

Projeto de *Software*



Cruzeiro do Sul Virtual
Educação a distância

Material Teórico



Desenvolvimento dos *Mockups*
para *GUI – Graphic User Interface*

Responsável pelo Conteúdo:

Prof. Me. Afonso M. Luiz R. Pavão

Revisão Textual:

Prof.^a Esp. Kelciane da Rocha Campos

UNIDADE

Desenvolvimento dos *Mockups* para *GUI – Graphic User Interface*



- Desenvolvimento de *Mockups*;
- Interface Gráfica para Usuários;
- *Wireframe, Protótipo e Mockup*;
- Mapa Navegacional do *Software*.



OBJETIVOS DE APRENDIZADO

- Identificar em protótipos a percepção das necessidades do projeto de *software* para atender ao negócio;
- Conhecer e produzir artefatos necessários para o desenvolvimento de projeto do *software* e sua implementação.



Orientações de estudo

Para que o conteúdo desta Disciplina seja bem aproveitado e haja maior aplicabilidade na sua formação acadêmica e atuação profissional, siga algumas recomendações básicas:

Determine um horário fixo para estudar.

Mantenha o foco! Evite se distrair com as redes sociais.

Procure manter contato com seus colegas e tutores para trocar ideias! Isso amplia a aprendizagem.

Seja original! Nunca plágie trabalhos.

Aproveite as indicações de Material Complementar.

Conserve seu material e local de estudos sempre organizados.

Não se esqueça de se alimentar e de se manter hidratado.

Assim:

- ✓ Organize seus estudos de maneira que passem a fazer parte da sua rotina. Por exemplo, você poderá determinar um dia e horário fixos como seu “momento do estudo”;
- ✓ Procure se alimentar e se hidratar quando for estudar; lembre-se de que uma alimentação saudável pode proporcionar melhor aproveitamento do estudo;
- ✓ No material de cada Unidade, há leituras indicadas e, entre elas, artigos científicos, livros, vídeos e sites para aprofundar os conhecimentos adquiridos ao longo da Unidade. Além disso, você também encontrará sugestões de conteúdo extra no item **Material Complementar**, que ampliarão sua interpretação e auxiliarão no pleno entendimento dos temas abordados;
- ✓ Após o contato com o conteúdo proposto, participe dos debates mediados em fóruns de discussão, pois irão auxiliar a verificar o quanto você absorveu de conhecimento, além de propiciar o contato com seus colegas e tutores, o que se apresenta como rico espaço de troca de ideias e de aprendizagem.

Desenvolvimento de Mockups

O início da resposta para a pergunta acima começa com outras perguntas: os usuários sempre conseguem ter a visão geral de um *software*? Os desenvolvedores facilitam essa visão ao usuário? E os desenvolvedores têm essa visão?

Lembre-se de acessar a bibliografia recomendada para complementar este conteúdo bastante específico.

Para que um produto de *software* seja bem-sucedido, deve apresentar boa usabilidade (medida qualitativa da facilidade e eficiência com a qual um ser humano consegue empregar as funções e os recursos oferecidos pelo produto de alta tecnologia).

Pressman (2011) afirma que:

o projeto de interfaces do usuário se inicia pela identificação do usuário, das tarefas e dos requisitos do ambiente. Uma vez que as tarefas de usuário tenham sido identificadas, os cenários são criados e passa-se a definir um conjunto de objetos e ações. Esses cenários formam a base para a criação do *layout* da tela, que representa o projeto gráfico e o posicionamento de ícones, a definição de texto descritivo na tela, a especificação e seus títulos, bem como a especificação de itens de menu principais e secundários. São usadas ferramentas para criar protótipos e, por fim, implementar o modelo de projeto e o resultado é avaliado em termos de qualidade.

Uma vez identificadas as necessidades dos usuários (os requisitos do *software*), o processo de elaboração do projeto de interface torna-se bem mais simples, pois se faz uma análise mais detalhada antes de se pensar na interface.

A construção da interface em geral se inicia com a criação de um protótipo que permite que cenários de uso possam ser avaliados. À medida que o processo de projeto iterativo prossegue, um *kit* de ferramentas de interfaces do usuário pode ser usado para completar a construção da interface.

