

INSTITUTO INFNET - FACULDADE DE TECNOLOGIA, ENGENHARIA,
COMPUTAÇÃO, SISTEMAS E PRODUTOS DIGITAIS

AIRTON CANDIDO DE OLIVEIRA

DESENVOLVIMENTO DE SERVIÇOS WEB E TESTES COM JAVA

PROF. TIAGO AGUIAR

AT

Rio de Janeiro

01/04/2024

Sumário

PRINTS SOLICITADOS NO TP1.....	3
Introdução.....	9
Importância de testar o software.....	9
Diferentes tipos de testes.....	9
Testes de unidade.....	9
Testes de integração.....	9
Testes funcionais.....	9
Testes de ponta a ponta.....	10
Testes de aceitação.....	10
Testes de desempenho.....	10
Teste de fumaça.....	10
Execução dos testes.....	10
Configuração Junit.....	11
APIs Web Restful:.....	12
HTTP e HTTPs.....	13
Bibliografia.....	14

PRINTS SOLICITADOS NO TP1

1 -

The screenshot shows the IntelliJ IDEA IDE interface. The top toolbar includes icons for running, debugging, and other IDE functions. The left sidebar displays the Project view with a tree structure of the 'Estoque' project, including folders like 'src', 'main', 'java', and 'resources'. The central editor pane shows the 'Estoque.java' file with the following code:

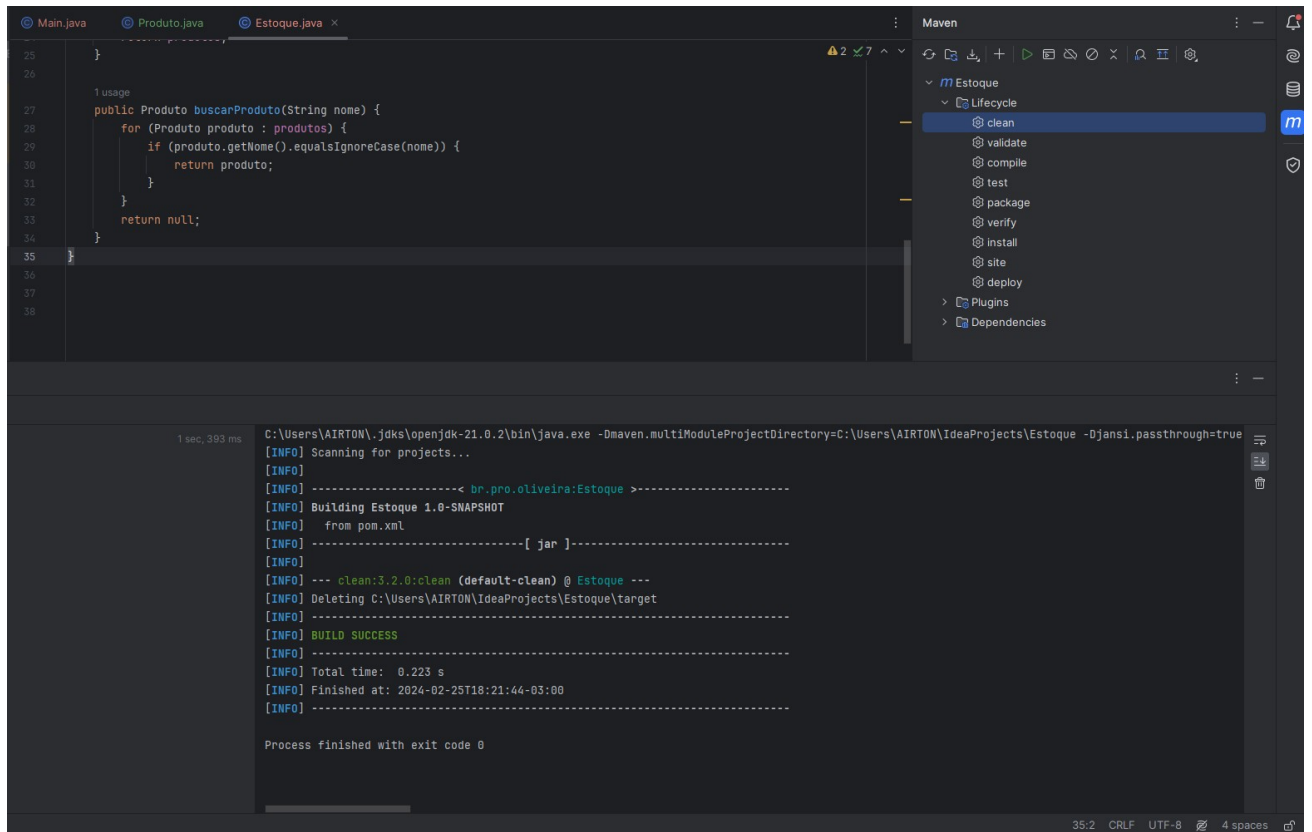
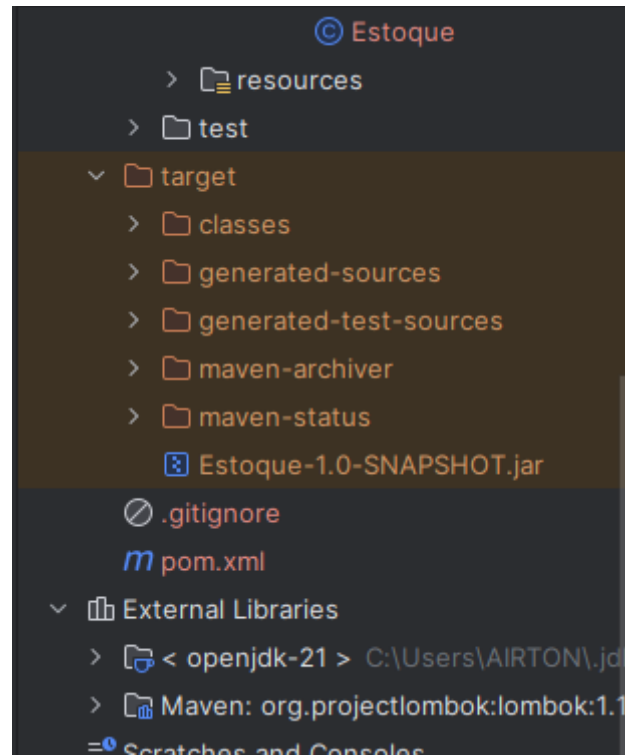
```
1 usage
23 public List<Produto> listarProdutos() {
24     return produtos;
25 }
26
27 1 usage
28 public Produto buscarProduto(String titulo) {
29     for (Produto produto : produtos) {
30         if (produto.getNome().equalsIgnoreCase(titulo)) {
31             return produto;
32         }
33     }
34     return null;
35 }
36
37
38
```

The right sidebar shows the Maven view with the 'Lifecycle' tab selected, displaying a list of Maven goals: clean, validate, compile, test, package, verify, install, site, and deploy. Below the Maven view, the Run console shows the output of the application:

```
Run Main x
C:\Users\AIRTOM\.jdk\openjdk-21.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.3.3\lib\idea_rt.jar=52386:C:\Program Files\JetBrains\IntelliJ IDEA 2023.3.3\bin" -Dfile.encoding=UTF-8 -Dsun.stdout
Todos os produtos do estoque:
Nome: Computador
Preço: $100.0
Código: 85-7542-239-1
Nome: TV
Preço: $154.0
Código: 9788595080805
Produto encontrado por nome: 'Computador'
Nome: Computador
Preço: $100.0
Código: 85-7542-239-1
Process finished with exit code 0
```

The bottom status bar indicates the current file is 'Estoque.java' and the encoding is UTF-8.

