GOPAS

Java Days

9. - 10. 11. 2022

Praha / ONLINE

Konference

pro Java vývojáře

Jak testujete webové aplikace?

Víťa Plšek | JavaDays 2022

@vitaplsek

Víťa Plšek

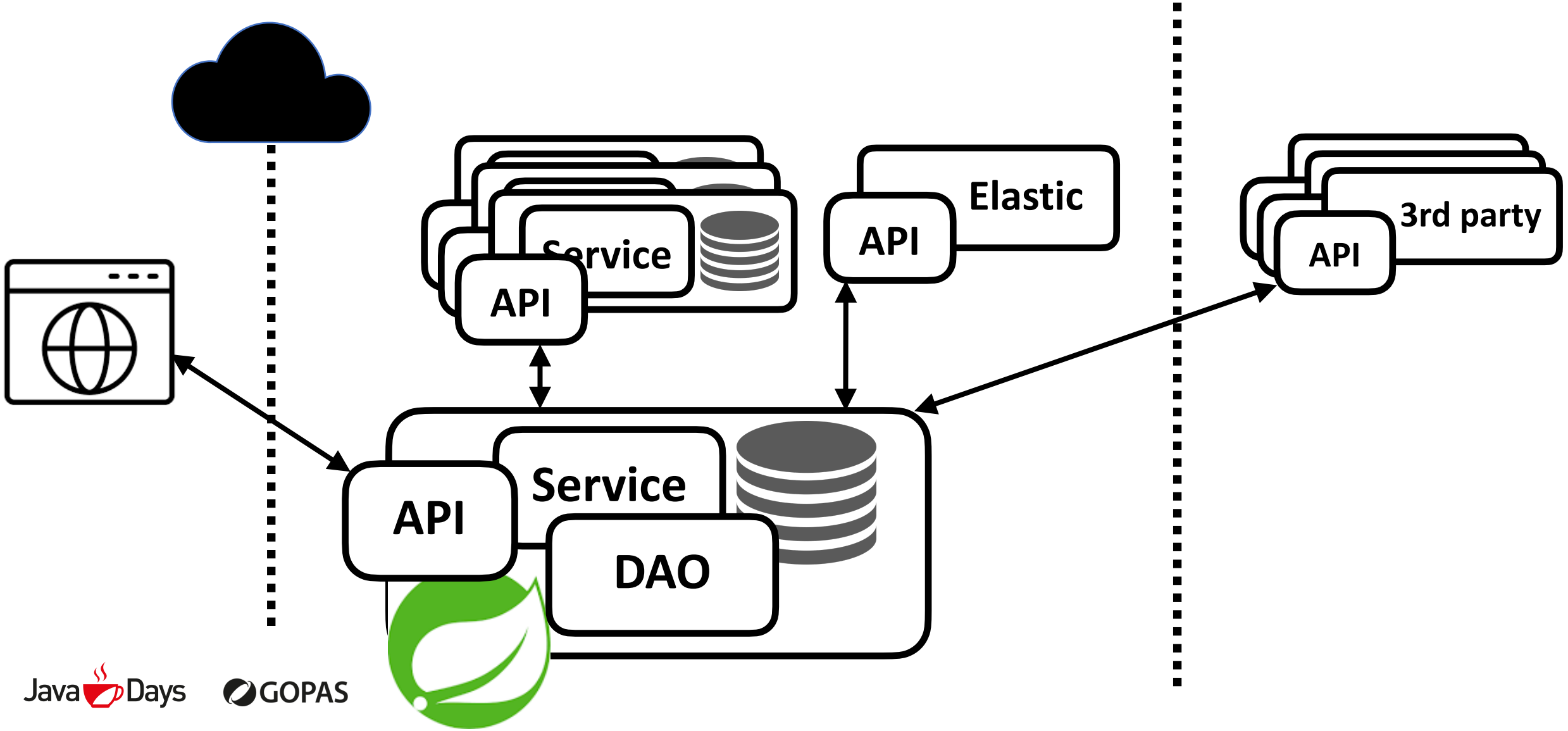
MOROSYSTEMS
PROGRESS THROUGH COOPERATION

