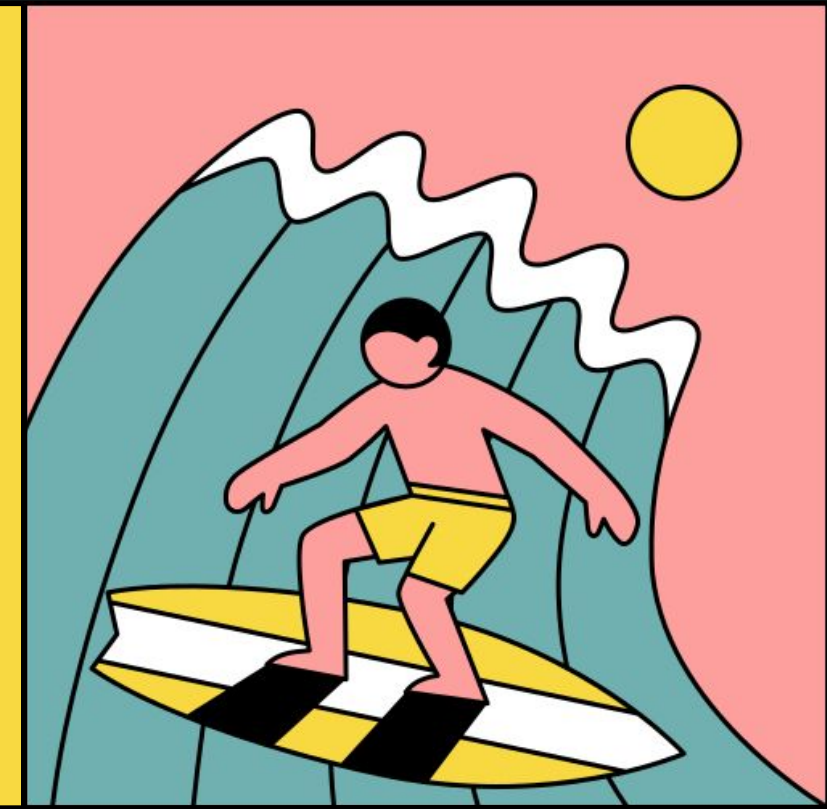


SoftUni | May 2023



Testing In Software Development

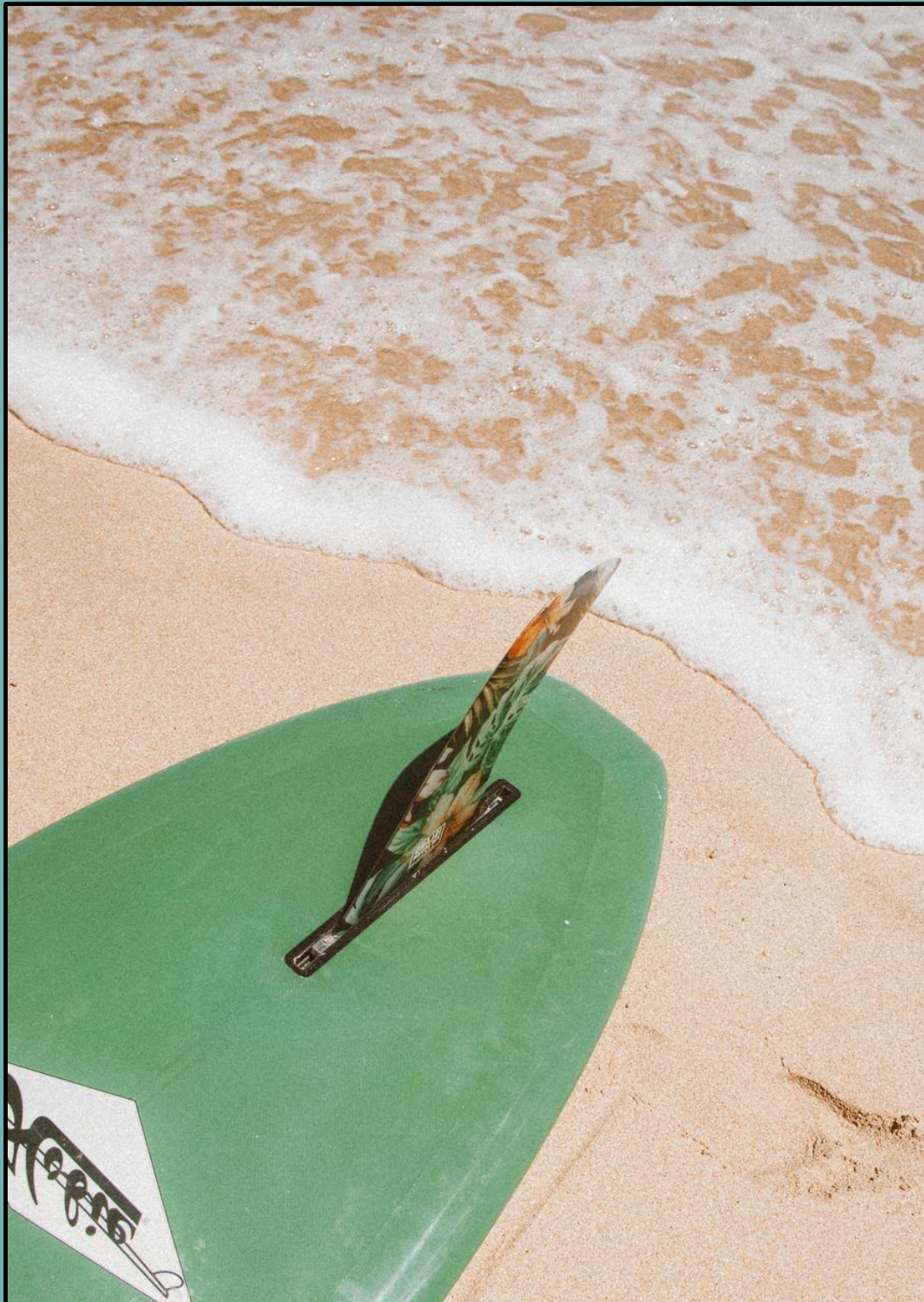
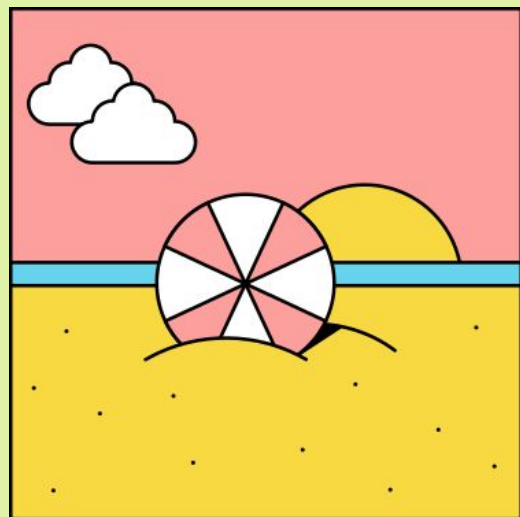


Table Of Contents

01

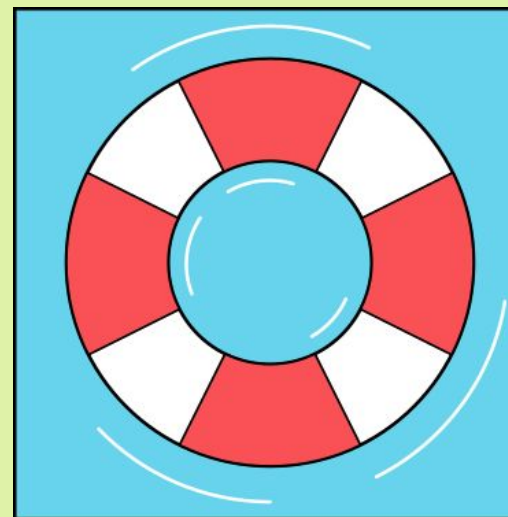
Introduction



What is quality and why you should care about it.

02

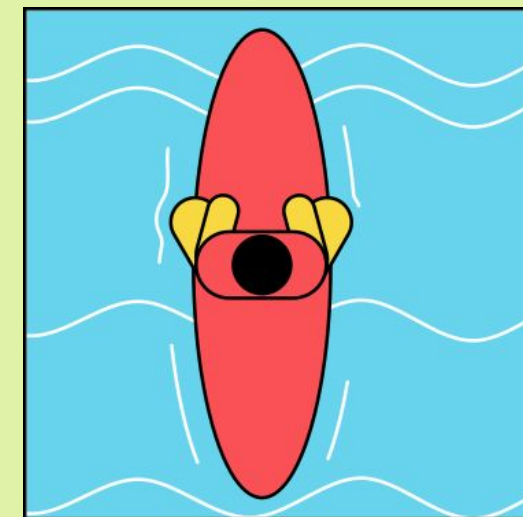
Process And Tools



The practicalities of creating and testing software.

