TÓPICOS INTEGRADORES I - ADS

UNIDADE II

ser educacional

Edição, revisão e diagramação:
Equipe de Desenvolvimento de Material Didático EaD

Cordeiro Brasileiro, Alberto Fagner

Tópicos Integradores I - ADS - Unidade 2 -
Recife: Grupo Ser Educacional, 2018.

Grupo Ser Educacional
www.sereducacional.com

SUMÁRIO

PARA INÍCIO DE CONVERSA

Bem-vindo(a) estudante!

Você está convidado(a), a partir de agora, a partilhar o conhecimento integral desenvolvido até esta etapa do seu curso. Vamos continuar o Tópico Integrador numa estratégia de ensino e aprendizagem que objetiva proporcionar a interdisciplinaridade dos temas abordados nos módulos.

Nesta sala, teremos instrumentos de integração entre ensino e a reflexão prática do que foi trabalhado até este momento no seu curso. Nesta etapa de sua vida acadêmica, este processo deve fornecer subsídios para a avaliação das competências relacionadas ao perfil profissional adquirido nos temas tratados nos módulos.

Esta disciplina, conforme você pode ver no AVA, está dividida em quatro unidades das quais já passamos pela primeira. Na unidade anterior estudamos sobre os pilares de um projeto integrador, trazendo alguns conceitos relevantes como, por exemplo, a importância da leitura, a segmentação do processo de software, o arcabouço para multidisciplinaridade, a gestão do conhecimento e da competência, entre outros.

Já nesta unidade iremos abordar alguns elementos importantes acerca da aplicação e uso das linguagens de programação e seus paradigmas no contexto dos negócios empresariais e na automação de tarefas.

Na terceira unidade entenderemos a importância da propriedade intelectual e a lei de software que foram desenvolvidas no intuito de deixar claros os deveres e direitos de cada parte, já que a tecnologia tem alcançado tamanha importância dentro das empresas. E, na última unidade aplicaremos os conhecimentos adquiridos nas diversas disciplinas do curso de Análise e Desenvolvimento de Sistemas (ADS) sobre a aplicação de TI à luz da competitividade nos negócios empresariais, estudando casos reais para entendermos o nosso papel dentro desse processo.

Em cada unidade você encontrará vídeos que apresentarão, de forma consolidada, os objetivos de cada unidade. Nos Guias de Estudo, serão encontradas todas as referências aos conteúdos complementares indicados, os vídeos internos das disciplinas passadas e os vídeos externos (YouTube, por exemplo) a fim de ampliar ainda mais o seu conhecimento acerca do tema tratado em cada unidade.

Ao final de cada unidade você deverá realizar as atividades que constam no Ambiente Virtual de Aprendizagem (AVA).

Esperamos que nas próximas semanas você desfrute desta disciplina que foi feita especialmente para você e que ao final dela esteja mais preparado(a) e capacitado(a) para enfrentar o mercado profissional.

ORIENTAÇÕES DA DISCIPLINA

Esta sala virtual não trata de propor a eliminação de disciplinas, mas sim da criação de movimentos que propiciem o estabelecimento de relações entre as mesmas, tendo como ponto de convergência a ação que se desenvolve numa PARTICIPAÇÃO cooperativa e reflexiva através do Ambiente Virtual de Aprendizagem, webconferências, Guias de estudo e vídeoaulas das diversas disciplinas do curso de ADS, unificados para este fim.

Ao final de cada unidade você encontrará atividades avaliativas como questionários relacionados com os assuntos vistos no seu guia de estudo.

Mas, antes de iniciar o estudo do Tópico Integrador, faça a você mesmo as seguintes perguntas:

- O que é linguagem de programação?
- O que é paradigma de programação?
- Qual é a importância desse conteúdo para a minha formação profissional?
- Quais as principais linguagens?
- Quais os principais paradigmas do mercado?
- Como posso escolher o melhor conjunto de ferramentas para solução dos meus problemas?

Vamos começar?

Para obter essas e outras respostas, é só se debruçar com disposição sobre o roteiro de estudo. Siga em frente com determinação! Vamos juntos!

LINGUAGENS DE PROGRAMAÇÃO

Caro(a) estudante, de forma conceitual, uma linguagem de programação é um método padronizado para comunicar instruções para um computador.

