



CENTRO UNIVERSITÁRIO UNIVATES
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
CURSO DE SISTEMAS DE INFORMAÇÃO

DIOGO DE JESUS

**SOFTWARE PARA PROTOTIPAÇÃO
DE INTERFACES DESKTOP**

Lajeado
2011

DIOGO DE JESUS

SOFTWARE PARA PROTOTIPAÇÃO DE INTERFACES DESKTOP

Trabalho de Conclusão de Curso apresentado ao Centro de Ciências Exatas e Tecnológicas do Centro Universitário UNIVATES, como parte dos requisitos para a obtenção do título de bacharel em Sistemas de Informação.

Área de concentração: Sistemas de Informação

ORIENTADOR: Fabrício Pretto

Lajeado

2011

DIOGO DE JESUS

SOFTWARE PARA PROTOTIPAÇÃO DE INTERFACES DESKTOP

Este trabalho foi julgado adequado para a obtenção do título de bacharel em Sistemas de Informação do CETEC e aprovado em sua forma final pelo Orientador e pela Banca Examinadora.

Orientador: Prof. Fabrício Pretto, UNIVATES

Mestre pela Pontifícia Universidade Católica do Rio
Grande do Sul - PUCRS

Banca Examinadora:

Prof. Pablo Dall'Oglio, UNIVATES

Mestre em Computação Aplicada pela UNISINOS (2010)

Prof. Evandro Franzen, UNIVATES

Mestre em Computação pela Universidade Federal do Rio Grande do Sul,
Brasil (2002)

Prof. Fabrício Pretto, UNIVATES

Mestre pela Pontifícia Universidade Católica do Rio Grande do Sul - PUCRS

Coordenador do Curso de Sistemas de Informação:

Prof. Evandro Franzen

Lajeado, Junho de 2011

Dedico este trabalho a minha esposa e aos meus pais, pelo incentivo e apoio em todos os momentos difíceis.

AGRADECIMENTOS

A Deus por toda luz que nos dá, e por nos propiciar o livre arbítrio de escolhermos e trilharmos nosso caminho. Gostaria de deixar aqui registrado meu eterno agradecimento a minha amada esposa Adriana que sempre paciente me incentivou e acalmou em momentos complicados dessa trajetória, aos meus pais que são meus ídolos por tudo que fizeram e fazem por mim, sempre tendo uma palavra amiga na hora certa, as orações da minha mãe que sempre mostram o caminho da luz em busca no aperfeiçoamento.

A todos meus colegas que de uma forma e de outra sempre ajudaram com alguma dica, trocando alguma experiência, aos meus amigos que sempre me apoiaram.

Um agradecimento especial ao meu orientador Fabrício Pretto, que foi uma pessoa incrível se mostrando sempre pronto para ajudar não importasse a hora e o dia da semana, sempre com palavras chaves na hora das minhas dúvidas, ajudando a desenvolver minhas idéias que fizeram com que esse trabalho fosse realizado.

Para finalizar não poderia esquecer de agradecer ao nosso animal de estimação, a Laika, que soube me distrair naquelas horas que os pensamentos pareciam confusos.

RESUMO

Durante o processo de desenvolvimento de software, o projeto da interface do sistema merece uma atenção especial. A área de IHC (Interface Homem Computador) é responsável por empregar técnicas que controlam a qualidade e usabilidade das interfaces. Através da interface ocorrerá toda a interação do usuário com as funções disponíveis no sistema. Nesse sentido, projetar uma interface de fácil manuseio e de forma organizada diminui erros de interação do usuário e estimula sua capacidade de percepção e a atenção, para que ele possa interagir com o software de maneira mais eficaz. Protótipos de interface são construídos durante o processo de modelagem do sistema com a finalidade de rastrear requisitos de sistema e fornecer um primeiro contato visual do usuário com o sistema proposto. Conforme estudos realizados foram detectadas algumas ferramentas que possibilitam a criação de interfaces com funcionalidades limitadas, que são destinadas a profissionais da área de software que se dedicam a desenvolver interfaces desktop e web. Nesse sentido, serão abordadas neste trabalho algumas técnicas para projeto de interface, bem como uma comparação entre ferramentas existentes para prototipação. Uma ferramenta de prototipagem rápida e eficiente foi desenvolvida implementando os conceitos de IHC.

Palavras-chave: Prototipação de Interfaces, Software, Interface Homem Computador.

ABSTRACT

During the software developing process, the system interface project deserves special attention. The HCI area (Human Computer-Interaction) is responsible for the use of techniques that control the interface quality and usability. Through the interface will happen all the interaction of the user with the functions available in the system. Considering this, projecting an easy handling and organized interface reduces the user's interaction mistakes and stimulates attention and perception capacity, so the user can interact with the software in a more effective way. Interface prototypes are built during the modeling process of the system in order to trace system requirements and provide the first visual contact of the user with the proposed system. There are tools that enable the creation of interfaces with limited functionality, dedicated for software professionals who work on the development of desktop and web interfaces. Considering this, this paper will approach some techniques for interface projects and will compare the existing tools for prototyping. It was developed a fast and efficient prototyping tool which implements the concepts of HCI.

Key-words: Interface Prototyping, Software, Computer-Human Interface.

LISTA DE FIGURAS

Figura 1 Ciclo de vida de software, modelo cascata (SOMMERVILLE, 2008).....	18
Figura 2 Fases metodologia RUP (Rational Unified Process) (SOMMERVILLE, 2008).....	21
Figura 3 Exemplo de um cartão de tarefas para baixar documentos (SOMMERVILLE, 2008)	21
Figura 4 Elementos de um diagrama de atividade representado pela UML (Quatrani 2001) ..	26
Figura 5 Objetos e mensagens em um diagrama de sequência. (Quatrani 2001).....	26
Figura 6 Disciplinas que contribuem para o IHC (Preece, 1994).....	27
Figura 7 Interface desktop com widgetst (http://www.slideshare.net/fabianodamiati/ihc-aula15).....	30
Figura 8 Exemplo de um menu seguro e um não seguro (PREECE et. al. 2005).	31
Figura 9 Modelo de site com seu espaço de tela mau utilizado (NIELSEN, 2000).	33
Figura 10 Modelo de site mostrando vários links na mesma série (Nielsen, 2000).	34
Figura 11 Layout de site que exhibe de forma clara o emprego de multimídia (NIELSEN, 2000).....	36
Figura 12 Exemplo de um Storyboard (PREECE et. al. 2005).	38
Figura 13 Exemplo de Wireframe Alta fidelidade, demonstrando a organização dos elementos	39
Figura 14 Processo de desenvolvimento de protótipo (SOMMERVILLE, 2008)	40
Figura 15 Tela de Cadastro de Clientes desenvolvida no Software Designer Vista	44
Figura 16 Tela de Cadastro de Clientes desenvolvida no Software Gui Design Studio	45
Figura 17 Tela de Cadastro de Clientes desenvolvida no Software Wiremaster.....	46
Figura 18 Tela de Cadastro de Clientes desenvolvida no Software Axure RP Pro 5.6.....	48
Figura 19 Arquitetura do software Easy Screen.....	51
Figura 20 Diagrama de Casos contemplados no Sistema Easy Screen	52
Figura 21 Diagrama Classes Software Easy Screen.....	53
Figura 22 Classes GenericModel e jGenericComponent	54
Figura 23 Diagrama de Componentes demonstrando o uso das classes externas.	55
Figura 24 Visão da interface inicial do Software Easy Screen.....	57
Figura 25 Tela dos Componentes e Ícones.....	60
Figura 26 Tela do painel de Propriedades dos Elementos.....	61
Figura 27 Imagem ilustrando ação do redimensionar e o componente com a ação selecionada	62
Figura 28 Tela da Área de trabalho do software Easy Screen.....	63
Figura 29 Tela Mapa do Projeto mostrando suas opções de interação.....	64
Figura 30 Tela exibindo o menu com suas funções	66
Figura 31 Tela desenvolvida por usuário na etapa de Validação	70
Figura 32 Gráfico Estatístico demonstrando a avaliação da ferramenta Easy Screen.....	71

LISTA DE CÓDIGOS

Listagem 1 Código XML referente ao componente botão inserido na tela do Easy Screen	56
Listagem 2 Código representando a inserção de um componente.....	59

LISTA DE TABELAS

Tabela 1 Classificação dos Protótipos em Baixa Fidelidade e Alta Fidelidade	38
Tabela 2 Tabela comparativa entre principais funcionalidades dos softwares citados	49
Tabela 3 Tabela comparativa das funcionalidades do Easy Screen com demais softwares.....	67

LISTA DE ABREVIATURAS

IHC: Interação Humano-Computador

RNF: Requisitos Não Funcionais

UML (*Unified Modeling Language*): é uma linguagem para especificação, documentação, visualização e desenvolvimento de sistemas orientados a objetos.

IDE (*Integrated Development Environment*) – Ambiente de Desenvolvimento Integrado.

SUMÁRIO

1	INTRODUÇÃO.....	14
1.1	Motivação do Trabalho.....	14
1.2	Objetivos.....	15
1.3	Estrutura do Documento	15
2	REVISÃO DE LITERATURA	17
2.1	Engenharia de Software.....	17
2.1.1	Software.....	17
2.1.2	Ciclo de vida de um software	18
2.1.3	Metodologias de desenvolvimento de software.....	19
2.1.4	Projeto de Software	22
2.1.4.1	Requisitos de Software	23
2.1.4.2	Modelagem de software	24
2.2	IHC (Interação Homem Computador).....	27
2.2.1	Projeto de Interface.....	28
2.2.1.1	Design de interfaces desktop e web.....	28
2.2.1.2	Design da Tela.....	31
2.2.3	Protótipos.....	37
2.2.4	Avaliação de IHC	41
3	ANÁLISE DE FERRAMENTAS DE PROTOTIPAÇÃO.....	43
3.1	Designer Vista	43
3.2	Gui Design Studio	44
3.3	Wiremaster.....	46
3.4	Axure RP Pro 5.6.....	47
3.5	Análise de Softwares da Prototipação de interfaces.....	48
4	IMPLEMENTAÇÃO SOFTWARE EASY SCREEN	50
4.1	Modelagem do Sistema	50
4.1.1	Requisitos do Sistema.....	51
4.1.2	Diagrama de Classes.....	53
4.2	Aplicação desenvolvida.....	56
4.2.1	Visão Geral Easy Screen	57
4.2.2	Componentes e Ícones	58
4.2.3	Propriedade dos Componentes	60
4.2.4	Área de Trabalho	61
4.2.5	Árvore do Projeto	63
4.2.6	Menu e Barra de Ferramentas.....	64
4.2.7	Análise do Easy Screen com Softwares similares	66
5	VALIDAÇÃO DO SOFTWARE EASY SCREEN	68
6	CONCLUSÃO.....	72
6.1	Trabalhos Futuros	72
	APÊNDICE A: PROTOCOLO DE TESTES	75

APÊNDICE B: QUESTIONÁRIO PARA AVALIAÇÃO DO SOFTWARE EASY SCREEN	
.....	76

1 INTRODUÇÃO

Para que um projeto de desenvolvimento de software tenha sucesso, vários fatores são importantes, um deles é a capacidade de percepção do que o cliente realmente deseja, quanto antes isso for captado, melhor serão os resultados. Se conseguirmos captar as reais necessidades do usuário, compreenderemos mais facilmente no que o sistema a ser produzido irá ajudar em relação às regras de negócio de uma empresa.

É comum existirem problemas de comunicação entre usuários e profissionais da área de desenvolvimento de sistemas (PRESSMAN, 1995). A disciplina de IHC (Interação Humano-Computador) é composta por algumas técnicas e regras para projeto de interfaces que fazem com que seja diminuída a distância entre o que está sendo desenvolvido e os seus usuários. Para que haja uma intermediação entre a comunicação de desenvolvedor e usuário, existem técnicas que definem padrões de formas, desenhos, símbolos e cores.

A prototipação de interfaces, foco deste trabalho, é uma técnica que tem um papel importante na etapa de aproximação entre os *stakeholders*¹ de um projeto, pois através dessa prototipação os *stakeholders* conseguem ter o primeiro contato visual com o sistema que será desenvolvido. Através dessa primeira interação o usuário conseguirá expor de forma mais fácil o que realmente deseja no sistema. O design desse protótipo de interface através de técnicas de IHC, auxilia no desenvolvimento de uma interface mais intuitiva, que facilitará o uso do sistema.

1.1 Motivação do Trabalho

Em épocas tão competitivas como as atuais, é importante que as equipes de desenvolvimento possam desenvolver sistemas que atendam cada vez mais as necessidades do cliente, de uma forma mais rápida e mais eficaz. Seguindo essa linha de raciocínio o software de prototipação de interfaces, encaixa-se como uma ferramenta que apóia o desenvolvedor a diminuir a distância que existe entre o software e o usuário, fazendo com que o usuário possa ter uma visão mais simplificada da parte visual do software.

Essa primeira visão do software, fará com que o usuário e desenvolvedor consigam se comunicar melhor para o andamento do projeto, com isso poderão ser detectados problemas na parte inicial da implantação do projeto, tudo isso poderá diminuir custos e prazo de entrega do sistema.

¹ Stakeholders – São pessoas que serão afetadas pelos sistema e que tem a influência direta ou indireta na elaboração dos requisitos (SOMMERVILLE, 2004).

1.2 Objetivos

O objetivo principal deste trabalho é o desenvolvimento de um software de prototipação de interfaces de sistemas, que auxilie os profissionais da área de projeto e desenvolvimento de software, implementar um protótipo inicial, para melhor interagir com o cliente, podendo dar uma ideia do futuro sistema. Essa ferramenta possibilitará a criação de interfaces *desktop* seguindo os padrões de IHC. A escolha pela área desktop foi feita por não existirem tantas ferramentas no mercado voltadas a essa área em específico.

Os objetivos específicos desse trabalho são:

- Desenvolver o software de forma que auxilie o profissional desenvolvedor o projeto de interface de um sistema de forma rápida e clara;
- Avaliar *softwares* já existentes na área de prototipação de interfaces;
- Implementar uma ferramenta de prototipação fazendo uso da linguagem Java;
- Validar a ferramenta com profissionais da área de desenvolvimento de software;

Não é objetivo desse projeto avaliar o software desenvolvido com base em todas as técnicas de avaliação sugeridas pela área de IHC, como *walkthrough* cognitivo e pluralístico, avaliação heurística, *checklist*, dentre outras, mas sim uma avaliação de usabilidade rápida.

1.3 Estrutura do Documento

O presente trabalho encontra-se estruturado em 5 capítulos.

O capítulo 1 tem como conteúdo a introdução, que é formada pelos comentários iniciais e define os objetivos gerais e específicos do trabalho.

O capítulo 2 é composto pela revisão bibliográfica onde serão abordados os seguintes temas: Engenharia de Software dando uma ênfase nas etapas de desenvolvimento de um software, enquadrando a área de IHC, onde será falado do projeto de interface, design de interfaces *desktop* e web, e protótipos.

Na sequência no capítulo 3, serão descritas e avaliadas algumas ferramentas de prototipação já existentes no mercado.

O capítulo 4 relata a proposta do projeto que será desenvolvido, descrevendo seus requisitos, suas funcionalidades, abordando a linguagem de programação que será adotada, bem como o *framework* e IDE (*Integrated Development Environment*).

Finalizando o projeto é relatado no capítulo 5 a etapa de validação do software Easy Screen, no qual o mesmo foi avaliado por 20 usuários experientes da área de informática, os quais eram alunos da UNIVATES que estavam cursando a disciplina de IHC no semestre A do ano de 2011, são mostrados os testes que os mesmos executaram, bem como o resultado das avaliações.

2 REVISÃO DE LITERATURA

O presente capítulo contém a revisão bibliográfica de assuntos relacionados a este trabalho, divididos em: engenharia de software, conceitos sobre a área de IHC, requisitos de software, protótipos, projetos de interface *Desktop* e *Web*, e avaliação de interfaces.

2.1 Engenharia de Software

A Engenharia de software é uma das áreas da engenharia que trata dos aspectos do desenvolvimento de software, ela está presente em todas as etapas do desenvolvimento do software começando pela especificação e seguindo até o final do ciclo na parte da manutenção do sistema (SOMMERVILLE, 2004).

Segundo Pressman (2010), Engenharia de software consiste numa disciplina que compõe o processo, ferramentas e métodos para o desenvolvimento de softwares, conjunto de tarefas que são conduzidas durante as etapas de fabricação do mesmo.

2.1.1 Software

O conceito de software de um modo geral está muitas vezes implícito na mente das pessoas como sendo somente programas que executam em computadores. Uma ideia mais ampla está associada a esse conceito, pois o software não consiste em ser somente um programa, mas em muitos dados associados como documentação e configuração, dados esses que são imprescindíveis para que o software possa ser operado de forma que atenda as necessidades dos usuários (SOMMERVILLE, 2004).

Segundo Pressman (2010), são conjuntos de documentos que mostram como deve ser a operação e como deve ser o uso dos programas. O software necessita muito além de um conceito formal que o defina, é necessário entender as características que fazem com que o software seja desenvolvido de uma forma diferente do que outros produtos.

Pressman (2010) descreve que “software é um elemento de um sistema lógico e não de um sistema físico”.

Os softwares desenvolvidos pelos engenheiros de software transformam-se em um produto de software, os quais podem ser utilizados nas próprias empresas que possuem seu departamento de desenvolvimento de software, ou criados por empresas especializadas para serem comercializados. Os dois modelos fundamentais de produtos de software são: (SOMMERVILLE, 2008).

- Softwares Genéricos: definem-se por sistemas independentes que são desenvolvidos por uma empresa e são colocados à venda para qualquer cliente adquiri-los. Exemplificando esses produtos de software existem os processadores de texto e os programas para desenhos gráficos.
- Softwares Personalizados: consistem no desenvolvimento de sistemas para uma empresa/cliente em particular, o mesmo software será projetado de maneira específica para esse cliente que contratou a empresa desenvolvedora do software (SOMMERVILLE, 2004).

2.1.2 Ciclo de vida de um software

O ciclo de vida de um software é considerado uma sequência de fases e tarefas, que são desenvolvidas no transcorrer do projeto de desenvolvimento de um sistema. O software passa por várias fases desde a ideia inicial, até o seu desenvolvimento e implantação. Uma das principais finalidades do ciclo de vida é desenvolver softwares mais estruturados e com maior documentação (PRESSMAN, 2010).

O processo do ciclo de vida de um software desenvolve-se em várias fases, cada uma possui várias atividades que são realizadas pelas partes envolvidas. O modelo cascata, exemplo utilizado durante anos no processo de criação de sistemas ilustra a metodologia a ser aplicada, a figura 1 mostra o ciclo de vida de software no modelo cascata nas fases que o compõem. Nesse modelo a fase seguinte somente ocorre quando a fase anterior estiver sido finalizada.

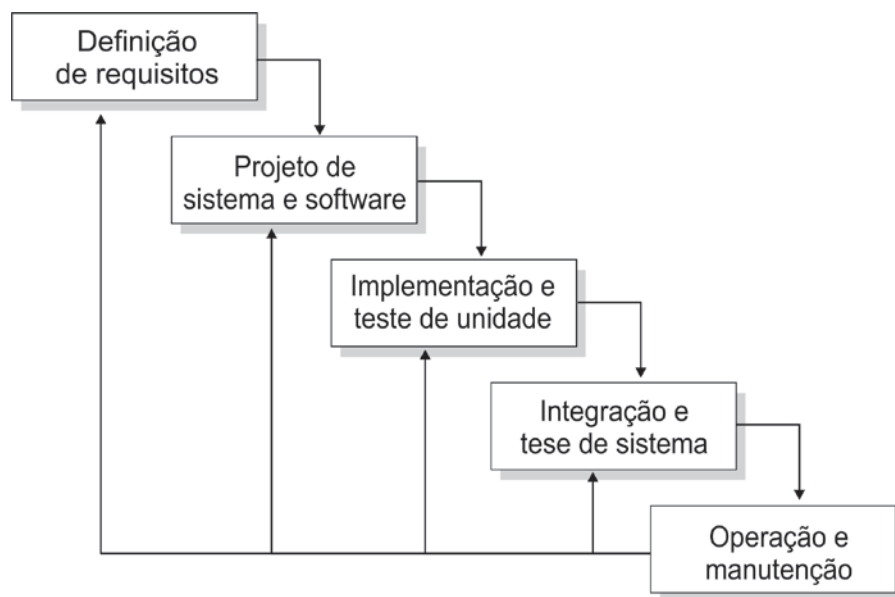


Figura 1 Ciclo de vida de software, modelo cascata (SOMMERVILLE, 2008)

Essas fases exemplificadas na figura 1, que podem ser genéricas ou de um modelo específico, podem ser descritas conforme segue: (PRESSMAN 2010, SOMMERVILLE 2008)

- Definição de requisitos: procura a identificação através de consultas aos usuários, metas que devem ser atingidas e objetivos que deverão ser alcançados.
- Projeto de Sistema: nesse processo encontra-se a divisão dos requisitos em sistemas de hardware ou software. Contempla-se a representação das funções do sistema de uma forma que se possa ser transformada em programas executáveis.
- Implementação e teste de unidade: a etapa da implementação é a parte onde o projeto do sistema transforma-se em programas usando alguma linguagem de programação escolhida pela equipe de desenvolvimento. Por sua vez o teste unitário verifica se cada unidade atende à sua especificação.
- Integração e teste de sistema: nesta etapa aplica-se a união dos programas onde ocorrem os testes de todo o sistema, para que possa cumprir todos os requisitos especificados, na fase descrita acima.
- Operação e manutenção: na operação o sistema é instalado e colocado em uso. Na fase de manutenção, ocorre a correção de erros não antes detectados e também podem ocorrer melhorias de qualidade.