2 -



The screenshot displays an IDE interface with three tabs: Main.java, Produto.java, and Estoque.java. The active tab, Estoque.java, contains the following Java code:

```
25 }
26
27 1 usage
28 public Produto buscarProduto(String nome) {
29     for (Produto produto : produtos) {
30         if (produto.getNome().equalsIgnoreCase(nome)) {
31             return produto;
32         }
33     }
34     return null;
35 }
36
37
38
```

The Maven tool window on the right shows the 'Maven' tab with the 'Lifecycle' section expanded. The 'compile' goal is selected under the 'm' icon.

The console at the bottom shows the output of the Maven compile goal:

```
2 sec, 539 ms [INFO] Copying 1 resource from src\main\resources to target\classes
[INFO]
[INFO] --- compiler:3.11.0:compile (default-compile) @ Estoque ---
[INFO] Changes detected - recompiling the module! :source
[INFO] Compiling 3 source files with javac [debug target 21] to target\classes
[INFO] Annotation processing is enabled because one or more processors were found
on the class path. A future release of javac may disable annotation processing
unless at least one processor is specified by name (-processor), or a search
path is specified (--processor-path, --processor-module-path), or annotation
processing is enabled explicitly (-proc:only, -proc:full).
Use -Xlint:-options to suppress this message.
Use -proc:none to disable annotation processing.
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 1.281 s
[INFO] Finished at: 2024-02-25T18:22:17-03:00
[INFO] -----
Process finished with exit code 0
```

The status bar at the bottom indicates the file encoding is UTF-8 and the line ending is CRLF.

The screenshot shows an IDE with three tabs: Main.java, Produto.java, and Estoque.java. The 'Estoque.java' tab is active, showing a Java class with a method `buscarProduto(String nome)`. The Maven sidebar on the right shows the lifecycle with 'package' selected. The terminal at the bottom displays the output of a Maven build, showing successful compilation and packaging of the 'Estoque' project.

```

25 }
26
27 1 usage
28 public Produto buscarProduto(String nome) {
29     for (Produto produto : produtos) {
30         if (produto.getNome().equalsIgnoreCase(nome)) {
31             return produto;
32         }
33     }
34     return null;
35 }
36
37
38

```

Maven

- Estoque
 - Lifecycle
 - clean
 - validate
 - compile
 - test
 - package
 - verify
 - install
 - site
 - deploy
 - Plugins
 - Dependencies

```

[INFO] --- resources:3.3.1:testResources (default-testResources) @ Estoque ---
[INFO] skip non existing resourceDirectory C:\Users\AIRTON\IdeaProjects\Estoque\src\test\resources
[INFO] --- compiler:3.11.0:testCompile (default-testCompile) @ Estoque ---
[INFO] Nothing to compile - all classes are up to date
[INFO] --- surefire:3.1.2:test (default-test) @ Estoque ---
[INFO] No tests to run.
[INFO] --- jar:3.3.0:jar (default-jar) @ Estoque ---
[INFO] Building jar: C:\Users\AIRTON\IdeaProjects\Estoque\target\Estoque-1.0-SNAPSHOT.jar
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 0.969 s
[INFO] Finished at: 2024-02-25T18:22:34-03:00
[INFO] -----
Process finished with exit code 0

```

35:2 CRLF UTF-8 4 spaces

3 -

The screenshot shows a Java class with a method `toString()` annotated with `@Override`. The method body contains a return statement with a string concatenation expression. A tooltip is visible over the expression, showing the result of the concatenation: `"Preço: $" + nome + "Código: " + "Computador"`.

```

26 new *
27 @Override
28 public String toString() {
29     return "\nNome: " + nome + "\n" + nome: "Computador"
30         "Preço: $" +
31         "Código: " +
32     }
33 }
34