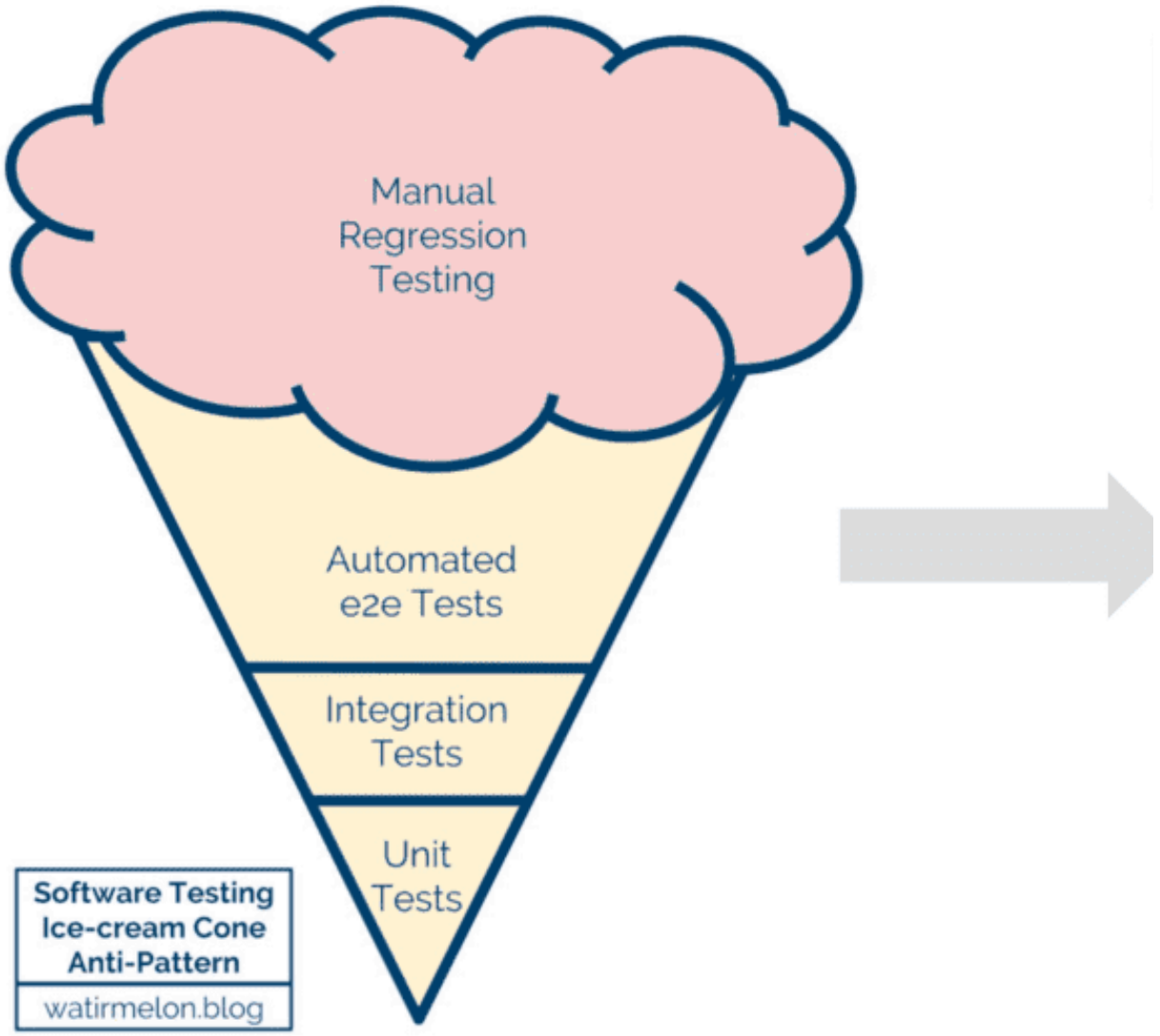
Generální partner

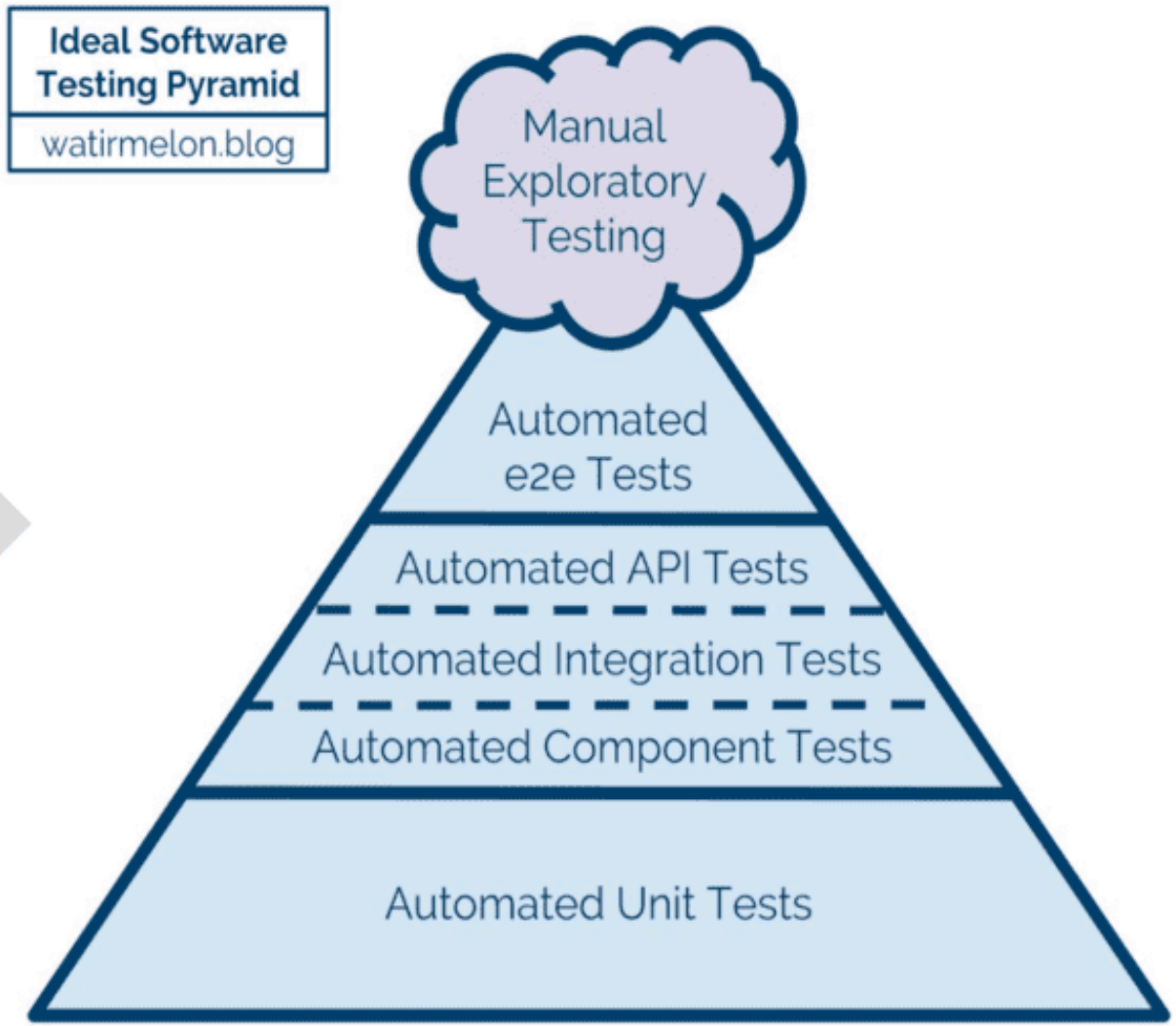
COMMERZBANK

JavaDays GOPAS

Jak vypadá typická webová aplikace




Manual
Regression
Testing
Automated
e2e Tests
Integration
Tests
Unit
Tests
Software Testing
Ice-cream Cone
Anti-Pattern
watirmelon.blog


Ideal Software Testing Pyramid
watirmelon.blog
Manual Exploratory Testing
Automated e2e Tests
Automated API Tests
Automated Integration Tests
Automated Component Tests
Automated Unit Tests

Co testuju já ?

E2E

rest api - akceptační kritéria

Integrační

databázové dotazy
komunikace se službami

Unit testy

validátory, algoritmy

Akceptační kritéria od analytika

- Je možné přidat, upravit a smazat pobočku zákazníka
- Deaktivace zákazníka zneaktivní jeho pobočky
- Nelze smazat zákazníka, který má pobočky
- Výpis je seřazen podle abecedy a lze je stránkovat

Co testuju já ?

