TRABALHO DE ENGENHARIA DE SOFTWARE I – 2. BIMESTRE

- Realizar o trabalho em grupo de até 3 alunos;
- Assuntos tratados no trabalho podem ser cobrados em prova
- Trabalho incompleto **-3,0 pontos.**
- **Peso 4,0 Nota Bimestre**

Trabalho 1– Estudo de Caso – Peso 1,0 – Entrega: 28/06/2019

Para o estudo de caso escolhido, faça:

a) Diagrama de Casos de Uso;
b) Diagrama de Classes de Negócio;
c) Modelo de Dados E-R;
d) Implementação do protótipo Interface WEB (Arquitetura em Camadas) das propostas de interfaces apresentadas (veja as informações complementares parte 1 e parte 2). Neste caso a tela "Manter Paciente ou Cliente" e a tela da consulta "OS ou prontuário Paciente". Observar que a Interface Web neste caso somente terá lógica de apresentação. Não será necessário implementar ainda camada de negócio – será como se fosse um protótipo do sistema.

Restrições para implementação de um caso de uso:

a) Usar ferramenta case para modelagem UML (diagrama de casos de uso e de classes);
b) Usar ferramenta case para modelo de dados;
c) Arquitetura em N Camadas: separação lógica entre camada de apresentação e negócio
d) Utilizar tecnologias java;
e) Uso de SGBD relacional para dados a serem armazenados;
f) Interface para WEB

OPÇÕES DE ESTUDO DE CASO:

Estudo de Caso 1: **Controle de uma Clínica Médica**

Desenvolver uma solução que melhore o funcionamento de uma clínica médica, facilitando os processos de marcação de consulta e visibilidade das informações detalhadas de cada paciente para o médico, durante uma consulta. Cada vez que o paciente se consulta, antes da mesma, é colhido alguns dados do paciente: peso e pressão arterial. O médico, durante uma consulta deve dispor das seguintes informações sobre o paciente: tipo sanguíneo, idade, os registros do peso e pressão arterial das últimas consultas, médico e motivo da consulta, exames realizados e medicamentos prescritos. Todas as vezes que um médico solicita algum tipo de exame, o paciente, após realizá-lo, envia para clínica para que o médico analise. Tudo que é feito para o paciente numa consulta é registrado, assim, todas as vezes que o paciente volta, o médico que realizará a consulta terá as informações disponíveis. O paciente também poderá acessar a clinica para consultar seus dados (prontuário).

Estudo de Caso 2: Controle de uma Oficina de Automóvel

Desenvolver uma solução que melhore o funcionamento de uma oficina mecânica, disponibilizando uma forma de:

a) Realizar um orçamento prévio do que deve ser feito no veículo: Serviços, Produtos a substituir, prazo para atendimento e Custo dos serviços e produtos;
b) O cliente poder aprovar o orçamento e acompanhar eletronicamente o serviço a ser executado;
c) O dono da oficina pode consultar o sistema para saber quais veículos estão ou foram atendidos;
d) Disponibilizar um serviço que notifique o dono do veículo de futuras manutenções preventivas.

INFORMAÇÕES COMPLEMENTARES PARTE 1 DO TRABALHO

Engenharia de Software

Segue, para cada estudo de caso, os dados principais que envolve cada problema.

OPÇÕES DE ESTUDO DE CASO:

Estudo de Caso 1: **Controle de uma Clínica Médica**

TELA DE CONSULTA DOS DADOS A SEREM MODELADOS PARA ESTUDO DE CASO 1:

PRONTUÁRIO DO PACIENTE

Cod. Paciente: ____ Nome Paciente: _____ CPF: _____
Endereço: CEP _____ Rua: _____ Nro:_____ Bairro:_____
Cidade: _____ UF: _____ Complemento: _____
Fone(s): _____, _____, _____
Email(s): _____, _____
Data Nascimento: _____ Sexo: _____

CONSULTAS REALIZADAS (*várias*)

Dt Consulta	Tipo da Consulta	CID Consulta	Nome Medico	CRM Medico
_____	_____	_____	_____	_____
_____	_____	_____	_____	_____
_____	_____	_____	_____	_____

