VOLTAR

As 10 Heurísticas de Nielsen

Jakob Nielsen desenvolveu uma lista de dez heurísticas para auxílio no desenvolvimento de interfaces. Apresentaremos neste tópico, essas dez heurísticas e a importância delas na atualidade, mesmo com aplicações dinâmicas e cada vez mais robustas.

NESTE TÓPICO

O que é uma heurística?

Antes de começarmos você deve estar se perguntando: Mas que raios é uma heurística? Bem, essa é uma pergunta pertinente, afinal de contas não estamos acostumados com este termo.

Segundo nosso bom e velho amigo, o dicionário, heurística é:

> 1. Ciência ou arte que leva à invenção e descoberta dos fatos.
> 2. Método de ensino que consiste em que o educando chegue à verdade pelos próprios meios.
> 3. Ramo da ciência histórica que consiste na pesquisa de documentos e fontes do passado.
> 4. Procedimentos e normas usados em pesquisa feita por meio da quantificação de proximidade a um determinado objetivo
>
> (Michaelis online, disponível em http://michaelis.uol.com.br (http://michaelis.uol.com.br/busca?r=0&f=0&t=0&palavra=heur%C3%ADstica), acessado em 30 de Setembro de 2016)

Veja cuidadosamente a definição de número quatro, que está diretamente relacionada a nossa área (informática): **um conjunto de procedimentos ou normas.**

Isso quer dizer que as famosas 10 heurísticas de Nielsen são o mesmo que “as 10 normas de Nielsen”. Devemos tratá-las, então, com muito respeito.

Mas quem é Nielsen mesmo? Bem Nielsen é apenas um dos autores mais consagrados na área de design de interfaces, não apenas por ser um dos primeiros mas, também, por ser um dos mais atuais e mais respeitados.

Imagem ilustrativa: Lista heurística

As 10 heurísticas de Nielsen

Sabendo o que significa heurística, vamos agora conhecer as famosas 10 heurísticas de Nielsen.

1. Visibilidade do Status do Sistema

O usuário precisa saber o que está acontecendo na aplicação o tempo inteiro, para não se sentir perdido, ou seja, todas as ações desempenhadas pelos usuários precisam de feedback instantâneo.

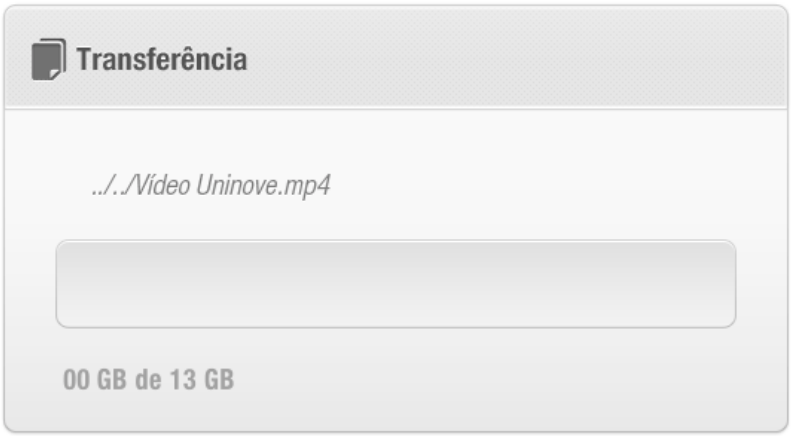
Isso quer dizer que mesmo que o software esteja processando algo, é preciso manter o usuário informado sobre o processamento. É por isso que quando copiamos um arquivo grande de uma pasta para outra é exibida uma tela com o progresso e alguma animação, mesmo que essa animação seja meramente ilustrativa.

Muito sistemas implementam barras de status, normalmente na parte inferior da interface. Essas barras também são uma boa opção pois podem mostrar informações relevantes ao usuário sobre o que está acontecendo.

Imagine, por exemplo, dirigir um carro sem velocímetro. Seria um perigo, não? Bem, além de velocímetro, os carros hoje possuem vários instrumentos e computadores de bordo que mostram o que está acontecendo exatamente

com o carro, em tempo real.

Resumindo: Essa heurística é alcançada com barras de progresso, barras de status, ferramentas que de exibição em tempo real de informações (gráficos, barras de leds, contadores, instrumentos de medição etc.), entre outros.