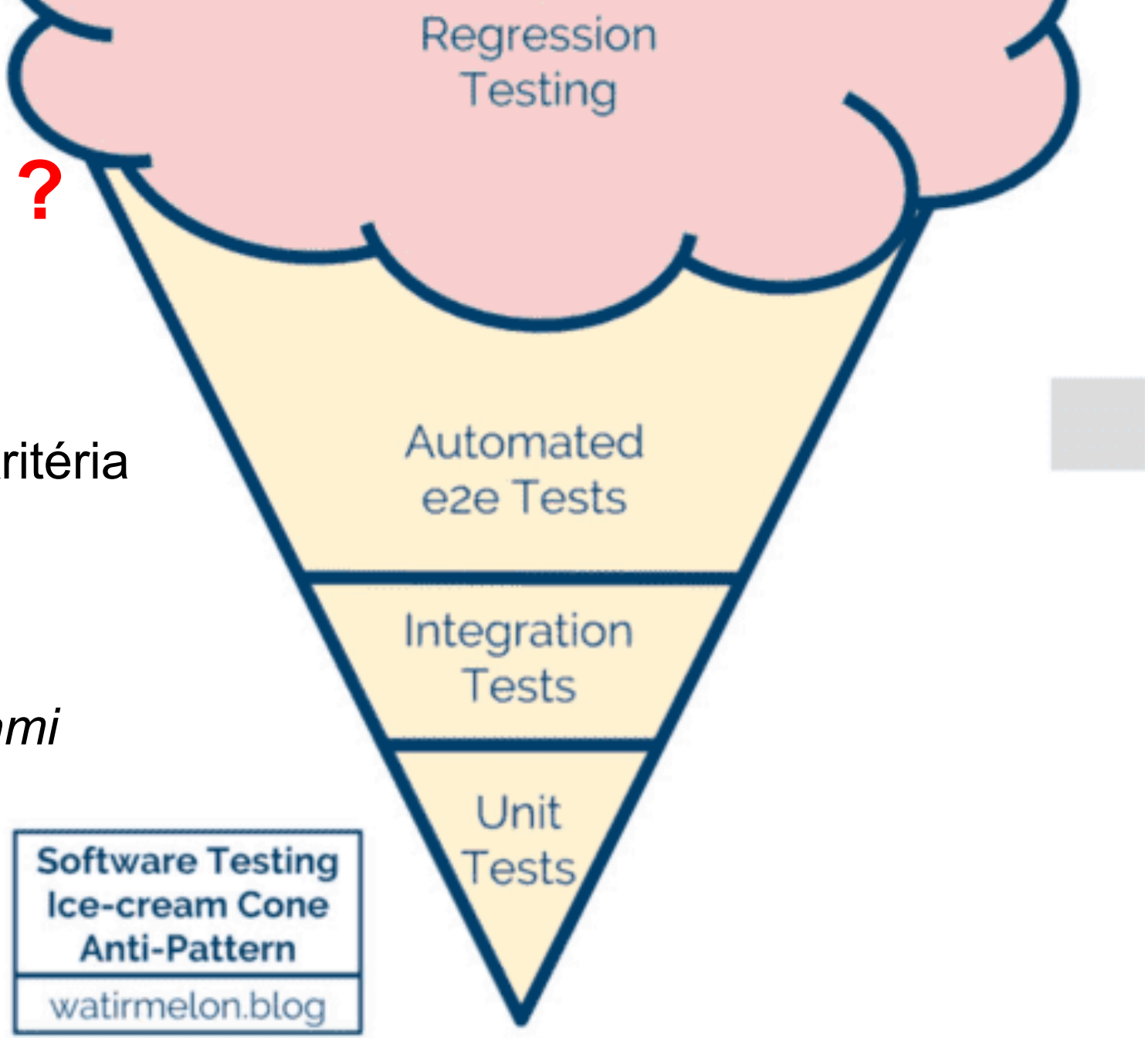
E2E

rest api - akceptační kritéria

Integrační

databázové dotazy
komunikace se službami

Unit testy
validátory, algoritmy



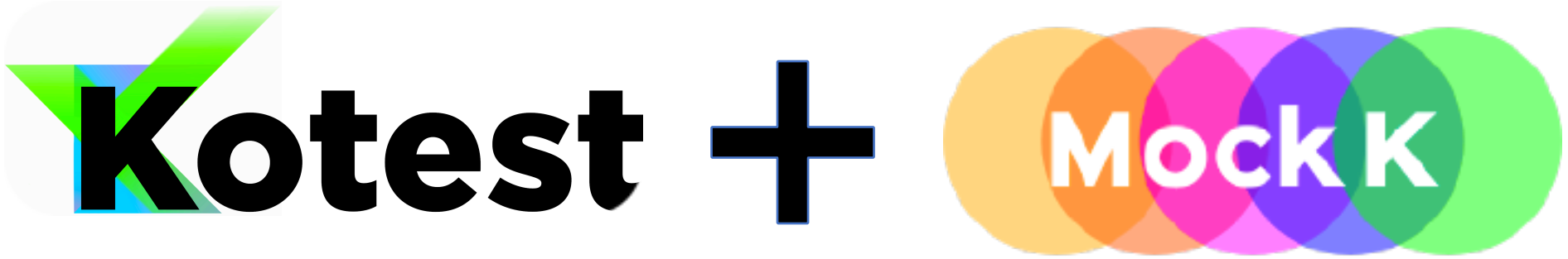
“Pokud se něco osvědčilo, dělejte to neustále”

Kent Beck - Extrémní programování

Konzervativní toolset Java BE vývojáře

JUnit + mockito + PowerMock + AssertJ

Toolset Kotlin vývojáře



Co dalšího nám může pomoct?