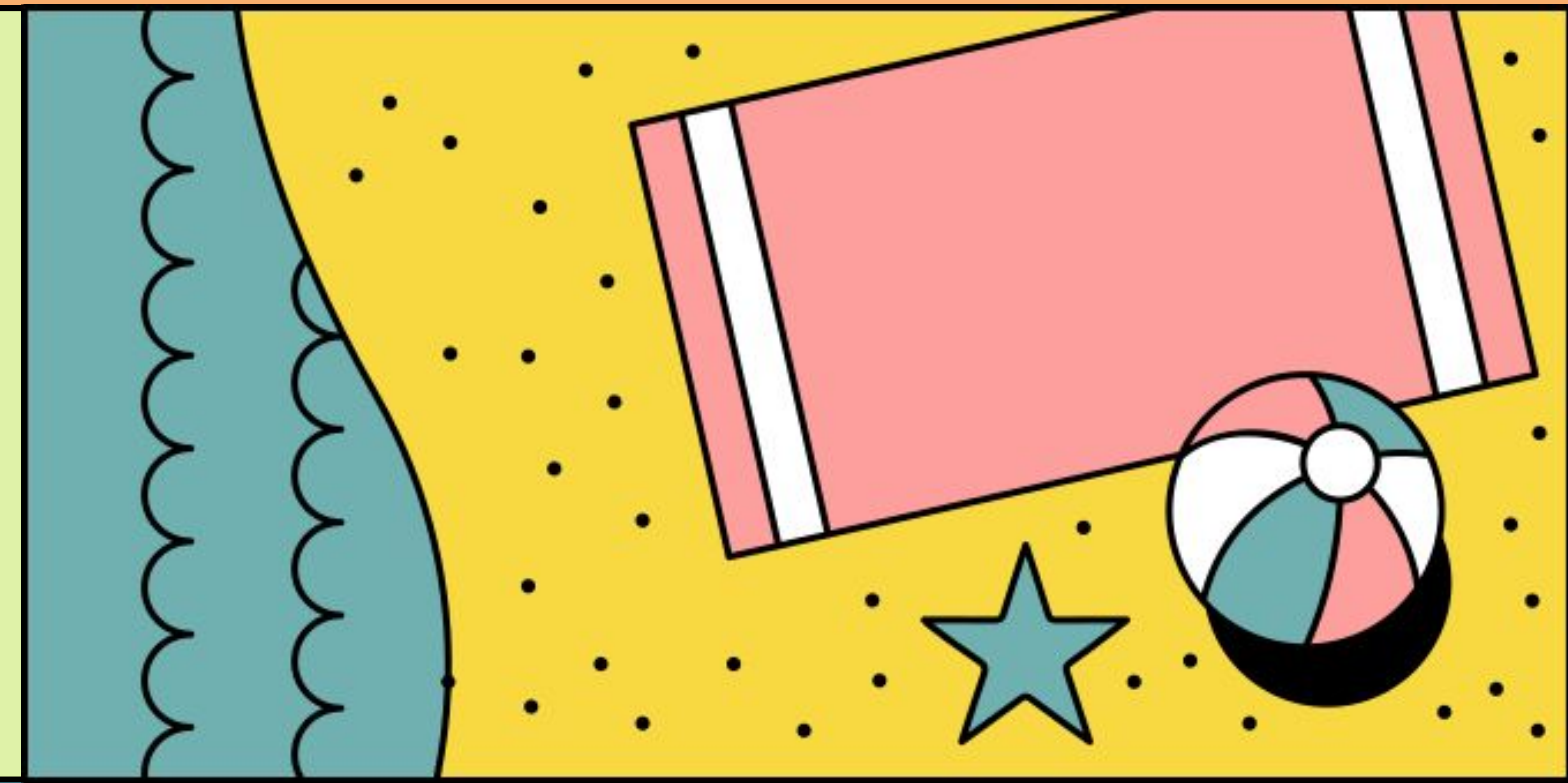
03

Bugs, Releasing and Running Application



So you have written some code.
Now what?

About Me



Yana Parvanova



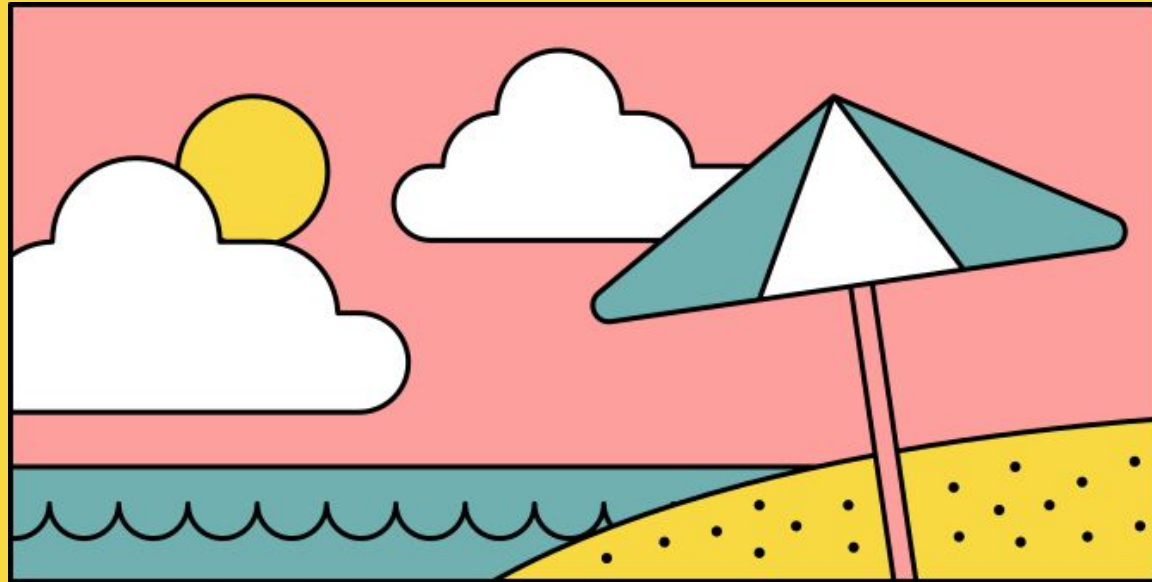
Director of Engineering
Taulia, Sofia

Taulia



FinTech, Supply Chain Finance
Headquartered in San Francisco
Founded in 2009. Acquired by SAP in 2022

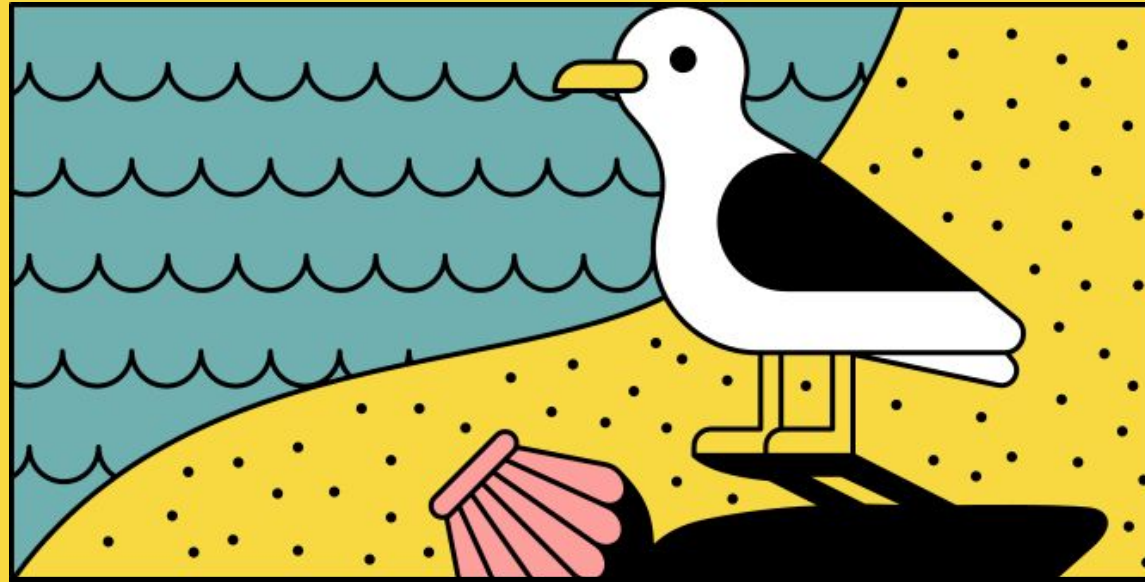
Quality



Why

This is your job.

