N_FAT HUM SIST COMP_A3 - Texto de apoio

Site: EAD Mackenzie

Tema: FATORES HUMANOS EM SISTEMAS COMPUTACIONAIS {TURMA 04A} 2023/2

Livro: N_FAT HUM SIST COMP_A3 - Texto de apoio

Impresso por: FELIPE BALDIM GUERRA .

Data: terça, 17 out 2023, 00:02

Descrição

Índice

1. PRINCÍPIOS DE DESIGN DE INTERFACES E PROTOTIPAGEM

Princípios de Design de Interfaces

Alguns defensores do design centrado no usuário apresentaram, ao longo dos anos, **regras práticas que os designers de interfaces** podem e devem utilizar como **diretrizes para orientar a concepção de suas interfaces**, de modo a obter **máxima usabilidade**.

É muito importante ressaltar que, embora o uso de princípios de design de interfaces auxilia no processo de criação de interfaces com alto grau de usabilidade, eles jamais substituem o processo cuidadoso referente às atividades de análise de usuários e tarefas, que você viu na Aula 2.

As **regras práticas** mais utilizadas, segundo (GONÇALVES et al., 2017) são:

- **Princípios de Design de Norman.**
- **Regras de Ouro de Shneiderman.**

Heurísticas de Nielsen.

1.1. Princípios de Design de Norman

A utilização de objetos do nosso dia a dia – portas, torneiras, escadas etc. – deveria ser o mais simples possível. No entanto, várias pessoas continuam a ter problemas de interação com estes objetos, como empurrar portas que deveriam ser puxadas.

Donald Norman, grande pesquisador da área de IHC, identificou um **conjunto de princípios de design** para garantir uma **boa usabilidade** (GONÇALVES et al., 2017):

1. **Visibilidade.**
2. **Feedback (retorno).**
3. **Restrições.**
4. **Coerência.**
5. **Mapeamento.**
6. **Evidência (ou *affordance*).**

1. Visibilidade

A visibilidade está relacionada com aquilo que se consegue ver em determinado passo da interação. As interfaces devem mostrar seu estado e as possíveis ações que os usuários podem realizar, ou seja, **procure garantir que as coisas sejam visíveis**, de forma que os **usuários possam ver quais funções estão disponíveis** e o que **o sistema está fazendo atualmente**.

Nos sistemas complexos que têm muitas funcionalidades, em vez de todas estarem visíveis ao mesmo tempo, uma sugestão é usar menus para as funções menos utilizadas.

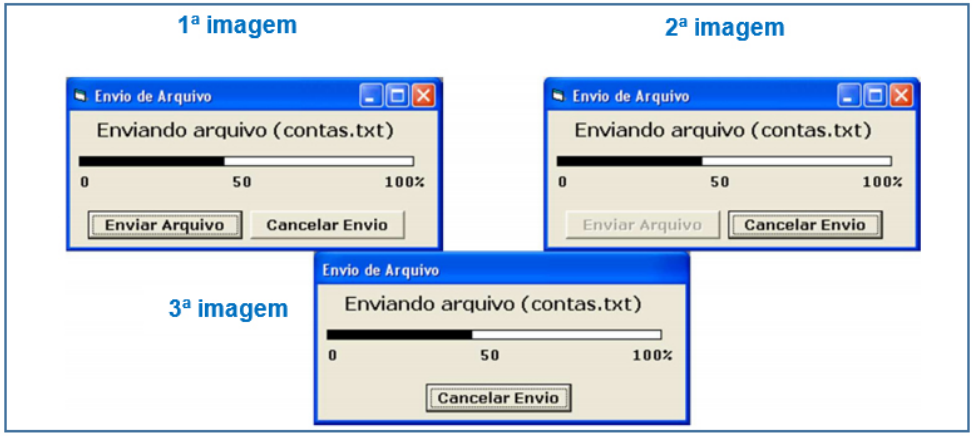
Exemplo: os controles dos carros – luzes, pisca-alerta, limpa-vidros – estão posicionados de modo a serem facilmente encontrados e usados, conforme você pode observar na Figura 1.

Figura 1 – Controles de um carro

Neste princípio de visibilidade, apenas as coisas necessárias devem estar visíveis para indicar quais partes podem e devem ser operadas naquele momento da interação.

Se você observar as imagens da Figura 2, verá que a terceira imagem é a mais adequada, pois a única coisa que o usuário poderá fazer enquanto o arquivo está sendo enviado é cancelar o envio.

Figura 2 – Envio de arquivos



Fonte: Elaborada pela autora.

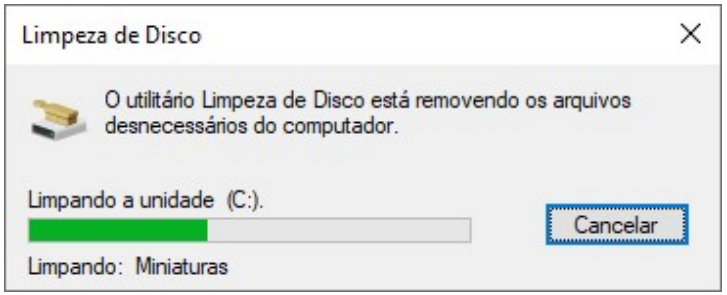
2. Feedback (retorno)

Refere-se ao **retorno de informações a respeito da ação que foi feita e do que foi realizado**, permitindo ao usuário saber se a ação foi realizada com ou sem sucesso.

Exemplo: se um usuário clica em um botão e nada acontece, ele ficará na dúvida se sua ação foi realizada com sucesso ou não. Então, **retornar rapidamente a informação do sistema para os usuários é importante para que eles saibam que efeito suas ações causaram**.

O feedback pode ser dado por meio de mensagens, sons, animação, realces etc. A Figura 3 está dando um feedback sobre a Limpeza de Disco que está sendo realizada.

Figura 3 – Feedback da Limpeza de Disco



Fonte: Elaborada pela autora.

3. Restrições

As **restrições limitam as possibilidades de uso de uma interface**. É importante inserir restrições de modo que os usuários não façam coisas inadequadas naquele momento de interação.

