UNINOVE

# Servidores de Aplicação

APRESENTAR O CONCEITO DO SERVIDOR DE APLICAÇÕES, E OS PRINCIPAIS CONCEITOS RELACIONADOS À PLATAFORMA JAVA EE.

AUTOR(A): PROF. PAULO RICARDO BATISTA MESQUITA

Num primeiro momento, o nome *Servidor de Aplicação* pode parecer bem estranho, pois é um termo mais comum entre os desenvolvedores que usam a linguagem de programação Java, mas na verdade, ele não deixa de ser muito diferente dos servidores usados no mercado, e que costumam ser usados para os mais variados fins.

A função de um servidor é prover algum tipo de serviço para seus usuários, como por exemplo: armazenamento de dados, serviços de correio eletrônico (e-mail), compartilhamento de arquivos; apenas para citar os usos mais comuns. Além deles, também há os servidores web, que são usados para hospedar e executar aplicativos usados na Internet.

De uso menos comum, há os servidores usados para gerenciar comunicação entre usuários, servidores de multimídia, servidores de conteúdo, usados em aplicações como What App, Youtube, Instagram, respectivamente, ou em aplicações que usam todos esses recursos, como o Facebook, por exemplo, que permite a seus usuários compartilhar conteúdo escrito, multimídia e se comunicar através de serviços de troca de mensagens ou de transmissão de vídeos em tempo real.

Continuando a análise do Facebook, vemos que ele é um aplicativo que pode ser acessado por seus usuários através de browsers, ou de dispositivos móveis, como smartphones ou tablets. O mesmo pode ser dito do Youtube, ou de aplicações como o Google+. Para todas essas aplicações, podem ser identificados os seguintes requisitos:

- Necessidade de uso intensivo de uma rede de dados;
- Autenticação de usuários para o acesso ao conteúdo postado, e gerenciamento de seu conteúdo;
- Uso de um sistema eficiente para armazenar dados;
- Sincronização entre as informações exibidas para um determinado usuário, tanto pelos browsers quanto pelos dispositivos móveis. Ou seja, garantir que ele veja exatamente a mesma coisa nos dois casos;

Além desses, há requisitos mais específicos para que essas aplicações possam ser executadas, e que não convém detalhar neste momento. Mas diante do que foi falado anteriormente, acredito que é possível entender que essas aplicações usam vários dos servidores que foram mencionados no início do capítulo.


Legenda: FIGURA 1 - REDES SOCIAIS USADAS NA INTERNET E EM DISPOSITIVOS MóVEIS.

A função do desenvolvedor dessas aplicações é fazer com que o software seja capaz de se conectar e gerenciar aos vários tipos de servidores usados, para acessar os dados e o conteúdo armazenados pela aplicação, gerenciar e disponibilizá-los de acordo com o desejo do usuário. Mas essas características fazem com que uma aplicação como o Facebook, até mesmo pela sua complexidade, deixe de ser entendido como uma aplicação, e possa ser entendido como uma plataforma, que permite a interação entre vários usuários.

Sendo uma plataforma, ele possibilita que outras aplicações possam usar seus recursos, e há vários aplicativos on-line, desde jogos até aplicações sérias. E isso traz outros requisitos:

- Verificar se o aplicativo é uma fonte segura;
- Verificar quais os tipos de informações do usuário que o aplicativo pode acessar;

E em certos aspectos, esses requisitos ainda precisam permitir que dois aplicativos remotos se comuniquem, e que essa comunicação seja feita de forma segura entre as duas aplicações. E uma vez que essas aplicações são executadas em servidores, isso é o mesmo que dizer que é necessário que a comunicação entre os servidores seja segura.