2.1.3 Metodologias de desenvolvimento de software

A aplicação de um conjunto de práticas durante o desenvolvimento de software resulta nas metodologias de desenvolvimento de software. Existem várias metodologias, dependendo da qual for utilizada será produzida maior ou menor interatividade com o cliente, e também a quantidade de documentação que será desenvolvida. Algumas das atividades envolvidas no desenvolvimento do software são comuns entre si, independentes da metodologia que será utilizada.

Entre várias metodologias de desenvolvimento de software, podem ser citadas: modelo cascata, RUP, modelo espiral, evolucionário, modelo iterativo incremental, metodologia extreme programming (XP) e modelo de prototipagem (PRESSMAN 2010, SOMMERVILLE 2008).

A metodologia RUP (*Rational Unified Process*), uma das mais utilizadas na atualidade, é considerada um processo iterativo e incremental, não é um processo adequado a todos os tipos de desenvolvimento, mas é um dos processos mais recentes e que vem sendo muito utilizado. A metodologia RUP encaminha para o controle de qualidade e para que haja gerenciamento de riscos contínuos e objetivos. Este processo é desenvolvido com uma arquitetura robusta, o que diminui o retrabalho e aumenta a reutilização de componentes e a manutenção do sistema (SOMMERVILLE, 2008).

A metodologia RUP é composta por 4 fases, conforme é mostrado na Figura 2. Estas fases estão mais relacionadas aos negócios do que os assuntos técnicos, diferentemente do modelo cascata, no qual as fases contemplam as atividades do processo.

As fases do processo RUP são:

- Concepção: fase que ocorrem as reuniões onde será discutido o problema, também irá ocorrer um levantamento do que será necessário para que o projeto seja executado, o escopo do projeto também é definido nessa fase.

- Elaboração: nesta fase a finalidade principal é que seja analisado o domínio sobre o problema, a eliminação de elementos que contenham alto risco, riscos esses que podem ser tecnológicos ou referente às habilidades das pessoas envolvidas no projeto.

É considerada uma fase essencial, pois nesta etapa a parte de engenharia está completa, e qualquer mudança a partir dessa etapa implicará em custos altos.

- Construção: nesta fase será construída a parte de modelagem a qual deve aplicar algumas das notações definidas pela UML.

- Transição: é a fase na qual o sistema está concluído, inicia-se a etapa de implantação do sistema para o usuário, os treinamentos também começam a ser aplicados aos usuários que irão manter e utilizar o sistema.

Na sequência a Figura 2 demonstra o processo RUP juntamente com suas 4 fases que foram descritas acima.

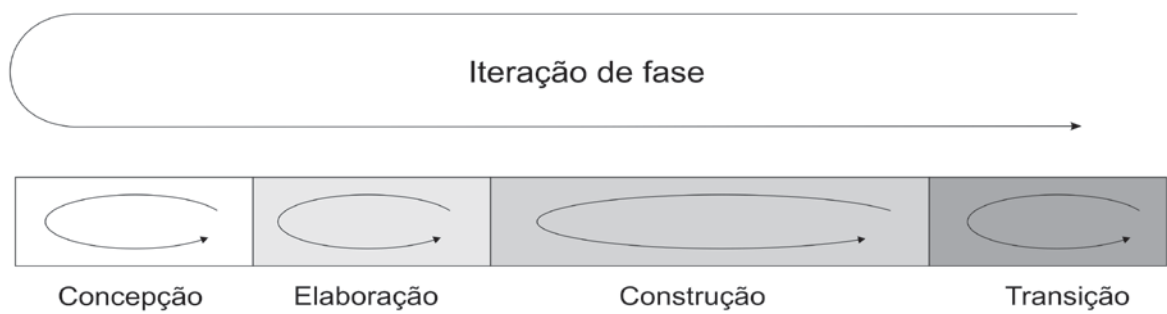


Figura 2 Fases metodologia RUP (Rational Unified Process) (SOMMERVILLE, 2008)

Uma metodologia ágil, muito conhecida e uma das mais aplicadas é a extreme programming (XP), nesse método os requisitos são demonstrados como cenários que são aplicados com várias tarefas. Antes da escrita do código de programação são desenvolvidos testes por dois programadores que trabalham juntos nessa metodologia (SOMMERVILLE, 2008)

Nesse método o cliente participa do desenvolvimento e é o responsável pela aplicação e aprovação dos testes que serão efetuados no sistema.

Existem algumas práticas no extreme programming (XP), que se encaixam nos métodos ágeis, nos itens abaixo serão citadas algumas delas:

- Planejamento incremental: todos os requisitos são inseridos em cartões de histórias e as mesmas serão inseridas em release. As histórias serão divididas em tarefas conforme exemplifica a figura 3:

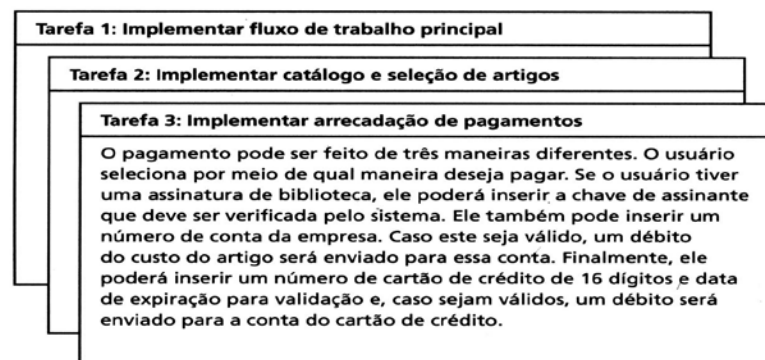


Figura 3 Exemplo de um cartão de tarefas para baixar documentos (SOMMERVILLE, 2008)

- Pequenos *releases*: consiste em funcionalidades úteis que agregam utilidades ao negócio.

- Projeto Simples: é desenvolvido um projeto pequeno que seja suficiente para atender os requisitos necessários no momento.

- Programação em pares: para que o trabalho seja executado de forma eficaz, os desenvolvedores do projeto trabalham em duplas, dessa forma um consegue avaliar e verificar o trabalho do outro (SOMMERVILLE, 2008).

Para especificação desses diagramas citados acima existe uma linguagem visual para especificação chamada UML, surgida em 1996 e iniciou um esforço para a padronização na área de métodos.

Neste mesmo ano a Rational Software Corporation teve a participação de outras Instituições parceiras como: Hewlett-Packard, IBM, Microsoft, Oracle, Unisys, Platinum Technology, etc. Essa colaboração contribuiu para a produção da versão 1.0, uma versão bem definida, poderosa e aplicável, a qual foi submetida a OMG para adoção.

No ano de 1997 foi liberada a versão 1.1 da UML e ainda no mesmo ano a mesma foi aprovada como padrão pela OMG.

Ficou definido então, que a UML seria uma linguagem de modelagem designada para especificar, visualizar, construir e documentar um sistema. A linguagem de modelagem é a notação que o método utiliza para expressar projetos enquanto que o processo indica quais passos seguir para desenvolver um projeto. (QUATRANI, 2001)

2.1.4 Projeto de Software

Considerado como uma etapa importante para que o sistema seja desenvolvido de maneira que contemple todas as funções que lhe forem solicitadas, o projeto de software fornece uma visão ampla do que deve ser executado e aplica a técnica que melhor atenderá as necessidades do software.

O objetivo do Projeto de Software é transformar os resultados da análise de requisitos de software em documentos que possam ser interpretados pela equipe de desenvolvimento. Existem algumas ferramentas que ajudam na etapa do projeto do sistema, são elas: diagramas de casos de usos, diagramas de entidade e relacionamento, diagramas de estados, diagramas de classes, dentre outros.

2.1.4.1 Requisitos de Software

O termo requisito abrange os limites de um software, especifica quais as funções ele deve executar, quais são suas regras operacionais. O levantamento de requisitos apresenta muitas formas de descrição, cabe ao engenheiro de software extrair de forma clara, para que todos possam compreender os requisitos quando preenchidos (PREECE et. al. 2005).

Os requisitos de software podem ser entendidos de forma mais clara quando classificados em requisitos funcionais, requisitos não funcionais e requisitos de domínio.

Segundo (SOMMERVILLE, 2008), os requisitos funcionais (RF) representam as funcionalidades que o sistema deve oferecer, o comportamento do mesmo dependendo onde o mesmo será empregado. Os requisitos irão sofrer influências dependendo do modelo, da estrutura e da complexidade de software que será criado. Um sistema deve ter em seu projeto requisitos funcionais que sejam completos, sólidos para que garantam que todas as necessidades solicitadas pelo usuário tenham sido especificadas e para que não haja requisitos ambíguos ou incompatíveis. Muitas vezes esses erros de especificação dos requisitos podem acontecer em sistemas grandes, ou também por causa dos *stakeholders* envolvidos no projeto, pois os mesmos no início do projeto têm suas necessidades frequentemente inconsistentes, essas inconsistências podem não ser visualizadas no início da especificação dos requisitos, porém quando o projeto chega ao final, ou até mesmo quando entrega-se o produto finalizado ao cliente, é que as mesmas tornam-se mais visíveis (SOMMERVILLE, 2008).

Requisitos Não Funcionais (RNF), na maioria das vezes não estão ligados às peculiaridades únicas do sistema e a funções diretas do sistema, os RNF estão ligados a tecnologia, tais como velocidade processamento/tempo de resposta, formato de armazenamento de dados, tipos de transmissão em redes como wireless ou cabeadas. Por isso os mesmos especificam a proteção, a disponibilidade e a robustez do sistema.

Para que se possa indicar os requisitos de um sistema, é utilizada a coleta de dados que constitui-se em uma tarefa importante no que diz respeito a avaliação e indicação dos mesmos. A finalidade da mesma é coletar informações importantes para que se possa produzir um requisito consistente (PREECE et. al. 2005).

Existem poucas técnicas de coleta de dados, mas elas podem unir-se uma com as outras, formando assim formas mais claras para que seja entendido o requisito necessário. Algumas dessas técnicas são: questionários, entrevistas *workshops*, observação natural e estudo de documentação (PREECE et. al. 2005).

Na medida que os requisitos são abstraídos, o sistema começa a gerar uma idéia geral das funções e os principais características do sistema começam a se concretizar. Mas para que haja avanço para as partes mais específicas e técnicas e a equipe de software compreenda como essas características serão empregadas, os envolvidos com o sistema projetam cenários que mostram os caminhos para o software ser desenvolvido (PRESSMAN, 2010).

Esses cenários consistem em criar atividades ou tarefas humanas, como se fossem inseridos em uma história, isso viabiliza a discussão do conteúdo desse cenário, necessidades e requisitos. São utilizados vocabulários de fácil entendimento aos *stakeholders*, pois a construção de cenários pelos mesmos são as primeiras formas de estabelecer requisitos, por isso não é utilizada uma linguagem tecnológica para realizar a descrição dos cenários (SOMMERVILLE, 2008).

2.1.4.2 Modelagem de software

Na etapa de modelagem de sistemas, é necessário conseguir padronizar através de uma modelagem gráfica o que o sistema irá conter, para que o profissional que irá desenvolver o sistema possa interpreta-lo (MELO 2003).

Uma linguagem de modelagem que atende plenamente essas necessidades de demonstrar o sistema de uma forma gráfica é UML (Unified Modeling Language), a mesma consegue fornecer um auxílio na construção e facilitar a documentação do software.

Para o desenvolvimento do Projeto de Software é preciso que seja feito um levantamento dos modelos que podem elucidar o projeto, alguns modelos conhecidos para modelagem dos sistemas são os modelos de casos de uso. Eles são usados para que possam ser descobertas as interações que ocorrem entre usuário e sistema e quais são os usuários ou outros sistemas externos que estão envolvidos (SOMMERVILLE, 2008). Esses usuários e sistemas externos que interagem com os sistemas são chamados de atores, por isso antes de começar a descrever os casos de uso é preciso identificar quem são esses atores (PRESSMAN, 2010).

No contexto geral do sistema é importante ter a atenção que atores e usuários finais não têm o mesmo significado, pois usuários interagem com o sistema de várias formas enquanto o ator representa uma classe que seria uma entidade externa que tem apenas uma representação no caso de uso.

Nas primeiras etapas de identificação de casos de uso, nem todos os atores serão localizados, nessas primeiras etapas quem é mais facilmente localizado são os atores principais que interagem diretamente com o sistema para executar as funções principais. Nas próximas etapas quando a equipe sabe mais sobre o sistema, os atores secundários são identificados, esses atores são aqueles que dão apoio ao sistema (PRESSMAN, 2010).

Segundo (PRESSMAN, 2010 (Jacobson [JAC92])), para que sejam desenvolvidos os casos de uso algumas questões devem ser respondidas pelos mesmos, tais como:

- Quem é (são) o(s) ator(es) principal(is) e o(s) ator(es) secundários?
- Quais são as metas dos atores?
- Que pré-condições devem existir antes da história começar?
- Que tarefas ou funções principais são desempenhadas pelo ator?
- Que variações na interação dos atores são possíveis?
- Que informações do sistema o ator vai adquirir, produzir ou modificar?
- O ator terá de informar o sistema sobre alterações no ambiente externo?
- Que informações o ator deseja com o sistema?
- O ator deseja ser informado sobre modificações inesperadas?

A seguir serão descritos alguns diagramas e modelos que são ferramentas úteis na modelagem de dados, pois auxiliam a interação entre os objetos de um sistema.

Diagrama de classes – conforme um modelo vai aumentando de tamanho com o acréscimo de muitas classes, começa a ficar difícil uma representação textual dessas classes. Neste contexto o diagrama de classes exibe uma estrutura estática das classes do sistema, as mesmas representam os objetos que são gerenciados pela aplicação modelada (QUATRANI, 2001).

O diagrama de classes mostra uma coleção de elementos, como classes, tipos com seus conteúdos e as relação entre as classes, esse diagrama apresenta 4 tipos de relacionamentos mais usados, generalização/especificação, agregação, associação, dependência (FURLAN, 1998).

Diagrama de Estados – expõe o comportamento dos objetos, tendo por objetivo mostrar um comportamento que demonstra a sequência de estado que um objeto percorre durante o ciclo de vida, obedecendo a eventos, responsabilidades e suas ações (MELO 2003).

Diagrama de Atividades – demonstram o fluxo de trabalho de um software, exibem qual atividade está sendo processada em paralelo e quais os possíveis caminhos podem ser mudados durante o fluxo. São compostos por atividades, transições das atividades, decisões e as barras de sincronização. A figura 4 apresenta os elementos do diagrama de Atividades UML (QUATRANI, 2001).

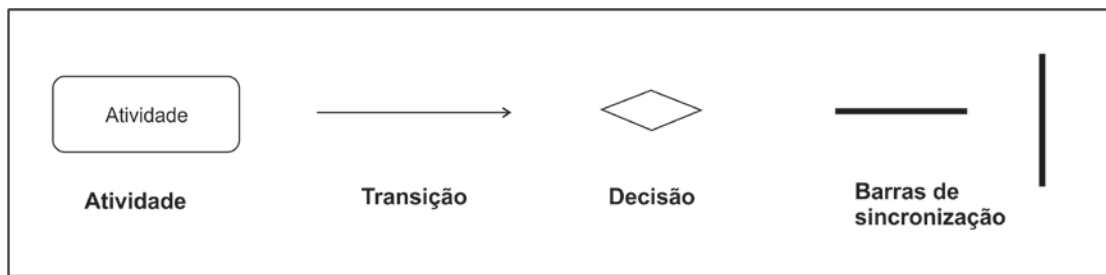


Figura 4 Elementos de um diagrama de atividade representado pela UML (Quatrani 2001)

Diagrama de Sequências – exibe a sequência de mensagens que são enviadas entre os vários objetos do sistema. O mesmo mostra as classes e os objetos e classes que encontram-se no cenário, também apresenta a sequência das mensagens que são alternadas entre os objetos.

Na linguagem UML, um objeto inserido num diagrama de sequência é representado por um retângulo que contém o nome do objeto, cada objeto possui uma linha de tempo tracejada, que encontra-se abaixo do retângulo, a setas representam a troca de mensagens entre os objetos(QUATRANI, 2001). A figura 5 exibe um diagrama de sequência usando a notação UML, o qual exibe um cliente cadastrando uma religião, primeiramente ele valida o usuário, acessando a classe usuário, após informa os dados e o método cadastrarReligião() é chamado fazendo a inserção.

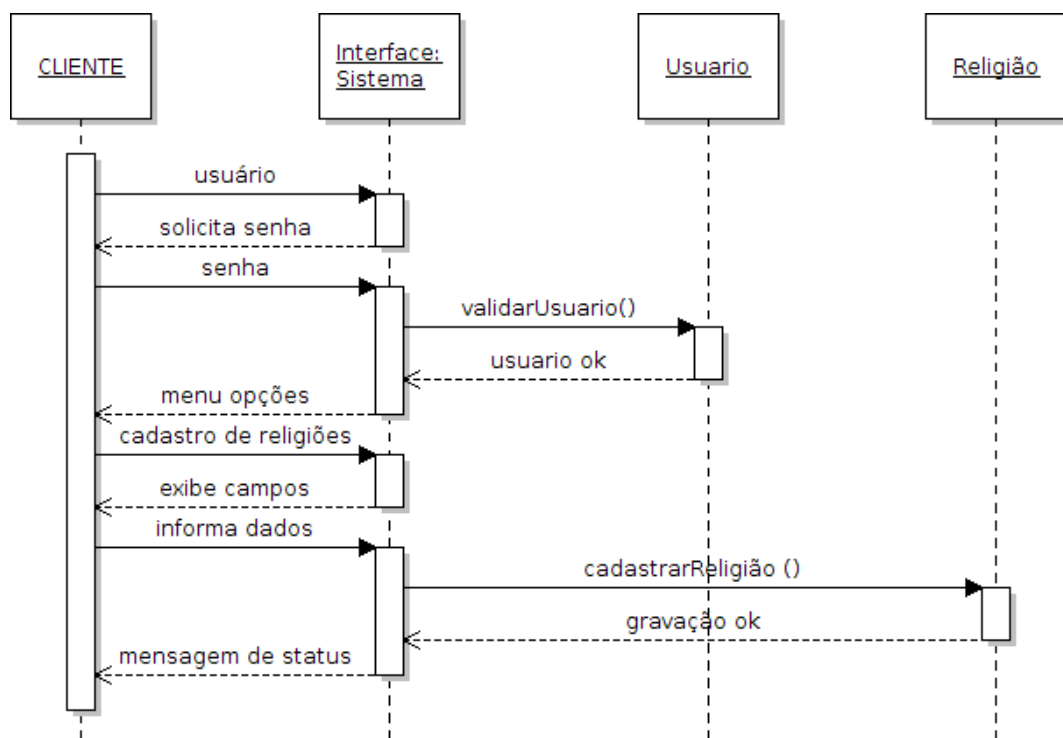


Figura 5 Objetos e mensagens em um diagrama de sequência. (Quatrani 2001)

2.2 IHC (Interação Homem Computador)

IHC é uma área preocupada com *design*, avaliação e implementação de sistemas computacionais com foco na interação entre as pessoas, as máquinas e com o ambiente que as cercam. Uma função da área IHC é o desenvolvimento de interfaces com o usuário, mas além das métricas para elaboração de projeto de interfaces, a mesma apresenta estudos para a comunicação e interação entre o usuário, computadores e sistemas (PREECE et. al. 2005).

A figura 6 exibe um conjunto de disciplinas que contribuem para que se desenvolva um IHC com maior satisfação, produtividade e segurança.