Podemos entendê-la, como um conjunto de regras usadas para definir um programa de computador que permite ao programador especificar, precisamente, como o computador deve se comportar e como os dados serão armazenados ou transmitidos a partir de várias fontes diferentes.

Uma linguagem de programação pode expressar algoritmos que, de maneira bastante precisa, correspondem a todo o conjunto de palavras que formam a linguagem de programação sendo denominada de código-fonte de um software. Esse código-fonte, depois de traduzido, se transforma em código de máquina sendo, por sua vez, possível de ser entendido pelo processador da máquina.

O motivo pelo qual as linguagens de programação foram desenvolvidas foi para facilitar o desenvolvimento de softwares pelos seus programadores. Anteriormente, no início da programação, os desenvolvedores

programavam em linguagens chamadas de baixo nível, pois eram linguagens muito parecidas com as linguagens de máquina, ou seja, a linguagem que é entendida pelo processador, porém algum tempo depois desenvolvedores melhoraram essas linguagens criando as linguagens de alto nível.

As linguagens de alto nível tentam trazer a linguagem de programação para o mais próximo possível da linguagem humana, facilitando assim a interpretação dos seus códigos e sintaxe. Dessa maneira, com as linguagens de alto nível, os desenvolvedores ganharam mais produtividade e eficiência no seu dia-a-dia de desenvolvimento de códigos para produção de softwares dos mais diversos tipos.

GUARDE ESSA IDEIA!

As linguagens de programação são de suma importância para que os desenvolvedores possam escrever programas cada vez mais organizados e com maior qualidade, entregando assim softwares cada vez melhores para os usuários.

VISITE A PÁGINA

Para saber mais sobre a história das linguagens acesse o LINK

VOCÊ SABIA?

A linguagem de programação é um tipo de linguagem escrita e formal, porque ela tem sua sintaxe bem definida e serve para gerar programas, ou seja, softwares. Um software pode ser desenvolvido para funcionar em computadores, dispositivos móveis, ou em qualquer equipamento que permite a sua execução.

Muitas linguagens têm propósitos específicos, como criar um software ou controlar um carro. Um aplicativo do seu celular provavelmente foi desenvolvido a partir de uma linguagem de programação, um game que roda no seu console ou pc também, uma TV Smart que possui a funcionalidade de navegar na internet, também tem uma linguagem de programação tanto para controlar o software como de hardware para controlar seus componentes eletrônicos. Já um ar-condicionado, por exemplo, que não tem uma interface de comunicação com o usuário, tem um software embarcado que controla circuitos eletrônicos, esse software é chamado de firmware, e também é escrito usando uma linguagem de programação.

A internet das coisas (IoT) demanda um crescimento cada vez maior de soluções em firmware e software para dispositivos comuns do nosso dia a dia, trazendo cada vez mais inteligência para dentro desses dispositivos, como geladeiras, ar condicionado, ou até mesmo um fogão.

PRINCIPAIS LINGUAGENS DE PROGRAMAÇÃO

Em continuidade ao nosso assunto trago para você aluno(a), conforme o índice TIOBE, a classificação das principais linguagens de programação em agosto de 2018:

Posição	Linguagem de programação	Posição	Linguagem de programação
1	Java	11	Swift
2	C	12	Delphi/Object Pascal
3	C++	13	MATLAB
4	Python	14	Objective-C
5	Visual Basic .NET	15	Ruby
6	C#	16	Perl
7	PHP	17	GO
8	JavaScript	18	R
9	SQL	19	Visual Basic
10	Assembly language	20	PL/SQL

Fonte: Autoria própria com base no LINK

VOCÊ SABIA?

A TIOBE é uma das maiores empresas especializadas em qualidade de software do mundo.

VISITE A PÁGINA

Se você quiser saber um pouco mais sobre as linguagens de programação listadas acima e suas características, acesse o site e fique por dentro LINK

Agora, aluno (a), vamos abordar um pouco sobre as principais linguagens do ranking acima. A linguagem Java você já conhece, está em primeiro lugar, e é a linguagem base do seu curso. Temos também a linguagem C que é uma linguagem bem antiga, mas muito usada até hoje bem como C++ que a versão de C com suporte a orientação a objetos. Temos, ainda, o Python que tem se destacado muito ultimamente e é uma linguagem multiparadigma de fácil compreensão e muito utilizada em aplicações web vale a pena saber mais sobre essa linguagem que está muito em alta na atualidade.