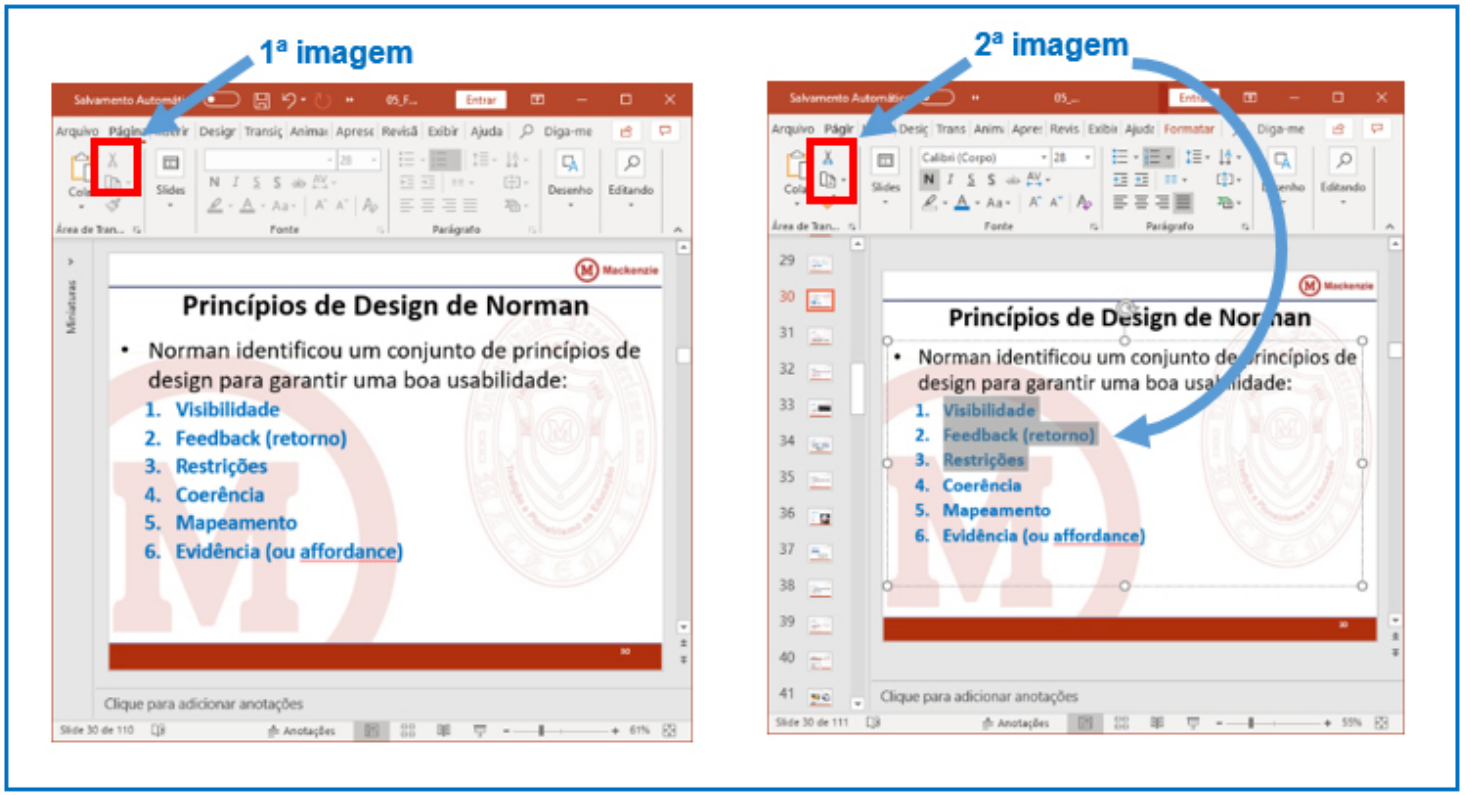
As restrições evitam a introdução de dados inválidos e a realização de ações inválidas, refreando os usuários de cometerem erros de interação.

Alguns exemplos:

- desativar o botão "Seguinte" em um formulário, enquanto todas as informações requeridas não forem preenchidas;
- **desativar** as opções **"Recortar"** e **"Copiar"** em um menu **enquanto não tiver nenhum texto selecionado**.

Analise as imagens da Figura 4 e observe que, na primeira imagem, as opções "recortar" e "colar" não estão ativas, pois nenhum texto foi selecionado. Na segunda imagem, por sua vez, estas opções estão ativas devido à seleção de texto.

Figura 4 – Opções "Recortar" e "Colar"



Fonte: Elaborada pela autora.

4. Coerência

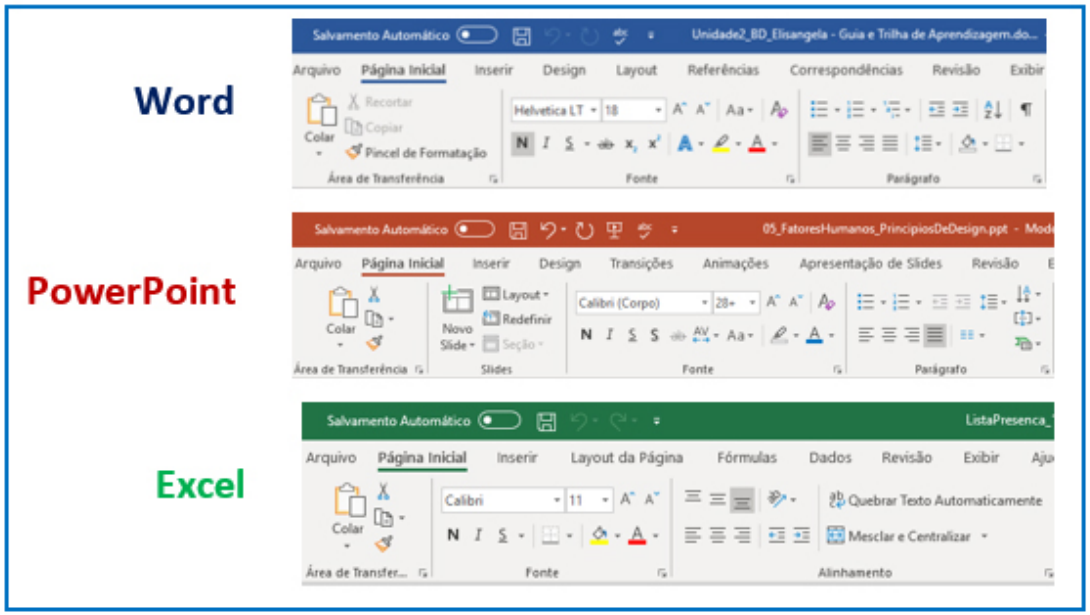
A coerência está relacionada com o uso de **operações similares e elementos similares** para alcançar **tarefas similares**. É um princípio crítico para a aprendizagem, pois ajuda os usuários a reconhecer e a aplicar conhecimento já adquirido quando surgem novas situações.

Alguns exemplos:

- se o usuário aprendeu que texto azul e sublinhado representa um link, da próxima vez que ele vir algo parecido, reconhecerá que é um link (https://www.skoob.com.br/livro/6357ED7526);

• a Figura 5 mostra um exemplo de coerência entre aplicativos, na qual os menus do Word, PowerPoint e Excel são bem semelhantes entre si, apesar de serem aplicações independentes.

Figura 5 – Coerência entre aplicativos



Fonte: Elaborada pela autora.

As incoerências, por outro lado, causam confusão, pois as coisas não funcionam como os usuários esperam, obrigando-os a memorizar exceções, o que aumenta a carga cognitiva. Isso causa indignação e pode aumentar os erros de interação.

5. Mapeamento

O mapeamento refere-se à relação entre duas coisas. No caso da interface, é a **relação entre os controles e os resultados de sua atuação**.

Os mapeamentos devem ser tão naturais, claros e evidentes quanto possível. O usuário deve se sentir confiante sobre o que acontecerá quando ele interagir com os controles da interface.

Alguns exemplos:

- se o usuário clicar na seta para cima do teclado, o cursor deve se mover para cima;
- se o usuário clicar na seta para baixo, o cursor deve se mover para baixo.

Os controles devem ser posicionados de forma lógica, de modo a terem uma correspondência com os objetos do mundo real ou com as convenções gerais.

A Figura 6 mostra a disposição das bocas e dos botões de dois fogões. O fogão da primeira imagem mostra um mapeamento que precisa de informação adicional para ser compreendido, enquanto o fogão da segunda imagem dispensa informação adicional, pois seu mapeamento é natural, oferecendo uma correspondência direta entre o botão e a boca a ser controlada.

Figura 6 – Exemplo de dois fogões com disposições diferentes dos botões

6. Evidência (*affordance*)

Evidência é a **capacidade** que um **objeto tem de nos dar pistas** sobre **sua utilização**. É uma dica de como devemos interagir com o objeto. Quanto maior for a evidência de um objeto, melhor será a identificação de seu uso e mais fácil será para o usuário saber interagir com o mesmo.

Alguns exemplos:

- botões propiciam ser apertados;
- cadeiras propiciam que se sentem nelas;
- portas de emergência – Figura 7 – têm apenas uma barra para pressionar e empurrar, não deixando dúvidas sobre o que fazer ou o sentido de abertura da porta.

Figura 7 – Porta de emergência


EXIT

2. REGRAS DE OURO DE SHNEIDERMAN

Ben Shneiderman, pesquisador da área de IHC, propôs um conjunto de “Oito regras de ouro” (GONÇALVES et al., 2017):

1. **Manter a coerência.**
2. **Oferecer usabilidade universal.**
3. **Fornecer retorno informativo.**
4. **Desenhar diálogos que indiquem a sequência de ações.**
5. **Evitar erros.**
6. **Permitir a reversão de ações.**
7. **Fornecer controle e iniciativa ao usuário.**
8. **Reduzir a carga de memória de curta duração.**

1. Manter a coerência

A **interface deve ser coerente em vários níveis**: sequência de ações, funcionalidades, aparência e terminologia. É importante, por exemplo:

- utilizar sequências de ações similares para procedimentos similares;
- manter um padrão visual para as cores, layout e fontes;
- utilizar a mesma terminologia em todos os componentes de sua interface.