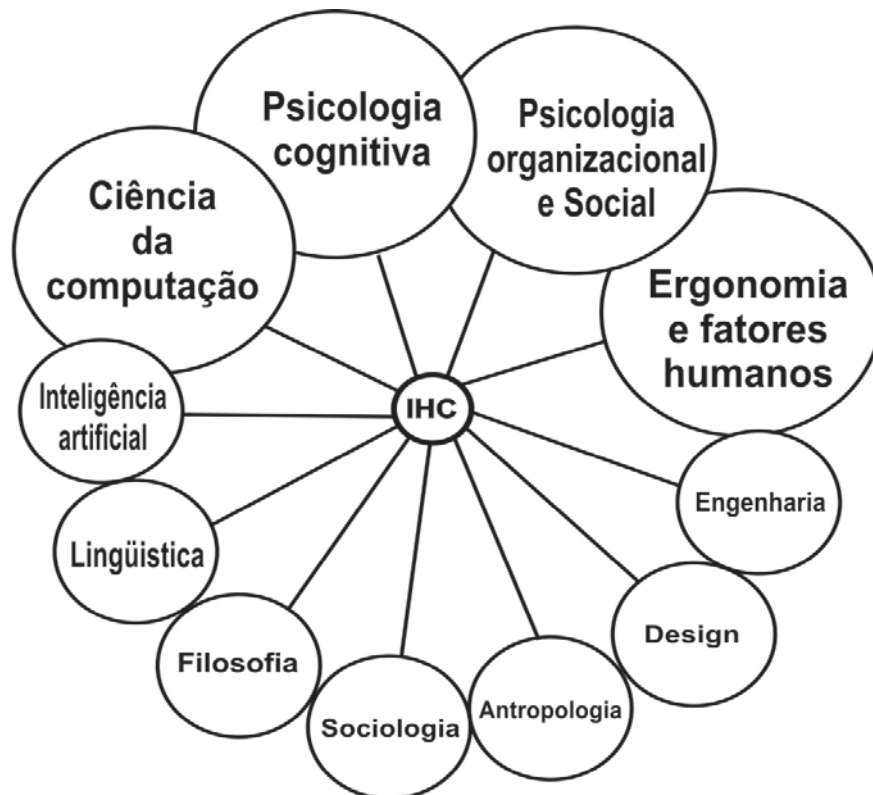


Figura 6 Disciplinas que contribuem para o IHC (Preece, 1994)

O projeto de IHC aproveita as fases gerais de elicitação das necessidades de um software, do ponto de vista de suas funcionalidades, para coletar informações a respeito da interface do produto a ser desenvolvido. Tais informações, obtidas através de entrevistas e outras técnicas de descoberta de informação, são nesse momento, analisadas sob o ponto de vista da interação do usuário com o sistema.

Através das métricas sugeridas pela área de IHC, o desenvolvedor consegue gerar interfaces de sistemas que se adaptem melhor aos seus usuários e o modo que eles realizam suas tarefas. Interfaces que demonstrem um melhor aprendizado fazem com que o usuário tenha uma melhor interação com o sistema, o qual se torna mais confiante com sua interação

com o software. Por outro lado uma interface de difícil interação causa desmotivação ao usuário, isso faz com que os usuários procurem outras ferramentas (CIBYS, 2007).

Durante as fases de desenvolvimento de um sistema, a interface desempenha um papel muito importante. O projeto de interface deve ser elaborado a fim de contemplar as funções que serão aplicadas no desenvolvimento do sistema. Neste contexto a área de IHC é responsável por ditar as regras de projeto, implementação e validação das interfaces (CIBYS, 2007).

2.2.1 Projeto de Interface

O desenvolvimento de uma interface pode ser considerado um ponto muito importante no que diz respeito ao sucesso do projeto, pois as telas acabam sendo os únicos elementos do sistema que o usuário visualiza. Através da prototipação da interface, o usuário consegue ter uma ideia mais precisa do que os desenvolvedores do projeto estão propondo, e essas interfaces são importantes também para que seja feita uma análise mais precisa do nível de complexidade do projeto (CIBYS, 2007).

Para que seja desenvolvida uma interface de acordo com as normas do IHC, é preciso que seja levado em conta algumas dicas e regras para interfaces desktop e web, como será demonstrado nos itens que seguem.

2.2.1.1 Design de interfaces desktop e web

Para tornar concreta a interface do sistema, existe o design físico, o qual lida com questões específicas em torno do sistema, seriam elas: o *layout* da tela, os ícones que serão escolhidos e a estrutura dos menus (PREECE et. al. 2005).

A interface deve ser projetada de tal maneira que não entre em conflito com a percepção, atenção e memória, que seriam características humanas que o design deve ter em mente. Algumas recomendações muito úteis devem ser informadas ao design, para que o mesmo projete uma interface simples, organizada, com a finalidade de tornar a comunicação entre usuário e software a mais clara possível, seriam:

- Esforço pela consistência: projetar a tela que tenha um certo padrão para que o usuário consiga fixar a posição de certos objetos, como menus por exemplo. Questionar o usuário, antes de colocar em risco alguma ação que possa fazer com que o usuário perca algum dado importante.

- Utilização de atalhos pelos usuários mais assíduos: proporcionar para que o usuário, além das funções oferecidas por menus e ícones, possa executar essas funções também através das teclas de atalhos.
- *Feedback* informativo: desenvolver mensagens de erros que sejam exibidas aos usuários de uma forma clara e objetiva. Um exemplo seria, em vez de colocar um simples erro técnico do que aconteceu, colocar em linguagem simples o que aconteceu, no caso de procurar um arquivo e não encontrar, gerar a mensagem “Arquivo não encontrado”.
- Diálogos usados pelo sistema: informar ao usuário quando alguma ação foi realizada da forma correta, exemplo “arquivo salvo com sucesso”.
- Permitir a reversão de ações: oferecer mecanismos na tela, na qual o usuário consiga desfazer alguma ação que ele possa ter feito por engano.
- Diminuição da carga de memória de curto prazo: oferecer mecanismos na qual o usuário não precise gravar todo o *layout* da tela para realizar uma determinada tarefa no caso de uma mudança de tela.

Alguns aspectos gerais também devem ser levados em consideração na prototipação da interface:

- Evitar as escalas nos tamanhos das telas, as mesmas deverão ter a mesma dimensão da tela original.
- Apresentar o desenvolvimento do *layout* da tela em um monitor similar ao do usuário.
- As cores são um fator importante e merecem atenção. O usuário deve ser questionado e interagir sobre as cores que serão utilizadas no sistema.
- Os textos devem ser legíveis, obedecer um tamanho de fonte pré-estabelecido, devem ser usadas cores de textos que se diferenciem e deem um contraste com a cor do fundo onde os mesmos serão aplicados.
- Se for utilizado algum recurso adicional do tipo visual ou sonoro, deve haver a opção de utilizá-lo ou não.

As caixas de diálogo, ícones, menus, barras de ferramentas e outros elementos de interface são construídos por *widgets*² (componentes de interface gráfica com o usuário).

² Widgets – são componentes de interface gráfica com o usuário (GUI), Ex. botões, ícones, menus, janelas.

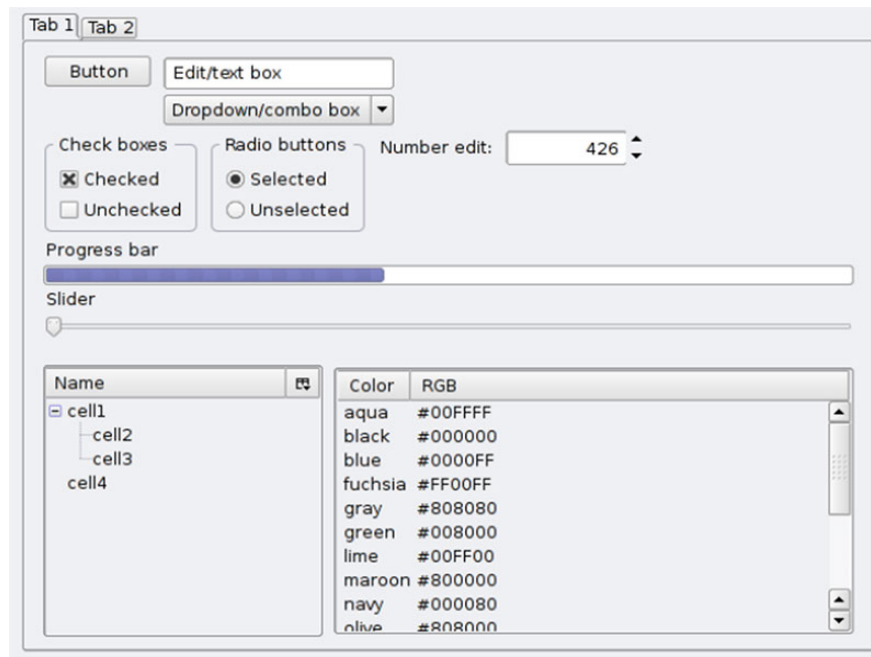


Figura 7 Interface desktop com widgetst (<http://www.slideshare.net/fabianodamiati/ihc-aula15>)

Outros aspectos são considerados principais no design de interfaces, esses podem ser aplicados em interfaces padrão (desktop) com facilidade.

Os menus fornecem aos usuários as opções de selecionar alguma tarefa diretamente, ou alguma opção que se relaciona indiretamente a algum comando.

O design que envolve o desenvolvimento do menu pode parecer ser uma tarefa simples, mas para que o menu consiga satisfazer o usuário, alguns detalhes devem ser levados em consideração, como por exemplo: os menus suspensos (*drop-down*) e instantâneos (*pop-up*), contêm algumas funções que são mais utilizadas e mais importantes do ponto de vista da utilização do sistema, por isso precisam estar disponíveis nas primeiras opções no topo, para evitar que o usuário perca tempo procurando (PREECE et. al. 2005).

A Figura 8 demonstra um exemplo que contém dois menus com funções para o usuário interagir com o software, a primeira imagem é considerada um menu mais seguro, por ter seus itens melhor distribuídos conforme a função de cada um; já a segunda imagem é considerado menu não seguro pois exhibe itens que não poderiam estar juntos, a figura ilustra que a função *Save* (Salvar) encontra-se muito perto da função *Quit* (Sair), a função *Save* deveria estar inserida mais ao topo do menu como demonstra a dica de interface de menu citada no parágrafo anterior, essa proximidade de funções que causam resultados opostos, poderá fazer com que o usuário cometa erros.

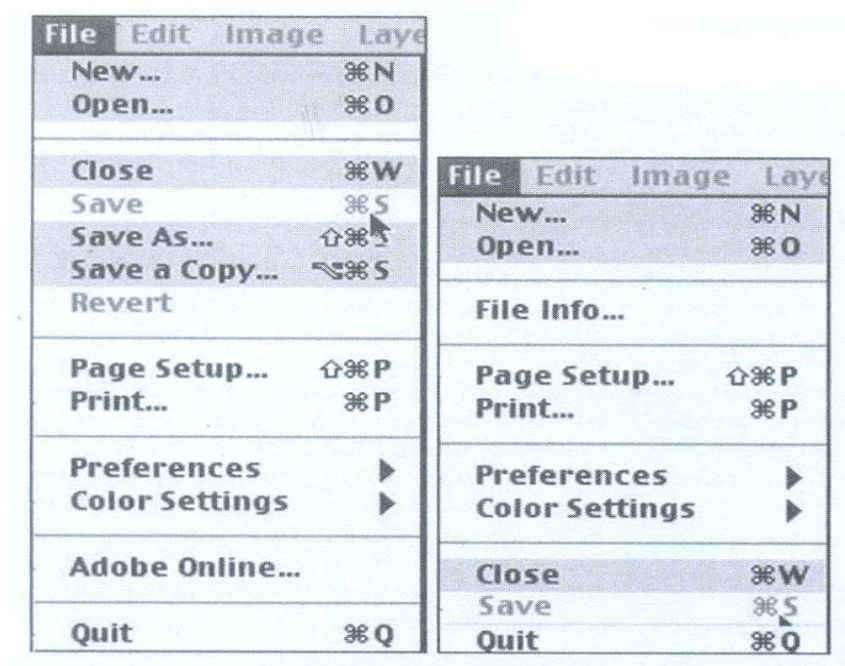


Figura 8 Exemplo de um menu seguro e um não seguro (PREECE et. al. 2005).

Os ícones também têm um papel fundamental no desenvolvimento da interface do sistema. Para a criação de um ícone que seja aceito pelo grupo de usuários de um sistema, consome um tempo considerável, pois o ícone deve ser de fácil entendimento e sua imagem deve ter uma relação bem associada com a função que o irá desempenhar. O ícone deve possibilitar que o usuário consiga fazer uma descoberta rápida do seu significado.

Os tamanhos dos ícones deverão sempre obedecer a um padrão, no qual ele consiga expressar sua complexidade, por exemplo: se um ícone for muito detalhado e o seu tamanho em relação a tela for muito pequeno, o mesmo terá dificuldade de representação, esse detalhe deixa o usuário com dificuldade de diferenciar esses detalhes no ícone.

2.2.1.2 Design da Tela

Um outro aspecto que deve ser contemplado no projeto da interface é o design da tela, dois aspectos são importantes: como as telas individuais são projetadas e como uma mesma tarefa pode ser dividida numa diferente tela.

A divisão de uma tarefa em diferentes telas deve mostrar todas as informações importantes, nos momentos relevantes do sistema. Esse fator acaba sendo uma ponderação que deve ser feita no design da tela, por exemplo, é necessário analisar as tarefas que serão colocadas na tela, pois não deve-se desenvolver uma tela confusa com muitas tarefas, o que

deixaria o usuário com dificuldade de memorização das mesmas, outro ponto que requer a atenção é a divisão das tarefas em várias telas, deve ser tomado cuidado em desenvolver várias telas dividindo a complexidade da tarefa para que a mesma não fique com uma complexidade muito simples, isso acaba sendo tão desmotivador quanto uma tela carregada de tarefas.

Para o planejamento do design de telas individuais, alguns outros princípios visuais devem ser contemplados, a utilização de cores leves e agradáveis em letras, tabelas, gráficos, botões que demonstrem uma visualização agradável. Animações, caixas e agrupamentos, são detalhes que quando bem projetados, captam a atenção do usuário para pontos importantes do sistema, sempre tendo o cuidado para que as animações sejam usadas de uma forma ponderada para que as mesmas não tirem a atenção do usuário.

O design da tela precisa ser projetado de uma forma organizada, os agrupamentos acabam contemplando essa organização, devem ser feitos agrupamentos de elementos semelhantes, e construir uma separação para que os itens semelhantes não se relacionem com itens diferentes, para que seja feita estas separações de agrupamentos poderá ser utilizado o uso de cores que se diferenciem, o uso de molduras ou estruturas (PREECE et. al. 2005).

Após as técnicas e dicas para um bom desenvolvimento de interfaces desktop, será descrito a seguir alternativas para que possam ser desenvolvidas interfaces voltadas para web, que consigam uma melhor interação com o usuário.

Segundo (NIELSEN, 2000), nos sistemas voltados para o ambiente Web o espaço da tela precisa ter um design que se preocupe com o conteúdo específico da página em desenvolvimento, pois em média os usuários ficam poucos segundos de visita em uma página, desta forma é necessário que o *layout* da tela ofereça, mecanismos para que o usuário se detenha maior parte do tempo em conteúdo específico na página, isso não acontece, pois normalmente somente 20% dos pixels (é a menor parte de uma imagem, quanto mais pixels, mais resolução terá uma imagem) da página estão exibindo conteúdo desejável.

A Figura 9 mostra um layout de site que desperdiça a maioria dos pixels de sua tela, com elementos que não são os mais importantes para o usuário em termos de conteúdo específico do site.

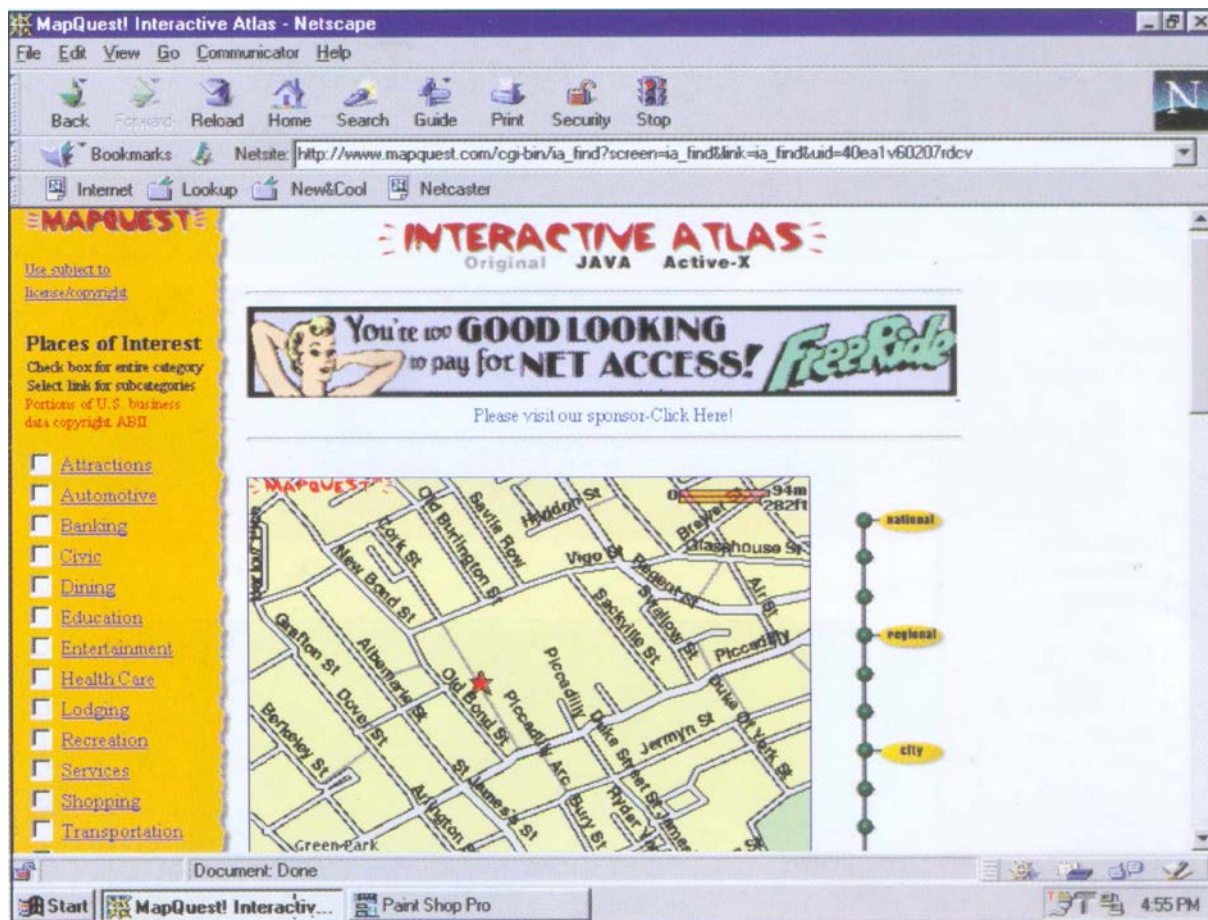


Figura 9 Modelo de site com seu espaço de tela mau utilizado (NIELSEN, 2000).

O *layout* de uma página contém espaços em branco que fazem parte do seu conteúdo. Eles não devem ser considerados áreas inúteis, em algumas vezes os espaços em branco esteticamente são melhores representados do que uma linha ou traço demarcador, os mesmos contribuem para um carregamento mais rápido da página e ajudam a adaptar a página a vários tamanhos de exibição (NIELSEN, 2000).

No desenvolvimento de uma interface desktop, não existem surpresas para o desenvolvedor, pois a tela será demonstrada para o usuário conforme projetada, pois não existem fatores do sistema que contribuirão para a mudança de visualização da mesma. Na web existem alguns fatores que poderão influenciar em mudanças no resultado do layout da tela, esses fatores podem ser navegadores web, fontes que falta no computador do usuário (NIELSEN, 2000).

As barras laterais nos sites demonstram que existe conteúdo que ultrapassa a área visível da tela, permite que seja possível deslocar a janela para alcançar este conteúdo. Mas deve-se ter cuidado ao usar as barras de rolagem, pois apenas 23% dos usuários que navegam por um site pela primeira vez usam a barra de rolagem nesta página, e nas visitas futuras essa

porcentagem pode cair ainda mais. Com a análise desses fatos é recomendado o desenvolvimento de páginas menores as quais consigam mostrar todo seu conteúdo sem o emprego de barras de rolagem, caso não for possível fazer o emprego do mínimo possível das mesmas (NIELSEN, 2000).

A figura 10 demonstra um exemplo de emprego de *links* que podem aumentar o uso do site. O texto a seguir dará algumas dicas no emprego de *links* nas páginas.



Figura 10 Modelo de site mostrando vários links na mesma série (Nielsen, 2000).

O emprego de *links* é muito útil e essencial para que os internautas consigam se conectar a várias páginas e visitar sites. Existem algumas formas diferentes links:

- Existem os links de navegação estrutural, através de um link ou botão dentro da página o usuário consegue navegar para outros sites subordinados a página atual.
- Links associativos são links dentro do conteúdo da página, geralmente são representados por palavras sublinhadas e ligam para outras páginas que contém um conteúdo sobre o texto principal da página.

- Links que são listas de referências adicionais, ajudam o usuário a encontrar conteúdos similares ao que foi solicitado caso a página atual não seja localizada.