"Anything worth doing, is worth doing right."



What

Consider context.

Consider short- and long-term implications.

Use objective metrics.



How

Quality as a mindset and culture.

Quality is a teamwork.

Requires discipline, creativity, tools.

Discovery & Analysis

Are you solving a real problem?
Is it the right problem to solve?
Who are the users?

Scope and expectations.
Prototyping.

Functional and non-functional
requirements (volume, load,
performance, security etc.)
Handling errors.

Testing.

Design & Development

Architecture, components,
communication and invocation.
Third-party libraries.

Code quality, code reviews, readability.
Complexity.

Testing. Manual and automated testing.
Test coverage. Bugs.

Learn, adapt, improve.

Release, Deploy, Run, Use

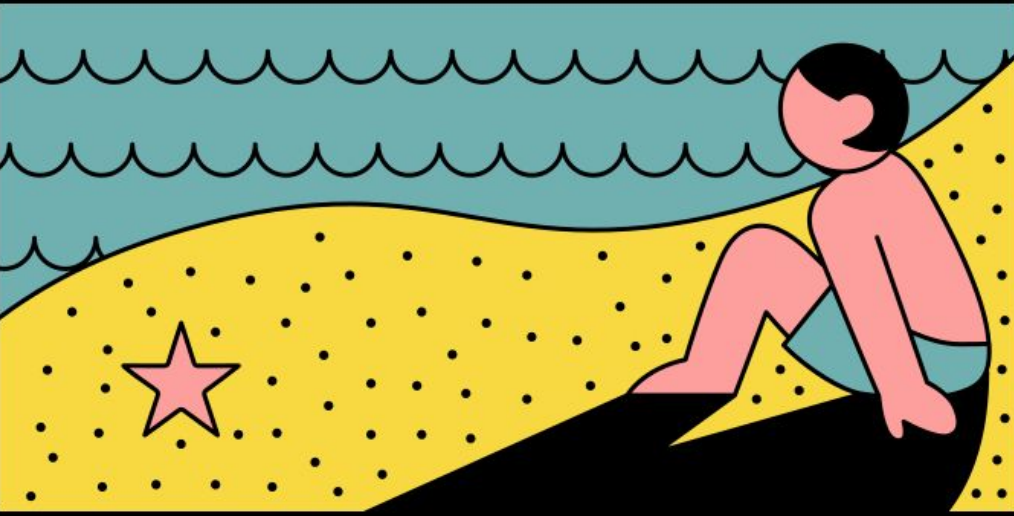
CI/CD.

Documentation and training.

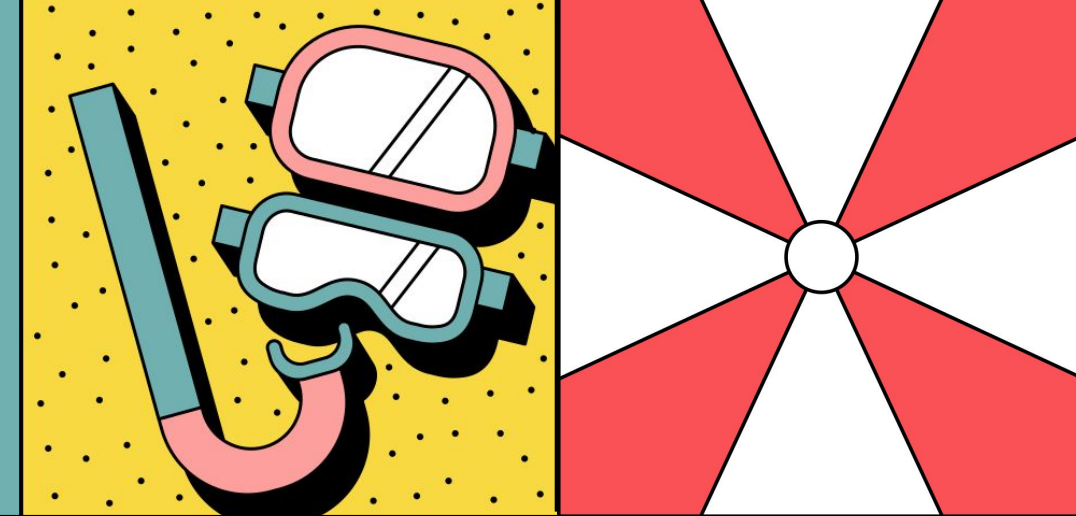
Monitoring, logs, alerting.

Testing (yes, again).





Types Of Tests

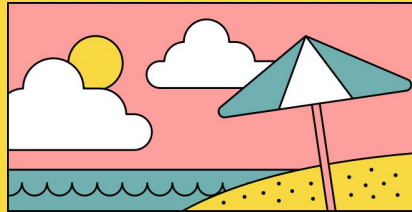
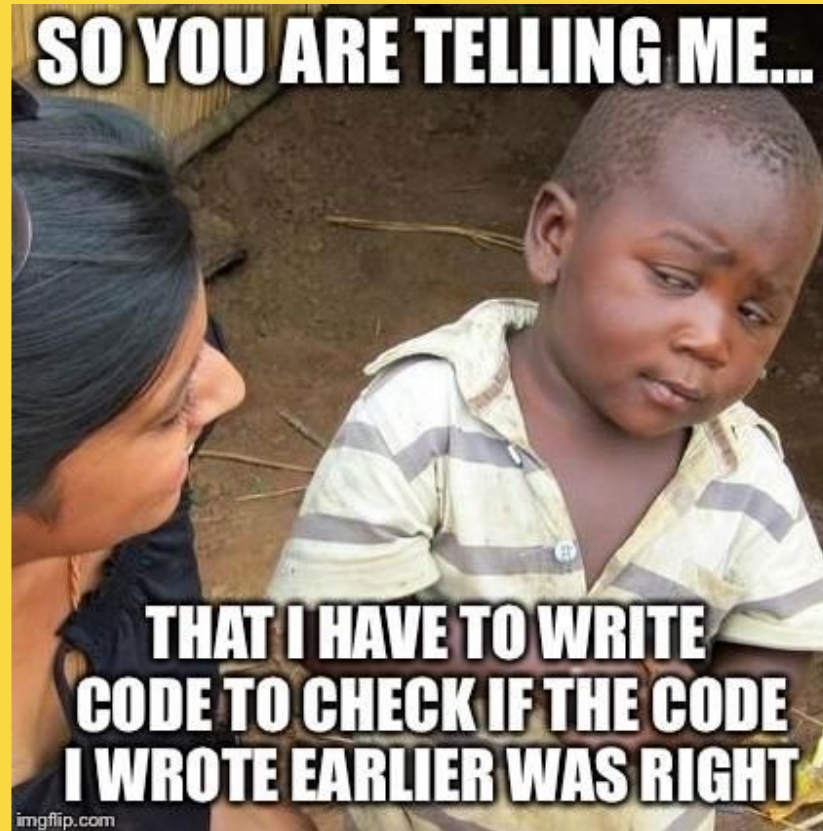


- Manual vs automated
- Exploratory testing
- Unit tests
- Integration/service tests
- End-to-end tests
- Regression testing

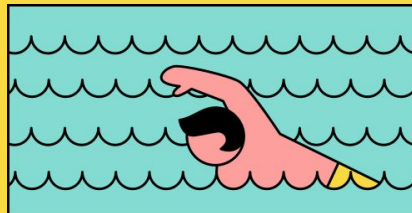
- Smoke/stability vs sanity tests
- Load/performance/stress tests
- User acceptance testing (UAT)
- Vulnerability testing
- Accessibility testing
- **(Quality Gates...)**



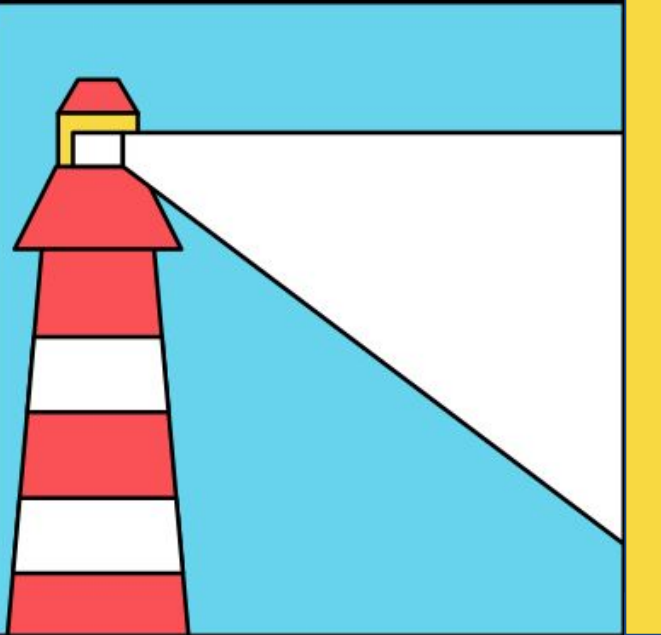
Test Automation



- Quality
- Safe refactoring
- Faster releases
- Absolutely needed to scale up
- Manual < Automated < **Continuous**



- Maintenance
- Does not pay off immediately
- Requires long-term commitment (effort, infrastructure and process)



Quality Gateways (Example)

[Open](#) **PLS-455-RemoveFunderQuoteRetrievalSetting** #1616
genadimitev-taulia wants to merge 6 commits into `taulia:develop` from `genadimitev-taulia:PLS-455-RemoveFunderQ...`

Kudos, SonarQube Quality Gate passed!

