Instituto Infnet

TESTE DE PERFOMANCE 2

JAVA WEB - EDC/LIVE

ALUNO: ELDER DE ANDRADE SILVA
PROFESSOR: ELBERTH MORAES

04/03/2024

Importância dos testes de software

Desde a popularização dos recursos da informática, como PC`s, smartphones e internet os softwares vem ganhado mais espaço no dia a dia da população. Hoje vivemos num mundo em que não podemos mais imagina-lo sem a presença da informática. Cada vez mais assuntos críticos são tratados com auxilio dos recursos computacionais e esse cenário levou a necessidade de produzir softwares mais confiáveis.

O teste de software surgiu como a melhor e mais prática forma de garantir a qualidade do software. Fundamentalmente, e atendendo o paradigma básico da computação que se divide em entradas, processamento e saídas, ele consiste em inserir um dado controlado no software e conferir se sua saída é consistente com a saída esperada pelo programador. Se para qualquer entrada pertencente ao domínio de software ele produz uma saída esperada, podemos dizer que o software funciona como esperado.

Entretanto, não é recomendável que o software seja testado apenas na sua versão final, pois, isto permitiria que erros fossem detectados apenas na fase final do desenvolvimento. Ao contrario, o software deve prever testes desde suas fases iniciais, mesmo antes que algum código seja escrito. Testar o software durante todo seu desenvolvimento evita a perda de produtividade (corrigindo códigos que irradiam correções para outras partes do programa), de custos (com mão de obra fazendo retrabalho), de reputação junto ao cliente (pelas perdas geradas ao mesmo pelo funcionamento inadequado do programa)

Existem vários tipos de testes que possuem escopos diferentes para atender a necessidade de um software de qualidade, dentre eles podemos citar o teste unitário.

O teste unitário é o responsável por testar as menores unidades funcionais do programa. Nesta etapa cada função, cada classe ou qualquer unidade que possua a capacidade de isoladamente receber uma entrada e produzir uma saída deverá ser testado. Isto garantirá que isoladamente as partes do software funcione corretamente. Por ser feito nas etapas iniciais do desenvolvimento ele permite que erros sejam corrigidos rapidamente, evitando a propagação do mesmo pelo código.

Por exemplo, podemos citar uma função que não retorne o valor esperado para determinada entrada, faria com que toda funcionalidade que dependesse desta função se comportasse de forma inconsistente.

Tipos de testes

Teste unitário - é o responsável por testar as menores unidades funcionais do programa. Nesta etapa cada função, cada classe ou qualquer unidade que possua a capacidade de isoladamente receber uma entrada e produzir uma saída deverá ser testado. Isto garantirá que isoladamente as partes do software funcione corretamente. Por ser feito nas etapas iniciais do desenvolvimento ele permite que erros sejam corrigidos rapidamente, evitando a propagação do mesmo pelo código.

Teste integração - é o teste que verifica o comportamento dos blocos testados na fase do teste unitário após a integração entre eles. Neste teste o programa é visto como um conjunto de subsistemas, e cada subsistema é testado isoladamente. Erros como incompatibilidade de tipos entre entes podem ser identificados neste momento. Apesar de ser um teste mais completo que o teste unitário, ele também não é capaz de garantir que o programa não contem erros.

Teste de sistema - é o teste em que todos os subsistemas testados na etapa de integração trabalham em conjunto dando a forma final do programa. Este é o ultimo nível na etapa puramente de desenvolvimento

