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).

**OBS.: Antes da conclusão desse teste, foram corrigidos diversos BUGs, recomendo verificar a diferença entre os fontes;**

|  |  |
| --- | --- |
|  | 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.

**OBS.: Campos Incluídos o script AJUSTA [NOTAFISCALITEM].sql disponível no mesmo diretório que os .SQL enviados;**

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.

**OBS.: Script incluído no mesmo diretório que os .SQL enviados;**

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.

**OBS. Essa relação entre as UFs já havia sido implantada, o problema era um BUG generalizado onde trocava as UFs (origem e destino), vide diferenças de fonte;**

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.

**OBS. A quantidade de melhorias propostas foi proporcional ao tempo disponível para o teste, vide diferenças de fonte;**

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.

**Deve-se dar atenção especial ao método EmitirNotaFiscal ele erroneamente contem regras de cálculo e atribuições de valores que não condizem com sua definição;**

**OBS.: Vide OBS do exercício 01 e 06**

|  |  |
| --- | --- |
|  | 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.

**Recomendo fortemente trocar os TXTs de UFs do Form por Comboboxes limitando suas opções aos domínios de UFs contidos no banco;**

**Trocar as atribuições de CFOP por uma matriz de valores e, incluir no momento de salvar a notificação do valor para o usuário, dada a complexidade da regra de atribuição;**

**Colocar tratamento de NULL nos atributos de objetos nullabe do banco (todas as classes);**

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.

Seguem alterações mais relevantes no código:

* App.config:
  + Incluídas as TAGS:
  + CaminhoXML: Diretório onde serão salvos os arquivos XML com as informações das notas ficais;
  + cnxDBNetShoes: Connection String da aplicação
* FormImposto.cs:
  + Incluído o campo de desconto, com tratamento básico para itens não numéricos (mensagem de erro caso negativo);
  + Correção de BUG do pedido ItensDoPedido pois o mesmo estava mantendo o state dos cadastros anteriores;
  + Tratamento básico do check de brinde, não estava tratando como false quando não checados;
  + Inclusão de tratamento básico de exceção para tentativas de gravação (tanto no arquivo quanto na base);
  + Inclusão de notificação para erro de gravação;
  + Inclusão de rotina de limpeza do form após salvamento com sucesso;
* FormImposto.Designer.cs
  + Ajuste de design para acomodar o novo campo (desconto);
* NotaFiscalService.cs
  + Inclusão das chamadas de gerar o XML e Persistir em Base;
* NotaFiscalRepository.cs
  + Criação dos métodos de geração de XML e persistência na base;
* NotaFiscal.cs
  + Ajuste na classe “NotaFiscal” para que seja facilmente possível agrupar os Itens da nota no objeto NotaFiscal;
  + Alterar o atributo Pedido do método EmitirNotaFiscal para um tipo REF para aproveitar sua população pelos outros métodos;
  + Alteração dos atributos EstadoDestino e EstadoOrigem para correção de BUG;
  + Inclusão dos novos atributos no notaFiscalItem
* NotaFiscalItem.cs
  + Inclusão de novos atributos;
* Pedido.cs
  + Inclusão de novos atributos;
* AJUSTA [NOTAFISCALITEM].sql
  + Ajustes na tabela NotaFiscalItem (TODO, incluir verificação de já existência dos campos)
* P\_DETALHA\_NOTA\_FISCAL\_ITEM.SQL
  + Criação da proc P\_DETALHA\_NOTA\_FISCAL\_ITEM
* P\_NOTA\_FISCAL\_ITEM.sql
  + Inclusão de novos campos

**TEMPO APROXIMADO 03:30**