EXAMES REALIZADOS

Dt Exame	Tipo Exame	Resultado
_____	_____	_____
_____	_____	_____
_____	_____	_____

Remédios Prescritos (receitados)

Data	Remédio	Posologia (forma de uso)
_____	_____	_____
_____	_____	_____
_____	_____	_____

Observações:

Tipo de consulta: Refere-se a especialidade da consulta. Por exemplo:
CONSULTA CLINICO GERAL, CONSULTA COM NEUROLOGISTA, CONSULTA COM OTORRINOLOGISTA
CID Consulta: Refere-se a tabela internacional de doenças. Exemplo:
Código: A90 Descrição: Dengue
Código: D50 Descrição: Anemia por deficiência de ferro.
Fonte: *http://www.previdencia.gov.br/dados-abertos/estatsticas/tabelas-cid-10/*
Tipo de Exame: Refere-se a algum tipo de exame pedido, por exemplo:
Exame Sangue Colesterol, Exame RAIO-X, etc.
O campo *Resultado* refere-se a um texto sobre o resultado específico do tipo de exame realizado.
Posologia: forma de uso do remédio receitado. Por exemplo: tomar o comprimido de 8h em 8h horas por 3 dias. Texto livre, escrito pelo médico na prescrição da receita.

Estudo de Caso 2: Controle de uma Oficina de Automóvel

TELA DE CONSULTA DOS DADOS A SEREM MODELADOS PARA ESTUDO DE CASO 2:

Nro. Pedido Manutenção (OS): _____ Data OS : ___/___/___

Dados do Cliente:
Cod. Cliente: ____ Nome Cliente: _____ CPF/CNPJ: _____
Endereço: CEP _____ Rua: _____ Nro:_____ Bairro:_____
Cidade: _____ UF: _____ Complemento: _____
Fone(s): _____, _____, _____
Email(s): _____, _____

Dados do Veículo:
Nome Veículo: _____Modelo:_____ Marca: _____ Placa:_____
Km do veículo na OS: _____

Serviços Realizados *(vários)*

Cod Serviço	Tipo Serviço	Valor Mão de Obra
_____	_____	_____
_____	_____	_____

TOTAL DE SERVIÇOS: _____

Produtos Utilizados *(vários)*

Cod. Produto	Nome Produto	Qtde	Valor Item	Total
_____	_____	_____	_____	_____
_____	_____	_____	_____	_____
_____	_____	_____	_____	_____

TOTAL PRODUTOS: _____

TOTAL DA OS: _____

OBS da OS: _____ (*texto livre)*

Estudo de caso 1 ou 2 - Tela para Manter Paciente ou Cliente

Nome: _____

Nome Meio:_____ Ultimo Nome:_____

CPF ou CNPJ: _____ < QUANDO DIGITAR VERIFICAR SE JÁ EXISTE>

CEP: _____ <digita CEP e sistema retorna endereço correspondente>

Endereço: .....

Nro. : _____ Complemento Endereço: _____

Foto: _____ < indicar aqui foto para reconhecimento>

Descrição Foto: _____ <valor preenchido no sistema a partir da FOTO reconhecida via Microserviço da IBM Cloud>

Outros dados da modelagem de classes Cliente / Paciente

<Salvar Dados> <Consultar novo cliente/paciente Id: _____ >

2 Trabalho 2 – Peso 1,0 - Implementação Serviços de Endereço (Grupo) – Entrega: 16/08/2019

De acordo com o diagrama de classes discutido em sala de aula implementar em Arquitetura N Camadas, orientada a serviços os serviços de Endereço:

Serviço na classe: **UCEnderecoGeralServicos**.

SERVICO	**Descrição**	**Parâmetro Entrada**	**Parâmetro Saída**
obterEnderecoPorCEP	Dado um CEP específico, retornar o lista de endereço correspondente	CEP	Lista de endereços relacionados ao CEP
obterEnderecoPorID	Dado um ID específico, retornar o endereço correspondente	Objeto Endereço com o ID setado.	**Em caso de erro:** EnderecoException caso erro validação ou Exception se erro infra **Se OK:** Objeto Endereco
obterEnderecoExterno	Dado um site, microserviço, webservice nas nuvens qualquer, retornar no objeto Endereço o endereço	Site a pesquisar	**Em caso de erro:** EnderecoException caso erro validação ou Exception se erro infra **Se OK:** Objeto EnderecoEspecifico
obterCidade	Dado uma cidade qualquer, retornar o objeto Cidade	Objeto cidade com o idcidade	**Em caso de erro:** EnderecoException caso erro validação ou Exception se erro infra **Se OK:** Objeto Cidade com UF relacionado.

