Relatório de Cobertura de Testes

Para a realização de testes unitários da aplicação proposta foi utilizado o xUnit e Mocks. Com auxílio da extensão *Fine Code Covernage* para Visual Studio, foi realizado mapeamento de cada uma das classes quanto a cobertura de testes em seus métodos. Conforme os testes eram sendo implementados, os gráficos e percentuais eram atualizados.

Uso de Mock

```
1 referência
public class DadosRepositorioTeste
{
    private DadosRepositorio _dadosRepositorio;

    private Mock<IRegistrosRepositorio> _mockRegistrosRepositorio;

    0 referências
    public DadosRepositorioTeste()
    {
        _mockRegistrosRepositorio = new Mock<IRegistrosRepositorio>();

        _dadosRepositorio = new DadosRepositorio(_mockRegistrosRepositorio.Object);
    }
```

Exemplo de Testes Desenvolvidos

```
[Fact]
| 0 referências
public void AdicionarNovaListaAtivos_RetornandoListaComMesmaQtdDeItensInformada()
{
    _mockRegistrosRepositorio.Setup(x => x.GerarCodigoLetrasNumerosAleatorio()).Returns("AAA1234");
    var qtdItensLista = _dadosRepositorio.AdicionarNovaListaAtivos(5);
    Assert.Equal(5, qtdItensLista.Count());
}

[Fact]
| 0 referências
public void AdicionarNovaListaAtivos_NaoInformandoQtdDeItens()
{
    var exception = Assert.Throws<Exception>(() => _dadosRepositorio.AdicionarNovaListaAtivos(0));
    Assert.Equal("Nenhuma quantidade informada!", exception.Message);
}
```

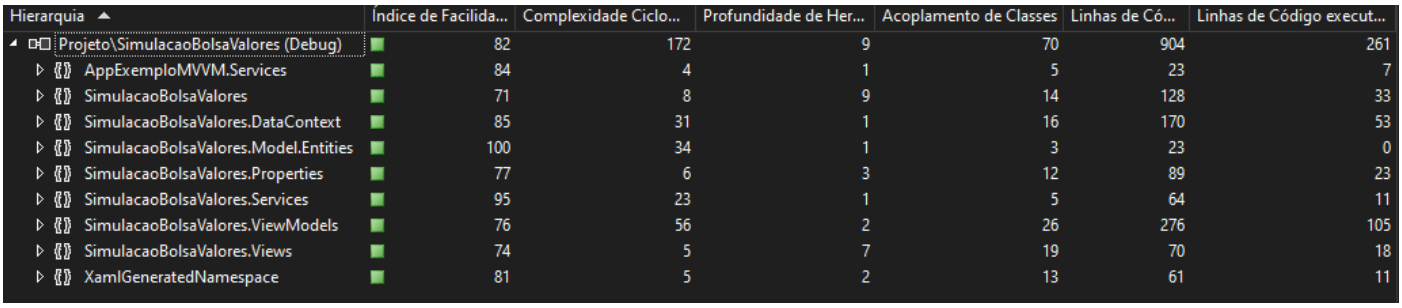
Resultado Final

Name	Covered	Uncovered	Coverable	Total	Line coverage	
− SimulacaoBolsaValores	252	0	252	541	100%	
AppExemploMVVM.Services.CommandBase	5	0	5	27	100%	
SimulacaoBolsaValores.DataContext.DadosRepositorio	69	0	69	99	100%	
SimulacaoBolsaValores.DataContext.RegistrosRepositorio	36	0	36	64	100%	
SimulacaoBolsaValores.Model.Entities.AtivoED	17	0	17	30	100%	
SimulacaoBolsaValores.Services.AtivoController	27	0	27	59	100%	
SimulacaoBolsaValores.ViewModels.InicioViewModel	98	0	98	262	100%	
+ SimulacaoBolsaValores.Testes	221	0	221	467	100%	

Conforme mostra o gráfico, 100% do código foi coberto com teste unitários.

Outras Métricas do Código

As a seguir foram geradas no Visual Studio. Segue abaixo Resultado:



Hierarquia	Índice de Facilida...	Complexidade Ciclo...	Profundidade de Her...	Acoplamento de Classes	Linhas de Có...	Linhas de Código execut...
Projeto\SimulacaoBolsaValores (Debug)	82	172	9	70	904	261
AppExemploMVVM.Services	84	4	1	5	23	7
SimulacaoBolsaValores	71	8	9	14	128	33
SimulacaoBolsaValores.DataContext	85	31	1	16	170	53
SimulacaoBolsaValores.Model.Entities	100	34	1	3	23	0
SimulacaoBolsaValores.Properties	77	6	3	12	89	23
SimulacaoBolsaValores.Services	95	23	1	5	64	11
SimulacaoBolsaValores.ViewModels	76	56	2	26	276	105
SimulacaoBolsaValores.Views	74	5	7	19	70	18
XamlGeneratedNamespace	81	5	2	13	61	11

Índice de Facilidade de Manutenção

Esta métrica calcula um valor de índice de 0 a 100 que representa a facilidade de manutenção do código. Um valor alto significa uma melhor capacidade de manutenção. O cálculo do índice e baseado em outras métricas como Complexidade Ciclomática e Linhas de Código.

Escala	Definição	Símbolo
20 a 100	Boa	
10 a 19	Razoável	
0 a 9	Baixa	

Complexidade Ciclomática

Esta métrica mede a complexidade estrutural do código. Ela é criada calculando o número de caminhos de código diferente no fluxo do programa. Um programa que tem o fluxo de controle complexo exigira mais testes para atingir a cobertura de código em boas condições e será menos passível de manutenção.

Escala	Definição
1 a 10	Simples programa, sem muitos riscos.
11 a 20	Programa mais complexo, risco médio.
21 a 50	Programa complexo, alto risco.
> 50	Programa Intestável, risco muito alto.

Profundidade de Herança

Esta métrica indica o número de níveis dos quais uma classe herda métodos e atributos. Quando maior a hierarquia, mais difícil fica a compreensão de onde estes estão definidos. Deve-se evitar uma profundidade maior do que 5.

Acoplamento de Classes

Este indicador mede o acoplamento de classes por meio de parâmetros, variáveis locais, tipos de retorno, chamadas de método, instanciações genéricas ou modelo, classes base, implementações de interface, campos definidos em tipos externos e decoração de atributo. Um acoplamento forte significa que as classes relacionadas precisam conhecer detalhes uma das outras, as alterações se propagam pelo sistema e o mesmo fica difícil de entender.

Linhas de Código

Esta métrica mostra o número aproximado de linhas de código. Não existe um número exato de linhas para definir se está bom ou ruim. Um número muito alto pode ser um aviso de que um método está muito extenso e deve ser dividido. Também pode indicar que o método pode ser de difícil manutenção.

Referências

MSDN, M. (2012). Valores de métricas de código. Disponível em:

http://msdn.microsoft.com/pt-br/library/bb385914.aspx