Exemplo de visibilidade e status do sistema: Barra de progresso é um tipo de barra de status

2. Relacionamento entre a interface do sistema e o mundo real

Toda a comunicação do sistema precisa ser contextualizada ao usuário, e ser coerente com o chamado modelo mental do usuário. Lembra-se do princípio de usabilidade de Nielsen sobre bons modelos conceituais? Pois bem, é exatamente essa heurística que cobre este princípio

Os usuários possuem em mente uma forma de resolver o problema deles, pois eles conhecem o domínio de seu trabalho. As interfaces precisam permitir que o usuário implemente seu processo de trabalho, ou seja, o software deve ser uma ferramenta facilitadora e não ditadora de como o usuário deve fazer seu trabalho.

3. Liberdade e controle do usuário

Permitir que o usuário desfaça e refaça suas ações é uma forma de dar liberdade e controle a ele, pois errar é humano, certo?

É importante deixar o usuário no controle o tempo todo, mesmo quando o sistema possui opções de "faça por mim", mas é importante não se esquecer de que o usuário não pode ficar pensando o tempo inteiro. O trabalho que exige pensamento profundo precisa ser feito pela máquina.

Ícone de

4. Consistência

Não podemos identificar opções e ações similares com palavras ou termos diferentes, ou seja, precisamos tratar as coisas que são similares no sistema com os mesmos termos, para facilitar que a ação possa ser rapidamente identificada pelo usuário.

Precisamos manter a consistência na aplicação inteira, com informações precisas, corretas e que o usuário possa confiar.

Por exemplo, se temos a opção de imprimir no menu e também um botão próximo ao relatório sendo exibido, a palavra imprimir deve ser clara, assim como a tecla de atalho deve ser consistente, ou seja, deve envolver teclas que tenham relação com a impressão.

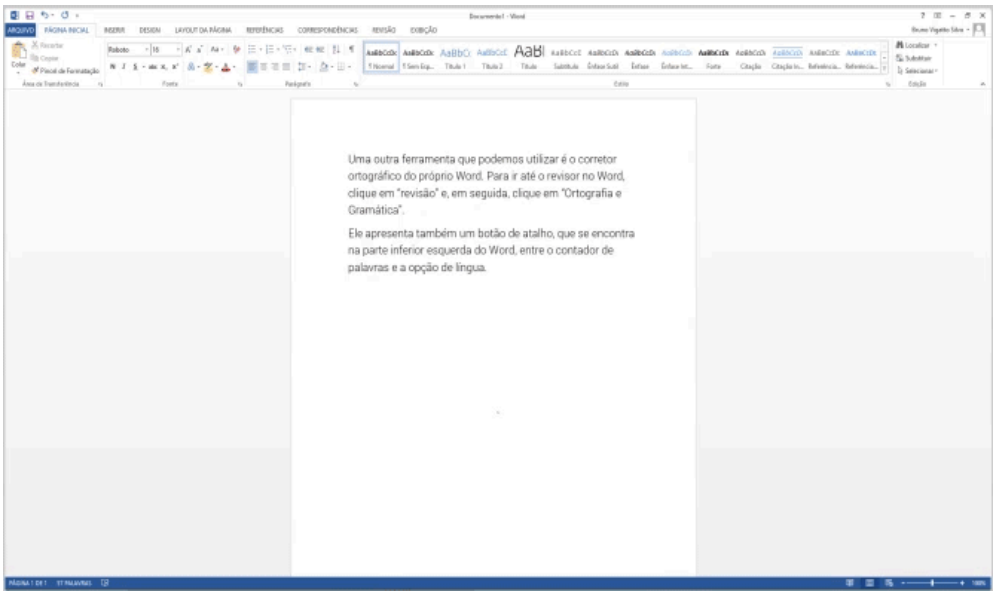
Ícones diferentes que significam a mesma coisa

5. Prevenção de erros

É sempre melhor prevenir do que remediar, não? Bem, Nielsen ainda completa: “Ainda melhor que uma boa mensagem de erro é um design cuidadoso que possa prevenir esses erros” (Nielsen).

Precisamos criar interfaces amigáveis que façam com que o usuário não erre ou, ao menos, minimize os erros dos usuários.

Alguns exemplos: Corretores automáticos de ortografia (como aqueles de nossos celulares que sempre colocam palavras erradas no lugar e nos constrangem as vezes, mas funcionam muito bem na maior parte do tempo!), interfaces inteligentes que percebem o erro do usuário, validadores de formulários (que não permitem o usuário digitar coisas que não podem, como texto no lugar de data, por exemplo), entre outros recursos.



