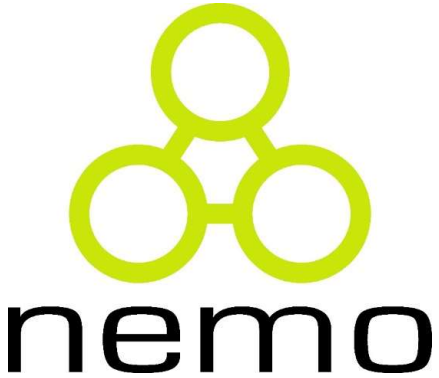


Projeto de Sistemas de Software

Jordana S. Salamon

jssalamon@inf.ufes.br

jordanasalamon@gmail.com

DEPARTAMENTO DE INFORMÁTICA

CENTRO TECNOLÓGICO

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

Introdução

- Sistemas são desenvolvidos para serem utilizados por pessoas. Assim, um aspecto fundamental no projeto de sistemas é a interface com o usuário (IU).

- O projeto da IU estabelece uma forma de comunicação entre as pessoas e o sistema computacional. A IU define como um usuário comandará o sistema e como o sistema apresentará as informações a ele.

- Um dos princípios fundamentais para um bom projeto de software é a separação da apresentação (camada de IU) da lógica de negócio (camada de LN).

Introdução

- Essa separação é importante por diversas razões, dentre elas:
- • O projeto de IU e o projeto da LN tratam de diferentes preocupações. No primeiro, o foco está nos mecanismos de interação e em como dispor uma boa IU. O segundo concentra-se em conceitos e processos do negócio.

- • Usuários podem querer ver as mesmas informações de diferentes maneiras (p.ex., usando diferentes interfaces, tais como interfaces ricas de sistemas desktop, interfaces de aplicações Web tradicionais, interfaces de linha de comando etc.). Neste contexto, separar a IU da LN permite o desenvolvimento de múltiplas apresentações.

- • Objetos não visuais são geralmente mais fáceis de testar do que objetos visuais. Ao separar objetos da LN de objetos de IU, é possível testar os primeiros sem envolver os últimos.

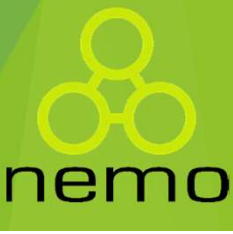
Introdução

- Dada a importância dessa separação, é importante usar algum padrão arquitetônico que trabalhe essa separação, tal como o padrão Modelo-Visão-Controlador (MVC).

- A camada de IU envolve dois tipos de funcionalidades:

- • **Visão**: refere-se aos objetos gráficos usados na interação com o usuário;

- • **Controle de Interação**: diz respeito ao controle da lógica da interface, envolvendo a ativação dos objetos gráficos (p.ex., abrir ou fechar uma janela, habilitar ou desabilitar um item de menu etc.) e o disparo de ações.



O Padrão Modelo-Visão-Controlador

- O padrão Modelo-Visão-Controlador (MVC) considera três papéis relacionados à interação humano-computador.

- O **modelo** refere-se aos objetos que representam alguma informação sobre o negócio e corresponde, de fato, a objetos da camada de Lógica de Negócio.

- A **visão** refere-se à entrada e à exibição de informações na IU.

- Qualquer requisição é tratada pelo terceiro papel: o **controlador**. Este pega a entrada do usuário, envia uma requisição para a camada de lógica de negócio, recebe sua resposta e solicita que a visão se atualize conforme apropriado.

O Padrão Modelo-Visão-Controlador

- **Assim, a IU é uma combinação de visão e controlador.** Em outras palavras, elementos da visão representam informações de modelo e as exibem ao usuário, que pode enviar, por meio da visão, requisições ao sistema.

- Essas requisições são tratadas pelo controlador, que as repassa para classes do modelo. Uma vez alterado o estado dos elementos do modelo, o controlador pode, se apropriado, alterar elementos de visão a serem exibidos ao usuário.

- Assim, o controlador situa-se entre o modelo e a visão, isolando-os um do outro.

O Padrão Modelo-Visão-Controlador

- Neste ponto é importante distinguir os controladores do padrão MVC das classes gerenciadoras de caso de uso do Componente de Gerência de Tarefas (cgt).

- Estas últimas representam classes da lógica de negócio (lógica de aplicação) que encapsulam e centralizam o tratamento de casos de uso.

- Já um controlador do padrão MVC é um controlador de interação, ou seja, ele controla a lógica de interface, abrindo e fechando janelas, habilitando ou desabilitando botões, enviando requisições etc.

O Padrão Modelo-Visão-Controlador

- O padrão MVC trabalha dois tipos de separação.
- Primeiro, separa a apresentação (visão) da lógica de negócio (modelo), conforme advogado pelas boas práticas de projeto.
- Segundo, mantém também separados o controlador e a visão.
- Essa segunda separação (entre a visão e o controlador) é menos importante que a primeira (entre a visão e a lógica de negócio). Vários sistemas têm um único controlador por visão e, por isso, a separação entre a visão e o controlador muitas vezes não é feita.

O Padrão Modelo-Visão-Controlador

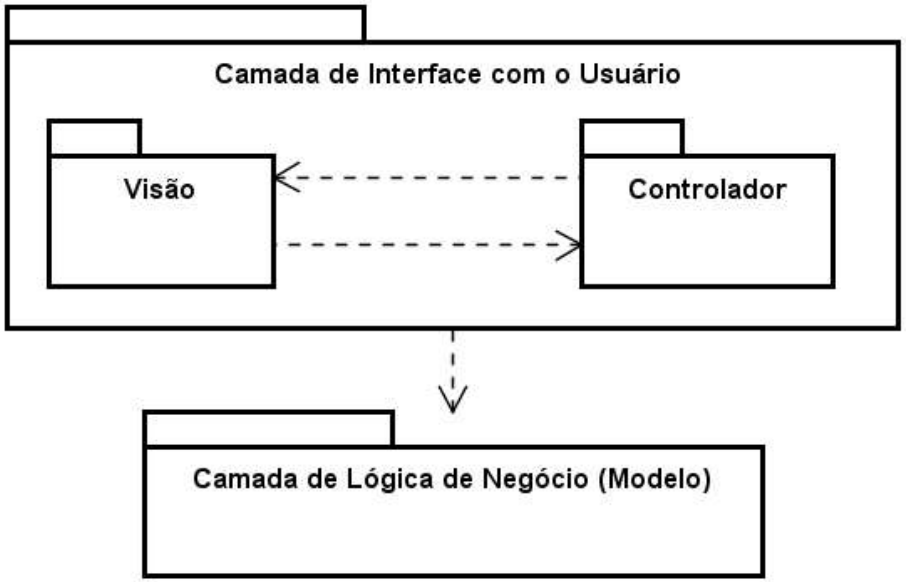
- Em sistemas de interfaces ricas desktop ela é muitas vezes desprezada. Contudo, em interfaces Web, essa separação é comum, já que a parte de visão front end é naturalmente separada do controlador. De fato, a maioria dos padrões de projeto de interfaces Web é baseada nesse princípio.

- A separação entre visão e controlador dá origem a dois tipos de classes que podem ser organizados em dois pacotes na camada de interface com o usuário:

  - o **Componente de Interação Humana** (cih), que é responsável pelas interfaces com o usuário propriamente ditas (janelas, painéis, botões, menus etc.) e representa a visão no modelo MVC;

O Padrão Modelo-Visão-Controlador

- e o **Componente de Controle de Interação** (cci), que é responsável por controlar a interação, recebendo requisições da interface, disparando operações da lógica de negócio e atualizando a visão com base no retorno dessas operações. O cci é, portanto, o controlador do modelo MVC.



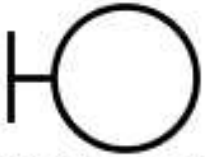
nemo

O Padrão Modelo-Visão-Controlador

- É importante frisar que, mesmo quando se opta por não fazer a separação física em pacotes de visão e controlador, é útil ter classes distintas para desempenhar esses papéis.

- As classes controladoras de interação devem ser marcadas com o estereótipo <<control>> para diferenciá-las das classes de visão, que devem ser marcadas com o estereótipo <<boundary>>.

Controlador de Interação (Control)



Objeto de Interface (Boundary)

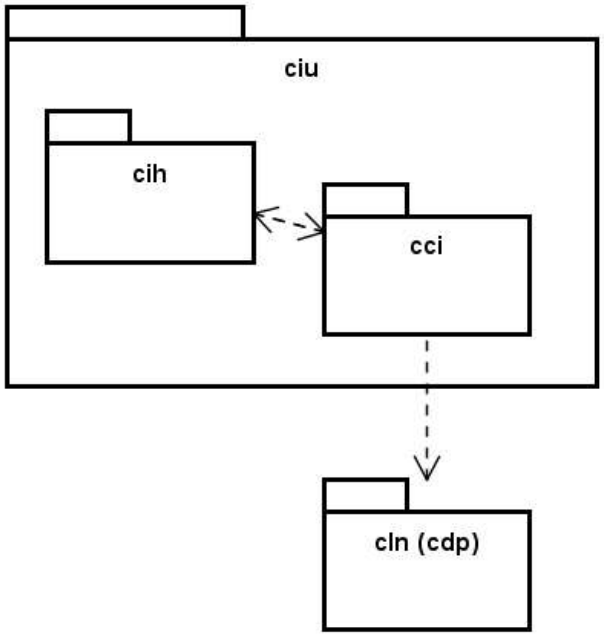
O Padrão Modelo-Visão-Controlador

- No que se refere à interação entre as camadas de IU e Lógica de Negócio (modelo), ela se dá de maneiras distintas em função do padrão arquitetônico adotado nesta última.