A coerência desempenha um papel importante, ajudando os usuários a se familiarizarem com a interface, o que lhes permite alcançar os objetivos com facilidade e confiança.

Importante: você deve ter observado que "**Coerência"** também é um dos **Princípios de Norman.**

2. Oferecer usabilidade universal

Uma **boa interface** deve oferecer usabilidade universal, **satisfazendo** as **necessidades dos vários tipos de usuários** que irão utilizá-la.

A inclusão de funcionalidades para usuários novatos, como **explicações detalhadas**, e recursos para usuários experientes, como **atalhos**, enriquecem o design da interface e melhoram a qualidade do sistema.

3. Fornecer retorno informativo

Para **qualquer ação realizada pelo usuário deve existir uma resposta por parte do sistema** (em um tempo aceitável), de modo que o usuário saiba onde está e o que se passa.

É essencial que o usuário saiba o que está acontecendo com o sistema para ter segurança sobre a execução das tarefas requisitadas.

Um **bom exemplo de retorno** seria indicar ao usuário em que página ele se encontra de um questionário. E um **mau exemplo** seria mensagens de erro com códigos de erro, em vez de mensagens que o usuário compreenda.

Importante: você deve ter observado que "Feedback (retorno)" também é um dos Princípios de Norman.

4. Desenhar diálogos que indiquem a sequência de ações

As **sequências de ações** devem ser **organizadas em grupos com início, meio e fim**, de tal forma que o usuário consiga entender os passos e saiba quando cada um deles foi executado com sucesso.

Exemplo: depois que o usuário passa pelo processo de *check-out* de uma compra online e finaliza o pagamento, geralmente a loja virtual mostra uma mensagem agradecendo pela compra e convidando-o para voltar à loja e olhar outros produtos.

5. Evitar erros

Projete seu sistema de forma que os **usuários não consigam cometer erros graves**. Tente prever as situações e projete de forma a evitar erros.

Alguns exemplos para evitar que os usuários cometam erros:

- desativar opções dos menus que não são permitidas em determinado momento da interação;
- não permitir a introdução de caracteres em campos numéricos;
- privilegiar a seleção de valores em vez de escrita livre;
- ter mecanismos de preenchimento automático.

Caso o **usuário cometa erros**, devem haver **mecanismos que tratem e corrijam** na medida do possível e, caso não seja possível, **instrua o usuário para uma possível solução**.

Exemplo: em um formulário de cadastro, não permita que o usuário digite letras no campo do CPF, ou não deixe que ele envie arquivos de outros tipos, sendo que o sistema só pode aceitar arquivos do tipo JPG, por exemplo.

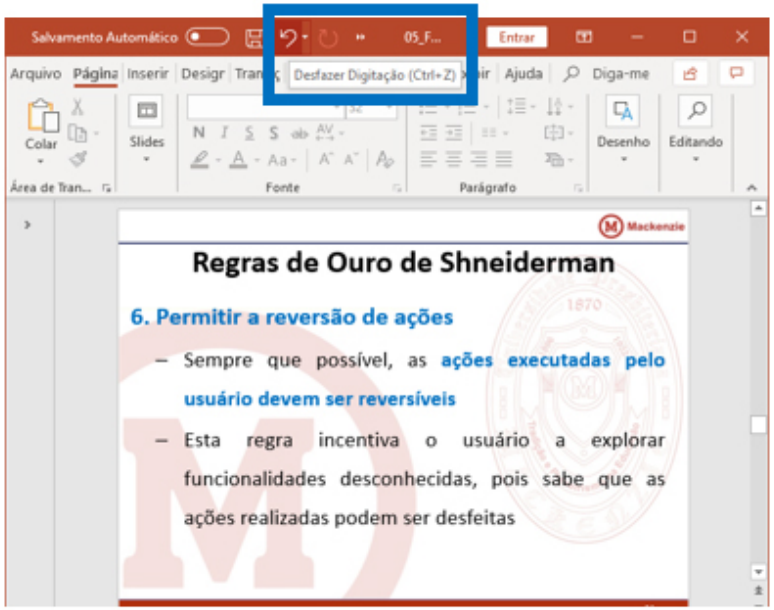
Importante: você deve ter observado que "**Restrições"** é um dos **Princípios de Norman.**

6. Permitir a reversão de ações

Sempre que possível, as **ações executadas pelo usuário devem ser reversíveis**. Esta regra incentiva o usuário a explorar funcionalidades desconhecidas, pois sabe que as ações realizadas podem ser desfeitas.

Exemplo: funcionalidade desfazer, conforme mostra a Figura 8.

Figura 8 – Funcionalidade desfazer



Fonte: Elaborada pela autora.

7. Fornecer controle e iniciativa ao usuário

Os **usuários** devem ter a **sensação de que controlam o sistema** e de que este apenas responde às suas ações. Caso contrário, os usuários sentirão ansiedade, insatisfação e frustração.

Ao criar uma interface, evite:

- surpresas ou alterações nos comportamentos conhecidos;
- dificuldades na obtenção de informações importantes;

- incapacidade, por parte do usuário, de produzir o resultado desejado.

8. Reduzir a carga da memória de curta duração

O sistema deve ser concebido para que **haja o menor esforço possível do usuário em memorizar** ou relacionar elementos na interface.

Estudos mostram que as pessoas conseguem se lembrar de sete, mais ou menos dois, itens.

Exemplo:

- Considere a sequência de letras – **H-I-C-S-A-U-I-W-M-P** – lidas sem qualquer diferença de entonação e de intervalo.
- Deve ser difícil para você lembrar desta sequência de letras.
- A sequência **I-H-C-U-S-A-W-I-M-P**, por sua vez, composta com as mesmas letras, mas em outra ordem, pode ser facilmente reproduzida por você, já que conhece bastante sobre esta disciplina.
- Para você, essa segunda sequência de letras representa **apenas três itens a serem lembrados** – **IHC USA WIMP** – ao invés de dez itens, como na sequência anterior.

3. HEURÍSTICAS DE NIELSEN

Jakob Nielsen, grande pesquisador da área de IHC, definiu as **dez heurísticas de Nielsen** (GONÇALVES et al., 2017):

1. **Tornar o estado do sistema visível.**
2. **Correspondência entre o sistema e o mundo real.**
3. **Usuário controla e exerce livre-arbítrio.**
4. **Consistência e padrões.**
5. **Evitar erros.**
6. **Reconhecimento em vez de lembrança.**
7. **Flexibilidade e eficiência.**
8. **Desenho estético e minimalista.**
9. **Ajudar o usuário a reconhecer, diagnosticar e recuperar erros.**
10. **Dar ajuda e documentação.**

1. Tornar o estado do sistema visível

O sistema deve **manter os usuários sempre informados sobre o que está acontecendo**, fornecendo **feedback adequado**, dentro de um **tempo razoável**.

Os usuários devem ser informados do tempo que falta para concluir determinada operação:

- Operações rápidas (menos de um segundo) não precisam de retorno, basta mudar o cursor.

- Operações que levam de um a dez segundos requerem indicação do tempo que falta.
- Operações que levam mais de dez segundos necessitam de uma barra de progresso.

Alguns exemplos:

- realçar a opção do menu que foi selecionada;
- indicar em que passo o usuário está, dentre um conjunto de passos para concluir uma tarefa;

indicar quanto tempo falta para concluir uma ação que o usuário solicitou, conforme mostra a Figura 9.

Figura 9 – Usuário sendo informado sobre o processo de download

Fonte: Elaborada pela autora.

Importante: você deve ter observado que **feedback** é um dos **Princípios de Norman** – "**Feedback (retorno)",** e, também, uma das **Regras de Ouro de Shneiderman** – "**Fornecer retorno informativo"**.