A validação da interface se concentra: (1) na capacidade da interface implementar corretamente todas as tarefas de usuário, levando em conta todas as variações de tarefas, bem como atender a todos os requisitos gerais dos usuários; (2) no grau de facilidade de uso e aprendizado da interface; e (3) na aceitação dos usuários da interface como sendo uma ferramenta útil no seu trabalho.

Quando a análise de interface tiver sido completada, todas as tarefas (ou objetos e ações) requeridas pelo usuário final já foram também identificadas detalhadamente, e a atividade de projeto de interface pode ser iniciada.

O projeto de interface, tal qual a engenharia de *software*, é um processo iterativo e cada etapa desse projeto ocorre uma série de vezes, elaborando e refinando as informações desenvolvidas na etapa anterior.

Os autores sugerem combinar as seguintes etapas:

- usar informações desenvolvidas durante a análise de interfaces e definir objetos e ações (operações) da interface;
- definir eventos (ações de usuário) que provocarão a mudança de estado da interface do usuário e modelar esse comportamento;
- representar cada estado da interface como realmente aparecerá para o usuário final;
- indicar como o usuário interpreta o estado do sistema, com base em informações fornecidas através da interface.

Pressman (2011) lembra ainda que “à medida que o projeto de uma interface do usuário evolui, quatro questões de projeto comuns quase sempre vêm à tona: o tempo de resposta do sistema, os recursos de ajuda ao usuário, as informações de tratamento de erros e a atribuição de nomes a comandos”.

Pressman também sugere um fluxo de trabalho para a implementação de interfaces para Web, o qual eu recomendo que seja utilizado em qualquer tipo de *software*:

1. Revise as informações contidas no modelo de requisitos e refina conforme for necessário;
2. Desenvolva um esboço do *layout* para as interfaces;
3. Mapeie os objetivos do usuário em ações de interface específicas;
4. Defina um conjunto de tarefas de usuário que está associado a cada ação;
5. Elabore as imagens de tela *storyboard* para cada ação de interface;
6. Faça o refinamento do *layout* de interface e dos *storyboards* usando entrada do projeto estético;
7. Identifique objetos de interface com o usuário necessários para implementar a interface;
8. Desenvolva uma representação procedural da interação do usuário com a interface;
9. Desenvolva uma representação comportamental da interface;
10. Descreva o *layout* da interface para cada estado; e
11. Faça o refinamento revisando o modelo de projeto de interface.

Tais instruções irão facilitar a documentação visando desenvolver a interface. Veja a Figura 1.

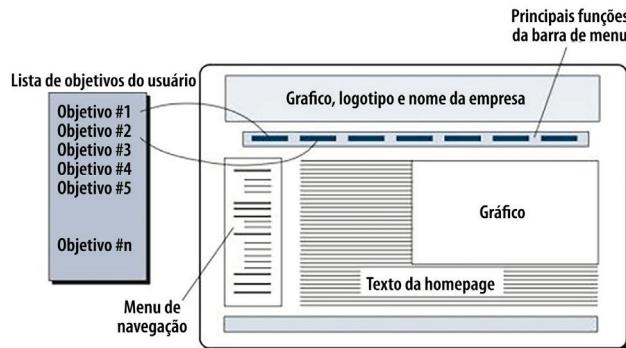


Figura 1 – Mapeamento de objetivos

Fonte: Adaptado de PRESSMAN (2011)

É sempre salutar revisar as interfaces dos usuários. A Figura 2 representa o ciclo de avaliação de uma interface.

