*CCEE - Centro de Ciências Exactas em Engenharia*

# *WebSketch* 2009

Proposta de uma aplicação baseada na Web para a modelação de Protótipos Canónicos Abstractos.

**Cristian Eduardo de Abreu Gonçalves**
(Licenciado)

*Tese submetida à Universidade da Madeira para a obtenção de Grau de Mestre na área de Engenharia de informática.*

Funchal, Madeira – Portugal
Novembro 2009

## *Orientador*

*Professor Doutor Leonel Nóbrega*

*Professor Auxiliar do Departamento de Matemática e Engenharias*

*Universidade da Madeira*

# *Abstract*

Nowadays the development of Web-Based applications is growing exponentially because of their benefits. With the increasing use of the Web and its evolution as a platform, new technologies have emerged that revolutionize the development of applications for this platform. With richer and more dynamic interfaces, web applications are similar to typical desktop applications with the difference of being executed in a completely different environment, a shared and easy to access environment, being the browser the main application to access any web-based application. Often called services, web-based applications provide functionality similar to desktop applications and they are mostly free software. Being Google a pioneer in this field, other large institution have seen an opportunity to distribute their software in an easy and inexpensive way, and allowing them to reach million of users. Even though Web applications are similar to desktop applications, from their development process emerge a new set of challenges caused by the fact that applications are executed in a completely different environment.

This dissertation will focus on the analysis of these challenges, the study and evaluation of technologies available for the implementation of such systems, study of the available interaction styles and the user behavior on this new platform. It will also be made an identification of the new features allowed by Web applications, such as communication and collaboration, an analysis of the visual structure of applications interfaces and especially their architectural organization.

As a result of this dissertation the implementation of a Web-Based tool for editing Prototype Canonical Abstracts models is presented. That way, this tool is intended to expose the potential of Web-Based applications and their challenges, highlighting their advantages over desktop applications.

# Keywords

Web

Desktop

Services

MVC

RPC

Evaluation

Applications

Frameworks

# Resumo

Actualmente o desenvolvimento de aplicações baseadas na Web é uma área em crescimento exponencial, graças aos benefícios que estas trouxeram consigo. Com a crescente utilização da Web e a sua evolução como plataforma, surgiram novas tecnologias que vieram revolucionar o desenvolvimento de aplicações sobre esta plataforma. Com interfaces mais ricas e consequentemente mais dinâmicas, este tipo de aplicações assemelham-se às típicas aplicações Desktop com a diferença que estão a ser executadas em um ambiente completamente distinto, um ambiente partilhado e de fácil acesso, sendo o browser a aplicação universal de acesso a qualquer aplicação Web. Designadas serviços, as aplicações Web fornecem funcionalidades semelhantes às das aplicações Desktop, sendo na maioria das vezes software gratuito. Sendo a Google a grande pioneira nesta área, outras grandes entidades viram aqui a oportunidade de distribuir o seu software de uma forma fácil e barata, ficando esta de imediato disponível a milhões de utilizadores. Embora as aplicações Web se assemelhem às aplicações Desktop, ao seu processo de desenvolvimento surgem um conjunto de novos desafios provocados pelo facto de estas se encontrarem em um ambiente completamente distinto.

# Palavras-Chave

Web

Desktop

Serviços

MVC

RPC

Avaliação

Aplicações

Plataformas

# Agradecimentos

O meu agradecimento é dirigido ao Professor Doutor Leonel Nóbrega por ter aceitado o convite de ser meu orientador neste projecto e consequentemente pelo tempo e esforço dedicados.

Agradeço aos meus pais e irmãos por todo o apoio e confiança dada ao longo do meu percurso académico, bem como aos meus amigos e colegas pela amizade e companheirismo demonstrados.

# Índice

# Índice de Figuras

# Índice de Tabelas

# 1 Introdução

Inicialmente criada para dar acesso a documentos remotos, hoje em dia a Web é uma plataforma que permite a execução de aplicações, as designadas *Rich Internet Applications* (RIA's). Tais aplicações deram vida a novas formas de desenvolver e estruturar um produto de software, fortemente baseada em arquitecturas Cliente/Servidor e onde a noção de Serviço se impõe como elemento facilitador da reutilização. Além disso, este tipo de aplicações constitui uma solução à uma questão que já há algum tempo era colocada: Como distribuir o software de uma forma prática, simples e barata? Até à data as aplicações desenvolvidas eram executadas localmente (aplicações Desktop), pelo que a distribuição tinha de ser realizada pelos seus utilizadores, tendo estes a responsabilidade de efectuar a respectiva manutenção/actualização. A evolução do conceito Web para plataforma veio responder à esta questão de uma forma surpreendentemente positiva, uma vez que disponibilizando a software através da Web era possível usufruir de todas as suas vantagens.

Muitas entidades apercebendo-se do poder desta nova plataforma, viram aqui não só uma grande oportunidade de disponibilizar os seus serviços, como também criar um novo conjunto de serviços que tirassem partido de todas as vantagens por ela oferecidas, que serão especificadas mais adiante.

## 1.1 Motivação

Com a evolução da Web surgiram um conjunto de novas tecnologias responsáveis pelo desenvolvimento das RIA's, permitindo que estas viessem simular o comportamento das aplicações Desktop sobre um ambiente em verdadeira revolução. Tais tecnologias trouxeram consigo uma nova perspectiva de cliente, fornecendo um conjunto de novas funcionalidades e potencialidades, permitindo a divisão de tarefas entre o lado cliente e servidor, nomeadamente no processamento de dados e na dinâmica de interacção com o utilizador. Além de reaproveitar todas as características das aplicações Desktop, as RIA's, pelo facto de serem executadas sobre a Web, conseguem usufruir de um conjunto de vantagens que as tornam ainda mais atractivas. Vantagens como acessibilidade, facilidade de manutenção ou ainda mesmo comunicação e partilha de tarefas, dado que se trata de um ambiente partilhado onde a colaboração é um factor muito presente, fazem com que as RIA's sejam cada vez mais adoptadas por inúmeras entidades, com as mais variadas finalidades (perspectiva de negócio, entretenimento, cultura, entre outros).

Esta Tese vem, deste modo, apresentar os principais desafios que se colocam no desenvolvimento das aplicações baseadas na Web, evidenciando as suas vantagens e consequentes desvantagens, especificando e justificando as alternativas tecnológicas existentes, analisando estilos de interacção e possíveis soluções arquitecturais.

Por outro lado e de acordo com as razões acima referidas, foi também vista aqui a oportunidade de, em contributo à Engenharia de Software, desenvolver uma aplicação baseada na Web que desse suporte ao processo de desenvolvimento de software. Sendo este um processo complexo, onde a comunicação e colaboração são factores fundamentais, seria

interessante e certamente uma mais-valia implementar uma ferramenta que desse suporte à descrição de soluções bem como mecanismos que permitissem publicar e partilhar as mesmas. Assim, como resultado desta Tese foi implementada uma ferramenta para a edição de modelos, mais concretamente os PAC's (1) (Protótipos Canónicos Abstractos), sendo esta uma ferramenta que, para além da possibilidade de edição de modelos, fornece como base um conjunto de potencialidades tais como comunicação e partilha de informação.

## 1.2 Contribuição

A realização desta Tese permitiu um conjunto de contribuições, nomeadamente:

1. Identificação e descrição das vantagens/desvantagens das aplicações Web;
2. Identificação dos estilos de interacção nas aplicações Web;
3. Descrição de uma Framework de avaliação para plataformas Web;
4. Implementação de uma ferramenta baseada na Web para a modelação de Protótipos Canónicos Abstractos.

## 1.3 Organização

Esta Tese está organizada de acordo com um conjunto de capítulos especificados de seguida:

No capítulo 2 é feita uma abordagem inicial às aplicações Web. São estudadas as tecnologias mais relevantes e feitas comparações entre as aplicações Web e aplicações Desktop, estudando os estilos de interacção mais relevantes. Neste capítulo é ainda descrita uma Framework de avaliação de plataformas Web na qual é baseada a decisão da escolha da plataforma a utilizar para a implementação da aplicação proposta.

No capítulo 3 é então apresentada a proposta de implementação da ferramenta já referida. São definidas e descritas as principais funcionalidades da aplicação bem como o estudo e desenho da sua interface geral. A nível de implementação são apresentadas diversas vistas da arquitectura da aplicação de forma a especificar a sua estrutura e consequentemente o seu comportamento, sendo também apresentadas e descritas as bibliotecas utilizadas.

No capítulo 4 são apresentados finalmente os resultados finais da ferramenta implementada, dando uma noção geral da sua apresentação, abordando algumas das suas principais funcionalidades.

No capítulo 5 são apresentadas as conclusões finais da dissertação resumindo tudo o que foi discutido durante a mesma, bem como as perspectivas futuras tendo como base tudo o que foi feito. São também apresentadas as principais dificuldades encontradas durante a elaboração da dissertação, nomeadamente na implementação da ferramenta.

# 2 Estado da Arte

## 2.1 Introdução

Neste capítulo será feita uma análise àquilo que tem sido desenvolvido na área das aplicações Web. Desde a análise de plataformas já criadas para dar suporte ao desenvolvimento de aplicações Web, o estudo da migração de aplicações Desktop para o ambiente Web, estilos de interacção entre os diferentes ambientes ou ainda o comportamento do utilizador perante os distintos contextos, é pretendido focar as principais dificuldades e os principais desafios que se colocam ao desenvolvimento de aplicações Web.

## 2.2 RIA's – Rich Internet Applications

As aplicações Web convencionais consistem numa arquitectura cliente/servidor com uma comunicação baseada em request/response, onde o cliente faz um conjunto de pedidos ao servidor e este fornece as respectivas respostas. Todo o processamento da informação é feito do lado do servidor e o cliente simplesmente tem a função de mostrar os resultados ao utilizador final.

O surgimento de novas tecnologias tais como o AJAX (Asynchronous Javascript and XML) (2), Flash (3), JavaFX (4), entre outros, permitiu o desenvolvimento das *RIA's* muito semelhantes às aplicações Desktop, fornecendo aos utilizadores uma maior experiência de utilização, novas formas de interacção, maior satisfação e consequentemente maior produtividade. Embora em muitos dos casos seja necessário a instalação de um plugin no browser para que a aplicação possa ser correctamente executada, como é o caso das aplicações baseadas no Flash, Silverlight (5) ou mesmo Java (6), algumas plataformas já se baseiam na tecnologia AJAX, não sendo necessário neste caso a instalação de plugins uma vez que a aplicação já é implementada na(s) linguagens(s) suportadas directamente pelo browser.

### 2.2.1 Desktop vs Web

Com a crescente importância da Web como plataforma, criou-se a oportunidade de migrar aplicações Desktop, aplicações estas desenvolvidas para serem inteiramente executadas localmente, para um ambiente partilhado, a Web. Esta nova aproximação trouxe consigo novos desafios, na medida em que as aplicações deixam de estar disponíveis localmente apenas para um utilizador, para serem partilhadas por milhões de utilizadores sobre a plataforma Web. Estes desafios traduzem-se não só numa alteração radical nas arquitecturas das aplicações, pois novas questões têm de ser ponderadas, a nível de armazenamento de dados, comunicação cliente/servidor, segurança e integridade da aplicação dado que se trata de um ambiente partilhado, como também a nível de interacção com o utilizador pois, com a crescente utilização de aplicações Web novas técnicas de interacção foram desenvolvidas na procura de melhorar a experiência de utilização.

#### 2.2.1.1 Vantagens e Desvantagens

Embora as aplicações Web tenham características muito semelhantes às aplicações desktop, estas possuem alguns aspectos muito distintos derivados do seu ambiente de execução. As aplicações Web possuem algumas desvantagens, sendo as mais importantes:

> Segurança: Uma vez que as aplicações Web são executadas em um ambiente partilhado, os dados do utilizador estão mais vulneráveis a ataques, pelo que surge a necessidade de criar mecanismos de segurança mais fiáveis. Nas aplicações Desktop, uma vez que os dados são mantidos localmente, são menos susceptíveis de ataque.
>
> Performance: Como na base das aplicações Desktop encontra-se uma arquitectura cliente/servidor em que a comunicação é feita através da rede, a performance da aplicação irá depender da largura de banda disponível ao utilizador. No caso das aplicações Desktop, uma vez que o *front-end* e o *back-end* da aplicação encontram-se no mesmo local físico, a performance será mais eficiente.

