**JSON**

Podemos entender o **JSON como uma espécie de “concorrente” da XML na área de troca de informações**. Nesta seção, apresentamos algumas das principais semelhanças e diferenças entre os dois modelos para a representação de informações.

**Semelhanças:**

* Os dois modelos representam informações no formato texto.
* Ambos possuem natureza auto-descritiva (ou seja, basta “bater o olho” em um arquivo JSON ou em um arquivo XML para entender o seu significado).
* Ambos são capazes de representar informação complexa, difícil de representar no formato tabular. Alguns exemplos: objetos compostos (objetos dentro de objetos), relações de hierarquia, atributos multivalorados, arrays, dados ausentes, etc.
* Ambos podem ser utilizados para transportar informações em aplicações AJAX.
* Ambos podem ser considerados padrões para representação de dados. XML é um padrão W3C, enquanto JSON foi formalizado na RFC 4627.
* Ambos são independentes de linguagem. Dados representados em XML e JSON podem ser acessados por qualquer linguagem de programação, através de API’s específicas.

**Diferenças:**

* JSON não é uma linguagem de marcação. Não possui tag de abertura e muito menos de fechamento!
* JSON representa as informações de forma mais compacta.
* **JSON não permite a execução de instruções de processamento, algo possível em XML.**
* JSON é tipicamente destinado para a troca de informações, enquanto XML possui mais aplicações. Por exemplo: nos dias atuais existem bancos de dados inteiros armazenados em XML e estruturados em SGBD’s XML nativo.
* **JSON.stringify**  - recebe um objeto JavaScript e retorna uma string no formato JSON.
* **JSON.parse  -** recebe uma string no formato JSON e retorna um objeto JavaScript.

JSON está constituído em duas estruturas:

* Uma coleção de pares nome/valor. Em várias linguagens, isto é caracterizado como um object, record, struct, dicionário, hash table, keyed list, ou arrays associativas.
* Uma lista ordenada de valores. Na maioria das linguagens, isto é caracterizado como uma array, vetor, lista ou sequência.

Windows - Linux

Os perfis disponíveis para o controle de acesso no *Firewall*com segurança avançada do Windows Server 2016 são**: Domínio, Particular e Público.**

**Quando está habilitada a “Enumeração baseada em acesso”, em um compartilhamento de rede no Windows Server 2016, o** conteúdo do diretório compartilhado fica oculto para os usuários que não possuem acesso ao recurso.

O **renice** também altera a prioridade de execução dos programas, porém ele é para programas que já estão em execução, se não é claro para você, o comando nice é usado na hora de se iniciar um arquivo, dando assim à ele uma prioridade desejada.  
  
Um exemplo do renice segue abaixo:  
  
**# renice -14 kedit**  
  
Assim você estará dando uma nova prioridade de execução para o *kedit* (como já dito, temos a situação na qual o programa já está em execução).  
  
Vale lembrar que as prioridades vão de -20 a +19, onde quanto menor o número dado na prioridade, maior será a mesma aplicado ao programa.

Existem várias maneiras para enviar um comando para segundo plano, porém, a mais simples é colocando um & (E comercial) ao final do comando

find / -iname \*.mp3 >> musicas.txt &

Runlevels:

**Padrão LSB (Linux Standard Base):**

.: 0 ➝ Desliga;

**.: 1 ➝ Modo usuário simples;**

.: 2 ➝ Multiusuário sem rede;

.: 3 ➝ Multiusuário com rede;

.: 4 ➝ Reservado;

.: 5 ➝ Multiusuário com ambiente gráfico;

.: 6 ➝ Reinicializa

**Hexadecimal** literal values are written using X'***val***' or 0x***val*** notation, where ***val*** contains hexadecimal digits (0..9, A..F). Lettercase of the digits and of any leading X does not matter. A leading 0x is case-sensitive and cannot be written as 0X.