Corretor ortográfico

6. Reconhecimento ao invés de lembrança

Precisamos evitar acionar a memória do usuário o tempo inteiro. Pensar cansa! Lembre do princípio de usabilidade de Steve Krug: "Não me faça pensar".

O sistema deve pensar pelo usuário e apenas acionar sua memória quando for necessário. A ideia é que os sistemas orientem as ações dos usuários, minimizando o acionamento da memória de longa duração (lembra do MPIH?) para evitar uma fadiga mental.

Normalmente conseguimos isso com o uso de ícones e cores corretas, palavras certas, coisas certas no lugar certo e assim por diante.

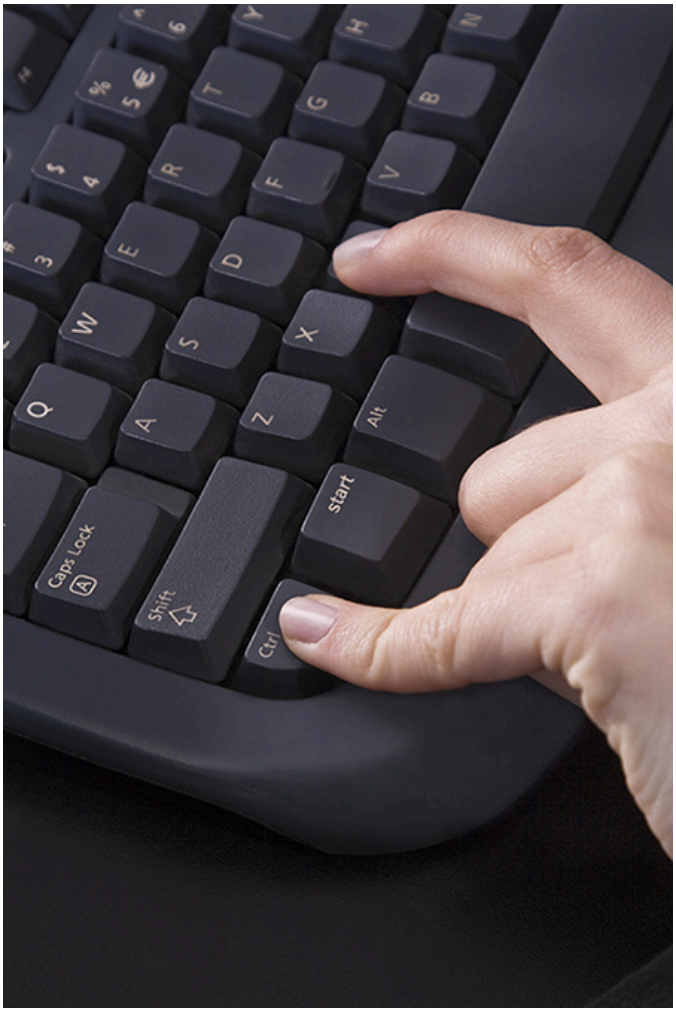
Acho que estou perdido

7. Flexibilidade e eficiência de uso

O sistema precisa ser fácil de usar, tanto para quem é iniciante quanto para usuários avançados, ou seja, deve-se permitir um uso do básico ao avançado.

Normalmente, para se conseguir isso, acrescenta-se menus ocultos que podem ser exibidos por usuários avançados, acréscimo de diversas teclas de atalhos etc.

Por exemplo, usuário novatos (realmente novatos no mundo da informática) raramente usam teclas como o "Ctrl + C" e "Ctrl + V" (Copiar e colar, respectivamente), mas ao longo do tempo aprendem as teclas e passam a usar naturalmente, assim como as demais, como atalhos de selecionar tudo (que as vezes é diferente para cada aplicativo), recortar, imprimir, procurar etc.



Ctrl + C

8. Estética e design minimalista

Lembra-se de um dos princípios de design sobre "menos é mais"? Bem é exatamente isso que essa heurística quer dizer. Menos é mais sempre, ou seja, torne visível apenas o que é preciso estar visível em cada parte de sua interface e que realmente faça parte do contexto.

É importante que a interface não "diga" mais do que o usuário precisa saber. Todas as caixas de diálogos, menus, textos etc., precisam sem o mais simples, naturais e diretor o possível.

Steve Krug acrescenta sobre este aspecto que devemos "omitir o que é desnecessário", ou seja, se não precisa estar lá ou se é opcional, não deve aparecer. Isso vale não apenas para componentes de interface, mas também para textos informativos. Os usuários precisam de agilidade e facilidade e textos longos muitas vezes atrapalha mais do que ajuda.