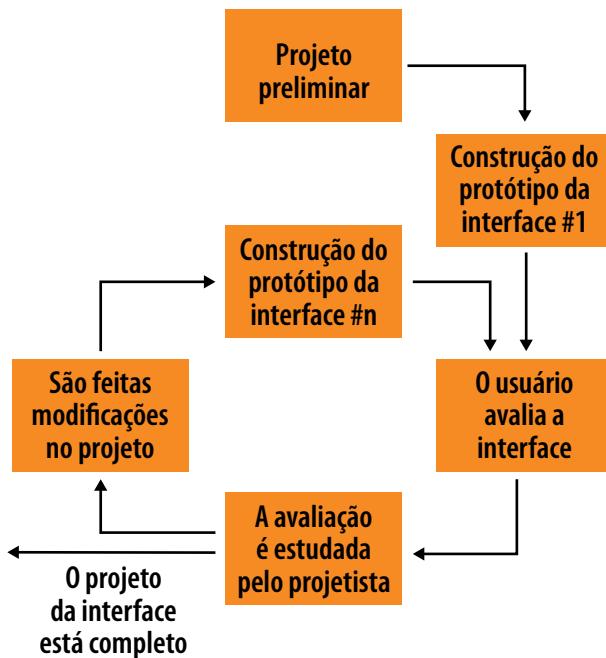


Figura 2 – Ciclo de avaliação de interfaces

Interface Gráfica para Usuários

Conforme a evolução tecnológica ocorria, aumentava a necessidade de comunicação do ser humano com seu meio ambiente. Isto possibilitou aprimorar sua forma de comunicar – com quem quer que seja, de modo direto ou indireto.

Analogamente, essa necessidade capacitou o ser humano a criar e a desenvolver diversos dispositivos eletrônicos para a comunicação, como, por exemplo, os celulares. O termo interação humano-computador ou homem-máquina - IHC foi incorporado no dia a dia das empresas que desenvolviam as tecnologias de informação e comunicação.

A *Graphical User Interface* – GUI – interface gráfica do usuário é um tipo de interface que possibilita interagir com dispositivos digitais com meios gráficos, tais como indicadores visuais ou botões (ícones).

Tudo começou com pesquisas em Stanford durante os anos 60. Na década de 70, as pesquisas foram refinadas, surgindo a interface do computador da Xerox, contendo *widgets* gráficos com janelas, menus, caixas de opções e de seleção, e ícones. Tinha um dispositivo ponteiro, que posteriormente deu origem ao *mouse*.

A Xerox criou essa facilidade, mas foi a *Apple* que revolucionou, transformando isto num produto, em substituição aos comandos digitados, com o uso do *mouse* ou do teclado, permitindo a seleção e a manipulação de símbolos conhecidos como *widgets*.

Com Steve Jobs, a partir de 1979, os projetistas do *Lisa* e do *Macintosh* permaneceram utilizando e desenvolvendo aquelas ideias da Xerox. Assim, foi sucesso a apresentação da interface gráfica: arquivos pareciam folhas de papel, diretórios lembravam pastas do antigo arquivo de aço e surgiam os tais utilitários: calculadora, bloco de notas, despertador e lixeira.

Entre criações, cópias, ações judiciais, surgimento e desaparecimento de interfaces gráficas, *Apple*, *Atari*, *Commodore* e *Microsoft*, o *Unix* vai ganhando espaço e deixa seguidores com o *Linux*.

A plataforma *Personal Computer* popularizou os computadores e surgiu um grande mercado para explorar as interfaces de uso fácil. Em paralelo, a evolução de tecnologias gráficas com mais cores e placas de vídeos mais rápidas contribuíram para o surgimento de sistemas com melhor usabilidade.

Esse ambiente gráfico é desenvolvido para facilitar e garantir melhores práticas na utilização de computadores. Apenas para exemplificar, para o sistema *Windows* há o ambiente padrão, mas nas versões *Windows 7* e *Vista*, a interface era conhecida como *Windows Aero*. Para a *GNU/Linux*, há inúmeros ambientes gráficos, como os padrões *KDE*, *Gnome*, *BlackBox*, *Xfce*, *LXDE*, por exemplo.

Observe a estrutura das interfaces nas figuras que se seguem.

