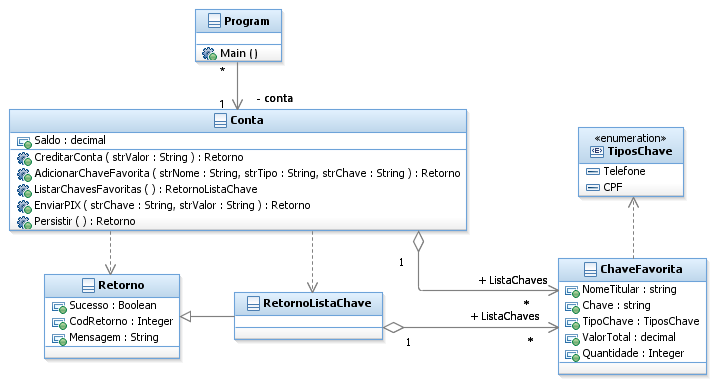
Exercício C# - PIX

A aplicação a ser criada consiste na simulação de uma conta com saldo, uma lista de chaves pix favoritas e o envio de pix para estas chaves. Sendo o envio do pix nada mais que atualização de informações da chave e do saldo.

Segue abaixo o detalhamento do que precisa ser desenvolvido.



No diagrama acima estão todas as classes e elementos públicos obrigatórios. A nomenclatura dos artefatos deve estar tal qual o diagrama. As classes devem estar contidas no **namespace Bergs.ProvacSharp**. Elementos privados ficam conforme a organização pessoal de cada um.

A classe conta deve ter um construtor em que recebe por parâmetro o caminho completo para o arquivo do banco de dados(mais informações no anexo ao final).

**Todas as validações detalhadas abaixo tem códigos de retorno específicos que devem ser seguidos** com a utilização da classe Retorno.

Criar um console application com nome “ProvaCSharp”, preferencialmente na pasta c:\desenvhome-pxc\, NÃO selecione a opção “Create directory for solution”.

Nas propriedades do projeto, em B*uild*, altere o *Output path* para c:/soft/pxc/bin.

Ao iniciar a aplicação apresenta o menu abaixo e o saldo deve estar zerado.

== Saldo atual: R$ 0,00 ==

1. Efetuar crédito em conta
2. Adicionar chave favorita
3. Listar chaves favoritas
4. Enviar PIX
5. Persistir chaves
6. Sair

Informe a opção desejada: \_

Após a execução de cada uma das ações exibir mensagem de sucesso ou falha e esperar que o usuário pressione alguma tecla, limpar a tela e reapresentar o menu.

Ao exibir o menu, sempre apresentar o saldo atualizado.

1. **Efetuar crédito em conta**

- Aplicação solicita que informe o valor a ser creditado.

- Validar se valor é numérico e maior que zero.

- Somar valor ao saldo.

- Apresenta mensagem de sucesso.

Falhas:

* 10 – Valor inválido.
* 20 – Valor menor ou igual a zero.

1. **Adicionar chave favorita**

- Aplicação solicita que informe:

\* Nome do titular

\* Tipo de chave: [1-Telefone, 2-CPF]

\* Chave

- Validar a chave conforme o tipo informado.

- Validar chave duplicada.

- Adicionar na lista de chaves favoritas da conta.

Para validar as chaves:

* CPF: 99999999999

regex = "^\\d{11}$";

* Telefone: ”+5551999999999”

regex = "^\\+[1-9][0-9]\\d{11}$";

if (System.Text.RegularExpressions.Regex.IsMatch(chave, regex))

Falhas:

* 30 – Tipo de chave inválido.
* 40 – CPF inválido.
* 50 – Telefone inválido.
* 65 – Chave duplicada.

1. **Listar chaves**

- Exibe lista de chaves: para cada item exibir a chave, nome titular, quantidade e o valor total de pix feitos para a chave.

1. **Enviar PIX**

- Aplicação solicita que informe a chave e o valor do pix.

- Verificar se existe saldo suficiente na conta.

- Verificar se a chave existe na lista.

- Valor deve ser numérico e maior que zero.

- Somar quantidade e valor de pix para a chave.

- Subtrair valor do saldo.

