CODIFICAÇÃO BACK-END SP2

GRUPOS DE RESPOSTAS À REQUISIÇÕES HTTP:

Respostas de informação (100 – 199)

Respostas de sucesso (200 – 299)

Redirecionamentos (300 – 399)

Erros do cliente (400 – 499)

Erros do servidor (500 – 599)

TIPOS DE SITE

ESTÁTICO => Retorna sempre o mesmo conteúdo. Não possui ferramentas de gerenciamento incorporadas. Sõa usados para consultar informações sobre as empresas, como sedrviços e produtos, sem a necessidade de atualização contínua.

Caso houver alterações, será necessário mexer no código-fonte (HTML, CSS e JavaScript)

DINÂMICO => Todo ou parte do conteúdo é gerado dinamicamente. Há gerenciamento de conteúdo (inclusão, alteração ou exclução de informações).

São exemplos: Portais de notícias, redes sociais, bancos de imagens e sites com muito conteúdo.

As aplicações Back-End pdoem ler informações dos bancos de dados e organizá-las dinamicamente.

FRAMEWORK

é um conjunto de bibliotecas utilizado para desenvolver aplicações com uma estrutura pré-definida, podendo simplificar o processo de desenvolvimento e de ganho de performance.

Biblioteca: Conjunto de subprogramas independentes que permitem o compartilhamento, o uso modular e a personalização de códigos.

Linguagem C# - Framework : .NET

C# - Lançado em 2002

O .NET é uma plataforma de desenvolvimento de software cujo tempo de vida e maturidade de desenvolvimento a tornam uma das plataformas mais utilizadas por empresas. Além de compatível com o C#, o .NET oferece suporte a outras linguagens, como o F# e o Visual Basic. O .NET fornece a base para a criação de vários tipos de aplicativos, principalmente, aplicativos desktop, aplicativos móveis, aplicativos web ou jogos, e tem suporte para os sistemas operacionais Windows, macOS e Linux.

TIPOS DE APLICATIVOS .NET

APLICATIVOS DESKTOP -> Não fazem uso de um navegador WEB

APLICATIVOS MÓVEIS -> Instalados e utilizados em aparelhos celulares (Android ou iOS), disponíveis nas web Stores

APLICATIVOS WEB -> Armazenados em um servidor e podem resonder tanto a conteúdos estáticos ou dinâmicos.

.NET --🡪 pacotes de distribuição:

SDK: Software Development Kit

Viabiliza o processo de compilação e de execução da criação de programas . Possui o RUNTIMe junto

RUNTIME: Se houver a aplicação pronta, é possível usar somente o Runtime para seu funcionamento.

**LINGUAGEM DE PROGRAMAÇÃO – CLASSIFICAÇÃO**

Podem ser : Compiladas, Interpretadas, Baixo ou Alto nível

Tipo de PARADIGMAS

PARADIGMA : É uma abordagem utilizada para resolver problemas a partir de técnicas e de ferramentas disponíveis e que pdem ser aplicadas a diferentes linguagens.

São abordagens diferentes para a criação de aplicações

**LINGUAGEM ESTRUTURADA**

**Controle de repetição, de condição e de sequência**

**ORIENTADA A OBJETOS (OOP)**

**Aproxima a representação de itens do mundo real aos do mundo digital**

**Baseia-se em CLASSES E OBJETOS**

**PROGRAMAÇÃO FUNCIONAL**

**o processo de construção de sistemas baseado em funções puras, ou seja, evitando-se o *compartilhamento de estado*. A partir de um dado de entrada, constrói-se uma saída esperada**

**PROGRAMAÇÃO IMPERATIVA**

Baseada em instruções e comandos, ordens sob as quais o sistema deve operar (como e o que o sistema deve realizar.

CARACTERÍSTICAS DA LINGUAGEM C#:

1. Alto Nível
2. Orientada a objetos
3. Fortemente tipada
4. Case-Sensitive
5. Compilada :

Por ser uma linguagem compilada, o C# fornece a visualização dos erros em tempo de desenvolvimento e, em caso de falha, o código não é executado

PARADIGMA : PROGRAMAÇÃO ORIENTADA A OBJETOS

Atributos (Características, Propriedades)

Métodos (Funcionalidades, ações)

Atributos + Métodos = Classe (Conceito, Molde)

**=============================================================================**

Propriedades e Métodos

MODIFICADORES DE ACESSO

***Public***

trata-se de qualquer classe pode ter acesso, ou seja, o acesso é irrestrito.