Há, ainda, uma antiga publicação, de 1959 (isso mesmo, antes mesmo dos computadores existirem), nomeada de "os elementos do estilo", por E. B. White, onde uma das 18 regras do estilo é justamente essa: "Omita palavras desnecessárias".

Menos é mais!

9. Ajude os usuários a reconhecer, diagnosticar e sanar erros

A heurística número 5 fala para criarmos interfaces que evitem erros, mas eles podem acontecer, eventualmente.

As mensagens de erros precisam ser claras e simples e não podem intimidar o usuário com códigos ou termos técnicos. Elas precisam ajudar o usuário a resolver o problema e não deixá-lo ainda mais perdido. Por isso temos que tratar os erros com um carinho especial, pois eles podem acontecer e

precisam ser resolvidos. É importante, também, evitarmos códigos de erros que não dizem nada para os usuários, como por exemplo 404 (de página não encontrada). Isso não quer dizer nada para a maior parte dos usuários!

Muitos sistemas já reconhecem os erros e promovem recursos para que o erro não ocorra mais, por exemplo se o usuário clica em um botão para submeter um formulário sem ter digitado algum item obrigatório, o sistema não pode retornar um "NullPointerException", mas sim uma informação construtiva como "olha, parece-me que faltou a informação XYZ no formulário, por favor, reveja-o e tente submeter novamente".

Erros não podem ser intimidadores

10. Ajuda e documentação (ajuda não ajuda)

Se suas interfaces são suficientemente claras, os usuários jamais precisarão usar o material de ajuda (help) de seu sistema, mas eventualmente podem precisar. É comum as pessoas lerem manuais de carros, de microondas etc., mas é raro consultarem manuais de aplicações. Mas algumas leem!

Bem, independente disso, o material de ajuda precisa ser claro e construtivista, ou seja, deve ser facilitador e não piorar a situação do usuário ou intimidá-lo.

Quantas vezes não apertamos "sem querer" a tecla padrão de ajuda (F1) e vemos um texto enorme sendo exibido na tela. Pois bem, isso é passado. Materiais de ajuda precisam ser simples e fáceis de consultar. A ajudar tem

que ajudar!

É muito importante que todo o sistema esteja documentado e dentro do material de ajuda e não apenas as funções principais. Separar um tempo para cuidar deste material é primordial em qualquer projeto.

Ah! O material de ajuda precisa estar sempre atualizado, pois muitas vezes os sistemas sofrem atualizações e o material de ajuda não acompanha.

Ajuda tem que ajudar

Você foi apresentado(a) aqui às 10 famosas heurísticas de Nielsen. É muito importante tê-las em mente durante o processo de desenvolvimento de suas interfaces, escolhas das cores, escolhas das mensagens, escolhas dos posicionamentos etc. O importante é não esquecermos nunca que nosso papel, como designer, é facilitar a vida dos usuários e não dificultar. Não se esqueça nunca dessa lista:

1. Visibilidade de Status do Sistema
2. Relacionamento entre a interface do sistema e o mundo real
3. Liberdade e controle do usuário
4. Consistência
5. Prevenção de erros
6. Reconhecimento ao invés de lembrança
7. Flexibilidade e eficiência de uso
8. Estética e design minimalista (menos é mais)
9. Ajude os usuários a reconhecer, diagnosticar e sanar erros

10. Ajuda e documentação (ajuda não ajuda)

Não deixe de ler e reler as heurísticas e o que cada uma significa. Já comece a pensar em sistemas que você usa hoje em dia se eles atendem ou não as heurísticas listadas aqui.

Quiz

Exercício Final

As 10 Heurísticas de Nielsen

INICIAR

Referências

Rocha, H. V. e Baranauskas, M. C. C., 2003, Design e Avaliação de Interfaces Humano-Computador, Instituto de Computação, Universidade Estadual de Campinas

Krug, S., 2008, Não me faça pensar - Uma abordagem de bom senso à usabilidade na Web, 2ª Ed., Alta Books

Ferreira, S. M., Nunes, R. R, 2008, e-Usabilidade, 1ª ed., Editora LTC

Nielsen, J. (1993) Usability Engineering. Academic Press, Cambridge, MA

Norman, D. A. (1988) The Psychology of Everyday Things. Basic Books, New York

Michaelis Online, disponível em http://michaelis.uol.com.br/ (http://michaelis.uol.com.br/), acessado em 15 de setembro de 2016

Avalie este tópico

ANTERIOR
A Interação Humano-Computador

Índice

PRÓXIMO
Framework PACT para sistemas interativos