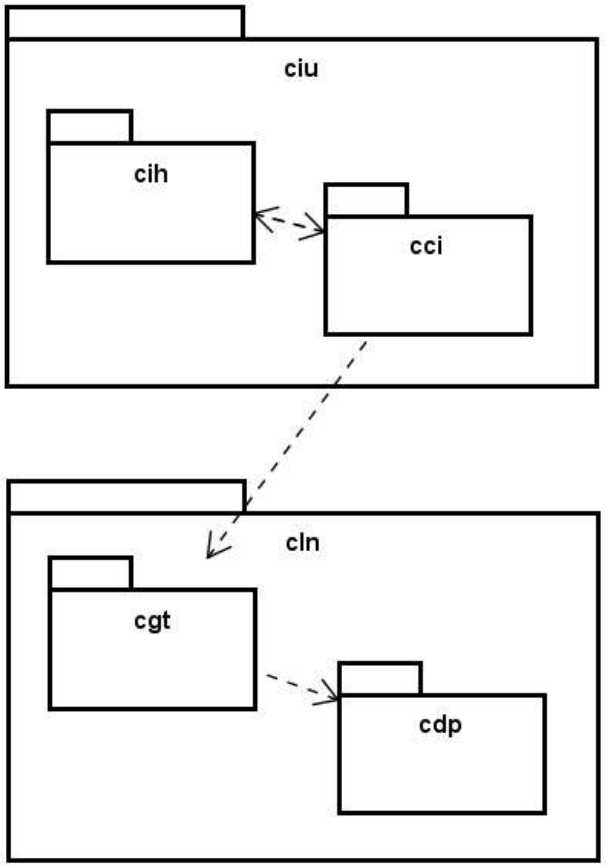
- Quando o padrão Modelo de Domínio é adotado, os controladores de interação enviam as requisições diretamente para os objetos do domínio do problema (cdp), uma vez que, neste caso, não existem objetos gerenciadores de tarefa (cgt).

- Quando o padrão Camada de Serviço é adotado, as requisições dos controladores de interação são enviadas para os objetos gerenciadores de tarefas (cgt).

O Padrão Modelo-Visão-Controlador



Padrão Modelo de Domínio



Padrão Camada de Serviço

nemo

O Padrão Modelo-Visão-Controlador

- É importante destacar que, mesmo habitando o mesmo pacote, são as classes controladoras de interação que requisitam serviços da camada de lógica de negócio.

- Em outras palavras, são as classes controladoras de interação que disparam a lógica de aplicação.

- Outro ponto a ser destacado é que a figura considera que apenas os objetos controladores de interação se comunicam com objetos da lógica de negócio.

- Contudo, essa abordagem pode ser flexibilizada. É bastante comum que os próprios objetos de visão (cih) se comuniquem com objetos da lógica de negócio, mas apenas para montar os objetos gráficos e não para requisitar serviços.

O Padrão Modelo-Visão-Controlador

- O projeto da camada de IU é fortemente relacionado ao projeto da lógica de aplicação e ambos são apoiados pelo modelo de casos de uso.

- Assim, sobretudo quando o padrão Camada de Serviço é adotado, é uma boa estratégia elaborar um único diagrama de classes envolvendo as classes do cgt e as classes do ciu.

- Quando essa estratégia é adotada, é bastante importante usar as notações especializadas da UML para classes controladoras de interação e classes de interface, de modo a destacar os tipos das diferentes classes no diagrama.

O Processo de Projeto da Interface com o Usuário

- O projeto de interface com o usuário envolve não apenas aspectos de tecnologia (facilidades para interfaces gráficas, multimídia, etc.), mas principalmente o estudo das pessoas.

- Quem é o usuário? Como ele aprende a interagir com um novo sistema? Como ele interpreta uma informação produzida pelo sistema? O que ele espera do sistema?

- Estas são apenas algumas das muitas questões que devem ser levantadas durante o projeto da interface com o usuário.

O Processo de Projeto da Interface com o Usuário

- O princípio básico para o projeto de IU é o seguinte: Conheça o usuário e as tarefas. Assim, é importante ter modelos tanto do usuário quanto das tarefas que os mesmos vão desempenhar no sistema.

- Os modelos de casos de uso têm precisamente essas informações. As tarefas são os casos de uso; os usuários são agrupados em atores. Assim, o modelo de casos de uso é a base principal para o projeto da IU.

O Processo de Projeto da Interface com o Usuário

- De maneira geral, o projeto de interfaces com o usuário envolve os seguintes passos:

- 1. **Definir as funcionalidades acessíveis a partir da IU do sistema:** este passo visa estabelecer como as tarefas que as pessoas fazem normalmente no contexto do sistema (casos de uso) podem ser mapeadas em um conjunto similar (mas não necessariamente idêntico) de tarefas a serem implementadas no contexto da interface humano-computador.

- Deve-se definir, também, o fluxo global da interação, aglutinando os diversos casos de uso na forma de um ou mais aplicativos.

O Processo de Projeto da Interface com o Usuário

- 2. **Estabelecer o perfil dos usuários**: A interface do sistema deve ser adequada ao nível de habilidade dos seus futuros usuários.

- Assim, é necessário estabelecer o perfil desses potenciais usuários e classificá-los segundo aspectos como nível de habilidade, nível na organização e membros em diferentes grupos.

- Uma classificação possível considera os seguintes grupos:

- • *Usuário Novato*: não conhece a dinâmica de interação requerida para utilizar a interface eficientemente (conhecimento sintático; p.ex., não sabe como atingir uma funcionalidade desejada) e conhece pouco a semântica da aplicação, isto é, entende pouco as funções e objetivos do sistema, ou não sabe bem como usar computadores em geral;

nemo

O Processo de Projeto da Interface com o Usuário

- • *Usuário conhecedor, mas esporádico*: possui um conhecimento razoável da semântica da aplicação, mas tem relativamente pouca lembrança das informações sintáticas necessárias para utilizar a interface;

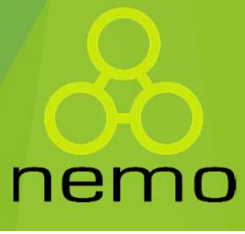
- • *Usuário conhecedor e frequente*: possui bom conhecimento tanto sintático quanto semântico e busca atalhos e modos abreviados de interação.

O Processo de Projeto da Interface com o Usuário

- 3. **Considerar princípios gerais de projeto de IU**, tais como facilidades de ajuda, mensagens de erro, tipos de comandos, entre outros, de modo a prover uma IU adequada para os perfis de usuários estabelecidos.

- Este passo pode ser visto como a definição de quais táticas de usabilidade devem ser aplicadas no projeto da IU de um sistema.

- 4. **Construir protótipos** e, em última instância, implementar as interfaces do sistema, usando ferramentas apropriadas. A prototipagem abre espaço para uma abordagem iterativa de projeto de interface com o usuário.



O Processo de Projeto da Interface com o Usuário

- Nessa abordagem, utilizando diversos protótipos construídos iterativamente, o usuário avalia a adequação da IU para o uso em seus processos de negócio.

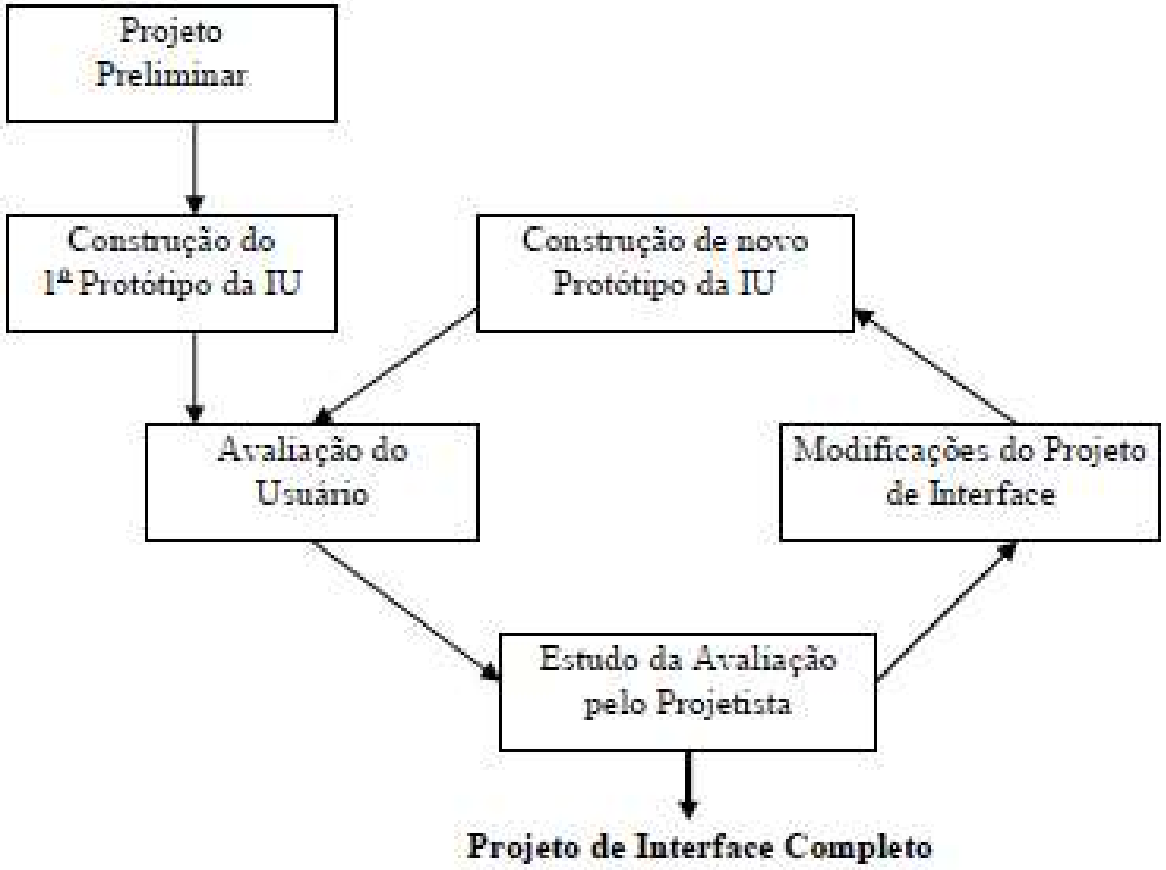
- Em uma abordagem de prototipagem, é imprescindível o uso de ferramentas para a construção de interfaces, provendo facilidades para manipulação de janelas, menus, botões, comandos etc.

O Processo de Projeto da Interface com o Usuário

- 5. **Avaliar o resultado**: A construção de protótipos é feita com a participação de apenas uns poucos usuários. Uma vez que se obtém o projeto considerado completo da IU, idealmente, deve-se avaliar seu resultado para o conjunto (ou uma amostra significativa) de usuários.

- Para tal, pode ser útil coletar dados qualitativos e quantitativos por meio, p.ex., de questionários distribuídos a uma amostra significativa de usuários.

O Processo de Projeto da Interface com o Usuário



nemo

Projeto de Visão

- *A porção do sistema que lida com a visão da interface com o usuário deve ser mantida tão independente e separada do restante da arquitetura do software quanto possível.*

- Aspectos de interface com o usuário provavelmente serão alvo de alterações ao longo da vida do sistema e essas alterações devem ter um impacto mínimo nas demais partes do sistema.

