**pontifícia universidade católica de minas gerais**

**Instituto de Ciências Exatas e Informática**

**Curso de Sistemas de Informação**

Amanda Rhodes Alves

Gabriel Kennedy Almeida Resende

Kennedy Marciano Fortunato

Pedro Henrique Amado de Andrade

Zamur Borges Vedovatto

**PLANO DE TESTES**

**Estacionamento park-away**

**Belo Horizonte**

**2017**

SUMÁRIO

[1. Introdução 3](#_Toc497157481)

[2. Módulos testados 3](#_Toc497157482)

[3. Métodos que serão testados 3](#_Toc497157483)

[4. Características que serão testadas 4](#_Toc497157484)

[5. Abordagem 4](#_Toc497157485)

[A. Níveis de teste 4](#_Toc497157486)

[b. Controle de versão e configuração 4](#_Toc497157487)

[c. Ferramentas para realização dos testes 4](#_Toc497157488)

[6. Testes 5](#_Toc497157489)

[7. Conclusão 8](#_Toc497157490)

1. Introdução

A realização de **testes é uma atividade fundamental dentro do desenvolvimento de sistemas. A principal meta por trás do teste é garantir que o produto atenda ao que foi especificado durante o projeto.**

Esse plano de testes tem por objetivo especificar e detalhar os testes que serão realizados no projeto Park-Away, um estacionamento de veículos com funcionamento diário. Os testes irão abranger todos os módulos do projeto e as classes principais que foram criadas seguindo paradigmas de Programação Orientada a Objetos.

O plano contemplará os seguintes tipos de teste: de unidade e de integração.

Ao final dos testes será possível afirmar se o projeto garante a integridade dos módulos e se fornece dados confiáveis seguindo a regra de negócios do estacionamento.

1. Módulos testados

* Classe Estacionamento (Estacionamento.cs) v1.0.0.
* Classe Veículo (Veiculo.cs) v1.0.0.

1. Métodos que serão testados

Serão testadas as instâncias das classes Estacionamento e Veículo e os métodos e procedimentos da classe Estacionamento listados abaixo:

* public bool novoVeiculo(string placa, int hora, int minutos)

Adiciona um novo veículo a uma lista.

* public double saidaVeiculo(string placa)

Retira um veículo da lista.

* public double calculaValor(Veiculo veic)

Calcula o valor que será cobrado para cada saída de veículo estacionado.

* public Veiculo buscarVeiculo(string placa)

Verifica se um determinado veículo está estacionado no estacionamento em questão.

* public bool placaValida (string placa)

Verifica se a placa inserida possui pelo menos 1 (um) inteiro e pelo menos (1) uma letra

1. Características que serão testadas

* Entrada e registro de novos veículos no estacionamento;
* Cálculo do valor a ser pago pelo tempo decorrido do veículo dentro do estacionamento;
* Verificação de presença de determinado veículo no estacionamento.
* Saída de veículos do estacionamento.

1. Abordagem
   1. Níveis de Teste

Os testes contemplarão testes de unidade e de integração. Os desenvolvedores serão os responsáveis pela criação e depuração dos testes, que serão realizados tanto individualmente como em equipe. Para cada caso de teste, vinculamos um método de teste codificado.

* 1. Controle de Versão e Configuração

Para que haja conhecimento e controle de todo o projeto por todos da equipe, utilizaremos o controle de versão utilizando o Git e manteremos o repositório público no GitHub. Também utilizaremos a ferramenta SourceTree, que dá maior agilidade no controle de versão, de alterações, de configurações e de versionamento, facilitando assim a identificação de modificações e realização de testes.

* 1. Ferramentas para Realização dos Testes

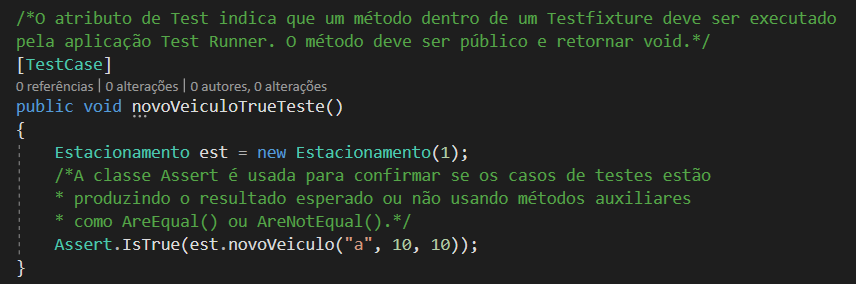
Para realização de testes de unidade e integração utilizaremos o framework Nunit, que auxilia tanto na criação dos testes quanto na análise. O NUnit é um framework livre utilizado para testes unitários, tornando possível criar casos de teste no Visual Studio, gerando resultados de forma que se permita a visualização do sucesso ou fracasso dos testes executados.

Também aproveitaremos a GUI disponível no próprio Visual Studio, utilizada para desenvolvimento do projeto na linguagem C#.

