DevX Labs
Delphi avançado
Instrutor: Marcio Michelluzzi

Exercícios de fixação Delphi avançado

Com base no projeto desenvolvido em sala, devem ser realizadas as mudanças descritas abaixo

Correções e Melhorias

1 - Correção / Melhoria

Classe: TValidadorUsuario

Método: TValidadorUsuario.Validar(PUsuario: TUsuario; PSenhaConfirmacao: String)

Categoria: Melhoria

Descrição: Necessário realizar outras validações referentes aos campos do cadastro. Fica aberto para validações que sejam julgadas necessárias. Ex: Validação de CPF, validação de caracteres permitidos no campo de Nome, ...

2 - Correção / Melhoria

Classe: TdmRavin

Método: BeforeConnect / AfterConnect

Categoria: Melhoria

Descrição: Nesses dois métodos existe um caminho fixado no código fonte (caminho do banco de dados assim como as propriedades de conexão com o banco de dados). Seria interessante que esse caminho e essas propriedades de conexão fossem lidos de propriedades criadas do arquivo configurações (.ini).

Tratativa: É necessário que esses textos que estão fixos no código sejam repassados para propriedades dentro de um arquivo (.ini). Para isso será necessário criar novos itens dos enums TSECAO e TPROPRIEDADE para representar cada um desses valores que estão fixos. Além disso será necessário que esses novos valores já estejam pré cadastrados dentro do arquivo configurações (.ini) para que eles possam ser lidos do arquivo no momento em que as rotinas aconteçam.

3 - Correção / Melhoria

Classe: TdmRavin

Método: CriarTabelas / InserirDados

Categoria: Melhoria

Descrição: Nesses dois métodos existe um caminho fixado no código fonte (nome dos arquivos de criação das tabelas e inserção dos dados no banco). Seria interessante que esse caminho estivesse em uma propriedade do arquivo do configuracoes.ini

Tratativa: É necessário que esses textos que estão fixos no código sejam repassados para propriedades dentro de um arquivo (.ini). Para isso será necessário criar novos itens dos enums TSECAO e TPROPRIEDADE para representar cada um desses valores que estão fixos. Além disso será necessário que esses novos valores já estejam pré cadastrados dentro do arquivo configurações (.ini) para que eles possam ser lidos do arquivo no momento em que as rotinas aconteçam.

4 - Correção / Melhoria

Classe: Classes que possuem o método SetarFormularioPrincipal

Método: SetarFormularioPrincipal

Categoria: Melhoria

Descrição: Esse método está duplicado em várias classes do código.

Tratativa: É necessário que ele seja unificado em uma única unit. Essa unit pode receber o nome de TFormUtils deve estender de TObject e implementar o método SetarFormularioPricipal. O método terá o mesmo comportamento que o método existente nas outras classes. Num futuro essa classe poderá abrigar outros métodos comuns aos formulários da aplicação.

5 - Correção / Melhoria

Classe: TFormPainelGestao

Método: frmMenuItemSairlblTituloClick

Categoria: Melhoria

Descrição: Esse evento faz o logout do usuário do sistema, registrando que o usuário foi deslogado dentro do arquivo de configurações (.ini), porém quando o usuário realiza logout o usuário não é levado para a tela de login (o sistema é fechado).

Tratativa: É necessário que a aplicação não mais feche quando o processo de logout acontece. Deve-se fazer com que a aplicação redirecione o usuário para a tela de login. Para isso deve-se utilizar a rotina (SetarFormularioPrincipal) para que seja modificado o formulário da aplicação e após isso seja fechada a tela de Painel de Gestão e mostrado o form de autenticação.

6 - Correção / Melhoria

Classe: TFormLogin/TFormAutenticar

Método: frmBotaoPrimarioAutenticarspbBotaoPrimarioClick

Categoria: Melhoria

Descrição: Hoje o processo de login acontece de maneira correta, porém a forma como a liberação de memória dos objetos acontece pode ser otimizada.

Tratativa: No evento que é executado para realizar o processo de login, pode-se fazer com que o processo de liberação de memória aconteça utilizando o try finally garantindo assim que mesmo algum tipo de erro ocorra os objetos que precisam ser desalocados serão desalocados.