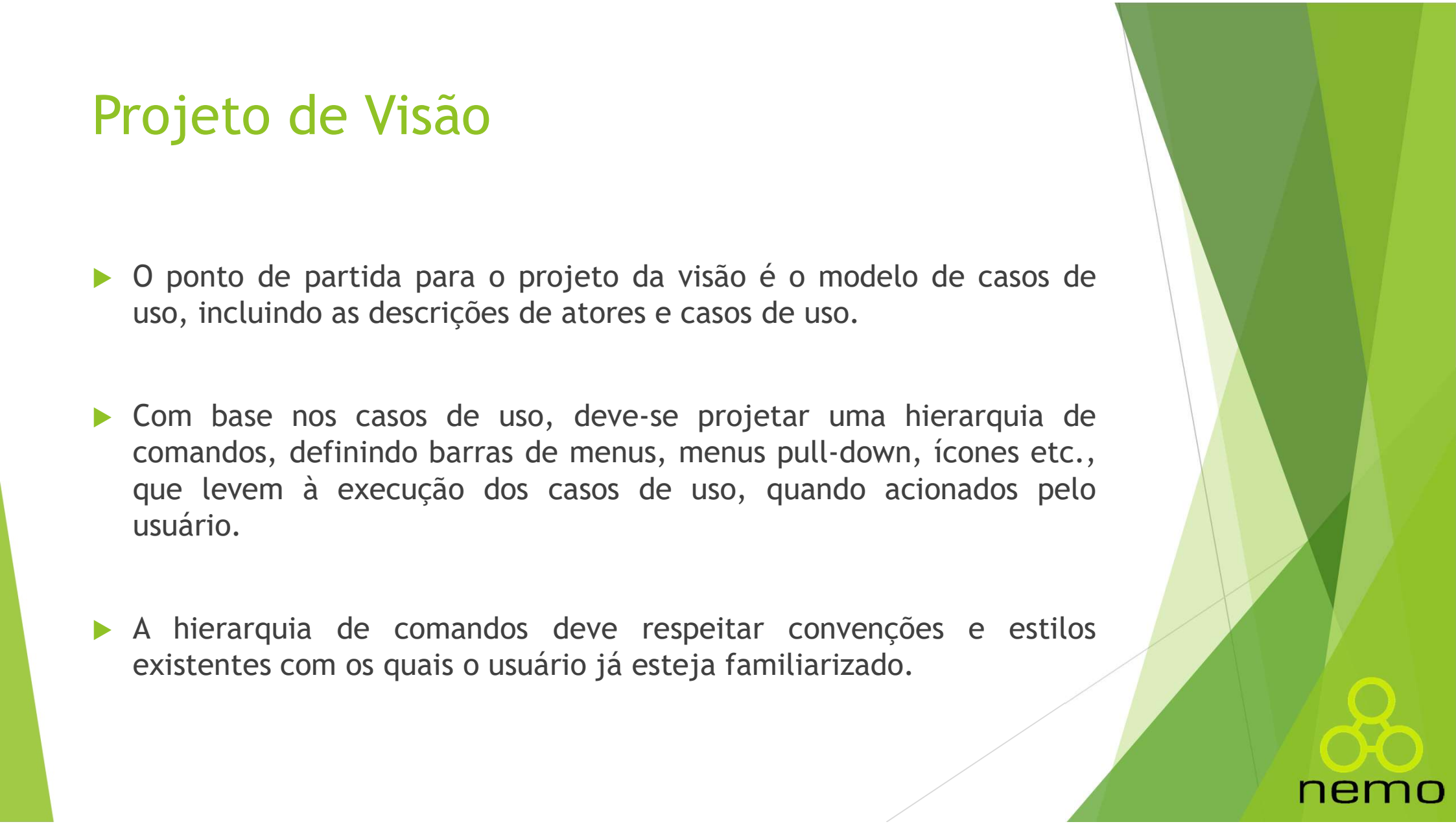
- A visão trata do projeto da interação humano-computador, definindo formato de janelas, formulários, relatórios, entre outros. Durante o projeto da visão, é muito útil construir protótipos, de modo a apoiar a escolha e o desenvolvimento dos mecanismos de interação a serem usados.

Projeto de Visão

- O ponto de partida para o projeto da visão é o modelo de casos de uso, incluindo as descrições de atores e casos de uso.

- Com base nos casos de uso, deve-se projetar uma hierarquia de comandos, definindo barras de menus, menus pull-down, ícones etc., que levem à execução dos casos de uso, quando acionados pelo usuário.

- A hierarquia de comandos deve respeitar convenções e estilos existentes com os quais o usuário já esteja familiarizado.



Projeto de Visão

- Note que a hierarquia de comandos é, de fato, um meio de apresentar ao usuário as várias funcionalidades disponíveis no sistema.

- Assim, a hierarquia de comandos deve permitir o acesso aos casos de uso do sistema. Uma vez definida a hierarquia de comandos, as interações detalhadas entre o usuário e o sistema devem ser projetadas.

- Neste momento, é útil observar atentamente as táticas de usabilidade.

Projeto de Visão

- Normalmente, não é necessário projetar as classes básicas de interfaces gráficas com o usuário.
- Existem vários ambientes de desenvolvimento de interfaces oferecendo classes reutilizáveis (janelas, ícones, botões etc.) e, portanto, basta especializar essas classes e instanciar os objetos que possuem as características apropriadas para o problema em questão.
- Ainda assim, é muito útil desenvolver classes gerais de visão visando à uniformidade da apresentação e ao reúso.
- Essas classes podem ser organizadas em hierarquias de classes, de modo que, no projeto de um sistema específico, o projeto da sua visão seja realizado por meio da especialização de classes de visão já existentes ao invés de ter de se compor todas as classes de visão a partir de componentes básicos de interface providos pelo ambiente de desenvolvimento.

Projeto de Visão

- O projeto detalhado das classes de visão do sistema pode ser mais facilmente compreendido pela visualização das próprias telas, sendo pouco útil indicar os atributos de uma classe de visão em um diagrama de classes.

- Para compreender a estrutura interna de uma classe de visão, é mais indicado prover o layout correspondente.

- Entretanto, ainda é útil mostrar em um diagrama de classes as classes de visão e as suas relações de especialização com outras classes de visão e, sobretudo, as suas associações com as classes controladoras de interação, o que permite capturar como se dará efetivamente o tratamento da interação humano-computador do sistema.

Táticas de Usabilidade

- Diversas táticas relativas à usabilidade podem ser aplicadas durante o projeto de IU.

- Algumas dessas táticas incluem:

- (i) provisão de facilidades de ajuda,
- (ii) apresentação de mensagens de aviso e de erro significativas,
- (iii) oferta de diferentes tipos de comandos, adequados para diferentes perfis de usuário e
- (iv) visão do estado do sistema quando em processamento.

Facilidade de Ajuda

- Ajuda é fundamental para os usuários, sobretudo para os novatos ou para aqueles conhecedores do problema, mas usuários esporádicos do sistema.

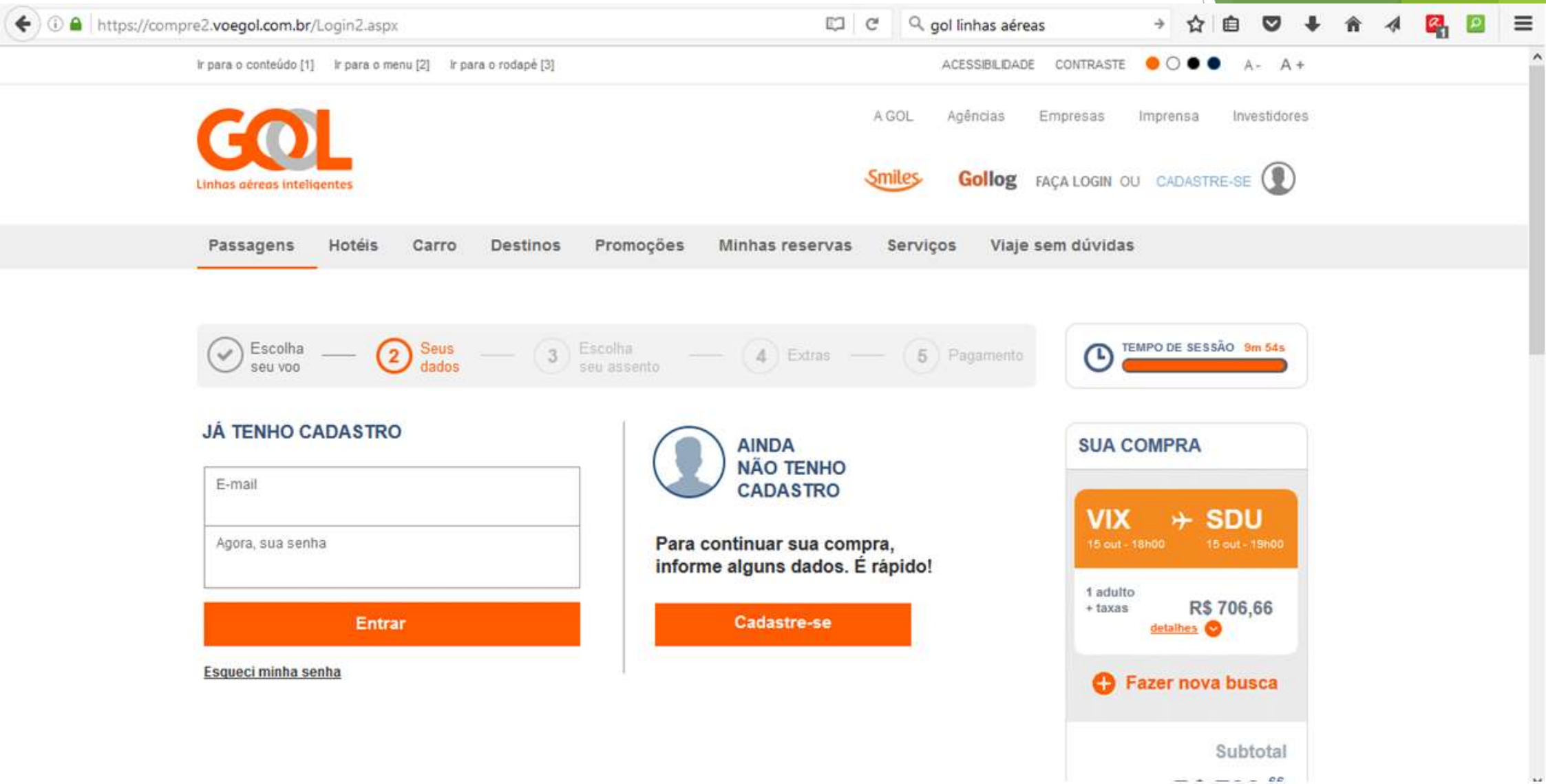
- Para projetar adequadamente facilidades de ajuda, é necessário definir vários aspectos, dentre eles:

- (i) quando a ajuda estará disponível e para que funções do sistema ou campos da IU;
- (ii) como ativar (botão, tecla de função, menu etc.);
- (iii) onde apresentar (janela separada, local fixo da tela etc.);
- (iv) como retornar à interação normal (botão, tecla de função);
- (v) como estruturar a informação (estrutura plana, hierárquica, hipertexto).