Características do projeto:

CAMADA DE NEGÓCIO:

Projeto 1 - MyEnderecoBO

- classes tipo entidade (Business Object / Value Objets)

Pacotes: unioeste.geral.endereco.bo

Observações: Cada classe de negócio incluir: ***implements serializable***.

Projeto 2 - MyInfraAPI

- classe de conectividade com SGBD

Pacote: unioeste.apoio.<nome pacotes conforme tipo infra> (BD, etc)

Projeto 3 - MyEnderecoServicos

- classe UCEnderecoGeralServicos, cols e daos

Pacotes:

unioeste.geral.endereco.manager

unioeste.geral.endereco.col - para objetos tipo Col

unioeste.geral.endereco.dao – para objetos DAO

CAMADA INTERFACE WEB:

- Caso de Uso: Manter Endereço

- Opção para demonstrar funcionalidade – mostrar endereço obtido de um site

Projeto 4 – Camada Interface – Projeto Teste

Teste unitário para cada serviço disponibilizado.

Projeto 5 – Camada WebService do Serviço (opcional – veja Bonus 1,0 ponto trabalho 3)

Criar projeto para Expor via WebServices os serviços.

OBSERVAÇÕES GERAIS:

- Espera-se todo tipo de validação no serviço de negócio, como validação de todos os atributos / objetos de parâmetro de entrada. Eventualmente, isso pode demandar outros serviços intermediários não citados.

O trabalho deve ser implementado de acordo com a arquitetura de desenvolvimento em Camadas, em linguagem JAVA, conforme explicação dada em sala de aula.

3 - Trabalho 3 – Peso 2,0 - Estudo de Caso – Entrega: 16/08/2019

- BÔNUS DE 1,0 Ponto se os serviços implementados forem expostos e consumidos na camada WEB via WebServices.

De acordo com o estudo de caso escolhido (Clínica Médica ou Oficina Mecânica) faça:

a) Diagrama de Sequência dos serviços Implementados na camada de negócio – em conformidade com a Arquitetura de Desenvolvimento discutida em sala de aula e compatível com o código fonte implementado;

e) Conforme o estudo de caso escolhido implemente um Caso de Uso completo: Camada de Apresentação (Interface Web e Interface Teste), Camada de negócio e BD correspondente, conforme Arquitetura de Desenvolvimento discutida em sala de aula.

Casos de Uso:

Clínica Médica: **Manter Paciente**

Oficina Mecânica: **Manter Clientes**

Requisitos de Implementação:

a) Seja Paciente ou cliente, deve-se usar o componente Endereço para obter os objetos de endereço a partir do CEP ou ID;
b) Seja qual for o caso de uso o sistema deve pedir uma foto qualquer e utilizar o microserviço da IBM Cloud ou outra cloud para analisar a foto e compor uma descrição com os itens presentes na foto (índice de acerto maior que 0,5) – mesmos moldes do trabalho do 1. Bimestre.
c) Observar que as classes de negócio mais gerais (usadas em qualquer sistema, tais como Pessoa / Pessoa Física / Pessoa Jurídica, etc devem estar em projetos separados gerando .jar independente. Por exemplo: projeto **MyPessoaBO**
d) **Observar que as classes de negócio específicas do estudo de caso escolhida deve estar em outro projeto. Por exemplo: MySistemaXXXXBO**
e) **As implementações de comportamento referentes ao estudo de caso escolhido devem seguir a mesma recomendação de projeto referente ao endereço. Por exemplo: MySistemaXXXXBO, MySistemaXXXServico, , MySistemaXXXWeb , MySistemaXXXTeste**
f) Implementar todas as camadas para o caso de uso.
g) Os serviços devem ser especificados (design) de acordo com o padrão de Arquitetura em Camadas discutida em sala de aula
h) Tratamento de exceção- ter classes Exception diferenciadas, para que se possa claramente não confundir erro de infra (conexão com banco, SQL errado, falha de rede, etc) de erro de negócio (por exemplo: “Paciente já cadastrado”, etc.).
i) Diagrama de sequência – Critério de avaliação: Diagrama de sequência coerente com o código fonte do serviço gerado.