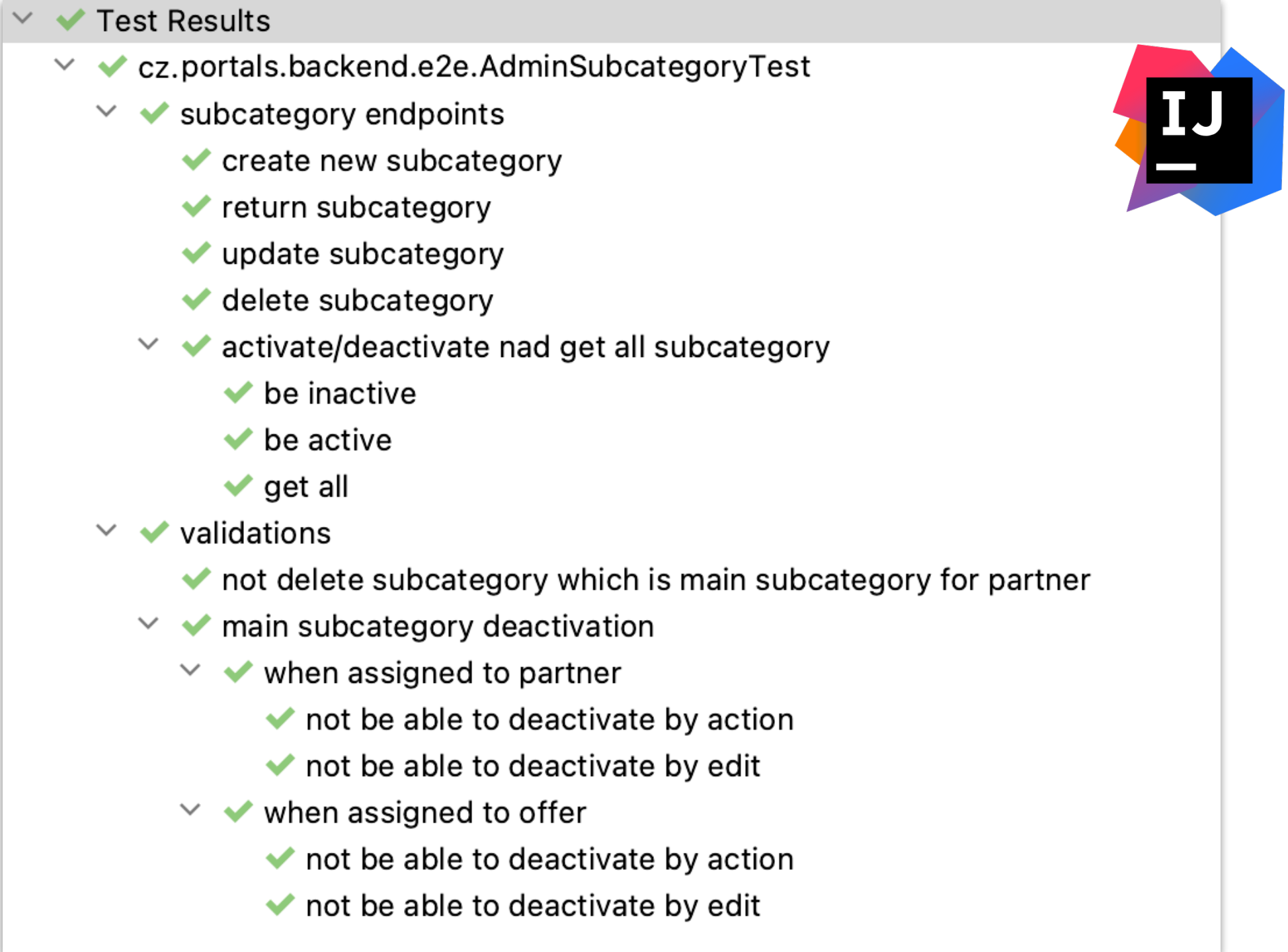
WIREMOCK

Kotest


```
class AlgorithmTests {
    @Test
    fun `count the result of algorithm`() {
        // asserts here
    }
}
```

```
class AlgorithmTests : ShouldSpec({
    should("count the result of algorithm") {
        // asserts here
    }
})
```

```
class SubcategoryTest() : ShouldSpec() {

    context("subcategory endpoints") {

        // příprava obecných dat
        val subcategoryForCreate = createSubcategoryDto()

        context("when creating new subcategory") {

            // další příprava, nebo provedení akce
            val createdSubcategory = restTemplate.createSubcategory(subcategoryForCreate)

            should("create basic info") {
                // samotný test / asserty
                createdSubcategory.id shouldNotBe null
                createdSubcategory.name shouldBe subcategoryForCreate.name
                createdSubcategory.active shouldBe true
            }

            should("contain default image") { ... }

            context("validations") {
                should("not allow empty name") { ... }
                ...
            }
        }
```


Test Results
cz.portals.backend.e2e.AdminSubcategoryTest
subcategory endpoints
create new subcategory
return subcategory
update subcategory
delete subcategory
activate/deactivate nad get all subcategory
be inactive
be active
get all
validations
not delete subcategory which is main subcategory for partner
main subcategory deactivation
when assigned to partner
not be able to deactivate by action
not be able to deactivate by edit
when assigned to offer
not be able to deactivate by action
not be able to deactivate by edit

Matchery

```
infix fun <T> Collection<T>.shouldHaveSize(size: Int): Collection<T> {
    this should haveSize(size = size)
    return this
}
```

```
createdSubcategory.id.shouldNotBe(null)
createdSubcategory.id shouldNotBe null

createdSubcategory.name shouldBe subcategoryForCreate.name
createdSubcategory.active shouldBe true
createdSubcategory.labels shouldContain (Labels.NEW)

partners shouldHaveSize 1
partners.shouldNotContainDuplicates()
partners shouldExist { it.id == partner.id }
```

