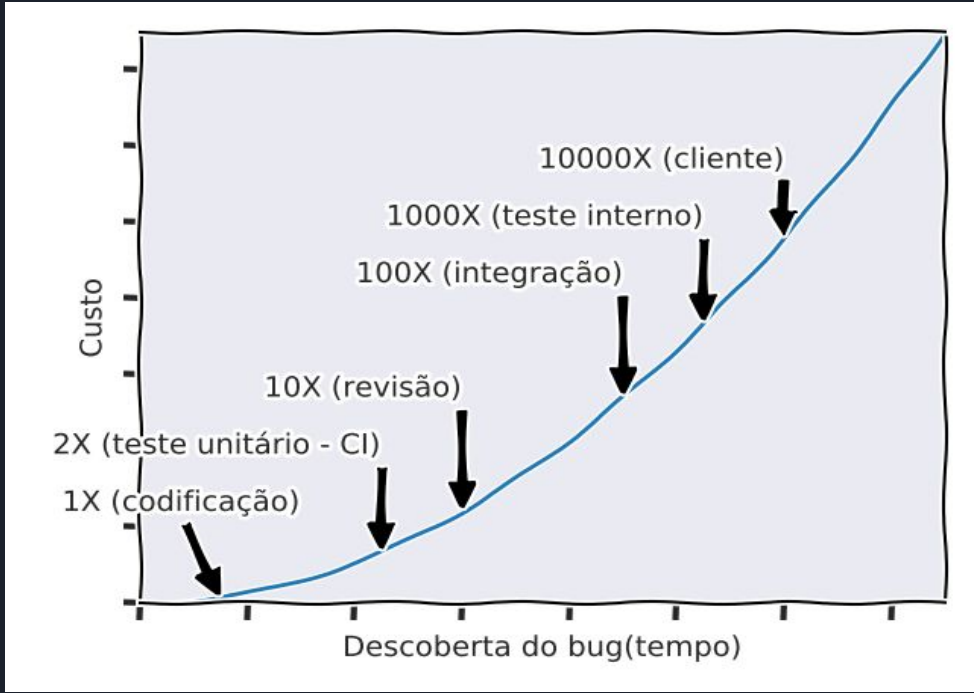
Testes unitários em Delphi

Cristiano W. Araújo
Matheus Correa

Porque testes unitários ?

PROBLEMA: Convencer colegas e superiores que usar testes unitários é uma boa solução para vários casos.

- Shift Left
- Execução rápida
- Como fazer ?



Exemplo de código:

```
function SimpleMovingAverage(nPeriod: Integer; lstPrices: TList<Double>):
Double;
var
  nIndex: Integer;
  nLenght : Integer;
begin
  try
    Result := 0;
    nLength  := lstPrices.Count;
    if (nPeriod > 0) and (nLength >= nPeriod) then
      begin
        for nIndex := 1 to nPeriod do
          Result := Result + lstPrices[nLength - nIndex];

        Result := Result / nPeriod;
      end;
  except
    Result := 0;
  end;
end;
```

Exemplo de um teste simples

```
procedure TUtilsTest.SimpleMovingAverage_InvalidPeriod;
var
  sResult: Double;
  sExpected : Double;
  nInvalidPeriod: Integer;
begin

  // Arrange
  nInvalidPeriod := 0;
  sExpected := 0;

  // Act
  sResult := Utils.SimpleMovingAverage(nInvalidPeriod, m_lstPrices);

  // Assert
  Assert.AreEqual(sExpected, sResult);
end;
```

Teste com múltiplos casos de uso

```
procedure TUtilsTest.SimpleMovingAverage_VerifyResult(sPrice1, sPrice2, sPrice3,
sPrice4, sPrice5, sExpected: Double);
const
  c_nPeriod = 5;
var
  sResult: Double;
begin
  // Arrange
  m_lstPrices.Add(sPrice1);
  m_lstPrices.Add(sPrice2);
  m_lstPrices.Add(sPrice3);
  m_lstPrices.Add(sPrice4);
  m_lstPrices.Add(sPrice5);

  // Act
  sResult := Utils.SimpleMovingAverage(c_nPeriod, m_lstPrices);

  // Assert
  Assert.AreEqual(sExpected, sResult);
end;
```

Teste com múltiplos casos de uso (declaração)

```
type
  [TestFixture]
  TUtilsTest = class
  private
    m_lstPrices: TList<Double>;
    procedure LoadFromCSV(const strFileName: String; out lstOutList: TList<Double>);
  public
    [Setup]
    procedure Setup;
    [TearDown]
    procedure TearDown;
    [Test]
    procedure SimpleMovingAverage_InvalidPeriod;
    [Test]
    [TestCase('TestCase0', '25,25,25,25,25,25')]
    [TestCase('TestCase1', '20,25,24,23,25,23.4')]
    [TestCase('TestCase2', '11,15,10,9,5,10')]
    procedure SimpleMovingAverage_VerifyResult(sPrice1, sPrice2, sPrice3,
                                               sPrice4, sPrice5, sExpected: Double);
  end;

implementation
...
```

Teste com múltiplos casos de uso (carga externa)

```
procedure TUtilsTest.SimpleMovingAverage_ValidateAverage;
var
  lstExpected: TList<Double>;
  sResult: Double;
  nIndex: Integer;
  nLenght: Integer;
begin
  // Arrange
  lstExpected := TList<Double>.Create;
  try
    LoadFromCSV('PETR4_average', lstExpected);
    LoadFromCSV('PETR4', m_lstPrices);

    nLenght := m_lstPrices.Count - 1;

    for nIndex := 0 to nLenght do
      begin
        // Act
        sResult := Utils.SimpleMovingAverage(nIndex + 1, m_lstPrices);
        // Assert
        Assert.AreEqual(lstExpected[nLenght - nIndex], sResult);
      end;
  finally
    lstExpected.Free;
  end;
end;
```

Exemplo de um teste com Mock

Problemas com Mock

Trabalho demais → Não use Mock para tudo.