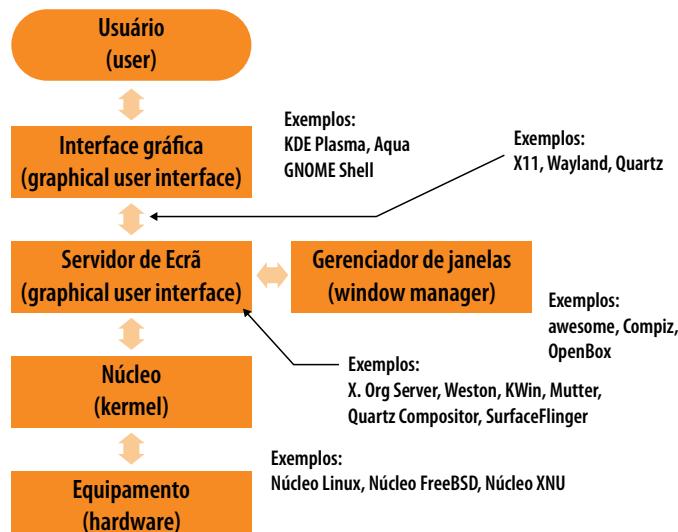
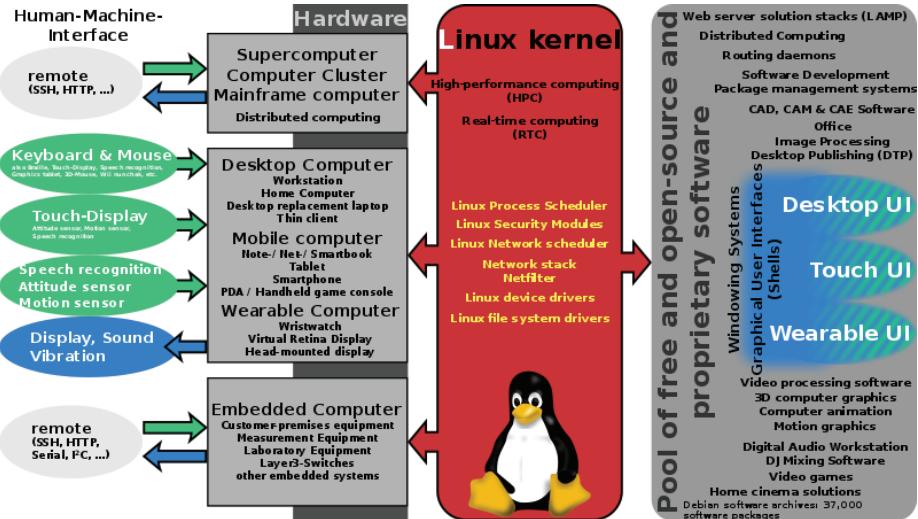


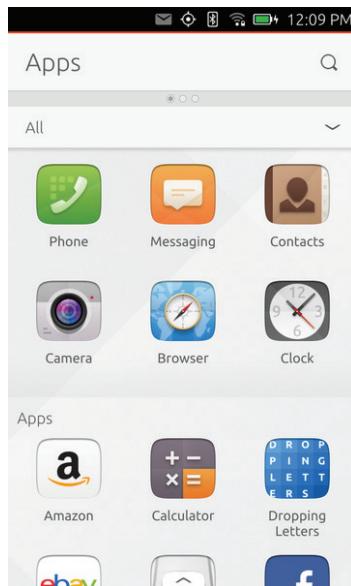
Figura 3 – Camadas da interface gráfica

Figura 4 – Proximidades entre *Linux e Kernel*

Fonte: Wikimedia Commons

A interface gráfica combina tecnologias e dispositivos e disponibiliza uma plataforma simples para o utilizador interagir. Em computadores pessoais, isto é conhecido como o WIMP, que possui janelas, ícones, menus e ponteiros.

O mouse controla a posição do cursor e apresenta informação organizada em janelas representadas por ícones. Os comandos são compilados por menus e acionados por um ponteiro. O programa que gerencia a janela ajuda na interação entre janelas, aplicações e os dispositivos de *hardware*.

Figura 5 – Interface gráfica do sistema operacional *Ubuntu Touch*

Fonte: Wikimedia Commons



Interface gráfica do utilizador, disponível em: <http://bit.ly/2RYGg0J>

Graphical User Interface (GUI), disponível em: <http://bit.ly/2S2r2f>

Wireframe, Protótipo e Mockup

Há diferenças entre *wireframe*, protótipo e *mockup*???

Um **wireframe** é a representatividade com baixa fidelidade do *design* de um projeto e visa demonstrar **o que** ou quais serão os grupos de conteúdo, **onde**, ou seja, qual é a estrutura da informação e **como** será a visualização e a interação da interface com o usuário.

Sua característica é que deve ser criado rapidamente, independentemente do tempo gasto na comunicação do time de projeto na solução de problemas da arquitetura ou da usabilidade do *software*.

Não tem um *design* bem elaborado, mas é considerado seu esqueleto (ou *grid*), pois tem todas as partes importantes do projeto.

Por isso é mais recomendável usá-lo como parte da documentação de projeto, para algumas situações diárias do projeto, para se obter rápidas ideias do projeto ou, ainda, no contexto do processo de *design*.

Já o **protótipo**, muitas vezes confundido com o **wireframe**, visa criar protótipos relativamente fiéis do projeto final. Usualmente, serve para simular a interação do usuário e para testar o conteúdo e as interações.