No entanto é maior o número de vantagens em relação às aplicações Desktop:

> *Acessibilidade:* As aplicações desktop necessitam de ser instalados individualmente em cada computador sendo esta tarefa da responsabilidade do utilizador, enquanto uma aplicação Web ao ser instalada num servidor fica automaticamente disponível a milhões de utilizadores, que a podem aceder através de um browser.
>
> Multi-Plataforma: Uma vez que as aplicações Web são executadas no browser, e atendendo que este já é software integrante de qualquer sistema operativo, significa que as aplicações Web podem ser facilmente acedidas independentemente do sistema operativo em uso. No caso das aplicações Desktop, dependendo da linguagem em que estão implementados, estão limitados a determinados sistemas operativos.
>
> Fácil manutenção: Como as aplicações Web estão alojadas em um servidor remoto, todas as actualizações/modificações são feitas na versão existente no servidor, pelo que os utilizadores irão ter sempre acesso à versão mais actualizada da aplicação. No caso das aplicações Desktop, é da responsabilidade do utilizador realizar a manutenção e actualização do mesmo.
>
> Espaço: estando as aplicações Web instaladas em um servidor remoto, estas não ocupam qualquer espaço no computador do utilizador, enquanto que as aplicações Desktop exigem espaço disponível no computador do utilizador.

Deste modo tendo em conta a perspectiva do utilizador, este não necessita de:

- Instalar a aplicação;
- Realizar as actualizações quando necessárias;
- Ou ainda realizar qualquer tipo de configurações.

O utilizador apenas necessita de possuir um browser para poder ter acesso a uma qualquer aplicação Web.

#### 2.2.1.2 *Estilos de interacção*

As aplicações Web têm tido desde sempre o objectivo de "imitar" as aplicações desktop sobre a Web, pelo que na maioria das vezes a primeira assemelha-se muito ao último, com a diferença de que estão a ser executadas em ambientes completamente distintos. Com a migração das aplicações Desktop para o ambiente Web, muitos dos estilos de interacção têm também sido fielmente "transportados". Todavia, embora os diferentes estilos tenham vindo enriquecer a interacção e as próprias interfaces das aplicações Web é necessário ter sempre alguns cuidados na sua utilização, pois trata-se de um ambiente diferente, com características diferentes e onde os utilizadores possuem também um comportamento diferente.

De entre os diferentes estilos de interacção tais como menus flutuantes, ícones, botões, manipulação directa e, tratando-se neste caso de aplicações Web, é importante evidenciar as vantagens e desvantagens dos estilos mais relevantes a fim de concluir que forma é que estes podem influenciar positiva ou negativamente na organização e compreensão das aplicações.

De seguida são especificados alguns dos estilos de interacção mais usados:

##### Menu Flutuante

Trata-se de um estilo de interacção de grande importância, pois este permite organizar conjuntos de funcionalidades em categorias, organizando-as e tornando-as mais perceptíveis ao utilizador. No entanto, se mal utilizados, este tipo de menus poderão se tornar confusos se possuírem demasiada profundidade, ou seja, se contiverem demasiadas funcionalidades. Neste caso, será mais difícil ao utilizador procurar por determinada funcionalidade traduzindo-se numa maior complexidade da aplicação.

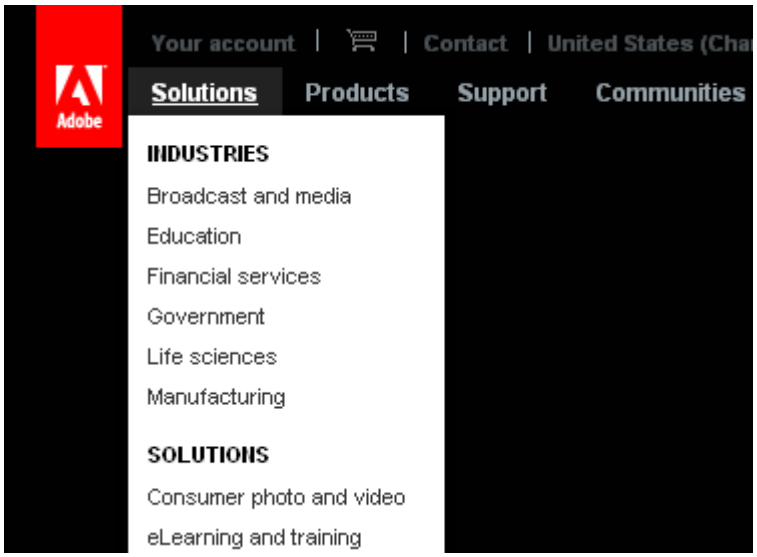


Figura 1 – WebSite da Adobe – Menu Flutuante (3)

Este tipo de estilo de interacção sofreu algumas evoluções favoráveis, na medida em que este é apresentado numa organização em abas, onde cada aba apresenta vários conjuntos de

funcionalidades relacionadas entre si, cada uma delas é acompanhado de um *icon* para um mais rápido e fácil reconhecimento da funcionalidade por parte do utilizador.

Este tipo de menu pode ser visualizado nas ferramentas disponibilizadas pelo Office 2007.

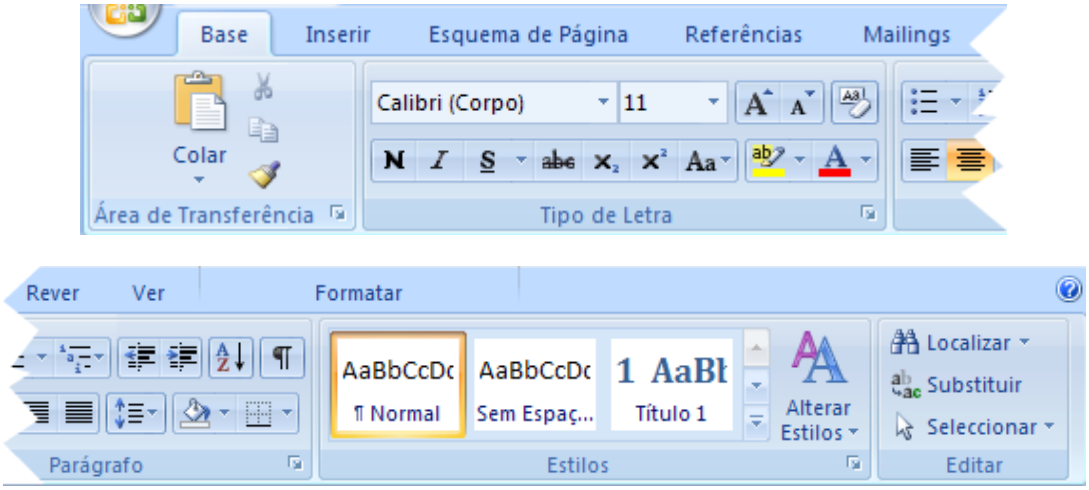


Figura 2 – Word 2007 [Menu Flutuante]

Pela observação das duas figuras anteriores é possível comparar e concluir que a evolução do estilo de interacção em causa é satisfatória uma vez que a segunda mostra-se mais elegante e intuitiva que a primeira.

Embora este estilo de interacção esteja muito presente nas aplicações Web actuais, é necessário, pelas razões já acima referidas, ter alguns cuidados na sua utilização. Na Web a simplicidade e facilidade de compreensão são factores de sucesso das aplicações, e como tal, é necessário apresentar de uma forma simples e clara aquilo que a aplicação é capaz de fazer. Menus com palavras intuitivas e de pouca profundidade tem maior probabilidade de sucesso.

## Manipulação Directa - "Drag-andDrop"

A manipulação directa de objectos ou também conhecido como *drag-and-drop* (termo em inglês) é um dos estilos de interacção mais interessantes, na medida em que este permite, tal como o nome indica, a interacção directa de determinados objectos na aplicação. Esta funcionalidade é muito útil, por exemplo, em editores de imagens ou mesmo em aplicações e-commerce onde os utilizadores arrastam os seus itens de compra para o "carro de compras" virtual.

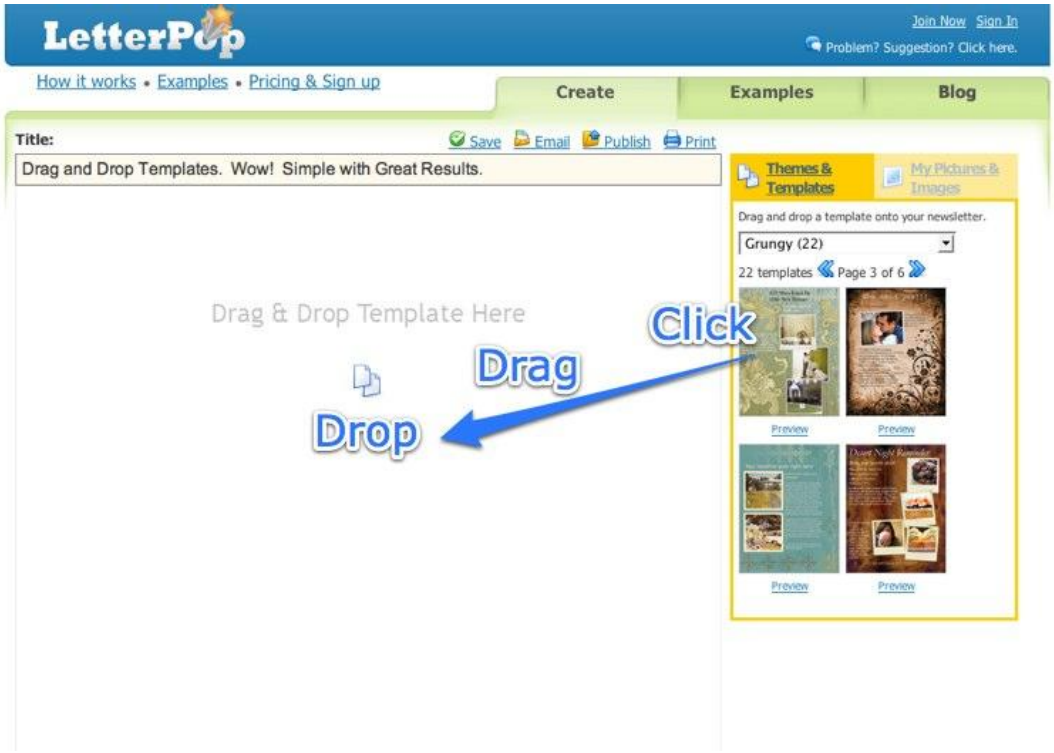


**Figura 3 – Exemplo estilo Manipulação Directa (Drag-and-Drop) (57)**

Este estilo oferece algumas vantagens tais como a capacidade visualização do objecto que se pretende manipular e fácil reconhecimento do conceito que este representa, sendo também um estilo de fácil aprendizagem. No entanto é sempre um mecanismo de difícil implementação.

No caso de uma aplicação Web, pela fácil utilização e o claro feedback que oferece este é um estilo a ter em importante consideração, tendo sempre em mente o tipo de aplicação e o contexto em que é aplicado.

## Teclas de Atalho

Seja numa aplicação Desktop ou numa aplicação Web, este é um estilo de interacção que não pode ser esquecido. Embora exija algum treino, a medida que um utilizador começa a ganhar experiência na aplicação, as teclas de atalho tornam-se facilitadores na utilização da ferramenta aumentando a eficiência e produtividade do utilizador. Uma vez que as aplicações Web são executadas dentro do browser, este por si só já contém as suas teclas de atalho, pelo que o programador tem de ter algum cuidado na escolha de teclas de atalho, de forma a não ocorrerem inconsistências durante a sua execução.

## Link

Este é um dos estilos de interacção mais importantes e mais interessantes de se observar na medida em que tem sofrido ao longo do tempo alterações/extensões no seu conceito de acordo com o contexto em que se encontra.

O *link* (termo inglês) ou hiperligação é um dos estilos de interacção mais utilizados na Web. Estes são usados na construção de páginas Web e consistem em elementos clicáveis em forma de texto ou imagem direccionando o utilizador para outras páginas ou dando acesso a um conjunto de recursos variados (7).

Esta é a definição por defeito para o conceito de link, que o identifica como sendo um elemento de navegação. De acordo com esta, o utilizador sempre que clica num link é direccionado para outra página.