Podpora springu

```
class MyTests(val myService: MyService) : ShouldSpec() {
    override fun extensions() = listOf(SpringExtension)

    @Autowired
    lateinit var myService2: MySecondService

    init {
        should("do something") {
            myService.answer(myService2.question) shouldBe 42
        }
    }
}
```


MockK

Mock, stub, verify …

```
val container = mockk<ValueContainer>()

every { container.getValue() } returns 41

increaseValue(container) shouldBe 42

verify { container.getValue() }
```

```
every { container.getNextValue(anyInt()) } answers { firstArg<Int>() + 2 }
container.getNextValue( anyInt: 40) shouldBe 42
```

```
val product: Product = mockk()
every { product.currentOffer.isAcive } returns true
```

Co jinde nenamokujete

```
fun nextValue(value : Int) = value + 1
```

```
mockkStatic(::nextValue)
every { nextValue(anyInt()) } answers { firstArg<Int>() + 2 }

nextValue( value: 0) shouldBe 2
```

- Funkce
- Extension funkce, i vnořené
- Objekty, statické vlastnosti
- Třídy, konstruktory

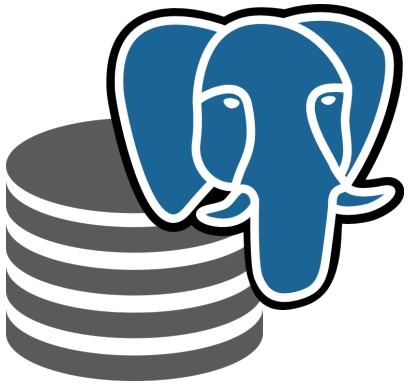
Nerad píšu kopie implementace

```
val pdfService: PdfService = mockk()
val emailService: EmailService = mockk()
val invoiceService = InvoiceService(pdfService, emailService)

val invoice = createTestInvoice(email = "jan@novak.cz")
val pdf = mockk<Pdf>()

every { pdfService.generate(invoice) } returns pdf

invoiceService.sendInvoice(invoice)

verify { emailService.send("jan@novak.cz", pdf) }
```

```
class InvoiceService(
    val pdfService: PdfService,
    val emailService: EmailService,
) {
    fun sendInvoice(invoice: Invoice) {
        val pdf = pdfService.generate(invoice)
        emailService.send(invoice.customerEmail, pdf)
    }
}
```

Co naopak často testuju

- DAO
  - Mapování zanořených entit do DTO
- Dotazy do db
  - “Získej všechny nabídky, které mají právě jednu aktivní podkategorii”
  - “Smaž všechny logy starší než 30 dní, které nikdo neotevřel”




Testcontainers

Spuštění kontejneru - v kódu

```
private const val POSTGRES_IMAGE_NAME = "postgres:13.8"

val postgresSQLContainer = PostgreSQLContainer<Nothing>("postgres:13.8")
postgresSQLContainer.start()

postgresSQLContainer.host
postgresSQLContainer.firstMappedPort

postgresSQLContainer.getJdbcUrl()
postgresSQLContainer.username
postgresSQLContainer.password
```

Spuštění kontejneru - pomocí driveru

```
spring:
    datasource:
        driver-class-name: "org.testcontainers.jdbc.ContainerDatabaseDriver"
        url: "jdbc:tc:postgresql:13.8://localhost:45432/postgres"
        username: postgres
        password: postgres
```

E2E Testy

REST API

Spring podpora

```
@SpringBootTest
class E2ETest(
    val mockMvc: MockMvc,
    val restTemplate: TestRestTemplate,
    val webTestClient: WebTestClient
) : ShouldSpec() {

    ...
}
```

WebTestClient

```
webTestClient
    .post()
    .uri("/admin/partner")
    .bodyValue(partnerCreateDto)
    .exchange()
    .expectStatus()
    .is2xxSuccessful
    .expectBody(PartnerDto::class.java)
    .returnResult()
    .responseBody!!
```

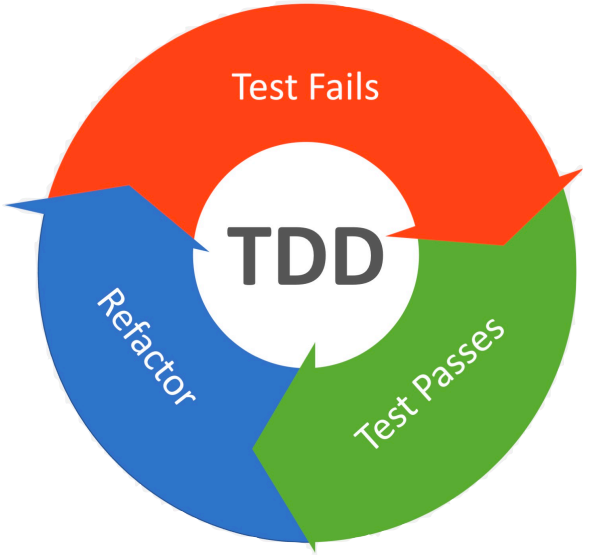
Mockování služeb

- @MockkBean - SpringMockK

```
@MockkBean
private lateinit var greetingService: GreetingService
...
every { greetingService.greet("John") } returns "Hi John"
```