```

Tooltip: `"Preço: $" + nome + "Código: " + "Computador"`

4 -

```
new *
26 @Override
27 public String toString() {
28
29     return "\nNome: " + nome + "\n" + nome: "Computador"
30         "Preço: $" +
31         "Código: " +
32 }
33 }
34
```

```
26 @Override
27 public String toString() {
28
29     return "\nNome: " + nome + "\n" + nome: "Teste\n"
30         "Preço: $" + p
31         "Código: " + c
32 }
33 }
34
35
```

Debugger interface:

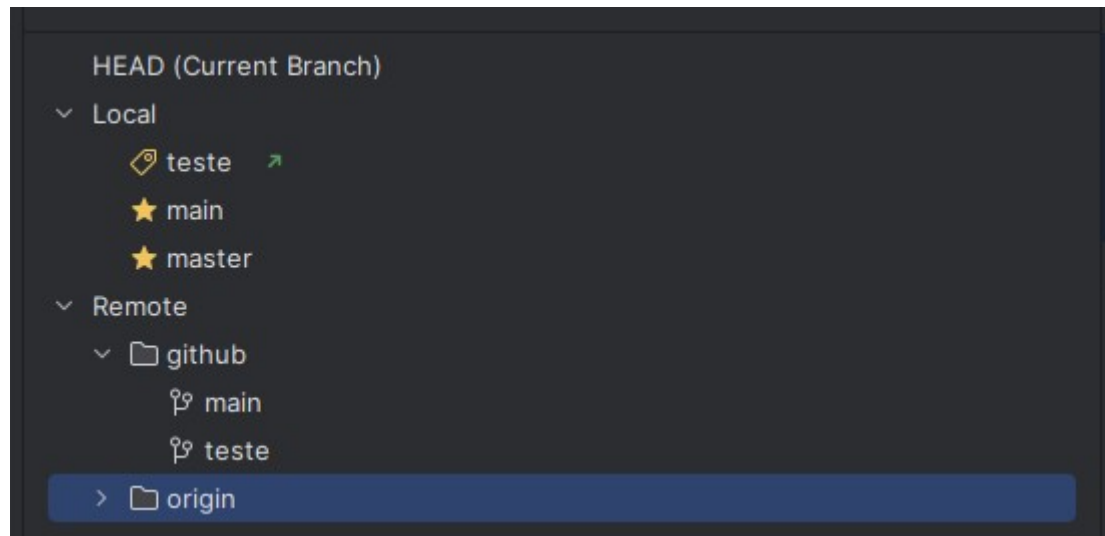
Debugger toolbar:

Evaluate expression (Enter) or add a watch (Ctrl+Shift+Enter)

Debugger output:

- > this = {Produto@788} "\nNome: Teste\n\nPreço: \$100.0\nCódigo: 85-7542-239-1" ... View
 - preco = 100.0
 - codigo = "85-7542-239-1"
 - nome = "Teste\n" ... View

5 -



Introdução

Desenvolvimento de software pode ser entendido como o processo de construção de sistemas para computadores, através de códigos de programação. Esta tarefa é comumente realizada por um desenvolvedor ou uma equipe de desenvolvedores, especializados em linguagens de programação.

Importância de testar o software

Testar um programa é importante para identificar buga e outros tipos de erros, principalmente se for em sua fase de desenvolvimento, visto que o custo para repará-lo será múltiplas vezes menor do que identificá-los quando já encontra-se em produção, pondo em risco a reputação da empresa que o desenvolveu.

Diferentes tipos de testes

Testes de unidade

Testes de unidades é realizado no próprio código fonte do programa, para se avaliar o bom funcionamentos de métodos e funções pertencentes as classes, componentes ou módulos em uso pelo programa. Geralmente, esse tipo de teste é de baixo custo e podem ser executados rapidamente.

Testes de integração

Os testes de integração são usados para verificar como diferentes módulos e serviços da aplicação se comportam juntos. Esses testes exigem que outras partes da aplicação estejam em execução e por isso eleva-se o custo de sua realização.