PARA PESQUISAR

No site da linguagem Python Brasil você pode encontrar várias informações e pode até começar a aprender a programar nela. Clique no link e estude um pouco sobre essa linguagem você não vai se arrepender. LINK

Ainda temos o Java Script que tem sido amplamente utilizado também em vários projetos. Ainda mais depois da criação do nodeJS que possibilita com que o código seja executado no lado do servidor e não no lado do cliente, pois a princípio todo JavaScript era baseado em códigos que executavam na máquina do cliente no seu navegador e isso deixava o código exposto. Temos também o SQL como a principal linguagem utilizada para persistência de dados. O SQL é utilizado como linguagem base de vários bancos de dados, entre eles, o SQL Server da Microsoft, o MySQL e o PostgreSQL esses bancos utilizam o SQL como linguagem principal, logo é muito importante que saibamos um pouco sobre essa linguagem, pois isso vai nos ajudar na construção dos nossos softwares.

DICAS

É muito importante que você separe um tempo para estudar as outras linguagens de programação, pelo menos conhecer um pouco sobre elas sobre como elas funcionam, a que paradigma elas estão diretamente ligadas, ou se elas são multiparadigmas, pois esse estudo vai te ajudar a escolher sempre a melhor linguagem para resolver o problema que você vai encontrar pela frente. Com a diferença entre as linguagens é importante que saibamos que diferenças são essas e como elas podem nos ajudar no nosso dia-a-dia de programação.

PARADIGMAS DE PROGRAMAÇÃO

Caro(a) aluno(a), os paradigmas de programação são um meio de classificar as linguagens de programação, e eles se baseiam em suas funcionalidades. Uma linguagem de programação pode estar classificada em mais de um paradigma, dependendo da forma como ela pode ser escrita.

Um paradigma de programação mostra como o programador pode estruturar e executar o seu programa, um exemplo é a programação orientada a objetos, como um dos paradigmas mais conhecidos da atualidade. A orientação a objetos dá ao programador a capacidade de abstrair dentro do programa uma coleção de objetos que interagem entre si, enquanto que no paradigma funcional vemos a possibilidade de enxergar o programa como uma sequência de funções executadas uma após a outra.

GUARDE ESSA IDEIA!

Dessa maneira como diferentes grupos da engenharia de software propõem diferentes processos de software e suas metodologias, assim também é com as linguagens de programação, pois estas propõem diferentes paradigmas de programação, sendo algumas linguagens desenvolvidas para um paradigma específico e outras que suportam vários paradigmas diferentes.

Os paradigmas são comumente diferenciados pelas técnicas de programação que eles impõem, permitindo ou proibindo determinadas estruturas dentro do código e, dessa maneira, fazendo o programador entender a melhor forma de se tratar aquele problema com uma determinada linguagem.

VISITE A PÁGINA

Para saber mais sobre a história dos paradigmas de programação acesse o LINK

VOCÊ SABIA?

A definição do dicionário Aurélio para "paradigma":

1. Algo que serve de exemplo geral ou de modelo.
2. Conjunto das formas que servem de modelo de derivação ou de flexão.
3. Conjunto dos termos ou elementos que podem ocorrer na mesma posição ou contexto de uma estrutura.

O paradigma, quando relacionado a uma linguagem de programação, é sua identidade. Correspondem às características que juntas definem como ela funciona e qual o problema ela resolve. Como já comentamos anteriormente, algumas linguagens podem trabalhar em mais de um paradigma, são as chamadas multi paradigmas.

PRINCIPAIS PARADIGMAS

Alguns dos principais paradigmas utilizados atualmente no mercado são:

- Funcional
- Lógico
- Declarativo

- Imperativo
- Orientado a Objetos
- Orientado a Eventos

Vamos falar um pouco de cada um deles e como eles funcionam, para que na hora que precisarmos utilizá-los, saibamos como escolher o mais adequado para o nosso projeto de software.