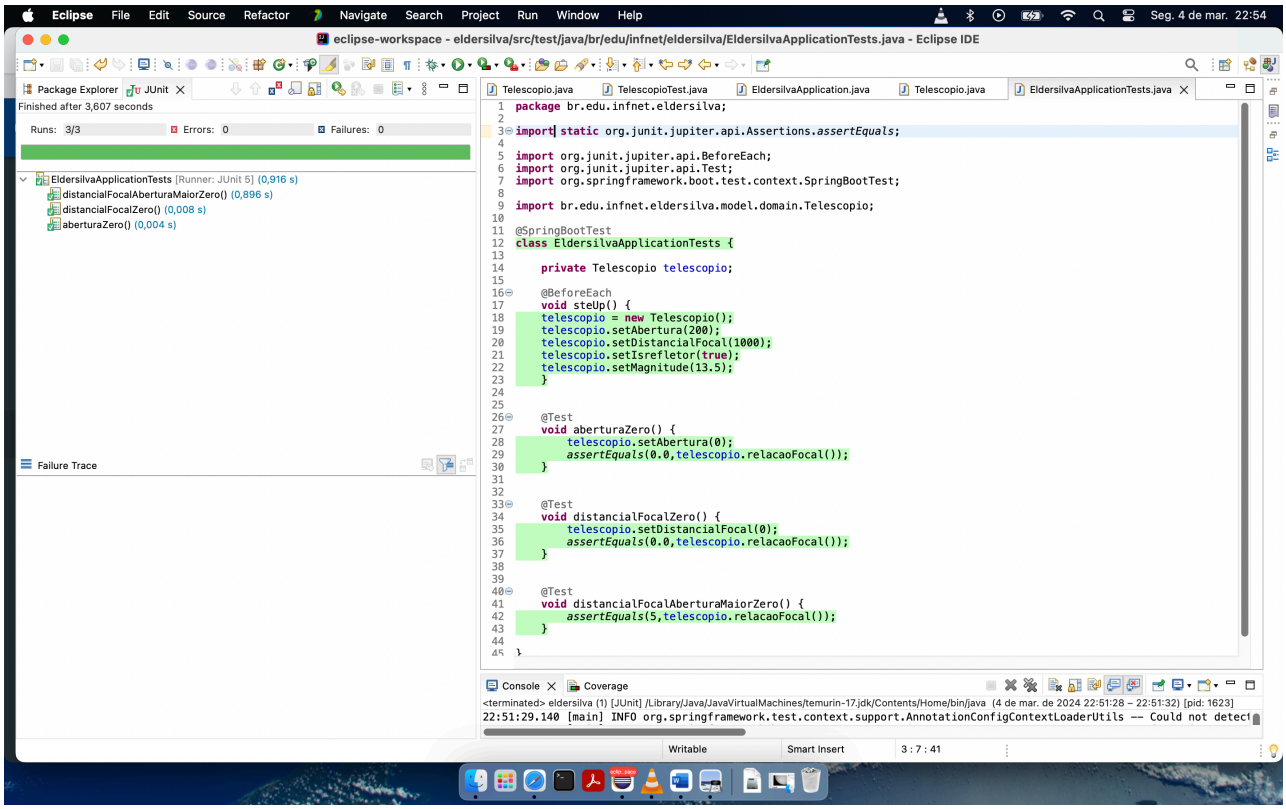
Teste de aceitação - é o teste que feito com a participação do usuário, ele é o responsável pela sinalização se software atende as suas necessidades. É uma etapa que se repete ate que o usuário valide a aplicação, ou caso o desenvolvimento seja incremental, a cada nova versão do software.

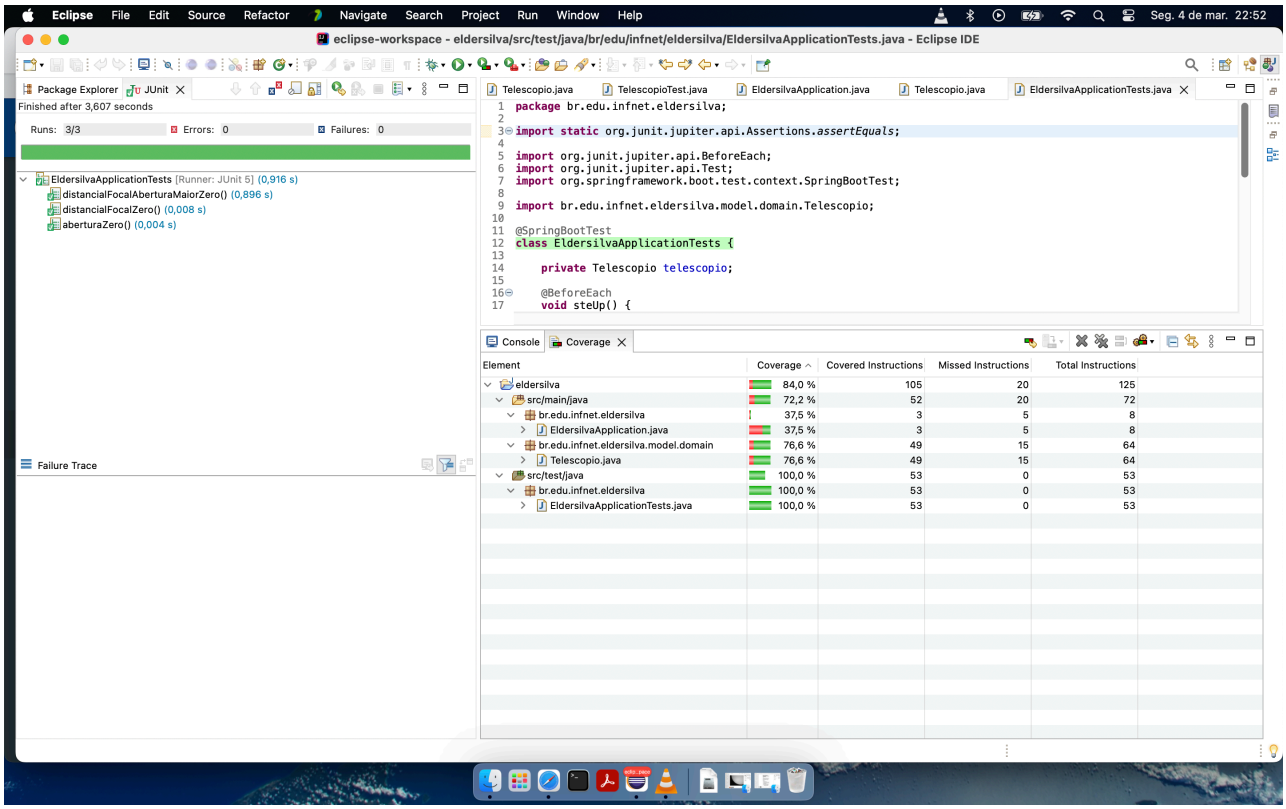
Teste de regressão – após o feedback do cliente sobre possíveis correções, o software volta para a produção e as implementações são feitas. Essas modificações poderiam gerar novos erros no sistema. Para evitar esse inconveniente o software passa por uma fase de teste conhecida como regressão. Neste momento os testes existentes das etapas anteriores são reaplicados a fim de encontrar possíveis novos erros introduzidos