7 - Correção / Melhoria

Classe: TFormSplash / TFormLogin/TFormAutenticar

Método: InicializarAplicacao/ frmBotaoPrimarioAutenticarspbBotaoPrimarioClick

Categoria: Melhoria

Descrição: Hoje o processo de login faz com que o usuário fique eternamente logado, porém é importante que o usuário seja forçado a refazer o login no sistema.

Tratativa: Quando o processo de autenticação for realizado com sucesso, deve-se registrar no arquivo de configurações (.ini) a hora que o evento de login ocorreu para que, numa próxima abertura da aplicação (no método em que a rotina InicializarAplicacao for executada) seja verificado a quanto tempo foi feito o ultimo login na aplicação (desconsiderando qual foi ultimo usuário que logou) e caso esse tempo seja superior a 5 dias o sistema deve exigir um novo processo de login do usuário.

8 - Correção / Melhoria

Classe: TFormLogin/TFormAutenticar (são a mesma tela)

Método: N/A

Categoria: Melhoria

Descrição: Hoje o campo de cpf está aberto a qualquer informação (números, textos ...). É necessário que seja feita uma substituição do componente responsável por coletar o dado de CPF. Existem opções mais interessantes como o TMaskField.

9 - Correção / Melhoria

Classe: TFormRegistrar

Método: frmBotaoPrimarioRegistrarspbBotaoPrimarioClick

Categoria: Melhoria

Descrição: Hoje o processo de login apenas registra um registro de usuário, porém não cria uma pessoa para que seja vinculada ao usuário.

Tratativa: Deve-se fazer com que quando um novo usuário for cadastrado, uma nova pessoa seja criada e vinculada ao. Será necessário primeiramente criar uma nova pessoa, cadastra-la no banco de dados, recuperar o id com o qual essa pessoa foi cadastrada e após isso criar um novo usuário modificando nesse novo usuário o id da pessoa (chave estrangeira) que foi previamente cadastrada.

10 - Correção / Melhoria

Classe: Todas as telas

Método: Todos os eventos de tela

Categoria: Melhoria

Descrição: Hoje todos os eventos (dos componentes visuais) possuem dentro deles lógica de programação. Sabemos que não é uma boa prática colocar a lógica de programação dentro dos eventos e sim criar novos métodos que sejam chamados a partir dos eventos.

Tratativa: É necessário que sejam as mudanças para que os eventos não mais tenham lógica de negócio e sim comecem a chamar métodos que sejam responsáveis pelas lógicas de negócio.

11 - Correção / Melhoria

Classe: TdmRavin

Método: DataModuleCreate

Categoria: Melhoria

Descrição: Hoje o caminho do driver de conexão com a aplicação (drvBancoDeDados) está com a propriedade “VendorLib” (responsável por especificar onde fica a dll do MySQL) dentro do nosso computador. Porém caso o projeto seja clonado para uma nova máquina esse caminho ficará errado fazendo com que o projeto não possa ser executado. O mesmo vale caso queiramos executar o projeto numa outra máquina.

Tratativa: É necessário que logo no começo do evento DataModuleCreate seja feita configuração da propriedade “VendorLib” do drvBancoDeDados a partir de uma propriedade setada no arquivo de configurações. Também deve-se assumir que essa lib ficará na mesma pasta de aplicação dos nossos scripts, fazendo com que esse problema de caminho da lib seja resolvido (C:/Users/UsuarioLogado/Documents/ravin).

12 - Funcionalidade

Classe: TdmRavin

Método: DataModuleCreate

Categoria: Funcionalidade

Descrição: Para que nossa aplicação seja executada é preciso que nosso arquivo .ini esteja pré configurado com algumas propriedades e com valores padrões.

Tratativa: É necessário criar uma rotina que verifique se o arquivo padrão está criado na pasta da aplicação (assim como se a pasta existe). Caso a pasta não exista (ou mesmo o arquivo não exista) é necessário criar as estruturas necessárias (pasta) assim como gravar dentro do arquivo de configuração as configurações padrões do sistema.

