**Pontifícia Universidade Católica de Minas Gerais**

**Unidade Barreiro**

**Sistemas de Informação**



TDD – Desenvolvimento Dirigido por Testes

Brazucas

**Alunos:**   
Allan Junior Nogueira Santos,  
Eduardo Dias Salomão,  
Tales Moreira Correa,  
Thiago Rodrigues de Araújo.  
 **Professora:**   
Maria Augusta Vieira Santos.

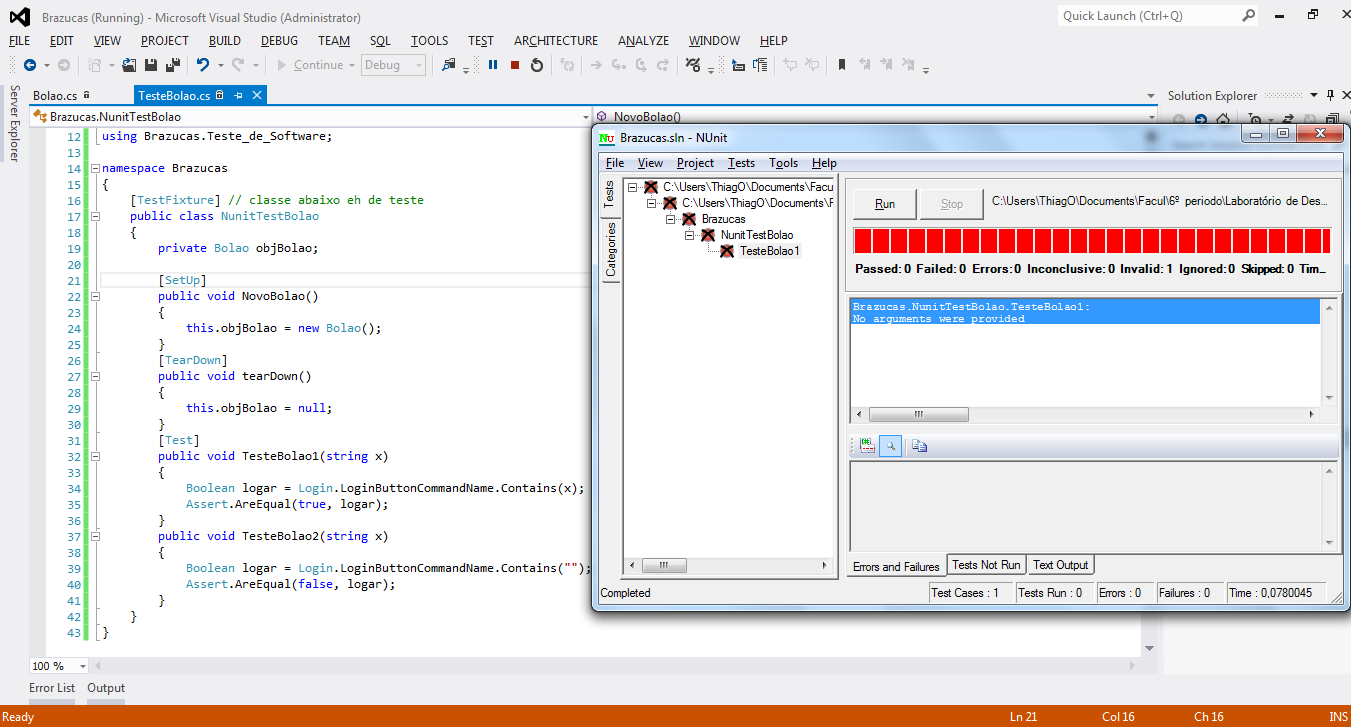
Maio 2013

TDD – Desenvolvimento Dirigido por Teste



Utilizaremos este recurso  
Teste que falha, Teste que passa, Refatora

* 1ª regra: Garantir que todos os usuários cadastrados tem acesso ao Bolão  
    
  Falha!!!



Teste com falha!