2. Correspondência entre o sistema e o mundo real

O **sistema deve falar a linguagem do usuário**, utilizando palavras, frases e conceitos familiares a ele, em vez de termos orientados ao sistema. Se o usuário fala português, a interface deve estar em português e, também, não utilizar termos técnicos.

Exemplo: se você for desenvolver uma interface para médicos, utilize termos da área da medicina, pois os usuários entenderão a mensagem.

3. Usuário controla e exerce livre-arbítrio

Os **usuários** devem ter **liberdade para selecionar e realizar as tarefas pela ordem que quiserem**, em vez de o sistema impor esta sequência.

O sistema deve fornecer maneiras para permitir que os usuários saiam facilmente do estado indesejado em que se encontram, utilizando "**saídas de emergência" claramente identificadas**.

Exemplos: um sistema deve:

- oferecer operações desfazer e/ou refazer uma operação (você já viu o "desfazer" na Figura 8);
- permitir o cancelamento de operações demoradas;
- possibilitar a saída do sistema a qualquer momento.

Importante: você deve ter observado que "**Fornecer controle e iniciativa ao usuário**" é uma das **Regras de Ouro de Shneiderman**.

4. Consistência e padrões

Os usuários não devem ter de se preocupar em adivinhar que palavras, situações ou ações em contextos diferentes significam a mesma coisa. **Elementos semelhantes** na interface **devem parecer** e **agir de forma similar**.

Exemplo: um tipo de consistência muito importante está relacionado com os termos usados na interface. Se a interface disser em uma parte "**Salvar**", em outra "**Guardar**" e em outra "**Arquivar**", os usuários podem pensar que são ações diferentes. Escolha, por exemplo, colocar "**Salvar**" em toda a interface.

Importante: você deve ter observado que "**Coerência"** é um dos **Princípios de Norman** e **"Manter a coerência"** é uma das **Regras de Ouro de Shneiderman**.

5. Evitar erros

A **interface deve fazer** com que **seja difícil cometer erros**. Melhor do que uma boa mensagem de erro é um design cuidadoso que previne a ocorrência de erros.

Alguns exemplos:

- a opção "Copiar" está desativada quando o usuário não selecionou nenhum texto, conforme você já observou na Figura 4;
- o botão "Next" está desativado, pois o usuário ainda não completou todas as informações necessárias para avançar;
- os campos de texto indicam o formato dos dados introduzidos;
- um diálogo de confirmação é apresentado quando o usuário realiza uma operação destrutiva.

Importante: você deve ter observado que "**Restrições"** é um dos **Princípios de Norman** e **"Evitar Erros"** é uma das **Regras de Ouro de Shneiderman**.

6. Reconhecimento em vez de lembrança

A interface deve **tornar visíveis objetos, ações e opções** de modo a **minimizar a carga cognitiva**.

A interface deve privilegiar o uso de:

- menus ao invés de linguagens de comandos;
- caixas de seleção ao invés de campos de texto;
- imagens ao invés do nome dos arquivos;
- ícones associados a ações etc.

As interfaces baseadas em **linhas de comando violam esta heurística**, pois exigem que o usuário se lembre de comandos. Os **menus já respeitam esta heurística**, pois o usuário não tem de lembrar dos comandos, mas apenas reconhecê-lo na lista de comandos apresentados no menu.

Exemplo: o usuário pode escolher um produto de uma lista, em vez de lembrar de seu nome ou código.

Importante: você deve ter observado que **"Reduzir a carga de memória de curta duração"** é uma das **Regras de Ouro de Shneiderman**.

7. Flexibilidade e eficiência

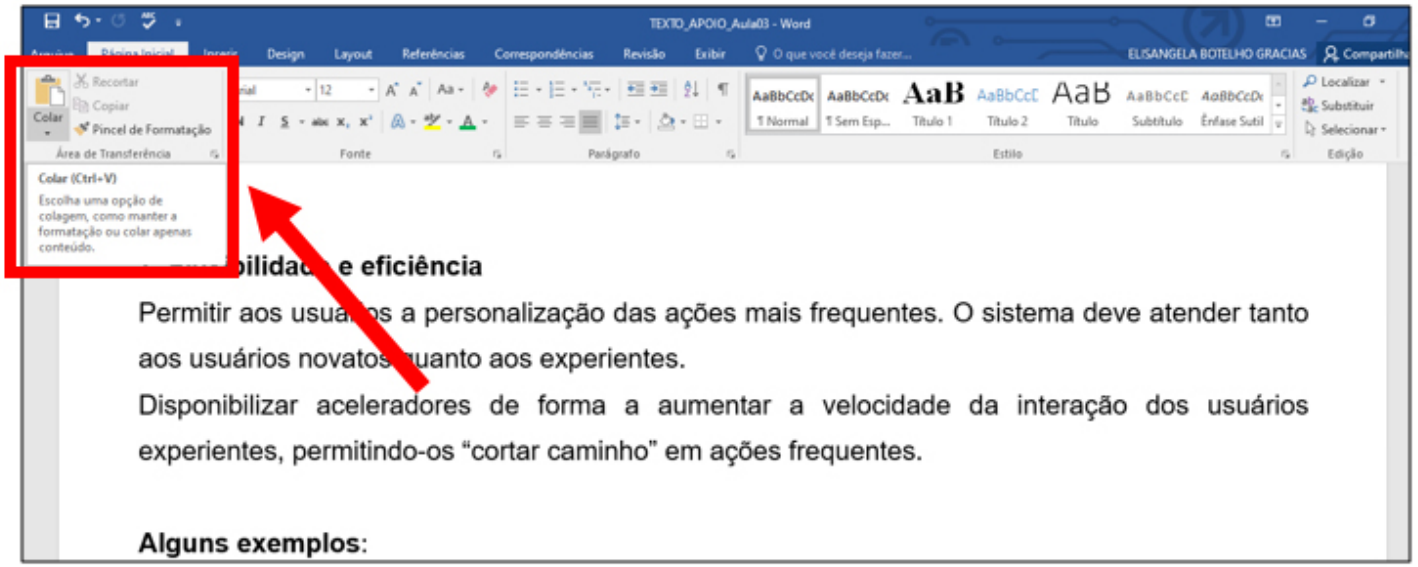
Permitir aos usuários a **personalização das ações mais frequentes**. O sistema deve atender tanto **usuários novatos** quanto **experientes**.

É importante disponibilizar aceleradores aos usuários experientes, de forma a aumentar a velocidade da interação, permitindo-os "cortar caminho" em ações frequentes.

Alguns exemplos:

- atalhos dos menus e dos botões, conforme você pode observar na Figura 10;
- histórico dos últimos arquivos abertos;
- escolha dos botões que desejam colocar na barra de ferramentas;
- possibilidade de navegar entre os campos de um formulário usando o mouse ou o teclado.

Figura 10 – Atalho para a ação “Colar”



Fonte: Elaborada pela autora.

Importante: você deve ter observado que **“Oferecer usabilidade universal”** é uma das **Regras de Ouro de Shneiderman**.

8. Desenho estético e minimalista

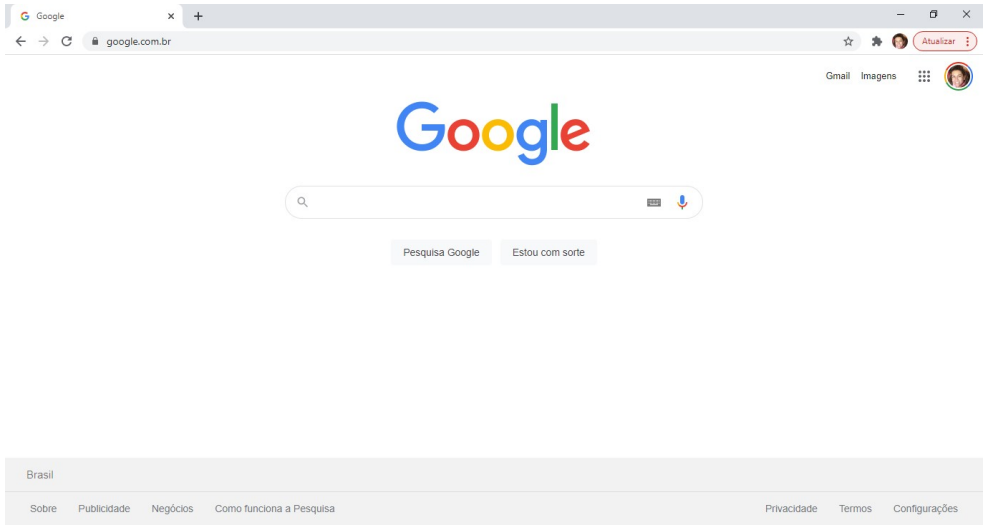
Os **diálogos da interface não devem conter informação irrelevante** ou raramente necessária. Qualquer unidade de informação, fora o diálogo, competirá com unidades relevantes de informação e diminuirá sua visibilidade relativa.