Para que os links sejam melhores representados e melhor entendidos pelos usuários é preciso que seja usado um padrão de cores, para sites ainda não visitados deve-se usar a cor azul e para links que o usuário ainda não navegou de ser empregada a cor roxa ou vermelha (NIELSEN, 2000).

O uso de folhas de estilo é uma técnica cada vez mais empregada no desenvolvimento de interfaces web, pois as mesmas não prejudicam o carregamento da página, e contribuem para o design da página, é uma tendência que vem substituindo o emprego de tabelas, pois no emprego de tabelas é necessária a utilização de tabelas com estruturas simples para não aumentar o peso de carregamento da página.

O usuário deve ter a possibilidade de poder navegar por todas as páginas existentes dentro do site conforme o mesmo espera, para isso acontecer a disposição dos elementos da página desempenha um papel significativo. O usuário deverá ter a possibilidade de navegar internamente pelo site independente de onde estiver. Algumas técnicas devem ser empregadas, como: utilizar menus que contenha opções que condizem com as informações do conteúdo, e os tópicos localizados dentro do menu devem ser breves e ter significado, e para torná-lo mais intuitivo associá-los com imagens que possam representar a sua utilização (NIELSEN, 2000).

Algumas regras fundamentais têm que ser levadas em conta quando se trata da legibilidade do texto num site. Um ponto importante são as cores do texto em relação ao fundo do site, devem ser utilizadas cores que causem contraste visual, o ideal é o emprego de cores como o texto branco aplicado sobre um fundo preto ou ao contrário, mas deve-se evitar cores nas quais o fundo e as letras possam se confundirem, um exemplo citado por Nielsen, seria o texto rosa aplicado a um fundo verde, seria duas cores que não contém contraste se aplicadas juntas, no caso de usuários daltônicos tornaria-se impossível a leitura.

As fontes devem obedecer um tamanho que seja possível a leitura, mesmo para usuários que não possuam uma visão considerada perfeita. Evitar o uso de fontes maiúsculas em textos mais extensos, pois isso pode dificultar a leitura do usuário.

Evitar animações, possibilidades de mover, *zoom*, esses detalhes dificultam uma boa leitura, tente sempre manter o texto imóvel, para que o usuário não tenha dificuldades de percepção.

O uso de imagens para fundos onde sejam aplicados textos, interferem na capacidade leitura, procure sempre usar cores lisas, ou cores de fundo que sejam leves.

O emprego multimídia no desenvolvimento de interfaces web ganha, cada vez mais espaço, para complementar o emprego de textos e imagens estáticos. Esses meios requerem uma certa ponderação, pois o uso abusivo desses recursos dificultam a compreensão dos conteúdos principais do site por parte dos usuários (NIELSEN, 2000).

A figura 11 exibe link para acesso a vídeos, esses resumos de cenas de vídeo ajudam o usuário a decidir se tem interesse no download.

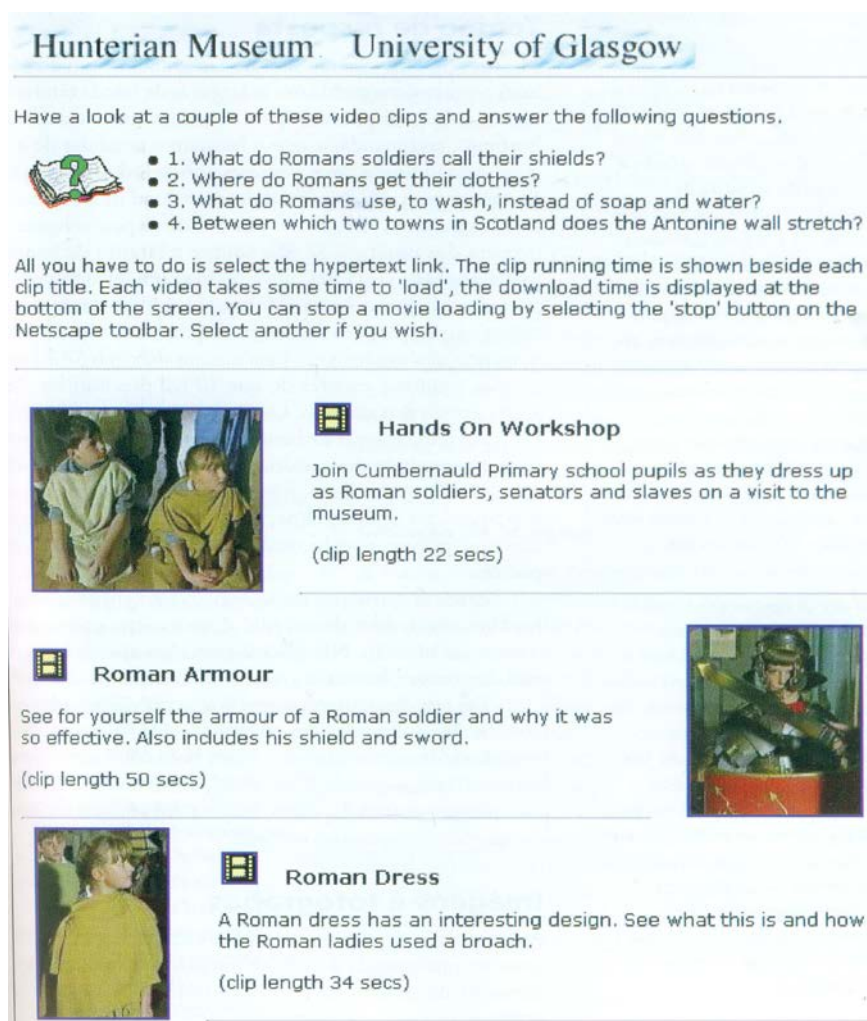


Figura 11 Layout de site que exibe de forma clara o emprego de multimídia (NIELSEN, 2000).

Segundo NIELSEN (2000), uma imagem equivale a duas mil palavras em tempo de *download*. Esta colocação leva o desenvolvedor a ter um melhor controle da quantidade de elementos gráficos em uma página web, para isso é recomendável evitar o uso de textos renderizados (textos em forma de imagem, escaneados), fugindo a regra quando forem legendas de imagens, que muitas vezes já vem inseridas junto com a mesma. Uma outra maneira para amenizar este problema seria aproveitar os recursos de hipertexto, esta técnica pode ser aplicada de forma que no conteúdo da página principal teria o mínimo de ilustrações

e mais textos, e na medida que o usuário mostre interesse no conteúdo, o mesmo se dirigiria a páginas mais específicas do assunto as quais mostrariam mais ilustrações, pois neste estágio o usuário já demonstrou uma real interesse no assunto.

Um ponto bastante questionado é a largura da página do site, na verdade não é recomendável desenvolver página para uma largura padrão, na verdade o apropriado seria o projetar a interface para uma largura que sirva para o máximo de usuários possíveis, mas um cuidado que deve ser tomado é para que não haja uso de barras horizontais nessas páginas o que causa uma certa irritação aos usuários (NIELSEN, 2000).

Na seção a seguir será abordado o assunto de prototipação, o qual fornece uma visão avançada em relação ao projeto, tanto para o usuário como para o desenvolvedor, pois os dois retiram do protótipo informações para as suas atividades. O desenvolvedor tem uma visão de como será o futuro do sistema e começa o processo computacional de desenvolvimento do que será implementado naquela aplicação, e o cliente irá começar a visualizar as vantagens que passará a ter com a utilização do software.

2.2.3 Protótipos

A ideia de um protótipo a nível de *design*, pode ser definida de uma maneira bem ampla, o mesmo pode variar desde um simples esboço de tela em um simples papel, até uma sofisticada tela de um sistema complexo, desenvolvida em um software específico para prototipação. Um protótipo possibilita que os *stakeholders* interajam com o produto imaginado por eles, fazendo com que o mesmo fique mais ambientado e obtenha experiência de como utilizar o software num ambiente real (PREECE et. al. 2005).

A nível de desenvolvimento de sistemas, o protótipo pode ser considerado como a primeira versão de um sistema, o mesmo ajuda os usuários a ter ideias novas para os requisitos e com isso achar as partes boas e as partes deficitárias do sistema. No contexto de custo, os protótipos podem ajudar a diminuir os mesmos. Isso é possível porque o usuário estará inserido em todas as etapas de desenvolvimento, causando assim menores custos caso o *software* precise ser ajustado (SOMMERVILLE, 2008).

A tabela 1 classifica os protótipos de acordo com a fidelidade, ou seja, o quanto o protótipo consegue se assemelhar ao projeto final e dar uma visão do software que está em desenvolvimento (PREECE et. al. 2005).

Tabela 1 Classificação dos Protótipos em Baixa Fidelidade e Alta Fidelidade

Baixa Fidelidade	Alta Fidelidade
<p>Protótipos que contém pouca semelhança ao produto que está sendo desenvolvido</p> <p>Não contém muita complexidade</p> <p>São baratos e sua produção ocorre de forma rápida</p> <p>Esse modelo de protótipo é utilizado no início dos projetos</p>	<p>Desenvolvido para comercializar sugestões mais claras ligadas sobre o software</p> <p>Utilizado para análise de questões mais técnicas sobre o sistema</p> <p>Protótipo após concluído se assemelha muito ao produto final</p> <p>Define de forma clara ao cliente como será a utilização do software</p>

Como exemplo de um protótipo de baixa fidelidade, encontra-se o *Storyboard*, que na maioria das vezes é utilizado juntamente com os cenários (citado no item 2.6.1). Quando usados juntamente oferecem de forma mais eficaz aos *stakeholders*, uma maneira de simular uma interação com o produto mediante o cenário. O *Storyboard* demonstra através de desenhos numa ordem sequencial a progressão em uma determinada tarefa utilizando o produto que está sendo desenvolvido, podem ser telas em forma de esboço, no caso ser um sistema baseado em interfaces gráficas (PREECE et. al. 2005).

A figura 12 mostra um modelo de baixa fidelidade que corresponde a um exemplo de *Storyboard*.



Figura 12 Exemplo de um Storyboard (PREECE et. al. 2005).

O exemplo mostrado na figura 13 corresponde a um *Wireframe*³ de alta fidelidade, podem ser construídos modelos de *Wireframes* de alta, média e baixa fidelidade, isso dependerá dos níveis de complexidade e das imagens que serão inseridas no mesmo. Consiste num documento que contém uma estrutura e conteúdo da interface, que indica o peso e a importância dos elementos do *layout*, e a sua relação com todos os elementos formadores.

The wireframe illustrates a web page structure. At the top is a 'Header' bar containing a search input field and five buttons labeled 'Option 1' through 'Option 5'. Below the header is a 'Page Heading (h1)' followed by a 'Section Heading (h2)'. Under the h2 heading is a paragraph of placeholder text: 'This is text providing any contextual help related to this section. In general, it should not need to be more than 2 or 3 sentences long. (p.instructional)'. The main content area contains a form with several fields: a text input, a dropdown menu, and four checkboxes, each preceded by a 'Field Name (label)'. Below these are four radio buttons, also each preceded by a 'Field Name (label)'. To the right of the main form is a 'Side Section (h3)' containing four blue links: 'Local Option 1', 'Local Option 2', 'Local Option 3', and 'Local Option 4'. At the bottom of the page, there are two buttons: 'Continue (input.mainButton)' and 'Cancel (input.button)'. The 'Continue' button is highlighted with a yellow border.

Figura 13 Exemplo de Wireframe Alta fidelidade, demonstrando a organização dos elementos

Os protótipos podem ser classificados quanto aos seus objetivos, sendo eles: protótipos horizontais e verticais.

Os Protótipos Horizontais consistem em demonstrar uma interface de software completa em termos de elementos, demonstrando superficialmente toda a interface, num protótipo horizontal o usuário tem uma visão geral e este tipo de protótipo permite testar a interface como um todo. Este modelo é utilizado como um protótipo inicial.

³ Wireframe é uma representação básica demonstrada por um desenho, que demonstra de forma direta a arquitetura de como o objeto (interface, página da internet) final será de acordo com as especificações relatadas.

Já os Protótipos Verticais focam as funcionalidades do sistema. Este modelo de protótipo permite que sejam testados módulos isolados do sistema. São implantadas poucas tarefas nesse protótipo, mas com funcionalidades aprofundadas (NIELSEN, 1993).

A área de desenvolvimento de software contém muitos riscos em sua implantação, um dos riscos pode ser a omissão de requisitos nas fases iniciais, o custo de correção de erros de requisitos em fases posteriores pode ser muito alto. Nesse contexto é que a prototipação reduz os problemas ligados a identificação dos requisitos, além disso, os custos totais poderão ser mais baixos se a técnica de prototipação for implementada (SOMMERVILLE, 2004).

Um segmento muito importante na área de prototipação, é que a mesma é utilizada como essencial ferramenta que estabelece uma ligação entre usuários e designers (NIELSEN, 1993).

A figura 14 demonstra as quatro fases no processo de prototipação, que no decorrer de suas iterações proporciona um ajuste fino no protótipo em desenvolvimento, isso possibilitará análises corretas dos requisitos do sistema com maior segurança para o produto final ser desenvolvido.

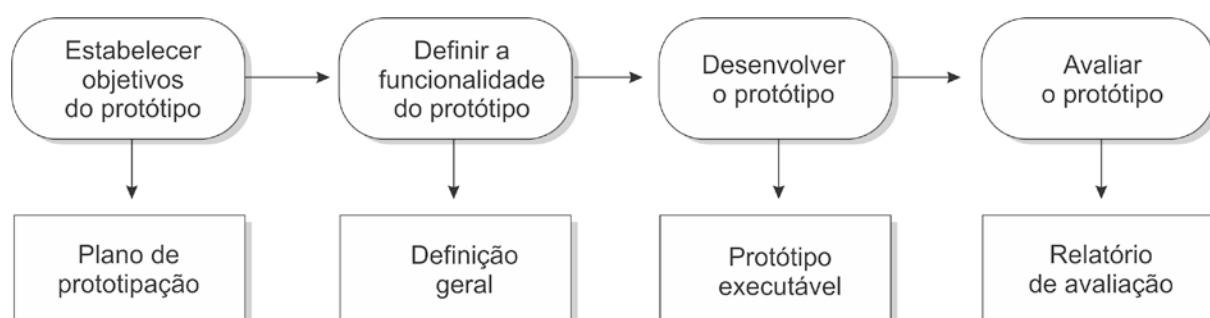


Figura 14 Processo de desenvolvimento de protótipo (SOMMERVILLE, 2008)

As etapas demonstradas na Figura 14 mostram de forma sequencial as atividades que são realizadas no processo de prototipação. Nos itens a seguir encontra-se a definição de cada uma das atividades (SOMMERVILLE, 2004):

- **Objetivos do protótipo:** o desenvolvimento da interface pode ser um dos objetivos dessa etapa. Além disso, o mesmo pode ser o desenvolvimento do protótipo para mostrar a viabilidade do software para o cliente ou gerente do projeto

- Definição das funcionalidades do protótipo: nesta etapa será decidido o que será inserido no sistema protótipo, e serão aprovadas e documentadas quais as funcionalidades mais importantes devem fazer parte do protótipo.
- Desenvolver o protótipo: através das especificações das fases anteriores é construído o protótipo.
- Avaliação do protótipo: nesta última etapa é o momento que será realizado um treinamento de usuário, e um plano de avaliação que utiliza os objetivos do protótipo.

2.2.4 Avaliação de IHC

Os métodos de avaliação de interfaces não devem ser considerados como uma única fase inserida no processo de design. A avaliação deve estar presente durante todo o ciclo de vida do design, para que se possa ter uma melhoria crescente no projeto da interface (OLIVEIRA, 2006).

A seguir serão descritos alguns testes de avaliação, que ajudarão ao design observar o usuário, para que com os dados que serão coletadas através dos testes, possam ser feitas melhorias no projeto da interface.

Os Testes de Usabilidade, por exemplo, tem por finalidade avaliar de uma forma quantitativa e individualmente o nível de usabilidade atingido. Testes práticos são aplicados a fim de avaliar o desempenho dos usuários típicos com tarefas bem preparadas para que seja testado o sistema que está em projeto (OLIVEIRA, 2006).

Uma característica importante no teste de usabilidade é que o mesmo é controlado pelo avaliador. Estes testes são realizados em locais apropriados, e os participantes ficam de certa forma isolados para que o mesmo possa estar bem concentrado para que os testes que serão ser aplicados. Qualquer ação que o participante tenha será anotada, para posteriormente serem usadas como dados para melhoria do processo, tais ações como toques de teclas, pausas, expressões, tempo para realizar determinada tarefa, tudo será levado em conta. Os participantes dos testes podem ser filmados, e todas suas interações captadas através de um software (PREECE et. al 2005).

O registro das ações geram subsídios de desempenho, isto pode ajudar na identificação de erros, e também a explicar a causa dos usuários terem feito determinada ação (PREECE et. al 2005).

Outra prática é a avaliação heurística, que é aplicada através de métodos de avaliação onde profissionais da área técnica fazem uma análise de interfaces de softwares. São utilizadas heurísticas testadas e experimentadas que foram desenvolvidas para avaliar protótipos de interface de softwares. As avaliações heurísticas não utilizam somente um conjunto de heurísticas, as mesmas podem ser modificadas, dependendo do software que está sendo projetado (PREECE et. al 2005).

Com a finalidade de avaliar a usabilidade de um produto, o método de avaliações heurísticas, pode ser aplicado em qualquer uma das fases do ciclo de vida do projeto, se for utilizado nas fases iniciais fará um maior impacto no design da interface.

Uma abordagem alternativa para a avaliação heurística é o percurso cognitivo, o qual consiste em um avaliador interagir com a interface do projeto, colocando-se no lugar o usuário e analisando se existe algum problema, o avaliador tenta prever se o usuário terá alguma dificuldade de interagir com a interface. Os percursos cognitivos apresentam os seguintes passos:

1. É preciso registrar as características dos usuários, para que se possa desenvolver tarefas ligadas aos aspectos de design que serão avaliados. Projeta-se um protótipo de interface, e as ações de uma forma sequencial para o usuário completar a tarefa.
2. É necessário reunirem-se designer e avaliadores especialistas para que sejam realizadas as análises.
3. Desenvolver um *feedback* com os usuários através de questões, as mesmas analisarão se o usuário sabe fazer e se entende o que está fazendo.

Durante o percurso deve ser realizado um registro das informações importantes, abrangendo o que está funcionando de forma correta e o que não está funcionando (PREECE et. al 2005).

Esta técnica contém detalhes muito importantes, dentre eles, o fato de descrever de forma detalhada as dificuldades e problemas dos usuários, isso sem que o usuário esteja presente.

Outro método que pode ser mencionado para a avaliação de interfaces são as *Checklists*, listas detalhadas que são utilizadas para a verificação através da qual designer ou profissionais que não necessitam ser especialistas, diagnosticam problemas gerais que possam existir nas interfaces (CIBYS, 2007).

Existem no mercado muitas ferramentas que aplicam algumas das técnicas usadas acima. No capítulo seguinte serão apresentadas algumas ferramentas de prototipação de interface e suas características.

3 ANÁLISE DE FERRAMENTAS DE PROTOTIPAÇÃO

Para a geração de interfaces utilizam-se ferramentas de prototipação, as quais são utilizadas nos estágios iniciais de *design* quando são feitas avaliações que preveem a possibilidade de utilizar-se o produto ou parte dele. Tais ferramentas são de grande utilidade para testes de novas ideias de maneira informal (PREECE et. al., 1994).

Essas ferramentas precisam possibilitar rapidez para quem está desenvolvendo um protótipo, por isso elas precisam ser intuitivas, para que os usuários possam utilizá-las de uma maneira eficaz (PREECE et. al., 1994).

A seguir será feita a descrição de algumas ferramentas com suas devidas características. Com o objetivo de auxiliar a análise entre as ferramentas, foi desenvolvido como exemplo, uma interface que simula um cadastro de cliente.

3.1 Designer Vista

O Designer Vista (disponibilizado em <http://www.designervista.com/>), é uma importante ferramenta e de fácil manuseio, através dela é possível gerar bons protótipos de interface.

Através desse software é possível gerar protótipos para Web e Desktop, o mesmo é executado em uma plataforma de trabalho que usa ambiente Windows. O Designer Vista é um software proprietário, apenas é permitido ao usuário baixar uma versão demo ou uma versão para avaliação.

Características gerais:

- É um software de fácil manuseio, não é necessário experiência para operá-lo.
- O Designer Vista utiliza um grande número de controles nativos, *widgets*, cliparts, ícones, dentre outros já embutidos no aplicativo. Esta ferramenta disponibiliza o recurso para utilização de *Templates*, é possível que se coloque elementos comuns em um modelo de página de GUI e *link* desta página em outras páginas. Não há necessidade de copiar os elementos comuns em cada página.
- A visualização da interface é exibida em vários temas: Vista, Sketch e Plain, com a possibilidade o usuário personalizar os temas existentes ou criar novos.
- Permite a importação de dados do Microsoft Excel.