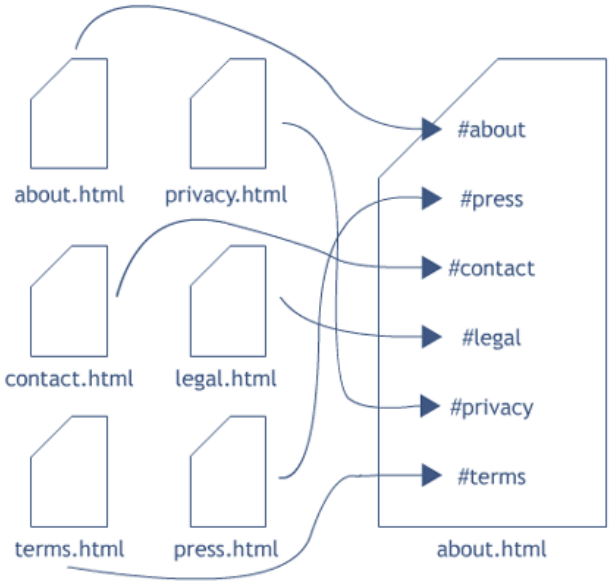


**Figura 4 – Representação lógica do funcionamento de uma hiperligação (58)**

Deste modo e durante muito tempo o modelo mental do utilizador associado ao link podia ser representado de acordo com a figura acima ilustrada.

No entanto, com a evolução da Web a definição de link também evoluiu. Com o aparecimento dos RIA o modelo mental do utilizador sofreu algumas transformações, isto porque as tecnologias usadas para o seu desenvolvimento permitiram que, dada uma acção do utilizador, em vez de se proceder a um refrescamento total da aplicação, fosse apenas actualizada uma parte da página/aplicação de acordo com a acção realizada (através de requisições assíncronas). Assim sendo, o link deixa de ser visto apenas como um elemento de navegação e passa a ser visto também como um elemento para evocação/requisição de serviços, pelo o que pode ser considerado neste último caso como um simples botão.

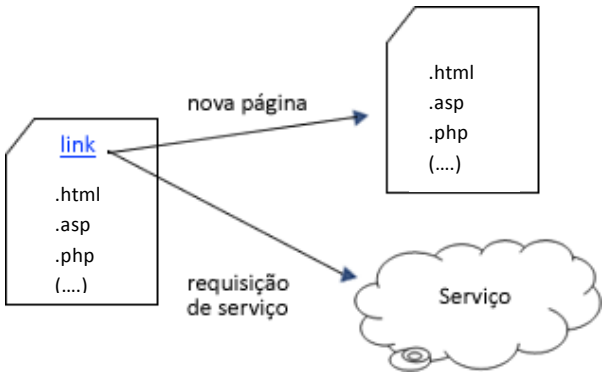


Figura 5 – Representação lógica da evolução do conceito de Link

É de realçar que nos casos de links, botões ou outro tipo de estilo de interacção que provoquem acção na aplicação, no caso em que são feitos apenas pequenos refrescamentos da mesma, é fundamental realçar tais alterações de forma a que o utilizador se aperceba das transformações ocorridas na aplicação.

Todos os estilos de interacção apresentados são já suportados sobre a Web pelo que podem ser utilizados no desenvolvimento de qualquer aplicação baseada na plataforma. A sua utilização deve ser feita tendo sempre em conta o ambiente em que são utilizados e o comportamento esperado do utilizador.

#### 2.2.1.3 Comportamento do utilizador

Embora as aplicações Web tentem simular o comportamento das aplicações Desktop, o comportamento do utilizador sobre ambos os ambientes não são o mesmo. Um dos principais factores de sucesso de uma aplicação Web é a sua simplicidade de utilização. No caso das aplicações Desktop, estas são sempre acompanhados de um guia/manual de utilização na qual o utilizador faz uso a fim de aprender a utilizar a ferramenta. Tratam-se muitas vezes de manuais extensos na qual o utilizador tem de investir algum do seu tempo no processo de aprendizagem.

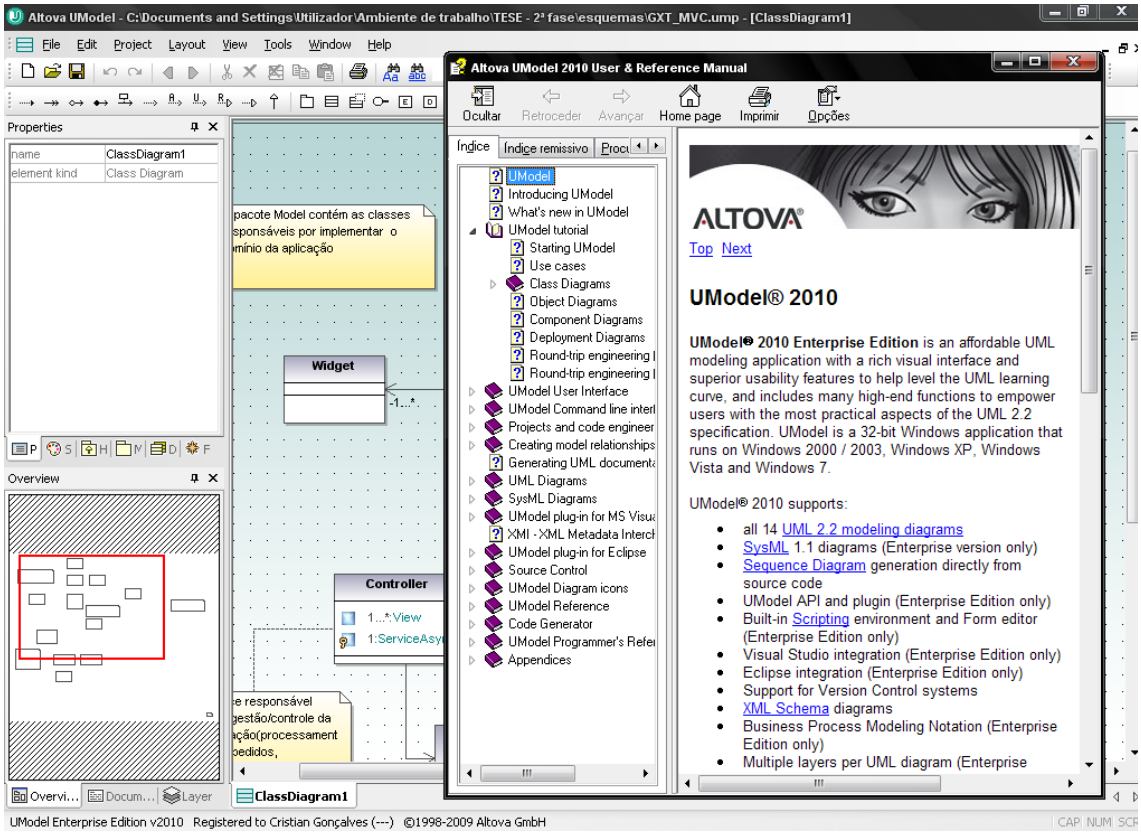


Figura 6 – Altova UModel (63) – Manual de Utilização

Nas aplicações Web, dado o conjunto de aplicações/soluções disponíveis para a execução de determinada tarefa, o utilizador não mostra grande disponibilidade de aprendizagem. Deste modo, ao deparar-se com uma aplicação mais complexa este irá logo à procura de uma ferramenta mais acessível, de mais simples utilização e que execute o mesmo tipo de tarefa. Assim, para cativar desde o início a atenção do utilizador, a maioria das aplicações Web têm na sua página de entrada uma frase ou um breve vídeo que resume de forma explícita para que serve a ferramenta, acompanhada dos designados "*quick tutorials*" que normalmente consistem em pequenos vídeos ou textos de como utilizar a ferramenta.

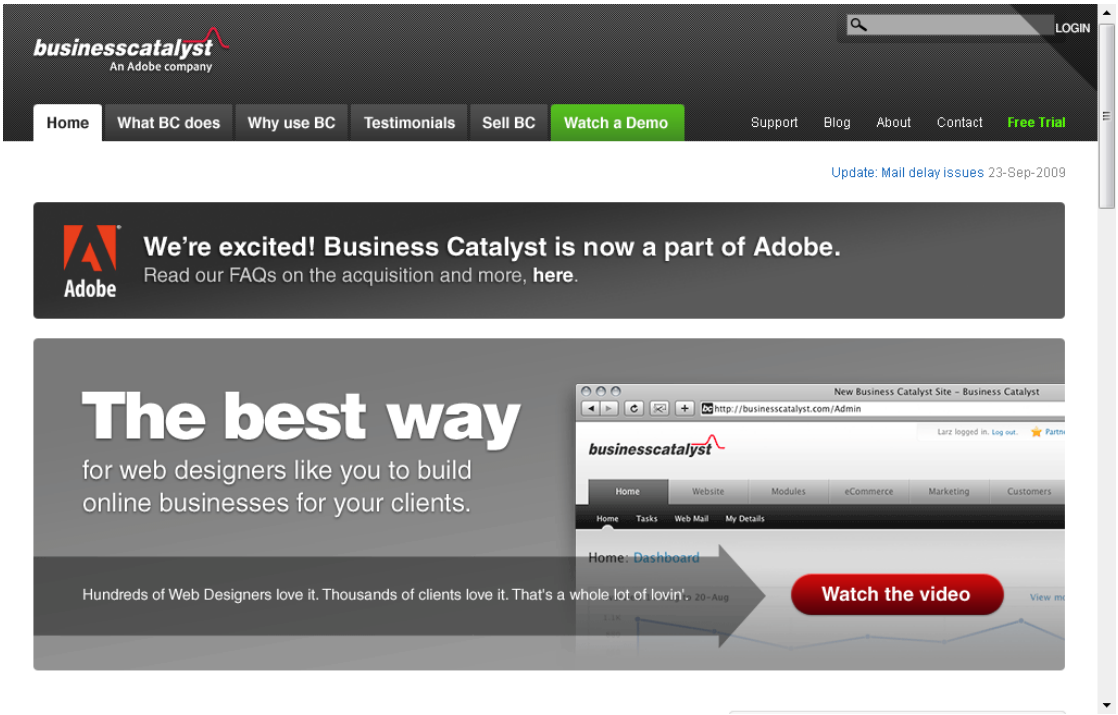


Figura 7 – Aplicação Web "BusinessCatalyst" (64) – Página de Entrada

Por outro lado, dado que a Web se caracteriza como um ambiente dinâmico, onde a comunicação e colaboração são características sempre presentes, é importante fornecer ao utilizador mecanismos de partilha, comunicação e colaboração. O grande sucesso da Web deve-se muito ao seu lado social, ou seja, a facilidade com que esta permite que as pessoas comuniquem e colaborem entre si. Assim, tal como já foi referido anteriormente, ao desenvolver aplicações sobre a Web é importante aproveitar as suas melhores características. Certamente a comunicação/colaboração são características fundamentais e quando inseridos em qualquer ferramenta são sempre uma mais-valia.

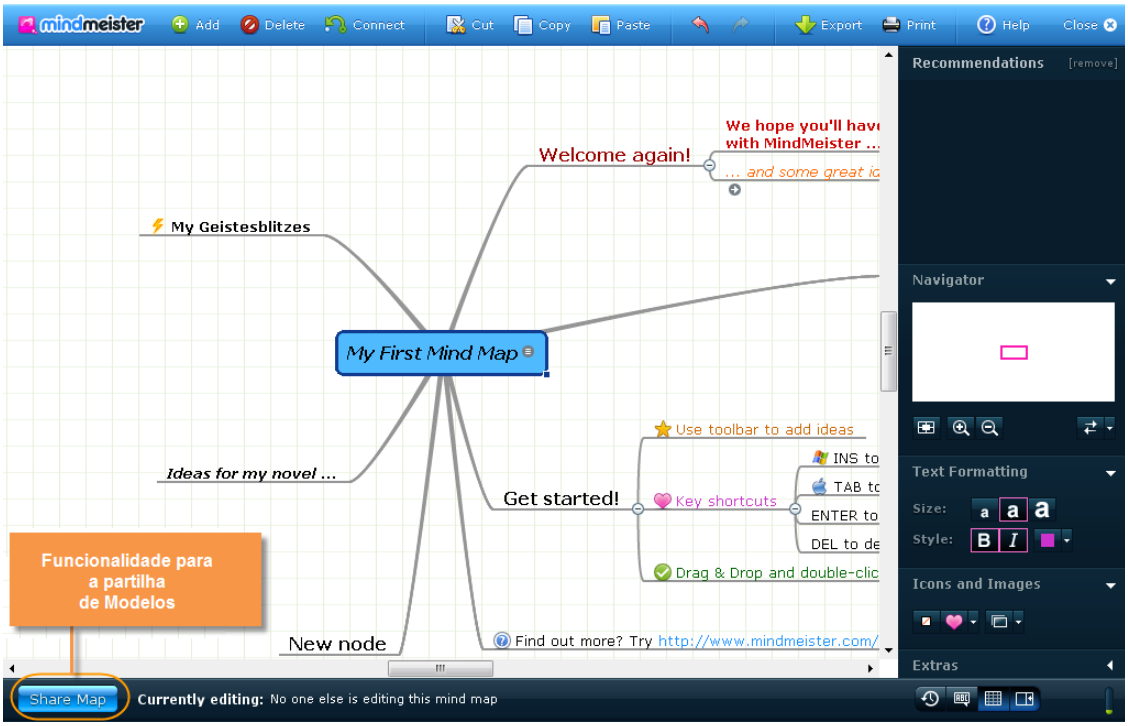


Figura 8 – MindMeister (65) – Partilha de Modelos

Existem várias técnicas que podem ser utilizadas para permitir ao utilizador partilhar o seu trabalho ou convidar outros para colaborarem consigo. Desde simples funcionalidades que permitem publicar o trabalho realizado pelo utilizador na ferramenta ou funcionalidades mais complexas que permitem colaboração em *runtime*, os designados *Real-Time Group Editors* (8), as ferramentas devem sempre tomar proveito do lado social que o ambiente Web oferece.