Código:  
using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using NUnit.Framework;

using System.Web.UI;

using System.Web.Security;

using System.Web.UI.Adapters;

using System.Web.UI.WebControls;

using System.Web.UI.HtmlControls;

using System.Data;

using Brazucas.Teste\_de\_Software;

namespace Brazucas

{

[TestFixture] // classe abaixo eh de teste

public class NunitTestBolao

{

private Bolao objBolao;

[SetUp]

public void NovoBolao()

{

this.objBolao = new Bolao();

}

[TearDown]

public void tearDown()

{

this.objBolao = null;

}

[Test]

public void TesteBolao1(string x)

{

Boolean logar = Login.LoginButtonCommandName.Contains(x);

Assert.AreEqual(true, logar);

}

public void TesteBolao2(string x)

{

Boolean logar = Login.LoginButtonCommandName.Contains("");

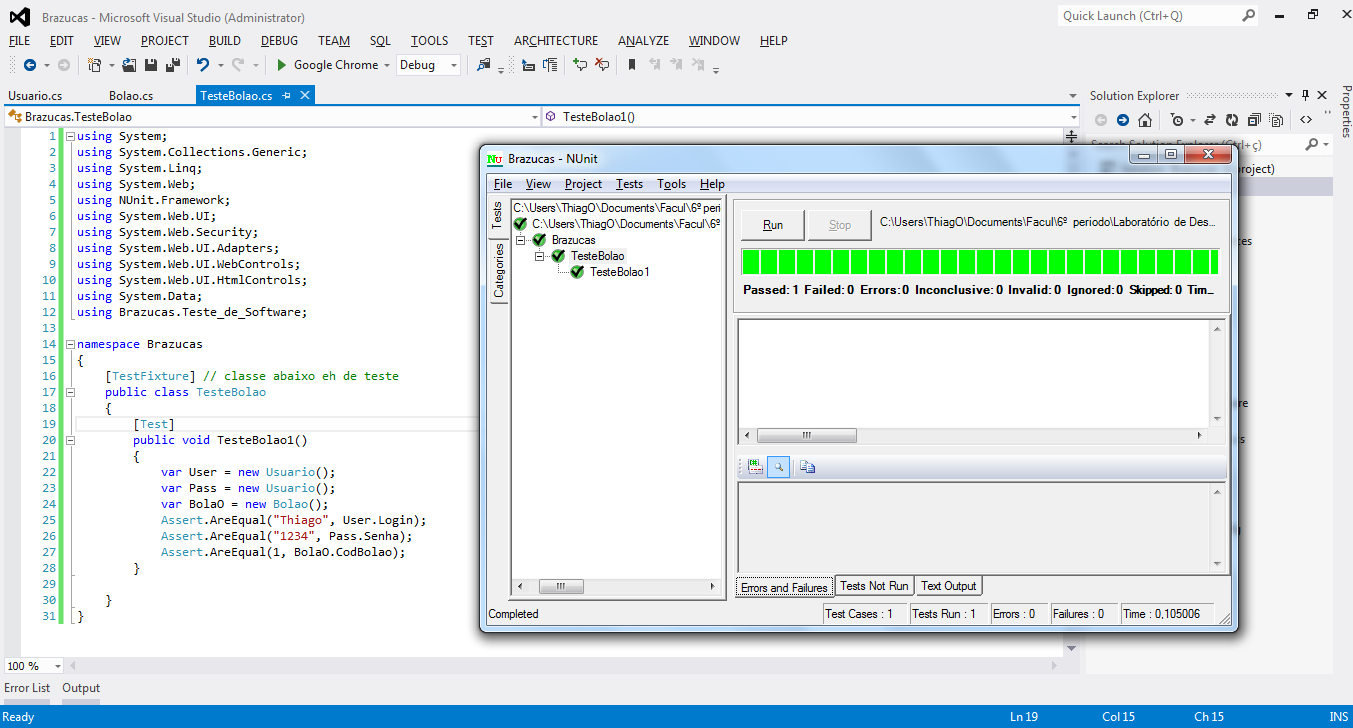
Assert.AreEqual(false, logar);

}

}

}

Agora o teste que passa!!!

**

Teste que Passa!!!