- O usuário poderá exportar os projetos como arquivos HTML com *links* para distribuição fácil.
- Através desta ferramenta é possível a geração de protótipos de baixa fidelidade.

Na figura 15 é demonstrada a interface construída através da ferramenta Designer Vista.

Figura 15 Tela de Cadastro de Clientes desenvolvida no Software Designer Vista

3.2 Gui Design Studio

A ferramenta GUI Design Studio (<http://www.carettasoftware.com/>), é caracterizada por ser um aplicativo de design de interface de usuário e uma ferramenta de prototipagem rápida, que não precisa de nenhum código. Dentre as principais características, destacam-se:

- Plataforma de trabalho: é possível gerar protótipo para Web e Desktop.
- Plataforma operacional: executa em ambiente Windows.
- Licença: é um sistema pago, está disponível para download gratuito por 30 dias. Apenas o visualizador que possibilita verificar o projeto é gratuito.

- Possibilita que um elemento da página possa ser associado a outro elemento existente em outra página do software, possibilitando com isso a navegação entre as páginas;
- É possível que sejam feitas anotações e associá-las aos elementos do protótipo;
- Gerar uma documentação do protótipo em HTML
- Exportação de imagens das telas do protótipo para formatos como jpg e png;
- Sem limite para desfazer, muito útil no desenvolvimento de interfaces;
- Disponibilidade dos comandos: cortar, copiar, colar e duplicar;
- A ferramenta possibilita que os elementos criados possam ser alinhados e possam ser controlados seus espaçamentos entre outros objetos da tela;
- Como não há codificação envolvida, design de produto torna-se acessível a todos, especialmente os designers de interface de usuário e especialistas que dela necessitam.

Na figura 16 foi desenvolvido um protótipo de interface com base na ferramenta descrita neste tópico.

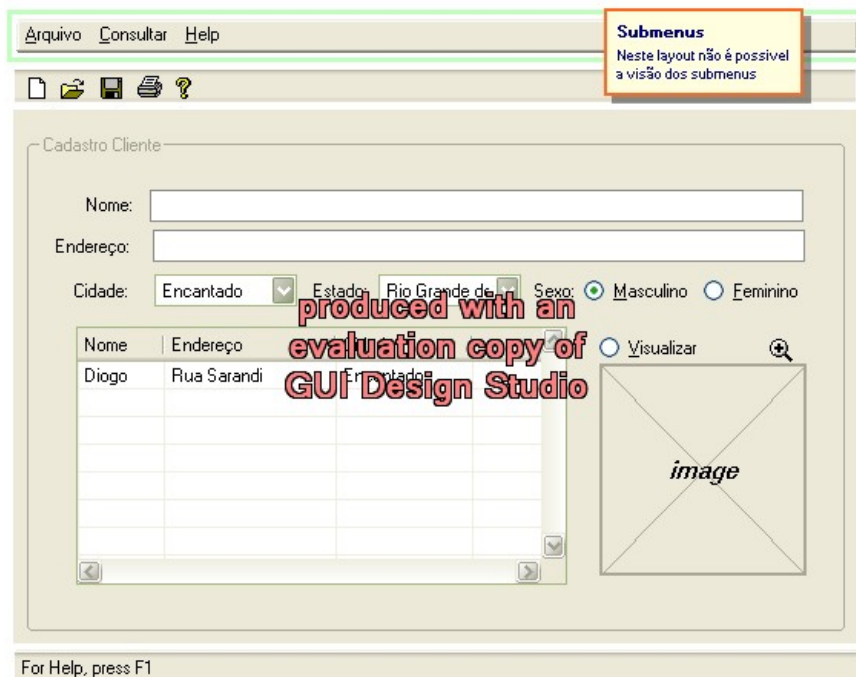


Figura 16 Tela de Cadastro de Clientes desenvolvida no Software Gui Design Studio

3.3 Wiremaster

A ferramenta de prototipação de interfaces Wiremaster encontra-se disponível para download em (<http://sourceforge.net/projects/wiremaster>).

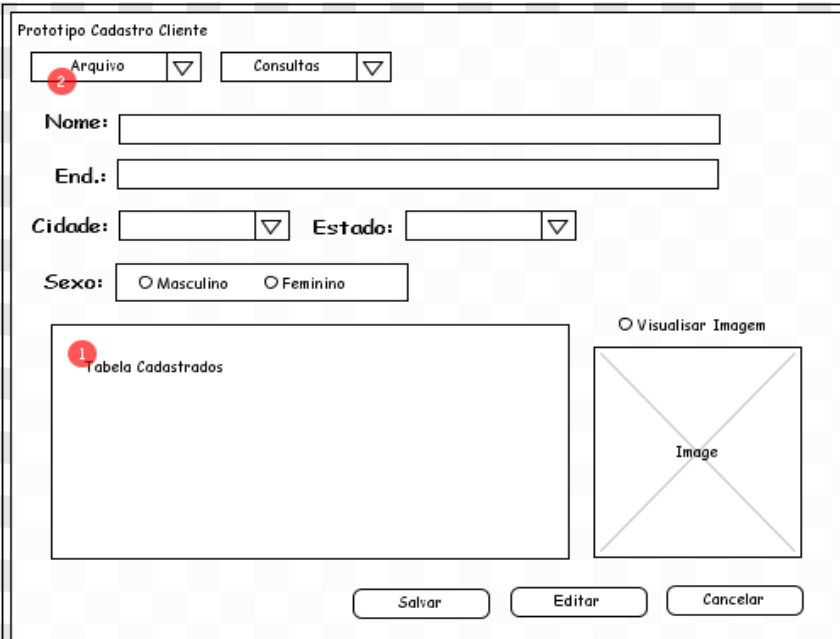
A seguir algumas características que definem o Wiremaster:

- Plataforma de trabalho: é possível gerar protótipo para Web.
- Plataforma operacional: roda em ambiente Windows e Linux, ferramenta open source desenvolvida na linguagem Java.
- Possui uma interface intuitiva
- As funcionalidades são de fácil acesso e apresentadas na tela principal, o que torna rápido o desenvolvimento de protótipos.

As principais limitações deste software são:

- Oferece poucos elementos de tela para serem utilizados na prototipação, que pode tornar o protótipo de interface não muito atrativo;
- Dificuldade de manusear os elementos adicionados na tela;
- Não permite adicionar comentários para documentação das telas;
- Não fornece a possibilidade de fazer associações entre as telas desenvolvidas;
- Sem opção de simulação da navegação.

Na figura 17 o Wiremaster apoiou o desenvolvimento de uma interface de cadastro.



Prototipo Cadastro Cliente

Arquivo Consultas

Nome:

End.:

Cidade: Estado:

Sexo: ☐ Masculino ☐ Feminino

1 Tabela Cadastrados

Visualisar Imagem

Image

Salvar Editar Cancelar

Detailed description: This is a wireframe of a 'Cadastro Cliente' (Client Registration) form. At the top, there are two dropdown menus labeled 'Arquivo' and 'Consultas'. Below these are input fields for 'Nome', 'End.', 'Cidade', and 'Estado'. A 'Sexo' section contains two radio buttons for 'Masculino' and 'Feminino'. On the left, there is a large rectangular area labeled '1 Tabela Cadastrados'. On the right, there is a placeholder for an image, labeled 'Image', with a diagonal cross and the text 'Visualisar Imagem' above it. At the bottom, there are three buttons: 'Salvar', 'Editar', and 'Cancelar'. Red circles with numbers '1' and '2' are placed over the table area and the 'Arquivo' dropdown respectively.

Figura 17 Tela de Cadastro de Clientes desenvolvida no Software Wiremaster

3.4 Axure RP Pro 5.6

Este software permite que os designers possam criar wireframes, protótipos e especificações para aplicações e sites web.

Seguem algumas características do software Axure:

- Plataforma de trabalho: é possível gerar protótipo para Web e aplicações desktop.
- Plataforma: é executado nativamente nas plataformas Windows e OS X. Os arquivos gerados pela Axure RP podem ser compatíveis com as duas plataformas citadas anteriormente desta forma é possível trabalhar compartilhado em seu projeto independente das plataformas e também permite que se crie um projeto na plataforma Windows e o mesmo possa ser aberto em uma plataforma Mac.
- Licença: software pago, gratuito somente para teste.

É uma ferramenta muito usada por profissionais da área no mundo inteiro. Contém peculiaridades de design necessárias para que seja possível obter rapidamente os benefícios da prototipagem sem problemas, tornou-se uma importante ferramenta para muitas equipes e organizações em uma ampla gama de projetos.

Permite fazer anotações wireframes para avaliar conceitos no início do processo. Possibilita adicionar cores, imagens, e uma vasta gama de interações com seu projeto como base de ligação para expandir e recolher o conteúdo. Gera uma especificação detalhada de seu projeto com *screenshots*⁴, interações e anotações para aprovação ou entregar para o desenvolvimento.

É possível também que sejam gerados protótipos HTML de forma interativa apenas com alguns cliques.

Podem ser desenvolvidas especificações funcionais para o formato Microsoft Word como imagens, anotações, e interações.

Através da ferramenta Axure RP, foi desenvolvido uma interface de cadastro para comparação com as outras ferramentas, a figura 18 mostra a tela desenvolvida.

⁴ Screenshot é a definição que é dada para a captura de uma tela do computador., pode ser uma imagem do próprio sistema, foto, uma captura de uma imagem de filme..

Arquivo Consultas Help

Cliente:

End.: Bairro:

Cidade: UF: Sexo: ☐ Masculino ☐ Feminino

Cod.	Cliente	Cidade	Sexo
01	Diogo de Jesus	Encantado	Masculino

☐ Visualizar Imagem

Salvar Editar Sair

Figura 18 Tela de Cadastro de Clientes desenvolvida no Software Axure RP Pro 5.6

3.5 Análise de Softwares da Prototipação de interfaces

Com base nas ferramentas apresentadas é possível constatar que existem opções que auxiliam na prototipação de interfaces de software. Porém, dentre as funcionalidades relatadas nos softwares, percebe-se que alguns não disponibilizam elementos de telas necessários para a criação de um protótipo de qualidade.

Dentre os elementos que foram analisados, constata-se que nenhum dos softwares analisados disponibilizam a funcionalidade para que sejam adicionados comentários aos componentes de tela ou ao protótipo desenvolvido, essa funcionalidade é de muita importância, pois esses comentários possibilitam um ótimo diálogo entre as pessoas envolvidas no projeto do sistema.

Outra funcionalidade que pode ser mencionada refere-se ao fato de todas as ferramentas citadas possuírem elementos de interface como ícones, menus, suficientes e atrativos para o desenvolvimento de um bom layout.

A tabela 2 apresenta uma comparação entre os quatro softwares acima citados de acordo com suas principais funcionalidades.

Tabela 2 Tabela comparativa entre principais funcionalidades dos softwares citados

Funcionalidades	Designer Vista	GUI Design Studio	Wiremaster	Axure RP Pro
Exportação em HTML	X			X
Exportação em imagem		X	X	X
Permite anotações nas telas	X	X		X
Simular Navegação	X	X		X
Exportação em XML	X	X	X	X
Variedade de elementos para criação da interface	X	X		
Visualização da tela criada	X	X		X
Licença	Software Proprietário. Versão para Teste	Software Proprietário. Versão para Teste	Software Livre e Multiplataforma	Software Proprietário. Versão para Teste
Diferencial	Permite importação de dados para Planilhas Eletrônicas	Teste de Navegação entre telas		Permite criar especificações para o formato word
Plataforma	Windows	Windows	Windows/Linux	Windows

A tabela 2 demonstra uma comparação das principais finalidades dos softwares citados. É possível constatar que essas ferramentas de prototipação de interfaces têm muitas funcionalidades em comum, apenas destaca-se o software não proprietário Wiremaster que possui menos funcionalidades na comparação com os outros 3 programas, apoiado nessa limitação de funcionalidades do software Wiremaster, abre-se a oportunidade de desenvolvimento de uma ferramenta não proprietária, que contemple algumas funcionalidades básicas como: exportação para XML, número considerável de elementos de interface, facilidade de navegação conforme os demais softwares, além de exportação para PDF e simulação de navegação.

4 IMPLEMENTAÇÃO SOFTWARE EASY SCREEN

Conforme exposto no capítulo 2, a prototipação de interfaces tem papel importante na concepção de um produto de software. Através de protótipos de interface que representam o futuro sistema, o diálogo entre desenvolvedor e cliente é facilitado, auxiliando nas demonstrações do produto, na detecção de erros de modelagem e projeto. O capítulo 3 apresentou um pequeno conjunto de ferramentas disponíveis atualmente, onde a análise de tais ferramentas somada a necessidade de construção de protótipos de interface, motivou o desenvolvimento do software Easy Screen, visando apoiar o processo de prototipação para ambientes desktop.

Nos tópicos a seguir será apresentada a implementação da ferramenta de prototipação de interfaces desktop Easy Screen. Primeiramente será mostrada a lista de requisitos que o sistema atende, logo após é demonstrada a estrutura e modelagem do sistema e para finalizar este capítulo, é feita a apresentação dos resultados da ferramenta juntamente com a explicação do funcionamento da mesma.

4.1 Modelagem do Sistema

O projeto desenvolvido fez uso da linguagem de programação JAVA SE versão 6.0, a escolha deu-se pelo fato da mesma ser multiplataforma não ficando restrita a somente um sistema operacional. Como ferramenta de desenvolvimento foi utilizado a IDE NetBeans 6.9, por ser uma ferramenta multiplataforma e por facilitar ações como: organização de código, auto-identação e proporcionar produtividade ao desenvolvedor.

A ferramenta Easy Screen utiliza componentes prontos Swing, disponibilizados pela linguagem Java. Para implementar algumas funções foram utilizadas bibliotecas disponíveis livremente, a fim de facilitar o desenvolvimento e manter o foco na área de negócio.

A figura 19 exibe uma visão geral da arquitetura do sistema, na qual exibe na parte direita os componentes internos que são usados para o desenvolvimentos dos protótipos de interface que são os componentes de tela e os ícones, na parte direita encontram-se as bibliotecas externas, a Xstream auxilia na gravação da tela desenvolvida em formato XML e a iText exporta as telas em formato PDF.

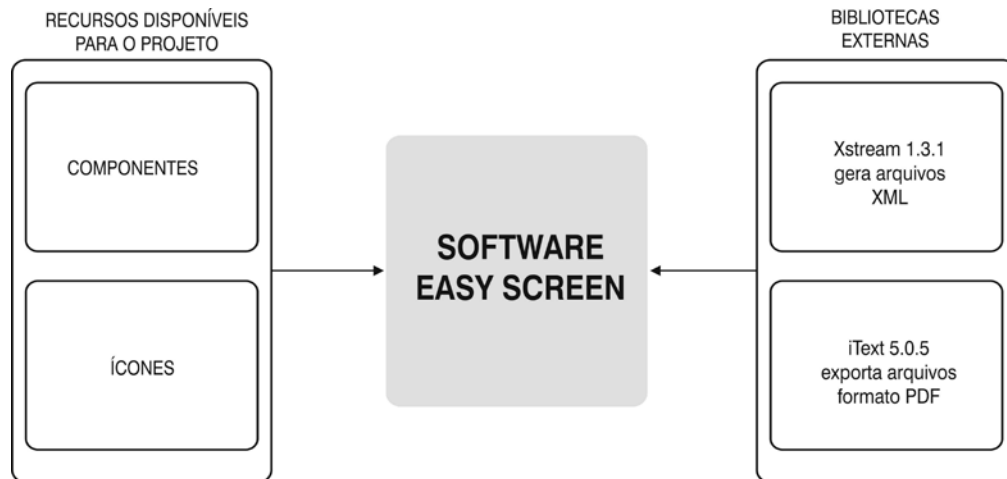


Figura 19 Arquitetura do software Easy Screen

4.1.1 Requisitos do Sistema

Na implementação da ferramenta Easy Screen foram contemplados vários requisitos, os quais foram fundamentais para o desenvolvimento da ferramenta, os mesmos descrevem serviços e funções que o sistema possui. Os requisitos que foram implementados no software Easy Screen são:

- **Criar Projeto e tela para desenvolvimento**

O software disponibiliza para o usuário, através do menu ou ícone na barra superior, a opção de criar nova tela para que seja desenvolvido o protótipo, antes de criar a tela é solicitado ao usuário o nome do projeto e após o nome da tela. A tela será salva dentro da pasta que terá o nome do projeto informado.

- **Desenvolver a interface desktop**

Esse requisito disponibiliza ao usuário a total interação com a ferramenta, o mesmo poderá inserir os componentes de tela disponíveis que necessitar, juntamente com os ícones que julgar necessário, fazendo uso do recurso de arrastar/soltar.

- **Inserir Imagens**

O requisito de inserção de imagens foi disponibilizado de maneira na qual o usuário insere um componente que simboliza uma imagem.

- **Editar propriedades dos elementos de tela**

Através deste requisito o usuário consegue editar os nomes dos componentes de tela, juntamente com a posição do componente e também modificar as propriedades de largura e altura do componente.

- **Adição de Comentários**

Esse recurso permite ao usuário inserir comentários gerais sobre a tela que foi desenvolvida e comentários específicos em cada componente que é inserido no projeto.

- **Gerar Visualização do Projeto**

O sistema dispõe de uma visualização geral do projeto com todas as telas que estão inseridas no mesmo. Permite que o usuário visualize as telas uma a uma, avançando e retrocedendo entre elas de acordo com a ordem que as mesmas foram inseridas ao projeto.

- **Exportar para XML**

Na solução implementada a tela desenvolvida é salva em formato XML.

- **Gerar imagem do Protótipo**

Esse requisito foi implementado com o objetivo de disponibilizar o recurso de gerar uma imagem da tela que está sendo desenvolvida na área de trabalho da ferramenta. Essa imagem é gerada no formato PNG. Todos os elementos que estão inseridos na área de trabalho são inseridos na imagem que será gerada.

- **Exportar PDF do Protótipo**

O requisito de gerar um arquivo no formato PDF da tela em desenvolvimento foi implementado para que o usuário possa gerar um arquivo de saída, o qual possa ser visualizado em qualquer sistema independente da plataforma de trabalho.

Somando-se a lista de requisitos exposta, a figura 20 apresenta um diagrama de casos de uso, o qual demonstra todos os requisitos que foram alcançados pelo sistema Easy Screen. Esse diagrama exhibe as funções que o usuário consegue desempenhar com o software.

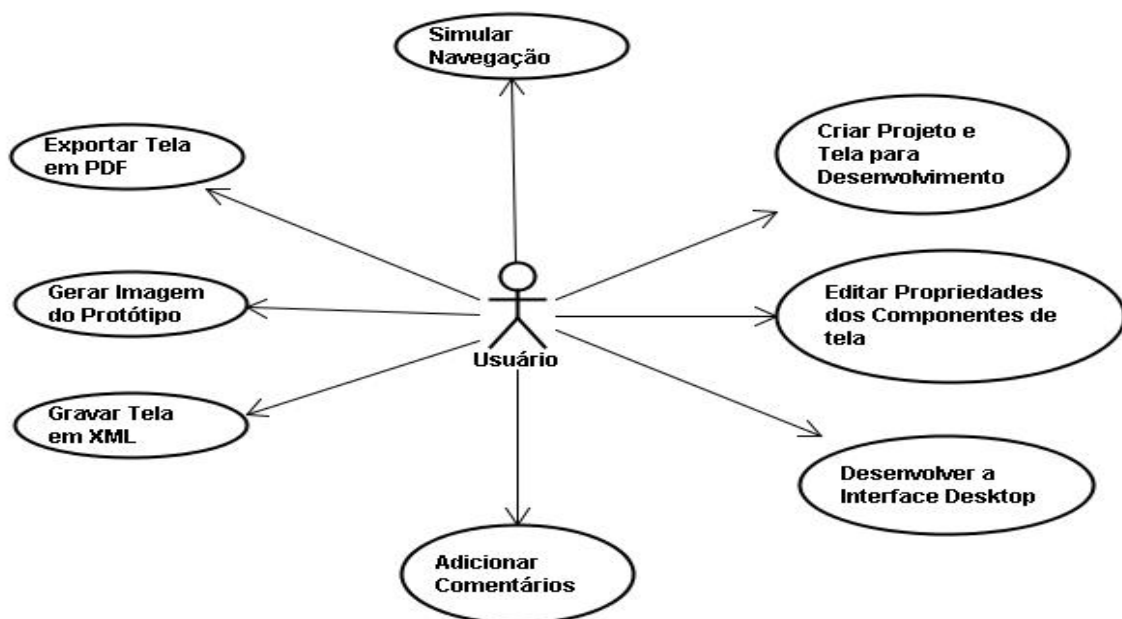


Figura 20 Diagrama de Casos contemplados no Sistema Easy Screen

4.1.2 Diagrama de Classes

Para melhor exemplificar o que foi implementado no sistema Easy Screen, é demonstrado a seguir um diagrama de classes. Neste diagrama são apresentadas somente as classes que foram desenvolvidas para o sistema, dando ênfase para as classes *GenericModel* e *jGenericComponent*. Em um segundo momento são apresentadas as classes externas que foram referenciadas ao sistema.