Campo Timestamp no Mysql:

CREATE TABLE t1(c1 TIMESTAMP DEFAULT CURRENT\_TIMESTAMP);

No Postgree, o comando que exibe o **plano** que o gerenciador irá utilizar para realizar uma determinada consulta é o **EXPLAIN** e pode ser utilizado junto ao **ANALYSE** para calcular o custo computacional. Válido também para MySQL. Tipos de Log do MySQL: **error log** e **general query log.** *Tipos de dados:* **BIGINT**= 8 bytes - **INT**= 4 bytes - **MEDIUMINT**= 3 bytes - **SMALLINT**= 2 bytes - **TINYINT**= 1 byte.

Existem dois tipos de banco de dados distribuídos, os **homogêneos** e os **heterogêneos**. Os homogêneos são compostos pelos **mesmos** bancos de dados, já os Heterogêneos são aqueles que são compostos **por mais de um tipo** de banco de dados.

Exemplo PHP:

<?php $x = “foo”; $$x = “bar”; ?>

Duas variáveis distintas serão inicializadas, a variável X com o valor foo e a variável foo com valor bar.

Definição de constante: define(“CONSTANTE”, 0);

Na linguagem HTML5, o atributo e seu valor, que devem ser utilizados para submeter um elemento ao corretor ortográfico, **é spellcheck=”true”.**

Na linguagem PHP, a atribuição das variáveis $a, $b e $c pode ser realizada simultaneamente com a sintaxe: **$a = $b = $c = 10;**

A declaração ***require*** é idêntica a ***include*** exceto que em caso de falha também produzirá um erro fatal de nível. Em outras palavras, ele parará o script enquanto que o include apenas emitirá um alerta () permitindo que o script continue.

Usabilidade

A usabilidade não é uma propriedade singular ou unidimensional de uma interface do sistema tecnológico de informação. A usabilidade está distribuída à diversos elementos, sendo tradicionalmente associada, segundo Jakob Nielsen, aos seguintes fatores:

**Facilidade de aprendizagem**: o sistema deve ser fácil de assimilar pelo utilizador, para que este possa começar a trabalhar rapidamente; **Eficiência**: o sistema deve ser eficiente para que o utilizador, depois de o saber usar, possa atingir uma boa produtividade; **Facilidade de memorização**: o sistema deve ser facilmente memorizado, para que depois de algum tempo sem o utilizar, o utilizador se recorde como usá-lo; **Segurança**: o sistema deve prever erros, evitar que os utilizadores os cometam e, se o cometerem, permitir fácil recuperação ao estado anterior. **Satisfação**: o sistema deve ser usado de uma forma agradável, para que os utilizadores fiquem satisfeitos com a sua utilização.

Pela definição da International Organization for Standardization, usabilidade é a medida pela qual um produto pode ser usado por usuários específicos ***para alcançar objetivos específicos com efetividade, eficiência e satisfação em um contexto de uso específico***(ISO 9241-11).

Os métodos de **Inspeção**, ou métodos analíticos ou de prognósticos, caracterizam-se pela **NÃO** participação direta do usuário do sistema na avaliação. Por ex: Percurso Cognitivo, Avaliação Heurística e Inspeção de padrões.

I - Avaliação Heurística é um método de avaliação que **NÃO** envolve usuários.

II - Inspeção é um método de avaliação que **NÃO** envolve usuários.

III - Estudo de Campo é um método de avaliação que **envolve** usuários.

Arquitetura de informação

Os arquitetos da informação concentram-se no design de sistemas de **organização, indexação, rotulagem e navegação** para dar suporte à navegação e pesquisa em todo o site.

Para a norma **ISO/IEC 9126**, **usabilidade** é um conjunto de atributos relacionados com o esforço necessário para o uso de um sistema interativo e com o julgamento individual de tal uso por determinado conjunto de usuários.  
  
