No Wildfly, a principal responsabilidade do Domain Controller é a manutenção da política

de gerenciamento central do domínio.

(X ) Certo

( ) Errado

❑ WildFly

• O WildFly, anteriormente nominado JBoss, é um servidor de aplicações certificado Java EE 8;

• O termo "servidor de aplicações" foi criado em relação a aplicações Java Enterprise; pode-se pensar nisso como se fosse um software Java em que uma aplicação pode ser provisionada usando os serviços fornecidos pelo servidor de aplicações;

• O WildFly simplifica o desenvolvimento de aplicações corporativas fornecendo uma lista de serviços prontos para uso:

➢ Um pool de conexão JDBC para gerenciar o acesso ao banco de

dados relacional;

➢ Um agente do Messaging compatível com as especificações do

JMS denominadas "ArtemisMQ";

➢ Uma arquitetura compatível com o Adaptador de Recursos para

que você possa se conectar a sistemas externos (por exemplo,

Tibco, Servidor Weblogic, Active MQ, etc);

➢ Um contêiner EJB onde pode-se implantar serviços remotos;

➢ Um servidor Web leve e eficiente chamado "Undertow";

➢ Um agendador de tarefas em lote para executar tarefas e

trabalhos;

❑ Modo Standalone x Domain

• O Wildfly tem dois modos de gerenciamento: Standalone e Domain;

• Cada modo permite que você gerencie seus servidores de modo

diferente utilizando topologias diferentes;

❑ Modo Standalone

• O modo Standalone é o modo tradicional das versões anteriores;

• Implica em ter uma instalação diferente (ou um diretório standalone diferente) para cada instância de Wildfly;

• Ou seja, para cada Wildfly rodando no seu ambiente é necessário alterar seus próprios arquivo de configuração, suas próprias opções de execução para JVM, etc;

• Para iniciar o WildFly no modo domínio basta executar o arquivo $JBOSS\_HOME/bin/standalone.sh

❑ Modo Domain

• O modo Domain é o modo que foi introduzido no JBoss AS 7 onde é

possível gerenciar um conjunto de instâncias Wildfly, agrupando-os e assim permitindo compartilhar configurações comuns entre ele;

• Além de compartilhar configurações, é possível também através de um único console de gerenciamento iniciar ou parar instâncias (ou grupos inteiros), verificar seu status e estatísticas de cada subsystem, etc;

• Para iniciar o WildFly no modo domínio basta executar o arquivo $JBOSS\_HOME/bin/domain.sh

**COBOL**

• COBOL significa COmmon Business Oriented Language;

• É uma das primeiras linguagens de programação de alto nível;

• O COBOL foi desenvolvido em 1959 pela Conference on Data Systems Languages (CODASIL). Esse comitê era formado por um esforço conjunto da indústria, universidades e o governo

dos Estados Unidos;

❑ COBOL

• COBOL foi desenvolvido dentro de um período de seis meses, e ainda está em uso mais de 40 anos depois;

• Desde 1960, o American National Standards Institute (ANSI) foi responsável para o desenvolvimento de novos padrões COBOL;

❑ Objetivos da Linguagem

• COBOL é projetado principalmente para o uso em empresas e escritórios do governo;

• O COBOL é projetado para desenvolver aplicativos comerciais, geralmente orientados a arquivos;

❑ Características

• Auto documentando: Uma das metas de design do COBOL era possibilitar que os não-programadores, como supervisores, gerentes e usuários, lessem e entendessem o

código COBOL;

• Simplicidade:

➢ COBOL é uma linguagem simples, sem ponteiros, sem funções definidas pelo

usuário, sem tipos definidos pelo usuário e com um escopo limitado de

funcionalidade;

➢ Os aplicativos COBOL são executados regularmente em áreas críticas de

negócios;

➢ Um aplicativo COBOL lida com um grande número de dados regularmente.

Por exemplo, processamento de transações bancárias;

❑ Folha de Código

• Antigamente, o programa COBOL precisa codificado através de perfurações em cartões de papel perfurado;

• Para identificar a codificação do programa pelo leitor de cartões perfurados, houve necessidade de uma folha de codificação;

• A folha de codificação COBOL era uma estrutura para identificar de

onde as instruções de codificação COBOL puncionavam / começavam

a gravar para diferenciar umas declarações de outras;

❑ Folha de Código

• A estrutura de codificação COBOL é conhecida como folha de codificação COBOL;

• Hoje em dia, cartões perfurados são substituídos por editores, mas a mesma planilha de codificação é usada para codificar os programas COBOL;

• A área de codificação contém 80 colunas para cada linha;

❑ Regras de Codificação

• As primeiras seis posições de caracteres são reservadas para números de sequência;

• A posição do sétimo caractere é reservada para o caractere de continuação ou para um asterisco que denota uma linha de comentário;

• O código real do programa começa na coluna 8;

❑ Regras de Codificação

• As quatro posições de 8 a 11 são conhecidas como Área A e as posições de 12 a 72 são Área B;

• Todos os nomes de divisão, nomes de seção, nomes de parágrafo e números de nível 01 devem começar na Área A;

• Todas as outras sentenças devem começar na área B;

❑ Regras de Codificação

1++++678++1112+++++++++++++++++++++++++++++++7273++++++++80

• Colunas 1-6: Serão tratados como números de sequência;

➢ Colunas 1-3: Reservado para o número da página. Os números de página

serão aumentados automaticamente junto com as linhas de código

adicionadas ao programa;

➢ Colunas 4-6: Reservado para o número da linha. Os números de linha

serão aumentados automaticamente junto com as linhas de adição de

código;

❑ Regras de Codificação

• 7a coluna - Reservada para características especiais:

\* - Usado para comentar uma linha. Se o \* estiver codificado na 7a coluna do

programa, o compilador COBOL considerará como comentário em vez de código.

