Itens que serão avaliados

Itens imprescindíveis:

1. Robustez
    - Código trata possíveis exceções;
2. Performance;
    - Uso de TFields em loopings;
    - Eventos de telas em operações em lote;
3. Manutenção e/ou diminuição das validações de micro e macro arquitetura;
    - Quantidade de linhas de código por classe e método;
    - Complexidade ciclomática;
    - Quantidade de métodos;
    - Quantidade de parâmetros por método;
    - Heranças;
    - Dependências;
    - Tamanho do projeto;
    - Violações de arquitetura;
    - Código duplicado;
    - TDD
    - Testes (Damos importância para pirâmide de testes)
4. Boas práticas de desenvolvimento
    - Clean Code;
    - Patterns;
    - SOLID;
    - KISS;
5. Clareza e fácil entendimento
    - Nomes de métodos/variáveis;
    - Métodos pequenos e reutilizáveis;
    - Uso de interface;
6. Testes unitários

Seleção de pessoa Desenvolvedora Delphi:

Se você chegou até aqui é porque se interessou em ser um softplayer. Como temos muitas oportunidades para você colocar a mão na massa, queremos ver como você se sai com o cenário abaixo, que conseguiremos avaliar várias de suas competências.

A demanda:

Você deverá criar uma aplicação em Delphi com as seguintes funcionalidades:

Criar uma aplicação VCL chamada “ProvaDelphiApp” com um formulário principal e um menu chamado “Tarefas” contendo um item de menu para cada questão: “Tarefa 1”, “Tarefa 2” e “Tarefa 3”.

Todos os formulários implementados na prova deverão ser do tipo MDICHILD, e deverão ser criados dinamicamente pelo formulário do menu principal e destruídos no evento FormClose dos mesmos;

1) Criar um pacote de RUNTIME, de nome “spComponentes.dpk” e adicionar ao grupo de projetos “ProvaDelphi.bpg” juntamente com o ProvaDelphiApp já criado.

Dentro do pacote “spComponentes.dpk”, criar a classe de componente TspQuery derivada a partir do componente TFDQuery para a realização de selects no banco, e que disponibilize as seguintes propriedades:

a. spCondicoes: TStringList

b. spColunas: TStringList

c. spTabelas: TStringList

Implementar no componente o método GeraSQL, que irá utilizar os valores destas propriedades para gerar automaticamente um SQL e também irá adicionar este SQL à propriedade SQL da classe TspQuery. Se for informado mais de uma tabela, deve emitir uma exceção informando que é permitido informar apenas uma tabela para a geração do SQL.

Deverá ser gerado apenas o SQL, não é necessário realizar a conexão com o banco de dados. O nome da unit do componente deverá ser “uspQuery.pas”.

Configurar os diretórios do pacote para gerar o arquivo BPL no diretório ‘.\Bin’ e para gerar os arquivos DCU e DCP no diretório ‘.\Lib’.

Criar um pacote de DESIGNTIME, de nome “spComponentesDT.dpk” e adicionar ao grupo de projetos “ProvaDelphi.bpg”. Este pacote de designtime deverá conter uma unit de nome “uspComponentesRegistro.pas”, a qual deverá registrar o componente TspQuery na paleta de componentes “spComponentes” do Delphi. Configurar os diretórios do pacote para gerar o arquivo BPL no diretório ‘.\Bin’ e para gerar os arquivos DCU e DCP no diretório ‘.\Lib’.

Para testar o componente, criar um formulário onde seja possível ao usuário inserir as tabelas, condições e colunas, e que permita gerar e visualizar o SQL gerado. O nome do arquivo do formulário deverá ser “ufTarefa1.pas” e o nome do form “fTarefa1”. Ao clicar no item de menu “Tarefa 1’ do menu principal da aplicação, abrir o formulário “fTarefa1”.

Dicas: utilize componentes TMemo no fomulário para o usuário informar as colunas, condições e tabelas, e também para exibir o SQL gerado. Abaixo segue um exemplo da tela sugerida:

2) Para demonstrar seu domínio em programação com Threads no Delphi, crie um formulário com o nome da unit “ufTarefa2.pas” e nome do form “fTarefa2” onde o usuário poderá pressionar um botão que criará e executará duas threads. Estas threads irão realizar um laço de 0 até 100, onde a cada iteração do laço elas deverão aguardar um tempo em milisegundos determinado pelo usuário. A cada iteração do laço, a thread deverá exibir o valor do contador de iterações do loop numa barra de progresso no formulário. Como serão duas threads rodando, o formulário deverá exibir duas barras de progresso.

O usuário deverá poder configurar para cada thread um tempo em milisegundos que a mesma irá aguardar a cada iteração do loop.

Dica: utilize o procedimento Sleep(pnMilisegundos: integer) para fazer a espera entre cada iteração do loop das threads.