Tela para Caso de Uso proposto Paciente ou Cliente

Tela para Manter Paciente / Cliente

Nome: _____

Nome Meio:_____ Ultimo Nome:_____

CPF ou CNPJ: _____ < QUANDO DIGITAR VERIFICAR SE JÁ EXSITE>

CEP: _____ <digita CEP e sistema retorna endereço correspondente>

Endereço: .....

Nro. : _____ Complemento Endereço: _____

Foto: _____ < indicar aqui foto para reconhecimento>

Descrição Foto: _____ <valor preenchido no sistema a partir da FOTO reconhecida via Microserviço da IBM Cloud>

Outros dados da modelagem de classes Cliente / Paciente

<Salvar dados> <Consultar Novo ID: _____ >

Estrutura mínima de projetos esperada:

Projeto 1 – Classes BO gerais

Projeto 2 – Classes BO especificas do estudo de caso

Projeto 3 – Camada de NegócioServiços (Classe serviço, controle e DAO dos objetos)

Projeto 4 – WebService (expõe a classe serviço do projeto 3 via Webservice)

Projeto 5 – Interface Web do estudo de caso.

Informações complementares trabalho parte 3

Objetivo: **Implementar dois serviços dos casos de uso do trabalho do 2. Bimestre de acordo com a especificação dada e caso de uso escolhido anteriormente (Manter Paciente ou Manter Cliente)**

Caso de Uso: Manter Paciente

a) **Criar interface WEB para cadastrar e consultar um paciente**
b) **Implementar a camada de negócio com os serviços abaixo:**

Serviço na classe: **UCManterPacienteManager (ou UCServicosPaciente)**

SERVICO	**Descrição**	**Parâmetro Entrada**	**Parâmetro Saída**
cadastrarPaciente	Incluir novo paciente. Considerar Validadações diversas.	Objeto Paciente	**Em caso de erro:** NegocioException caso erro validação ou Exception se erro infra **Se OK:** Objeto Paciente
consultarPaciente	Consultar os dados cadastrais de um paciente a partir do seu CPF.	Objeto CPF	**Em caso de erro:** NegocioException caso erro validação ou Exception se erro infra **Se OK:** Objeto Paciente com os dados cadastrais e objetos relacionados: endereço, sexo, fones e e-mails , conforme modelagem das classes de negócio.

c) **Refazer diagrama de sequência conforme a implementação sugerida para o serviço cadastrarPaciente().**

Especificação:

Serviço: cadastrarPaciente

Operação esperada (pseudoAlgoritmo):

- Pegar conexão com o BD

- Verificar se há algum paciente já cadastrado com o mesmo CPF. Se existir, retornar erro “Paciente com o mesmo CPF já informado”;

- Validar se o primeiro e último nome do paciente foi informado (mínimo 2 caracteres por campo);

- Validar se CPF do paciente é valido;

- Validar se a lista de telefones informados é valido;

- Validar se o Sexo informado é valido;

- Validar se o CEP e ID do endereço é valido;

- Inserir novo Paciente;

- Se houver telefones, gravar lista de telefones do Paciente;

- Se houver e-mails, gravar lista de e-mails do Paciente;

- Encerrar Transação (commit).

Implementação:

Projeto Camada Interface WEB:

Cenário: Cadastrar um novo Paciente

Após o usuário confirmar a gravação dos dados, no evento onde é tratado este algoritmo é esperado o seguinte código:

a) De / Para da estrutura de dados (campos) do formulário WEB para as classes de negócio envolvidas (Paciente, fone, ddd, cpf, email, sexo/gênero, endereço, etc.)
b) Instanciar a classe **UCManterPacienteManager chamando o serviço cadastrarPaciente (paciente);**
c) Mostrar erro caso receba alguma Exception.

