Conhecimento Técnico NV2

1. Explique o comando **lock** do C#.

Em uma aplicação multi-thread, quando duas threads simultâneas contêm um lock(), uma thread espera, até que os recursos sejam liberados. Neste caso, somente uma thread terá acesso ao recurso por vez e dessa forma torná-lo thread-safe.

1. O que é a interface **IEnumerable**?

A interface IEnumerable oferece suporte a uma iteração sobre uma coleção não genérica. Possui apenas o método GetEnumerator que retorna um enumerador que itera sobre a coleção. Todas as coleções a implementam.

1. Faça uma comparação entre o **Entity Framework** e **ADO .NET** informando as vantagens e desvantagens de cada uma das tecnologias.

Entity Framework => Mapeamento automático de objetos; abstração de queries; tempo de desenvolvimento reduzido; facilidade em futuras manutenções; desempenho reduzido devido as camadas de abstração.

ADO .NET => Conversão manual de objetos; Escrita manual de consultas; maior tempo de desenvolvimento; manutenção mais complexa; alto desempenho;

1. O que é um **ThreadPool**?

ThreadPool é o processo de criar uma coleção de threads durante a inicialização de um aplicativo multithread e, em seguida, reutilizar essas threads para novas tarefas, como e quando necessário, em vez de criar novas threads.

1. Explique as estruturas de dados **Array, List** e **Dictionary**.

Array => conjunto de elementos obrigatoriamente do mesmo tipo de dados. Cada elemento de um array pode ser acessado através de sua posição no array que e dada através de um índice. O tamanho do array deve ser pré definido.

List => representa uma coleção de objetos tipados. Cada elemento de uma List pode ser acessado através de seu índice. Esta classe fornece métodos para percorrer, filtrar, ordenar, procurar e manipular coleções de dados.

Dictionary => estrutura de dados que representa uma coleção de pares de dados compostos por chaves e valores.

1. Explique os comandos abaixo:

If/Else:

A instrução if/else é usada para executar blocos de código condicionalmente através de uma expressão booleana . A cláusula else é opcional e seu conteúdo será executado somente se o resultado da expressão booleana for falsa.

Switch/Case:

Switch/case é uma estrutura de condição que define o código a ser executado com base em uma comparação de valores. A estrutura também possui o comando break que é utilizado para especificar a última linha de código a ser executada dentro da condição. Se não declarado, os códigos implementados dentro dos cases seguintes serão executados. Também temos o operador default que é utilizado quando precisamos definir um fluxo alternativo para as situações em que o valor contido no switch não seja atendido por nenhum dos cases especificados.

For:

A estrutura de repetição for trabalhar checando uma condição para executar um bloco de código até que essa condição seja verdadeira.

While:

Semelhante ao for, o loop while é mais simples pois colocamos apenas a condição que queremos testar.

Desenvolvimento Prático

Desenvolva uma aplicação Console Application em C#, que leia o arquivo de entrada em anexo (.txt) que simula um arquivo de dados de faturas de clientes fictícios, separados por vírgulas, e aplique as regras definidas abaixo:

- Remova todos os registros com CEP Inválido (aplicar regras);

- Separe os registros com faturas zeradas e grave em um arquivo (.csv) a parte;

- Agrupe os arquivos de saída por números de páginas de faturas, levando em conta as regras abaixo:

- Teremos 3 arquivos de saída, **1 com registros de até 6 páginas**, **1 com registros de até 12 páginas** e **1 com registros com mais de 12 páginas**;

- Se o número de páginas informado no arquivo de entrada, for ímpar, somar mais uma página ao cálculo total de páginas do registro; A quantidade de páginas constantes no arquivo de saída deve ser sempre par.

- Todas as saídas deverão ser geradas em formato .csv respeitando o seguinte layout:

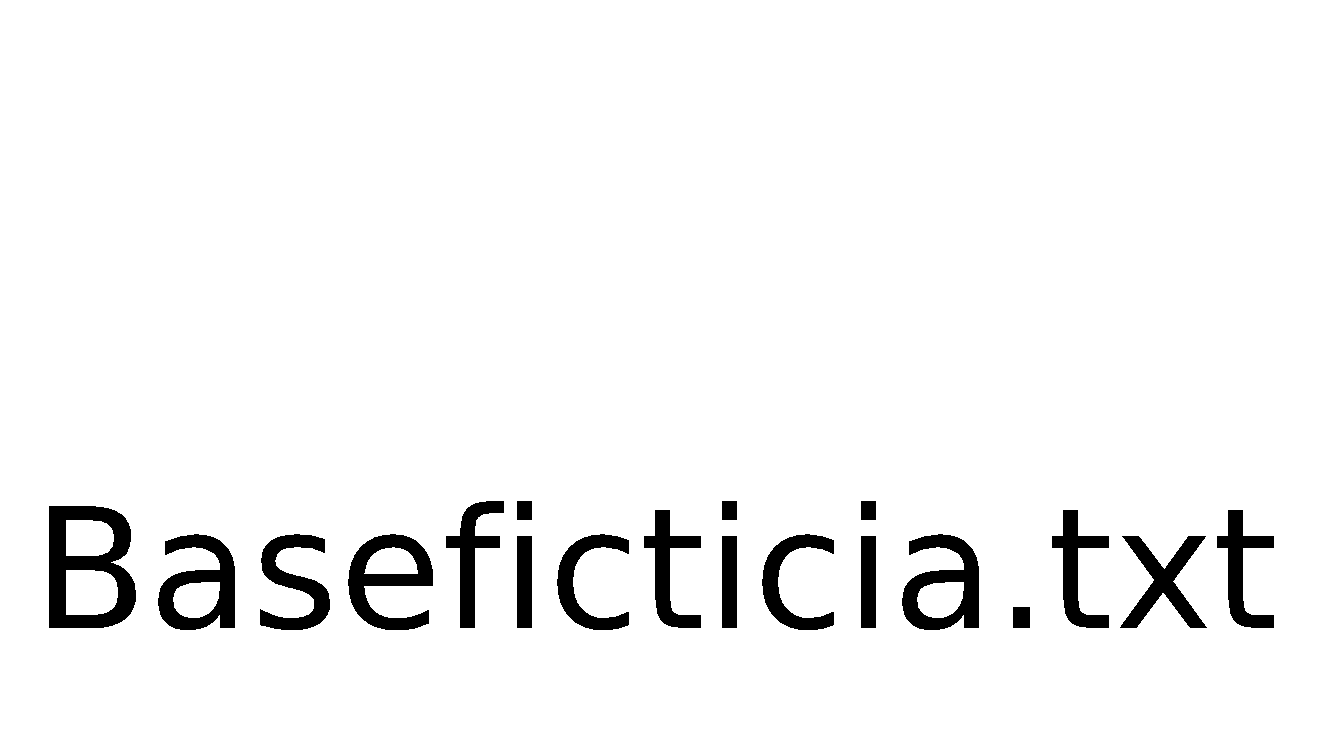
- NomeCliente;EnderecoCompleto;ValorFatura;NumeroPaginas;

**No final do processamento devem ser gravados os 3 .csv com os registros com faturas com valor e 1 .csv com faturas zeradas (sem levar em consideração número de páginas).**

A aplicação deve ser performática e obedecer às boas práticas de código limpo.

Fica livre o uso de Designe Patterns.

Zipar toda a aplicação (contendo os arquivos gerados na saída) e nos enviar junto ao restante do teste.



Banco de Dados

**1 - A solução correta para que uma consulta sql retorne às agências que possuem média dos saldos aplicados em conta maior que 1200 é:**

1. select nome\_agencia, avg(saldo)

from conta

group by nome\_agencia

having avg(saldo) > 1200

(b) select nome\_agencia, avg(saldo)

from conta

where ( having avg(saldo) > 1200 )

1. select nome\_agencia, avg(saldo)

from conta where avg(saldo) > 1200

group by nome\_agencia

1. select nome\_agencia, avg(saldo)

from conta

group by nome\_agencia

having saldo > 1200

RESPOSTA => A

**2 - Um programador precisa utilizar, em uma aplicação conectada a um banco de dados, uma instrução SQL para exibir apenas os nomes de funcionários da tabela** func **cujo campo nome se inicie pela letra P. A instrução correta que deve ser utilizada é:**

1. SELECT nome FROM func WHERE nome CONTAINS 'P%';

(b) SELECT nome FROM func WHERE nome LIKE 'P%';

(c) SELECT nome FROM func WHERE nome='P%';

(d) SELECT nome FROM func WITH 'P%' IN nome;

(e) SELECT nome FROM func LIKE nome='P%';

RESPOSTA => B

**3 - Considere uma tabela relacional TX, cuja instância é mostrada a seguir:**

A B

4 4

2 3

3 4

3 2

Considere também o comando SQL abaixo.

**delete from TX where exists (select \* from TX tt where TX.B=tt.A)**

O número de registros deletados da tabela TX por esse comando é:

(a) 0

(b) 1

(c) 2

(d) 3

(e) 4

RESPOSTA => B

**4 - Qual comando SQL cria a tabela ALUNO?**

ALUNO (cpf : string , nome : string , endereco : string, telefone : string)   
MATRICULA (cpf : string , cod-cad : string)   
CADEIRA (cod-cad : string , nome : string , creditos : number)

(a) CREATE TABLE ALUNO (cpf string , nome string , endereco string , telefone string) ;

(b) CREATE TABLE ALUNO (cpf : string , nome : string , endereco : string , telefone : string) ;

(c) CREATE TABLE ALUNO (cpf string PK, nome string , endereco string , telefone string) ;