Feature de cadastro de cliente

DOR - Definition of Ready

- **Qual problema e em qual contexto se encontra?**

  Hoje o sistema do App do Produtor não permite o cadastro de novos clientes. Essa situação faz com que os usuários (restaurantes) só possam atender (utilizando o sistema) clientes que já estejam cadastrados na base de dados. Isso, dificulta a adesão do sistema por novos usuários, assim como dificulta com que os usuários que já utilizam o sistema possam utilizar o produto de maneira integral em seus atendimentos.

- **Qual objetivo / Qual problema de mercado queremos resolver?**

  Permitir aos restaurantes que possam cadastrar novos clientes.

- **Por que escolhemos colocar esforço nisso?**

  Tendo em vista que o sistema já conta com conjunto de funcionalidades que atende o fluxo de gerenciamento de mesas de um restaurante essa é uma funcionalidade faltante que compõe esse processo e que impacta a utilização do sistema.

- **Quem é o ponto de contato?**

    1. Marcio Michelluzzi
- **Quais restrições/dúvidas?**
    1. N/D
- **Qual versão deverá ser liberada essa entrega?**
    1. Essa funcionalidade deve ser liberada até a data de 02/12

DOD - Definition of Done

- **Há algo importante a mencionar que a aplicação não fará?**
    1. N/A
- **Viabilidade sistêmica (arquitetura/análise de sistema)?**

A funcionalidade de cadastro de cliente é criada a partir da estrutura de banco de dados da tabela Pessoa. Sabe-se que um registro de pessoa no banco de dados que contenha o campo tipoPessoa com o valor “C” é uma pessoa cliente (com o tipo “F” é funcionário).

Devem ser criados novos campos para a tabela pessoa:

1. Data de Nascimento;
2. Email;

Alguns passos devem ser seguidos para a conclusão da criação da feature de cadastro de cliente:

1. Criação da classe TPessoa (com as propriedades baseados na estrutura de banco de dados);
2. Criação da classe TPessoaDao com os seguintes métodos:
    i. BuscarTodasPessoas(PTipoPessoa: char);
    ii. BuscarPessoaPorId(PIdPessoa: Integer);
    iii. InserirPessoa(PPessoa: TPessoa);
    iv. ExcluirPessoa(PIdPessoa: Integer);
    v. ExcluirTodasPessoas();
3. Criar o validador dos dados de Pessoa (definir quais são os campos);
4. Criação da tela de cadastro/listagem de pessoas (com base no protótipo de tela)
    i. Tela de cadastro:
        1. Campo de nome (TEdit);
        2. Campo de CPF (TMaskEdit);
        3. Campo de telefone (TEdit com a propriedade numbersOnly habilitada)
        4. Campo data de nascimento (TDateTimePicker)
    ii. Tela de listagem:
        1. Lista de usuários (TListBox)
        2. Botões: Frame primário (criado em aula) e foi criado um frameBotaoCancelar que é exatamente igual ao botão primário mudando apenas a cor de fundo do painel;
5. Implementação dos eventos com base na tela de protótipos:
    i. Tela de cadastro:
        1. No botão de salvar da tela de cadastro de cliente, os dados que foram preenchidos na tela devem ser validados e caso estejam válidos um novo registro de usuário deve ser inserido no banco de dados;
        2. No botão de excluir deve-se fazer a exclusão do registro atualmente aberto. Caso ele tenha sido carregado do banco de dados (Selecionado na lista de usuários) deve-se fazer a exclusão. Caso esse registro seja um novo registro e ainda não esteja cadastrado no banco de dados, deve-se apenas limpar os campos da tela;

