Netshoes

Teste para analista desenvolvedor

|  |  |
| --- | --- |
|  | Foi construído um projeto fictício com algumas regras de negócio já desenvolvidas, o teste consiste basicamente em entender o projeto e evoluir conforme as orientações citadas no decorrer deste documento.  O objetivo do teste é simular as atividades do dia-a-dia afim de avaliar as seguintes áreas:   * Desenvolvimento de novos recursos * Correções de BUG’s. * Melhorias técnicas * Analise das atividades |

Visão geral

O projeto consiste em uma tela cujo objetivo é digitar um pedido, e baseado nos dados da tela fazer a persistência dos dados.

|  |  |
| --- | --- |
|  | Dica: A base de código é bem reduzida, entenda a estrutura do projeto antes de iniciar a solução dos problemas. |

1 – Exercício (Novo recurso)

Está em desenvolvimento uma tela para emissão de notas fiscais, o cálculo dos impostos já está desenvolvido, porém é necessário fazer a persistência da nota fiscal em um arquivo XML, seguindo exatamente a mesma estrutura da classe: NotaFiscal.

O diretório onde serão gerados os XML’s é configurado pelo time de infra estrutura, o usuário não pode altera-lo (o usuário tem acesso apenas ao executável).

|  |  |
| --- | --- |
|  | Dica: Pode ser utilizado serialização de objetos para XML nesse caso. |

2 – Exercício (Novo recurso)

|  |  |
| --- | --- |
|  | Será necessário criar um banco de dados localmente utilizando o SQL Server Express. |

Na pasta SQL foi disponibilizado um conjunto de scripts, para que seja possível criar e popular o banco de dados.

Utilizar as procedures P\_NOTA\_FISCAL e P\_NOTA\_FISCAL\_ITEM para fazer a persistência dos dados no banco.  
A persistência só pode acontecer se o XML foi gerado com sucesso.

3 – Exercício (Novo recurso)

Adicionar na classe de itens da nota fiscal os campos relacionados ao imposto de IPI:

Base de cálculo de IPI: Igual ao valor total do produto.  
Alíquota de IPI: Se for brinde alíquota é igual a 0% se não brinde alíquota é igual a 10%.   
Valor de IPI: Base de cálculo \* Alíquota de IPI.

Esses campos serão persistidos na tabela de Itens de nota fiscal e também no arquivo XML.

4 – Exercício (Novo recurso)

Desenvolver uma stored procedure para ser utilizada por outros sistemas, o objetivo dessa procedure é devolver a seguinte estrutura:

CFOP | Valor Total da Base de ICMS | Valor Total do ICMS | Valor Total da Base de IPI | Valor Total do IPI

Deve estar agrupado por CFOP.

5 – Exercício (Correção de BUG)

Para o estado de origem SP e destino RO o sistema deveria definir a CFOP como 6.006, corrigir o erro no sistema para que seja definido a CFOP correta.

6 – Exercício (Correção de BUG)

A tela tem alguns pontos de melhoria a serem implementados, por exemplo: Ao salvar a tela não está limpando os campos, é possível indicar estados inválidos tanto na origem quanto no destino.

Levantar os pontos de melhoria e executar as correções.

7 – Exercício (Novo recurso)

Criar um campo desconto no item da nota fiscal.  
Preenchê-lo com a seguinte regra: Clientes do sudeste (campo “EstadoDestino”) tem 10% de desconto.   
Esse campo deve ser incluso no XML e no banco de dados.

|  |  |
| --- | --- |
|  | Dica: É sempre bem-vindo a criação de novas classes com apenas uma responsabilidade, utilizando os conceitos SOLID. |

8 – Exercício (Melhoria técnica)

O código tem uma complexidade ciclomática muito alta, refatorar o código para que seja possível entende-lo melhor e simplificar a manutenção.

|  |  |
| --- | --- |
|  | Dica: Não tem nenhum tipo de restrição com relação a reestruturação, sinta-se à vontade para criar mais classes, projetos, métodos e etc. |

9 – Exercício (Melhoria técnica)

Estão previstas muitas alterações nas regras desta aplicação, crie testes unitários para diminuir o risco e melhorar o design.

10 – Exercício (Analise)

É necessário gerar um report técnico das atividades desenvolvidas, descreva com suas palavras qual é o objetivo da tela e como foi implementado as alterações solicitadas.

**Será descrito abaixo de acordo com cada índice das atividades**.

1. Para gerar xml foi criado um arquivo Utils.cs na raiz do projeto Imposto.Core contendo duas funções genéricas, onde uma é para gerar o xml e outra para download. Para realização do download do xml foi adicionado no arquivo App.config uma key chamada **PathXML** para a infraestrutura configurar o caminho em que será efetuado o download.
2. Para efetuar a persistência com as procedures **P\_NOTA\_FISCAL** e **P\_NOTA\_FISCAL\_ITEM** foi criado um arquivo Conexao.cs dentro da pasta Data no projeto Imposto.Core genérica conectar na base de dados onde serão executadas as procedures. Foi criado também a tag connectionStrings dentro do arquivo App.config onde a key que chama a conexão do banco tem seu nome de **DEFAULT** para ser consumida pelo arquivo Conexao.cs.
3. Conforme solicitado foram criadas as propriedades do ipi dentro da classe item da nota fiscal e scripts para alteração da tabela dbo.NotaFiscalITem no banco de dados salvos na pasta SQL .
4. Criado procedure e salvo script de criação da procedure **P\_NOTA\_FISCAL\_ITEM\_IMPOSTOS\_CFOP** na pasta SQL.
5. Não foi possível reproduzir o BUG relatado.
6. Melhorias implementadas como criação de método genérico para limpar os campos do formulário no arquivo Utils.cs, e criado uma função que valida o que foi digitado nos campos de EstadoOrigem e EstadoDestino dando alerta de informando caso seja inserido uma UF inválida.
7. Campo desconto criado e gerado uma classe de desconto para implementar o desconto conforme a regra especificada.
8. Código refatorado para facilitar a manutenção caso haja alterações de regras como por exemplo foi criado uma classe apenas para tratar de ICMS separando responsabilidade da classe NotaFiscal que antes cuidava de toda regra do sistema.
9. Testes unitários criados com xUnit TDD de todas as funções para cada classe do sistema que possuem alguma regra de negócio.