Código:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using NUnit.Framework;

using System.Web.UI;

using System.Web.Security;

using System.Web.UI.Adapters;

using System.Web.UI.WebControls;

using System.Web.UI.HtmlControls;

using System.Data;

using Brazucas.Teste\_de\_Software;

namespace Brazucas

{

[TestFixture] // classe abaixo eh de teste

public class TesteBolao

{

[Test]

public void TesteBolao1()

{

var User = new Usuario();

var Pass = new Usuario();

var BolaO = new Bolao();

Assert.AreEqual("Thiago", User.Login);

Assert.AreEqual("1234", Pass.Senha);

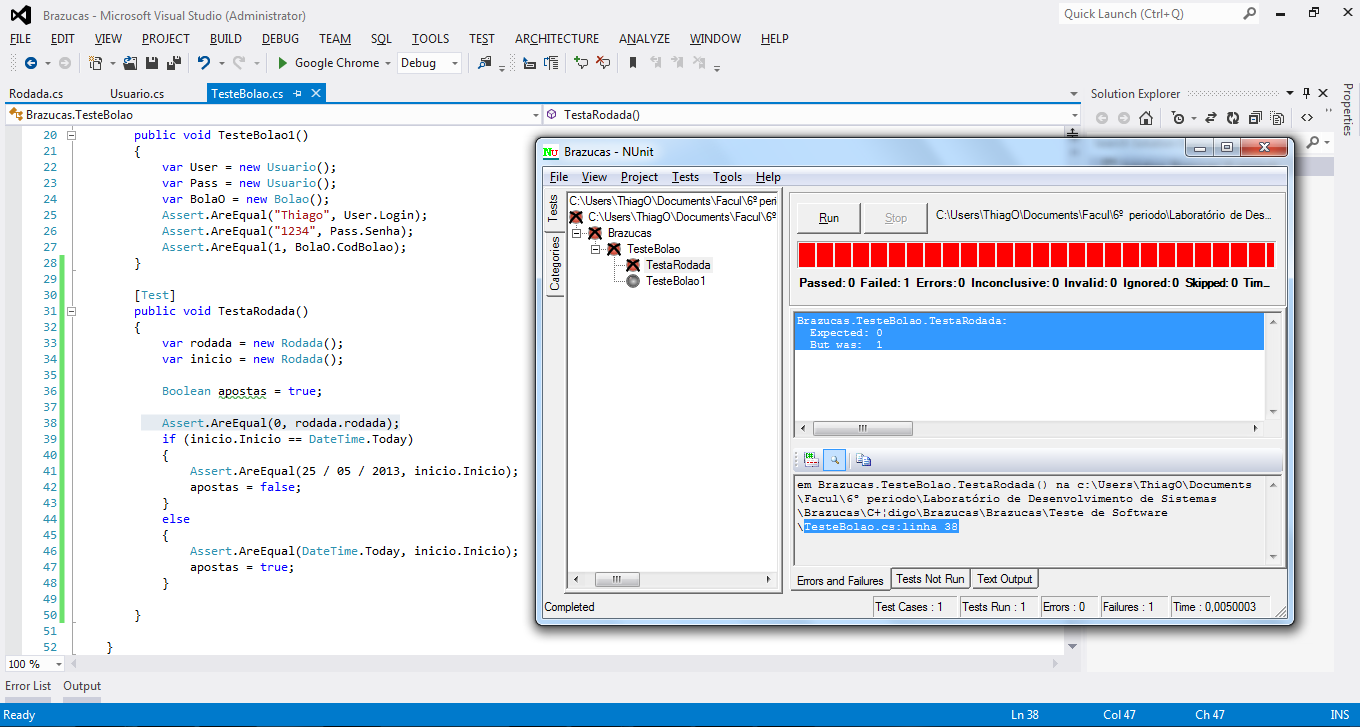
Assert.AreEqual(1, BolaO.CodBolao);

}

}

}

* 2ª regra: As apostas de uma rodada devem ser encerradas 24 horas antes do início da mesma. Após o encerramento das apostas os participantes conseguem visualizar as apostas dos demais participantes.



Teste Falha!!!

Código:

[Test]

public void TestaRodada()

{

var rodada = new Rodada();

var inicio = new Rodada();

Boolean apostas = true;

Assert.AreEqual(0, rodada.rodada);

if (inicio.Inicio == DateTime.Today)

{

Assert.AreEqual(25 / 05 / 2013, inicio.Inicio);

apostas = false;