A figura 21 exibe o diagrama de classes do Easy Screen com a finalidade de representar a estrutura do sistema, exibindo as classes internas do software juntamente com as relações entre as mesmas.

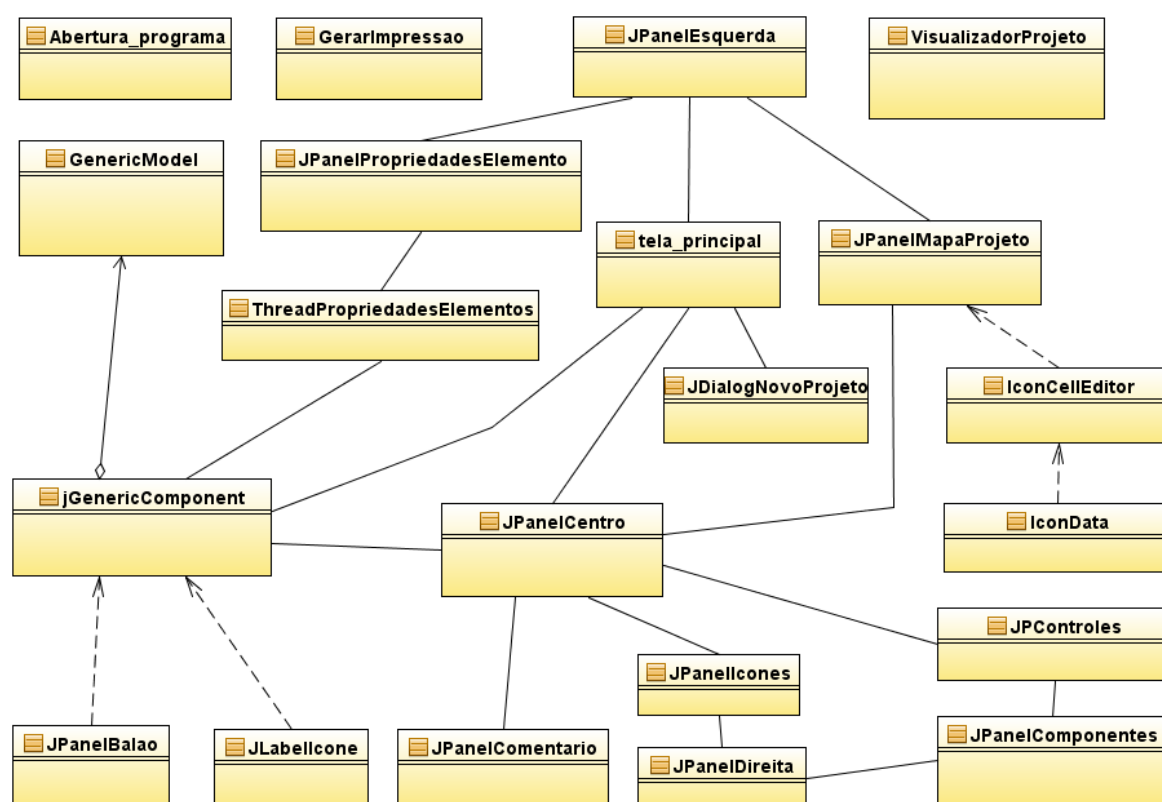


Figura 21 Diagrama Classes Software Easy Screen

No diagrama de classes representando na figura 21, existem duas classes que podem ser consideradas classes fundamentais para os objetivos que o software propõe, pois são classes que lidam diretamente com os componentes, a movimentação dos mesmos, o tamanho e a posição deles na tela, essas classes são a *GenericModel* e a *jGeneric Component*.

A classe *GenericModel* pode ser descrita como uma matriz dos componentes, a mesma irá conter as características referentes a todos os componentes usados para criar o protótipo de

tela. Essa classe contém o modelo a ser seguido pelos componentes, possui as características comuns a todos, como tamanho, tipo e nome.

O arquivo XML segue a mesma ordem de campos que estão na classe do Generic Model, isso inclui as variáveis e métodos.

Na classe denominada jGenericComponent é verificado se o componente pode ser redimensionado, movido de acordo com o tipo definido na classe GenericModel. A jGeneriComponent usa as características do Model e aplica as funcionalidades que ela possui nos componentes usados no desenvolvimento dos protótipos da interface. A posição do componente na tela, o redimensionamento do mesmo, a colisão entre os elementos também são características tratadas na classe jGenericComponent.

Na figura 22 visualiza-se as duas classes contendo nelas seus métodos e atributos.

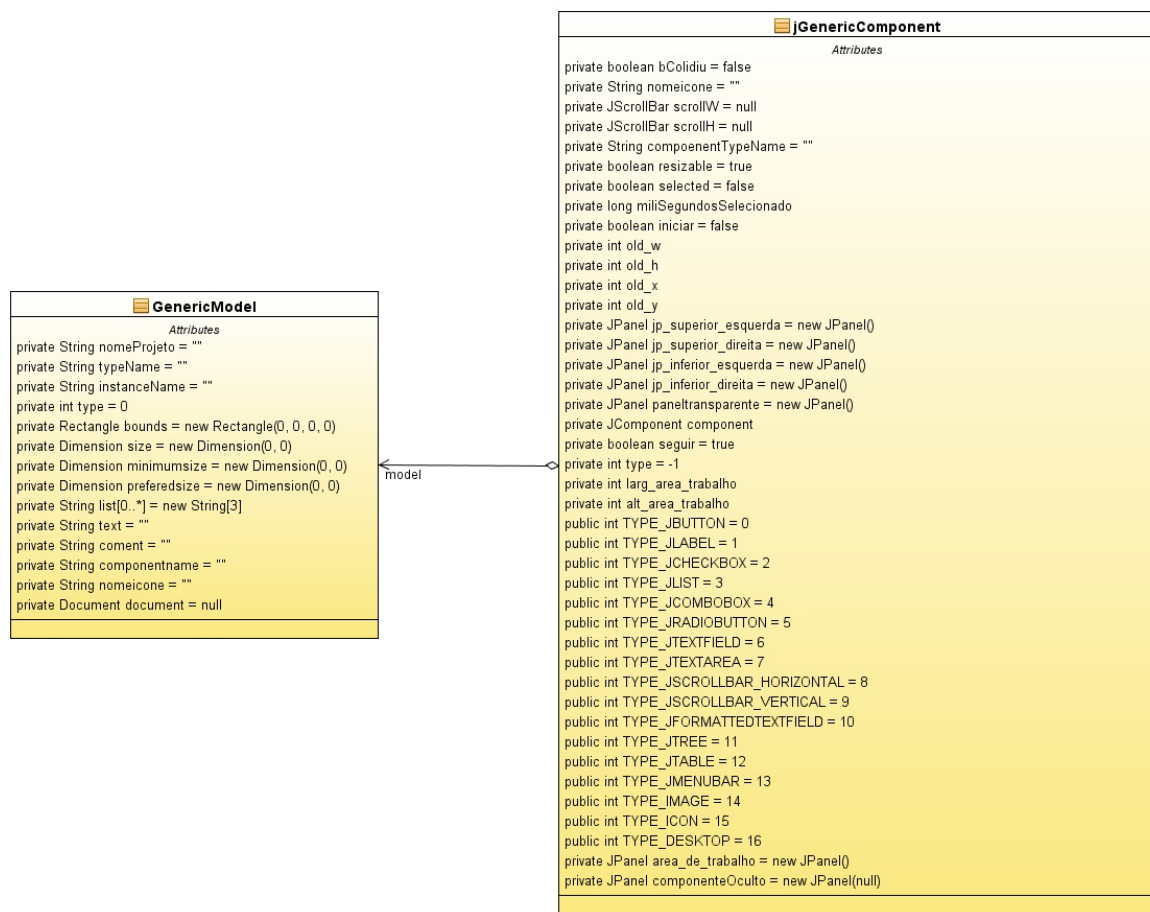


Figura 22 Classes GenericModel e jGenericComponent

O software Easy Screen possui duas bibliotecas que não foram citadas no diagrama de classes da figura 21, pois são bibliotecas externas, não são nativas da linguagem de programação Java, essas classes são XStream 1.3.1 e a iText 5.0.5.

O diagrama de componentes que encontra-se na Figura 23, exibe as classes do software Easy Screen que utilizam as classes externas XStream para gravar arquivos XML e a iText para exportação de arquivos no formato PDF.

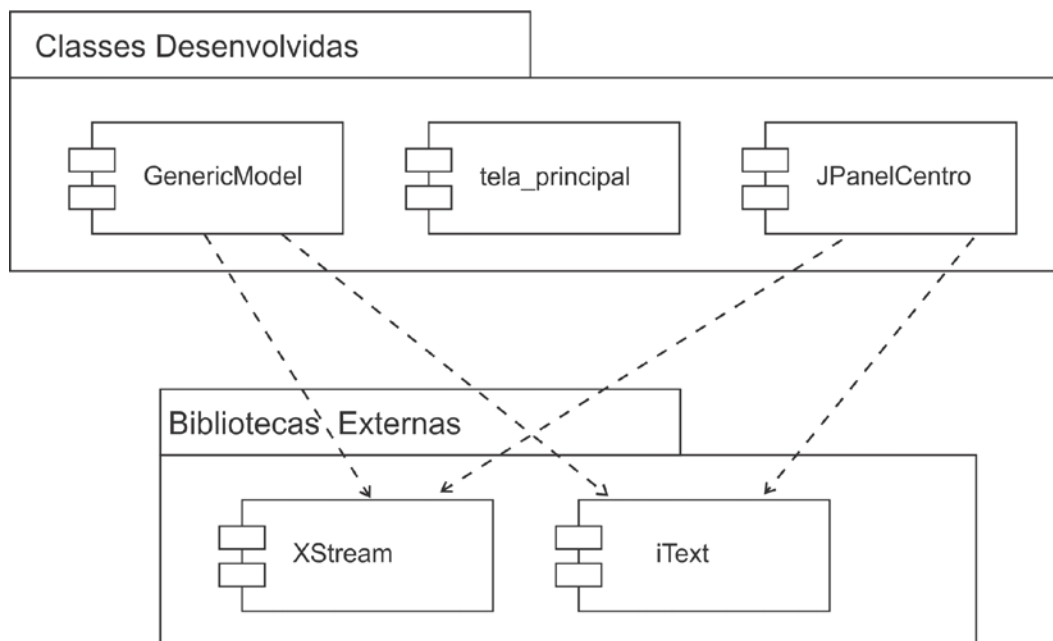


Figura 23 Diagrama de Componentes demonstrando o uso das classes externas.

A geração de XML através de objetos Java com a biblioteca XStream, tornou-se trivial porque ela implementa um conceito bem similar a serialização de objetos. A classe XStream é uma biblioteca muito útil para trabalhar com arquivos XML. Com essa biblioteca é possível criar e ler arquivos XML. A XStream permite controles parametrizados para definir a estrutura do documento.

Na ferramenta Easy Screen, a XStream faz a criação do arquivo XML através da classe GenericModel, a biblioteca percorre a classe inteira e gera um XML. A ordem dos elementos, dos métodos, enfim, toda a ordem do XML referente aos elementos segue a ordem da classe GenericModel.

A listagem 1 mostra o código em formato XML, gerado pelo Easy Screen. Nessa listagem é apresentado um componente de tela colocado na área de trabalho do software, este componente é um JTEXTFIELD, neste XML encontra-se registrado, a posição X/Y do mesmo na tela, o tamanho, tamanho mínimo e a parte do nome do componente.

```

1. <principal.GenericModel>
2.     <typeName>JTEXTFIELD</typeName>
3.     <instanceName>javax.swing.JTextField</instanceName>
4.     <type>6</type>
  
```

```

5.      <bounds>
6.          <x>129</x>
7.          <y>60</y>
8.          <width>382</width>
9.          <height>250</height>
10.     </bounds>
11.     <size>
12.         <width>382</width>
13.         <height>250</height>
14.     </size>
15.     <minimumsize>
16.         <width>110</width>
17.         <height>28</height>
18.     </minimumsize>
19.     <preferredsize>
20.         <width>582</width>
21.         <height>582</height>
22.     </preferredsize>
23.     <text>Texto do TextField</text>
24.     <coment></coment>
25.     <componentname></componentname>
26.     <nomeicone></nomeicone>
27. </principal.GenericModel>

```

Listagem 1 Código XML referente ao componente botão inserido na tela do Easy Screen

O Easy Screen também faz uso de uma biblioteca externa para a geração de arquivos em formato PDF, essa classe chama-se iText 5.0.5. Uma biblioteca Java Open Source, que permite a manipulação de arquivos PDF, possibilita a geração de documentos contendo texto, tabelas e imagens. Distribuída sob a licença GPL, proporciona seu uso em vários sistemas. Pode ser utilizada em aplicações *standalone*⁵ ou web.

4.2 Aplicação desenvolvida

A ferramenta de desenvolvimento de protótipo de interfaces desktop Easy Screen foi desenvolvida para ser uma opção mais completa aos usuários das áreas de projeto e desenvolvimento de software, esta ferramenta segue regras de usabilidade de IHC. Com o

⁵ Standalone - são programas autosuficientes: para seu funcionamento não é necessário que exista um software auxiliar, como um interpretador, sob o qual terão de ser executados.

intuito de demonstrar suas funcionalidades, a seguir serão apresentados e explicados os principais recursos do software desenvolvido.

4.2.1 Visão Geral Easy Screen

Com o objetivo de facilitar a apresentação das funcionalidades da ferramenta, será utilizada uma abordagem *top-down*. Antes de iniciar as partes específicas do software, será demonstrada uma visão geral da interface, a fim de visualizar o software como um todo. A figura 24 exibe a tela do software Easy Screen logo após seu carregamento. Nas marcações grifadas na imagem, encontram-se o que faz ou contém cada divisão da interface.

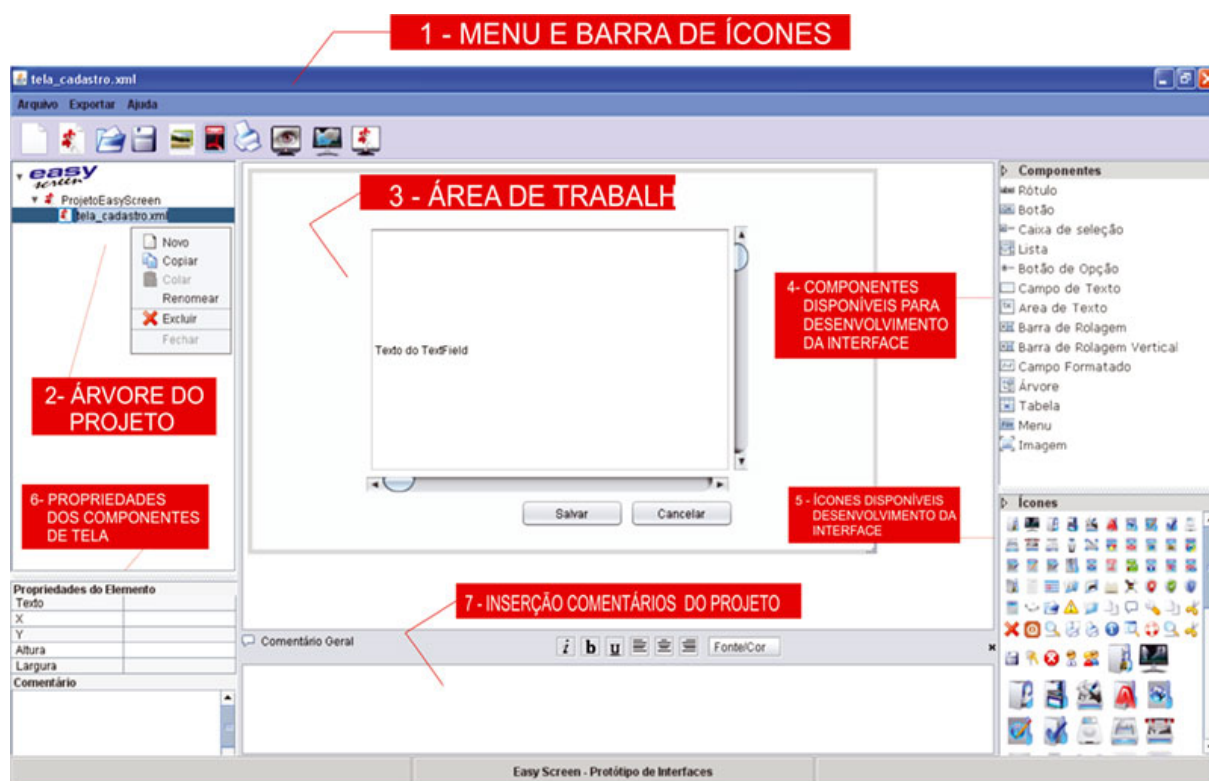


Figura 24 Visão da interface inicial do Software Easy Screen

A seguir encontra-se uma breve explicação de cada parte do software, de acordo com a numeração apresentada nas partes grifadas da figura 24.

1 – MENU E BARRA DE ÍCONES: o software contém um menu superior onde localizam-se as funções que o software deve desempenhar, todas as funções do menu contém teclas de atalho para dar mais praticidade ao usuário. Logo abaixo do menu encontra-se uma barra de ferramentas a qual contém todas as funções que o menu oferece, dando mais opções do usuário no uso da ferramenta, de acordo com as regras de usabilidade de IHC.

2 – ÁRVORE DO PROJETO: nesta parte do sistema é onde organizam-se os arquivos gerados pelo software. Através do menu acessível pelo botão direito do mouse, o usuário cria um projeto e dentro do mesmo irá gravar as telas desenvolvidas, podendo fazer as manipulações necessárias.

3 – ÁREA DE TRABALHO: permite ao usuário criar e desenvolver interfaces desktop para o seu projeto. É possível que o usuário interaja com os componentes os quais ficarão delimitados por essa área, nela é possível inserir componentes, movimentar, redimensionar, deletar, adicionar ícones e adicionar comentários específicos para cada componente.

4 – COMPONENTES – na lateral direita da interface encontram-se os componentes de tela que estão disponíveis para serem inseridos durante o desenvolvimento da interface, esses componentes seguem o padrão da IDE NetBeans.

5 – ÍCONES - nessa área encontram-se todos os ícones disponíveis pelo software, para que o usuário possa usá-los no desenvolvimento de sua interface.

6 – PROPRIEDADES DOS COMPONENTES DE TELA – na tabela que localiza-se do lado esquerdo inferior da tela, é o local onde são adicionados e modificados as propriedades dos componentes de tela e também onde é inserido o comentário específico de cada um.

7 – COMENTÁRIOS DO PROJETO: essa área localizada na parte inferior central da interface, a qual pode ser minimizada e maximizada conforme o usuário desejar, permite a inserção de comentários gerais referente a tela como um todo. Exemplo: escrever sobre como deve ser a interação de determinada tela com as demais ou alguma dica para o programador, enfim, tudo que julgar ser útil ser comentado. O usuário conseguirá, através da barra de ferramentas que existe nessa área, personalizar seu comentário no que diz respeito a cor de letra, tipo de fonte, tamanho da fonte, alinhamento do texto, recursos esses que facilitam a diferenciação entre os vários comentários que possam ser inseridos nessa área.

Nos tópicos a seguir, serão demonstradas imagens de partes importantes do software, juntamente com uma explicação mais detalhada da funcionalidade de cada uma dessas partes.

4.2.2 Componentes e Ícones

Os componentes que o software Easy Screen disponibiliza para os usuários são os mesmos usados na ferramenta IDE NetBeans, esse padrão foi adotado primeiramente por serem componentes já conhecidos pela maioria dos usuários, por contemplarem a maioria das necessidades que os usuários necessitam para o desenvolvimento de uma interface completa.

Neste conjunto de componentes o único que diferencia-se dos demais é o último item da lista chamado IMAGEM, o mesmo foi criado estendendo a Classe JLabel, sobrescrevendo o método *paint()*, existente em todos os componentes do Java. No mesmo é possível colocar borda, cor, tamanho e determinar posição X e Y.

Para que esses componentes de tela sejam inseridos na área de trabalho, existe uma interação do usuário que consiste em clicar com o botão do mouse sobre a imagem/ícone e após clicar com o mesmo botão do mouse sobre a área de trabalho. Com essa ação o usuário consegue inserir o objeto na área de trabalho e manipulá-lo conforme suas necessidades.