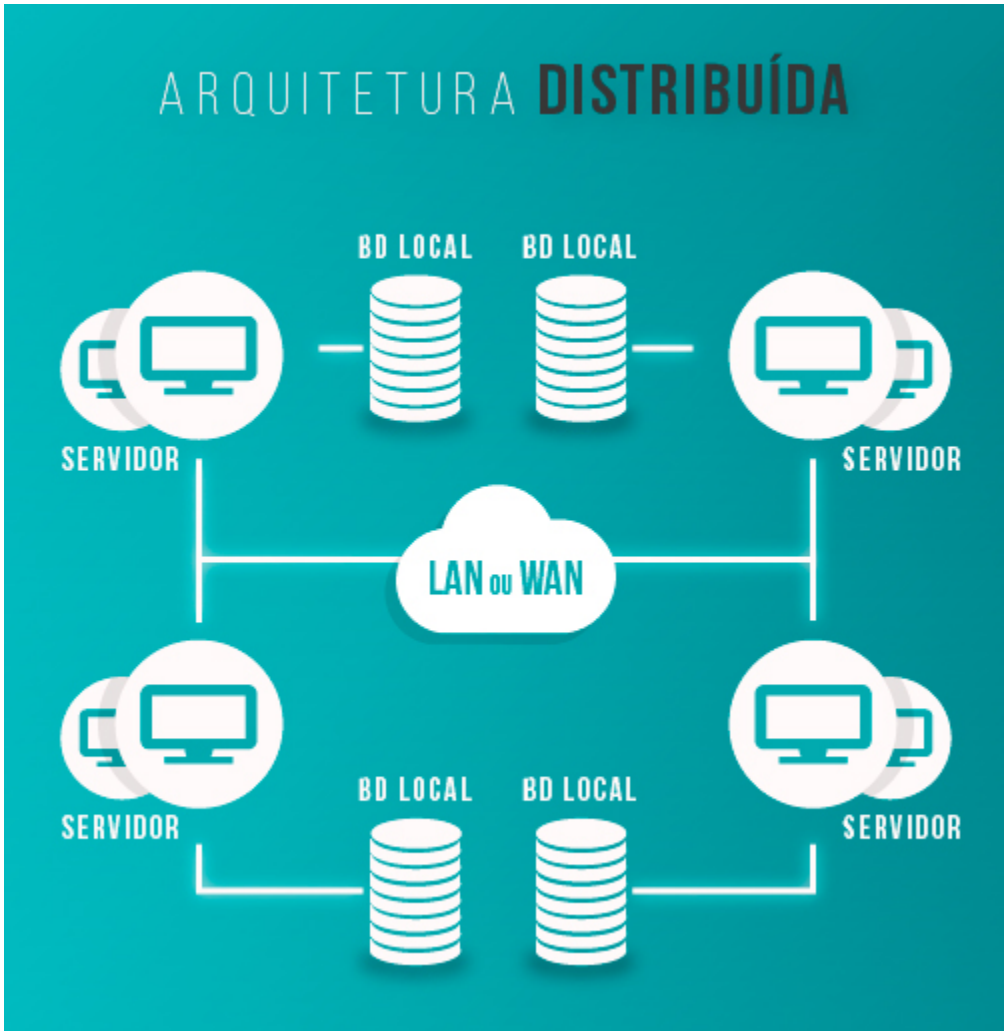
A manutenção dessa segurança é uma coisa que um desenvolvedor de software precisa considerar no desenvolvimento, da mesma forma que a forma de acesso ao conteúdo que está armazenado. Isso não faz parte do que o aplicativo precisa fazer, mas são itens que ele precisa ter para poder funcionar, são itens que normalmente são referenciados como sendo serviços de infraestrutura.

Para esses tipos de serviço, há muitas aplicações já prontas, e algumas delas estão disponíveis em várias plataformas de desenvolvimento, uma vez que os serviços de infraestrutura são baseados em soluções que evoluem mais lentamente, ou que evoluem até um ponto onde não é mais possível evoluir.

Uma das plataformas que permite fazer isso é o Java EE (Java Enterprise Edition), que possui um conjunto de frameworks, modelos de programação e APIs prontas, que executam vários dos serviços de infraestrutura que aplicativos como o Facebook necessitam.

Para ser usada, a plataforma Java EE precisa de um servidor de aplicações, que é a forma através do qual essa plataforma é disponibilizada a seus usuários. No caso, os desenvolvedores de software, não os usuários dos aplicativos finais.

Assim sendo, um servidor de aplicações pode ser definido como *"uma ferramenta para desenvolver e executar aplicações de grande porte"*. Entre as aplicações de grande porte, além do Facebook, também podem ser enquadrados os sistemas colaborativos, e os sistemas corporativos, como os sistemas usados na automação bancária, ou os sistemas de gestão de informações usados pelas grandes empresas, conhecidos como ERP.