Paradigma Funcional

Este paradigma trata a computação como avaliação de funções matemáticas. Este método enfatiza a aplicação de funções, as quais são tratadas como principais estruturas, ou seja, funções podem ser parâmetros ou valores de entrada para outras funções e podem ser os valores de retorno ou saída de uma função.

Você ficou confuso(a) agora? Vamos ver um exemplo para clarear as coisas.

EXEMPLO

Na matemática temos, por exemplo, uma função f(x): f(x) = x + 2
x é um parâmetro (valor de entrada) e após a expressão ser avaliada chegamos ao resultado.
Se o valor de entrada for 4 o resultado da avaliação da nossa função será 6.

Algumas das linguagens que atendem a esse paradigma: F# (da Microsoft), Lisp, Haskell, Mathematica, R e Erlang.

É possível ainda, utilizar linguagens não-funcionais para programar no paradigma funcional. Como PHP, por exemplo, que é uma linguagem multiparadigma. Fazendo um exemplo em PHP teríamos o seguinte código como resultado, programando de forma funcional.

```
<?php
$sum = function($value) {
   return $value + 2;
};
echo $sum(2); // 4
?>
```

GUARDE ESSA IDEIA!

Linguagens de programação puramente funcionais têm sido mais as usadas de maneira acadêmica, se comparada sua utilização em relação ao desenvolvimento comercial de software. Entretanto algumas linguagens são usadas na indústria, como é o caso da linguagem R usada em estatísticas e da linguagem Erlang usada em aplicações concorrentes.

- Desvantagens do Paradigma Funcional

Na programação funcional não existe algumas estruturas consideradas essenciais como, por exemplo, alocação de variáveis.

- Vantagens do Paradigma Funcional

A principal vantagem do paradigma funcional é alocação automática de memória, com isso o paradigma funcional consegue assegurar o resultado da função para um determinado conjunto de parâmetros, não importando onde ou quando seja avaliado. Dessa maneira a recursividade vira uma arma muito potente nas mãos do programador que usa esse paradigma, sendo mais eficiente até do que as estruturas de repetição do paradigma imperativo.

Paradigma Lógico

Também conhecido como RESTRITIVO muito utilizado em aplicações de Inteligência Artificial. Esse paradigma chega no resultado esperado a partir de avaliações lógico-matemáticas. Entender sobre lógica de predicados ajuda na compreensão de como as linguagens nesse paradigma funcionam.

O programa básico formará conclusões imediatas, a partir de uma lista de premissas e essas conclusões gerarão as sentenças como resultado de suas avaliações.

A idéia básica da programação em lógica é: "oferecer um arcabouço que permita inferir conclusões desejadas, a partir de premissas, representando o conhecimento disponível, de uma forma que seja computacionalmente viável". Palavras do professor Dr. Silvio do Lago Pereira – DTI / FATEC-SP.

Suas principais aplicações estão em prototipação em geral, sistemas especialistas e bancos de dados. A principal linguagem utilizada para esse paradigma é a Prolog.

EXEMPLO

Vamos ver um rápido exemplo para entendermos melhor como funciona:

Proposição: João é um gato.
Regra de inferência: Todo gato é um felino.
Busca: João é um felino?

A resposta para a Busca acima precisa ser verdadeira. A conclusão lógica é:
Se Chico é um gato e todo gato é felino, então Chico é um felino.

- Desvantagens do Paradigma Lógico
Variáveis não tipadas.

- Vantagens do Paradigma Lógico
Possui as vantagens do paradigma funcional e algumas vantagens do paradigma imperativo, tornando esse paradigma bem eficiente para o que ele se propõe, que é a prototipação de sistemas.

- Algumas linguagens do paradigma lógico:
Conniver, Planner, Mercury, Prolog, QLISP e etc.

VISITE A PÁGINA

Para saber mais sobre o paradigma lógico acesse o LINK

Paradigma Declarativo

O paradigma declarativo é baseado no lógico e funcional. Linguagens declarativas descrevem o quê elas fazem e não exatamente como suas instruções funcionam.