Cenário: Consultar um Paciente

Após o usuário informar o número do CPF :

a) De / Para da estrutura de dados (campos) do formulário WEB para as classes de negócio envolvidas (CPF).
b) Instanciar a classe **UCManterPacienteManager chamando o serviço consultarPaciente (cpf)**
c) Mostrar erro caso receba alguma Exception;
d) Mostrar os dados do paciente na interface

Projeto Camada de Negócio:

Serviço: cadastrarPaciente, classe UCManagerManterPaciente (ou UCServicosPaciente)

Assinatura Método classe:

Paciente cadastrarPaciente (Paciente paciente) throws Exception

Operação esperada (pseudoAlgoritmo):

- Pegar conexão com o BD (pegar uma conexão válida a partir de uma classe de conexão com o SGBD);

Parte: VALIDAR DADOS para Cadastrar

- Instanciar a classe **ColPaciente** e passar esta conexão para esta classe;

- Chamar o método obterPacientePorCPF (CPF cpf) throws NegocioException, Exception que estará na classe **ColPaciente**.

- Chamar o objeto Paciente e chamar o método **validarObjeto**();

- Instanciar a classe **ColFonePaciente**, passando a conexão BD para esta instância.

- Chamar o método validarFones (Fone[] listaFonesPaciente) throws NegocioException, Exception da classe ColFonePaciente

- Instanciar a classe **ColSexo**, passando a conexão BD para esta instância.

- Chamar o método obterSexoPorSigla (Sexo sexo) throws NegocioException, Exception da classe ColSexo;

- Instanciar a classe **UCEnderecoManager e chamar o serviço** obterEnderecoPorID. Considera-se que passando um ID inválido este serviço retorne um NegocioException ou alguma Exception com mensagem de erro;

Parte: PERSISTIR DADOS

- Chamar método da classe ColPaciente,   inserirPaciente (paciente)

- Chamar método da classe ColFonePaciente, inserirListaFones (listaFones do paciente)

- Chamar método da classe ColEmailPaciente, inserirListaEmails (listaEmails do paciente)

ENCERRAR TRANSAÇÃO

- Encerrar transação: Pegar a conexão aberta e enviar a instrução COMMIT para o SGBD.

Classe: ColPaciente

Refere-se a classe de controle dos objetos Paciente.

Cada objeto de negócio tem uma classe desta.

Método: obterPacientePorCPF (CPF cpf)

Operação:

- instanciar classe DAO e passar a conexão do BD

- chamar método selectPacientePorCPF da classe DAOPaciente e retornar objeto Paciente populado.

Classe: DAOPaciente

Refere-se a classe que persiste e interage com o BD (persistência) referente a objetos do tipo Paciente. Cada classe persistente tem uma.

Método: obterPacientePorCPF (CPF cpf)

Operação:

- Enviar SQL (Select * from tabeladoPaciente where cpf = nro cpf informado) para o SGBD através da conexão válida com o BD;

- Realizar o DE/Para do Resultset para a estrutura de objetos modelado.

- Não havendo nenhum registro(tupla, linha) no resultSET gerar exception

new NegocioException (“Paciente com o CPF Informado não existe”).

Classe: Paciente

Classe de negócio que mapea a estrutura de dados e objetos relacionados ao Paciente conforme requisitos e padrões estabelecidos no projeto.

Esta classe tem como propósito principal no design servir de instância válida para troca de parâmetros entre as camadas (camada WEB manda dados de Paciente através desta classe e camada de negócio recebe e reenvia dados de Paciente por esta classe).

Método: validaObjeto() throws NegocioException

- Validar se alguns atributos tem o formato básico conforme requisitos deste.

Operação:

Verificar se o tamanho do atributo primeiroNome tem no mínimo 2 bytes;

- Se não estiver, lançar exception NegocioException(“Primeiro Nome invalido”);

Verificar se o tamanho do atributo ultimoNome tem no mínimo 2 bytes;

- Se não estiver, lançar exception NegocioException(“Ultimo Nome invalido”);

Verificar se há CPF informado;

- Se não estiver, lançar exception NegocioException(“CPF não informado”).