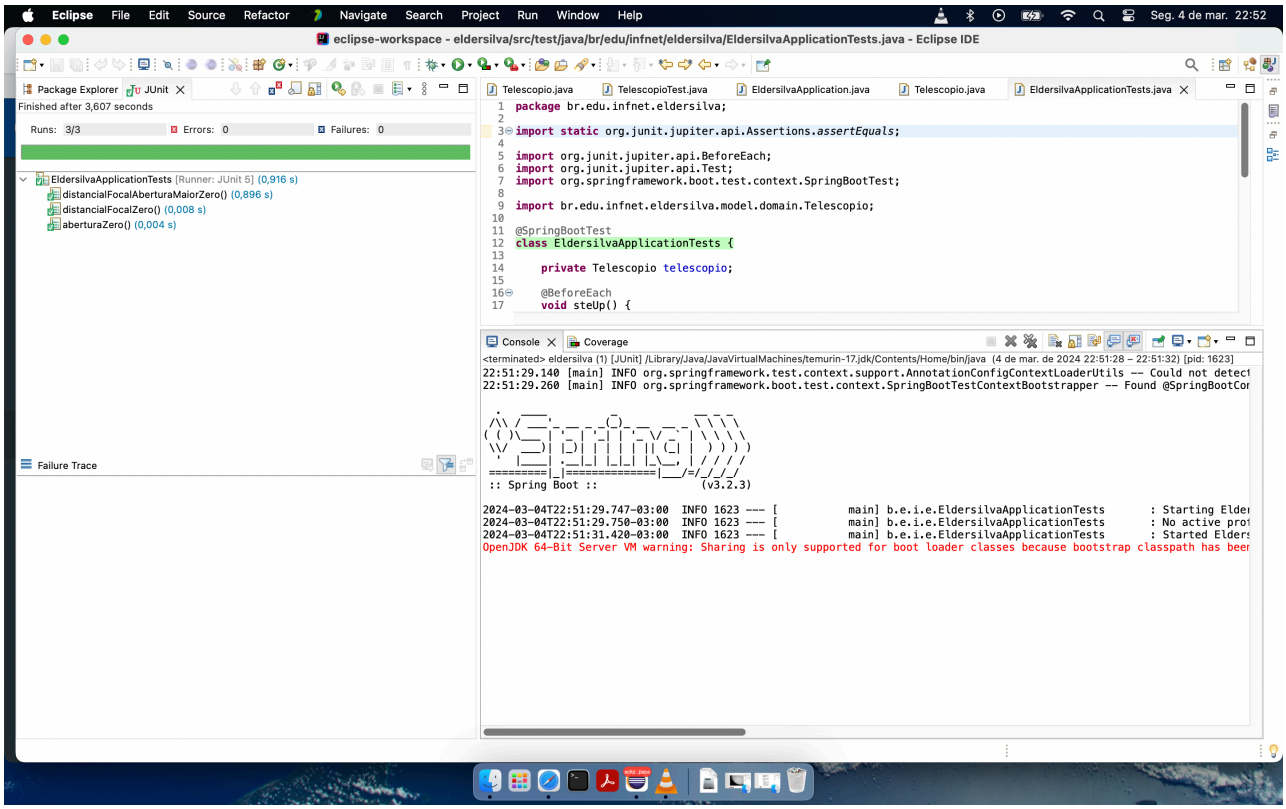
Teste de desempenho – após a validação das funcionalidades é feito o teste de despenho, que corresponde a etapa onde o programa é submetido a uma carga de trabalho que busca comprovar que o sistema está bem dimensionado para uma situação real de trabalho

Teste de segurança – busca entrar brechas que comprometam a integridade, confidencialidade e desempenho dos sistemas. Quanto maior for a criticidade nestes parâmetros maior será o foco nesta etapa.