Legenda: FIGURA 2 - ARQUITETURA DISTRIBUíDA, ARQUITETURA QUE EXEMPLIFICA COMO VáRIOS SERVIDORES, CONECTADOS A VáRIOS BASES DE DADOS, PODEM SER CONECTADOS PARA TROCA DE DADOS

Um desenvolvedor que usa a linguagem de programação Java, usa o Java EE para ter acesso aos serviços que um servidor de aplicações disponibiliza, e que resolvem problemas de infraestrutura, como:

- ter segurança no acesso às informações;
- garantir a disponibilidade dos serviços;
- balancear a carga de uso dos servidores;
- gerenciar as bases de dados usadas pela aplicação;

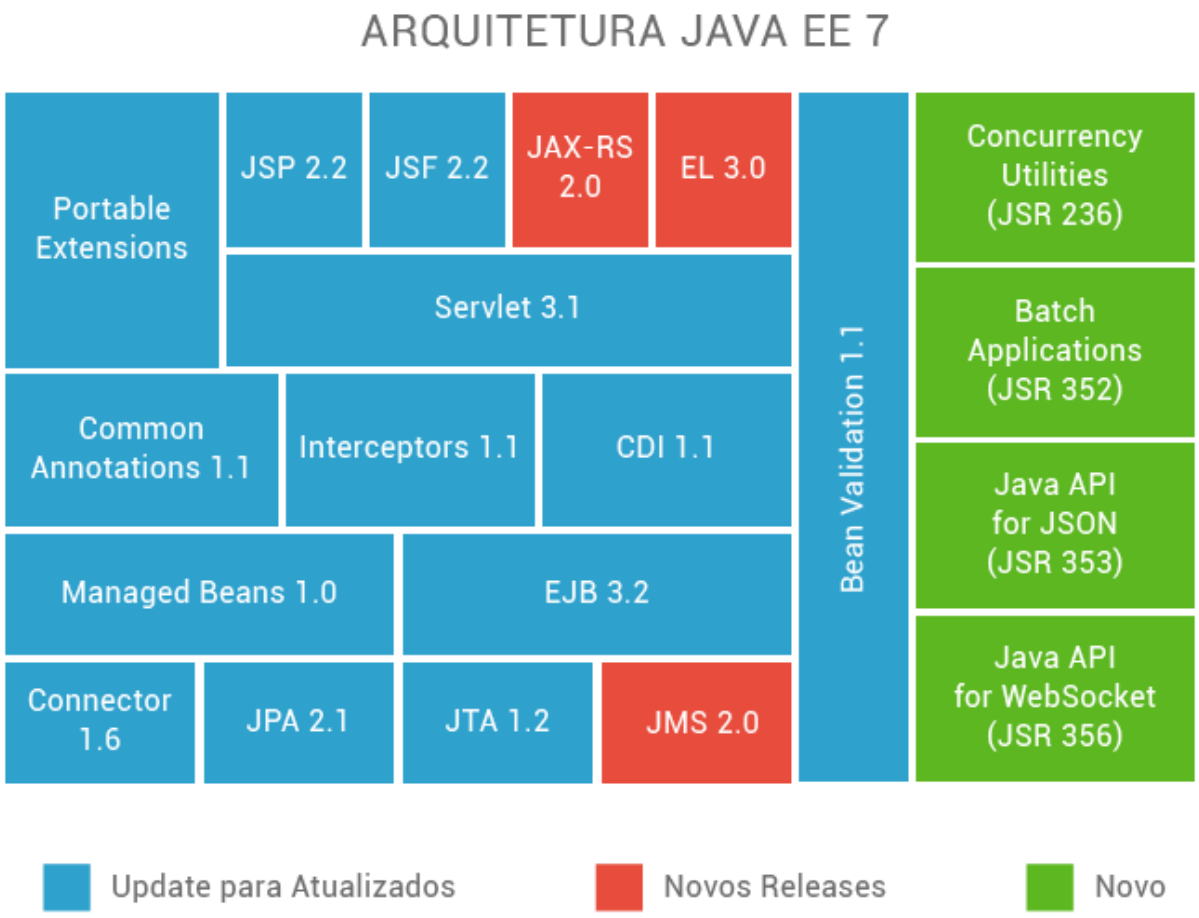
Isso permite que o desenvolvedor foque seus esforços e recursos para desenvolver as regras de negócio do aplicativo, uma vez que os serviços de infraestrutura já estão disponíveis no servidor de aplicações. Assim sendo, o desenvolvedor só precisa conectar seu aplicativo aos serviços disponibilizados pelo servidor de aplicações, obedecendo às seguintes regras:

- Seguir os padrões e especificações usados pela Java EE;
- Que ele possa prover serviços de infraestrutura que possam ser usados por outras aplicações;
- Que seu aplicativo possa gerenciar a comunicação com outros servidores de aplicações;
- Quando possível, disponibilizar frameworks para gerenciar/desenvolver serviços para outras aplicações;

Para os desenvolvedores Java, o servidor de aplicações é uma ferramenta, e as mais usadas no mercado são o JBoss e o WebSphere, que são instalados sobre os mais variados tipos de sistemas operacionais, como o Linux e Windows, entretanto, seu uso não se restringe aos desenvolvedores Java.