- 0 Bugs
- 0 Vulnerabilities
- 0 Security Hotspots
- 0 Code Smells

100.0% Coverage
0.0% Duplication

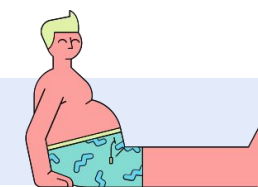
Add more commits by pushing to the **PLS-455-RemoveFunderQuoteSetting** branch on **genadimitev-taulia/intapi-payment-process-manager**.

Review required
At least 1 approving review is required by reviewers with write access. [Learn more.](#) [Add your review](#)

Some checks were not successful
1 errored and 2 successful checks

- continuous-integration/jenkins/pr-pde** — The build of this commit was aborted Required [Details](#)
- SonarQube Code Analysis** Successful in 21s — Quality Gate passed [Details](#)
- continuous-integration/jenkins/pr-merge** — This commit looks good Required [Details](#)

Test coverage of new code

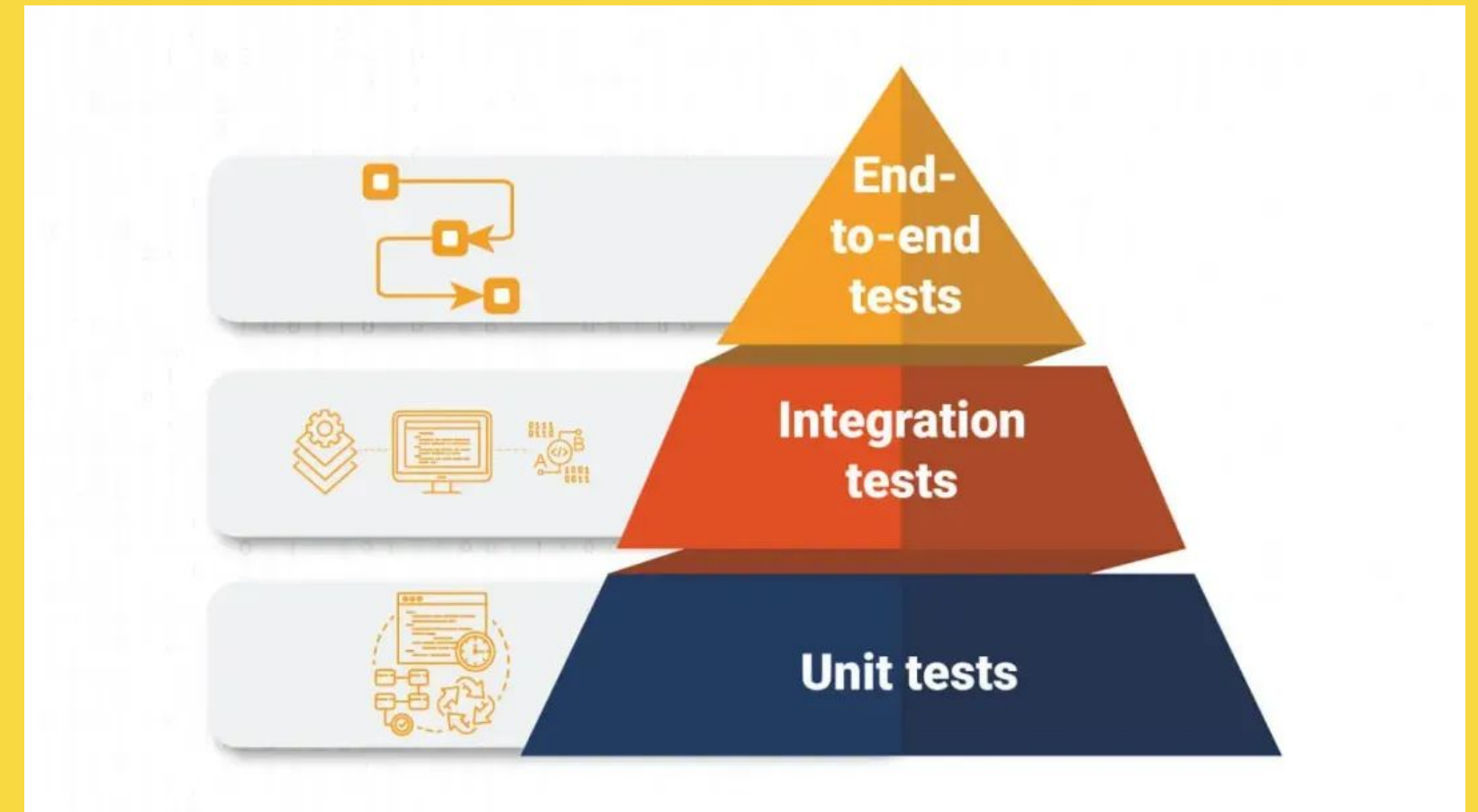


Stability tests on an on-demand integration environment

All unit & integration tests for the component

Unit Tests

- Small units of the implementation are tested in isolation
- Context is often mocked (database, external services)
- Small, fast, easy to maintain
- Can be data-driven
- Easy coverage for happy/unhappy paths and edge cases
- Test coverage can be measured (aim at 50-80%)
- Need to be maintained as features evolve.
- Usually created by developers as part of feature development and executed on regular basis – for example, each time before code is merged.



```

@Unroll
void "test Country is eligible for email when #programCountries with #supplierCountry -
should be #expectedResult"() {

    given:
    def supplierDetailsId = RandomStringUtils.random(32)

    when:
    def result = emailService.isSupplierCountryEligible(programCountries, supplierDetailsId)

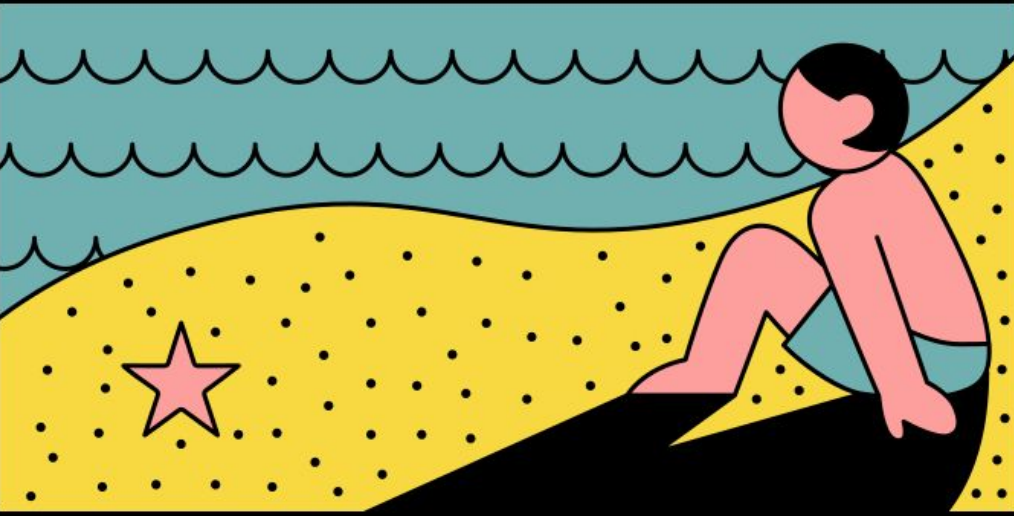
    then:
    invocationCount * companyService.getSupplierCountry(supplierCountry) >> [supplierCountry]

    and:
    result == expectedResult

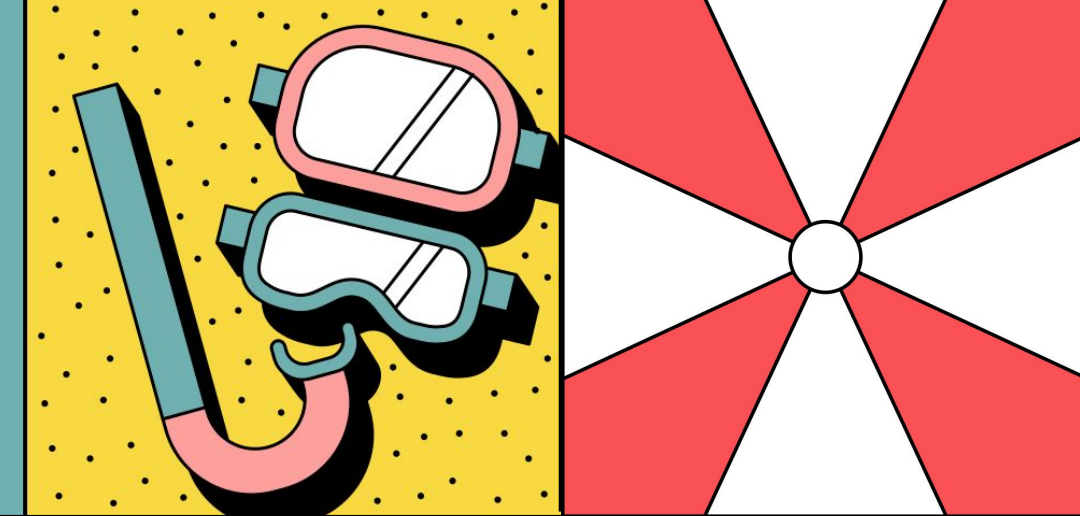
    where:
    programCountries | supplierCountry | expectedResult | invocationCount
    'MX, CA'         | 'CA'           | true          | 1
    '*'              | 'MX'           | true          | 0
    '*'              | null           | true          | 0
    'MX, CA'         | 'BG'           | false         | 1
    ''               | 'MX'           | false         | 0
    null             | null           | false         | 0
    'MX, CA'         | ''             | false         | 1
    'mx, Ca'         | 'CA'           | true          | 1
}