A usabilidade não é uma propriedade singular ou unidirecional de uma interface do sistema tecnológico de informação. A usabilidade está distribuída à diversos elementos, sendo tradicionalmente associada, segundo Jakob Nielsen, aos seguintes fatores:

**Facilidade de aprendizagem:**o sistema deve ser fácil de assimilar pelo utilizador, para que este possa começar a trabalhar rapidamente;

**Eficiência:**o sistema deve ser eficiente para que o utilizador, depois de o saber usar, possa atingir uma boa produtividade;

**Facilidade de memorização:**o sistema deve ser facilmente memorizado, para que depois de algum tempo sem o utilizar, o utilizador se recorde como usá-lo;

**Segurança:**o sistema deve prever erros, evitar que os utilizadores os cometam e, se o cometerem, permitir fácil recuperação ao estado anterior;

**Satisfação:**o sistema deve ser usado de uma forma agradável, para que os utilizadores fiquem satisfeitos com a sua utilização.

Um dos objetivos da **engenharia cognitiva** é design centralizado no **usuário**, suporte a atividades cognitivas, e redesign de trabalho para administrar cognitive workload, semelhante aos princípios da ergonomia consoante IEA (international ergonomics association).

***Princípios básicos de usabilidade na Web:***

**Diálogos simples e naturais**: Os diálogos devem ser simples contendo apenas informações relevantes.

**Falar a linguagem do usuário**: A compatibilidade entre o vocabulário do sistema e do domínio deve sempre ser levada em consideração. Devemos nos perguntar se a linguagem da interface está simples e se as palavras, conceitos, e frases são familiares ao usuário.

**Minimizar a sobrecarga de memória do usuário**: Devemos sempre usar o reconhecimento em vez de lembrança. Os objetos, ações e opções devem estar sempre visíveis, as instruções para uso do sistema devem estar visíveis e facilmente encontráveis quando necessárias. Portanto, evitar o usuário de lembrar como encontrar ou fazer as tarefas é essencial.

**Ser consistente**: O sistema deve manter a consistência fornecendo os mesmos comandos, termos e ações para situações similares no sistema. Um exemplo é o atalho Control-C (CTRL+C) para copia. Os sistemas Windows normalmente utilizam esse mesmo atalho para funções de copia.

**Dar feedback**: Nessa heurística devemos nos perguntar se está sendo fornecido feedback em tempo razoável e se o usuário é mantido informado do que está ocorrendo. Nesse caso as barras de progresso são excelentes componentes visuais que ajudam o usuário, a saber, se um arquivo está sendo processado e quanto tempo ele irá levar para finalizar.

**Saídas claramente marcadas**: O usuário deve ter controle explícito sobre o sistema. Dessa forma, ele deve ter a possibilidade de abandonar qualquer estado indesejado ou situações incorretas ou ainda realizar undo/redo.

**Fornecer atalhos**: Os usuários mais experientes buscam sempre os atalhos para melhorar a sua produtividade. Nos sistema que estamos desenvolvendo não deve ser diferente, sempre devemos fornecer atalhos para esse tipo de usuário.

**Boas mensagens de erro**: As mensagens devem ser expressas em texto claro, sem qualquer tipo de código, indicando precisamente qual problema está ocorrendo. Também é interessante sugerirmos possibilidades de correção para o erro.

**Prevenir erros**: Devemos sempre nos perguntar se é fácil cometer erros no sistema, se sim verificamos onde e por que. Devemos prevenir ao máximo os erros causados pelo usuário.

**Ajuda e documentação**: O sistema sempre deve fornecer ajuda que é facilmente seguida. Dessa forma a ajuda e a documentação devem ser precisas e claras.

A cartilha de Usabilidade do governo brasileiro estabelece que:

Avaliação de usabilidade é dividida em: **Métodos de investigação**, **Métodos de inspeção** e **Teste com usuários**.

**Métodos de investigação**: Observação do Campo, Grupo de observação dirigida, Entrevista, Gravação de uso e Questionários.

**Métodos de Inspeção**: **Percurso cognitivo**,***Avaliação Heurística*** e ***Inspeção de padrões***.

**Teste com usuário**: Não há subtipos e são baseados em técnicas de etnografia.