Técnicas e heurísticas para *design* de *interfaces*

Design de *interfaces*

- A criação de *interfaces* para *softwares* ganha mais importância após a grande expansão do acesso à *internet*, aos *smartphones* e computadores.
- Os *softwares*, antes utilizados apenas por um pequeno grupo de trabalhadores da academia ou de grandes empresas, agora fazem parte da vida da quase totalidade da população dos países mais desenvolvidos.

Design de *interfaces*

- Desenvolver *interfaces* para amplos grupos de pessoas com diferentes culturas, experiências e graus técnicos no uso de computadores torna o desenvolvimento de *software* mais difícil.
- Pode-se mitigar este problema através do uso de técnicas e heurísticas já largamente utilizadas na indústria.

Heurística

Definição

(Heurística) - uma regra que pode ajudar a resolver um dado problema tomando como base o conhecimento sobre a natureza do problema.

Heurísticas de Nielsen

- Em 1990, dois pesquisadores, Jakob Nielsen e Rolf Molich propuseram 10 heurísticas para auxiliar a criação em *interfaces*.
- As heurísticas de Nielsen não são regras extritas, mas parâmetros a serem observados para construção e avaliação de *interfaces*.

Heurísticas de Nielsen

São 10 as heurísticas de Nielsen:

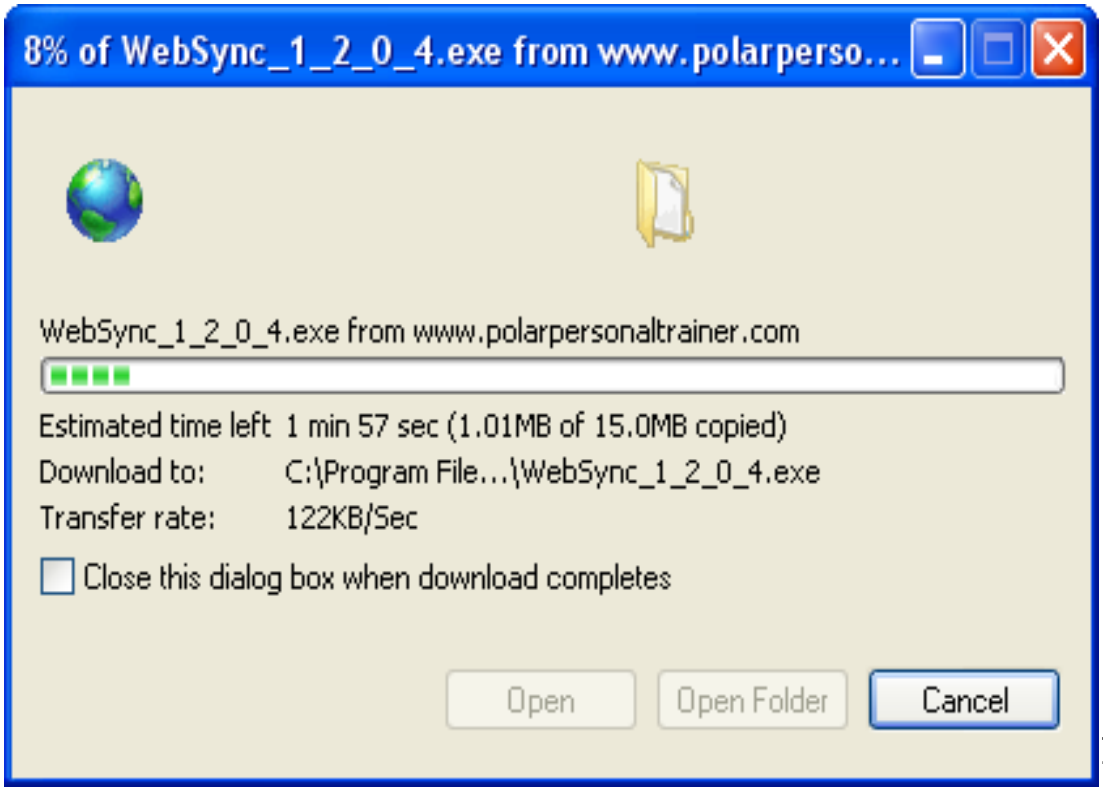
1. visibilidade do estado do sistema;
2. compatibilidade entre o sistema e o mundo real;
3. controle e liberdade para o usuário;
4. consistência e padronização;
5. prevenção de erros;
6. reconhecimento em vez de memorização;
7. eficiência e flexibilidade de uso;
8. estética e design minimalista;
9. ajude os usuários a reconhecerem, diagnosticarem e recuperarem-se de erros;
10. ajuda e documentação.

Visibilidade do estado do sistema

- É quase mandatório que o usuário esteja sempre a par do estado do sistema.
- Isso pode ser garantido por elementos de interface pequenos, mas que auxiliam bastante o usuário.
- São exemplos de visibilidade do estado do sistema:
  - a mudança do símbolo de *play* para o *pause* em aplicativos de músicas;
  - tempo estimado para o fim de um *download*;
  - barra indicando quanto do espaço de armazenamento foi utilizado;
  - a seleção da faixa que está tocando.

Visibilidade do estado do sistema

Barra de progresso para indicar o quanto de um *download* ou instalação de um *software* foi feito até então.



[1]

[1] Fonte: Flickr/Doug Beckers. Licença: CC BY-SA 2.0

Visibilidade do estado do sistema

Ampulheta ou roda para indicar que uma ação está sendo realizada.