Facilidade de Ajuda

- Em relação às funções do sistema, é importante prover ajuda ao usuário para que este esclareça o objetivo de uma funcionalidade do sistema, qual a sua sequência de passos (fluxo do caso de uso) e em que passo ele está no momento (e, por conseguinte, como chegou até ali).

- P.ex., em um sistema de compra de passagens aéreas, é importante identificar para o usuário quais os passos a serem realizados e em que passo o usuário se encontra em um dado momento.


https://compre2.voegol.com.br/Login2.aspx
gol linhas aéreas
Ir para o conteúdo [1]
Ir para o menu [2]
Ir para o rodapé [3]
ACESSIBILIDADE
CONTRASTE
A-
A+
GOL
Linhas aéreas inteligentes
A GOL
Agências
Empresas
Imprensa
Investidores
Smiles
Gollog
FAÇA LOGIN OU CADASTRE-SE
Passagens
Hotéis
Carro
Destinos
Promoções
Minhas reservas
Serviços
Viaje sem dúvidas
Escolha seu voo
2 Seus dados
3 Escolha seu assento
4 Extras
5 Pagamento
TEMPO DE SESSÃO 9m 54s
JÁ TENHO CADASTRO
E-mail
Agora, sua senha
Entrar
Esqueci minha senha
AINDA NÃO TENHO CADASTRO
Para continuar sua compra, informe alguns dados. É rápido!
Cadastre-se
SUA COMPRA
VIX
15 out - 18h00
SDU
15 out - 19h00
1 adulto + taxas
R$ 706,66
detalhes
Fazer nova busca
Subtotal

Facilidade de Ajuda

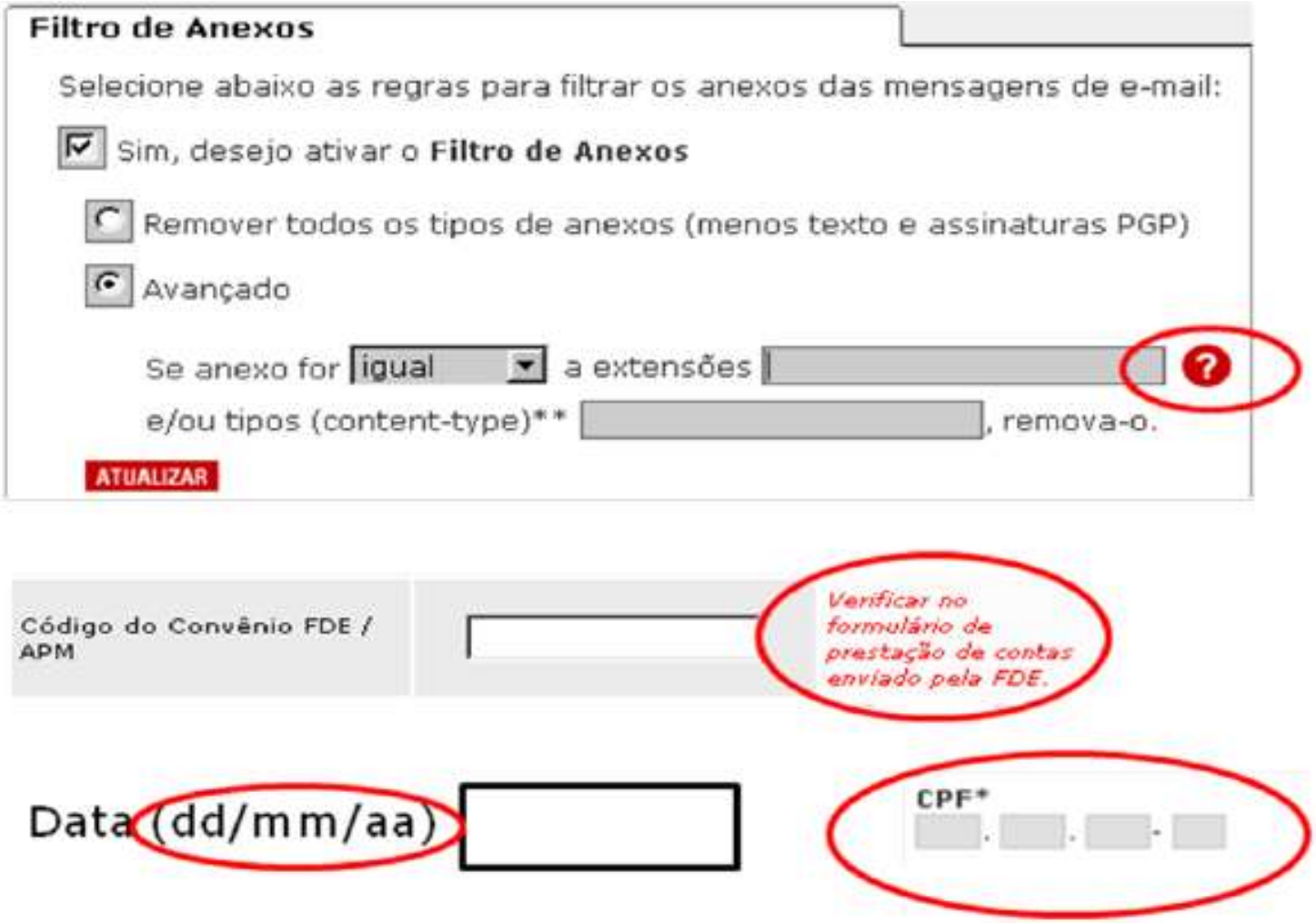
- Em relação aos campos da IU, é importante informar o significado de campos não óbvios para o usuário, como preenchê-los e, eventualmente, onde obter a informação para seu preenchimento.

- P.ex., em um sistema de registro de empregados para emissão de guia de recolhimento de impostos, é importante dizer o que significa o campo NIT, como preenchê-lo e onde obter essa informação.

- Além disso, para campos que são apresentados no mundo real geralmente formatados (p.ex., datas, CPF etc.) é importante dizer como os mesmos devem ser preenchidos.

- P.ex., em sistemas que requerem o preenchimento de CPF, deve-se informar a forma de preenchimento (apenas números ou com uso de separadores).

Facilidade de Ajuda



nemo

Facilidade de Ajuda

- Em relação a como ativar as facilidades de ajuda, há diferentes maneiras de se prover acesso às facilidades de ajuda.

- Quando a ajuda é pontual (p.ex., o significado de um campo ou a sua formatação), a informação pode estar prontamente disponível na própria interface ou pode-se disponibilizar um recurso de ajuda (p.ex., um botão ou um campo sensível à presença do mouse) próximo ao elemento para o qual se deseja prover a ajuda.

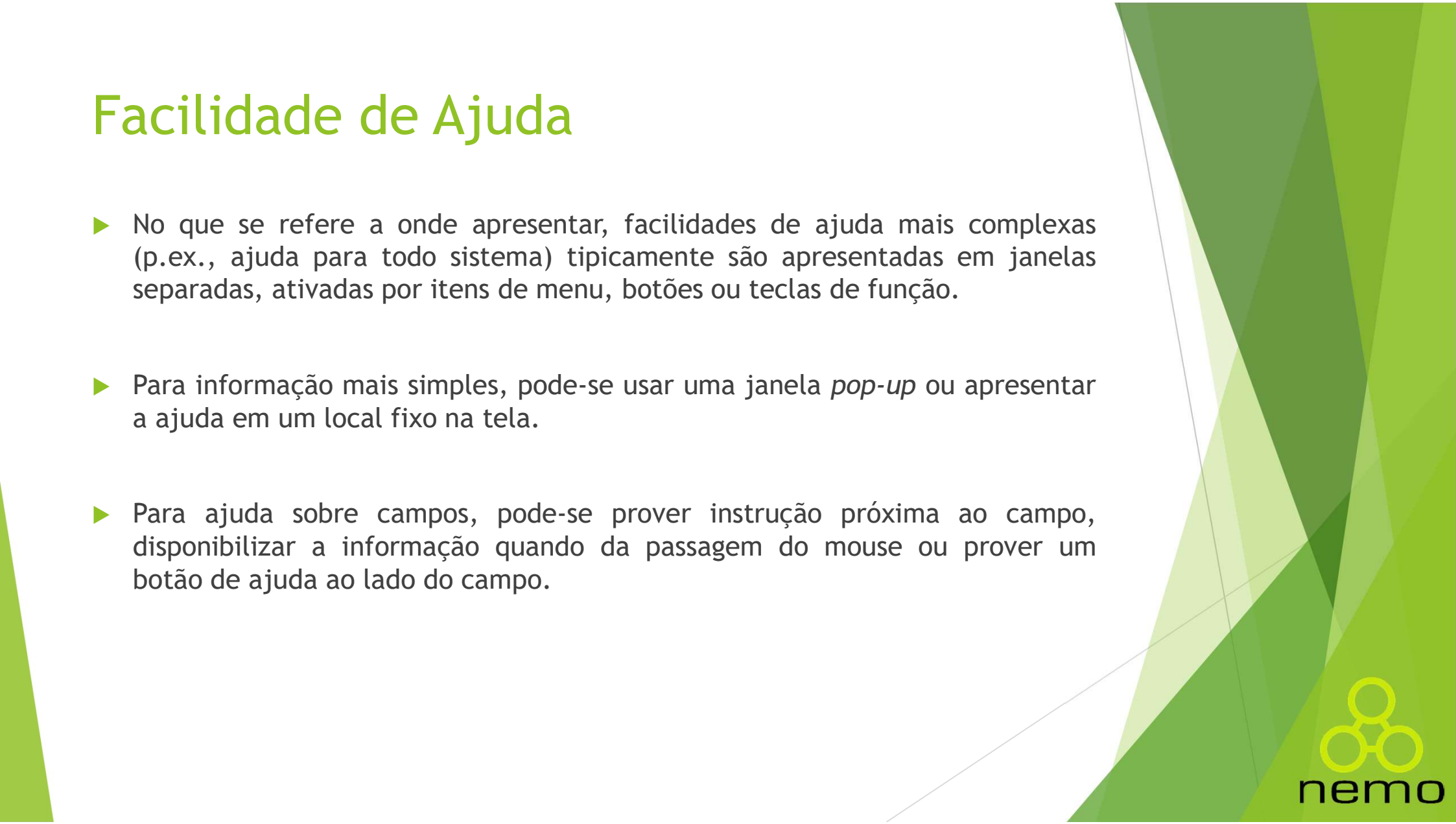
- Para facilidades de ajuda mais sofisticadas, que envolvem muitas informações (tal como um help completo do sistema) normalmente usam-se botões, itens de menu e teclas de função (p.ex., F1) ou uma combinação desses recursos.