A plataforma .Net é similar à plataforma Java EE, e é bastante usada pelos desenvolvedores de aplicações Windows. Mas ao contrário do que ocorre com o Java, o servidor de aplicações é qualquer versão de servidor do Windows, e não uma ferramenta específica, como ocorre com o Java.

Há aspectos positivos e negativos no uso dessas duas plataformas, e no início dos anos 2000, havia uma disputa acirrada entre os usuários das duas plataformas sobre qual delas seria a melhor. Entretanto, o fato, é que nas aplicações usadas nas grandes empresas, muitas vezes, essas duas plataformas são usadas de forma complementar, e não concorrentes, como muitos imaginam. Isso ocorre porque muitas vezes, não são considerados somente os aspectos técnicos na seleção de uma determinada tecnologia.

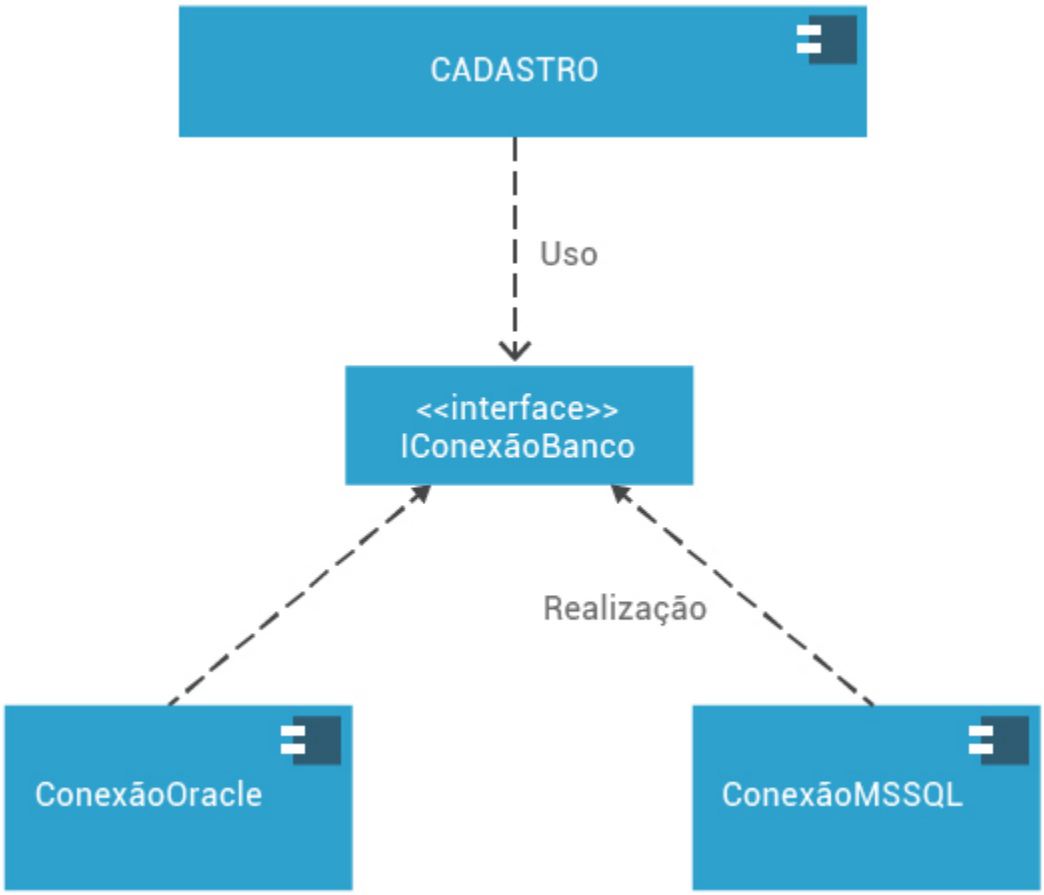



Legenda: ARQUITETURA JAVA EE 7

# Comparando Java EE com .Net

Para entender alguns dos aspectos técnicos associados ao uso dessas duas plataformas, segue uma rápida comparação entre os principais aspectos dessas duas plataformas:

- Os aplicativos criados em Java podem ser executados em qualquer sistema operacional, aproveitando o máximo do que a plataforma permite. Para .NET, apesar de iniciativas como Mono e DotGNU, runtimes para poder executar aplicativos .Net no Linux, para que o desenvolvedor aproveite todo potencial da plataforma, é necessário que o aplicativo desenvolvido seja executado no Windows;
- Para a escalabilidade dos aplicativos (capacidade para incrementar o potencial da aplicação simplesmente adicionando novos componentes de hardware ou de software), as duas plataformas possuem mecanismos eficazes;
- Quanto às linguagens de programação, .NET permite que os vários componentes da aplicação sejam escritos em VB.NET, C#, C/C++ ou J#. O uso de várias linguagens de programação diminui a facilidade no compartilhamento de informações e, pior, aumenta a complexidade na manutenção da aplicação, pois ser necessário contratar vários experts em cada uma dessas linguagens, aumentando o custo de desenvolvimento. Java EE usa apenas a linguagem Java.
- A curva de aprendizado das duas plataformas é similar, mas Java leva vantagem por haver uma quantidade maior de material disponível aos desenvolvedores.
- Por último, há a portabilidade das aplicações, e neste quesito, pouco pode ser comparado, pois graças à sua runtime, à padronização na forma de realizar a programação e à obediência das especificações de tecnologias e produtos associados ao Java, um aplicativo construindo com Java EE pode ser executado em qualquer sistema operacional, enquanto que o .NET somente pode ser totalmente aproveitado se executado sobre o sistema operacional da Microsoft, e do hardware que esse sistema está habilitado a usar.




Legenda: FIGURA 4 - EXEMPLO DE ARQUITETURA BASEADA EM COMPONENTES DE SOFTWARE, COM GERENCIAMENTO DE CONEXõES COM BASES DE DADOS.

# Tecnologias de Java EE

Tanto Java EE quanto .Net possuem um grande número de tecnologias para os desenvolvedores usarem, cada uma delas com sua própria especificação e modo de uso. Para os requisitos mais comuns ao desenvolvimento das aplicações de grande porte, as mais usadas são:

- Enterprise JavaBeans (EJB): é um modelo de programação para desenvolver componentes de software a serem desenvolvidos usando a linguagem Java. Esse modelo define como as classes Java devem se escritas para usar os outros componentes que compõem um servidor de aplicações, e para que outros aplicativos, dentro e fora desse servidor de aplicações, possam usar o componente de software escrito na forma de um EJB;
- API Java para Web Services (JAX-WS): permite desenvolver e disponibilizar web services na plataforma Java EE, que podem ser baseados em Servlets e em EJBs;
- Java Remote Method Invocation (RMI): é um conjunto de APIs que possibilitam que componentes de software escritos com a linguagem Java se comuniquem remotamente;

- Java Naming and Directory Interface (JNDI): é uma API que permite acessar sistemas de diretório e de nomenclatura, com o objetivo principal de acessar serviços que estão espalhados pela rede, sendo executados em equipamentos remotos ao servidor de aplicações;
- Java Database Connectivity (JDBC): API para acessar bancos de dados relacionais;
- Java Transaction API (JTA) e Java Transaction Service (JTS): permitem que os componentes de software possam realizar transações remotas, e acompanhar o resultado dessas transações;
- Java Messaging Service (JMS): permite a criação de componentes de software que executam transações baseada na comunicação através de mensagens;
- Java Web: conjunto de especificações Java que permitem o desenvolvimento de aplicações web dinâmicas, baseadas em páginas JSP e Servlets, e servem de base para muitos frameworks de desenvolvimento;
- Java Server Faces (JSF): framework para o desenvolvimento rápido de aplicações web em Java, baseadas em componentes de software, e no constante reuso de componentes de interface do lado do servidor;
- Java Persistence API (JPA): API que permite desenvolver componentes de software para gerenciar conexões com sistemas de base de dados relacionais, sem que o desenvolvedor precise escrever comandos todos os comandos SQL que um aplicativo precisa. A JPA usa um modelo de relacionamento baseado em classes de entidades, controladas por classes Java baseadas em beans de sessão, um tipo de EJB, e arquivos de configuração para localização física dos servidores e das bases de dados;

A plataforma Java EE possui várias outras APIs e frameworks para desenvolver componentes de segurança, manipulação de dados baseados em arquivos XML, JSON, interface com componentes específicos de um sistema operacional. Esse conjunto de APIs é continuamente evoluído de acordo com as necessidades da indústria de software, e muitos de seus componentes possuem equivalentes na plataforma Java EE.