Muitas mudanças no código → Requer interfaces.

Muitos mocks desnecessários → Estruture bem o código.

Próximos Passos:

- Rodar automaticamente
- Rodar em todos setups

Dunit com integração contínua

```
unittest:
    stage: test
    script:
        - call
run_tests.bat
    tags:
        - delphi
```

```
@SET returncode=0

SET DPF=%PROGRAMFILES(X86)%
if "%DPF%"=="" (
    SET DPF="%PROGRAMFILES%"
)
IF EXIST "%DPF%\Embarcadero\Studio\20.0\bin\rsvars.bat" (
  ECHO Found Delphi 10.3 Rio
  CALL "%DPF%\Embarcadero\Studio\20.0\bin\rsvars.bat"
) ELSE (
IF EXIST "%DPF%\Embarcadero\Studio\19.0\bin\rsvars.bat" (
  ECHO Found Delphi 10.2 Tokyo
  CALL "%DPF%\Embarcadero\Studio\19.0\bin\rsvars.bat"
))

MSBuild.exe TDCProject.groupproj /t:Build /p:config=Release /p:plataform="32-bit Windows"
/property:DCC_Define="$$(DCC_Define);CI"
if %errorlevel% neq 0 exit /b %errorlevel%
cd bin
.\UnitTest.exe
exit /b %errorlevel%
```

DunitX e Delphi Mocks como dependências

Problema: Instalações diferentes & Versões diferentes

Solução: Empacotar direto no repositório

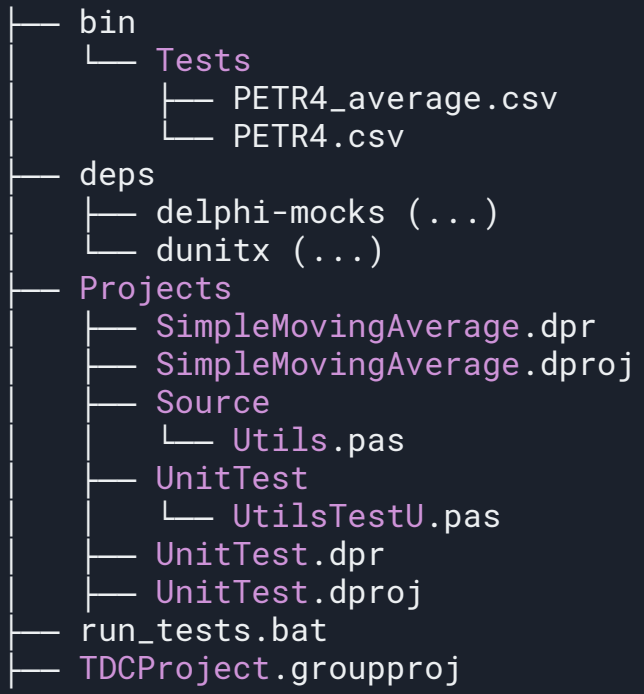
```
[submodule "deps/delphi-mocks"]
    path = deps/delphi-mocks
    url = https://github.com/VSoftTechnologies/Delphi-Mocks.git
[submodule "deps/dunitx"]
    path = deps/dunitx
    url = https://github.com/VSoftTechnologies/DUnitX.git
```

Estruturação do projeto

Groupproj

Casos de teste comitados

Script de execução commitado



Cobertura de testes em Delphi

Compilar com o .map

Usar o programa Delphi-Coverage

https://github.com/magicmonty/delphi-code-coverage

```
32 begin
33   try
34     Result := 0;
35     nLenght  := lstPrices.Count;
36     if (nPeriod > 0) and (nLenght >= nPeriod) then
37       begin
38         for nIndex := 1 to nPeriod do
39           Result := Result + lstPrices[nLenght - nIndex];
40
41         Result := Result / nPeriod;
42       end;
43   except
44     Result := 0;
45   end;
46 end;
47
```

```
codecoverage.exe -e bin\UnitTest.exe ^
                 -m bin\UnitTest.map^
                 -u Utils -od reports -html
                 -dproj Projects\UnitTest.dproj
```

Ok. Quero testar, como convenço o gerente ?

Teste somente **novas** funcionalidades

Escreva testes **pertinentes** (não teste o salvamento de um arquivo)

Escreva poucos testes e deixe eles **pegarem bugs (**i.e. tenha paciência).

www.nelogica.com.br

Cristiano W. Araújo
caraujo@nelogica.com.br
Matheus Correa
mcorrea@nelogica.com.br

Carreiras: jobs.kenoby.com/nelogica

linkedin.com/company/nelogica