## 2.3 Tecnologias Web relevantes

### 2.3.1 Arquitectura Model View Controller (MVC)

O padrão MVC (9) consiste num padrão arquitectural baseado em três componentes: Model, View e Controller. Embora possa ser confundida com um padrão de desenho, este é um padrão arquitectural na medida em que aborda as arquitecturas das aplicações, gerindo a sua organização. Este padrão surgiu da necessidade da separação das aplicações em camadas distintas, existindo uma separação de responsabilidades por cada camada. Esta separação permite uma maior organização das aplicações, apelando a modularização e à redução de acoplamento entre as componentes. Desde o seu aparecimento surgiram diversas interpretações da arquitectura MVC, abordando-se diversas perspectivas das possíveis relações entre as componentes mas sem nunca alterar as responsabilidades de cada uma delas. Deste modo, uma possível representação da arquitectura MVC pode ser dada pela figura 9:

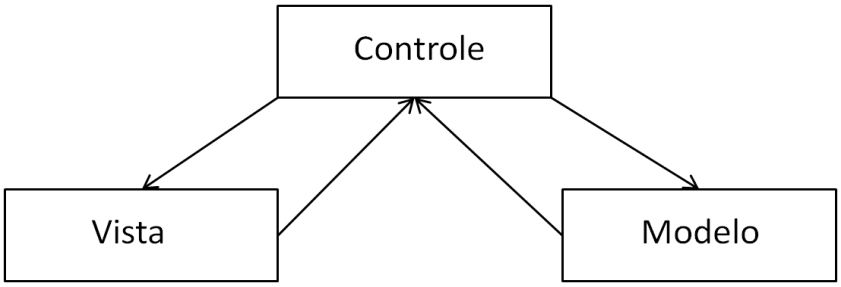


Figura 9 – Arquitectura MVC

De acordo com esta representação, a camada controle é a responsável por gerir as relações entre a vista e o modelo. A camada controle é responsável por tratar os pedidos vindos dessas camadas e pelo processamento de toda a informação, actualizando as anteriores sempre que necessário. A vista consiste na componente visual da aplicação, ou seja, na interface gráfica apresentada ao utilizador, delegando para o controle qualquer acção submetida por ele, enquanto o modelo representa os dados da aplicação, garantindo a persistência de dados.

De acordo com esta arquitectura, não existe qualquer relação directa entre o modelo e a sua representação (vista), pelo que ambas podem ser desenvolvidas de forma independente, não se correndo o risco de ao se alterar uma das camadas afectar a performance da outra. Assim sendo, é possível alterar a vista sem a necessidade de alterar a camada modelo ou vice-versa.

Outra representação mais comum e convencional da arquitectura MVC é dada pela seguinte figura:

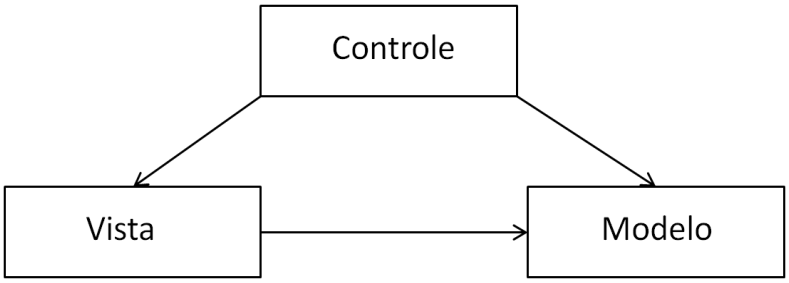


Figura 10 – Arquitectura MVC (66) – Representação mais comum

Neste caso é possível observar a existência de uma relação entre as camadas vista e modelo. No entanto, de forma a manter a independência entre as camadas é normalmente utilizado o padrão *Observer,* responsável por gerir a relação entre ambas, sendo possível estabelecer-se uma relação de subject/subscriber. Deste modo, a vista associa-se ao modelo sem que este tenha necessidade de saber, neste caso, quem o representa. Mais uma vez é possível realizar alterações numa das camadas sem a necessidade de se alterar a outra.

Inicialmente criada para dar apoio ao desenvolvimento de aplicações Desktop, a arquitectura MVC ganhou grande importância no seio das aplicações Web. O sucesso e os benefícios do MVC na separação das aplicações em camadas e a necessidade de escalabilidade das aplicações Web (uma vez que o lado cliente e o lado servidor encontram-se em locais físicos distintos e por consequência, geridos por máquinas distintas) fizeram com que o MVC se tornasse uma arquitectura fundamental neste tipo de aplicações.

No entanto, para adequar a arquitectura MVC às características das aplicações Web, foi necessário a introdução de algumas alterações, representadas na figura 11:

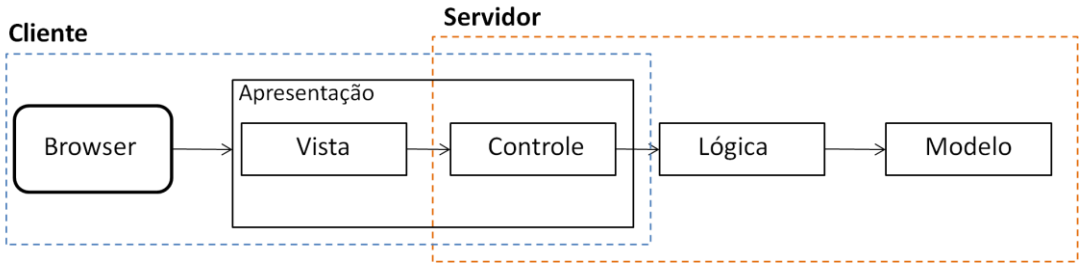


Figura 11 – Arquitectura MVC em aplicações Web

A arquitectura MVC nas aplicações Web permite a separação das camadas pelas diferentes máquinas. Como pode ser observado, no lado cliente temos o browser como a aplicação universal para correr as aplicações Web. Este tem acesso à camada apresentação da aplicação, que consiste numa arquitectura MVC simplificada apenas com vista e controle, onde a vista consiste geralmente em páginas HTML, estendendo-se o controle até ao lado servidor onde se encontram a lógica de dados e o modelo da aplicação. Deste modo o controle opera tanto do

lado cliente como do lado servidor, sendo responsável por transformar os eventos do utilizador em serviços que serão processados no lado servidor e, por realizar as actualizações na vista e no modelo sempre que necessário. O processamento de serviços/pedidos implica o acesso a camada lógica da aplicação que por sua vez implementa as regras de negócio da aplicação e abstrai o acesso à base de dados.

Este padrão arquitectural tornou-se de tal forma fundamental no desenvolvimento das aplicações Web, que actualmente as plataformas que dão suporte ao desenvolvimento deste tipo de aplicações já oferecem implementações por defeito do padrão, tal como é o caso do *Ruby on Rails* (10), *ASP.NET MVC* (11)*, CakePhp* (12), entre outros.

### 2.3.2 Cloud Computing

*Cloud Computing* ou Computação em Nuvem consiste basicamente na disponibilização de um conjunto de serviços através da Web (13). De acordo com este conceito, todas as aplicações que o utilizador necessita estão armazenados remotamente pelo que podem ser acedidos através da Web. Fala-se sobre a possibilidade de no futuro os Sistemas Operativos serem disponibilizados através de *cloud computing*, pelo que os utilizadores apenas necessitarão de ter instalado no seu computador uma aplicação para aceder ao sistema operativo, que por sua vez estará armazenado remotamente na Web. Para que isto aconteça, é necessária a existência de um servidor central que contenha uma infra-estrutura responsável por disponibilizar e gerir os recursos necessários, principalmente a nível de tráfego, espaço disponível e segurança nos dados dos utilizadores, implementando mecanismos de redundância para que em caso de falha estes não percam os seus dados.

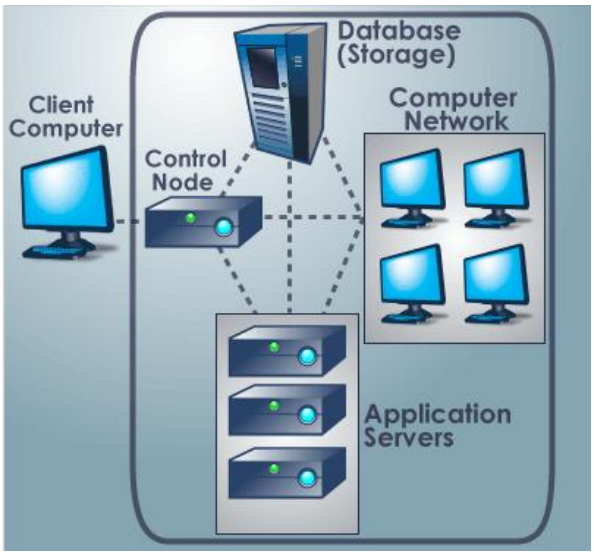


Figura 12 – Computação em Nuvem (61)

O surgimento do *Cloud Computing* (14) vem incrementar ainda mais a importância das aplicações Web, uma vez que estas representam os principais recursos disponibilizados por esta nova tecnologia.

O *Google Application Engine* (GAE) (15) da *Google* ou o *Windows Azure* (16) da *Microsoft* são exemplos de tecnologias *Cloud Computing* já existentes, verificando-se assim a forte aposta de grandes entidades nesta tecnologia, factor este que irá certamente influenciar o seu crescimento e consequente sucesso no futuro.

### 2.3.3 Browser

O browser é a aplicação através da qual acedemos às aplicações Web e, como tal, as suas características irão influenciar o desenvolvimento e a própria utilização das respectivas aplicações. É por isso importante entender as suas funcionalidades mais importantes a fim de saber qual o impacto que estas podem ter no funcionamento de uma aplicação Web.

#### *2.3.3.1 Operação Retroceder (Undo)*

O browser como aplicação contém um conjunto de funcionalidades utilizadas para configuração do próprio browser ou para funções de navegação. Sendo o botão retroceder uma funcionalidade de navegação, é uma das funcionalidades mais utilizadas no browser pois de acordo com o seu conceito convencional, permite ao utilizador voltar às últimas páginas consultadas através de um simples clique. No entanto, com o surgimento das aplicações Web, a utilização desta funcionalidade tornou-se crítica. Como já foi referido anteriormente, as aplicações Web vieram alterar o modelo mental dos utilizadores uma vez que o acto de clicar num botão ou link não implica necessariamente uma mudança de página, uma vez que a informação pode ser tratada na mesma. Assim sendo, perante uma aplicação Web, o acto de clicar no botão retroceder poderá não implicar numa mudança de página. Fazendo a analogia com as aplicações Desktop, aqui, o botão retroceder tem como função eliminar a última acção do utilizador. Deste modo o botão retroceder numa aplicação Web terá de ter o mesmo efeito.

Perante esta questão existem várias soluções possíveis:

1. Criar o botão retroceder na própria aplicação e desactivar o botão retroceder do browser. Esta é uma boa solução uma vez que a funcionalidade fica mais contextualizada na aplicação, tornando-se mais intuitiva ao utilizador.