- - Usado para a continuação da linha anterior. Se - está codificado na 7a coluna

do programa, o compilador COBOL entenderá que a linha atual é a continuação

da linha anterior.

/ - Usado para parar a impressora;

D - Indicador de depuração. Se o D for codificado na 7a coluna do programa, o

compilador COBOL entenderá que a linha atual é usada para a finalidade

DEBUGGING;

❑ Regras de Codificação

• Colunas 8-72 - É o corpo do programa onde os programas COBOL podem ser

codificados;

Esse corpo COBOL foi dividido em duas áreas:

• Colunas 8-11: Todas as DIVISÕES, SEÇÕES, PARÁGRAFOS devem ser codificadas na

ÁREA A.

• Colunas 12-72: Todas as outras declarações, sentenças devem ser codificadas na

área B;

• Colunas 73-80: Número Gerado pelo Sistema usado para fins de identificação;

❑ Regras de Codificação

• Colunas 8-72 - É o corpo do programa onde os programas COBOL podem ser

codificados;

Esse corpo COBOL foi dividido em duas áreas:

• Colunas 8-11: Todas as DIVISÕES, SEÇÕES, PARÁGRAFOS devem ser codificadas na

ÁREA A.

• Colunas 12-72: Todas as outras declarações, sentenças devem ser codificadas na

área B;

• Colunas 73-80: Número Gerado pelo Sistema usado para fins de identificação;

**SPAs**

• Em um aplicativo Web tradicional, toda vez que o aplicativo chama o servidor, o

servidor renderiza uma nova página HTML. Isso aciona uma atualização de

página no navegador;

• Em um SPA, após o carregamento da primeira página, toda a interação com o

servidor ocorre por meio de chamadas AJAX;

• Essas chamadas AJAX retornam dados - não marcação - geralmente no formato

JSON. O aplicativo usa os dados JSON para atualizar a página dinamicamente,

sem recarregar a página;

❑ Ciclo de Vida

❑ Vantagens

• O cenário exemplificado é na verdade uma vantagem bastante interessante dos

SPAS, pois os torna fáceis de implantar na produção;

• Um SPA é simples de se implantar se comparado aos aplicativos renderizados no

servidor mais tradicionais: na verdade, é apenas um arquivo index.html, com um

pacote CSS e um pacote JavaScript;

• Esses arquivos estáticos podem ser carregados em qualquer servidor de

conteúdo estático, como Apache, Nginx, Amazon S3 ou Firebase Hosting pois

independe da tecnologia do servidor;

❑ Vantagens

• O aplicativo precisará fazer chamadas para o back-end para obter seus dados,

mas esse é um servidor separado que pode ser criado, se necessário, com uma

tecnologia completamente diferente: como Node, Java ou PHP;

• Na verdade, se construíssemos nossa API REST ou outro tipo de back-end no

Node poderíamos usar a mesma linguagem (JavaScript) no servidor e no cliente

e até compartilhar código entre eles;

❑ Vantagens

• Outro benefício dos SPAs é óbvio: os aplicativos são mais fluidos e responsivos,

sem o efeito incômodo de recarregar e renderizar novamente a página;

• Outro benefício pode ser menos óbvio e diz respeito à maneira como se

arquiteta um aplicativo Web. O envio dos dados do aplicativo como JSON cria

uma separação entre a apresentação (marcação HTML) e a lógica do aplicativo

(solicitações AJAX mais respostas JSON);

• Essa separação facilita o design e a evolução de cada camada. Em um SPA bem

arquitetado, pode-se alterar a marcação HTML sem tocar no código que

implementa a lógica do aplicativo (pelo menos, esse é o ideal);

❑ Vantagens

• Em uma SPA pura, toda a interação da interface do usuário ocorre no lado do

cliente, por meio de JavaScript e CSS;

• Após o carregamento inicial da página, o servidor age puramente como uma

camada de serviço;

• O cliente só precisa saber quais solicitações HTTP enviar. Não importa como o

servidor implementa as coisas no back-end;

❑ Vantagens

• Com essa arquitetura, o cliente e o serviço são independentes;

• Pode-se substituir todo o back-end que executa o serviço e, desde que não

altere a API, não quebrará o cliente;

• O inverso também é verdadeiro – pode-se substituir o aplicativo cliente inteiro

sem alterar a camada de serviço;

• Por exemplo, pode-se escrever um cliente móvel nativo que consome o mesmo

serviço;

5.1 - Ano: 2021 Banca: CESPE / CEBRASPE Órgão: CODEVASF Cargo: Analista em Desenvolvimento Regional - Tecnologia da Informação

Acerca de desenvolvimento de sistemas web, julgue os item a seguir.

Acerca de desenvolvimento de sistemas, julgue o próximo item.

O modelo SPA (single page application) permite a otimização do desempenho da aplicação ao

transferir a renderização para a máquina do cliente e diminuir o tráfego de dados entre

cliente e servidor.

( X ) Certo

( ) Errado