Sua utilização tem por finalidade facilitar os testes com simulação das interfaces finais e auxiliar a documentar o projeto.

Por outro lado, o **mockup** é uma representação estática de média e alta fidelidade do projeto. Em bom português, significa amostra ou apresentação. Normalmente, está próximo do *design* final e ajuda na representação da estrutura da informação, na visualização do conteúdo, na demonstração das principais funcionalidades e no estímulo aos envolvidos a revisar o *design* final do projeto.

Seu uso no desenvolvimento de *software* é para se criar interfaces de usuário que possam mostrar sua semelhança com o *software* concluído. Pode variar desde layout de tela muito simples até interfaces desenvolvidas com o apoio de uma ferramenta de desenvolvimento de *software*.

Sendo um modelo em escala ou em tamanho real do dispositivo, é usado para ensino, demonstração, avaliação ou promoção. Se tiver ao menos parte das funcionalidades de um sistema e permite testar o projeto, o *mockup* pode ser considerado um protótipo.

É também utilizado para criar testes de unidade (chamados de objetos *Mock*), pois facilita o teste de uma parte do *software*, geralmente uma unidade, sem precisar ter todos os demais módulos – principalmente quando as funções que serão simuladas forem difíceis de se obter, ou devido a computações complexas ou não determinísticas, como a leitura de um sensor, por exemplo.

Seu uso também contribui para se vender a ideia do produto antes de este estar pronto, obter-se *feedback* de usuários e para auxílio na documentação do projeto. Além disso – e exatamente por isso, ajuda na redução dos custos e na diminuição de erros na produção do *software*.

Se o *mockup* for físico, pode ser chamado de **maquete**, embora este termo se aplique mais à arquitetura.

Observe a tabela resumida.

Tabela 1

Representação	Fidelidade	Custo	Uso	Características
Wireframe	Baixa	\$	Comunicação rápida e documentação	Rascunho e representação preta e branca da interface
Mockup	Média a alta	\$\$	Obter <i>feedback</i> e vender a ideia do produto	Visualização estática
Protótipo	Média a alta	\$\$\$	Teste de usabilidade e “esqueleto” reutilizável para desenvolver a interface	Interativo

Após os esboços (chamados também de *sketches*), estes devem ser posteriormente elaborados com o uso de ferramentas apropriadas, para melhor visualização e qualidade de apresentação ao usuário final, para sugestões e validação.

Há alguns *softwares* para esta finalidade, alguns pagos e outros não, os quais destacamos (em ordem alfabética):

Tabela 2

<i>Adobe Illustrator</i>	<i>Adobe Photoshop</i>	<i>Adobe XD</i>
<i>Affinity Designer</i>	<i>Antetype</i>	<i>Axure</i>
<i>Balsamiq</i>	<i>CSS3</i>	<i>Figma</i>
<i>Framer X</i>	<i>Framer (Classic)</i>	<i>GIMP</i>
<i>Gravit Designer</i>	<i>Hotjar</i>	<i>HTML</i>
<i>InVision Studio</i>	<i>Just in mind</i>	<i>Marvel</i>
<i>MockFlow</i>	<i>Mockplus</i>	<i>Net Beans</i>
<i>OmniGraffle</i>	<i>Origami Studio</i>	<i>PHP</i>
<i>Principle</i>	<i>Proto.io</i>	<i>Sketch</i>
<i>Studio</i>	<i>UXPin</i>	<i>UxTools</i>
<i>Vectr</i>		

As telas – *layouts* ou protótipos facilitam ao desenvolvedor ter a visão geral do sistema e permitem que o usuário compreenda e identifique as funcionalidades que especificou.

A participação do(s) usuário(s) no processo de desenvolvimento – e, em particular, nas metodologias ágeis – assegura que as etapas de desenvolvimento sejam melhor controladas e, portanto, possibilitem mais qualidade ao *software*.

Outro aspecto muito importante é que esse envolvimento dos usuários irá permitir que qualquer tipo de alteração, melhoria ou correção possa ser executada antes dos esforços de codificação.

A percepção visual da interface humano-computador, embora muitas vezes seja relegada a segundo plano, é também objeto de avaliação por parte da auditoria da qualidade.