O melhor exemplo para o paradigma declarativo são as linguagens de marcação, como por exemplo o HTML, XML, XSLT, XAML e etc. Porém existe uma contradição nesse ponto, pois as linguagens de marcação não são consideradas linguagens de programação e por isso acaba fugindo um pouco ao escopo dos paradigmas de programação.

EXEMPLO

Veja um exemplo da linguagem HTML:

```
<HTML>
  <header>
    <h1>Linguagens e paradigmas de programação</h1>
  </header>
</HTML>
```

Como exemplo de linguagens que usam paradigma declarativo existe o Prolog, que apesar de ser nativo da linguagem lógica, utiliza também essa abordagem declarativa não sendo uma linguagem puramente desse tipo, mas representa o seu conceito. Assim como o Prolog, a Erlang e a Haskell que são linguagens nativamente funcionais também admitem abordagem declarativa, podendo ser consideradas como parte desse paradigma.

No que diz respeito às atribuições, as linguagens declarativas possuem variáveis com um único valor e esse valor não pode ser alterado durante a execução do programa. Essa é uma das características mais fortes das linguagens declarativas.

Paradigma Imperativo

Um dos paradigmas mais conhecidos e mais utilizados no desenvolvimento de software é o paradigma imperativo. Se você já ouviu falar de programação procedural ou programação modular, de modo geral, esse tipo de desenvolvimento é imperativo.

GUARDE ESSA IDEIA!

Paradigma imperativo é baseado na arquitetura de Von Neumann. É o primeiro paradigma a existir e até hoje é o mais utilizado.

Assim como na linguagem natural expressa comandos para fazer determinadas ações, o paradigma imperativo expressa uma sequência de comandos para que o computador possa executá-los da maneira que eles foram descritos.

A própria arquitetura dos computadores tem uma grande semelhança com paradigma imperativo. Talvez por isso esse paradigma continue tendo tanto sucesso até os dias de hoje. Essa semelhança se dá pela maneira como as instruções são passadas da memória para CPU e da CPU de volta para memória, uma após a outra de forma organizada e sequencial, isso te faz lembrar alguma coisa?

O paradigma imperativo é bem diferente dos outros paradigmas que falamos até agora, como, por exemplo, o paradigma declarativo ou o paradigma lógico e essa diferença se concentra basicamente no sequenciamento dos comandos para execução das instruções pelo computador, pois nos outros paradigmas não temos essa necessidade da criação de um sequenciamento para o desenvolvimento de programas.

Vamos ver um exemplo?

EXEMPLO

```
if(option == 'A') {
   print("Opção 'A' selecionada.");
}
```

A impressão só será realizada se o valor da variável option for igual a A.

As principais linguagens que usam o paradigma imperativo são as linguagens clássicas como C, C++, PHP, C#, entre outras.\

- Desvantagens do paradigma imperativo

As principais desvantagens do paradigma imperativo são a dificuldade de se ler o código, pela grande quantidade de instruções que ele possui, e todas essas instruções são focadas em como fazer e não no que fazer, o que acaba gerando também essa dificuldade na compreensão do código que foi escrito.

- Vantagens do paradigma imperativo

As principais vantagens estão relacionadas com a eficiência, que está baseada nos modelos matemáticos, pela sua flexibilidade e a modelagem de forma natural das aplicações realizadas no mundo real através de algoritmos.

Paradigma Orientado a Objetos

Entre todos, talvez esse seja o mais conhecido na atualidade, graças ao sucesso da linguagem Java que foi desenvolvida para trabalhar em cima desse paradigma. O paradigma orientado a objetos ganhou força e conhecimento através da comunidade acadêmica que passou a adotar o Java como principal linguagem de estudo. O paradigma orientado a objetos trouxe algumas funcionalidades a mais que deram uma maior flexibilidade e capacidade de resolver problemas que até então não eram possíveis através dos paradigmas imperativos.

Paradigma orientado a objetos caiu na graça da comunidade acadêmica, que por um tempo, pensou que esse seria o paradigma capaz de resolver todos os problemas existentes na programação de computadores. O que no final das contas acabou não sendo bem verdade, mas a sua capacidade na resolução de problemas é irrefutável.