Nesta heurística **menos é mais**, ou seja, menos para aprender e para distrair. Deve-se apresentar apenas a informação de que o usuário necessita, deixando de fora tudo que é irrelevante ou desnecessário.

Alguns exemplos:

- alinhar o texto à esquerda e os números inteiros à direita;
- ter um bom contraste entre o texto e o fundo;
- a página de entrada do Google – Figura 11 – é um bom exemplo, pois apresenta apenas o que realmente é necessário para realizar a tarefa.

Figura 11 – Página de entrada do Google



Fonte: Elaborada pela autora.

9. Ajudar os usuários a reconhecer, diagnosticar e recuperar erros

As **mensagens de erro** devem ser expressas em **linguagem clara** (**sem códigos**), **indicando** precisamente o **problema** e, construtivamente, **sugerindo** uma **solução**.

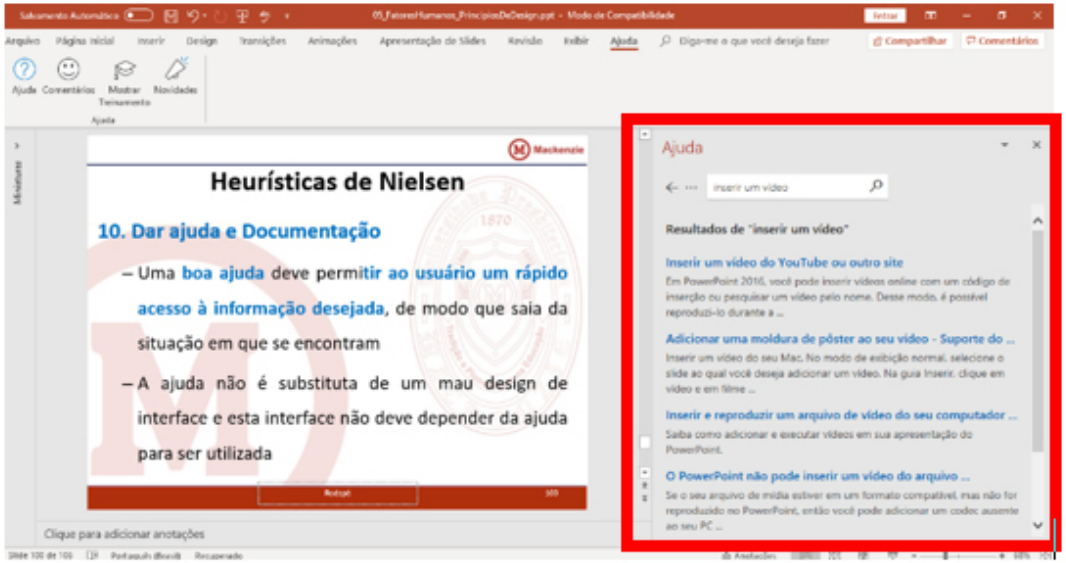
Quando não é possível prevenir o erro, é necessário **mostrar uma boa mensagem**, que deve:

- **ser precisa:** indicar o problema. *"Não é possível abrir o arquivo heuristicas.pdf"* em vez de *"Não é possível abrir o arquivo"*;
- **falar a linguagem do usuário:** evitar termos técnicos e códigos de erro. *"Falta preencher o campo Nome"* em vez de *"Erro na linha 273, código 123-56"*;
- **dar ajuda construtiva:** *"Número de telefone inválido no campo Telefone, por favor, utilize apenas dígitos"* em vez de *"Dados inválidos"*.

10. Dar ajuda e documentação

Embora seja melhor um sistema que possa ser usado sem documentação, é necessário prover ajuda e documentação. Essas **informações devem ser fáceis de encontrar**, **focadas na tarefa do usuário** e **não muito extensas**, conforme você pode observar na Figura 12.

Figura 12 – Ajuda para inserir um vídeo no PowerPoint



Fonte: Elaborada pela autora.

A ajuda não é substituta de um mau design de interface e a interface não deve depender da ajuda para ser utilizada.

4. CONCLUSÃO SOBRE OS PRINCÍPIOS DE DESIGN DE INTERFACES

Os princípios, regras de ouro ou heurísticas são recomendações ou regras práticas que podem orientar uma decisão de design. Podem ser usados durante a fase de concepção de uma interface ou usadas para avaliar uma solução já pronta.

Os **Princípios de Design de Norman** colocam as tarefas e as **necessidades dos usuários no centro do desenvolvimento das interfaces**.

As **Regras de Ouro de Shneiderman** fornecem um **bom resumo dos princípios fundamentais do design de interfaces**.

E as **Heurísticas de Nielsen** são as mais abrangentes e **podem ser aplicadas praticamente a qualquer tipo de sistema** e, por isso, **são as mais utilizadas.**

Importante: embora não sejam perfeitos, estes três conjuntos de regras são um bom ponto de partida para o desenvolvimento de interfaces com uma boa usabilidade. E eles devem ser interpretados e ajustados para o contexto da interface a ser desenvolvida.

Importante: Não deixe de assistir à videoaula **“Princípios de Design de Interfaces”**, com a professora Elisângela Botelho Gracias. Nesta aula, as regras práticas de design serão explicadas e exemplificadas.

5. PROTOTIPAGEM

Na Aula 1 já falamos, brevemente, sobre protótipos. Mas você sabe o que é um protótipo?

- É uma **representação concreta**, mas **parcial**, do **sistema** que se pretende **desenvolver**.
- Permite aos usuários interagirem com o sistema e explorar sua adequação.
- **Reduz** o **tempo** e o **custo** de **desenvolvimento**.