2. "Programar" o botão retroceder do browser no contexto da aplicação. Deste modo o botão retroceder do browser irá ter um comportamento na aplicação Web igual ao botão retroceder numa aplicação Desktop. Esta também é uma boa solução, no entanto poderá não ser tão intuitivo para o utilizador.

3. No caso de não serem implementados mecanismos de undo/redo na aplicação, desactivar o botão retroceder do browser ou implementar mecanismos para que o utilizador não perca dados. Para tal podem ser elaboradas por exemplo mensagens de alerta, informando o utilizador que irá perder os seus dados caso confirme a operação.

#### *2.3.3.2 Espaço vs Informação*

Outro dos desafios que o browser coloca é o espaço. O próprio browser já ocupa espaço no monitor pelo que ao desenvolver uma aplicação Web o programador terá de ter em conta o espaço restante disponível. Embora novos browsers, como é o caso do Google Chrome, já tentem minimizar o espaço ocupado, este é sempre um desafio que se coloca na organização da componente visual das aplicações.

Numa página Web quando existe muita informação a mostrar, utiliza-se o mecanismo de *scroll* do browser a fim de conseguir navegar em toda a página. Numa aplicação Web esta técnica não é aconselhada uma vez que podem existir funcionalidades importantes que podem deixar de ficar visíveis ao utilizador. Como solução é preferível criar "janelas internas" onde nova informação pode ser disponibilizada ao utilizador e ainda criar efeitos de maximização/minimização das mesmas de forma a criar reaproveitamento de espaço.

## 2.4 Plataformas Web

O número de plataformas disponíveis para o desenvolvimento de aplicações Web é vasto, pelo que é necessário recorrer a selecção de uma das plataformas. Tal selecção deve ser realizada de acordo com um conjunto de atributos, atributos estes relacionados com o tipo de aplicação a desenvolver. Atendendo aos objectivos irá ser avaliado um subconjunto das plataformas existentes considerando o suporte para tipo de aplicação pretendida.

### 2.4.1 Framework de avaliação

De seguida é proposta uma Framework de avaliação que irá ser utilizada neste contexto para avaliar um conjunto de plataformas, de forma a tentar concluir qual a plataforma que melhor oferece suporte ao desenvolvimento da aplicação desejada.

A Tabela 1, originalmente criada, especifica um conjunto de atributos que serão analisados na avaliação das plataformas. Estes são atributos qualitativos/quantitativos que não só ajudam na avaliação das principais características tecnológicas das plataformas *Web* como também evidenciam um conjunto de características relevantes para o programador, tais como documentação, linguagens suportadas, entre outros…

**Tabela 1 - *Framework* de Avaliação**

| Atributos | Valores de Medida | Descrição |
|---|---|---|
| ***Dimensão da Comunidade de suporte*** | Pequena, Média, Grande | --- |
| ***Quantidade Documentação disponível*** | Pouca, Razoável, Muita | --- |
| ***Qualidade Documentação disponível*** | Fraca, Razoável, Boa, Muito Boa | --- |
| ***Linguagens suportadas*** | Conjunto das linguagens suportadas pela plataforma | Especifica as principais linguagens suportadas pela plataforma |
| ***Comunicação cliente/servidor*** | Tecnologias utilizadas | Especifica as tecnologias que podem ser utilizadas para a comunicação entre cliente/servidor |
| ***Portabilidade*** | Escala de 1 à 5 | Verifica se é possível executar a plataforma em diferente máquinas e sistemas operativos. |
| ***Usabilidade*** | Escala de 1 à 5 | Verifica se a plataforma oferece uma boa experiência de utilização. |
| ***Facilidade de aprendizagem*** | Escala de 1 à 5 | A facilidade de aprendizagem advém também da usabilidade da mesma |
| ***Facilidade de instalação*** | Escala de 1 à 5 | Facilidade de instalação das ferramentas necessárias para a utilização da plataforma |
| ***Licenças*** | Software Proprietário, Software Livre | --- |
| ***Suporte a Testes*** | Escala de 1 à 5 | Especifica as técnicas de *testing* suportadas pela plataforma |
| ***API's para Componentes Gráficos*** | Especificação das Bibliotecas disponíveis | Especifica as bibliotecas disponibilizadas pela plataforma para a manipulação de componentes gráficos |

Para a avaliação de alguns dos atributos considerados, é utilizada uma escala de 1 a 5. A classificação atribuída a cada um dos valores da escala é a seguinte:

- 1 - Péssimo;
- 2 - Mau;
- 3 - Razoável;
- 4 - Bom;
- 5 – Muito Bom.

Aos atributos avaliados de acordo com esta escala é atribuída exclusivamente um dos possíveis valores.

#### *2.4.1.1 Descrição dos atributos/dimensões*

##### Dimensão da Comunidade de suporte

Ao avaliar uma plataforma de desenvolvimento de *software* é importante saber qual a comunidade que a sustenta e utiliza, e a sua consequente dimensão. É importante saber se a comunidade é participativa, dinâmica, inovadora, trabalhando constantemente na evolução da plataforma, pois desta forma transmite-nos mais confiança na sua utilização, na medida em que sabemos que esta está em constante optimização, não correndo o risco de se tornar obsoleta.

##### Quantidade/Qualidade Documentação disponível

Normalmente a qualidade da comunidade que suporta a plataforma transmite-se na quantidade/qualidade da informação liberada pela mesma. Uma boa plataforma é sempre acompanhada de uma documentação ordenada, clara, correcta e completa, orientando e esclarecendo o desenvolvedor na utilização da mesma. Deste modo, a adopção de uma nova plataforma deve ser sempre influenciada pela quantidade e qualidade da documentação que a sustenta.

##### Linguagens suportadas

Este é um atributo fundamental para o desenvolvedor, na medida em que a escolha da plataforma a adoptar é fortemente influenciada pela linguagem ou conjunto de linguagens que a respectiva suporta. Isto acontece porque o desenvolvedor sente-se sempre mais à vontade com umas linguagens do que outras. Obviamente este não é, ou não deve ser o factor decisivo na escolha de uma plataforma, no entanto, tem sempre grande relevância. As plataformas são mais privilegiadas se suportarem linguagens maduras, expressivas, robustas, que contribuam para um entendimento mais claro da lógica das aplicações desenvolvidas.

##### Comunicação cliente/servidor

Dado o conjunto já existente de tecnologias e técnicas utilizadas para a comunicação entre cliente/servidor, torna-se importante saber quais delas são suportadas pelas plataformas em avaliação, tendo em conta os requisitos de comunicação da aplicação a desenvolver, como por exemplo o tipo de dados a serem transmitidos pela rede, mecanismos de segurança para protecção dos dados, entre outros...

### Usabilidade

Quais as funcionalidades que a plataforma suporta e a forma como estas são disponibilizadas ao desenvolvedor, tentando sempre proporcionar uma melhor experiência de utilização, é um factor determinante na escolha da plataforma, na medida em que a usabilidade contribui para uma maior produtividade e posterior qualidade dos produtos desenvolvidos sobre a plataforma.

### Facilidade de aprendizagem

A facilidade de aprendizagem de uma plataforma advém em muito da sua usabilidade. Uma plataforma fácil de usar é consequentemente fácil de aprender. No entanto, não podemos unificar estas duas dimensões, pois apesar de serem proporcionais estas são também distintas, pelo que, na escolha de uma plataforma o desenvolvedor deve tentar calcular, de acordo com o seu conhecimento, o tempo e o esforço necessário para aprender a trabalhar com a mesma.

### Facilidade de instalação

Uma plataforma distingue-se também pelos recursos e facilidades que oferece para a sua instalação. Normalmente uma boa plataforma tem mecanismos que promovem a sua fácil instalação, bem como mecanismos de gestão que suportam configurações e actualizações automáticas, libertando o utilizador das tarefas de gestão da própria plataforma.

### Suporte a Testes

Sabemos que os testes são fundamentais para o desenvolvimento completo e correcto de qualquer aplicação. Deste modo, torna-se relevante saber que tipo de suporte as plataformas oferecem à integração de testes no desenvolvimento de aplicações, sendo que uma plataforma que ofereça um bom suporte a este nível, facilita em muito a tarefa do desenvolvedor.

### *API's* para Componentes Gráficos

Dado que é desejado desenvolver uma aplicação Web para a edição de modelos, trata-se então de uma aplicação com um elevado nível de interacção com o utilizador, onde este poderá criar ou editar modelos. Daí que seja necessário avaliar qual o suporte que as plataformas oferecem a nível de componentes gráficos, ou seja, quais são as bibliotecas disponibilizadas para o efeito e o tipo de eventos suportados pelos mesmos. Na avaliação em questão, esta será um atributo determinante para a escolha da plataforma a adoptar.

### 2.4.2 Plataformas a avaliar

Dado que é impraticável avaliar todas as plataformas Web existentes e, dado que se pretende seleccionar uma plataforma que seja minimamente madura e fiável, é prudente ter em conta o conjunto de plataformas desenvolvidas e disponibilizadas pelas grandes entidades do mundo do software, como é o caso da Google, da Microsoft e da Adobe, que podem ser considerados pilares fundamentais na área do desenvolvimento Web.

Os factores que contribuíram para a escolha das plataformas a avaliar foram:

- as entidades que as desenvolveram e a sua importância e dinamismo no mundo do software;
- tecnologias suportadas pelas plataformas;
- potencialidades das mesmas;
- comunidades existentes que colaboram para o seu suporte e contínua evolução.

Deste modo foi decidido realizar a seguinte selecção:

1. ***Google Web Toolkit*** (17), *Google*
2. ***ASP.NET*** e ***Silverlight***, *Microsoft*
3. ***Flex***, *Adobe*

Estas são plataformas de grande potencialidade, mostrando-se como grandes candidatas para dar suporte ao desenvolvimento da aplicação Web desejada.

Antes de realizar uma avaliação mais profunda de cada uma das plataformas, utilizando para tal a Framework de avaliação proposta, é apresentada uma visão geral de cada uma delas, como funcionam e quais os principais recursos que oferecem.

#### *2.4.2.1 Plataformas - Visão Geral*

##### Google Web Toolkit – Plataforma da Google

*Como surgiu?*

Esta plataforma foi inicialmente desenvolvida pelos e para os programadores da *Google* (18) que, tendo a experiência de desenvolver aplicações *Web* com tecnologia *AJAX* como o *GMail* (19), o *Google Calendar* (20), o *Google Maps* (21), entre outros, sentiram as dificuldades de utilizar a tecnologia *AJAX*, mais concretamente na programação em *JavaScript*, pois esta não é uma linguagem robusta, acrescida da enorme desvantagem de ser interpretada de forma diferente por cada um dos navegadores *Web* existentes.

Desta forma surgiu o *Google Web Toolkit* (*GWT*), uma plataforma de código aberto criada para o desenvolvimento de aplicações *Web* com tecnologia *AJAX*, com a particularidade de que toda a implementação é feita em Java, ou seja, a aplicação é puramente desenvolvida em *Java* e consequentemente compilada em Javascript para que possa ser então executada via *Web*. Esta é a enorme vantagem do *Google Web Toolkit* - *JAVA*. Depois de desenvolvida e utilizada pelos programadores da *Google* a plataforma foi liberada para a comunidade em geral.

### *Como funciona?*

O *GWT* consiste em um conjunto de ferramentas, *API's* e componentes visuais desenvolvidos para suportar o desenvolvimento de aplicações *Web.* Embora a plataforma possa ser utilizada num qualquer IDE Java, a Google desenvolveu um plugin para o eclipse, permitindo a automatização de muitas das tarefas facilitando assim a vida ao programador.