Agora, vamos falar um pouco do que o paradigma orientado a objetos realmente faz. O paradigma orientado a objetos é baseado na composição e interação de diversos componentes de software denominados objetos. O funcionamento de um software orientado a objetos acontece através da troca de mensagens entre esses objetos, que executam as mais diversas funcionalidades dentro do código. Esses objetos são chamados de classe e o comportamento dessas classes é chamado de métodos; os estados dessas classes são chamados de atributos; e são nos métodos e nos atributos que também são definidas as formas de relacionamento que esses objetos irão interagir.

O paradigma orientado a objetos aproxima o nosso entendimento através da comparação com estruturas do mundo real, ou seja, objetos ou estruturas que nós conhecemos e temos uma grande compreensão a respeito. Dessa forma facilita o entendimento daquilo que estamos desenvolvendo. Ele também tem uma grande preocupação em esconder o que não é importante e dá um destaque para as coisas que são realmente importantes.

Atualmente esse paradigma tem sido bastante usado também nas aplicações comerciais e as principais linguagens que implementam paradigma de programação orientado a objetos são: C#, Java, PHP, Ruby, C++, Python e etc.

- Desvantagens do paradigma orientado a objetos

Por ter uma forma de pensar diferente, o paradigma orientado a objetos, acaba por vezes confundindo a cabeça dos programadores. O que leva a uma compreensão errada desse paradigma de programação. Essa pode ser a sua principal desvantagem, porém depois de aprender como o paradigma funciona tudo fica muito mais simples.

- Vantagens do paradigma orientado a objetos

O paradigma orientado a objetos possui as vantagens do paradigma imperativo e mais algumas, como a possibilidade de alterar o módulo sem alterar outros módulos. Quanto mais um módulo for independente maior a chance dele poder ser reutilizado em outra aplicação.

E por fim vamos ao nosso último paradigma.

Paradigma Orientado a Eventos

O paradigma orientado a eventos é um paradigma bem conhecido por todos os usuários de computadores domésticos, pois as aplicações que possuem uma interface gráfica são baseadas nesse paradigma. Chamamos de paradigma orientado a eventos porque o processamento das informações e dos comandos dentro do código do software são ativados a partir de eventos externos ou seja sempre que clicamos no botão ou preenchemos um campo estamos ativando eventos que irão definir o fluxo do código dentro da aplicação.

EXEMPLO

Um Bom exemplo está na imagem abaixo, veja só:

Fonte: https://www.treinaweb.com.br/blog/linguagens-e-paradigmas-de-programacao/

Conforme a imagem acima, o usuário irá decidir quando vai escrever o seu nome, quando vai apertar o botão de salvar e, a partir desses eventos, é que vão se definir os fluxos dentro do software. A princípio o simples fato de digitar dentro da caixa poderia gerar um evento de checagem, se ele está escrevendo

apenas letras, da mesma forma que o botão de salvar, pegaria as informações que estão na caixa "Seu nome:" e as levaria até o banco de dados para serem armazenadas, ou ainda poderia fazer a checagem se algo foi digitado na caixa acima e caso contrário retornar uma mensagem para o usuário pedindo que ele insira o seu nome.

Consegue perceber que todos os eventos podem alterar o fluxo da aplicação? Por isso dizemos que as linguagens que utilizam interface gráfica, são linguagens que usam o paradigma orientado a eventos.

O usuário então é responsável por definir quando os eventos acontecerão, por isso o fluxo do programa fica atrelado a ativação dos eventos pelo por ele.

VOCÊ SABIA?

Você sabia que os sistemas operacionais que utilizam interface gráfica de usuário utilizam paradigma orientado a eventos? Pois é, os sistemas operacionais com interface gráfica de usuário dependem das ações do usuário para executar as suas funcionalidades, logo eles são considerados sistemas que usam paradigma orientado a eventos, e eles usam não só na questão de comunicação com o usuário, mas também no tratamento do Hardware com a CPU, através do tratamento de interrupções tendo a CPU o papel de disparador desses eventos.

As principais linguagens de programação que utilizam o paradigma orientado a eventos são: Delphi, Visual Basic, Java, C#, Python e etc.

- Desvantagens do paradigma orientado a eventos

A principal desvantagem do paradigma orientado a eventos é a dependência das atividades do usuário, por isso a previsibilidade do que vai acontecer fica mais difícil, trazendo ao desenvolvedor uma responsabilidade maior no tratamento de erros que podem ser gerados pelas ações dos usuários.