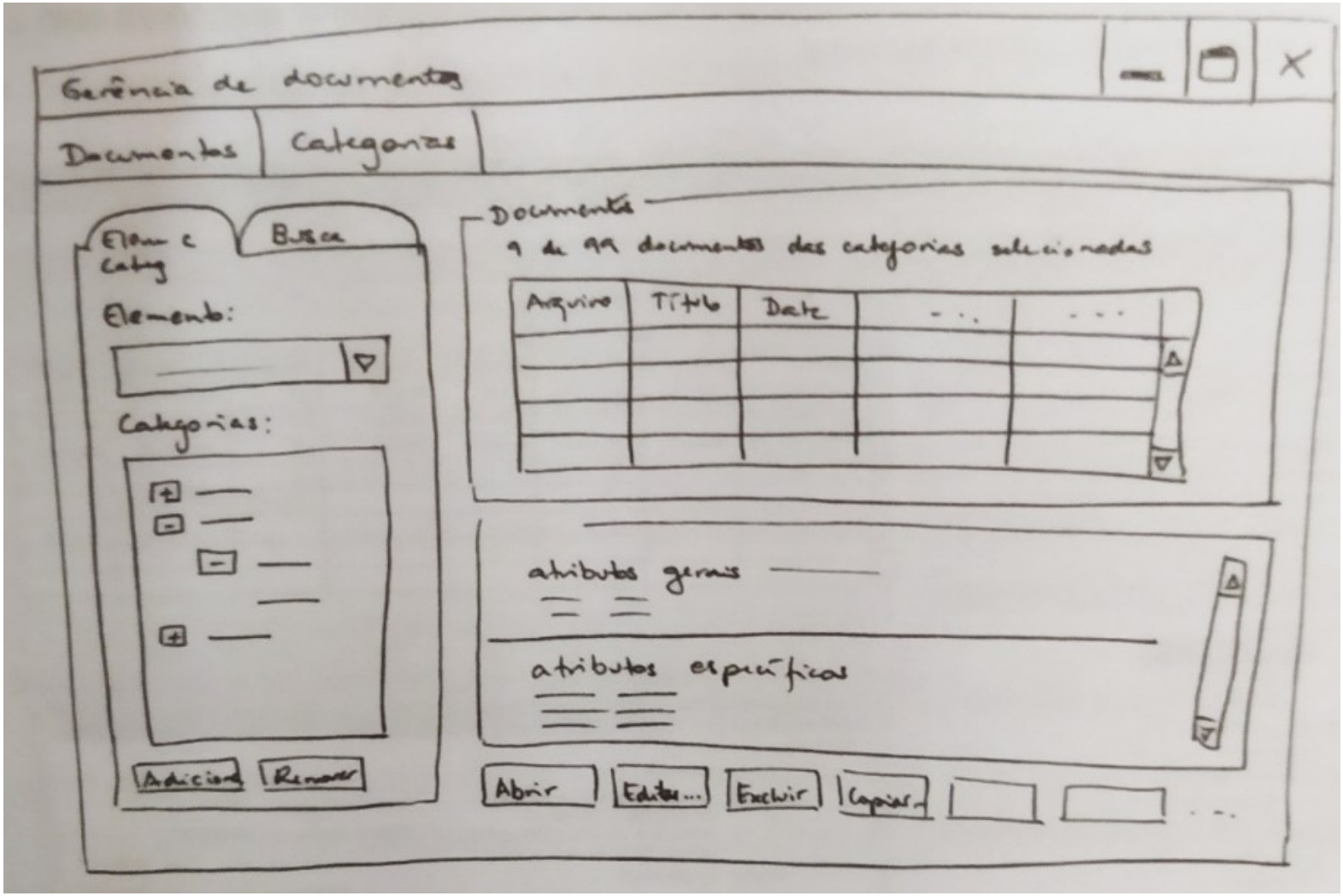
Importante: os usuários não sabem o que querem até que vejam algo que possam experimentar.

Um protótipo pode ser feito de algo simples, como papel ou um outro material adequado, ou, ainda, ser desenvolvido com a utilização de um software.

Alguns exemplos de protótipos:

- o rascunho de uma tela;
- conjunto de telas feito em papel, conforme você pode observar na Figura 13;
- um vídeo simulando uma tarefa etc.

Figura 13 – Protótipo em papel da tela de uma interface



Fonte: BARBOSA; SILVA (2010).

Você consegue imaginar as inúmeras **vantagens de um protótipo**?

- São mais rápidos de construir do que o produto final.;

- Podem ser avaliados, mesmo feitos em papel, sendo possível receber feedback dos aspectos bons e ruins desde o início do projeto.
- Pode-se experimentar várias alternativas de design.
- Quando for encontrado problema na solução, sua alteração é fácil e rápida.
- Falhas graves podem ser descartadas sem muito custo.
- Permite manter o design centrado no usuário, na medida em que se constrói algo de concreto para mostrar/ testar com os usuários, mantendo-os envolvidos no processo de design.

Na Aula 1 você aprendeu sobre as várias atividades da Engenharia de Usabilidade, e uma delas é "**Faça designs paralelos"**. É nesta atividade que você criará protótipos.

Nesta atividade, várias pessoas (três ou mais) criam soluções alternativas para a mesma interface, com base nos mesmos requisitos, ou seja, cada pessoa trabalha individualmente e, depois, compartilha sua solução com os demais. A equipe, então, analisa as várias sugestões e cada designer aproveita as melhores ideias para aprimorar sua solução.

Características dos protótipos

Os protótipos podem ser classificados por:

- abrangência e profundidade;
- fidelidade e funcionalidade.

Abrangência

A abrangência refere-se à **fração de tarefas oferecidas pelo protótipo**.

Um protótipo com muita abrangência permite testar várias tarefas, ou seja:

- inclui a interface de usuário (protótipo horizontal);

- pode-se considerar o *front-end* completo, mas sem o *back-end,* ou seja*,* não se consegue testar uma tarefa real;
- é mais rápido de desenvolver e permite testar toda a interface.

Profundidade

A profundidade está relacionada com a **quantidade de funcionalidade de cada tarefa.**

Um protótipo com muita profundidade (protótipo vertical) apresenta:

- muitas funcionalidades para poucas tarefas;
- permite testar apenas uma parte do sistema, usando uma base de dados real, ou seja, tem o *front-end* e o *back-end* de uma única tarefa implementados.

Importante: os protótipos horizontais são os mais comuns; no entanto, se existir um aspecto do sistema que não temos certeza se conseguiremos implementar de acordo com os requisitos, é melhor criar um protótipo vertical.

Fidelidade

A fidelidade refere-se ao **aspecto visual do protótipo** (interface ou dispositivo físico), como tipos de letra, mensagens, cores, imagens etc. Em termos de fidelidade, podemos ter:

- Protótipo de Baixa Fidelidade (PBF).
- Protótipo de Alta Fidelidade (PAF).

Protótipo de Baixa Fidelidade (PBF) são representações artísticas usando **esboços em papel**, sem muito esforço, e **omitindo muitos detalhes** (cores, imagens, ícones), conforme você pode observar nas Figuras 13 e 17.

Algumas características de um PBF:

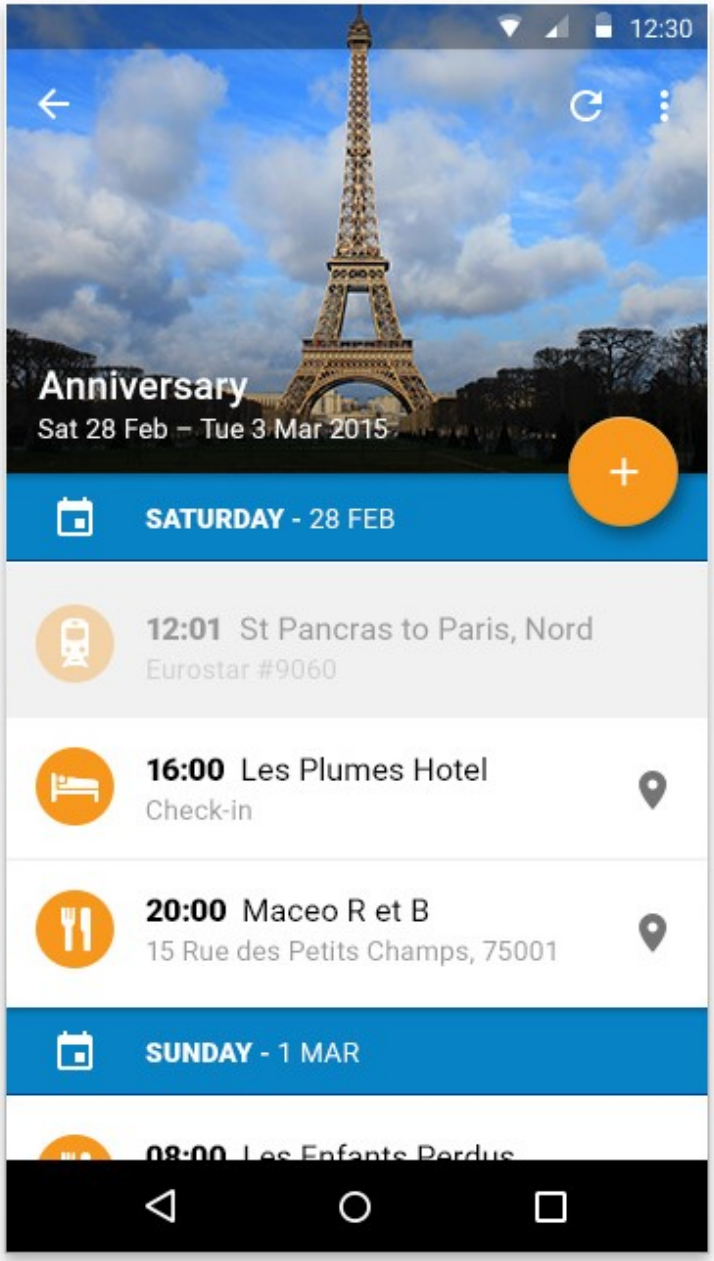
- Possui baixo nível de detalhes.
- É usado na fase inicial de desenvolvimento para compreensão dos requisitos.
- A representação das telas é feita em papel e caneta, preferencialmente. Mas também se pode utilizar ferramentas computacionais.
- Custo reduzido.
- Produção extremamente rápida.
- São descartados após a fase inicial.

