**Questão 1**

O que você entende por arquitetura MVC?

Entendo como um desenho de interfaces dividido em 3 camadas, onde cada camada tem sua responsabilidade própria.

**Model**: também conhecida como Modelo, representa a lógica do negócio. Aqui é definido as classes que compõem o sistema; faz a comunicação com o banco de dados. Depois de receber as informações do Controller, devolve a resposta.

**View**: também conhecida como Visão, é a camada responsável pela interface com o usuário. Nesta camada devem ter apenas recursos ligados à aparência (botões, mensagens, relatórios etc.), passa as informações para o Controller.

**Controller:** também conhecido como Controlador, atua como intermediário entre a View e o Model, recebe as informações da View e repassa para o Model, depois de refletidas as informações, a View reflete os dados; gosto de dizer que o Controller é como o maestro da orquestra.

**Questão 2**

Escolha um padrão de projeto e escreva seu conceito, objetivo e um cenário de uso.

**Abstract Factory:**

**Conceito**: Classe responsável que irá se relacionar com vários objetos sem especificar suas classes de origem.

**Objetivo**: Imagine uma loja onde podemos comprar os produtos, porém em kits, não de forma individual.

**Cenário de uso:**

*Vamos imaginar que iremos desenvolver um sistema para nota fiscal onde teremos o cliente e o número da nota.*

**1º Criar uma interface:**

unit model.interfaces;

interface

type

iPessoa = interface

['{EF2E972B-D8CB-4CAC-98EC-B5036E596DCB}']

function Nome: String;

end;

iNotaFiscal = interface

['{421786B6-F82B-4656-B287-84841938CCB7}']

function Numero: Integer;

end;

iFactory = interface

['{7E6FA38E-04D7-4556-B86E-C24FCA3D1525}']

function Pessoa: iPessoa;

function NotaFiscal: iNotaFiscal;

end;

implementation

end.

**2º Criar uma classe para implementar a interface iPessoa**

unit model.pessoa;

interface

uses

model.interfaces;

type

TModelPessoa = class(TInterfacedObject, iPessoa)

private

public

constructor create;

destructor destroy; override;

class function New: iPessoa;

function Nome: String;

end;

implementation

{ TModelPessoa }

constructor TModelPessoa.create;

begin

end;

destructor TModelPessoa.destroy;

begin

inherited;

end;

class function TModelPessoa.New: iPessoa;

begin

Result := Self.Create;

end;

function TModelPessoa.Nome: String;

begin

Result := 'Márcio';

end;

end.

**3º Criar uma classe para implementar a interface iNotaFiscal**

unit model.notafiscal;

interface

uses model.interfaces;

type

TModelNotaFiscal = class(TInterfacedObject, iNotaFiscal)

private

public

constructor create;

destructor destroy; override;

class function New: iNotaFiscal;

function Numero: Integer;

end;

implementation

{ TModelPessoa }

constructor TModelNotaFiscal.create;

begin

end;

destructor TModelNotaFiscal.destroy;

begin

inherited;

end;

class function TModelNotaFiscal.New: iNotaFiscal;

begin

Result := Self.Create;

end;

function TModelNotaFiscal.Numero: Integer;

begin

Result := 1234;

end;

end.

**4º Criar uma classe para implementar a interface iFactory**

unit model.factory;

interface

uses model.interfaces;

type

TModelFactory = class(TInterfacedObject, iFactory)

private

public

constructor create;

destructor destroy; override;

class function New: iFactory;

function Pessoa: iPessoa;

function NotaFiscal: iNotaFiscal;

end;

implementation

uses

model.notafiscal,

model.pessoa;

{ TModelPessoa }

constructor TModelFactory.create;

begin

end;

destructor TModelFactory.destroy;

begin

inherited;

end;

class function TModelFactory.New: iFactory;

begin

Result := Self.Create;

end;

function TModelFactory.NotaFiscal: iNotaFiscal;

begin

Result := TModelNotaFiscal.New;

end;

function TModelFactory.Pessoa: iPessoa;

begin

Result := TModelPessoa.New;

end;

end.

**O que está acontecendo no lado da View?**

Uma das vantagens da Abstract Factory é diminuir o acoplamento, se eu não tivesse feito dessa forma, no lado da view eu teria que declarar as classes

model.pessoa

model.notafiscal

No caso Abstract Factory, somente ela estaria declarada na view e toda criação das outras classes (model.pessoa, model.notafiscal), métodos, implementações, estaria dentro da classe model.factory, diminuindo assim o acoplamento dentro da view.

**Objetivo do Projeto, ferramentas e descritivo do projeto, tratamento de erros, casos de testes:**

**Objetivo:**

1. Desenvolver um software capaz de consultar um cep em 3 api’s diferentes.

**Ferramenta:**

1. Delphi 10.4.2, Boss, Dataset-Serialize, RESTRequest4Delphi
2. Links para download:

[GitHub - HashLoad/boss: Dependency Manager for Delphi and Lazarus](https://github.com/HashLoad/boss)

boss install github.com/viniciussanchez/dataset-serialize-adapter-restrequest4delphi

boss install github.com/viniciussanchez/RESTRequest4Delphi

Seguir as instruções do GitHub para instalação.

**Descritivo do projeto**

1. O projeto consiste em, ao digitar o cep e clicar no botão **Consultar** será passado essa informação para a ApiCep que irá consultar o cep passado e retornar as informações, porém, caso a ApiCep esteja fora do ar (erro 503 ou o serviço não esteja autorizado, erro 403), então o software irá tentar consultar o cep na api ViaCep e tentará retornas as informações porém caso o ViaCep também esteja fora do ar (erro 503 ou o serviço não esteja autorizado, erro 403), o software irá tentar consultar o cep na ApiAwesomeApi e tentará retornar as informações, porém caso a ApiAwesomeApi também esteja fora do ar (erro 503 ou o serviço não esteja autorizado, erro 403), o software retornará a seguinte mensagem:

**As apis do viacep, apicep, awesomeapi estão fora do ar. Tente mais tarde ou a consulta não foi autorizada. Tente mais tarde**

Caso qualquer dessas api’s esteja no ar e a consulta for autorizada, então será retornado as informações daquele cep, como (cep, endereço, bairro, cidade, estado, complemento, código do IBGE).

OBS: **No caso da API awesomeApi, será retornado também a latitude e a longitude da localização do cep.**

**Tratamento de erros**

1. Caso o usuário digite um cep com tamanho menor que 8, retornará uma mensagem: CEP inválido.
2. Caso as api’s não consigam localizar o cep digitado, retornará uma mensagem: CEP não encontrado.

**Casos de testes:**

1. Dentro do projeto tem uma pasta chamada “testes\_unitarios”, criei um projeto de testes com DUnitX.

Criei uma variável com o retorno desejado.

Criei uma variável com resultado do cep passado, depois faço a comparação das duas variáveis para verificar se os testes retornam verdadeiro ou falso.

OBS: Criado os testes para as 3 api’s.