A lógica usada nessa ação consiste em: existe uma classe JPControles do tipo JPanel que controla quando algum componente é clicado, ela identifica qual componente foi clicado e cria um objeto do tipo GenericModel, classe essa apresentada na seção X e Y. Com este GenericModel é criado então uma nova instância do JGenericComponent, o qual captura as características do componente armazenadas no GenericModel e aplica as funcionalidades para os componentes. Na listagem 2 a seguir pode-se observar um exemplo usado no Easy Screen, de inserção de um componente.

```
1. //Selecionando um JLabel na área dos componentes
2. GenericModel model = new GenericModel(); //Cria um GenericModel
3. model.setType(jGenericComponent.TYPE_JLABEL); //diz pro model que
   ele é um JLabel
4. model.setText("Texto do JLabel"); //Seta o texto inicial
5. jGenericComponent generic = new jGenericComponent(model);
6. // cria um jGenericComponente com base no model é passado para o
   construtor.
```

Listagem 2 Código representando a inserção de um componente

Na parte destinada aos ícones a interação que o usuário executa é a mesma usada na parte citada para os componentes, basta um clique no ícone e após um clique na área de trabalho que o mesmo é inserido para o desenvolvimento da interface.

A variedade de ícones disponibilizados pelo software Easy Screen para o usuário é bem ampla, em torno de 50 ícones, que podem ser usados para diversos temas de tela. São disponibilizados ícones em 2 tamanhos, com dimensões de 16x16 pixels e os maiores com as dimensões 32x32 pixels. Por ser uma ferramenta de código livre, caso o desenvolvedor deseje inserir mais ícones ao projeto, basta que o mesmo os insira na pasta IMAGES/ICONES_USUARIO que encontra-se dentro do projeto. Na figura 25 é exibida a

parte lateral direita do software Easy Screen que mostra os componentes e ícones que serão usados pelo usuário para desenvolvimento das interfaces.



Figura 25 Tela dos Componentes e Ícones

4.2.3 Propriedade dos Componentes

Neste painel encontram-se os valores dos componentes que são inseridos na área de trabalho. No momento que o usuário clicar em um componente, o painel mostrará as propriedades atuais do mesmo na posição que ele encontra-se no momento do clique, propriedades essas como: nome, posição do componente no eixo X, posição no eixo Y, largura do componente, altura do componente, além da opção para inserir comentários específicos referente a cada componente.

Em cada alteração realizada pelo usuário através do painel de propriedades ou também pelas alterações que o componente pode sofrer através de interações feitas na área de trabalho, o sistema atualiza as propriedades do componente. Ações essas são possíveis devido a execução de uma *thread* que fica percorrendo a área de trabalho em busca do componente selecionado, quando esse é achado passa o componente para uma *TableModel*, se o usuário

fizer modificações de alguma informação, a tabela é modificada diretamente na *TableModel*, que por sua vez modifica o *GenericModel* do *JGenericComponent* selecionado. Qualquer mudança no *GenericModel* mudará o componente que está na área de trabalho. Estas ações valem para o balão de comentário que está localizado na classe *JGenericComponent*. Em caso de mudança do *GenericModel*, a mudança ocorrerá automaticamente na posição do balão. Essas alterações são aplicadas através da *Thread* que a cada 300 ms faz busca de componentes selecionados e fica atualizando as informações.

A figura 26 exibe o painel de propriedades dos elementos com os valores preenchidos e com um comentário relacionado ao componente editado.

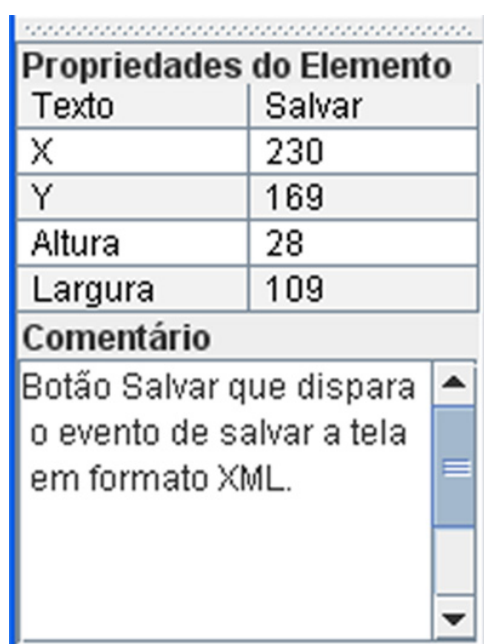


Figura 26 Tela do painel de Propriedades dos Elementos

4.2.4 Área de Trabalho

Na área central do software Easy Screen encontra-se a área de trabalho, espaço esse destinado para que o usuário possa fazer várias interações com os componentes de tela a fim de desenvolver uma interface que satisfaça os objetivos estabelecidos. As interações que o usuário pode fazer com os componentes nessa área são ações como: movimentar os componentes, redimensioná-los, excluir e visualizar comentários inseridos em cada componente.

A ação de movimentar o componente, sob o ponto de vista do usuário, baseia-se em manter clicado o componente e movimentá-lo para a área de tela que julgar melhor. A lógica da ação movimentar é que quando o usuário clicar e arrastar o mouse, o componente fica preso ao mouse e somente quando o usuário soltar o botão que o componente desprende-se da

ação do mouse. Para saber a posição do componente quando o usuário soltar o mesmo, são feitos cálculos baseados na distância do componente em relação a tela tanto no eixo X como no eixo Y.

O redimensionamento é realizado pelo usuário através de uma interação com o componente, na qual o usuário clica sobre o mesmo, neste momento aparecem quatro quadros de cor azul nas extremidades do componente. Para que o usuário consiga redimensionar o componente, basta posicionar o mouse sobre um dos quadros clicar e arrastar o quadro para o tamanho que desejar, após o usuário largar o botão do mouse o objeto aparecerá em tamanho maior ou menor.

Para implementar a ação de redimensionar foi adicionado um painel transparente que encontra-se na classe JGnericComponent e acima desse painel foram adicionados 4 JPanels que no momento que são clicados disparam as ações de redimensionar. Cada um desses JPanels leva em consideração a posição absoluta da tela, altura e largura do componente, local do componente na tela e na área de trabalho, também o local da área de trabalho na tela. Cada um dos JPanels de redimensionamento tem um cálculo diferente. Para melhor ilustrar a ação de redimensionamento segue a figura 27 ilustrando as camadas construídas para a ação de redimensionamento.

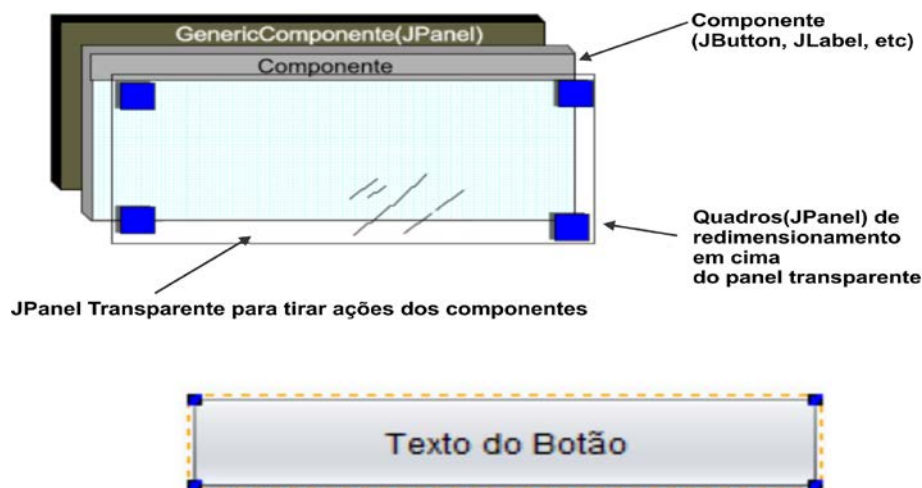


Figura 27 Imagem ilustrando ação do redimensionar e o componente com a ação selecionada

Na ação de excluir o usuário interage clicando sobre o componente e teclando a tecla DELETE. Essa ação é feita da seguinte forma: é capturada a camada Root Panel e associada a ela a ação do DELETE, qualquer componente ou camada que encontra-se acima dessa camada é excluído.

Além das interações ligadas a estrutura do componente, o Easy Screen dispõe da ação de inserir um comentário específico sobre o componente conforme já comentado no item

4.2.3. Quando esse comentário é inserido ele pode ser visualizado ou não de acordo com a necessidade do usuário. Para visualizar o comentário basta o usuário clicar com o botão direito do mouse sobre o componente, neste momento aparecerá um balão de cor azul contendo o comentário inserido. Para fechar a visualização basta o usuário clicar sobre o 'X' que existe no canto superior direito do balão.

A figura 28 exibe uma área de trabalho que contém uma interface em desenvolvimento, nessa imagem é possível visualizar vários elementos de tela, entre eles encontra-se um elemento clicado sendo possível visualizar os 4 quadros de redimensionamento em suas extremidades, bem como o componente botão, que demonstra logo acima do mesmo o balão de comentário com conteúdo em seu interior.

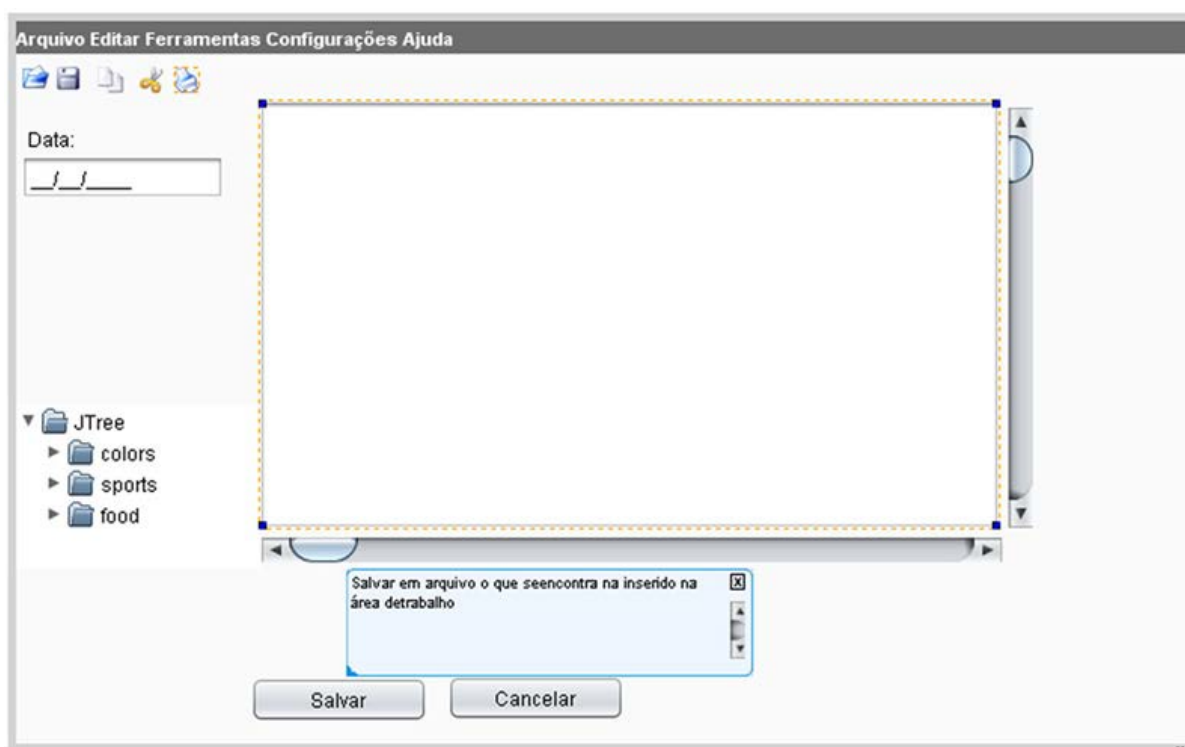


Figura 28 Tela da Área de trabalho do software Easy Screen

4.2.5 Árvore do Projeto

Essa parte do software que pode ser intitulada de Árvore do Projeto, está localizada na lateral esquerda da interface. O usuário interage com esta área do software da seguinte forma: primeiramente será criado um projeto através do menu principal ou da barra de ferramentas, somente após esse procedimento o usuário interage diretamente com a árvore. Um menu rápido é acionado quando o usuário clicar o botão direito do mouse, nesse menu, mostrado na

figura 29, contempla as funções de criar nova tela, copiar uma tela e colar em outra nova tela que pode ser do mesmo ou de outro projeto que esteja aberto na árvore. Essa função de cópia tem grande utilidade no caso do usuário querer ocupar itens que já desenvolveu em outra tela, é possível renomear uma tela diretamente através da árvore, excluir uma tela do projeto, além de também poder ser fechado o projeto que encontra-se na árvore.

Na árvore do projeto o usuário pode criar vários projetos e por sua vez vários arquivos dentro de cada projeto, durante a interação entre as telas, o usuário pode fazer várias mudanças em cada tela, no momento que o usuário mudar de uma tela para outra, é disparado um pedido para o usuário salvar as alterações feitas, com isso nenhuma interação que o usuário tenha feito com a tela é perdida.

A figura 29 mostra o mapa do projeto com dois projetos em desenvolvimento, cada um com suas telas, também exibe o menu rápido com as opções e ao lado a solicitação para o usuário salvar uma alteração realizada.

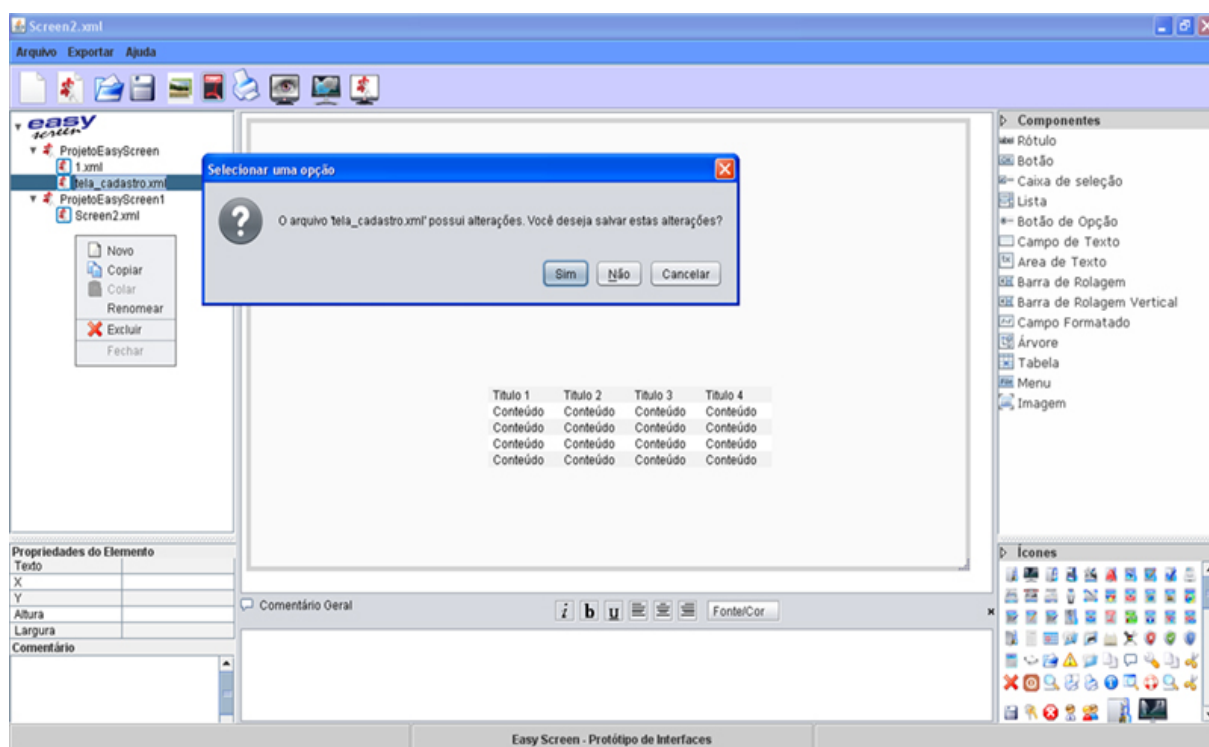


Figura 29 Tela Mapa do Projeto mostrando suas opções de interação

4.2.6 Menu e Barra de Ferramentas

Muitas etapas da interação do usuário com o software Easy Screen acontecem através do menu principal e da barra de ícones, os quais contemplam as mesmas funções. A seguir serão descritas as funções referentes ao menu 'Arquivo' e 'Exportar' e após na figura 30 será demonstrado um exemplo.

O menu Arquivo existente no Easy Screen contém itens de menu como Nova Tela, a qual contempla a função de criar um novo arquivo dentro da árvore no Mapa do Projeto. O segundo item contém o primeiro passo que deve ser executado pelo usuário ao abrir o Easy Screen, é o Novo Projeto. Ao clicá-lo, abrirá uma tela a qual solicitará ao usuário o Nome do Projeto, Localização do Projeto e o Nome da Nova (e primeira) Tela.

O terceiro item de menu corresponde a opção Abrir, no qual o usuário deve informar o local que exista um projeto para ser aberto, quando o Easy Screen encontrar uma pasta que seja um projeto do Easy Screen, a mesma aparecerá com o símbolo do software na frente do nome da pasta. Essa função verifica os diretórios a procura de um arquivo chamado “versão.cfg” que tem em seu conteúdo apenas o nome do projeto e a data de criação do mesmo. Ao encontrar este arquivo a imagem da pasta é mudada para o símbolo do Easy Screen.

O item na quarta posição é o Salvar, sua função é a mesma do salvar que encontra-se no menu que localizado no Mapa do Projeto, ele salva a tela no arquivo XML. No quinto item a função é o Imprimir, que captura toda a área de trabalho, transforma em uma imagem que é adaptada para o tamanho de impressão A4 (21 x29,7cm) e procura a impressora do sistema operacional em execução.

O segundo menu chamado Exportar, contém no seu primeiro item a função Exportar a tela para um arquivo do formato PDF, essa ação é feita pela biblioteca iText a qual foi explicada na seção 4.1.2. O segundo item contempla a função Exportar Imagem que captura a área de trabalho e transforma em uma imagem do formato PNG.

A barra de ícones localizada abaixo do menu principal do software Easy Screen, contempla todas as funções do menu principal menos os três últimos ícones, os quais representam os tipos de visualização que o software oferece. A primeira visualização, apenas oferece uma prévia da tela que está sendo desenvolvida, ao clicar no ícone o usuário poderá ver a interface que está desenvolvendo em uma tela separada do software conseguindo assim fazer uma melhor análise do que está sendo desenvolvido. A segunda visualização exibe a mesma visualização da primeira, porém, mostrando para o usuário todos os comentários feitos em cada componente da tela que está sendo desenvolvida. O último ícone gera uma visualização de todo o projeto em desenvolvimento, ao clicar o usuário poderá visualizar uma a uma todas as telas do seu projeto.

A figura 30 demonstra o menu, a barra de ícones e os visualizadores do software.

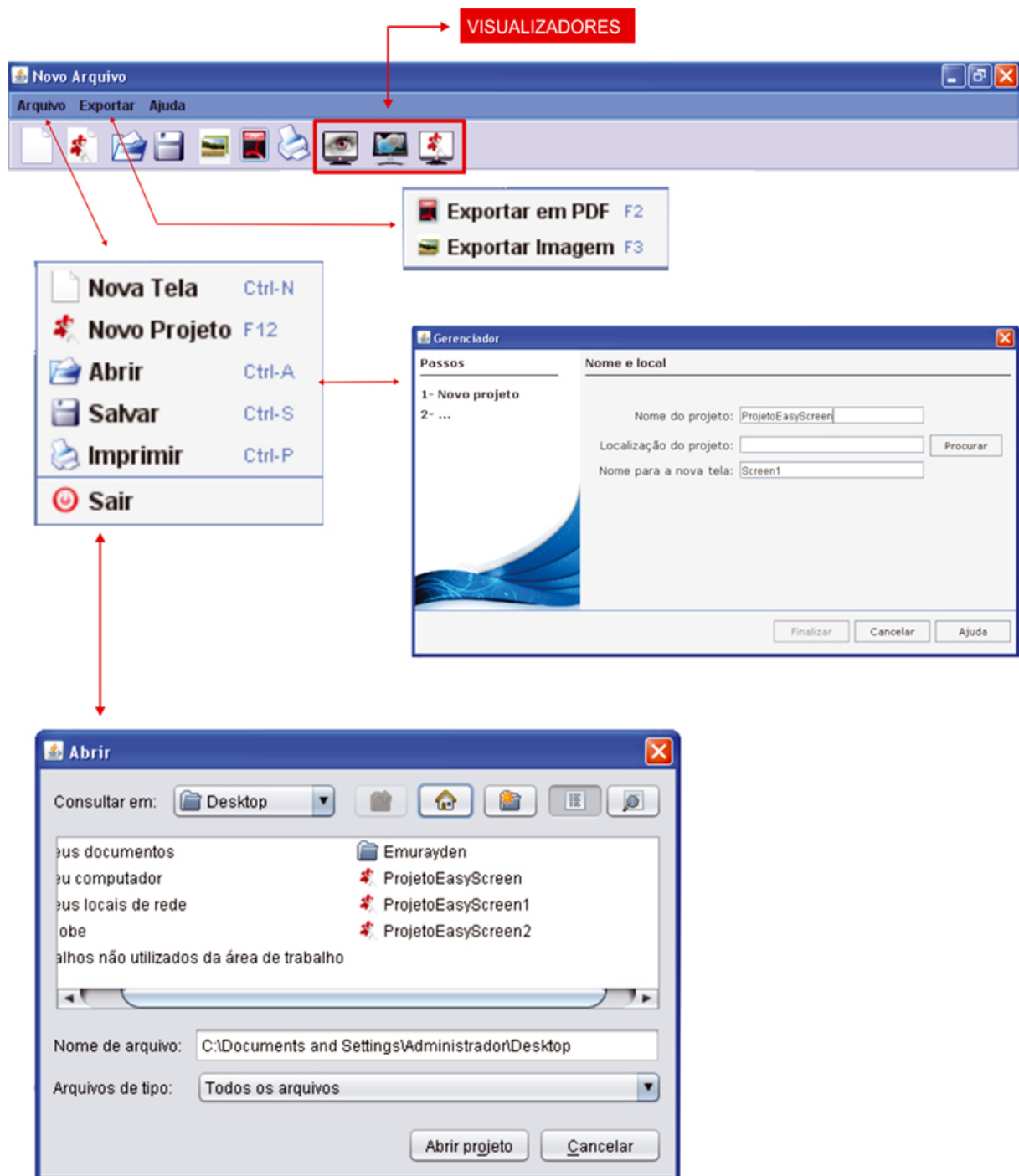


Figura 30 Tela exibindo o menu com suas funções

4.2.7 Análise do Easy Screen com Softwares similares

Conforme tabela apresentada no capítulo 3, a qual demonstrou uma comparação entre as ferramentas de prototipação de interfaces, é demonstrado na tabela 3 a comparação das funcionalidades do Easy Screen com os demais softwares de prototipação de interfaces, Designer Vista, Gui Design Studio, Wiremaster e Axure RP Pro.