Há normas que estabelecem a temática da ergonomia na interação homem-máquina, como, por exemplo, a NBR ISO 9241-171, que trata da Ergonomia da Interação Humano-Sistema – orientações sobre acessibilidade de *software*.

Esta parte da NBR ISO 9241 proporciona orientações de ergonomia e especificações para o projeto de *software* acessível para uso no trabalho, no lar, na educação e em lugares públicos.

Abrange questões associadas ao *software* acessível para pessoas com a mais ampla gama de aptidões físicas, sensoriais e cognitivas, incluindo aquelas que estão temporariamente incapazes e os idosos.

Ela aborda considerações de *software* para acessibilidade que complementam o projeto geral para a usabilidade, como tratado pela NBR ISO 9241-110, NBR ISO 9241-11, ISO 9241-17, ISO 14915 e ISO 13407.

Não vamos fazer apologia às normas ISO, mas sem dúvida alguma você já teve oportunidade de acessar (ou no mínimo ver) softwares com telas, digamos, inadequadas.

Apresentamos algumas telas para você perceber as diferentes nuances entre elas.

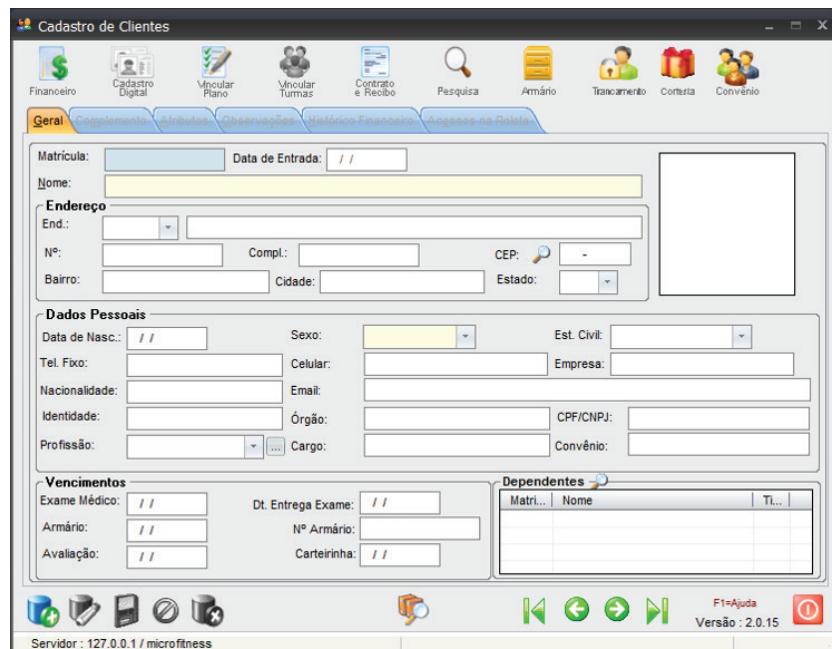


Figura 6 – Tela de cadastro de cliente

Fonte: Acervo do Conteudista

Observe a tela acima e identifique que a mesma:

- contém muitos campos a serem preenchidos, mas não necessita de rolagem de tela;
- é de fácil entendimento e tem os campos bem distribuídos;
- está dividida em 3 áreas de informações, bem definidas;
- contém ícones (em cima) e botões (embaixo) simples e de fácil compreensão;
- contém abas (em cima) que separam os assuntos a serem tratados pelo software.

Graças: Gerenciador Escolar – Tela Atualizar Registro de Aluno (Painel Dados do Responsável)

Formulário: Dados do Responsável.

Descrição: Painel onde serão inseridos os dados do Responsável legal do aluno, junto com uma tabela onde poderão ser incluídos contatos terceiros informados pelo responsável, deve ser preenchida durante do ato da matrícula, pois contém campos obrigatórios para o mesmo.