2

[2] Fonte: FreeSVG. Domínio Público

Compatibilidade entre o sistema e o mundo real

- O sistema deve fornecer uma linguagem familiar ao contexto do usuário.
- Por exemplo, se um sistema é voltado para crianças, então a linguagem deve ser compatível com esse público.
- Deve-se utilizar ícones que possibilitem ao usuário ter uma noção intuitiva do que um botão faz.

Controle e liberdade para o usuário

- Trata da capacidade do usuário poder desfazer uma ação ou não ser forçado a seguir um único caminho.
- Três exemplos clássicos desta heurística:
  - a lixeira presente na maioria dos SOs modernos;
  - a lixeira presente nos sistemas de e-mails;
  - a função desfazer (Ctrl+z).

Controle e liberdade para o usuário

- Outro exemplo é poder explorar um *site* de *e-commerce* sem obrigatoriamente ter de fazer *login*.
- O *site* da Amazon e um bom exemplo de liberdade para o usuário, pois permite que o usuário remova itens do carrinho durante o *checkout*.

Controle e liberdade para o usuário

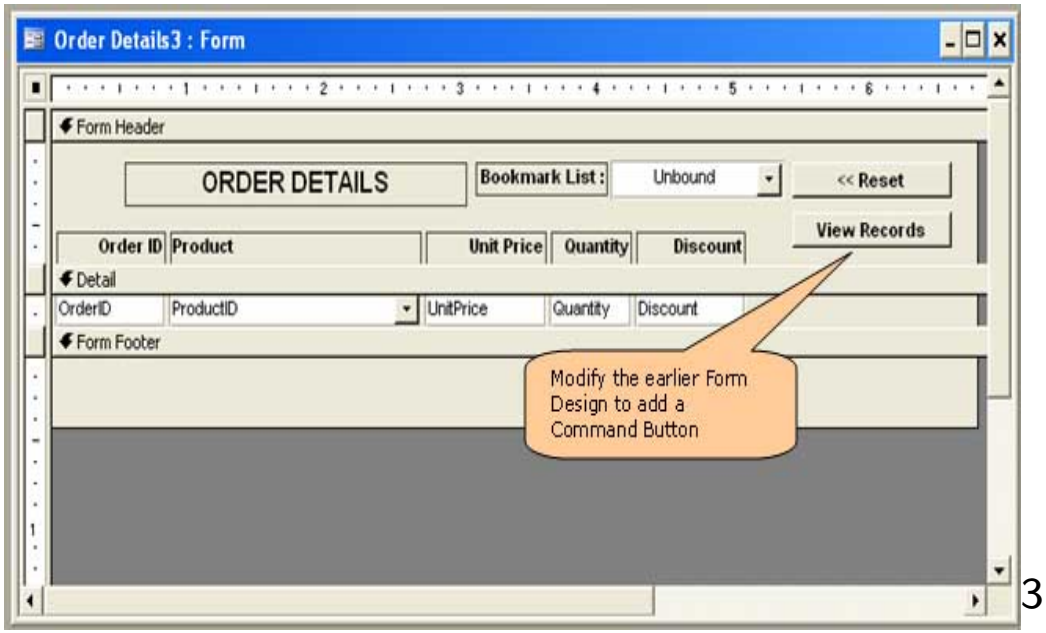
- Em alguns casos, será interessante restringir a liberdade do usuário.
- Por exemplo, adicionar um *wizard* para instalação e configuração de impressora.

Consistência e padronização

- O sistema deve possuir uma padronização léxica e gráfica.
- **Apenas um ícone ou um termo** devem indicar uma funcionalidade.
- Um ícone ou um termo devem indicar **apenas uma funcionalidade**.
- A disposição e o alinhamento de ícones, botões e textos também deve ser consistente.

Prevenção de erros

- A *interface* deve ser projetada para que o usuário cometa o menor número de erros possíveis.
- São exemplos os sistemas que sugerem uma busca com ortografia parecida ou de sinônimos da que foi buscada e a inserção de máscaras, dicas marginais e caixas de confirmação buscam garantir esta característica.



[3]

[3] Fonte: msaccesstips.com. Licença: CC BY-NC-ND 2.5

Reconhecimento ao invés de memorização

- Objetos, ações e opções devem ser visíveis ao usuário.
- O usuário não deve ter de lembrar da informação de uma parte do sistema em outra. Este deve estar sempre visível quando necessária.
- Sistemas que dependem de códigos são um mau exemplo.

Flexibilidade e eficiência de uso

- A *interface* além de ser bastante visual para os usuários mais inexperientes, também deve possuir "aceleradores" para os usuários mais avançados.
- São exemplos desses "aceleradores":
  - teclas de atalho;
  - boas opções padrão em formulários;
  - sistemas de recomendação.

Estética e *design* minimalistas

- Tanto a distribuição das cores quanto das informações não deve atrapalhar o usuário.
- Não se deve inserir funcionalidades ou elementos de *interface* que não serão utilizados.

Recuperação de erros

- As mensagens de erro devem ser claras e de fácil entendimento para o usuário.

Ajuda e documentação

- É importante que haja uma seção dedicada à ajuda e à documentação.
- Muitas vezes o usuário necessitará buscar mais detalhes sobre uma determinada funcionalidade, portanto, é importante que as seções de ajuda e documentação sejam bem arquitetadas e com linguagem simples e direta.

Referências

- Dix, Alan *et al.*. Human-Computer Interaction. 3ed. Pearson Education. 2004.
- Editora Alea. Nielsen's Heuristics: 10 Usability Principles to Improve UI Design. https://aelaschool.com/en/interactiondesign/10-usability-heuristics-ui-design/. 2022.