Facilidade de Ajuda

- No que se refere a onde apresentar, facilidades de ajuda mais complexas (p.ex., ajuda para todo sistema) tipicamente são apresentadas em janelas separadas, ativadas por itens de menu, botões ou teclas de função.

- Para informação mais simples, pode-se usar uma janela *pop-up* ou apresentar a ajuda em um local fixo na tela.

- Para ajuda sobre campos, pode-se prover instrução próxima ao campo, disponibilizar a informação quando da passagem do mouse ou prover um botão de ajuda ao lado do campo.



Facilidade de Ajuda

- Quando uma facilidade de ajuda é provida em outro objeto de interface (p.ex., janela separada ou janela pop-up), é preciso definir como retornar à interação normal.

- Janelas pop-up tipicamente têm botões para fechá-las. Janelas separadas podem usar os mecanismos típicos de encerramento. Outra opção é fazer uso de teclas de função (p.ex., ESC para retornar à interação normal).

- Por fim, para facilidades de ajuda com conteúdo complexo, é importante definir como estruturar a informação. Algumas opções são estruturas planas, hierárquicas e hipertextos, as quais podem ser combinadas.

Mensagens de Erro e Avisos

- Mais até do que a ajuda, as mensagens de erro e avisos são fundamentais para uma boa interação humano-computador. Ao definir mensagens de erro e avisos considere as seguintes diretrizes:

- • Descreva o problema com um vocabulário passível de entendimento pelo usuário.
- • Sempre que possível, proveja assistência para recuperar um erro.
- • Quando for o caso, indique as consequências negativas de uma ação.
- • Para facilitar a percepção da mensagem por parte do usuário, pode ser útil que a mesma seja acompanhada de uma dica visual (p.ex., destacar o campo em que o preenchimento apresentou problema) ou sonora.

nemo

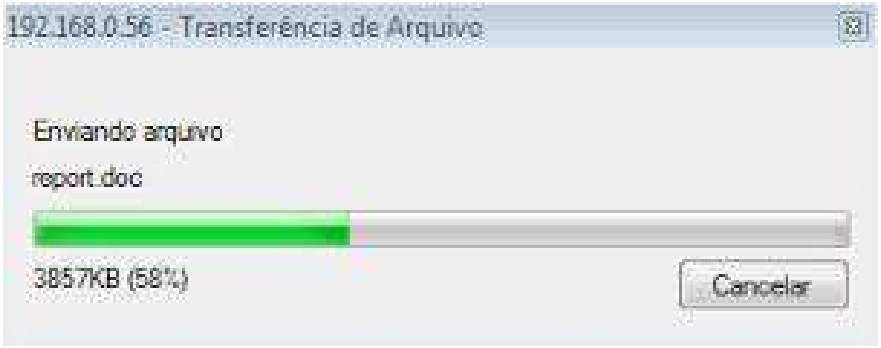
Tipos de Comandos

- Diferentes grupos de usuários têm diferentes necessidades de interação.
- Em muitas situações é útil prover aos usuários mais de uma forma de interação. Nestes casos, é necessário definir e avaliar:
- • Quais funções terão mais de uma forma de interação (p.ex., um item de menu e um comando correspondente);
- • Qual será a forma do comando. Controle de sequência (p.ex., ^Q), teclas de função (p.ex., F1) e comandos digitados são opções tipicamente combinadas com itens de menu;
- • Quão difícil é aprender e lembrar o comando;
- • Os padrões a serem adotados: Devem ser adotados padrões uniformes para todo sistema. Esses padrões devem estar em conformidade com outros padrões, tais como o definido pelo sistema operacional e por produtos de software tipicamente utilizados pelos usuários.

nemo

Progresso do Processamento

- É importante mostrar o progresso do processamento para os usuários, principalmente para eventos com tempo de resposta longo ou com grande variação de tempos de resposta.



nemo

Diretrizes para o Projeto da Visão

- Levando-se em conta princípios gerais de projeto de IU, algumas orientações adicionais devem ser consideradas, dentre elas:

- • Seja consistente. Use formatos consistentes para seleção de menus, entrada de comandos, apresentação de dados etc.
- • Ofereça retorno significativo ao usuário.
- • Peça confirmação para ações destrutivas, tais como ações para apagar ou sobrepor informações ou para terminar a seção corrente do aplicativo.
- • Permita reversão da maioria das ações (função Desfazer).

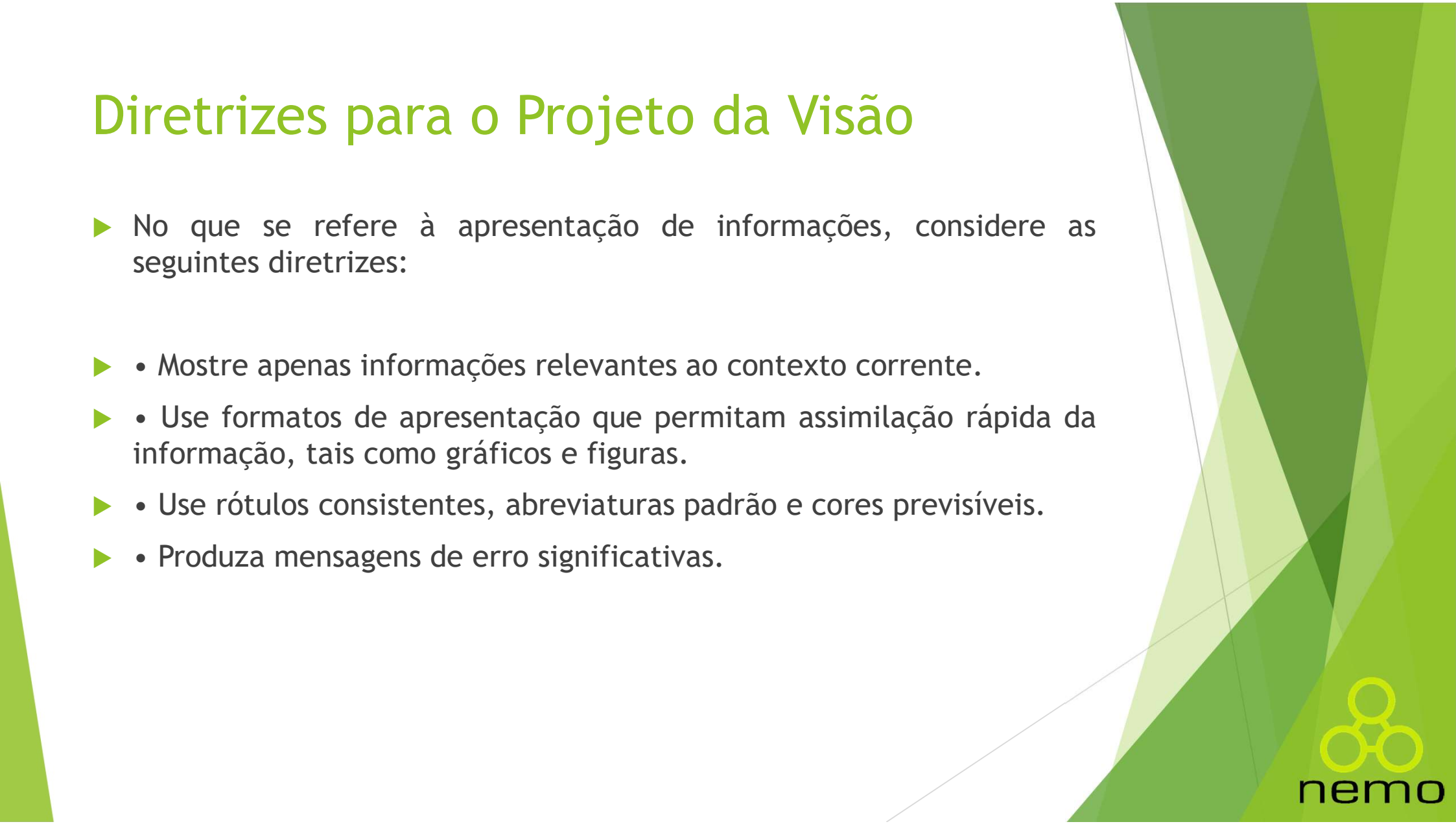
Diretrizes para o Projeto da Visão

- • Reduza a quantidade de informação que precisa ser memorizada entre ações.
- • Busque eficiência no diálogo (movimentação, teclas a serem apertadas).
- • Trate possíveis erros do usuário. O sistema deve se proteger de erros, casuais ou não, provocados pelo usuário.
- • Classifique atividades por função e organize geograficamente a tela de acordo. Menus do tipo pull-down são uma boa opção.
- • Proveja facilidades de ajuda sensíveis ao contexto.
- • Use verbos de ação simples ou frases curtas para nomear funções e comandos.

Diretrizes para o Projeto da Visão

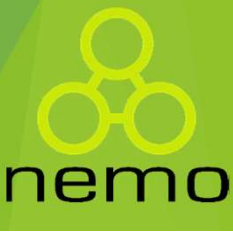
- No que se refere à apresentação de informações, considere as seguintes diretrizes:

- • Mostre apenas informações relevantes ao contexto corrente.
- • Use formatos de apresentação que permitam assimilação rápida da informação, tais como gráficos e figuras.
- • Use rótulos consistentes, abreviaturas padrão e cores previsíveis.
- • Produza mensagens de erro significativas.



Diretrizes para o Projeto da Visão

- • Projete adequadamente o layout de informações textuais. Leve em consideração o bom uso de letras maiúsculas e minúsculas, identação, agrupamento de informações etc.
- • Separe diferentes tipos de informação. Painéis podem ser usados para este fim.
- • Use formas de representação análogas às do mundo real para facilitar a assimilação da informação. Para tal considere o uso de figuras, cores etc.