Número	Campo	Descrição	Tipo	Tamanho	Máscara	Origem	RN	RT	RNT	UC
5	Nome do Responsável	Campo de Texto	VarChar	200	-	Possível	RN1025	RT1	RNT1	UC-004
6	CPF	Campo Formulado	VarChar	15	000.000-000	Possível	RN1025	RT1	RNT1	UC-004
8	E-mail	Campo de Texto	VarChar	100	-	Possível	RN1025	RT1	RNT1	UC-004
9	Telefone	Campo Formulado	VarChar	15	(000) 0000-0000	Possível	RN1025	RT1	RNT1	UC-004
10	Celular	Campo Formulado	VarChar	15	(000) 0000-0000	Possível	RN1025	RT1	RNT1	UC-004

Figura 7 – Tela de atualização de registro de aluno

Fonte: Acervo do Conteudista

Observe que a tela acima tem, além das mesmas observações feitas na tela anterior, algumas informações adicionais:

- um campo de descrição que explica ao usuário de que se trata – pode ser usada posteriormente no manual do usuário e no “help”;
- cada campo tem uma identificação numérica, que é devidamente explicada na tabela logo abaixo – esta contém o tipo de campo, tamanho, descrição e outros dados de grande utilidade para o desenvolvedor.

Ora, você deve estar pensando – o que devo fazer, então??

Estamos tratando de protótipos ou *mockups* e, portanto, não é necessário se preocupar com os detalhes adicionais das telas apresentadas.

Acontece que, após a elaboração dos protótipos, nem sempre a tela codificada terá as mesmas informações ou será da mesma forma. Até aí tudo bem, **se** tiver mais qualidade, melhor usabilidade e ficar mais bonita.

Mas isso nem sempre acontece. Pior – os usuários nem sempre entendem o porquê de a tela ficar nesta ou naquela forma e os desenvolvedores se esquecem de dar um tratamento mais adequado à documentação da tela.

Consequentemente, haverá maiores dificuldades quando for necessário fazer uma manutenção no *software* e, muitas vezes, com maior custo.

Mapa Navegacional do *Software*

Com as definições das telas, os usuários começam a compreender melhor o funcionamento do *software*.

Mas, ainda assim, não têm uma visão geral do mesmo. O desenvolvedor, por sua vez, preocupado com as linhas de codificação, nem sempre percebe essa mesma conexão.

O mapa navegacional contribui, e muito, para que tanto usuários quanto desenvolvedores tenham a visão das conexões entre as diversas partes pertencentes ao *software*.

Adicionalmente, o mapa navegacional irá ajudar ambas as partes interessadas (usuários e desenvolvedores) a testar e validar o *software* com maior fidedignidade.



Vejamos alguns exemplos.

- Exemplo de mapa navegacional. Disponível em: <http://bit.ly/2ReF0r1>
- Exemplo de mapa navegacional. Disponível em: <http://bit.ly/36htYpc>

Observe que nos *links* acima percebe-se claramente a sequência de execução dos processos, com as ações solicitadas e seus respectivos retornos.

Desta forma, é possível validar-se as transações que o *software* executa e conforme a sequência correta após a solicitação do usuário.

Essa facilidade contribui para melhor compreensão das funcionalidades do *software*, possibilita avaliação da qualidade dos requisitos dos usuários e garante aos desenvolvedores a visão completa do funcionamento do *software*, podendo-se, inclusive, fazer modificações de melhorias antes da entrega do *software* aos usuários e clientes.

Material Complementar

Indicações para saber mais sobre os assuntos abordados nesta Unidade:

Leitura

Wireframing, prototyping, mockuping: what's the difference?

<http://bit.ly/2UeRiBZ>

Mockup

<http://bit.ly/3aRnroY>

O que é *mockup*?

<http://bit.ly/2tX5k07>

7 sites para baixar *mockups* para (quase) todas as situações

<http://bit.ly/2S08s3v>

11 dicas de ferramentas que vão te ajudar a criar um *layout* excelente!

<http://bit.ly/36zPHZC>

Descubra 8 ferramentas para criar protótipos e *wireframes*

<http://bit.ly/38MGRJH>

Introdução a interface GUI no Java

<http://bit.ly/38PNE5x>

Mapa/*storyboard* navegacional (*whiteboarding*)

<http://bit.ly/3aSatag>

Crie *mockups* para exibir suas melhores ideias

<http://bit.ly/2RVildg>

Referências

PRESSMAN, R. S.; MAXIM, B. R. **Engenharia de software**: uma abordagem profissional. 7^a ed. Porto Alegre: AMGH, 2016. (*e-book*)

SOMMERVILLE, I. **Engenharia de software**. 8^a ed. São Paulo: Pearson Addison Wesley, 2007. (*e-book*)

WAZLAWICK, R. S. **Análise e projeto de sistemas de informação orientados a objetos**. São Paulo: Campus, 2004.



Cruzeiro do Sul
Educacional