***Protected***

pode ser acessado apenas pela própria classe ou pelas suas subclasses, ou seja, o acesso é limitado à classe ou a tipos que derivem da mesma classe

***Private***

apenas os métodos da própria classe pode manipular o atributo, ou seja, o acesso é limitado à própria classe.

Ex: class Livro

{ public string name; // public é modificador de acesso da propriedade nome

Private float preco; / private é modificador de acesso da propriedade preço

Public void CalcularDesconto(){} // public é modificador de acesso do método

}

String é uma variável de texti

Float é um tipo de variável numérica

Void é o tipo de retorno da função

===========================================

INSTÂNCIA = OBJETO DE UMA CLASSE

// Criando uma classe

OBS: **RECOMENDAÇÃO**

Uma regra simples de ser utilizada na identificação e análise de um problema é a seguinte: procurar por substantivos (nomes) para as **classes** e, para os **métodos**, procurar por verbos (ações).

public class Livro

{

public int Id {get; set;}

public string Titulo {get; set;}

public double Preco { get; set; }

}

Instanciando:

Tipo nome = método construtor ();

Livro livro = new Livro();

Static void Main (string[] args)

{

// criando uma instancia da classe Livro

Livro livro = new Livro();

Livro Preco = 10;

Console.Writeline(livro.Preco);

}

COMPOSIÇÃO DE UMA ESTÂNCIA:

Objetos

Listas =

public class Livro

{

public int Id {get; set;}

public string Titulo {get; set;}

public double preco {get; set; }

public List<Autor> Autores {get; set;}

}

Class Autor

{  
 public string Nome {get; set; }

Public string Email {get; set; }

}

List<Autor> autores = new List <Autor>();

autores.Add(new Autor { Nome = “Jorge Amado”});

autores.Add(new Autor { Nome = “Agatha Christie”});

Livro livro = new Livro();

Livro.Autores = autores;

livro.autores.ForEach (c => Console.WriteLine(c.Nome));

**PILARES DE POO**

ENCAPSULAMENTO / HERANÇA / ABSTRAÇÃO / POLIMORFISMO

ABSTRAÇÃO:

Atributo/ método que se repete

Super classe ou classe abstrata

Conceitual (acontece na construção do código)

**Não tem instância**

* Quando há atribuitos comuns em duas classes, criamos uma super-classe, chamada abstrata com esses atributos.

public abstract Livro { public double preco { get; set;}}

HERANÇA

Como **não podemos instanciar a SuperClasse**, instanciamos a classe filha da SuperClasse, que herda todos os atributos e métodos

* Public abstract livroInfantil : Livro { public double preco; }
* Public abstract livroAdulto: Livro { }

Para instanciar:

// Program.cs

Static void Main ( string [] args)

{

livroInfantil c = new livroInfantil();

c.preco = 10;

Console.WriteLine(c.preco);

}

POLIMORFISMO

1. SOBRESCRITA

Métodos da classe filho sobrepõe o comportamento da classe Pai

SuperClasse:

class Compra

{

public virtual void CalcularDesconto(){ }

}

class CompraDebito : Compra

{

public override void CalcularDesconto()

{ // cálculo para compra em débito

}

}

Class CompraCredito: Compra

{

Public override void CalcularDesconto()

{ // cálculo para compra em crédito

}

}

1. SOBRECARGA

Métodos com mesmo nome com cargas, argumentos e parâmetros diferentes

Sempre dentro da mesma classe

class CompraCreditos : Compra

{

Public virtual void CalcularJuros()

{ // cálculo dos juros no crédito para compras à vista }

Public virtual void CalcularJuros ( int parcelas )

{ // cálculo dos juros no c´redito conforme o nr. De parcelas }

ENCAPSULAMENTO (PROTEGER, ENCAPSULAR)

Private set ( só altera o valor dentro da mesma classe)

Nível maior de segurança

Ex.: Para alterar o saldo (abaixo) criamos método para depositar ou sacar

Public class Conta

{

public double saldo { get; private set; }

Public void Depositar (double valorDepositado)

{

this.saldo +- valor Deppositado;

}

Public void Sacar (double valorSolicitado)

{

This.saldo += valorSolicitado;

}