Tabela 3 Tabela comparativa das funcionalidades do Easy Screen com demais softwares

Funcionalidades	Easy Screen	Designer Vista	GUI Design Studio	Wiremaster	Axure RP Pro
Exportação em HTML		X			X
Exportação em imagem	X		X	X	X
Permite anotações nas telas	X	X	X		X
Simular Navegação		X	X		X
Exportação em XML	X	X	X	X	X
Variedade de elementos para criação da interface	X	X	X		
Visualização da tela criada	X	X	X		X
Incluir comentários em cada componente de tela	X				
Licença	Software Livre	Software Proprietário . Versão para Teste	Software Proprietário. Versão para Teste	Software Livre	Software Proprietário. Versão para Teste
Diferencial	Adição de comentário em cada componente adicionado	Permite importação de dados para Planilhas Eletrônicas	Teste de Navegação entre telas		Permite criar especificações para o formato word
Plataforma	Windows Linux	Windows	Windows	Windows Linux	Windows
Organização Telas em Projeto	X				

Observa-se na Tabela 3 que o software Easy Screen contempla a maioria das funcionalidades em comparação com os outros softwares, apenas não contemplando algumas funcionalidades como exportação para HTML e simulação de navegação. O Easy Screen contém como funcionalidades diferenciais dos outros softwares, a adição de comentários gerais e específicos em cada componente de tela, também permite organizar os protótipos de tela em projeto possibilitando assim uma melhor organização das telas desenvolvidas.

5 VALIDAÇÃO DO SOFTWARE EASY SCREEN

Contemplando a última etapa do projeto, o software Easy Screen foi testado. Esta validação ocorreu nas dependências da UNIVATES – Centro Universitário, no dia 21 de maio de 2011, com vinte alunos dos cursos de Sistemas de Informação e Engenharia da Computação, os quais cursavam a disciplina de IHC.

A realização dos testes ocorreu em etapas. Foi informado aos alunos que acessassem o endereço: <http://ensino.univates.br/~diogo/> para poderem acompanhar todas as etapas do teste que constava no conteúdo deste endereço eletrônico.

Primeiramente foi solicitado aos alunos para executarem o download da ferramenta, a qual foi disponibilizada em uma versão para Linux e uma versão para Windows com instalador. Após a turma foi orientada a ler um Termo de Consentimento, e foi solicitado aos alunos que interagissem de maneira livre com a ferramenta durante o período de 5 minutos, com intuito de que o usuário se ambientasse com a ferramenta.

Seguindo a aplicação da validação foi explicado aos alunos uma lista de itens, contendo as tarefas que os mesmos deveriam executar, tarefa esta que tinha o objetivo, fazer com que o usuário pudesse desenvolver algumas telas. A mesma consistia nos seguintes passos: solicitar ao usuário que montasse uma tela de cadastro de pessoa física e outro cadastro para pessoal jurídica, gerar uma imagem do cadastro de pessoa física, e um arquivo formato PDF da tela pessoa jurídica, após todo o exercício resolvido enviar os resultados por e-mail. A tarefa completa pode ser vista no Apêndice A.

Finalizando a validação, os avaliadores responderam um questionário *on line*, desenvolvido através da ferramenta Googledocs, disponível através do endereço eletrônico <http://ensino.univates.br/~diogo/> no item avaliação. Este questionário encontra-se no Apêndice B.

De posse de todas as informações a respeito do teste e com a ferramenta disponível, os alunos realizaram a tarefa proposta. Após a realização da validação foram extraídos os dados fornecidos pelos validadores a fim de captar a visão dos mesmos sobre o software. Nos itens a seguir encontram-se os pontos positivos e negativos citados pelos usuários (*ipsis litteres*) de acordo com o que foi solicitado no questionário.

a) Pontos Negativos

- Difícil alinhamento dos componentes;
- Não possibilita a seleção de mais de um componente de tela ao mesmo tempo;
- Apenas uma possibilidade de editar os componentes;

- Poucos elementos de interface;
- Falta da implantação do copiar (CTRL+C) e colar (CTRL+V);
- Falta da ação de editar os componentes através de 2 cliques sobre o mesmo;
- Não existe a possibilidade de cancelar um objeto que já foi clicado para ser inserido na área de trabalho;
- A edição dos componentes através da tabela propriedade dos elementos, aceita posição negativa para as posições X e Y dos componentes;

b) Pontos Positivos

- Fácil visualização dos componentes e itens;
- Fácil interação, não é necessária ajuda e nem documentação;
- Menus intuitivos;
- Facilidade de encontrar os componentes e usá-los;
- Padrão nos atalhos;
- Layout da Ferramenta e componentes parecida com Net Beans;
- Possibilidade de abrir vários projetos numa mesma tela;
- Várias exportações como imagem e PDF;

Ao final do questionário encontrava-se um espaço reservado a sugestões livres referentes ao software Easy Screen. Abaixo segue a lista de sugestões:

- Utilizar a tecla ESC para abandonar a ação de adicionar os componentes;
- Problema com alinhamento dos componentes na tela;
- Fiz algo errado, não permite voltar;
- Comentário só pode ser inserido através da aba de propriedades do componente;
- A idéia do software é excelente, porém vejo como forte limitador a linguagem Java, que embora possui recursos suficientes para um desenvolvimento, porém acaba tornando o processo de desenvolvimento muito trabalhoso devido a instabilidade do uso de certos componentes;
- No geral está boa a ferramenta;
- Seria interessante mas não importante ou necessário, fazer os comentários dos campos em tempo real.

Na figura 31 segue um exemplo de tela desenvolvida por um dos avaliadores.

Titulo 1	Titulo 2	Titulo 3	Titulo 4
Conteúdo	Conteúdo	Conteúdo	Conteúdo
Conteúdo	Conteúdo	Conteúdo	Conteúdo
Conteúdo	Conteúdo	Conteúdo	Conteúdo
Conteúdo	Conteúdo	Conteúdo	Conteúdo
Conteúdo	Conteúdo	Conteúdo	Conteúdo

Figura 31 Tela desenvolvida por usuário na etapa de Validação

Como constatação geral sobre a etapa de validação do software Easy Screen, ficou bem claro por parte dos usuários o interesse mostrado em realmente avaliar a ferramenta, esse fator foi um ponto positivo, pois com isso a ferramenta foi realmente testada e as críticas e sugestões surgiram de maneira natural, sem interferência por parte do desenvolvedor ou qualquer pessoa que estivesse envolvida nesse processo. Outro ponto positivo foi que a ferramenta foi avaliada por pessoas que estão inseridas no ramo de informática e desenvolvimento, e conhecedores de muitas regras de IHC, das quais o software se enquadrava.

Para ilustrar uma estatística das respostas dos avaliadores, segue na figura 32 um gráfico que demonstra a impressão dos usuários a respeito da ferramenta, quando foram perguntados: “Na sua avaliação, você acha que a ferramenta de estudo contribui para tarefa de desenvolvimento de protótipos?”.

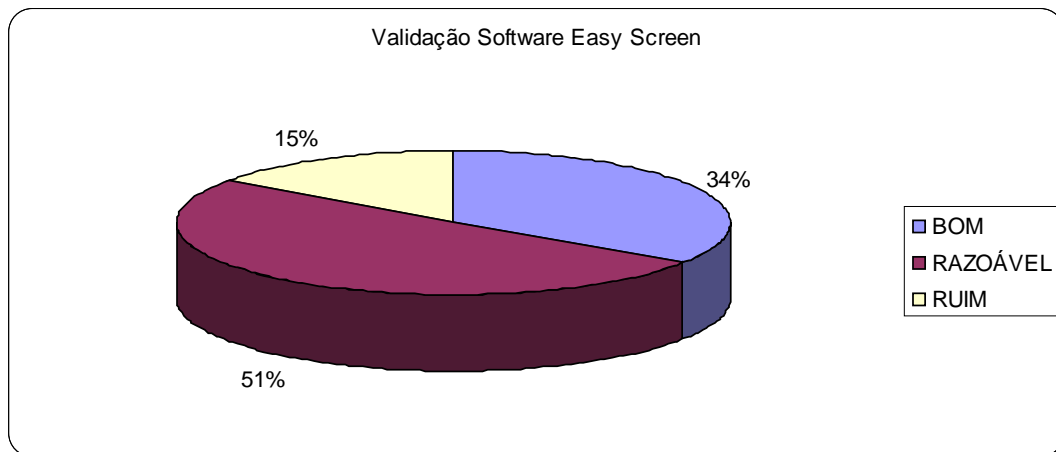


Figura 32 Gráfico Estatístico demonstrando a avaliação da ferramenta Easy Screen.

Entre as perguntas encontradas no Apêndice B, destacam-se duas perguntas, uma é a de número quatro “Quanto a flexibilidade, capacidade de realizar a mesma ação de mais maneiras, como a ferramenta se apresenta?”, essa pergunta teve o pior índice de avaliação entre os avaliadores, sendo Adequado 15%, Razoável 60%, Ruim 35%. Em contra partida, a pergunta número cinco “Como você classifica a diversidade de componentes e ícones oferecidos para o usuário através da ferramenta EasyScreen?”, obteve um índice do resultado melhor na avaliação, atingindo os índices de Adequado 45%, Razoável 45% e Ruim 10%. Essa pontuação demonstra que na questão cinco, 90% dos avaliadores julgou a diversidade de componentes entre Razoável e Bom, sendo positivamente qualificada. Através dessa pontuação pode-se concluir que a ferramenta Easy Screen, obteve um percentual muito bom por parte dos avaliadores, o que demonstra que a ferramenta atingiu seus objetivos.

Algumas das críticas e sugestões já foram incorporadas à versão, porém não todas, por causa do fator tempo e também levando em conta que uma nova validação não ocorreu.

As melhorias já incorporadas ao software levando em consideração as sugestões e críticas foram:

- O sistema somente aceita o desenho da interface após o usuário criar o projeto;
- O software não aceita valores negativos nas posições X e Y que dizem respeito aos componentes inseridos na tela;
- Foi retirada uma seta indicativa, a qual encontrava-se na parte da tela que continha os componentes, que dava ideia do usuário poder minimizar e maximizar a janela.
- Alguns dos componentes como botão, rótulo, botão de opção, já podem ter seu nome editado apenas com dois cliques sobre o componente, evitando assim que o usuário precise fazer o ajuste pela tabela Propriedades do Elemento.

6 CONCLUSÃO

Um dos grandes desafios dos desenvolvedores de softwares é conseguir extrair as opiniões necessárias dos futuros usuários sobre os sistemas que serão desenvolvidos. As empresas querem criar produtos que conquistem e mantenham a fidelidade do cliente, produtos que tragam facilidade no desempenho das tarefas a serem realizadas.

O que existe em muitos casos são reclamações dos usuários dizendo que os sistemas não contemplam as funções necessárias que os mesmos gostariam ou que muitos sistemas não são de fácil interação, o que acaba causando insatisfação, e muitas vezes fazendo com que o usuário desista da utilização do sistema.

Uma forma para diminuir reclamações futuras, é tentar reduzir a distância entre os usuários e os desenvolvedores, e em relação ao produto que está sendo desenvolvido. O usuário precisa fazer parte do desenvolvimento do sistema desde as primeiras etapas de implantação do mesmo, para que no caso de haverem mudanças percebidas dos usuários elas possam ser resolvidas com menor tempo, menor retrabalho e com menor custo.

Um dos aspectos que melhora a comunicação entre usuários e desenvolvedores é a técnica de prototipação. Pensando nestes fatores é que foi desenvolvida a ferramenta de prototipação de interfaces Easy Screen, com ela os desenvolvedores ou analistas conseguem apresentar ao cliente/usuário de forma rápida uma ideia geral de como será o futuro software. A utilização de uma ferramenta que desenvolve interfaces melhora a produtividade e amplia a capacidade do usuário testar a interface ao longo do seu desenvolvimento.

O software Easy Screen oferece aos usuários muitas opções para o desenvolvimento de uma interface desktop. Foi desenvolvida uma ferramenta de fácil utilização e com uma interface que segue regras das normas IHC.

6.1 Trabalhos Futuros

Para melhoramento da ferramenta desenvolvida, algumas funcionalidades poderiam ser implementadas, tais como: aumento de algumas funções que o software, uma delas seria gerar um padrão de XML que possa ser usado por outras ferramentas, poupando assim muitas linhas de código por parte dos desenvolvedores; um outro ponto é fazer do Easy Screen um *plugin* que pode ser acoplado a IDEs de desenvolvimento de software tais como NetBeans e Eclipse, difundido assim muito mais a ferramenta entre os desenvolvedores e até mesmo usuários finais.

REFERÊNCIAS

- AM Endler, MS Pimenta - **Proceedings of the VI Simpósio sobre Fatores**, 2004
<http://www.bamin.com.br/artigos/ihc_empresas.pdf> Acessado em: 29 de ago. de 2010.
- ANDRIANI, M.R. **Usabilidade, interagindo com interfaces funcionais**. Disponível em
<<http://www.grupos.com.br/blog/interacao-humano-computador/permalink/23586.html>>
Acessado em: 15 Setembro 2010..
- CYBIS, W.; BETIOL, A. H.; FAUST, R. **Ergonomia e Usabilidade: Conhecimentos, Métodos e Aplicações**. São Paulo: Novatec, 2007 ISBN 9788575221389
- FURLAN, José Davi. **Modelagem de objetos através da UML - the unified modeling language**. São Paulo: Makron Books, 1998.
- GUIMARÃES KARINA, disponível em <<http://www.grupos.com.br/blog/interacao-humano-computador/permalink/23586.html>> Acessado em: 12 Setembro 2010.
- HIX, D.; HARTSON, H.R. **Developing user interfaces: ensuring usability through product and process**. John Wiley: New York, 1993.
- ITEXT, disponível em < <http://www.itextpdf.com/>> Acessado em: 25 Março 2011.
- MACK,R. NIELSEN, J. **Usability Inspection Methods**. New York: NY: John Wiley & Sons, 1994.
- MACORATTI.NET – **Processo de Software** -
<http://www.macoratti.net/proc_sw1.htm>Acessado em: 20 de setembro de 2010.
- MELO, Ana Cristina. **Desenvolvendo aplicações com UML: do conceitual a implementação**. Rio de Janeiro: Brasport, 2003.
- NIELSEN, Jakob. **Projetando websites**. Rio de Janeiro: Elsevier, c2000.
- NIELSEN, J. (1993). **Usability Engineering**. Chestnut Hill, MA, Academic Press, 1993.
- OLIVEIRA, Alvim Antônio de. **IHC: interação humano computador**. Florianópolis: VisualBooks, 2006.
- PREECE, J.; ROGERS, Y.; SHARP, H.; BENYON, D.; HOLLAND, S.; CAREY, T. **Human-Computer Interaction**. EUA: Addison Wesley, 1994.
- PREECE, J.; ROGERS, I.; SHARP, H. **Design de Interação: Além da Interação Humano-Computador**; Porto Alegre: Bookman, 2005.

PRESSMAN, S. ROGER, Tradução de Rosângela Ap. D. Penteado. **Engenharia de Software**. 6ª Edição. **São Paulo**: ARTEMED Editora S.a. e McGraw-Hill Education, 2005 ISBN 9788563308009

QUATRANI, Terry. **Modelagem visual com Rational Rose 2000 e UML**. Rio de Janeiro: Editora Ciência Moderna Ltda., 2001.

SOMMERVILLE, I. Tradução de Selma Shin Shimizu Melnikoff, Reginaldo Arakaki, Edílson de Andrade Barbosa. **Engenharia de Software**. 8ª edição. **São Paulo**: Pearson Addison-Wesley, 2007 ISBN 978858863287

XSTREAM, disponível em < <http://xstream.codehaus.org/>> Acessado em: 15 Março 2011.

APÊNDICE A: PROTOCOLO DE TESTES

- * ler as instruções atentamente;
- * ambientar-se a ferramenta por um período de 5 minutos;
- * desenvolver a tarefa abaixo e após preencher o questionário que está no link avaliação;
- * não solicitar auxílio ao aplicador da avaliação, com exceção do acontecimento de alguma situação que impeça a continuação da tarefa.

Tarefa para ser desenvolvida com o Software Easy Screen

Com o intuito de manter a objetividade e homogeneidade na realização da tarefa, realize as seguintes ações:

- a) Crie um projeto, nomeie de PessoaFisica e salve no local que julgar adequado. Durante a criação do projeto, crie uma página com o nome cadastroF.
- b) Desenvolva um layout simulando a criação de um pequeno cadastro de cliente.
- c) Crie um novo projeto chamado PessoaJuridica e deixe o projeto vazio neste momento.
- d) Copie a tela “cadastroF” do projeto PessoaFisica e cole no projeto PessoaJuridica com o nome cadastroJ.
- e) Faça um comentário geral para a tela CadastroJ.
- f) Faça alguns comentários específicos para cada um dos componentes da tela cadastroF
- g) Crie um arquivo PDF da tela cadastroJ e salve onde julgar adequado.
- h) Crie uma imagem da tela cadastroF e salve onde julgar adequado.
- i) Envie os arquivos gerados para o e-mail: diogoj976@gmail.com

APÊNDICE B: QUESTIONÁRIO PARA AVALIAÇÃO DO SOFTWARE EASY SCREEN

De acordo com sua percepção sobre o software, os exercícios desenvolvidos e baseando-se nas regras de IHC, avalie a ferramenta EasyScreen conforme questionário abaixo:

1. Como você classifica a apresentação da ferramenta EasyScreen quanto ao seu layout:

- ☐ Adequado
- ☐ Razoável
- ☐ Ruim

2. Como você avalia a interação da ferramenta EasyScreen?

- ☐ Adequado
- ☐ Razoável
- ☐ Ruim

3. Na sua percepção como avaliador como você classifica a ferramenta EasyScreen, nos aspectos: percepção, capacidade de memorização, atenção?

- ☐ Adequado
- ☐ Razoável
- ☐ Ruim

4. Quanto a flexibilidade, capacidade de realiza a mesma ação de mais maneiras, como a ferramenta se apresenta ?

- ☐ Adequado
- ☐ Razoável
- ☐ Ruim

5. Como você classifica a diversidade de componentes e ícones oferecidos para o usuário através da ferramenta EasyScreen ?

- ☐ Adequado
- ☐ Razoável
- ☐ Ruim

6. Como você classifica a facilidade de entendimento dos atalhos usados no menu e os ícones da barra de ferramentas ?

- ☐ Adequado
- ☐ Razoável
- ☐ Ruim

7. Como você avalia a organização através de projetos das telas geradas pelo EasyScreen ?

- ☐ Adequado
- ☐ Razoável
- ☐ Ruim

8. Como você classifica as saídas geradas pelo software, tais como: impressão das telas, geração de PDF, e exportação para imagem formato PNG?

- ☐ Adequado
- ☐ Razoável
- ☐ Ruim

9. Na sua avaliação você acha que a ferramenta de estudo contribui para tarefa de desenvolvimento de protótipos ?

- ☐ Adequado
- ☐ Razoável
- ☐ Ruim