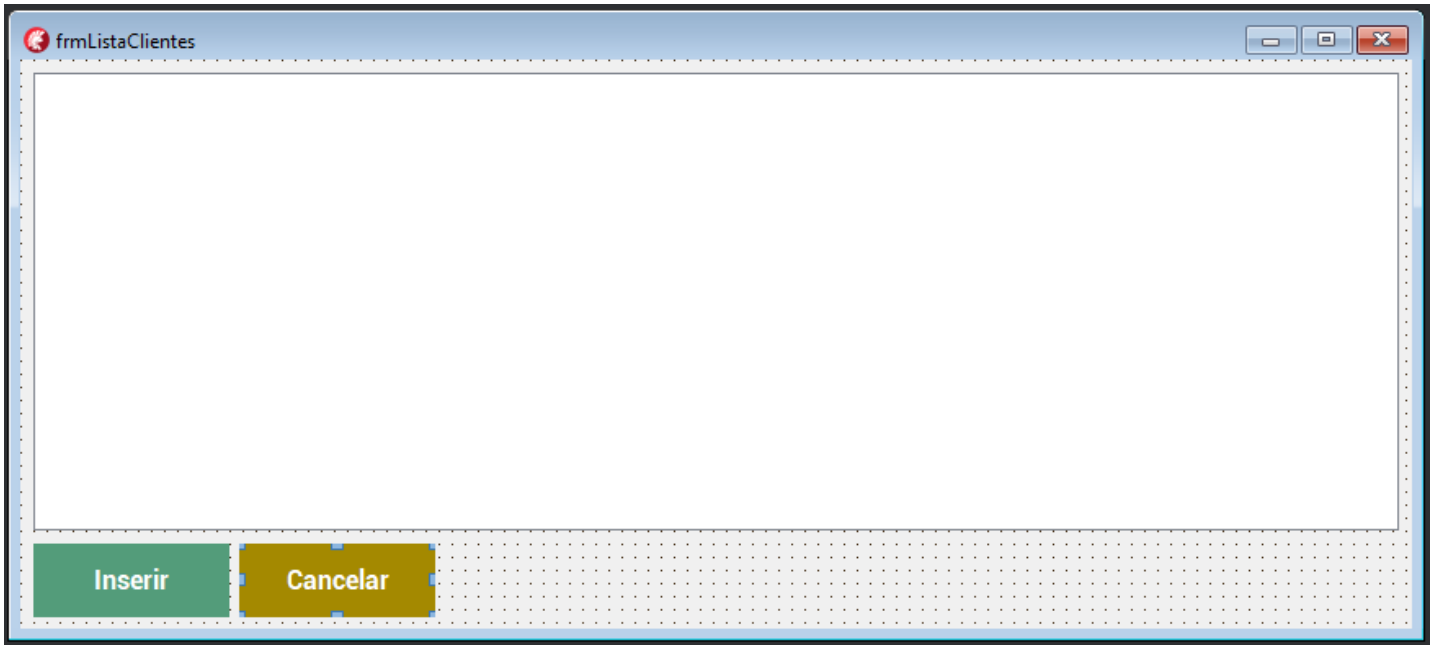
3. No botão de cancelar da tela de cliente faz com que quando clicado essa tela seja fechada sem que nenhuma alteração feita (e não foi salva) seja persistida no banco de dados;