- Vantagens do paradigma orientado a eventos

A sua principal vantagem também está ligada diretamente à interação com o usuário, já que no paradigma orientado a eventos nós conseguimos ter uma maior interação com o usuário.

GUARDE ESSA IDEIA!

Para concluir o nosso estudo a respeito dos paradigmas de programação é importante lembrar que os paradigmas não se restringem apenas a esses que vimos, existem ainda outros paradigmas que não estão listados aqui em no nosso caderno de estudo, porém o que é importante ter em mente é que nenhum desses paradigmas é a solução para todos os nossos problemas, pelo contrário cada paradigma vai trazer a solução para um problema específico.

Cabe a você escolher o melhor paradigma para resolução do seu problema. Tenha em mente que sempre que você se deparar com uma determinada situação você pode verificar se tem um paradigma que seja mais aderente para resolver aquela situação, assim como é importante a escolha do paradigma também é importante a escolha das linguagens. Como você viu muitas das linguagens que vimos aqui elas são multiparadigmas, ou seja, podem ser utilizadas nos diversos paradigmas mudando apenas a forma como programamos. Então não se desespere ao encontrar um problema que parece ser difícil de solucionar, apenas verifique qual o melhor Paradigma e a melhor linguagem e eu tenho certeza que você conseguirá resolver esse problema, inclusive juntando mais de um paradigma e fazendo uma aplicação multiparadigma.

PARA RESUMIR

Vimos neste guia de estudos da sala virtual do Tópico Integrador I, que existem vários paradigmas de programação e eles podem ser utilizados para as mais diversas soluções em termos de desenvolvimento de sistemas. Vimos também as especificidades de cada um deles, suas vantagens e desvantagens, e como podemos utilizá-los no nosso dia a dia de desenvolvimento. Trabalhamos os conceitos de linguagem e suas aplicações e vimos ainda o ranking de como está o uso dessas linguagens atualmente. Comentamos também sobre os seguintes paradigmas: funcional, lógico, declarativo, imperativo, orientado a objetos, orientado a eventos, escolhemos esses, pois são os principais paradigmas em uso na atualidade.

É importante lembrar que precisamos estar atentos às mudanças que acontecem no mundo da tecnologia. A qualquer momento novos paradigmas podem surgir e novas linguagens podem ser desenvolvidas e elas podem resolver um problema que tínhamos no passado, ou ainda implementar novas funcionalidades que vão facilitar o nosso dia a dia. Portanto não podemos ficar limitados ao estudo de apenas uma linguagem ou de um paradigma, precisamos abrir as nossas mentes e enxergar aquilo que é melhor para resolução dos problemas que temos no dia a dia.

PALAVRAS FINAIS

Caro(a) estudante, acredito que após a leitura do nosso guia de estudo, você está animado para continuar os seus estudos e entendeu a importância das ações desta sala virtual. Agora, vamos às próximas unidades para revisitar os conceitos aprendidos até este momento e vinculá-los a nossa prática profissional. Também será possível aprender novos conceitos e ver essa aplicação dentro do cotidiano do profissional de desenvolvimento de software.

Você deve ir ao Ambiente Virtual de Aprendizagem (AVA) e realizar as atividades referentes ao conteúdo aprendido nesta segunda unidade. Caso tenha alguma dúvida você deve entrar em contato com o tutor para que as esclareça.

Não se esqueça de resolver as atividades proposta referente ao conteúdo deste guia de estudo. A resolução das atividades irá consolidar o seu aprendizado e fará com que você esteja preparado(a) para as próximas etapas do seu curso.

Lembre-se, seu comprometimento nesta jornada de estudos é a chave para seu sucesso!

Bons estudos e até breve!

REFERÊNCIAS BIBLIOGRÁFICAS

1- Dershem, Herbert L.; Jipping, Michael J. (1995). Programming Languages. Structures and models (em inglês) 2ª ed. Boston: PWS Publishing Company. p. 1. 432 páginas. ISBN 0-534-94740-9

2- «TIOBE Index» (em inglês). www.tiobe.com. Consultado em 01 de setembro de 2018.