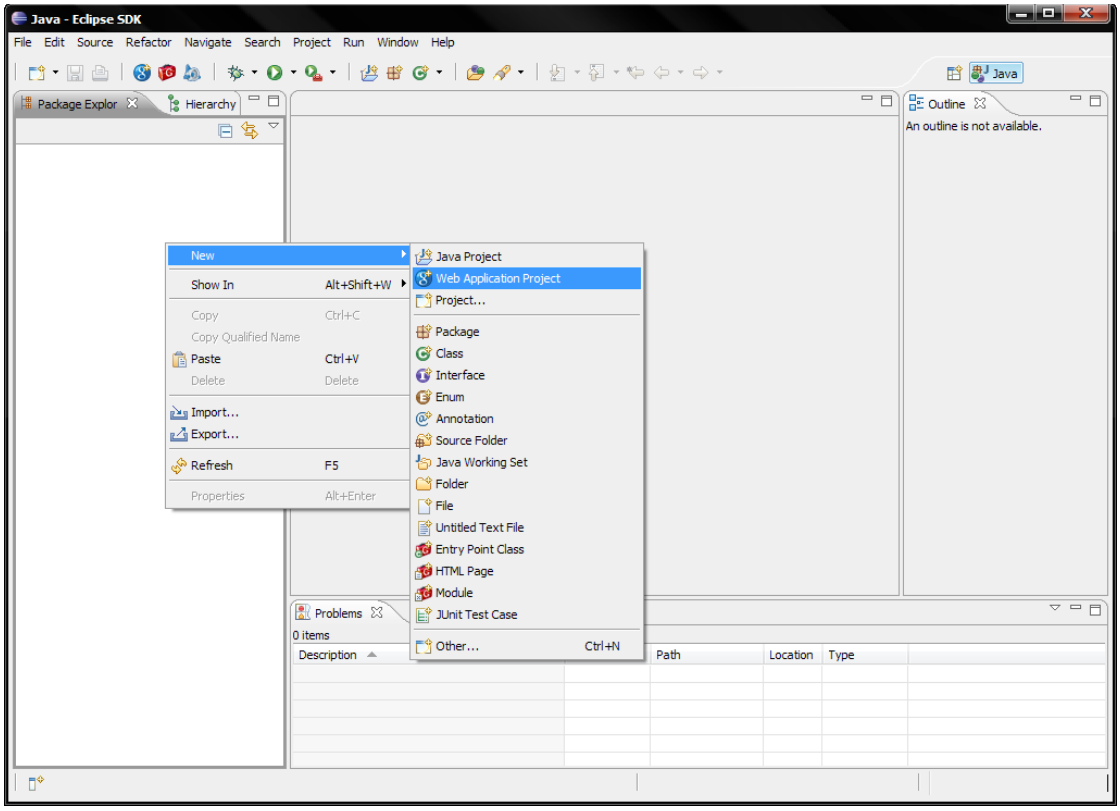


**Figura 13 – Exemplo de utilização do GWT no Eclipse**

### *Java para Javascript*

No GWT, todo o código *Java* criado para a aplicação é posteriormente compilado em código *Javascript* e *HTML* (código do lado cliente) sendo este compatível com os mais diversos browsers, ou seja, o *GWT* responsabiliza-se por colmatar as diferenças entre os vários browsers na interpretação do *Javascript*. O código *Java* do lado servidor será suportado directamente em Java através da integração de uma Java VM no servidor. Para uma melhor performance da aplicação o *GWT*, ao compilar o código *Java* para *Javascript*, percorre todo o código de forma a assegurar que todo o código compilado é executado na aplicação. Se houver alguma secção de código *Javascript* que nunca é executada, esta é simplesmente removida. Este mostra-se mais uma vez um aspecto muito positivo do *GWT*, contribuindo para a performance da aplicação e consequentemente conduzindo a uma melhor gestão da memória.

### *Hosted Mode versus Web Mode*

O GWT disponibiliza dois modos de execução para as aplicações: Hosted Mode e Web Mode. Em *Hosted Mode* o código é compilado e executado em *Java* bytecode pelo *Java Virtual Machine* (*JVM*), sendo que este código é executado num navegador próprio do *GWT*. Este modo é essencialmente utilizado durante o desenvolvimento da aplicação permitindo fazer "*debbuging"* sem a necessidade de andar constantemente a compilar o código *Java* em código *Javascript*, garantindo que quando a aplicação for migrada para a Web irá se comportar de forma adequada.

Em *Web Mode* a aplicação já se encontra sobre a plataforma Web, pelo que todo o código Java foi já compilado em Javascript e HTML para ser correctamente interpretado pelo lado cliente, enquanto o *back-end* da aplicação consiste num servidor com suporte a Java.

### *Comunicação Cliente/Servidor*

A comunicação entre o cliente e o servidor é feita através do mecanismo *RPC* (22) (*Remote Procedure Call*), permitindo que o cliente e o servidor "troquem" objectos *Java* entre si pelo protocolo *HTTP*. Para tal, o modelo de domínio da aplicação tem de implementar Serialização para que, na troca de dados, os objectos possam ser serializados e "deserializados" durante o processo de comunicação.

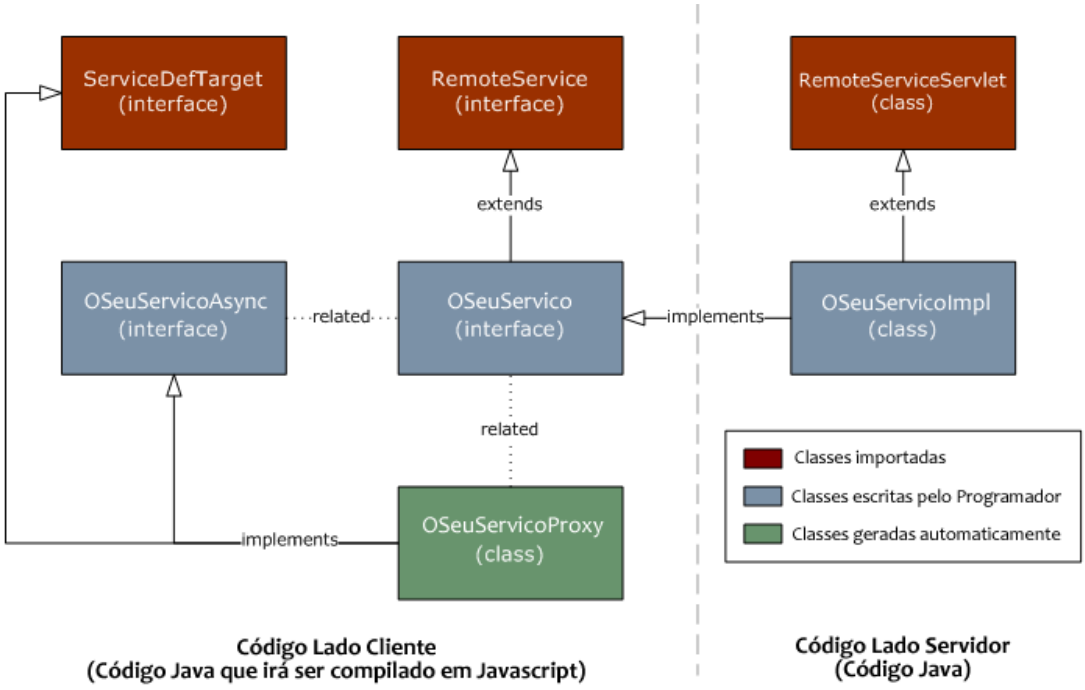


Figura 14 – RPC - Comunicação cliente/servidor no GWT (55)

Se bem utilizado, o *RPC* permite alocar toda a lógica *UI* para o cliente, deixando a lógica de negócio no lado servidor. Isto resulta numa maior performance, redução da largura de banda necessária e consequentemente uma melhor experiência de utilização ao utilizador final.

A Figura 35 demonstra o diagrama correspondente à implementação do mecanismo RPC no GWT.

A linha a tracejado simboliza a separação do lado cliente e lado servidor, demonstrando quais as classes que devem ser implementadas em cada uma das partes. Além disso, observam-se três tipos de cores distintas, que por sua vez distinguem as classes que pertencem à plataforma GWT (vermelho escuro), as classes a serem implementadas pelo programador (azul) e, as classes geradas automaticamente durante o processo de comunicação (verde). Deste modo, para uma correcta implementação do mecanismo RPC no GWT, o programador terá de, no lado cliente, implementar duas interfaces. A interface, no diagrama designada de "OSeuServico", irá declarar todos os serviços que o cliente irá requisitar ao servidor, ou seja, declara um conjunto de métodos que serão evocados pelo cliente ao servidor. Para o serviço (interface) definido, o programador terá ainda de declarar a versão assíncrona (segunda interface) do serviço, designado no diagrama de "OSeuServiçoAsync". Esta versão assíncrona do serviço irá permitir que quando os métodos sejam evocados ao lado servidor, a aplicação não fique "congelada" à espera da resposta deste último. Desta forma, ao fazer a requisição de um serviço, a aplicação cliente pode executar outras tarefas, sendo esta alertada quando receber a respectiva resposta do servidor. No lado servidor é então criada a classe, no diagrama designada "OSeuServicoImpl", que irá implementar a interface declarada no lado cliente. Assim sendo, os serviços são declarados no lado cliente e a sua respectiva implementação é feita no lado servidor. Como classe adicional, gerada automaticamente pelo GWT, temos a classe Proxy que funciona como um intermediário entre o cliente e o servidor. Atendendo aos pedidos do lado cliente, esta permite que o cliente evoque os serviços remotamente de forma transparente, como se de chamadas locais de tratasse.

Relembrando o funcionamento da plataforma GWT, todo o código declarado no lado cliente será convertido em Javascript enquanto o código do lado servidor será executado em Java bytecode.

### *GXT - Construindo Interfaces*

A biblioteca Ext-GWT (23), de acrónimo GXT, disponibiliza uma API que dá auxílio a construção de interfaces para aplicações GWT. Suporta a versão Java1.5 incluindo desde genéricos até enumerações, mecanismo RPC para comunicação cliente/servidor e é compatível com a maioria dos browsers: Internet Explorer 6+ (24) , Firefox 5+ (25), Safari 3+ (26), Opera9+ (27), Google Ghrome (28).

Esta biblioteca disponibiliza ainda um demo com vários exemplos de como as suas funcionalidades podem ser utilizadas, dando assim corpo às suas potencialidades.

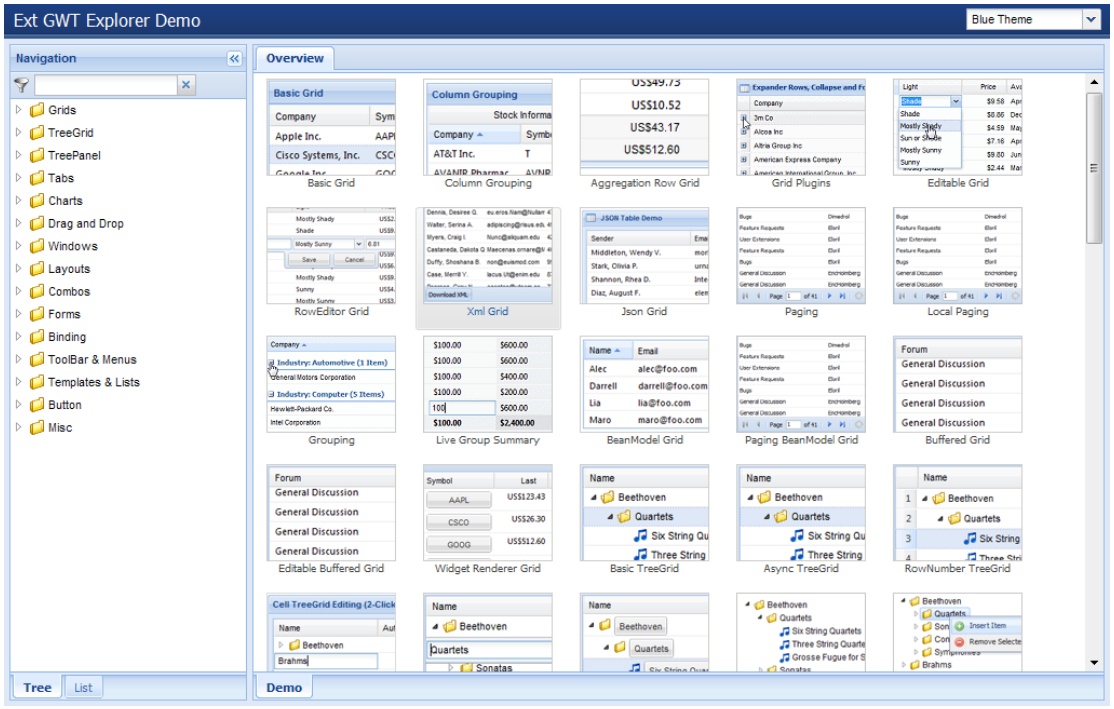


**Figura 15 – GXT Demo (52)**

O Explorer Demo do Ext-GWT pode ser consultado em http://www.extjs.com/products/gxt/.