Como a plataforma Java EE distribui esses componentes sob a forma de classes, ao usá-la, o desenvolvedor pode tornar sua aplicação bastante flexível, e usá-los numa variedade muito grande de aplicações. Além disso, ele pode usá-los para prover novos serviços, que podem ser usados para ampliar os usos das APIs da plataforma Java EE.

Na prática, dificilmente um aplicativo usa todos esses componentes, mas de modo geral, eles acabam usando pelo menos dois os três dos componentes dessa plataforma, acelerando o desenvolvimento do aplicativo. Como elas executam funções distintas, cabe ao desenvolvedor saber qual componente usar, diante do conjunto de requisitos que se apresentam a ele. Isso implica em dizer que no mercado, é muito difícil, senão praticamente impossível, encontrar um profissional que conheça em detalhes o que todas essas APIs fazem, e como elas são usadas.

Ao usar o Java EE, busca-se ter estas vantagens:

- Agilidade no desenvolvimento de software;
- Por usar componentes já bem testados, e muito reusados, garantir a qualidade mais efetiva em suas aplicações;

- Construir um aplicativo flexível, e que pode ser moldado nos vários modelos de arquitetura existentes no mercado;
- Desenvolver um software baseado em componentes, para serem reusados em outros aplicativos, ou disponibilizados sob a forma de novos serviços, expandindo as possibilidades da plataforma Java EE;

Além dessas vantagens, se o desenvolvedor não usar nenhuma função específica de um servidor de aplicações de um determinado fornecedor, e seguir os padrões e normas da plataforma Java EE, ele terá a garantia de que seu aplicativo será facilmente executado em qualquer servidor de aplicações, independentemente do fornecedor.

# Conclusão

Para os desenvolvedores de Java EE, um servidor de aplicações é um ambiente que permite desenvolver e executar aplicações de grande porte, baseadas em modelos de arquitetura distribuída. Os desenvolvedores o usam, porque ele possui a implementação dos serviços disponíveis na plataforma Java EE, e que podem ser usados para desenvolver os mais variados tipos de aplicações.

# ATIVIDADE FINAL

## Assinale a alternativa que NÃO POSSUI um requisito de software comum a aplicativos de grande porte.

A. Sincronização entre as informações exibidas para um determinado usuário, tanto pelos browsers quanto pelos dispositivos móveis. Ou seja, garantir que ele veja exatamente a mesma coisa nos dois casos;
B. Segurança no acesso às informações de usuários e ao conteúdo armazenado;
C. Gerenciamento no armazenamento de dados;
D. Redes de comunicação de dados;
E. Interface de usuário

## Assinale a alternativa que poderia ser um serviço disponível na plataforma Java EE, mas que não faz parte de Java EE. (Pode ser necessário pesquisa na WEB por alguns dos serviços apresentados na alternativas, antes de responder ao questionário).

A. Entity Frameworks

B. JPA

C. Servlets

D. JSP

E. JNDI

Para desenvolver um sistema que use totalmente os recursos de comunicação por rede de dados em qualquer sistema operacional, é recomendo usar qual plataforma de desenvolvimento?

A. Java EE

B. .NET

C. Ambas

Se o principal critério de desenvolvimento de uma aplicação de grande porte, baseado em componentes de software, for a velocidade de desenvolvimento, é recomendado usar qual plataforma?

A. Java EE

B. .NET

C. Ambas

Para economizar com a licença de ferramentas de desenvolvimento, o desenvolvedor deve selecionar a plataforma:

A. Java EE

B. .NET

C. Ambas

A plataforma que permite escrever uma aplicação que possa ser executada em qualquer sistema operacional, mas que apresenta limitações em alguns sistemas operacionais é:

A. Java EE

B. .NET

C. Ambas

Qual das alternativas não apresenta uma vantagem ao desenvolvedor de software, quando ele usa a plataforma Java EE.

A. Agilidade no desenvolvimento de software

B. Padronização no desenvolvimento

C. Reusar componentes de software

D. Desenvolver gerenciadores de conexão de banco de dados

E. Flexibilidade através do uso de componentes de software.

# REFERÊNCIA

Site do Java EE <http://www.oracle.com/technetwork/java/javaee/overview/index.html>, último acesso em 06/10/2016