}

else

{

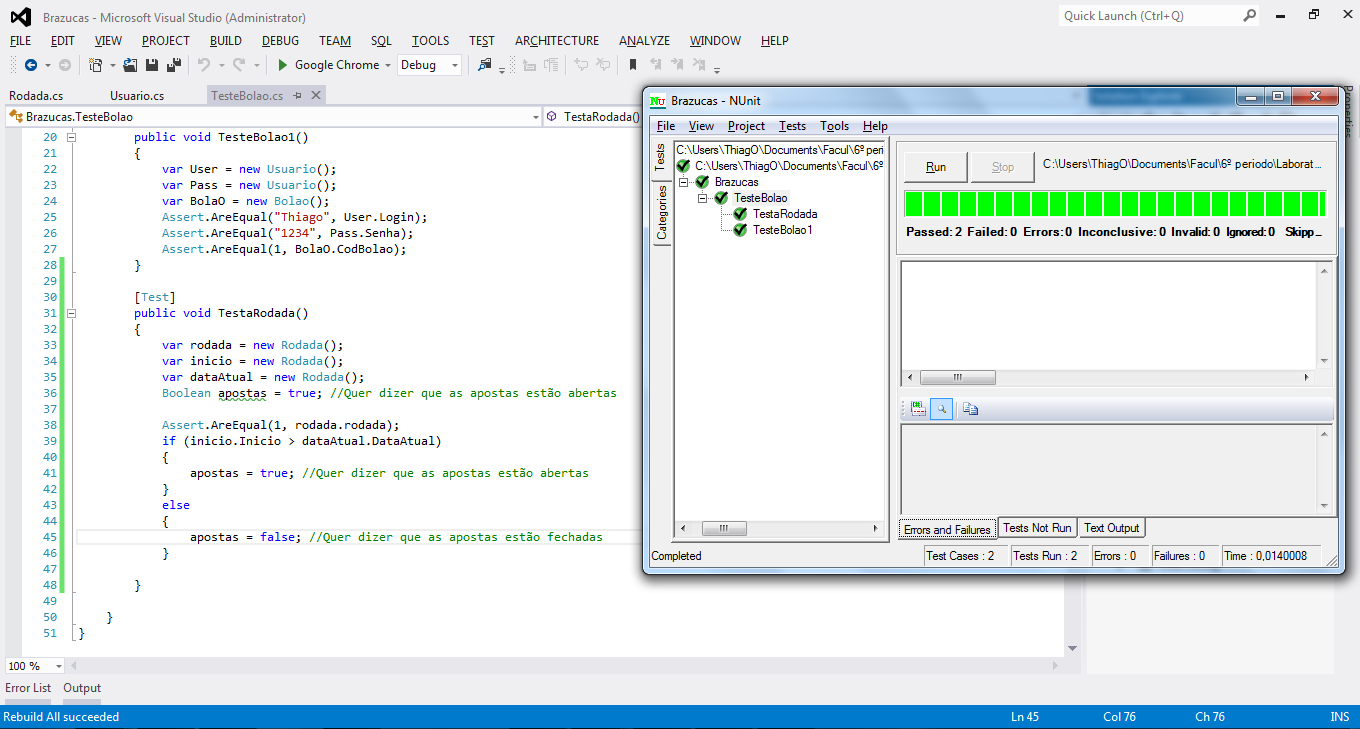
Assert.AreEqual(DateTime.Today, inicio.Inicio);

apostas = true;

}

}

Agora o teste Passa!!!

**

Passa!!!

Código:

[Test]

public void TestaRodada()

{

var rodada = new Rodada();

var inicio = new Rodada();

var dataAtual = new Rodada();

Boolean apostas = true; //Quer dizer que as apostas estão abertas

Assert.AreEqual(1, rodada.rodada);

if (inicio.Inicio > dataAtual.DataAtual)

{

apostas = true; //Quer dizer que as apostas estão abertas

}

else

{

apostas = false; //Quer dizer que as apostas estão fechadas

}

}

Regra 3: Ranking

Não compila, pois falta as classes Ranking e Jogos, e os métodos jogos, Resultado, Status.

[Test]

public void TestaRanking()

{ var rodada = new Rodada();

var user = new Usuario();

var rank = new Ranking();

var jogo = new Jogos();

if (rodada.Status == "fechada")