### *Vista Arquitectural do GWT*

### *Componentes do GWT*

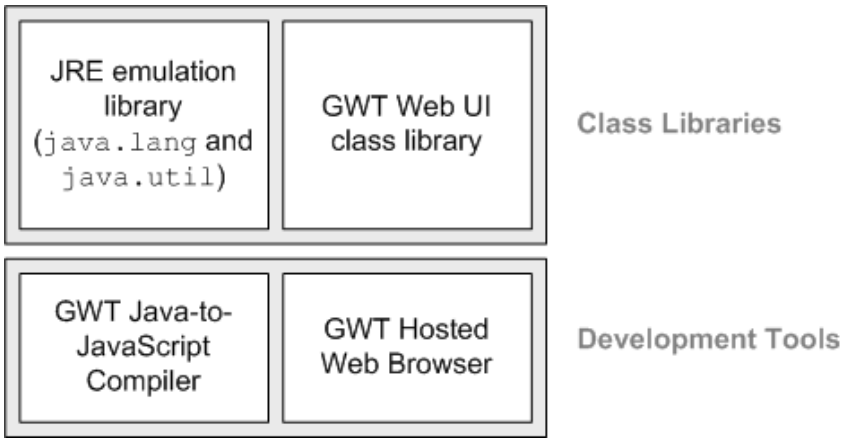


**Figura 16 – Representação lógica do GWT (29)**

### *Componentes arquitecturais do GWT*

A figura acima demonstra de uma forma simplificada a arquitectura da plataforma GWT. A camada superior é composta pelas bibliotecas oferecidas pela plataforma, sendo estas as bibliotecas Java que são reutilizadas no GWT para o desenvolvimento das aplicações e, a biblioteca GWT Web UI, esta última utilizada para o desenvolvimento das interfaces gráficas das aplicações GWT. Na camada inferior encontram-se as ferramentas do GWT que permitem a compilação e execução das aplicações, nomeadamente o "compilador Java para Javascript" que permite traduzir o código Java para Javascript para que este possa então ser executado directamente no browser e o Hosted Browser que permite que as aplicações sejam executadas em código Java, sendo esta funcionalidade muito útil durante o desenvolvimento das aplicações.

### *Resumo*

Com esta plataforma, o programador escreve as suas aplicações *Web* em *java*, tanto o lado cliente como o lado servidor, e o compilador do *GWT* compila o código *Java* do lado cliente para *Javascript* e *HTML*. Após isto basta colocar o código gerado em um servidor de forma que os utilizadores terão acesso à versão *Web* da aplicação, enquanto as classes *Java* do lado servidor deverão ser colocadas, por exemplo, no *Tomcat* ou outro *servlet container* à escolha.

## ASP.NET e Silverlight

Estas duas tecnologias desenvolvidas pela Microsoft oferecem imensos recursos para o desenvolvimento de aplicações Web, sendo que a segunda oferece uma nova experiência na Web explorando os benefícios do *RIAs* (*Rich Internet Applications*), permitindo novas e poderosas interfaces aumentando o dinamismo de interacção entre o utilizador e a aplicação Web.

### *ASP.NET*

#### *Como surgiu?*

O *ASP.NET* é uma plataforma desenvolvida e disponibilizada pela *Microsoft* de forma a permitir o desenvolvimento de aplicações *Web*. Esta plataforma foi construída sobre o *Common Language Runtime* (*CLR*), o "*virtual machine*" do *Microsoft.NET*, permitindo aos programadores escreverem código *ASP.NET* utilizando qualquer linguagem suportada pela plataforma *.NET*, como por exemplo *Visual Basic* e *C#* (c *sharp*).

O *ASP.NET 2.0* consiste na versão mais actual do *ASP.NET*, introduzindo um conjunto de novas funcionalidades, como por exemplo, traz maior suporte a integração de código no servidor com código *Javascript* do lado cliente, permitindo por exemplo digitar um valor numa caixa de texto e efectuar a consulta no banco de dados sem ter que enviar a página para o servidor. Esta é apenas um dos vários conceitos que o *ASP.NET 2.0* veio introduzir.

As novas funcionalidades no *ASP.NET 2.0* recaem sobre o controlo de funcionalidades e disponibilização de novos serviços e novas *API's*.

#### *Visual Web Developer*

O *Visual Web Developer* é a ferramenta disponibilizada pela *Microsoft* para o desenvolvimento de aplicações *Web* baseada na linguagem *ASP.NET*. De entre as características da ferramenta, destacam-se as seguintes:

- assistentes para facilitar a criação de aplicativos com suporte a função "*drag-and-drop"*;
- disponibilização do recurso *IntelliSense* (função que disponibiliza um conjunto de opções à medida que o desenvolvedor vai programando, aumentando de forma significativa a produtividade do mesmo);
- possui um editor gráfico para a criação e integração de bases de dados com a aplicação que está sendo desenvolvida;
- suporte a *HTML*, *CSS*, *RSS* e serviços *Web*.

Desta forma, o *Visual Web Developer* mostra-se uma ferramenta muito completa e prática para o desenvolvimento de aplicações *Web* com *ASP.NET*.

### *Arquitectura do ASP.NET (Visão geral)*

O *ASP.NET* utiliza o protocolo ***Internet Server Application Programming Interface*** (*ISAPI*), protocolo usado pelos computadores baseados no *Windows* para rodar uma aplicação dinâmica, para correr o Internet *Information Services* (*IIS* - servidor *Web* criado pela própria *Microsoft*) no servidor Windows 2000.

A plataforma *.NET* contém o compilador *CLR* (*Comon Language Runtime*) que compila e executa o código *ASP.NET* permitindo ainda a integração de outras linguagens suportadas pela plataforma *.NET*.

O *ADO.NET* permite o acesso remoto a dados armazenados numa base de dados.

De seguida, podemos ter uma visão geral da arquitectura do *ASP.NET*.

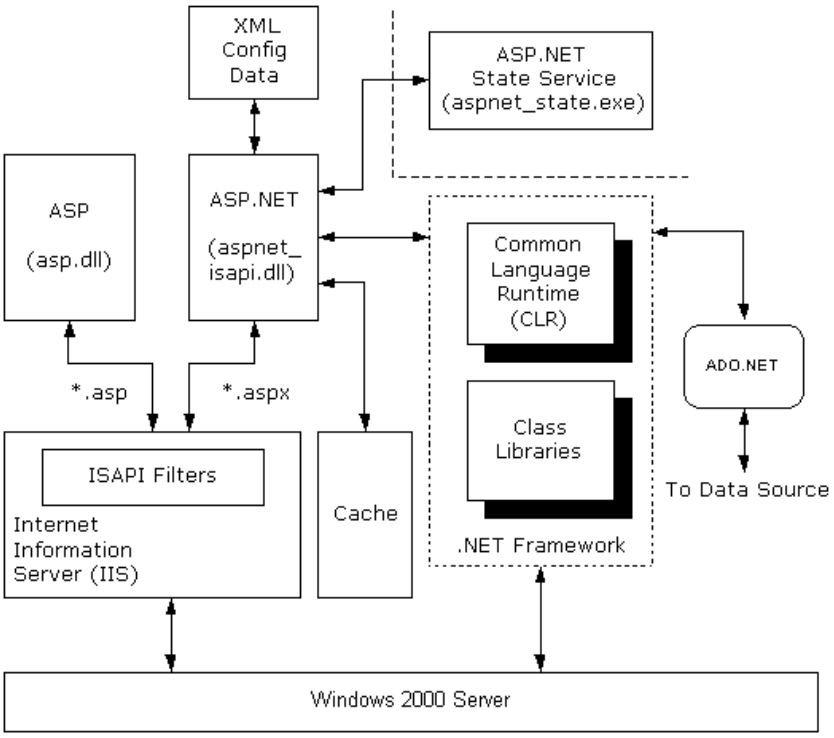


**Figura 17 - Visão geral da Arquitectura do ASP.NET (30)**

### *ASP.NET – Arquitectura Cliente/Servidor*

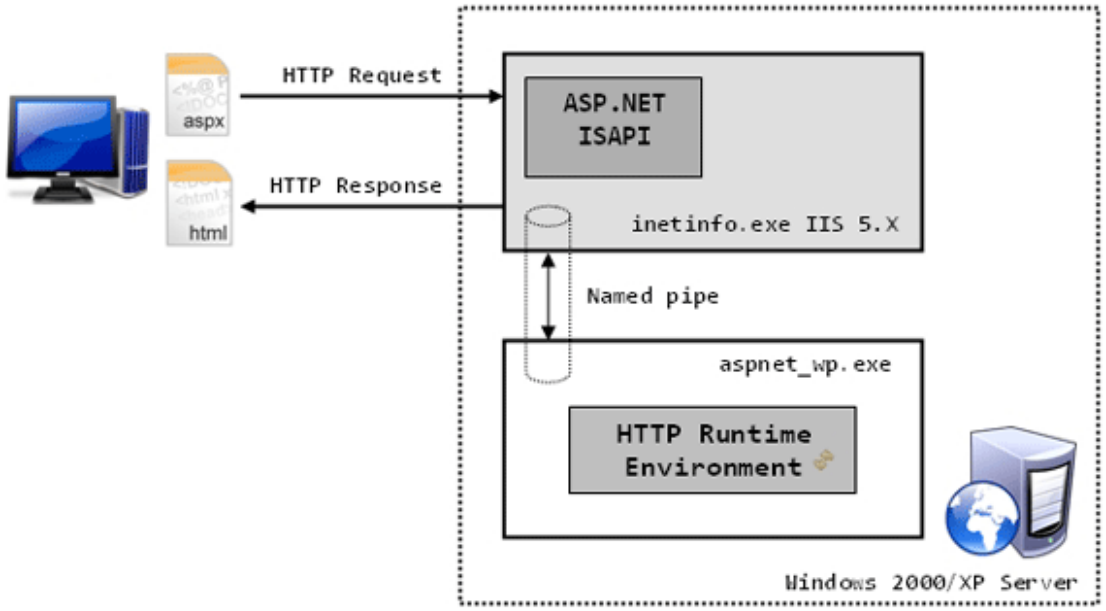


**Figura 18 - Processo comunicação cliente/servidor na plataforma ASP.NET (31)**

### *Silverlight*

> *"Silverlight is a cross-browser, cross-platform plug-in for delivering the next generation of media experiences and rich interactive applications (RIAs) for the Web."* (59)

#### *Como surgiu?*

O *Silverlight* inicialmente conhecido como *WPF/E*, é uma tecnologia desenvolvida e disponibilizada pela *Microsoft* e que surgiu da necessidade de criar uma tecnologia que fizesse concorrência ao *Flash*. Esta tecnologia criada pela *Macromedia* e actualmente muito presente na *Web*, veio revolucionar o modo de apresentação da informação na *Web*, permitindo conteúdo mais dinâmico e apelativo, criando-se aplicações mais atractivas e mais apelativas ao utilizador.

Do mesmo modo, o *Silverlight* surgiu com o objectivo de proporcionar ao desenvolvedores uma plataforma que permitisse a criação de aplicações ricas para a internet (*RIA's – Rich Internet Applications*) de forma a garantir uma boa experiência de utilização ao utilizador final, sendo que este é um factor que pesa cada vez mais no índice de sucesso das aplicações actualmente desenvolvidas, seja para o ambiente *Web* ou ambiente "*Desktop*".

Disponibilizado em modo *plugin*, o *Silverlight* pode ser facilmente instalado no computador para desta forma poder usufruir-se das aplicações desenvolvidas sobre esta plataforma. Esta é

uma plataforma que engloba os conceitos de "*cross-platform*" (multi-plataforma), funcionando assim em diferentes sistemas operativos, e "*cross-browser*" (multi-browser), podendo ser executado nos mais variados navegadores Web.

### *Quando usar o Silverlight*

Esta tecnologia deve ser utilizada para projectos que exijam um nível elevado de interacção com o utilizador sobre qualquer browser e, que envolvam também um elevado nível de processamento, ou seja, comunicação e transacção de dados entre cliente/servidor, pois uma vez que o *Silverligth* está inserido em um ambiente de desenvolvimento integrado torna-se muito mais fácil para o desenvolvedor implementar este tipo de tarefas.

### *Silverlight - Ferramentas necessárias*

Para o desenvolvimento completo de aplicações nesta plataforma são necessárias duas ferramentas: o *Expression Studio* para a criação de gráficos vectoriais e o *Visual Studio 2008 Standard Edition* para o desenvolvimento do lado transaccional, ou seja, regras de negócio e transacção de dados.