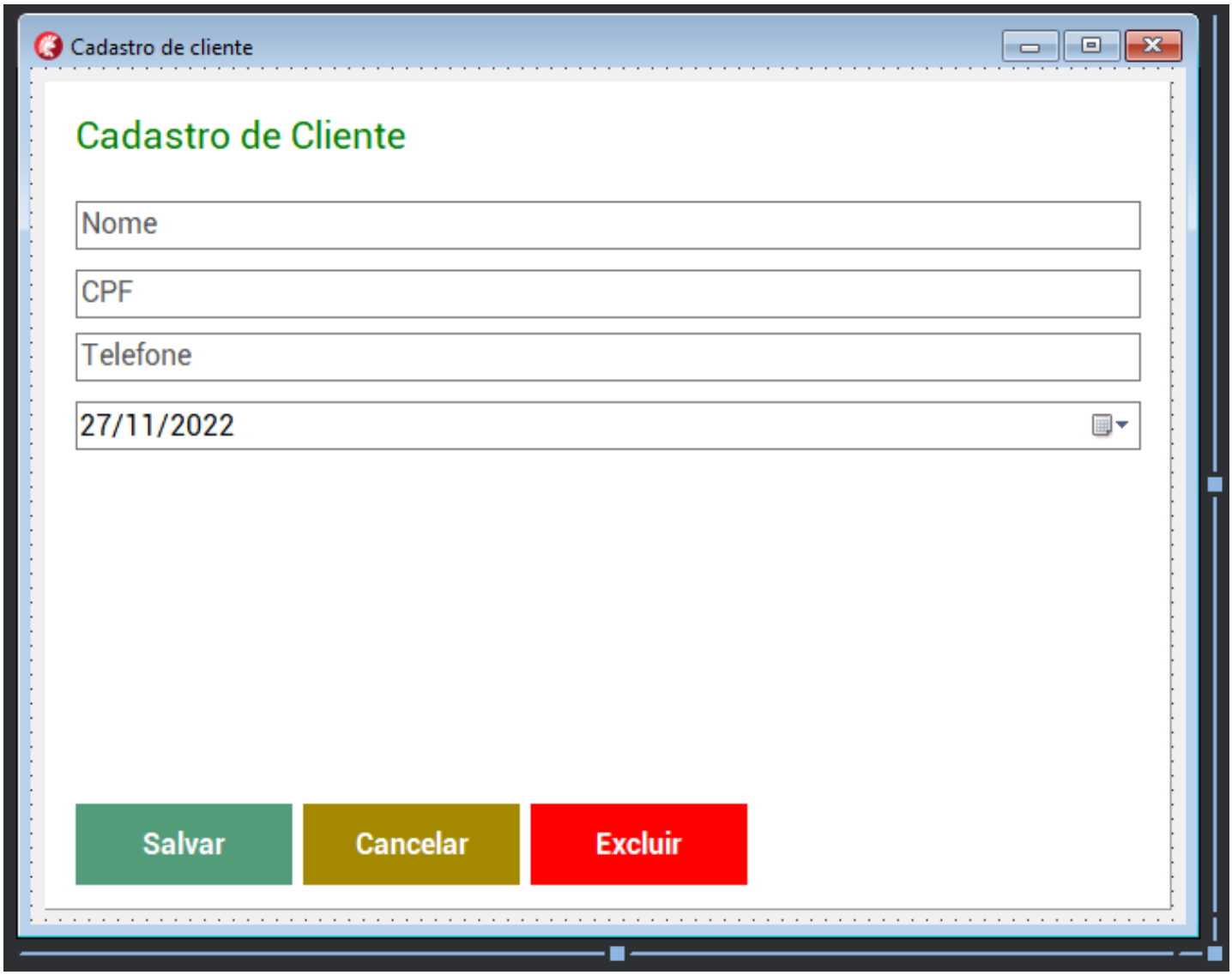
ii. Tela de listagem:
   1. No botão inserir o usuário do sistema deve ser levado a tela de cadastro do um cliente;
   2. No botão cancelar o usuário deverá ser levado para a tela de Painel de Gestão;
   3. Quando ocorrer o evento de click em um dos itens da lista de clientes, o usuário deverá ser levado a tela de cadastro de cliente para que possa cadastrar um novo cliente;

Um ponto importante que deve ser considerado é o tratamento de exceções nos casos onde elas possam ocorrer.

Outro ponto importante deve ser a validação dos dados do cadastro do cliente, campos que são obrigatórios devem ser validades para garantir que valores vazios não sejam permitidos.

- **Testar em qual banco de dados?**
  1. MySQL
- **Quais novos cenários de testes serão gerados?**
  1. N/D
- **Quais novos cenários de testes serão impactados?**
  1. N/D
- **Protótipo de tela?**





- **Qual parametrização necessária para execução da entrega?**
    1. Para realizar os testes da funcionalidade recomenda-se a exclusão do schemas existente para que o sistema recrie o database com todas as suas alterações;
    2. É necessário que os arquivos de scripts estejam dentro da pasta da aplicação;
    3. É necessário que os arquivos que estão dentro da pasta estejam atualizados;
    4. Colocar a lib do MySQL dentro da pasta da aplicação que contém os arquivos de scripts que são utilizados pelo sistema.