Caso de Uso: Manter Cliente

a) **Criar interface WEB para cadastrar e consultar um cliente - – Expor os serviços via WebServices**
b) **Implementar a camada de negócio com os serviços abaixo:**

Serviço na classe: **UCManterClienteManager (ou UCServicosCliente)**

SERVICO	**Descrição**	**Parâmetro Entrada**	**Parâmetro Saída**
cadastrarCliente	Incluir novo paciente. Considerar Validadações diversas.	Objeto Cliente	**Em caso de erro:** NegocioException caso erro validação ou Exception se erro infra **Se OK:** Objeto Cliente
consultarCliente	Consultar os dados cadastrais de um cliente a partir do CPF ou CNPJ.	CPF cpf, CNPJ cnpj	**Em caso de erro:** NegocioException caso erro validação ou Exception se erro infra **Se OK:** Objeto Cliente com os dados cadastrais e objetos relacionados: endereço, fones e e-mails , conforme modelagem das classes de negócio.

c) **Refazer diagrama de sequência conforme a implementação sugerida para o serviço cadastrarCliente().**

Especificação:

Serviço: cadastrarCliente

Operação esperada (pseudoAlgoritmo):

- Pegar conexão com o BD

- Verificar se há algum cliente já cadastrado com o mesmo CPF ou CNPJ. Se existir, retornar erro “Cliente com o mesmo CPF ou CNPJ já informado”;

- Validar se o primeiro e último nome do cliente foi informado (mínimo 2 caracteres por campo) – se PessoaFisica, se PessoaJuridica, ver se o nome tem no mínimo 2 caracteres;

- Validar se CPF ou CNPJ do cliente é valido;

- Validar se a lista de telefones informados é valido;

- Validar se o Sexo informado é valido (se PessoaFisica);

- Validar se o CEP e ID do endereço é valido;

- Inserir novo Cliente;

- Se houver telefones, gravar lista de telefones do Cliente;

- Se houver e-mails, gravar lista de e-mails do Cliente;

- Encerrar Transação (commit).

Implementação:

Projeto Camada Interface WEB:

Cenário: Cadastrar um Novo Cliente

Após o usuário confirmar a gravação dos dados, no evento onde é tratado este algoritmo é esperado o seguinte código:

- De / Para da estrutura de dados (campos) do formulário WEB para as classes de negócio envolvidas (Cliente, fone, ddd, cpf, email, endereço, etc.)
- Instanciar a classe **UCManterClienteManager chamando o serviço cadastrarCliente (cliente);**
- Mostrar erro caso receba alguma Exception.

Cenário: Consultar um Cliente

Após o usuário o número do CPF ou CNPJ do cliente:

- De / Para da estrutura de dados (campos) do formulário WEB para as classes de negócio envolvidas (CPF ou CNPJ) – se informar cpf, sete NULL para cnpj e vice-versa.
- Instanciar a classe **UCManterClienteManager chamando o serviço consultarCliente (cpf, cnpj)**
- Mostrar erro caso receba alguma Exception;
- Mostrar os dados do cliente na interface

Projeto Camada de Negócio:

Serviço: cadastrarCliente, classe UCManterClienteManager (ou UCServicosCliente)

Assinatura Método classe:

Paciente cadastrarCliente(Cliente cliente) throws Exception

Operação esperada (pseudoAlgoritmo):

- Pegar conexão com o BD (pegar uma conexão válida a partir de uma classe de conexão com o SGBD);

Parte: VALIDAR DADOS para Cadastrar

- Instanciar a classe **ColCliente** e passar esta conexão para esta classe;

- Se cliente for pessoa Fisica chamar o método obterClientePorCPF (CPF cpf) throws NegocioException, Exception que estará na classe **ColCliente**

- Se cliente for pessoa Juridica chamar o método obterClientePorCNPJ (CNPJ cnpj) throws NegocioException, Exception que estará na classe **ColCliente**

- Chamar o objeto Cliente e chamar o método **validarObjeto**();

- Instanciar a classe **ColFoneCliente**, passando a conexão BD para esta instância.

- Chamar o método validarFones (Fone[] listaFonesCliente) throws NegocioException, Exception da classe ColFoneCliente

- Instanciar a classe **ColSexo**, passando a conexão BD para esta instância.