```

- **Spock** is a testing and specification framework for Java and Groovy applications.
- **given-when-then** is a standard format for capturing *acceptance criteria* or *test cases*.
- **Spy** > **Mock** > **Stub** (Spock-specific)
- Assert expected **result** and/or **interactions**
- **Data-driven testing (DDT)**: test for different scenarios and edge cases based on data variations.



Integration Tests



- Test two or more components, layers, services working together – for example application code + database + message queue.
- Can be manual or automated.
- More complex and pricey to maintain.
- Can test just a couple of services or full flows.



```
private void setupEpCancellationPathClientMocks() {
    String eprId = earlyPaymentRequestTdo.id

    when(tcClient.cancelEarlyPayment(anyList(), anyString()))
        .thenReturn([new CancellationResult(earlyPaymentRequestId: eprId, success: true)])

    when(funderClientV2.cancelFundingRequests(any(FundingRequestCancellation)))
        .thenReturn(new FundingRequestCancellationResult(successful: [new FundingRequestResult(requestId: eprId)]))
}
```

```
@Test
void testEarlyPaymentArProcessCancellationPath() {
    setupEpCancellationPathClientMocks()

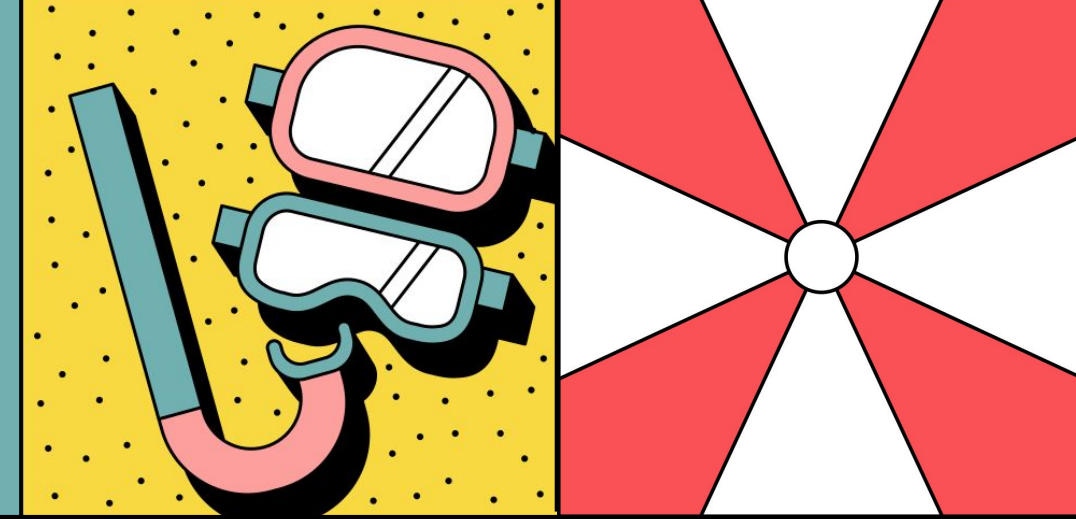
    ProcessInstance processInstance = startEarlyPaymentArProcess()

    await().atMost(Duration.TEN_SECONDS * 2).pollInterval(Duration.FIVE_HUNDRED_MILLISECONDS).untilAsserted {
        assertThat(processInstance).hasPassed(PAYMENT_INSTRUCTIONS_REQUEST)
    }

    assert earlyPaymentRequestPaymentInstructionsRepository.findByEarlyPaymentRequestId(processInstance.businessKey)
}
```

- **Mockito** is a mocking framework for Java applications.
- **JUnit** is the most popular testing framework for Java
- **Awaitility** is a DSL that allows you to express expectations of an asynchronous system in a concise and easy to read manner.