Ao clicar no item de menu “Tarefa 2” do menu principal da aplicação, abrir o formulário “fTarefa2”.

3) Para demonstrar domínio de operações em lote, crie um formulário com o nome da unit “ufTarefa3.pas” e nome do form “fTarefa3”, onde será possível visualizar valores de projetos. O formulário deverá conter:

- Uma grid com as colunas “idProjeto”, “NomeProjeto” e “Valor”;
- Um edit “Total R$” para mostrar a soma total da coluna “valor” da grid;
- Um botão “Obter total” para somar os valores da coluna “valor” da grid e atribuir para o edit “Total R$”. Utilize um laço para fazer a soma;

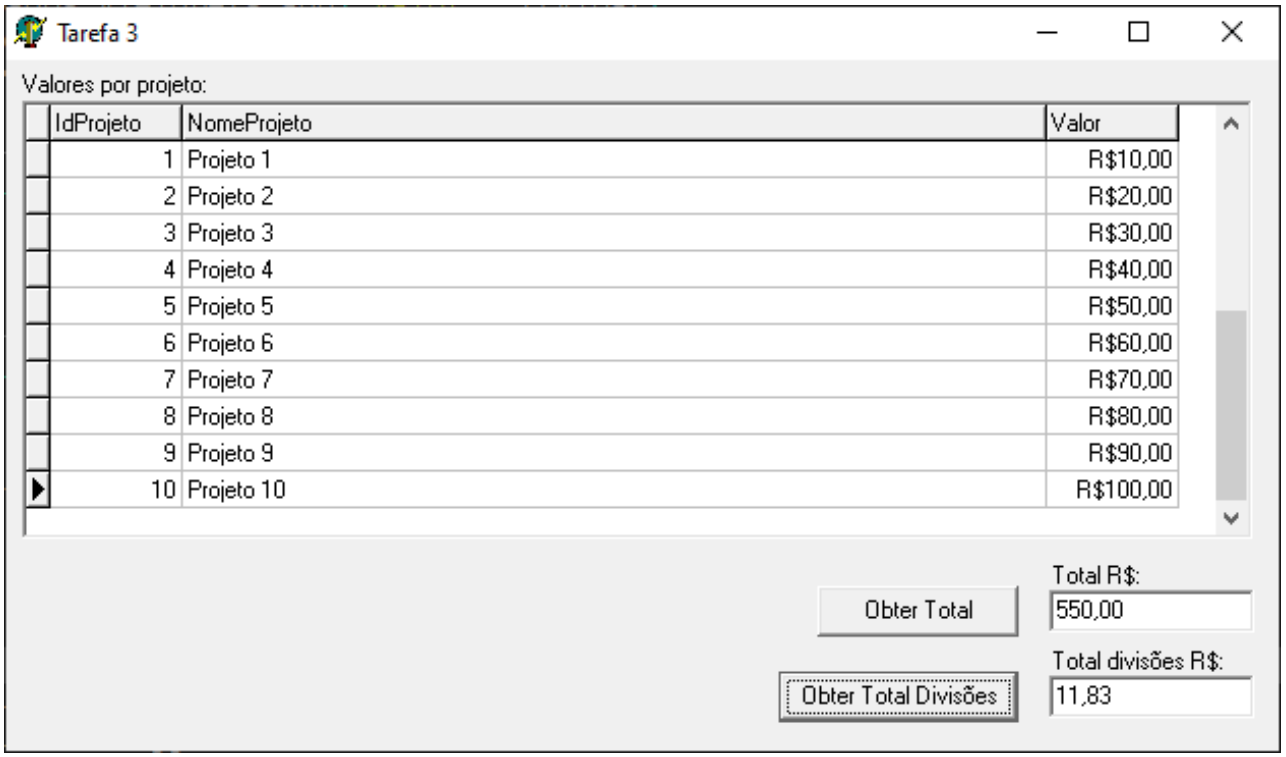
- Um botão “Obter Total Divisões” para somar o total das divisões do registro seguinte pelo registro anterior e atribuir no edit “Total divisões R$”. Ex:
    - Projeto 1 = 10,00 | Projeto 2 = 20,00 | Projeto3 = 30,00
    - (20 / 10) = 2
    - (30 / 20) = 1,5
    - Total a ser exibido em “Total divisões R$” = **3,5** (2+1,5)

Os componentes de dados que serão ligados na grid (clienteDataSet, dataSource) devem ser criados e destruídos em tempo de execução.

Ao abrir a tela, devem ser populados 10 registros dinamicamente de forma aleatória na grid com todas as colunas preenchidas. Utilize um laço para popular os registros.

Ao clicar no item de menu “Tarefa 3” do menu principal da aplicação, abrir o formulário “fTarefa3”.

Tela Tarefa 3:



Entrega:

Entregar um aplicativo (exe) funcional e os fontes do aplicativo e dos testes unitários.

Plus: Coverege da cobertura dos testes.