Diretrizes para o Projeto da Visão

- No que se refere à entrada de dados, considere as seguintes diretrizes:

- • Minimize o número de ações de entrada requeridas e possíveis erros. Para tal considere a seleção de dados a partir de um conjunto pré-definido de valores de entrada, o uso de valores default etc.

- • Mantenha consistência entre apresentação e entrada de dados; ou seja, mantenha as mesmas características visuais, dentre elas tamanho do texto, cor e localização.

Diretrizes para o Projeto da Visão

- • Permita ao usuário customizar a entrada para seu uso, quando possível, dando-lhe liberdade para definir comandos customizados, dispensar algumas mensagens de aviso e verificações de ações, dentre outros.
- • Flexibilize a interação, permitindo afiná-la ao modo de entrada preferido do usuário (comandos, botões, plug-and-play, digitação etc.).
- • Desative comandos inapropriados para o contexto das ações correntes.
- • Proveja ajuda significativa para assistir as ações de entrada de dados.
- • Nunca requeira que o usuário entre com uma informação que possa ser adquirida automaticamente pelo sistema ou computada por ele.

Projeto do Controle de Interação

- O projeto do Controle de Interação visa definir as classes responsáveis por controlar a interação (ativar/desativar objetos de visão) e enviar requisições para os objetos da Lógica de Negócio.

- Em sistemas rodando em plataforma desktop, deve haver pelo menos uma classe controladora, dita classe controladora de sistema, representando o sistema como um todo.

- Os objetos dessa classe representam as várias sessões (execuções) do sistema.

Projeto do Controle de Interação

- Neste contexto, é necessário levar em conta quantos executáveis devem ser gerados para o sistema.

- Se mais do que um executável for necessário, cada executável terá de dar origem a uma classe controladora. Esta, contudo, é apenas uma abordagem possível.

- A interação entre controladores e objetos da lógica de negócio se dá de maneiras distintas em função do padrão arquitetônico adotado no projeto da lógica de negócio.

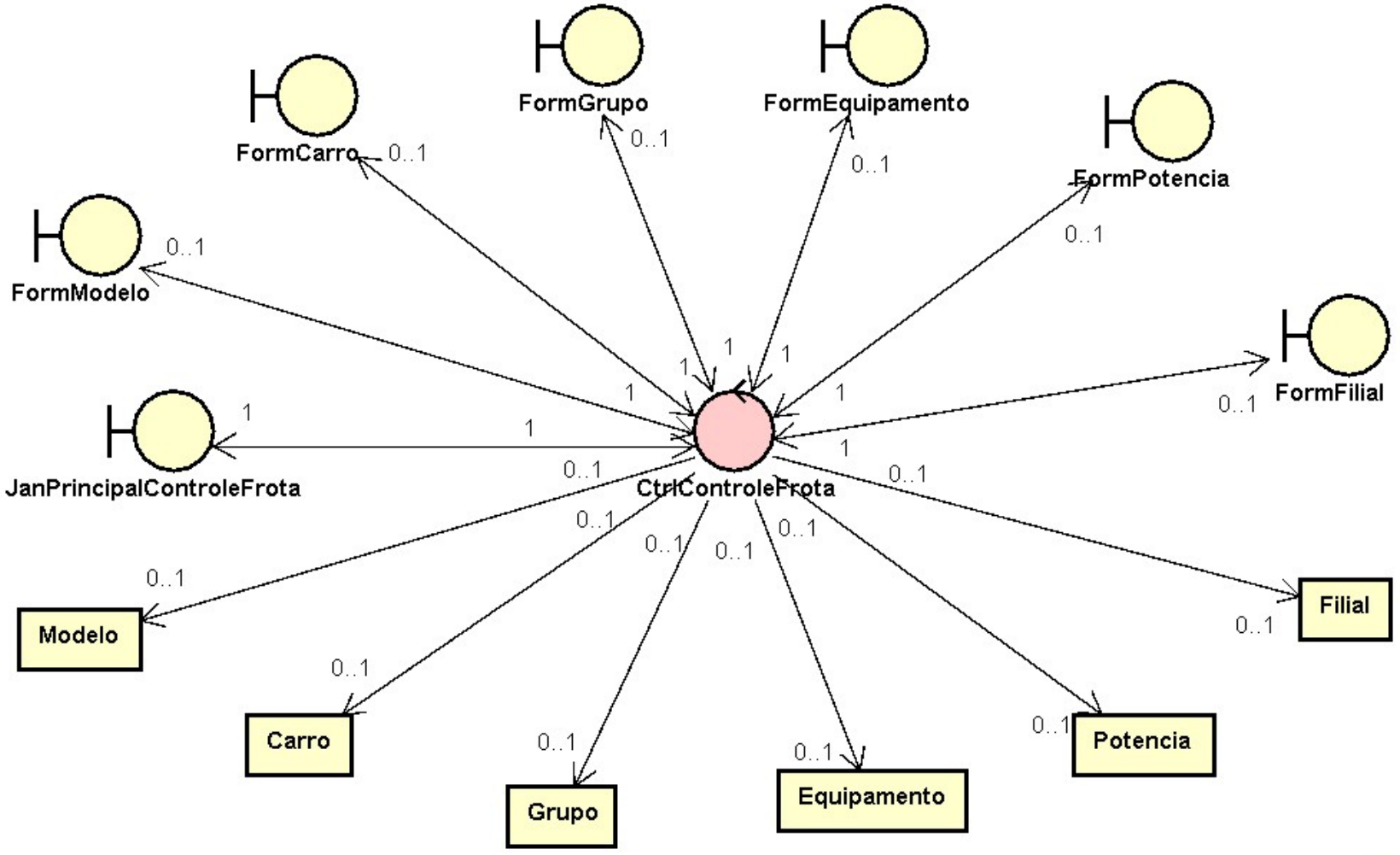
Projeto do Controle de Interação

- Quando o padrão Modelo de Domínio é adotado, os controladores de interação estão associados diretamente com os objetos do domínio do problema (cdp).

- Uma abordagem interessante consiste em relacionar a classe controladora do sistema (ou uma classe controladora responsável por parte da funcionalidade do sistema) com os objetos independentes do cdp.

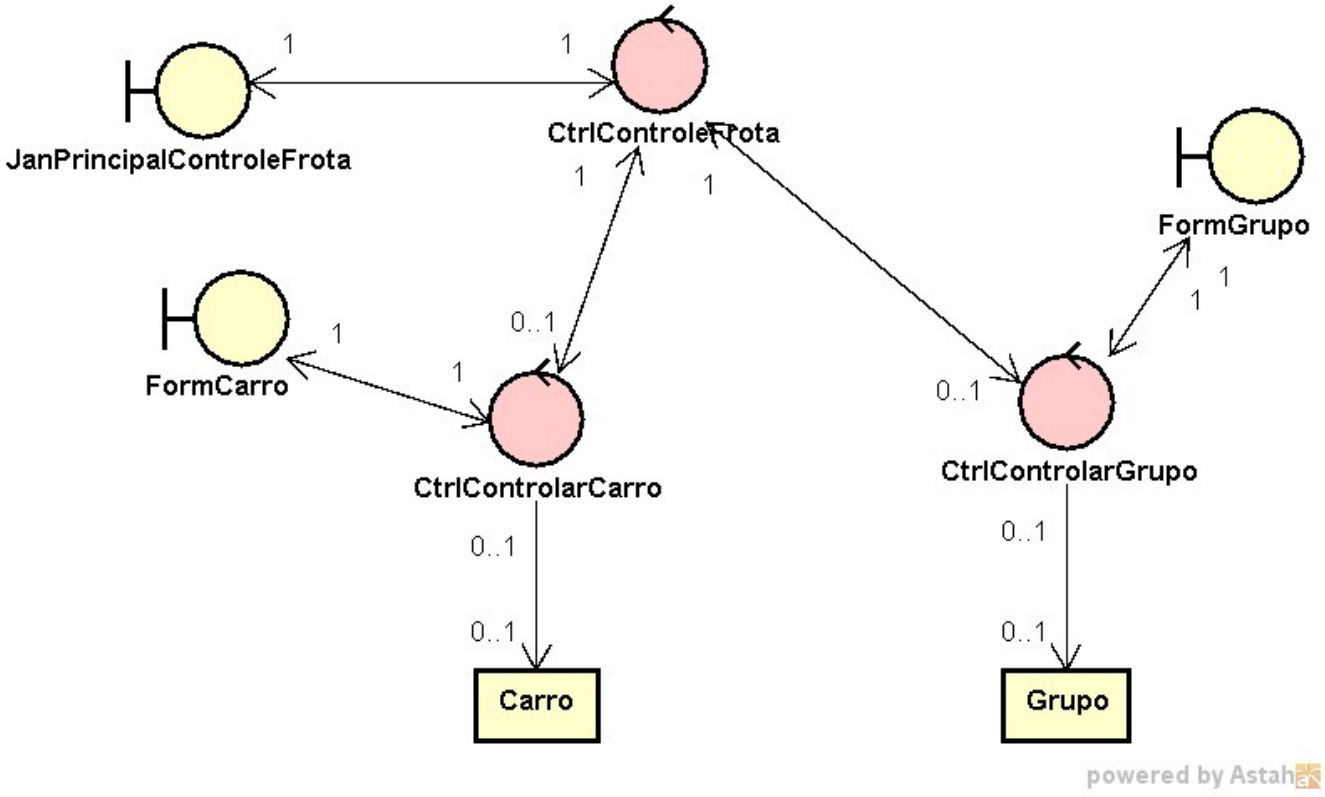
- Vale destacar que o conceito de objeto independente pode ser analisado no contexto do subsistema, i.e., se um objeto não depende de outros objetos no contexto do subsistema em questão, então ele pode ser considerado independente neste subsistema e estar ligado ao controlador.