{

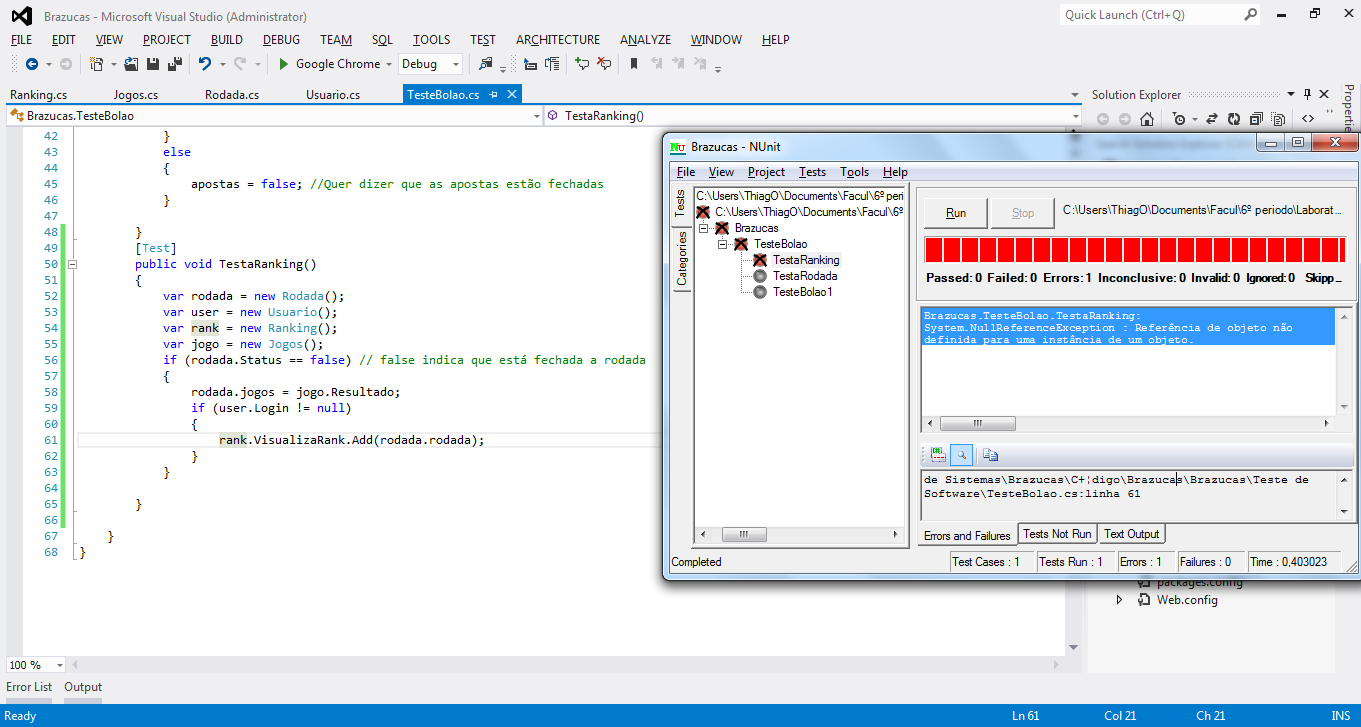
rodada.jogos = jogo.Resultado;

if (user.Login != null)

{

rank.VisualizaRank; }}}

Agora compila, mas não passa!

**

Falha!

Código:

[Test]

public void TestaRanking()

{

var rodada = new Rodada();

var user = new Usuario();

var rank = new Ranking();

var jogo = new Jogos();