- Se cliente for pessoa Fisica chamar o método obterSexoPorSigla (Sexo sexo) throws NegocioException, Exception da classe ColSexo;

- Instanciar a classe **UCEnderecoManager e chamar o serviço** obterEnderecoPorID. Considera-se que passando um ID inválido este serviço retorne um NegocioException ou alguma Exception com mensagem de erro;

Parte: PERSISTIR DADOS

- Chamar método da classe ColCliente,   inserirCliente (cliente)

- Chamar método da classe **ColFoneCliente**, inserirListaFones (listaFones do cliente)

- Chamar método da classe ColEmailCliente, inserirListaEmails (listaEmails do cliente)

ENCERRAR TRANSAÇÃO

- Encerrar transação: Pegar a conexão aberta e enviar a instrução COMMIT para o SGBD.

Classe: ColCliente

Refere-se a classe de controle dos objetos Cliente.

Cada objeto de negócio tem uma classe desta.

Método: obterClientePorCPF (CPF cpf)

Operação:

- instanciar classe DAO e passar a conexão do BD

- chamar método selectClientePorCPF da classe DAOCliente e retornar objeto Paciente populado.

Método: obterClientePorCNPJ (CNPJ cnpj)

Operação:

- instanciar classe DAO e passar a conexão do BD

- chamar método selectClientePorCNPJ da classe DAOCliente e retornar objeto cliente populado.

Classe: DAOCliente

Refere-se a classe que persiste e interage com o BD (persistência) referente a objetos do tipo Cliente. Cada classe persistente tem uma.

Método: selectClientePorCPF (CPF cpf)

Operação:

- Enviar SQL (Select * from tabeladoCliente where cpf = nro cpf informado) para o SGBD através da conexão válida com o BD;

- Realizar o DE/Para do Resultset para a estrutura de objetos modelado.

- Não havendo nenhum registro(tupla, linha) no resultSET gerar exception

new NegocioException (“Paciente com o CPF Informado não existe”).

Método: selectClientePorCNPJ (CNPJ cnpj)

Operação:

- Enviar SQL (Select * from tabeladoCliente where cnpj = nro cnpj informado) para o SGBD através da conexão válida com o BD;

- Realizar o DE/Para do Resultset para a estrutura de objetos modelado.

- Não havendo nenhum registro(tupla, linha) no resultSET gerar exception

new NegocioException (“Paciente com o CPF Informado não existe”).

Classe: Cliente

Classe de negócio que mapea a estrutura de dados e objetos relacionados ao Cliente conforme requisitos e padrões estabelecidos no projeto. Observar, que como o cliente pode ser tanto pessoa física quanto jurídica, então não há herança nas classes diretamente mais uma composição 1 para 1 com a classe Pessoa, onde espera-se que haja instâncias do tipo

PessoaFisica ou PessoaJuridica conforme o tipo de cliente a ser cadastrado (você pode considerar que todo cliente que tiver número CNPJ será PessoaJuridica e se não ter este número, será sempre Pessoa Física).

Esta classe tem como propósito principal no design servir de instância válida para troca de parâmetros entre as camadas (camada WEB manda dados de Cliente através desta classe e camada de negócio recebe e reenvia dados de Cliente por esta classe).

Método: validaObjeto() throws NegocioException

- Validar se alguns atributos tem o formato básico conforme requisitos deste.

Operação: (Se PessoaFisica) Verificar se o tamanho do atributo primeiroNome tem no mínimo 2 bytes; - Se não estiver, lançar exception NegocioException("Primeiro Nome invalido"); Verificar se o tamanho do atributo ultimoNome tem no mínimo 2 bytes; - Se não estiver, lançar exception NegocioException("Ultimo Nome invalido"); Verificar se há CPF informado; - Se não estiver, lançar exception NegocioException("CPF não informado").

Operação: (Se PessoaJuridica) Verificar se o tamanho do atributo nome tem no mínimo 2 bytes; - Se não estiver, lançar exception NegocioException("Nome invalido"); Verificar se há CNPJ informado; - Se não estiver, lançar exception NegocioException("CNPJ não informado").