- Vytvoření “dvojníka” na úrovni http volání

TDD - Test driven development

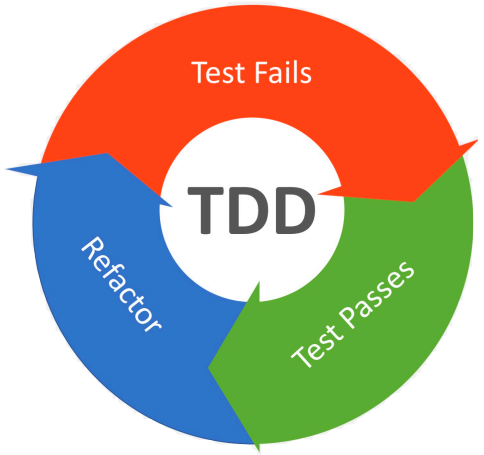


Vývoj BE jak jsem to dělával

- Návrh databáze
- Service / dao / logika <- spousta místa pro “to se může hodit”
- REST rozhraní
- Průběžně ladit *Curlem / Postmanem*
- Napsat testy, protože bych měl


“Mám h to v ještě opr ím testy”

Jak vypadá vývoj v rámci TDD



- Akceptační kritéria od analytika

- Testy
- REST rozhraní
- Logika
- Service / Dao / Databáze

Jak to dělat, aby Vás to bavilo


LEGO
NEXO KNIGHTS

Zapouzdřete si volání API

```
webTestClient
    .post()
    .uri("/admin/partner")
    .bodyValue(partnerCreateDto)
    .exchange()
    .expectStatus()
    .is2xxSuccessful
    .expectBody(PartnerDto::class.java)
    .returnResult()
    .responseBody!!
```

Zapouzdřete si volání API

```
webTestClient
    .post()
    .uri("/admin/partner")
    .bodyValue(partnerCreateDto)
    .exchange()
    .expectStatus()
    .is2xxSuccessful
    .expectBody(PartnerDto::class.java)
    .returnResult()
    .responseBody!!
```

Zapouzdřete si volání API

```
webTestClient
    .post()
    .uri("/admin/partner")
    .bodyValue(partnerCreateDto)
    .is2xxSuccessful
    .returnBody(PartnerDto::class.java)
```

Zapouzdřete si volání API

```
val createdPartner = webTestClient.createPartner(partnerCreateDto)
```

```
fun WebTestClient.createPartner(partnerCreateDto : CreatePartnerDto) =
    webTestClient
        .post()
        .uri("/admin/partner")
        .bodyValue(partnerCreateDto)
        .is2xxSuccessful
        .returnBody(PartnerDto::class.java)
```

Zapouzdřete si volání API

```
val createdPartner = webTestClient.createPartner(partnerCreateDto)
```

```
val errors = webTestClient.createPartnerForErrors(prepareCreatePartnerDto())
```

Připravte si potřebná data

- **Jak ?**
  - Pomocí REST volání
  - Přímo do DB

- **Kdy?**
  - ~~Jednou před všemi testy~~
  - Před testovací třídou
  - **Před jednotlivými testy**

Příprava dat pomocí REST

```
val createPartnerDto = newCreatePartnerDto(name = "Partner 1")
val createdPartner = webTestClient.createPartner(createPartnerDto)
```

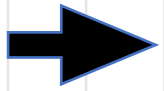
```
fun newCreatePartnerDto(name = "Partner_" + Random.nextInt()) =
    CreatePartnerDto(
        name = name,
        active = true,
        lastModified = OffsetDateTime.now()
)
```

Testy můžou navazovat …

```kotlin
context("Partner endpoint") {

1   val createdPartner = webTestClient.createPartner(prepareCreatePartnerDto())

2   should("create partner with all fields") {
        // test created partner
    }

3   should("update partner"){
        val updatedPartner = createdPartner.copy(name = "updated name" ...)
        // test updated partner
    }

4   should("deactivate partner") { ... }

5   should("reactivate partner") { ... }

6   should("delete partner") { ... }

7   should("not get deleted partner") { ... }

override fun isolationMode() = IsolationMode.SingleInstance
```