if (rodada.Status == false) // false indica que está fechada a rodada

{

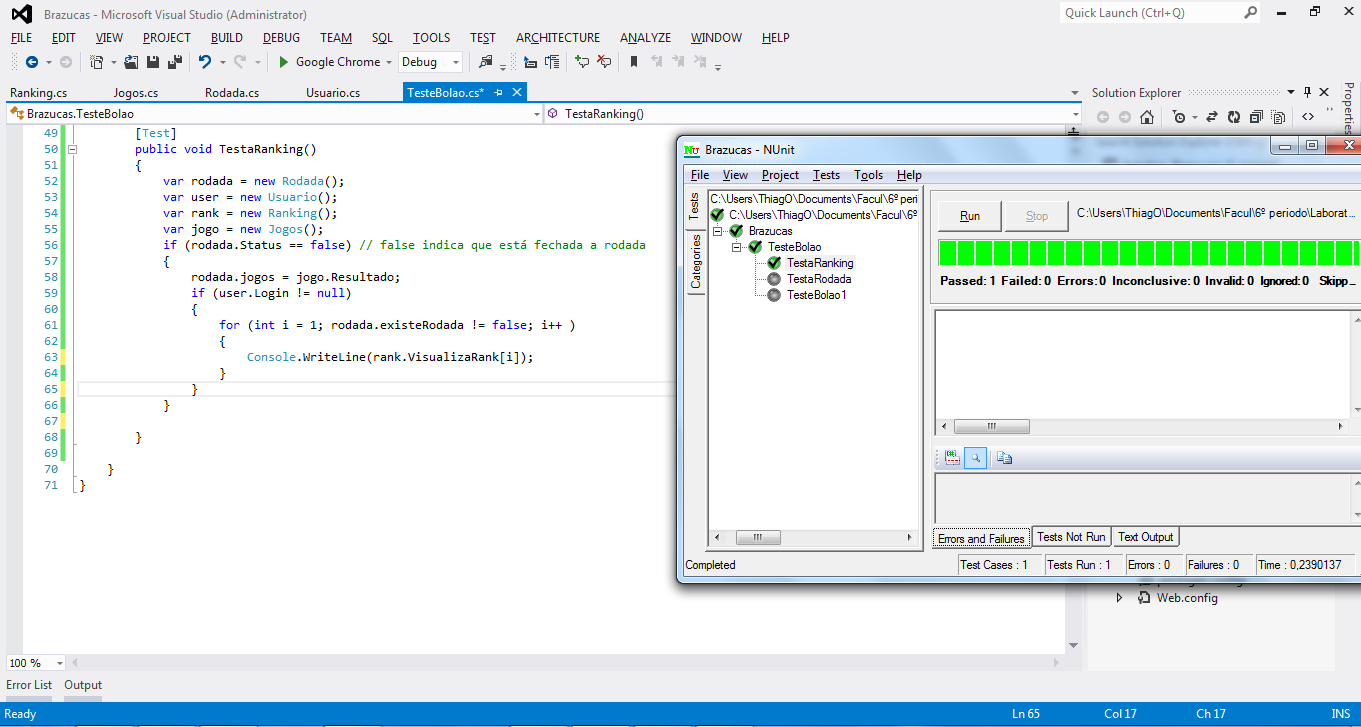
rodada.jogos = jogo.Resultado;

if (user.Login != null)

{

rank.VisualizaRank.Add(rodada.rodada);

}}}

Agora compila e passa:**

Passou!!!

Código:

[Test]

public void TestaRanking()

{

var rodada = new Rodada();

var user = new Usuario();

var rank = new Ranking();

var jogo = new Jogos();

if (rodada.Status == false) // false indica que está fechada a rodada

{

rodada.jogos = jogo.Resultado;

if (user.Login != null)

{

for (int i = 1; rodada.existeRodada != false; i++ )

{

Console.WriteLine(rank.VisualizaRank[i]);

}

}

}

}

Refatoramos agora e criamos novamente um teste que passa, mas não compila:

[Test]

public void TestaRanking()

{

var rodada = new Rodada();

var user = new Usuario();

var rank = new Ranking();

var jogo = new Jogos();

if (rodada.Status == false) // false indica que está fechada a rodada

{

rodada.jogos = jogo.Resultado;

if (user.Login != null)

{

for (int i = 1; rodada.existeRodada != false; i++ )

{

Console.WriteLine(rank.VisualizaRank[i]);

}

}

}

if (rodada.Status == false) // false indica que está fechada a rodada

{

rodada.jogos = jogo.Resultado;

if (user.Login != null)

{

for (int i = 1; rodada.existeRodada != false; i++)

{

Console.WriteLine(rank.VisualizaPontuacao[i]);

if (i != 1)

{ Console.WriteLine("A diferença de pontos entre esse usuário e o primeiro é de: " + rank.VisualizaPontuacao[i] - rank.VisualizaPontuacao[1]); }