Projeto do Controle de Interação



nemo

Projeto do Controle de Interação



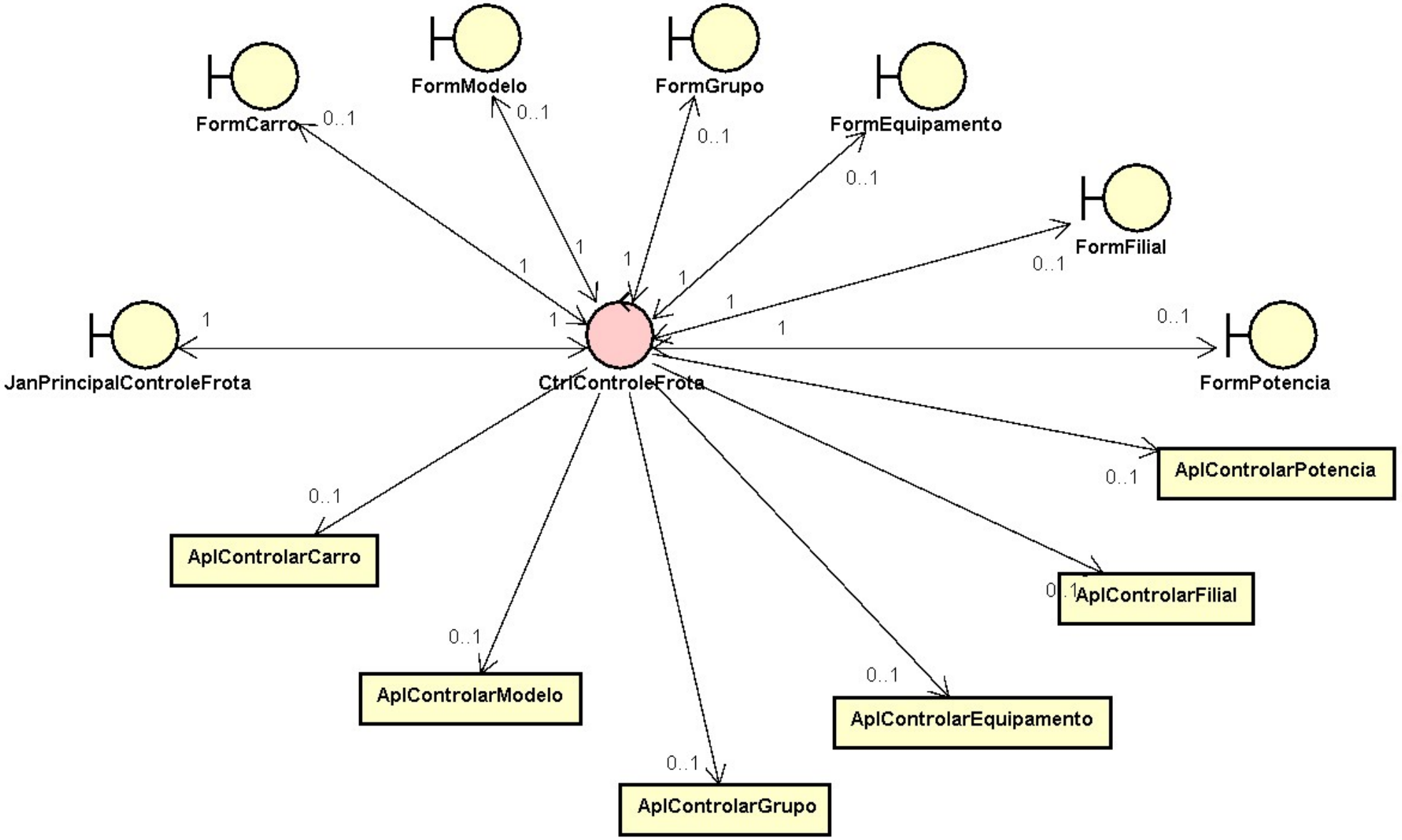
nemo

Projeto do Controle de Interação

- Quando o padrão Camada de Serviço é adotado, os controladores estão associados a objetos gerenciadores de tarefas (cgt).

- Analogamente ao projeto do Componente de Gerência de Tarefas (cgt), é possível definir um número arbitrário de controladores de interação.

- Uma opção é definir um único controlador para todo o sistema (controlador de sistema), o qual fica responsável por gerenciar a interação de toda aplicação.



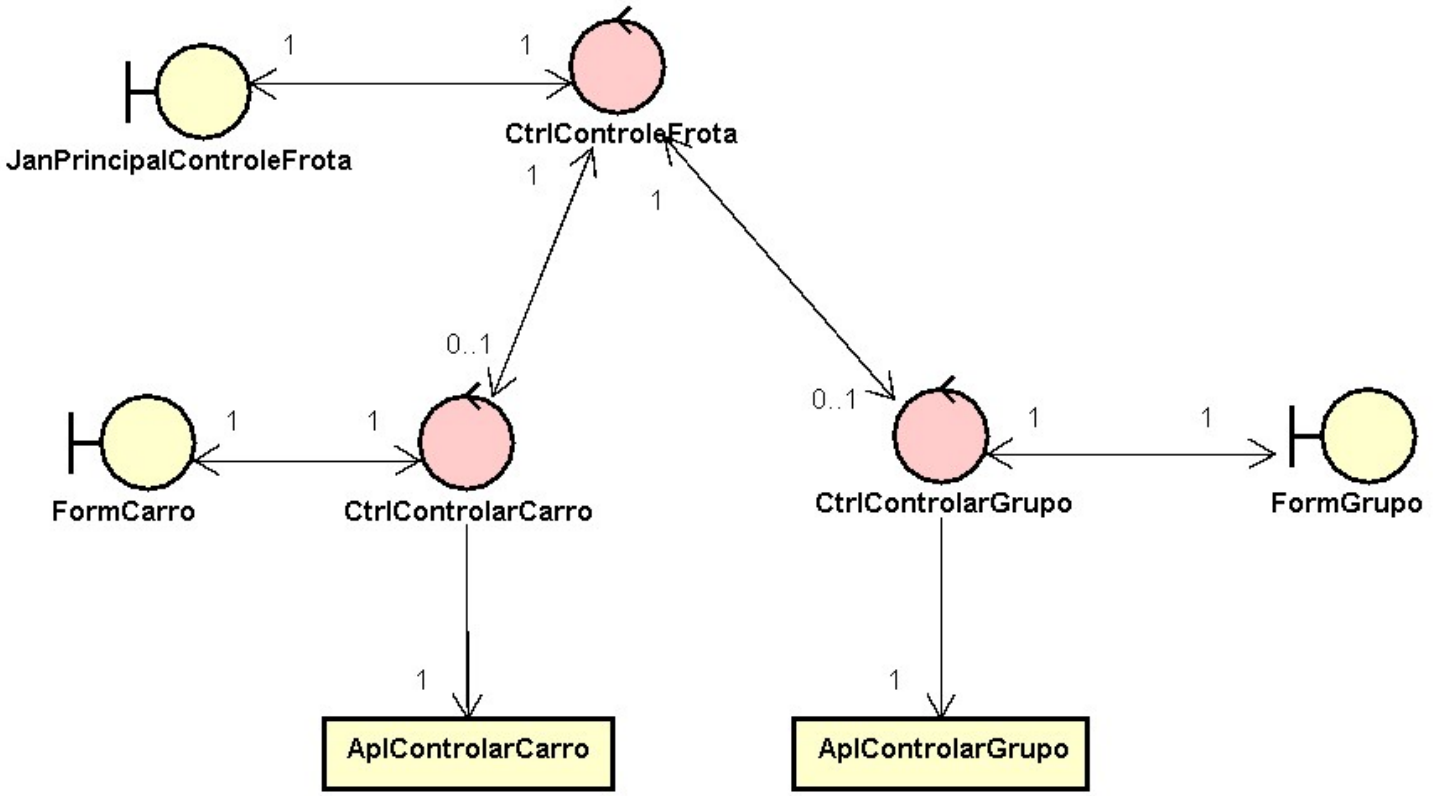
Projeto do Controle de Interação

- Assim como ocorre no projeto do cgt, essa opção tende a ser inadequada, pois a classe controladora de sistema pode ficar muito complexa.

- Uma opção no sentido de favorecer a manutenibilidade consiste em definir um controlador de interação para cada caso de uso.

- Em aplicações desktop, adicionalmente, é necessário ter ainda, pelo menos, uma classe controladora de sistema (ou uma classe controladora para cada executável).

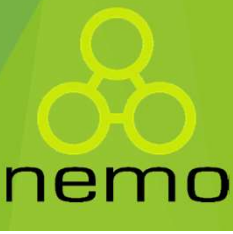
Projeto do Controle de Interação



Design Patterns no Projeto da Interface com o Usuário

- Alguns padrões de projeto *(design patterns)* são bastante utilizados para tratar problemas recorrentes no projeto da IU, dentre eles os padrões Decorador, Observador e Comando.

- O padrão Decorador (*Decorator*) anexa responsabilidades adicionais a um objeto dinamicamente, permitindo estender sua funcionalidade.



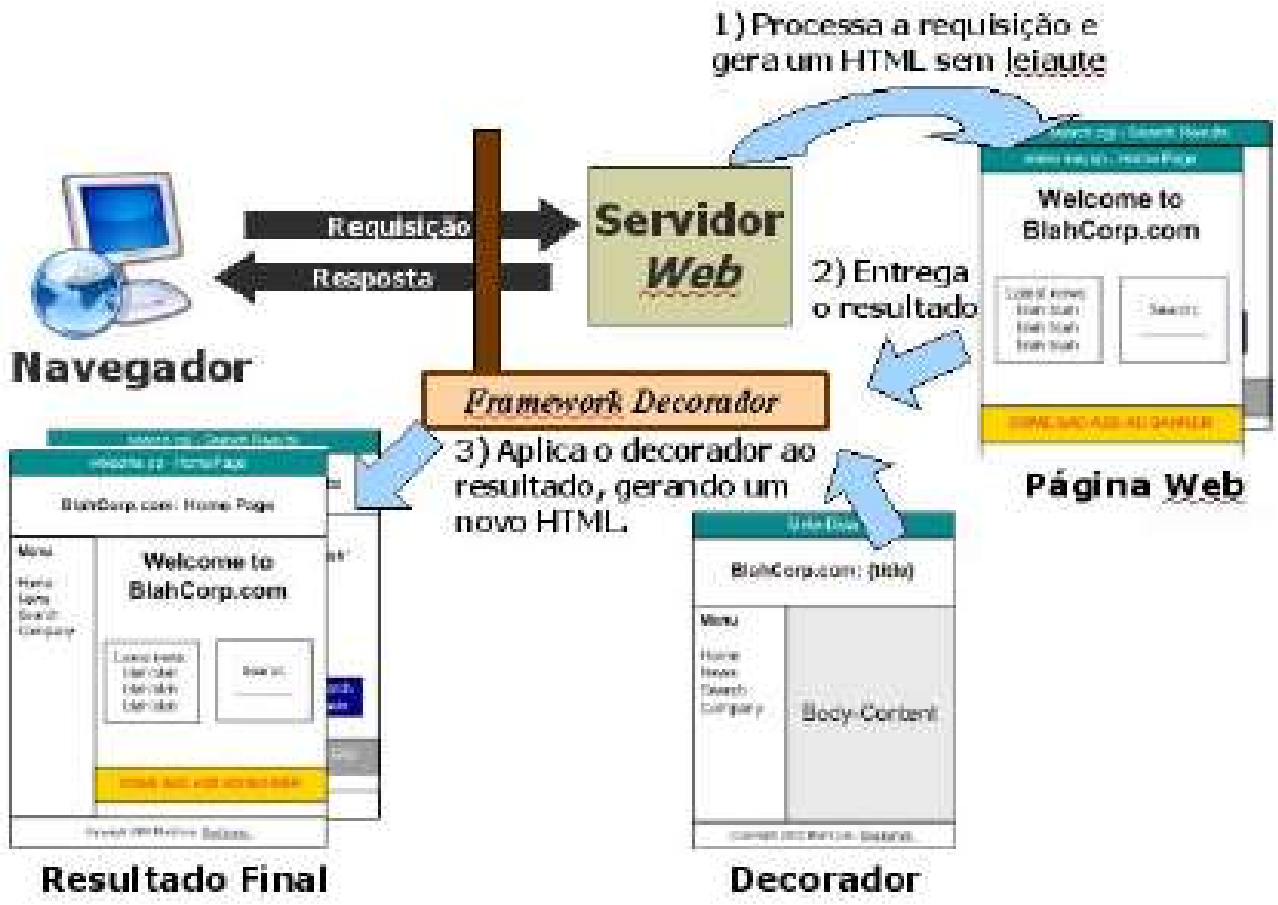
Design Patterns no Projeto da Interface com o Usuário