Testes funcionais

Os testes funcionam verificam a saída das ações executadas no programa para confirmar que estão de acordo com os requisitos de negócio.

Testes de ponta a ponta

Testes de ponta a ponta possuem como finalidade simular o uso do programa pela perspectiva do usuário em um ambiente completo. Vários fluxos de usuários são testados para verificar se estão funcionando de acordo com o esperado e esses fluxos podem variar do mais simples até cenários mais complexos.

Testes de aceitação

O teste de aceitação são feitos para verificar se o sistema atende os requisitos de negócios. Para este tipo de teste, é necessário que toda a aplicação já esteja ativo e em execução, e tem como foco principal se colocar na perspectiva do usuário para confirmar que as metas foram alcançadas.

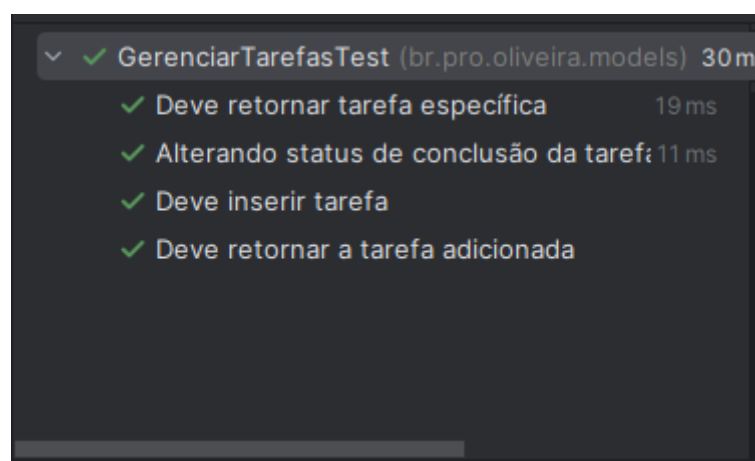
Testes de desempenho

Teste de desempenho busca avaliar o desempenho do sistema a partir de uma determinada carga de trabalho específica. Ele ajuda a medir a confiabilidade, velocidade, escalabilidade e capacidade de resposta de um programa.

Teste de fumaça

Os testes de fumaça são feitos para verificar funções básicas do programa e espera-se que os recursos principais dele estejam funcionando. Além disso, pode ser feito após uma implementação.

Execução dos testes



Configuração Junit

```
<dependencies>
  <dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter</artifactId>
    <version>5.10.2</version>
  </dependency>
</dependencies>
</project>
```

APIs Web Restful:

API Restful é utilizado por dois sistemas quando estes desejam trocar informações pela internet de maneira segura. Devido o uso de padrão de arquitetura de comunicação seguros, confiáveis e eficientes, por definir condições de como uma API deve funcionar.

Quando um cliente entra em contato com servidor através da API, os desenvolvedores deste recurso devem deixar explícito como o cliente deve usar na documentação da API. Geralmente são aplicados os seguintes méritos: get, post, put e delete.

HTTP e HTTPS

HTTP é o Protocolo de Transferência de Hipertexto que serve como base para que todos os clientes consigam navegar pela internet e solicitar dados de um servidor. Contudo, este protocolo não contém medidas de segurança, e com isso surgiu o HTTPS, que conta com princípios de segurança, como confidencialidade, integridade e autenticação.

Bibliografia

- <https://aws.amazon.com/pt/what-is/restful-api/>
- PITTET STERN. Diferentes tipos de testes de software. Disponível em: <https://www.atlassian.com/br/continuous-delivery/software-testing/types-of-software-testing>. 02/03/2024.
- https://github.com/AirtonOliveira20/Airton_de_Oliveira_DR3_AT_API
- https://github.com/AirtonOliveira20/Airton_de_Oliveira_DR3_AT_Client