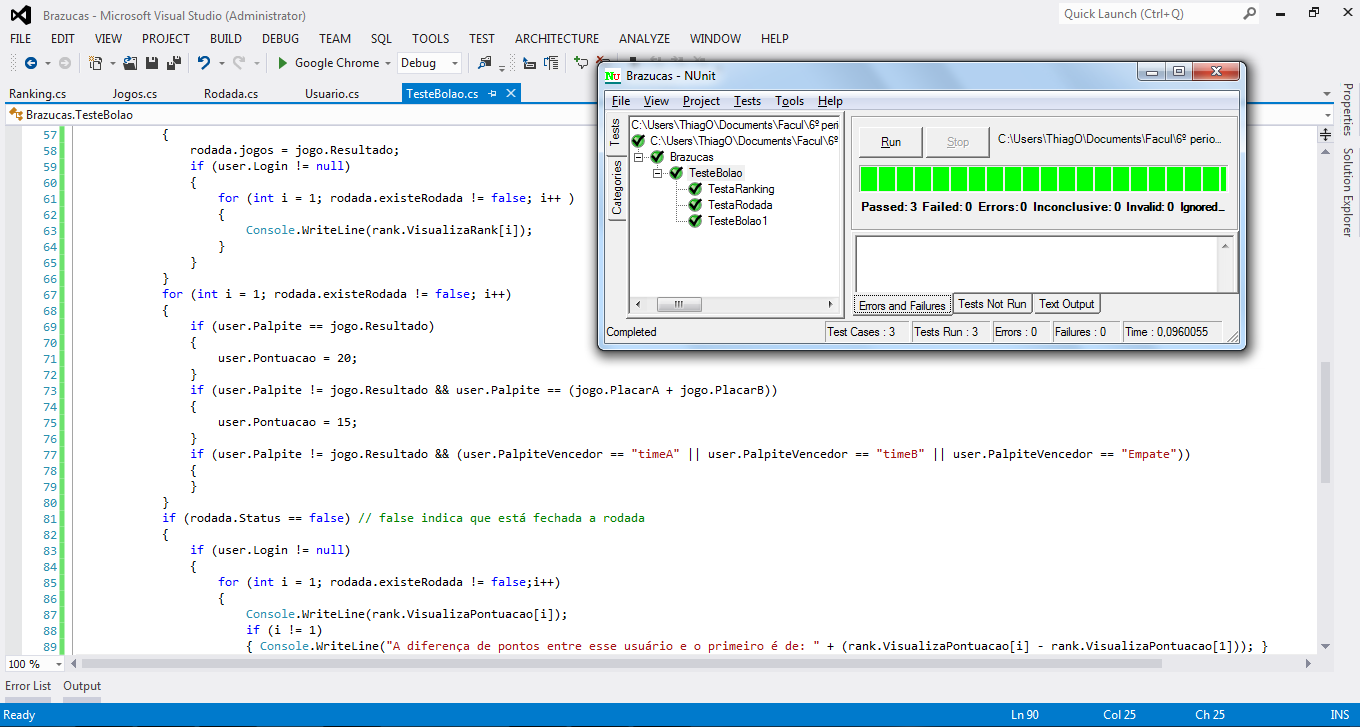
}

}

}

}

Agora criamos um que compila e que funciona:

**

Código:

[Test]

public void TestaRanking()

{

var rodada = new Rodada();

var user = new Usuario();

var rank = new Ranking();

var jogo = new Jogos();

if (rodada.Status == false) // false indica que está fechada a rodada

{

rodada.jogos = jogo.Resultado;

if (user.Login != null)

{

for (int i = 1; rodada.existeRodada != false; i++ )

{

Console.WriteLine(rank.VisualizaRank[i]);

}

}

}

for (int i = 1; rodada.existeRodada != false; i++)

{

if (user.Palpite == jogo.Resultado)

{

Assert.AreEqual( 20, user.Pontuacao++);

}

if (user.Palpite != jogo.Resultado && user.Palpite == (jogo.PlacarA + jogo.PlacarB))

{

Assert.AreEqual( 15, user.Pontuacao++);

}

if (user.Palpite != jogo.Resultado && (user.PalpiteVencedor == "timeA" || user.PalpiteVencedor == "timeB" || user.PalpiteVencedor == "Empate"))

{

Assert.AreEqual( 10, user.Pontuacao++);

}

if (user.Palpite != jogo.Resultado && (user.PalpiteVencedor != "timeA" && user.PalpiteVencedor != "timeB" && user.PalpiteVencedor != "Empate"))

{

Assert.AreEqual( 0, user.Pontuacao++);

}

}

if (rodada.Status == false) // false indica que está fechada a rodada

{

if (user.Login != null)

{

for (int i = 1; rodada.existeRodada != false;i++)

{

Console.WriteLine(rank.VisualizaPontuacao[i]);

if (i != 1)

{ Console.WriteLine("A diferença de pontos entre esse usuário e o primeiro é de: " + (rank.VisualizaPontuacao[i] - rank.VisualizaPontuacao[1])); }

}

}

}

}