Padrão Decorador

- Ele é utilizado por frameworks decoradores de interface para automatizar a tarefa de manter uma aplicação Web tradicional com a mesma aparência, ou seja, cabeçalho, rodapé, barra de navegação, esquema de cores e demais elementos gráficos de layout integrados num mesmo projeto de apresentação.

- Esse tipo de framework funciona segundo o padrão de projeto Decorador, se posicionando como um filtro entre uma requisição do cliente e um servidor Web.

Design Patterns no Projeto da Interface com o Usuário



nemo

Design Patterns no Projeto da Interface com o Usuário

Padrão Observador

- O padrão Observador (*Observer*) define uma dependência um-para-muitos entre objetos, de modo que, quando um objeto muda de estado, todos os seus dependentes são notificados e atualizados automaticamente.

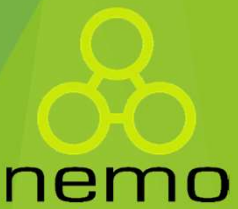
- Ele é uma boa opção para separar aspectos de apresentação dos respectivos dados da aplicação, permitindo o uso de múltiplas representações.

Design Patterns no Projeto da Interface com o Usuário

Padrão Observador

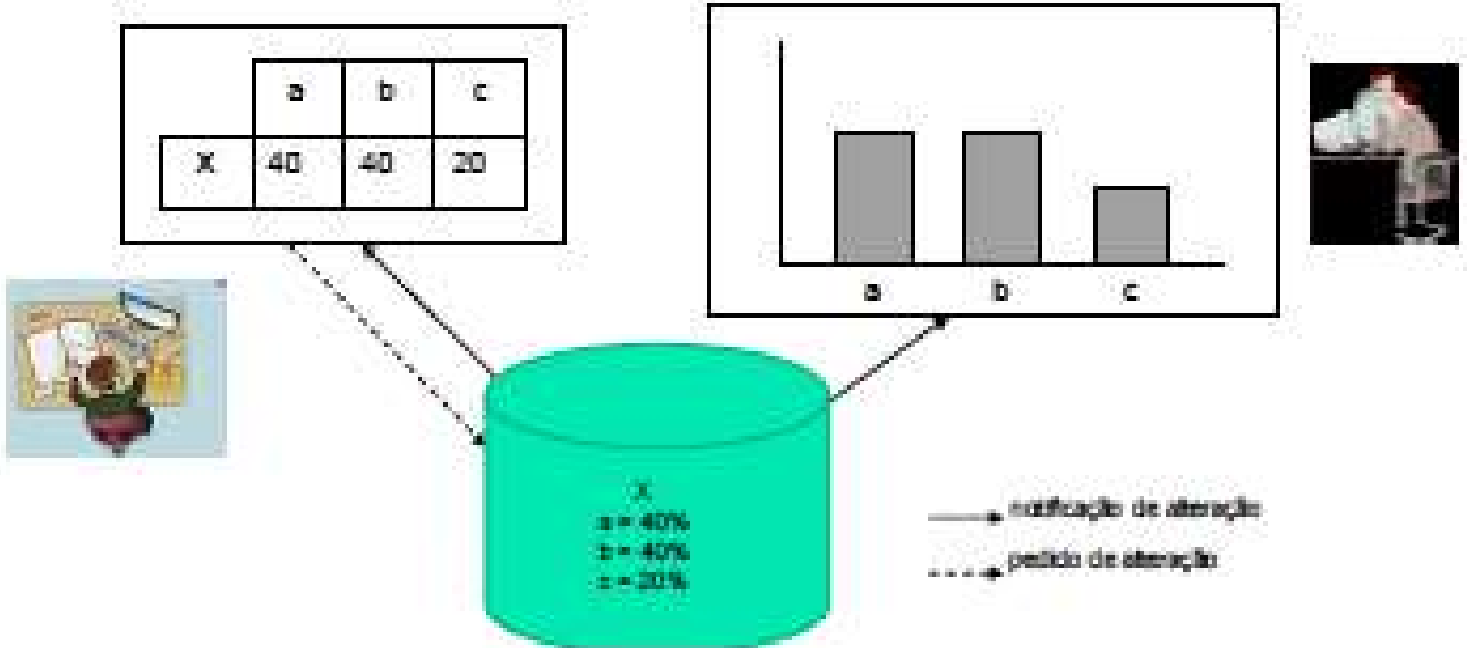
- Por exemplo, os mesmos dados estatísticos podem ser apresentados em formato de um gráfico de barras ou em uma planilha, usando apresentações diferentes. O gráfico de barras e a planilha devem ser independentes.

- Contudo, eles têm de se comportar consistentemente, isto é, quando um usuário alterar a informação na planilha, o gráfico de barras deve refletir a troca imediatamente e vice-versa.



Design Patterns no Projeto da Interface com o Usuário

- Na solução proposta pelo padrão observador, a apresentação atua como um observador do domínio do problema: sempre que o domínio do problema é alterado, ele envia um evento e a apresentação atualiza a informação.



Design Patterns no Projeto da Interface com o Usuário

Padrão Comando

- Finalmente, o padrão Comando (*Command*) encapsula uma requisição como um objeto, permitindo parametrizar clientes com diferentes requisições e desfazer operações.

- Em interfaces com o usuário, objetos gráficos (p.ex., botões e menus), quando acionados, devem disparam requisições para a execução de funcionalidades do sistema.

Design Patterns no Projeto da Interface com o Usuário

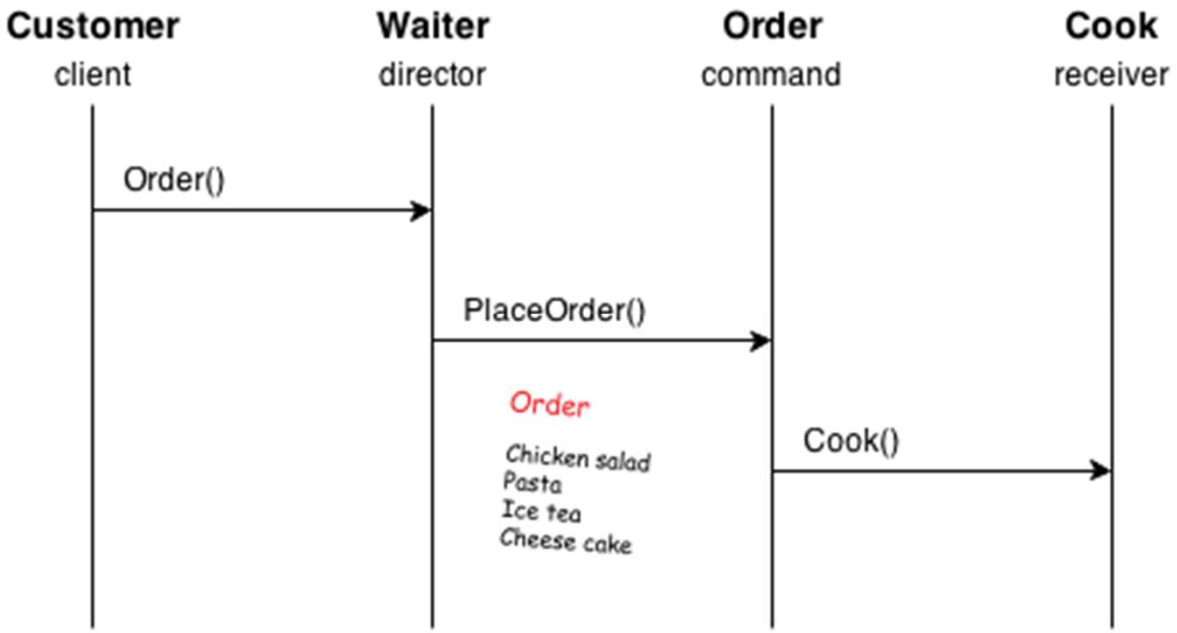
- Entretanto, essas requisições não podem ser implementadas diretamente nos objetos gráficos, pois somente a aplicação deve saber efetivamente o que deve ser feito.

- O padrão Comando permite que os objetos gráficos façam requisições de objetos não especificados, tornando a requisição em si um objeto.

- Esse objeto pode ser armazenado e repassado como qualquer outro objeto.

Design Patterns no Projeto da Interface com o Usuário

- **Exemplo Padrão Comando**



Fonte: https://sourcemaking.com/design_patterns/command

nemo

Design Patterns no Projeto da Interface com o Usuário

- Além dos *design patterns* propostos em (GAMMA et al., 1995), os quais tratam de problemas mais gerais, aplicáveis também ao projeto da IU, há padrões específicos dedicados ao projeto da visão.

- Por exemplo, Nudelman (2013) apresenta um catálogo de padrões (e alguns anti-padrões) de projeto de visão para a plataforma Android. Há padrões para experiência de boas-vindas, tela principal, busca, ordenação e filtragem, entradas de dados, formulários e navegação, dentre outros.

- Os padrões são descritos com foco em smartfones, contendo uma descrição geral do padrão, informações sobre como ele funciona, exemplo, quando e onde usar, por que usar, outros usos e variação para aplicativos para tablets.

That’s all Folks!