(d) CREATE ALUNO AS TABLE (cpf : string PK, nome : string , endereco : string , telefone : string)

(e) CREATE ALUNO AS TABLE (cpf string PK , nome string , endereco string , telefone string) ;

RESPOSTA => B

**5 - Qual o comando SQL é utilizado para alterar o nome do aluno com CPF 512.859.850-01 para “Jose da Silva”?**

(a) ALTER RECORD ALUNO SET nome='Jose da Silva' WHERE cpf='512.859.850-01'

(b) INSERT INTO ALUNO nome='Jose da Silva' AND cpf='512.859.850-01'

(c) UPDATE ALUNO WHERE nome='Jose da Silva' AND cpf='512.859.850-01'

(d) UPDATE ALUNO SET nome='Jose da Silva' WHERE cpf='512.859.850-01'

(e) INSERT INTO ALUNO nome='Jose da Silva' WHERE cpf='512.859.850-01'

RESPOSTA => D

**6 - Qual o comando SQL que obtém apenas os nomes de todos os alunos?**

(a) SELECT \* FROM ALUNO WHERE nome IS STRING

(b) SELECT nome FROM ALUNO

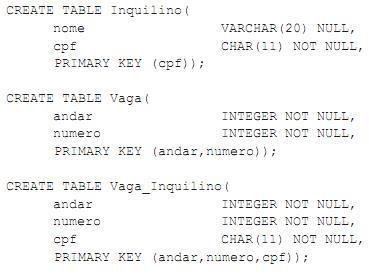
(c) LIST \* FROM ALUNO

(d) SELECT nome WHERE ALUNO

(e) LIST nome FROM ALUNO

RESPOSTA => B

**7 - Nessa situação, para se obter um relatório com a quantidade de vagas por cada inquilino, listadas e agrupadas por cpf, deve ser feita a seguinte consulta:**



(a) select cpf,andar,numero from Vaga\_Inquilino;

(b) select cpf,count(\*) from Vaga\_Inquilino group by cpf;

(c) select cpf,sum(cpf) from Vaga\_Inquilino group by cpf;

(d) select distinct cpf from Vaga\_Inquilino ;

(e) select distinct count(Inquilino.cpf) from Inquilino , Vaga\_Inquilino Where Inquilino.cpf = Vaga\_Inquilino.cpf order by Inquilino.cpf;

RESPOSTA => E

**8 - Considere a seguinte tabela de um banco de dados relacional:**

Cliente (CPF, Nome, Fone, End)

O comando SQL para obter o Nome dos clientes, cujo campo Fone tenha o valor “nulo”, é:

(a) SELECT Nome (Fone NULL) FROM Cliente

(b) SELECT Nome, Fone (NULL) FROM Cliente

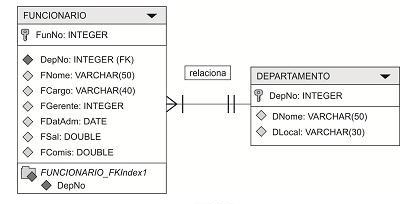
(c) SELECT Nome FROM Cliente WHERE Fone IS NULL

(d) SELECT Nome FROM Cliente WHERE Fone = “NULL”

(e) SELECT Nome FROM Cliente WHERE Fone LIKE “NULL”

RESPOSTA => C

**9 - Para exibir uma lista única de todos os cargos (FCargo) existentes no departamento de número (DepNo) 20, apresentando nesta lista apenas os cargos (FCargo) deste departamento e o nome do departamento (DNome), utiliza-se a instrução**



(a) select f.FCargo, d.DNome from FUNCIONARIO f, DEPARTAMENTO d where f.depNo = 20;

(b) select FUNCIONARIO.FCargo, FUNCIONARIO.DNome from FUNCIONARIO where FUNCIONARIO.depNo = DEPARTAMENTO.DepNo and FUNCIONARIO.DepNo=20;

(c) select f.FCargo, f.DNome from FUNCIONARIO f, DEPARTAMENTO d where f.depNo = d.DepNo and d.DepNo=20;

(d) select f.FCargo, d.DNome from FUNCIONARIO f, DEPARTAMENTO d where d.depNo = 20;

(e) select f.FCargo, d.DNome from FUNCIONARIO f, DEPARTAMENTO d where f.depNo = d.DepNo and d.DepNo=20;

RESPOSTA => A

**10 - É utilizado para combinar o resultado de dois ou mais comandos SELECT. Cada comando SELECT deve conter o mesmo número de colunas, as colunas devem contar tipos similares de dados e devem estar na mesma ordem em cada comando SELECT. O operador SQL, em questão, é chamado:**

(a) INTO.   
(b) UNION.   
(c) WHERE.  
(d) LEFT JOIN.   
(e) ORDER BY

RESPOSTA => B