Protótipo de Alta Fidelidade (PAF) assemelha-se ao produto final, em termos de aparência visual, interatividade e navegação. Você pode observar um PAF na Figura 14.

Algumas características de um PAF:

- É desenvolvido e apresentado no computador.
- Pode ser descartado ou evoluir até a versão final.
- Pode proporcionar testes de funcionalidades do sistema.
- Protótipos que evoluem reduzem o tempo e o custo de desenvolvimento do produto final.

Figura 14 – Protótipo de alta fidelidade

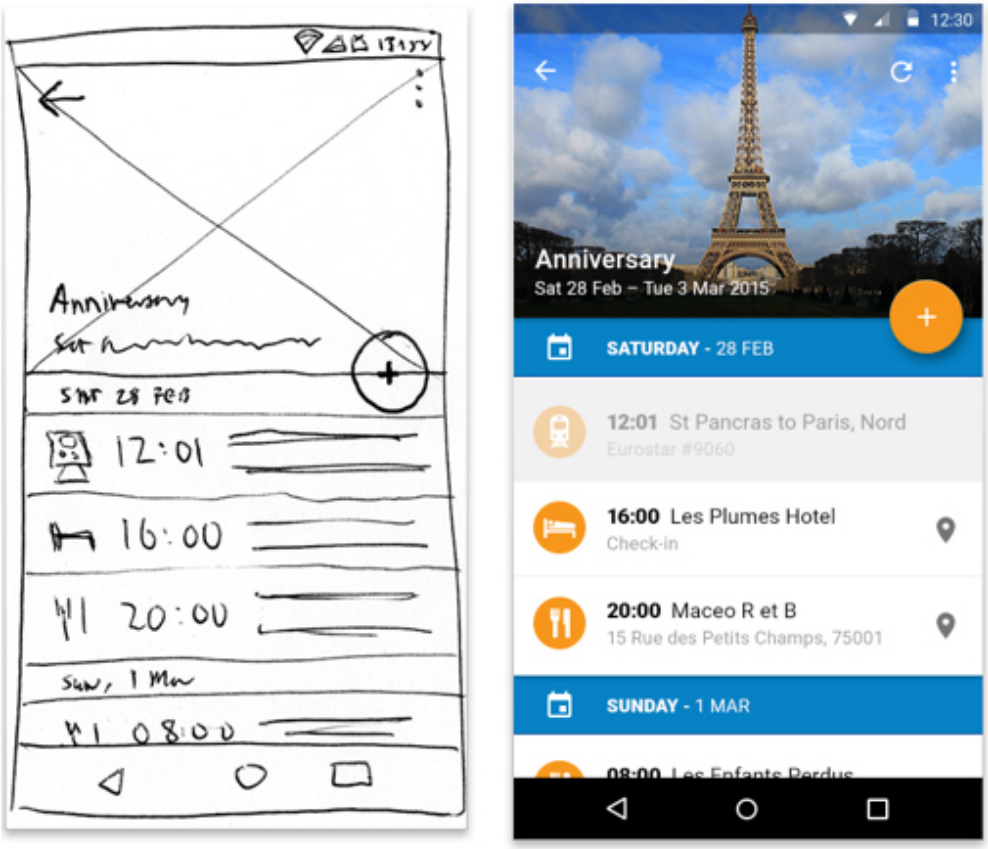

12:30
Anniversary
Sat 28 Feb – Tue 3 Mar 2015
SATURDAY - 28 FEB
12:01 St Pancras to Paris, Nord
Eurostar #9060
16:00 Les Plumes Hotel
Check-in
20:00 Maceo R et B
15 Rue des Petits Champs, 75001
SUNDAY - 1 MAR

Fonte: Curso Ironhack UX Design apud Castro (2019) .

PBF *versus* **PAF**

A Figura 15, a seguir, mostra um PBF (à esquerda) e um PAF (à direita). Observe bem as diferenças entre estes dois protótipos.

Figura 15 – PBF *versus* PAF



Fonte: Curso Ironhack UX Design apud Castro (2019).

A tabela, a seguir, faz um paralelo entre PBF e PAF, mostrando as vantagens e desvantagens de cada um.

	Protótipo de Baixa Fidelidade	**Protótipo de Alta Fidelidade**
Vantagem	Custo baixo	

Desvantagem		Custo mais alto
Vantagem	Útil para identificar requisitos	
Desvantagem		Não serve para identificar requisitos
Vantagem		Totalmente interativo
Desvantagem	Uso conduzido pelo facilitador	
Vantagem		Uso para exploração e teste
Desvantagem	Utilidade limitada para testes de usabilidade	
Vantagem		Serve como uma especificação viva
Desvantagem	Limitações de fluxo e navegação	

Enquanto os **PBF são mais rápidos de fazer**, os **PAF consomem mais tempo**, porque exigem a especificação de tipos de letra, mensagens, estilo, tamanho dos elementos, cores, imagens etc.

Um PAF distorce a percepção de quem testará o protótipo, pois tem o aspecto de um produto acabado, e os usuários comentam sobre as cores, o tipo de letra usado e os alinhamentos, em vez de se preocuparem com o fluxo da interação ou a terminologia utilizada. Além disso, o PAF desencoraja os usuários a pedirem alterações, pois acham que o design já está pronto.

Alguns autores também citam o **Protótipo de Média Fidelidade (PMF)**, que fica entre o PBF e o PAF, e possui as seguintes características:

- Apresenta aspectos visuais mais próximos do definitivo.
- Apresenta algumas sequências de diálogo com o usuário para simular a navegação.
- Pode fazer uso de ferramentas computacionais para sua criação.