1. Testes

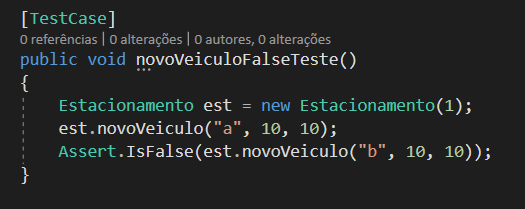
A seguir listamos os testes aprovados apresentados no gerenciador de Testes utilizando NUnit no Visual Studio, assim como o resultado obtido.

Teste 1: novoVeiculoTrueTeste



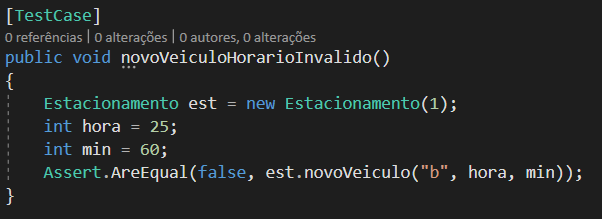
Verifica a assertividade de inclusão de uma instância da classe Veículo em uma lista genérica de Veículo em método presente em uma instância da classe Estacionamento.

Teste 2: novoVeiculoFalseTeste



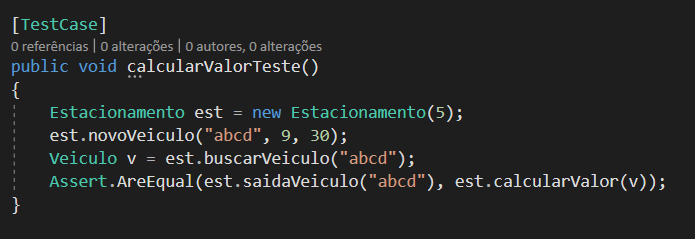
Verifica o limite de capacidade de veículos estacionados (instâncias da classe Veiculo presentes em uma lista genérica) em uma instância da classe Estacionamento.

Teste 3: novoVeiuloHorarioInvalido



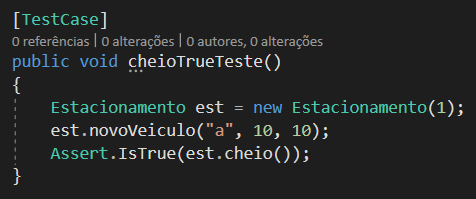
Verifica as entradas de hora e minuto na entrada de uma nova instância da classe Veiculo na lista de objetos genéricos presente em uma instância da classe Estacionamento.

Teste 4: calcularValorTeste



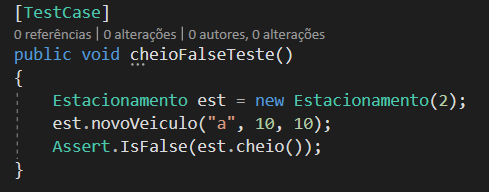
Verifica o valor final cobrado para uma instância da classe Veiculo que está sendo retirada do estacionamento (sendo removida da lista de objetos genérica presente em uma instância da classe Estacionamento).

Teste 5: cheioTrueTeste



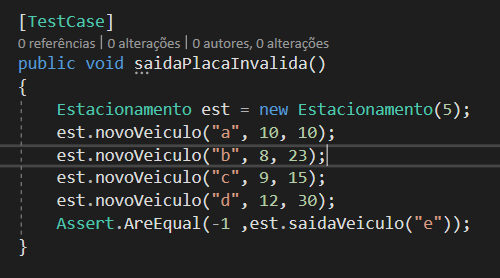
Verifica o retorno true caso a lista genérica de objetos presente em uma instância da classe Estacionamento não tenha atingido o seu limite máximo.

Teste 6: cheioFalseTeste



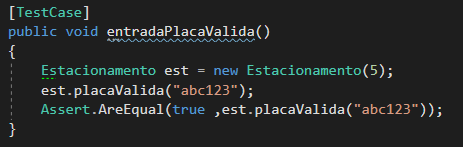
Verifica o retorno false caso a lista genérica de objetos presente em uma instância da classe Estacionamento tenha atingido o seu limite máximo.

Teste 7: saidaPlacaInvalida



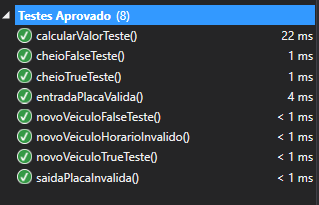
Verifica a assertividade da entrada do atributo placa de uma instância da classe Veiculo presente em uma lista genérica de objetos presente em uma instância da classe Estacionamento.

Teste 8: entradaPlacaValida



Verifica o retorno true caso a placa atenda aos requisitos mínimos de pelo menos 1 (um) inteiro e 1(uma) letra.

Resultados finais dos testes



1. Conclusão

De acordo com os testes de unidade e de integração listados acima, podemos concluir que os métodos/procedimentos criados trazem os resultados esperados de acordo com o planejado, sendo assim, possível mantermos o projeto atual para futuras implementações e atualizações.