Functional/UI Tests

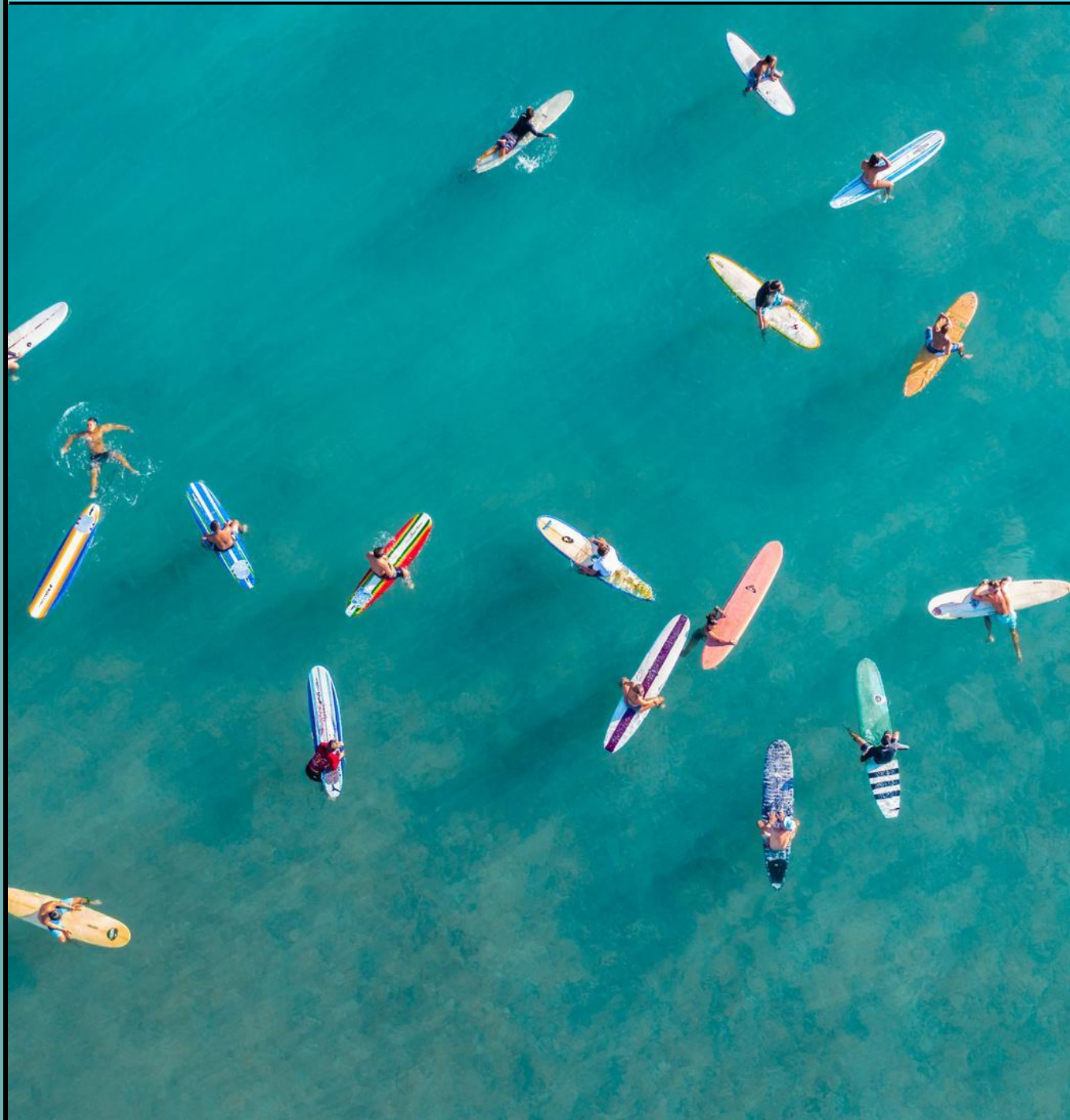


- Can be manual or automated.
- Due to complexity, these are hard to maintain.
- Slow to execute.
- Usually mimic user's behaviour as they executed specific actions in the application (i.e. log in -> submit an invoice -> wait for approval -> download pdf).
- Often involves tools and frameworks that allow for UI interaction (click, find an UI element etc.)
- Selenium is a popular framework for Web applications testing.

```
@Test
@Category(ClubSuedeRegressionUIBuyerGateway)
void GBP040_DisplayTauliaTermsAndConditions_Success() {
    def tac = termsAndConditionsHelper.createTermsAndConditions(buyerId:
buyerCompany.id, lang: Locale.US.language)
    loginToBuyerAdminArea()
    validateTac(tac)
}
```

```
def validateTac(tac) {
    to BuyerTermsAndConditionsPage
    at BuyerTermsAndConditionsPage
    assert tacContent.text() == tac.content
}
```

JUnit & Spock

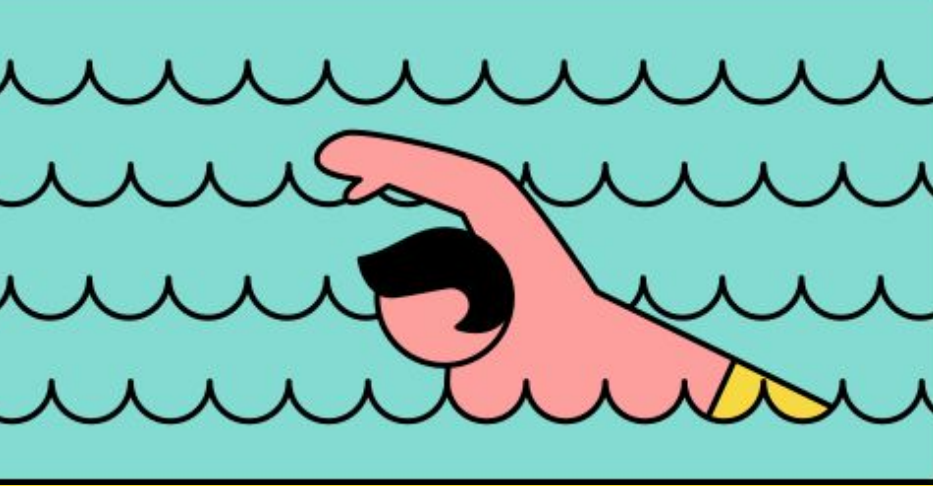


JUnit

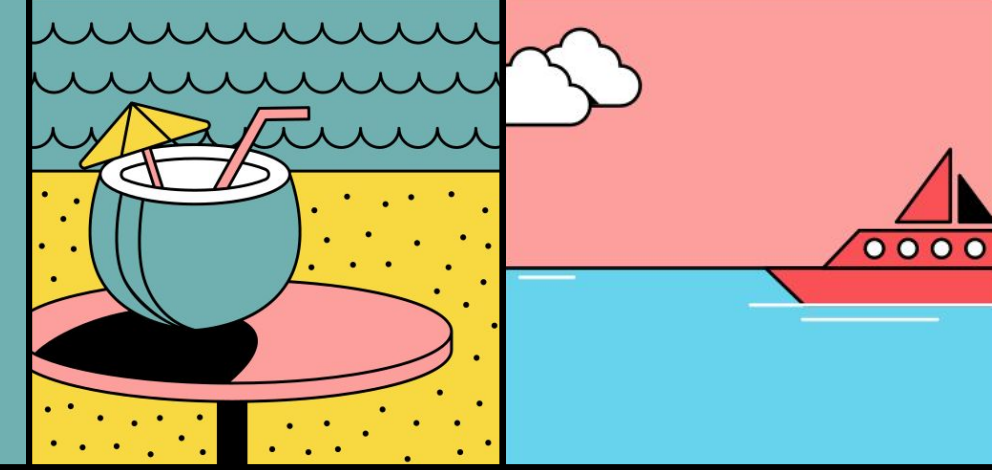
- Why is it so popular?
- Features
 - Runners (`@RunWith(SpringJUnit4ClassRunner.class)`...)
 - Tests (`@Test`, `@Suite`, `@Ignore`...)
 - DDT (`@ParameterizedTest`, `@ValueSource`)
 - Fixtures (`@Before`...)
 - Assertions (`assertThat`...)
 - Results
- Lacks in mocking abilities (need to use a third-party library like **Mockito**.)
- Very mature project with lots of documentation and examples.

Spock

- Covers all JUnit features
- Built-in mocking
- Self-documented tests
- Younger project but still very mature



Test Documentation



Test Plan

All test activities for a project – object of testing, work schedules, criteria for the beginning and end of testing, strategy, risks, communication plan, owners and a list of work performed.

Checklist

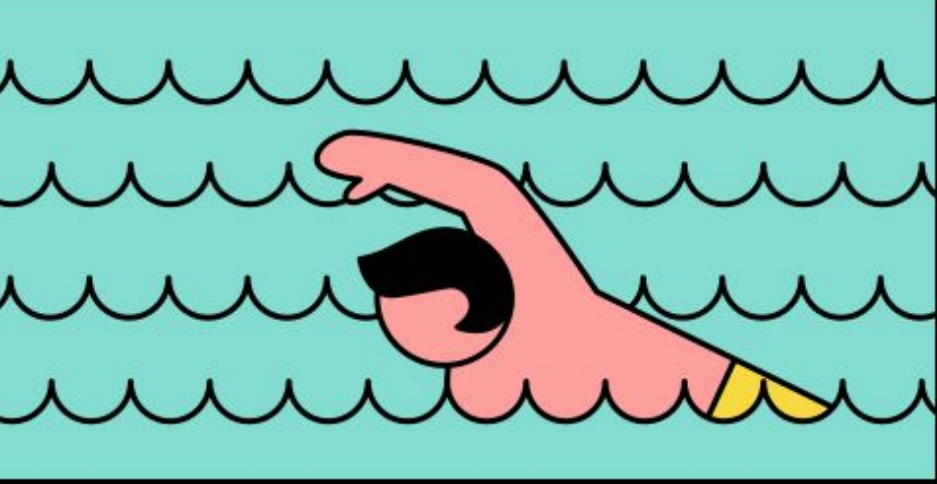
A list of features to test and statuses – the results you observe.

Test Scenarios & Test Cases

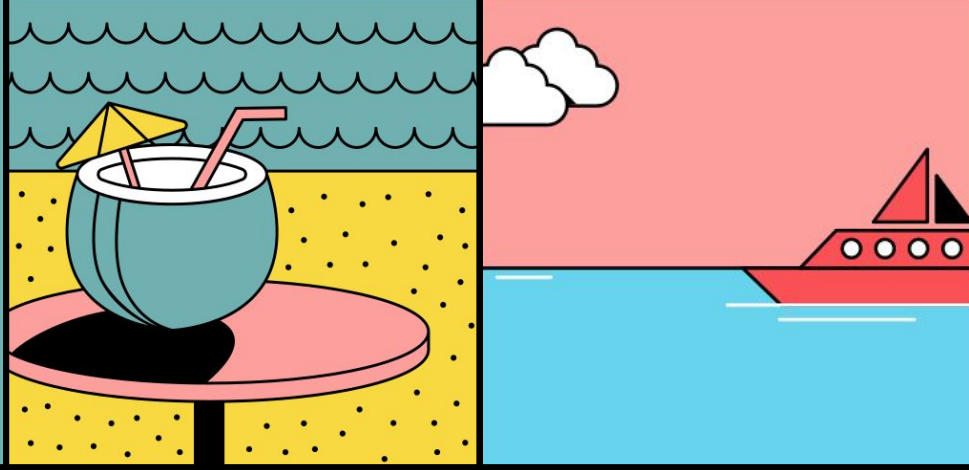
A feature that is tested – with preconditions, happy/unhappy paths, expected results.

