Plano de Testes

1. Introdução

Esse plano de testes tem por objetivo especificar e detalhar os testes que serão realizados no projeto Park-away, um estacionamento de veículos que tem 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. Unidade e Integração são as camadas de testes que esse plano contemplará, e cada uma será detalhada e analisada.

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

1. Itens que serão testados

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

1. Métodos que serão testados

Serão testados além de instâncias das classes Estacionamento e Veículo, criação de objetos, os métodos e procedimentos da classe Estacionamento listados abaixo.

* + public bool novoVeiculo(string placa, int hora, int minutos);
  + public double saidaVeiculo(string placa);
  + public double calculaValor(Veiculo veic);
  + public Veiculo buscarVeiculo(string placa);

1. Características que serão testadas

* Entrada de novos veículos no estacionamento
* Cálculo do valor a ser pago pelo tempo passado
* Saída de veículos do estacionamento

1. Características que não serão testadas
2. Abordagem

6.1 Níveis de Teste

Os testes contemplarão testes de Unidade e de Integração. Os desenvolvedores serão os responsáveis pelos testes, que serão realizados tanto individualmente, como em equipe.

6.2 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 mais agilidade no controle, alterações de configurações e versionamento, facilitando assim a identificação de modificações e realização de testes.

6.3 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. Também aproveitaremos recursos disponíveis no próprio Visual Studio, que será a GUI utilizada para desenvolvimento do projeto na linguagem C#.