCIES

devNpm optionalNpm peerNpm

```
implementation(npm("@fortawesome/fontawesome-svg-core", "^1.2.36"))
```



EXTERNAL LIBRARIES

```kotlin
// kotlinJS

@file:JsModule("react")
@file:JsNonModule

@JsName("useState")
external fun <T> useFunctionState(
   initialState: T,
): FunctionStateInstance<T>
```

```kotlin
// JS

export function useState<S>(
 initialState: (() ⇒ S) | S,
): [S, Dispatch<BasicStateAction<S>>]
```

AWESOME THINGS IN KOTLIN JS: DUKAT

https://github.com/Kotlin/dukat

“Converter of <any kind of declarations>
to Kotlin external declarations” (c)

AWESOME THINGS IN KOTLIN JS: DUKAT

```
$ ./gradlew generateExternals
```

```
> Task :save-frontend:generateExternals
Exception in thread "main" java.lang.StackOverflowError
        at kotlin.collections.CollectionsKt___CollectionsKt.joinTo(_Collections.kt:3299)
        at kotlin.collections.CollectionsKt___CollectionsKt.joinToString(_Collections.kt:3321)
```

KOTLIN-WRAPPERS

https://github.com/JetBrains/kotlin-wrappers

README	Note	Version
kotlin-browser		maven-central v0.0.1-pre.429
kotlin-cesium		maven-central v1.99.0-pre.429
kotlin-css		maven-central v1.0.0-pre.429
kotlin-csstype		maven-central v3.1.1-pre.429
kotlin-emotion		maven-central v11.10.5-pre.429
kotlin-extensions	archived	maven-central v1.0.1-pre.429
kotlin-history		maven-central v5.3.0-pre.429
kotlin-js		maven-central v1.0.0-pre.429
kotlin-mui		maven-central v5.9.1-pre.429
kotlin-mui-icons		maven-central v5.10.9-pre.429
kotlin-node		maven-central v18.11.9-pre.429
kotlin-popper		maven-central v2.11.6-pre.429
kotlin-react		maven-central v18.2.0-pre.429
kotlin-react-beautiful-dnd		maven-central v13.1.0-pre.429
kotlin-react-core		maven-central v18.2.0-pre.429
kotlin-react-dom		maven-central v18.2.0-pre.429
kotlin-react-dom-legacy		maven-central v18.2.0-pre.429
kotlin-react-legacy		maven-central v18.2.0-pre.429
kotlin-react-redux		maven-central v7.2.6-pre.429

HOW REACT-LIKE SYNTAX LOOKS LIKE

```
div {
    className = ClassName("row justify-content-center")
    h1 {
        className = ClassName("text-dark")
        +"Hello world"
    }
}
```

HOW REACT-LIKE SYNTAX LOOKS LIKE

```
myProjects.forEach {
    div {
        className = ClassName("row justify-content-center align-items-center")
        h4 {
            +(it.contestRating.toFixed(2).toString())
        }
    }
    div {
        className = ClassName("row justify-content-center align-items-center")
        p {
            +it.name
        }
    }
}
```

DSL CHURCH SPLIT

https://github.com/JetBrains/kotlin-wrappers/blob/master/CHANGELOG.md#pre282

kotlin-react-legacy

RBuilder

```
div("col") {
    attrs {
        title = "hello"
    }
}
```

kotlin-react

ChildrenBuilder

```
div {
    className = "col"
    title = "hello"
}
```

HOW REACT WORKS WITH KOTLIN

```
val App = VFC {
   val (tests, setTests) = useState<List<TestDto>>(emptyList())

    // your proper wrapper over 'useEffect' with coroutines handler
    useRequest {
        val updatedTests: List<TestDto> = /* myGetRequest */
            .decodeFromJsonString()

        setTests(updatedTests)
    }

    customRendering(tests)
}
```

HOW REACT WORKS WITH KOTLIN

```
val App = VFC {
    val (tests, setTests) = useState<List<TestDto>>(emptyList())

    // your proper wrapper over 'useEffect' with coroutines handler
    useRequest {
        val updatedTests: List<TestDto> = /* myGetRequest */
            .decodeFromJsonString()

        setTests(updatedTests)
    }

    customRendering(tests)
}
```

HOW REACT WORKS WITH KOTLIN

```
val App = VFC {
   val (tests, setTests) = useState<List<TestDto>>(emptyList())

   // your proper wrapper over 'useEffect' with coroutines handler
   useRequest {
       val updatedTests: List<TestDto> = /* myGetRequest */
           .decodeFromJsonString()

       setTests(updatedTests)
   }

   customRendering(tests)
}
```