Bug Reports

Full information about a bug (its description, severity and priority, etc.) and documents the steps and conditions for reproducing this bug.



Test Execution



Manual

Using UI, APIs etc. to invoke the software under test.

May be exploratory or following a test plan for specific project/features.

Automated

- Can be part of build/release process (triggers on pull requests, image builds, pre-deploy etc.)
- Scheduled (for example, run fast functional tests each hour and slow tests once a day).
- Started manually on demand (pre-release etc.)

smoke-test ▶ master ▶ 3700 - #maritsa-tests - integration ▶ Test Results

Test Result

4 failures (-16) , 2 skipped (+2)

36 tests (-28)
Took 2 hr 1 min.
[add description](#)




All Failed Tests

Test Name	Duration	Age
run jobs in parallel / Maritsa Team Tests Integration / com.taulia.test.camunda.EPAutomatedCancellationTest.AC080 testEPsNotSentToFunder	7 min 24 sec	1
run jobs in parallel / Maritsa Team Tests Integration / com.taulia.test.scfp.ScfpProvisionalAcceptanceE2ETests.ScfpProvisionalAcceptanceFunderFinallyAccepted	8 min 42 sec	1
notify that all jobs are done / com.taulia.test.camunda.EPAutomatedCancellationTest.AC080 testEPsNotSentToFunder	7 min 24 sec	1
notify that all jobs are done / com.taulia.test.scfp.ScfpProvisionalAcceptanceE2ETests.ScfpProvisionalAcceptanceFunderFinallyAccepted	8 min 42 sec	1

All Tests

Package	Duration	Fail (diff)	Skip (diff)	Pass (diff)	Total (diff)
com.taulia.test.camunda	26 min	2 +2	0	6 +6	8 +8
com.taulia.test.dd	3 min 20 sec	0	0	4 +4	4 +4
com.taulia.test.platform.fundingRequirements	6 min 17 sec	0	0	4 +4	4 +4
com.taulia.test.platform.prefundingNotice	5 min 28 sec	0	0	2 +2	2 +2
com.taulia.test.platform.supplier.earlyPayment.virtualCards	18 min	0	0	2 +2	2 +2
com.taulia.test.platform.supplier.myInvoices	2 min 29 sec	0	0	2 +2	2 +2
com.taulia.test.scfp	59 min	2 +2	2 +2	10 +10	14 +14

Bugs

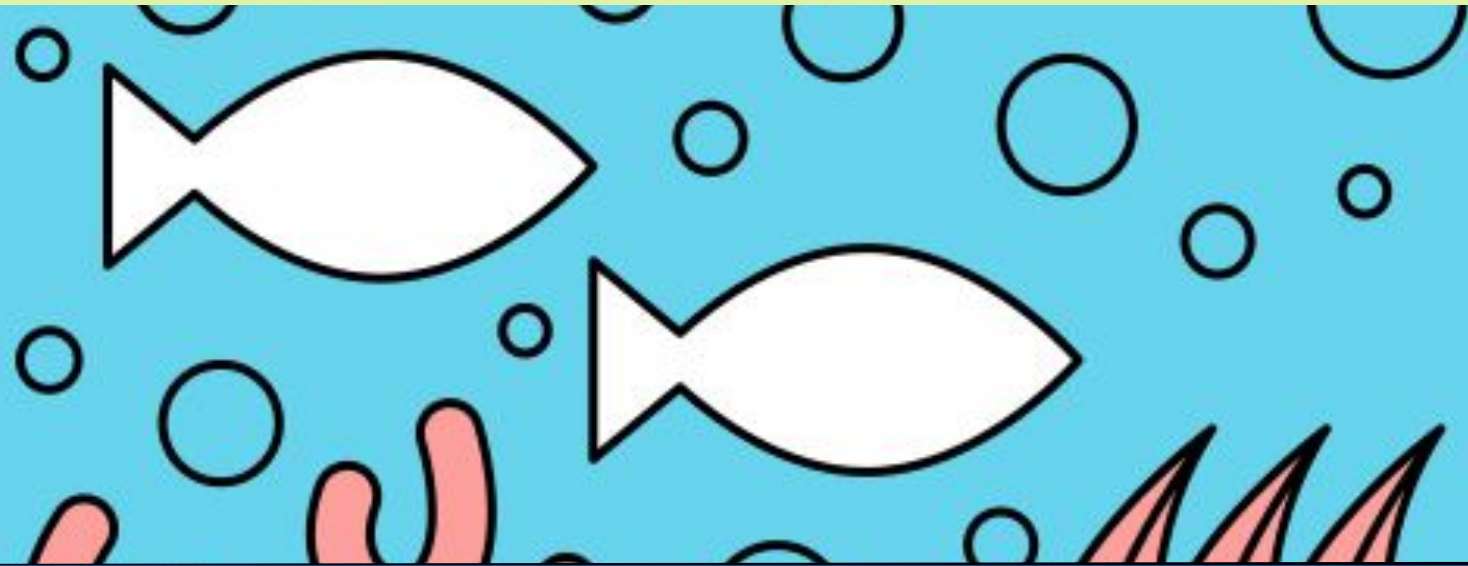
-  **10–70 bugs per 1000 lines of code**
-  **0.5–15 bugs per 1000 lines of code in released code**
-  **10–15 hours per week spent on bug fixing**

Bug Description

- Title/Bug ID.
- Environment – OS, browser version, application version etc.
- Steps to reproduce the bug.
- Expected result.
- Actual result.
- Visual proof (screenshots, videos, text) of the bug.
- Severity/Priority.