… ale nemusí

```
17  context("subcategory endpoints") {
18
19      // příprava obecných dat
20      val subcategoryForCreate = createSubcategoryDto()
21
22      context("when creating new subcategory") {
23
24          // další příprava, nebo provedení akce
25          val createdSubcategory = restTemplate.createSubcategory(subcategoryForCreate)
26
27          should("create basic info") {
28              // samotný test / asserty
29              createdSubcategory.id shouldNotBe null
30              createdSubcategory.name shouldBe subcategoryForCreate.name
31              createdSubcategory.active shouldBe true
32          }
33
34          should("contain default image") { ... }
35
36          context("validations") {
37              should("not allow empty name") { ... }
38              ...
39          }
40      }
41  }
42
```



```
override fun isolationMode() = IsolationMode.InstancePerLeaf
```

```
17    context("subcategory endpoints") {
18
19        // příprava obecných dat
20        val subcategoryForCreate = createSubcategoryDto()
21
22        context("when creating new subcategory") {
23
24            // další příprava, nebo provedení akce
25            val createdSubcategory = restTemplate.createSubcategory(subcategoryForCreate)
26
27            should("create basic info") {
28                // samotný test / asserty
29                createdSubcategory.id shouldNotBe null
30                createdSubcategory.name shouldBe subcategoryForCreate.name
31                createdSubcategory.active shouldBe true
32            }
33
34            should("contain default image") { ... }
35
36            context("validations") {
37                should("not allow empty name") { ... }
38                ...
39            }
40        }
41    }
42
```

1

2

3

```
override fun isolationMode() = IsolationMode.InstancePerLeaf
```

```
17    context("subcategory endpoints") {
18
19        // příprava obecných dat
20        val subcategoryForCreate = createSubcategoryDto()
21
22        context("when creating new subcategory") {
23
24            // další příprava, nebo provedení akce
25            val createdSubcategory = restTemplate.createSubcategory(subcategoryForCreate)
26
27            should("create basic info") {
28                // samotný test / asserty
29                createdSubcategory.id shouldNotBe null
30                createdSubcategory.name shouldBe subcategoryForCreate.name
31                createdSubcategory.active shouldBe true
32            }
33
34            should("contain default image") { ... }
35
36            context("validations") {
37                should("not allow empty name") { ... }
38                ...
39            }
40        }
41    }
42
```

1

2

3

```
override fun isolationMode() = IsolationMode.InstancePerLeaf
```

```
17  context("subcategory endpoints") {
18
19      // příprava obecných dat
20      val subcategoryForCreate = createSubcategoryDto()
21
22      context("when creating new subcategory") {
23
24          // další příprava, nebo provedení akce
25          val createdSubcategory = restTemplate.createSubcategory(subcategoryForCreate)
26
27          should("create basic info") {
28              // samotný test / asserty
29              createdSubcategory.id shouldNotBe null
30              createdSubcategory.name shouldBe subcategoryForCreate.name
31              createdSubcategory.active shouldBe true
32          }
33
34          should("contain default image") { ... }
35
36          context("validations") {
37              should("not allow empty name") { ... }
38              ...
39          }
40      }
```

1

2

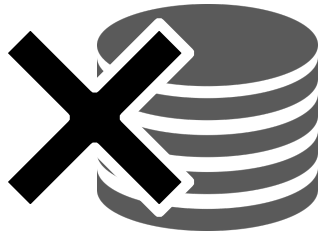
3

4

```
override fun isolationMode() = IsolationMode.InstancePerLeaf
```

```
42  }
```

```
afterSpec {
    dslContext.deleteDatabase()
}
```



Příprava dat přímo do DB

```
val customer = dslContext.createCustomer { name = "Customer 1" }
val office = dslContext.createOffice(customer)
```

```
fun DSLContext.createCustomer(applyFunction: CustomerRecord.() → Unit = {}) =
    newRecord(Tables.PARTNER)
        .apply {
            name = "customer" + Random.nextInt()
            active = true
            lastModified = OffsetDateTime.now()
            apply(applyFunction)
            store()  ^apply
        }
```

A co rychlost ?

Nestartujte při TDD kontejnery

```
docker-compose up -d
```

```
spring:
  datasource:
    driver-class-name: "org.testcontainers.jdbc.ContainerDatabaseDriver"
    url: "jdbc:tc:postgresql:13.8://localhost:45432/postgres?TC_REUSABLE=true"
```

```
val postgresSQLContainer = PostgreSQLContainer<Nothing>(POSTGRES_IMAGE_NAME)
    .apply { this: PostgreSQLContainer<Nothing>
        withReuse( reusable: true)
        start()
    }
```

Zrychlete build a start testů

```
./gradlew testClasses
    --exclude-task generateJooq
    --exclude-task update

    -PturnOffSlowAction=true
```

- Upravený build testů
- Šablona pro Kotest

- Log level = INFO
- Profil pro spring

Pojďme si TO ukázat

Odkazy

- https://kotest.io/
- https://mockk.io/ / https://github.com/Ninja-Squad/springmockk
- https://www.testcontainers.org/
- https://wiremock.org/

Zdrojáky: https://github.com/VitaPlsek/javadays2022

Kontakt: vita.plsek@morosystems.cz / www.linkedin.com/in/vitaplsek

GOPAS

Java Days

Děkuji Vám za pozornost