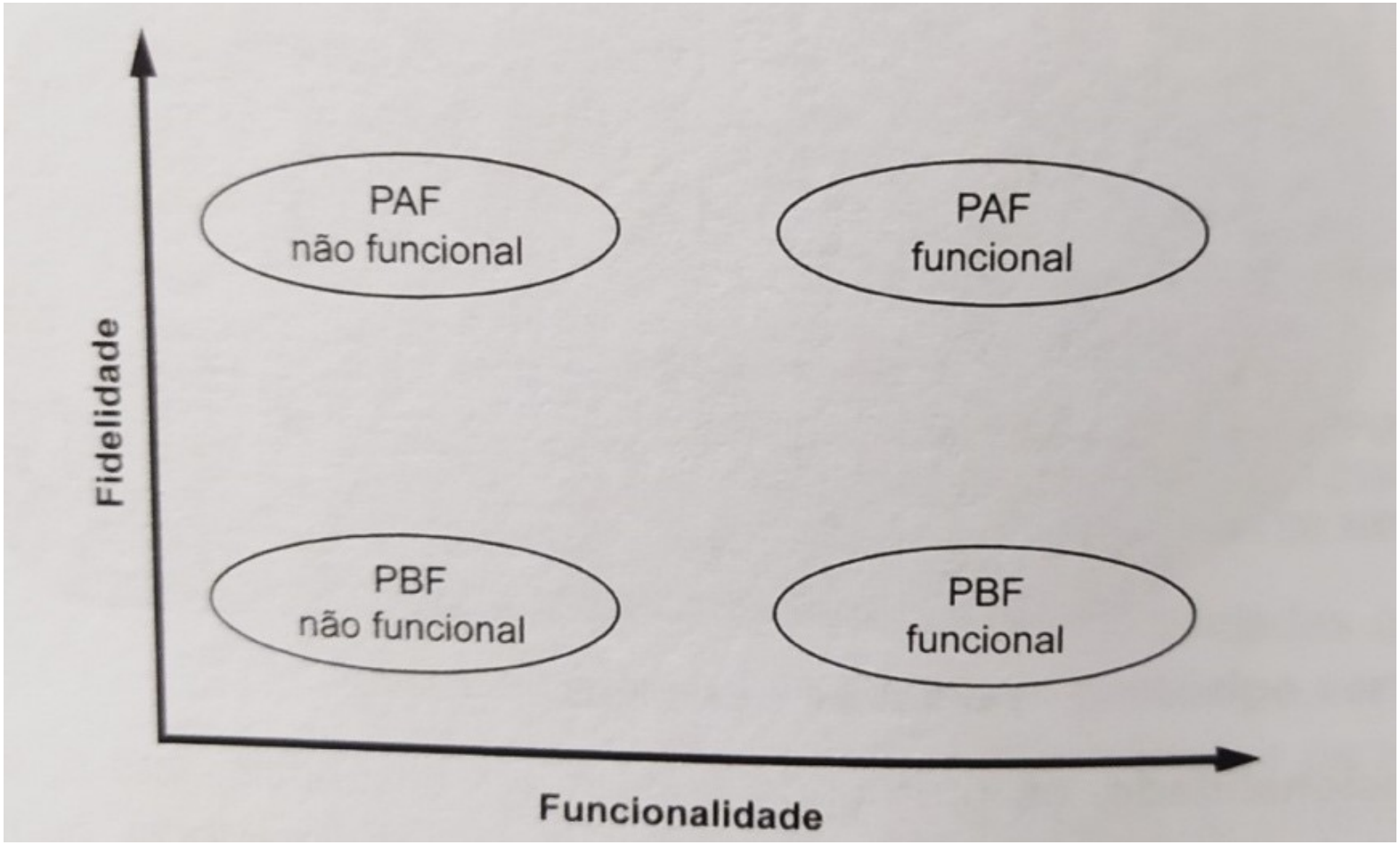
Funcionalidade

Quanto à **funcionalidade**, os protótipos podem ser:

- Funcionais: implementados utilizando código. Portanto, são executáveis em um sistema computacional.
- Não funcionais: necessitam da intervenção humana para funcionar.

A Figura 16 mostra como os **PBF e PAF** se posicionam nas características **Fidelidade e Funcionalidade**.

Figura 16 – Como os PBF e PAF se posicionam nas características funcionalidade e fidelidade



Fonte: Adaptado de GONÇALVES et al. (2017).

Como você pode observar na Figura 16, podemos ter:

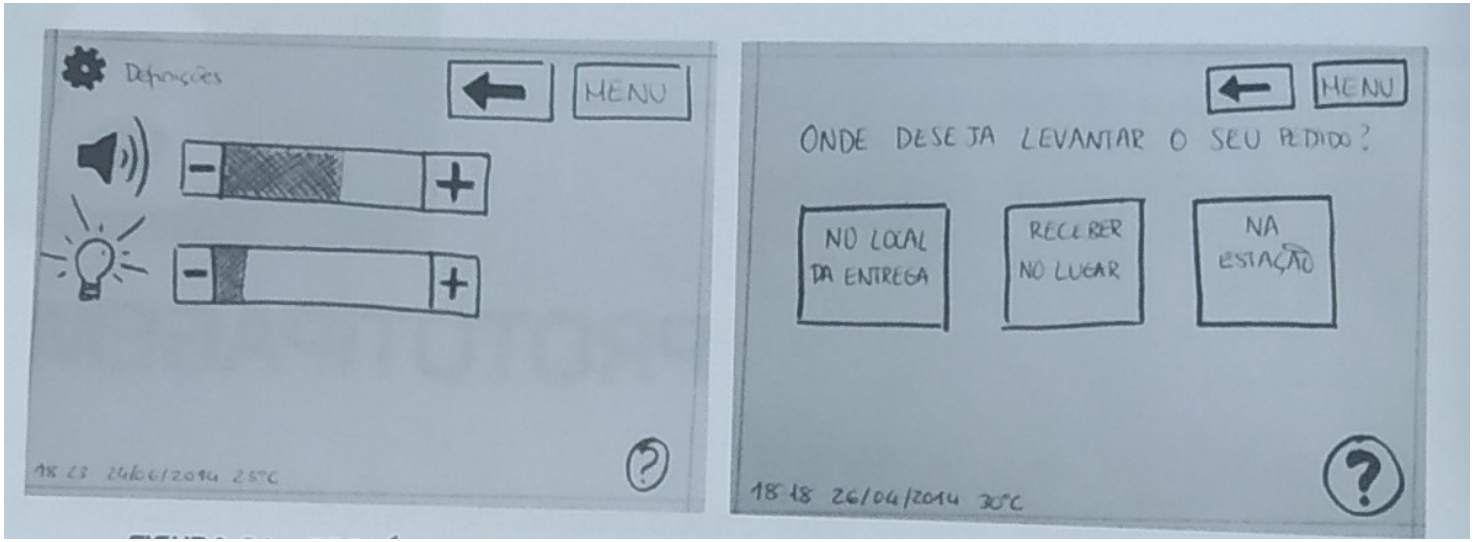
- **PAF** que **não são funcionais.**
  - Exemplo: protótipo em papel, mas com aspecto muito semelhante ao protótipo final.

- **PAF funcional** (semelhante ao sistema funcional final).
- **PBF** que **não são funcionais.**
  - Exemplo: protótipo em papel.
- **PBF** que **são funcionais.**
  - Exemplo: implementados em código, cujo aspecto da interface continua a ser o de um esboço (sem cores, sem imagens etc.)

Protótipo em papel

Os **protótipos em papel** são uma excelente escolha para as **fases iniciais** no **desenvolvimento de uma interface**. Observe o exemplo na Figura 17.

Figura 17 – Protótipo em papel de duas telas de uma interface



Fonte: GONÇALVES et al. (2017).

Você acha que vale a pena fazer um protótipo em papel? Analise as vantagens:

- É mais rápido construir uma interface utilizando lápis e papel do que programá-la usando código.
- É mais rápido e fácil fazer alterações no papel do que no código.
- Considerando o tempo que levaria para programar, pode-se criar vários esboços alternativos.
- Concentram a atenção do design no que realmente importa, ao invés de se preocupar com detalhes (tipo de fonte, cores, alinhamento etc.), o que poderia distrair o designer.
- A utilização de um programa é mais lenta do que desenhar à mão livre e obriga o designer a tomar um conjunto de decisões relacionadas com detalhes que tomam muito tempo.

Importante: protótipos em papel melhoram o feedback que recebemos dos usuários, pois eles estarão menos propensos a procurar pequenos defeitos em detalhes que não são relevantes nesta fase de desenvolvimento e não se queixarão do esquema de cores, pois ainda não existem. Com isso, os **usuários farão sugestões mais criativas**.

Os protótipos em papel devem ser maiores que o tamanho real do sistema, para que os usuários possam apontar com o dedo, e monocromáticos, para não distrair a atenção de coisas importantes.

Importante: Não deixe de assistir à videoaula **“Características e Vantagens de um Protótipo”**, com a professora Elisângela Botelho Gracias. Nesta aula, você conhecerá as características de um protótipo e suas vantagens.

6. REFERÊNCIAS

BARBOSA, S. D. J.; SILVA, B. S. **Interação humano-computador**. Rio de Janeiro: Elsevier, 2010.

BENYON, D. **Interação Humano-Computador**. 2. ed. São Paulo: Pearson, 2011.

GONÇALVES, D.; FONSECA, M. J.; CAMPOS, P. **Introdução ao Design de Interfaces**. 3. ed. Lisboa: FCA Editora, 2017.