Falhas:

* 10 – Valor inválido.
* 20 – Valor menor ou igual a zero.
* 60 – Chave inexistente.
* 70 – Saldo insuficiente.

1. **Persistir lista**

A lista de chaves favoritas deve ser salva em tabela do banco de dados.

- Deve limpar a tabela e inserir os novos dados.

- Este processo deve ser atômico, ou seja, caso aconteça alguma falha a tabela deve voltar para o estado inicial (*rollback*).

- Para acesso ao banco de dados e informações sobre a tabela veja o anexo ao final.

Falhas:

* 980 – Falha ao deletar registros. Método ExecutarDelete retornar *false* ou ocorrer uma exceção.
* 990 – Falha ao incluir registro. Método ExecutarInsert retornar *false* ou ocorrer uma exceção.

1. **Sair**

Encerrar aplicação.

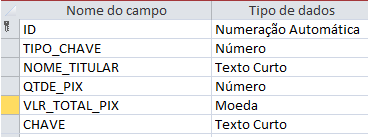
Observações:

- Todos os valores de moeda devem ser exibidos com R$, 2 decimais separados por vírgula e utilizando ponto na separação de milhar, ex.: R$ 10.000,00.

Anexo I

Acesso ao banco dados

O banco de dados tem apenas uma tabela chamada CHAVE:



O campo ID deve ser ignorado, ele é autoincrementado.

O tamanho máximo do campo NOME\_TITULAR é 20, porém isso **NÃO deve ser validado**, pois será usado para causar exceção e assim testar o correto tratamento.

Para acessar o banco de dados utilize as instruções abaixo:

1. Copie a AcessoBancoDados.dll para a pasta c:\soft\pxc\bin; caso não tenha esta pasta

em sua máquina, crie-a.

1. Copie o banco de dados Pxcz02da.mdb para a pasta c:\soft\pxc\data; caso não tenha

esta pasta em sua máquina, crie-a;

1. A classe de acesso a dados pertence ao *namespace* Bergs.ProvacSharp.BD;
2. Como utilizar a classe AcessoBancoDados:
   1. Nas propriedades do projeto da prova, na aba “*Build*”, trocar o *output path* para “..\..\soft\pxc\bin\”
   2. Adicionar uma referência a esta DLL no seu projeto ProvaCSharp (trocar *copy*

*local* para *false*);

* 1. Ao criar uma instância, informar no construtor o caminho completo do arquivo

de banco de dados *access* (conforme item 2 acima);

* 1. Após criar a instância, executar o método “Abrir”; este método cria a conexão

com o banco de dados e já inicia uma transação (*beginTrans*);

* 1. Para cada chave a ser inserida, enviar o respectivo comando *SQL* para o

método “ExecutarInsert”. Caso a inserção ocorra com sucesso, este método

devolverá *true*. Para os demais casos, *false, ou ainda pode ocorrer uma exceção*;

* 1. Para confirmar todas as modificações no banco de dados, ao término das

inserções deve ser chamado o método “EfetivarComandos” (*Commit*);

* 1. Para encerrar a conexão com o banco de dados, basta invocar o método

“Dispose”.

* 1. A classe AcessoBancoDados implementa a interface IDisposable, logo, para

controlar o fechamento da conexão é possível utilizar a diretiva *using*;

* 1. Encerrar a conexão com o banco de dados SEM executar o comando “EfetivarComandos” acarretará em perda de TODAS as modificações realizadas no banco de dados (*rollback*);

Anexo II

Regras

1. A avaliação é individual;
2. Não é permitido o auxílio de colegas da turma, somente dos monitores e instrutores;
3. Não é permitido a utilização de e-mail, skype, whatsapp ou qualquer outro aplicativo de conversa entre integrantes da turma.
4. É permitido a consulta ao material de aula e seus artefatos localizados na máquina local ou área privada;
5. É permitida a consulta ao Google e stack overflow para auxílio na sintaxe dos comandos.

Anexo III

Forma de entrega

1. Compactar no formato “.zip” as pastas que contemplam os componentes gerados (não enviar pasta “bin” e “obj”).
2. Enviar o arquivo zipado por e-mail para “Nelio Maraschin; Dalton Torres” com o título “Avaliação C#”.
3. Confirmar com algum dos destinatários a recepção do arquivo.