Prints dos testes com Junit




Eclipse File Edit Source Refactor Navigate Search Project Run Window Help
Seg. 4 de mar. 22:52
eclipse-workspace - eldersilva/src/test/java/br/edu/infnet/eldersilva/EldersilvaApplicationTests.java - Eclipse IDE
Package Explorer JUnit
Finished after 3,607 seconds
Runs: 3/3 Errors: 0 Failures: 0
EldersilvaApplicationTests [Runner: JUnit 5] (0,916 s)
distancialFocalAberturaMaiorZero() (0,896 s)
distancialFocalZero() (0,008 s)
aberturaZero() (0,004 s)
Failure Trace
Telescopio.java TelescopioTest.java EldersilvaApplication.java Telescopio.java EldersilvaApplicationTests.java
package br.edu.infnet.eldersilva;
import static org.junit.jupiter.api.Assertions.assertEquals;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import org.springframework.boot.test.context.SpringBootTest;
import br.edu.infnet.eldersilva.model.domain.Telescopio;
@SpringBootTest
class EldersilvaApplicationTests {
private Telescopio telescopio;
@BeforeEach
void steUp() {
Console Coverage
Element Coverage Covered Instructions Missed Instructions Total Instructions
eldersilva 84,0 % 105 20 125
src/main/java 72,2 % 52 20 72
br.edu.infnet.eldersilva 37,5 % 3 5 8
EldersilvaApplication.java 37,5 % 3 5 8
br.edu.infnet.eldersilva.model.domain 76,6 % 49 15 64
Telescopio.java 76,6 % 49 15 64
src/test/java 100,0 % 53 0 53
br.edu.infnet.eldersilva 100,0 % 53 0 53
EldersilvaApplicationTests.java 100,0 % 53 0 53


Eclipse File Edit Source Refactor Navigate Search Project Run Window Help
Seg. 4 de mar. 22:52
eclipse-workspace - eldersilva/src/test/java/br/edu/infnet/eldersilva/EldersilvaApplicationTests.java - Eclipse IDE
Package Explorer JUnit
Finished after 3,607 seconds
Runs: 3/3 Errors: 0 Failures: 0
EldersilvaApplicationTests [Runner: JUnit 5] (0,916 s)
distancialFocalAberturaMaiorZero() (0,896 s)
distancialFocalZero() (0,008 s)
aberturaZero() (0,004 s)
Failure Trace
Telescopio.java TelescopioTest.java EldersilvaApplication.java Telescopio.java EldersilvaApplicationTests.java
package br.edu.infnet.eldersilva;
import static org.junit.jupiter.api.Assertions.assertEquals;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import org.springframework.boot.test.context.SpringBootTest;
import br.edu.infnet.eldersilva.model.domain.Telescopio;
@SpringBootTest
class EldersilvaApplicationTests {
private Telescopio telescopio;
@BeforeEach
void steUp() {
Console Coverage
<terminated> eldersilva (1) [JUnit] /Library/Java/JavaVirtualMachines/temurin-17.jdk/Contents/Home/bin/java (4 de mar. de 2024 22:51:28 – 22:51:32) [pid: 1623]
22:51:29.140 [main] INFO org.springframework.test.context.support.AnnotationConfigContextLoaderUtils -- Could not detect
22:51:29.260 [main] INFO org.springframework.boot.test.context.SpringBootTestContextBootstrapper -- Found @SpringBootCon
:: Spring Boot :: (v3.2.3)
2024-03-04T22:51:29.747-03:00 INFO 1623 --- [main] b.e.i.e.EldersilvaApplicationTests : Starting Elder
2024-03-04T22:51:29.750-03:00 INFO 1623 --- [main] b.e.i.e.EldersilvaApplicationTests : No active prof
2024-03-04T22:51:31.420-03:00 INFO 1623 --- [main] b.e.i.e.EldersilvaApplicationTests : Started Elders
OpenJDK 64-Bit Server VM warning: Sharing is only supported for boot loader classes because bootstrap classpath has been