HOW REACT WORKS WITH KOTLIN

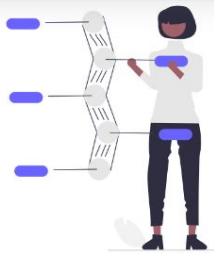
```
val App = VFC {
   val (tests, setTests) = useState<List<TestDto>>(emptyList())

   // your proper wrapper over 'useEffect' with coroutines handler
   useRequest {
       val updatedTests: List<TestDto> = /* myGetRequest */
           .decodeFromJsonString()

       setTests(updatedTests)
   }

   customRendering(tests)
}
```

```
// don't call Hooks inside loops, conditions, or nested functions

private fun ChildrenBuilder.customRendering(
    tests: List<TestDto>
) {
     tests.forEach {
         h1 {
              +it.name
         }
     }
}
```

HOW REACT WORKS WITH KOTLIN

```
val App = VFC {
   var tests by useState<List<TestDto>>(emptyList())

   // your proper wrapper over 'useEffect' with coroutines handler
   useRequest {
       val updatedTests: List<TestDto> = /* myGetRequest */
           .decodeFromJsonString()

       tests = updatedTests
   }

   customRendering(tests)
}
```

```
// don't call Hooks inside loops, conditions, or nested functions

private fun ChildrenBuilder.customRendering(
    tests: List<TestDto>
) {
     tests.forEach {
         h1 {
              +it.name
         }
     }
}
```



Total Benchmarks:

26

Checkout updates and new benchmarks.

Check the GitHub

News

SAVE

Checkout latest updates in SAVE project.

SAVE-cloud
SAVE-cli

Search for the benchmark...

Awesome Benchmarks Archive

ALL | AI | AUDIT | CODING_STANDARD | PERFORMANCE | STATIC_ANALYSIS

CodeXGLUE
CodeXGLUE includes a collection of code intelligence tasks and a platform for model evaluation and comparison. It stands for General Language Understanding Evaluation benchmark for CODE.

#java #toolkit

Docs | Sources | More →

Juliet Test Suite for C#
A collection of test cases in the C# language. It contains examples organized under 105 different CWEs.

#static_analysis #juliet #security

Docs | Sources | More →

kotlinx
kotlinx.benchmark is a toolkit for running benchmarks for multiplatform code written in Kotlin and running on the next supported targets: JVM,

java	5
c#	1
kotlin	2
php	1
Clojure	1
python	1
c	4
c/c++	2

Purpose of this list

As a group of enthusiasts who create dev-tools (including static analysis tools), we have seen a lack of materials related to testing scenarios or benchmarks that can be used to evaluate and test our applications. So we decided to create this curated list of standards, tests and benchmarks that can be used for testing and evaluating dev tools. Our focus is mainly on the code analysis, but is not limited by this category, in this list we are trying to collect all benchmarks that could be useful for creators of dev-tools.

Easy contribution steps

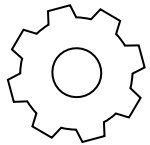
1. Go to the awesome-benchmarks repository
2. Create a fork to your account
3. Create the description in a proper format
4. Add your benchmark to benchmarks dir
5. Validate the format with `./gradlew build`
6. Create the PR to the main repo

ПАРА-ПАРА-ПАМ!

05 CONCLUSION

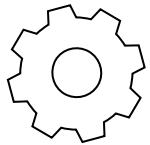
EXTRA THINGS

CONCLUSION



- **Kotlin - is a great language** that gives us awesome user experience

CONCLUSION



- **Kotlin - is a great language** that gives us awesome user experience

- Finally JAVA backend developers **can write a frontend** without ugly frameworks

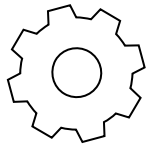
CONCLUSION

- **Kotlin - is a great language** that gives us awesome user experience
- Finally JAVA backend developers **can write a frontend** without ugly frameworks
- Common code can be shared between backend and frontend

CONCLUSION

- **Kotlin - is a great language** that gives us awesome user experience
- Finally JAVA backend developers **can write a frontend** without ugly frameworks
- Common code can be shared between backend and frontend
- But there are some compatibility problems that we should be aware about

CONCLUSION



- **Kotlin - is a great language** that gives us awesome user experience
- Finally JAVA backend developers **can write a frontend** without ugly frameworks
- Common code can be shared between backend and frontend
- But there are some compatibility problems that we should be aware about
- So let’s help the community to grow and improve!

THANK YOU FOR LISTENING!

https://github.com/**akuleshov7**
https://github.com/**saveourtool**