Release & Run

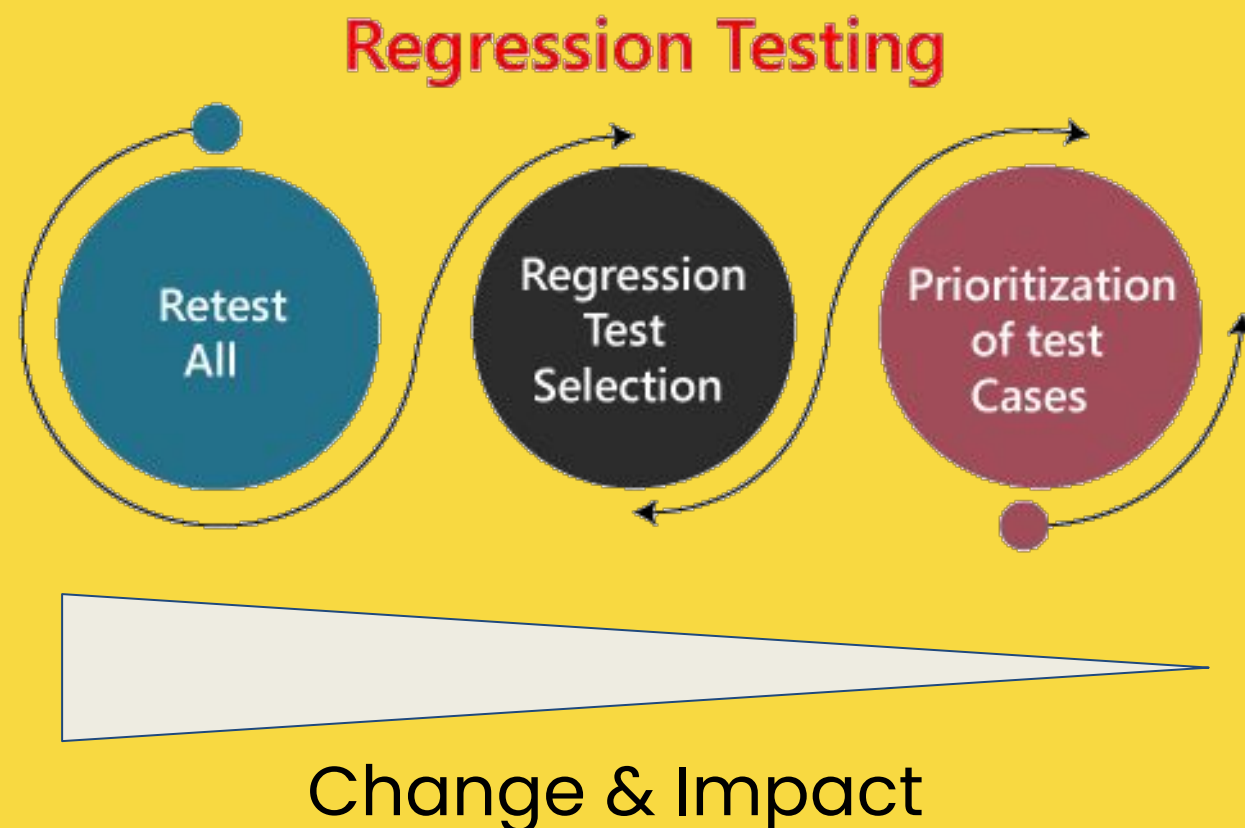


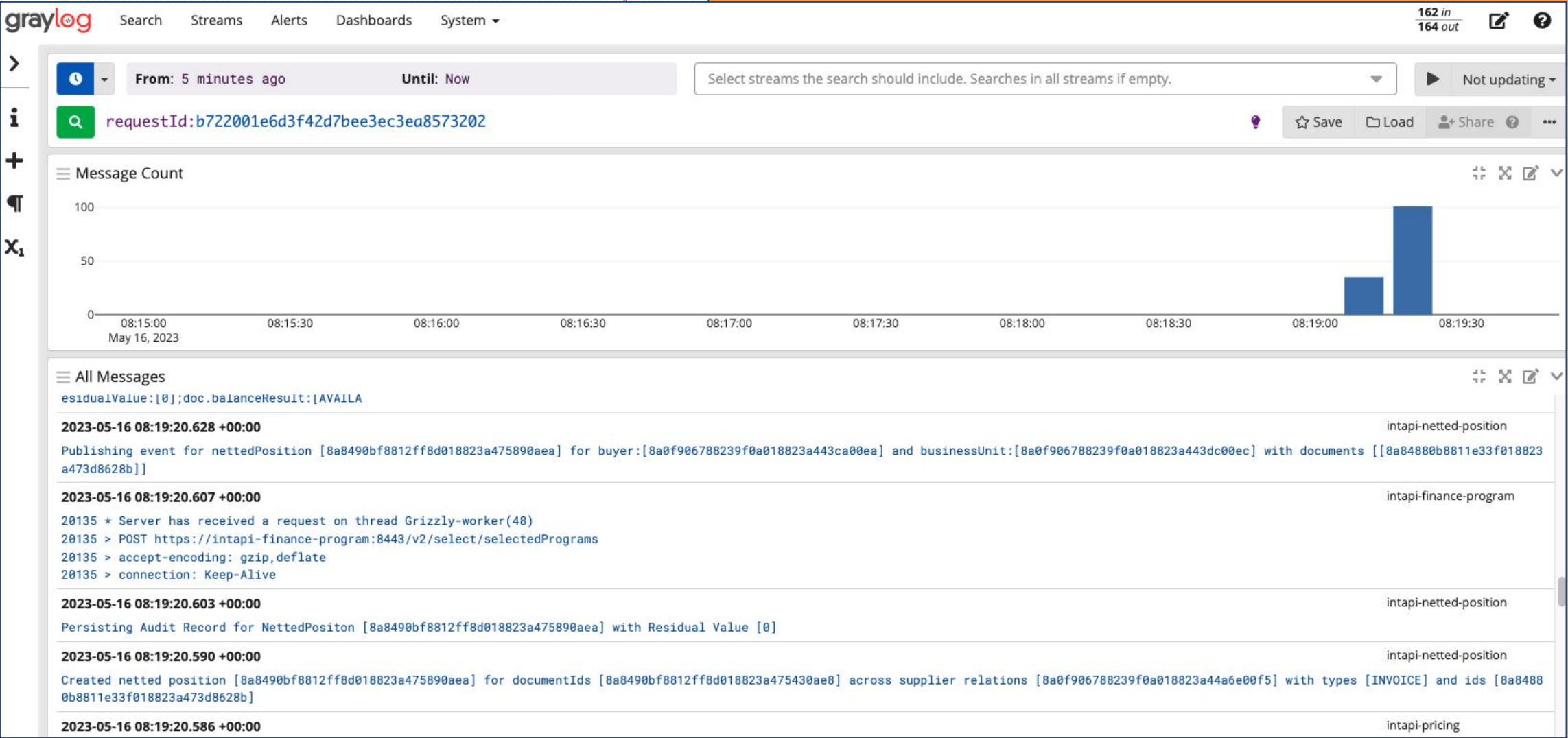
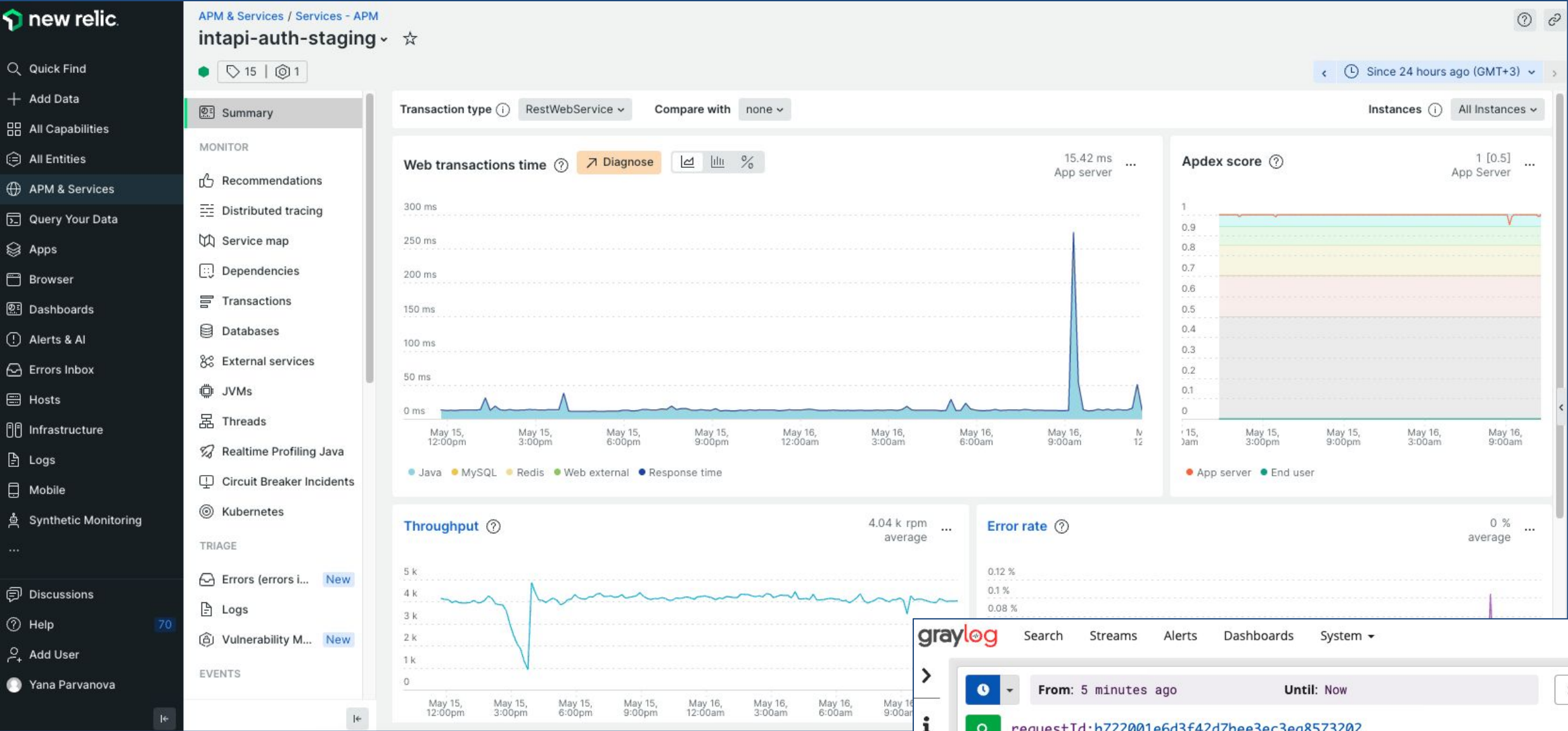
Stability/Smoke & Sanity Tests

- Discover quickly major issues on any environment (including Production)
- **Smoke:** all major functionality covered. If they fail, application is considered broken and rapid response needs to be started.
- **Sanity:** A subset of end-to-end tests run after a change (hotfix, configuration change etc.) used to verify application works approximately as expected.

Logging, Monitoring & Alerting

- **Logging:** consolidation, distributed tracing, meaningful data (IDs etc.), sensitive data, levels (debug <-> error).
- **Monitoring:** Track application, infrastructure and business metrics.
- **Alerting:** Push notifications based on metrics and thresholds to get the attention of the relevant team.





Thank You!