O grande senão é o facto do *Visual Studio 2008 Standard Edition* não ser gratuito [1] (é disponibilizado apenas um *trial* que pode ser utilizado para testar o Silverlight), no entanto, já é possível instalar o Silverlight no *Visual Web Developer 2008 Express Edition*, gratuitamente disponibilizado pela *Microsoft*. Desta forma, podemos integrar as tecnologias *Silverlight* e *ASP.NET* para a criação de aplicações *Web*.

### *Silverlight 2.0*

O *Silverlight* 2.0 é a última versão disponibilizada pela *Microsoft*. Nesta versão, foram aproveitadas todos os recursos do *.NET* para desenvolver aplicações *Silverlight*, pelo que, integrando a plataforma *Silverlight* na plataforma *.NET* pode-se então usufruir de todas as linguagens suportadas pela última, como *C#*, *VB* e mesmo *ASP.NET*, e dispomos também de um conjunto alargado de controlos que vêm facilitar em muito a tarefa do desenvolvedor.

Deste modo, o *Silverlight* oferece um modelo de desenvolvimento flexível e consistente podendo até mesmo ser integrado em aplicações *Web* já existentes.

[1] A Microsoft disponibiliza o seu software a estudantes do secundário e universitários. Mais informação em Microsoft DreamSpark.

### *Silverlight - XAML*

A grande diferença no *Silverlight* está no facto de que as *interfaces* são criadas em ficheiros *XAML* (*EXtensible Application Markup Language* - lê-se "*Zammel*") com grande suporte a *SEO* (*Search Engine Optimization*). O *XAML* é uma linguagem declarativa de dialecto *XML*, utilizado então no *Silverlight* para definir as *interfaces* da aplicação, pelo que estas consistem simplesmente em um ficheiro com estrutura *XML*.

O *XAML* separa a parte gráfica da parte lógica da aplicação através dos arquivos *code-behind* (arquivos *.cs* se estivermos utilizando *C#*), facilitando e organizando o trabalho do desenvolvedor, permitindo mesmo uma separação de responsabilidades para o caso da aplicação ser desenvolvida por pessoas com diferentes funções - programador/*designer*.

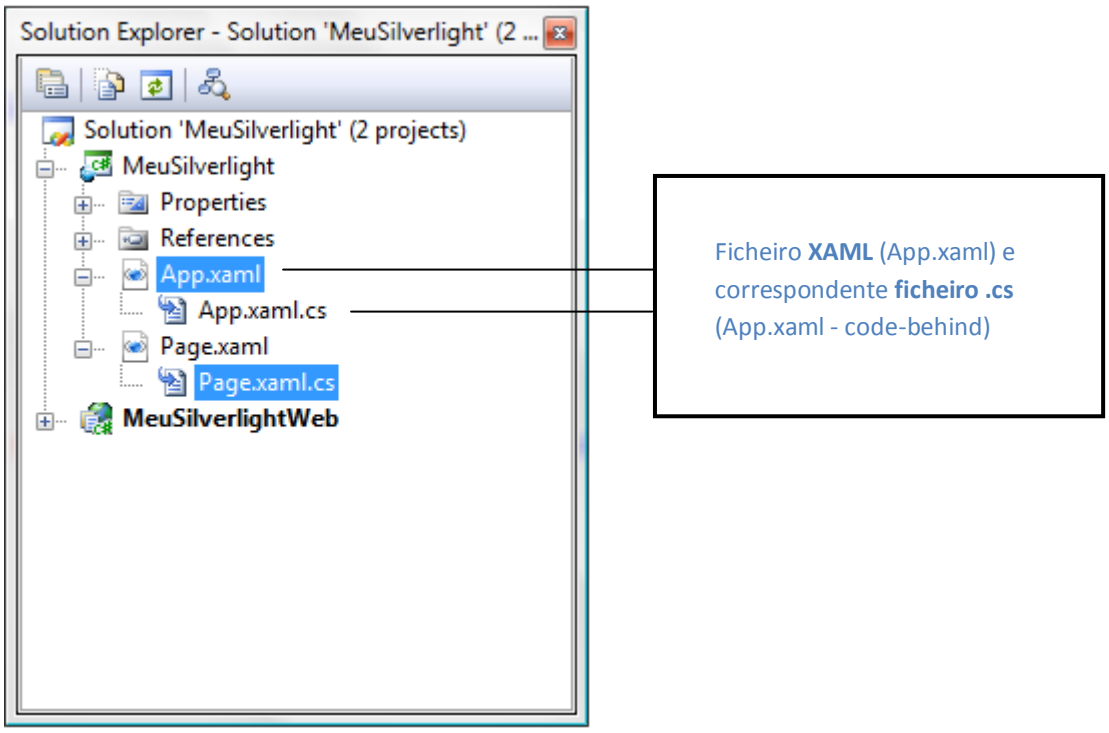


**Figura 19 - Estrutura de um projecto *Silverlight – XAML* no *Silverlight***

### *Arquitectura do Silverlight 2.0*

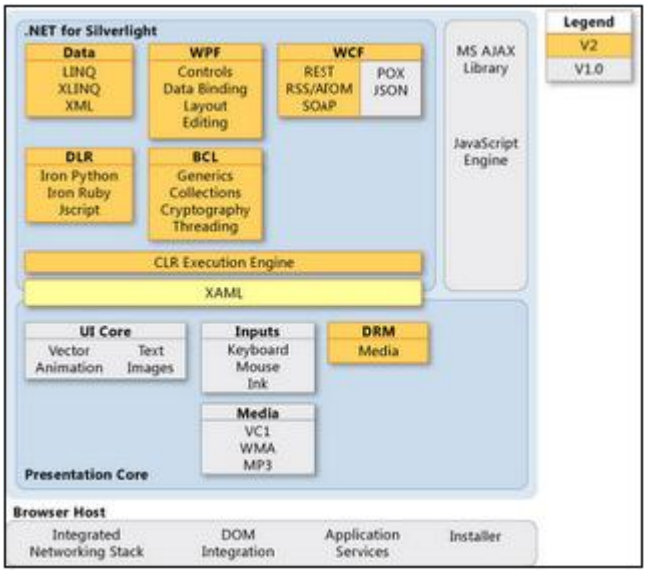


**Figura 20 - Arquitectura *Silverlight* v1.0 vs Arquitectura *Silverlight* v2.0**

A arquitectura acima descrita demonstra a grande evolução da versão 1.0 para a versão 2.0 do *Silverlight*.

A nova versão da *framework .NET* do *Silverlight 2.0* traz consigo um novo subconjunto do modelo de programação *UI* do *WPF*, incluindo suporte à formas, documentos, conteúdo multimédia e animação de objectos do *WPF* e ainda mais de 30 controlos *UI*, como por exemplo *TextBox*, *CheckBox*, *ScrollViewer*, etc… A nova biblioteca *BCL* (*Base-Class Library*) inclui os conceitos de colecções, reflexão, expressões regulares, entre outros, dando também suporte ao conhecido *LINQ*.

O *Silverlight 2.0* inclui o designado *DLR* (*Dynamic Language Runtime*) que permite a compilação e execução dinâmica de linguagens dinâmicas, incluindo o Iron Phyton e o *Iron Ruby*. Outra das novidades que o *Silverlight 2.0* traz consigo é a tecnologia *Deep Zoom*, que permite aos utilizadores fazerem "*zoom in*" e "*zoom out*" de uma imagem, com transições suaves, utilizando para tal o botão do meio do rato (*mouse wheel*).

### *Resumo*

O *Silverlight* é realmente uma ferramenta muito poderosa e perfeitamente adequada para o desenvolvimento de aplicações *Web* que exijam grande interacção com o utilizador, interacção essa sobre editores gráficos, como mapas, editores de diagramas, etc. Integrando o *Silverlight* com *ASP.NET* é possível aproveitar todos os recursos que estes dispõem de forma a facilitar o desenvolvimento de complexas aplicações *Web*.

## Adobe *Flex*

### *Como surgiu?*

O *Adobe Flex* foi uma tecnologia lançada para permitir aos *designers* e desenvolvedores criarem aplicações sobre a plataforma *WEB2.0* baseadas na plataforma *Macromedia Flash*. Embora o *Flex* seja baseado na plataforma *Macromedia Flash*, estas duas ferramentas são um pouco distintas, na medida em que a primeira é mais orientada aos desenvolvedores aproximando-se mais ao desenvolvimento de aplicações, enquanto a segunda é uma ferramenta mais orientada ao desenvolvimento de gráficos vectoriais e deste modo mais orientada aos *designers*. No entanto, a integração destas duas tecnologias permite a criação de aplicações ricas (*RIA's*) para a plataforma Web. Tal como o *Flash*, o *Flex* produz ficheiros *.swf* pelo que apenas é necessário instalar o *plugin* do Flash para que se possa correr as aplicações *Flex*, mediante um *browser*.

### *Linguagens do Flex – ActionScript e MXML*

Oferecendo cerca de 100 componentes visuais, o Flex foca-se na utilização de *ActionScript* e *MXML* para a criação das aplicações.

### *ActionScript 3.0*

Esta é uma linguagem de programação orientada aos objectos também utilizada no *Flash* de forma a adicionar interactividade a aplicações dinâmicas. O *ActionScript* é uma linguagem *ECMAScript* (linguagem baseada em scripts) e é utilizada no *Flex* para criar a lógica do lado cliente e a consequente interacção com o utilizador.

### *MXML (Multimedia Extensible Language Markup)*

Esta linguagem é baseada na linguagem *XML* respeitando assim a estrutura desta última e é utilizada no *FLEX* para a descrição das componentes gráficas. Uma vez que a ferramenta oferece uma componente gráfica para a criação de interfaces, o desenvolvedor não precisa de possuir conhecimento profundo desta linguagem para o desenvolvimento de aplicações *Flex*.

### *Recursos do Flex*

#### *Flex Software Development Kit (SDK)*

O *Flex SDK* é *open source* e portanto de livre utilização. Este é um *kit* que inclui a *framework* do *Flex* e as bibliotecas e compilador do respectivo e que permite portanto a criação e desenvolvimento de aplicações *Web*.

#### *Adobe Flex Builder 3*

A *Adobe* oferece uma ferramenta de desenvolvimento designada *Flex Builder*, sendo que a última versão lançada é o *Adobe Flex Builder 3*. Esta é uma ferramenta baseada no *Eclipse* que por sua vez oferece ferramentas para *debugging* e que contém a *framework* completa do *Flex*, incluindo todos os recursos necessários para o desenvolvimento de aplicações *Flex*. Esta não é uma ferramenta de livre utilização, sendo disponibilizado pela *Adobe* uma demonstração de 60 dias para utilização. No entanto, a ferramenta pode adquirida sem custos desde que seja utilizada para fins educacionais (http://www.adobe.com/devnet/edu/).

### *Flex - Comunicação Cliente/Servidor*

O *Flex* possui um conjunto de técnicas para a comunicação entre cliente/servidor e, como tal, podem ser utilizados para a transferência de dados entre os respectivos. Tais técnicas traduzem-se em:

- *Web Services*;
- *Http Services*;
- *Remote Objects (RPC)*.

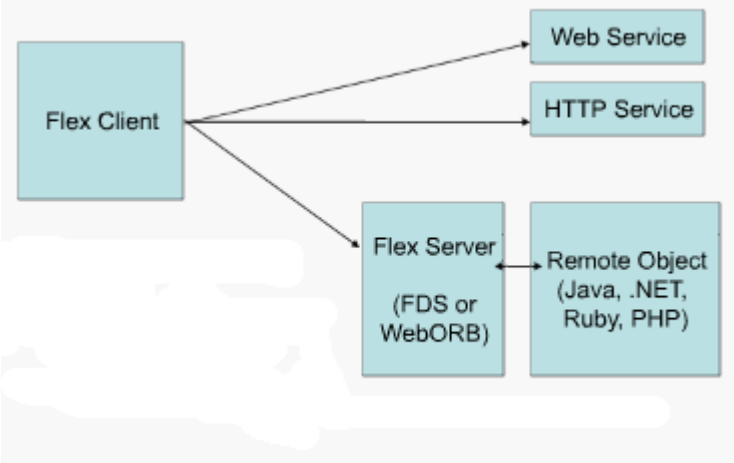


**Figura 21 - *Flex* – Comunicação Cliente/Servidor (32)**

### *Arquitectura do Flex*

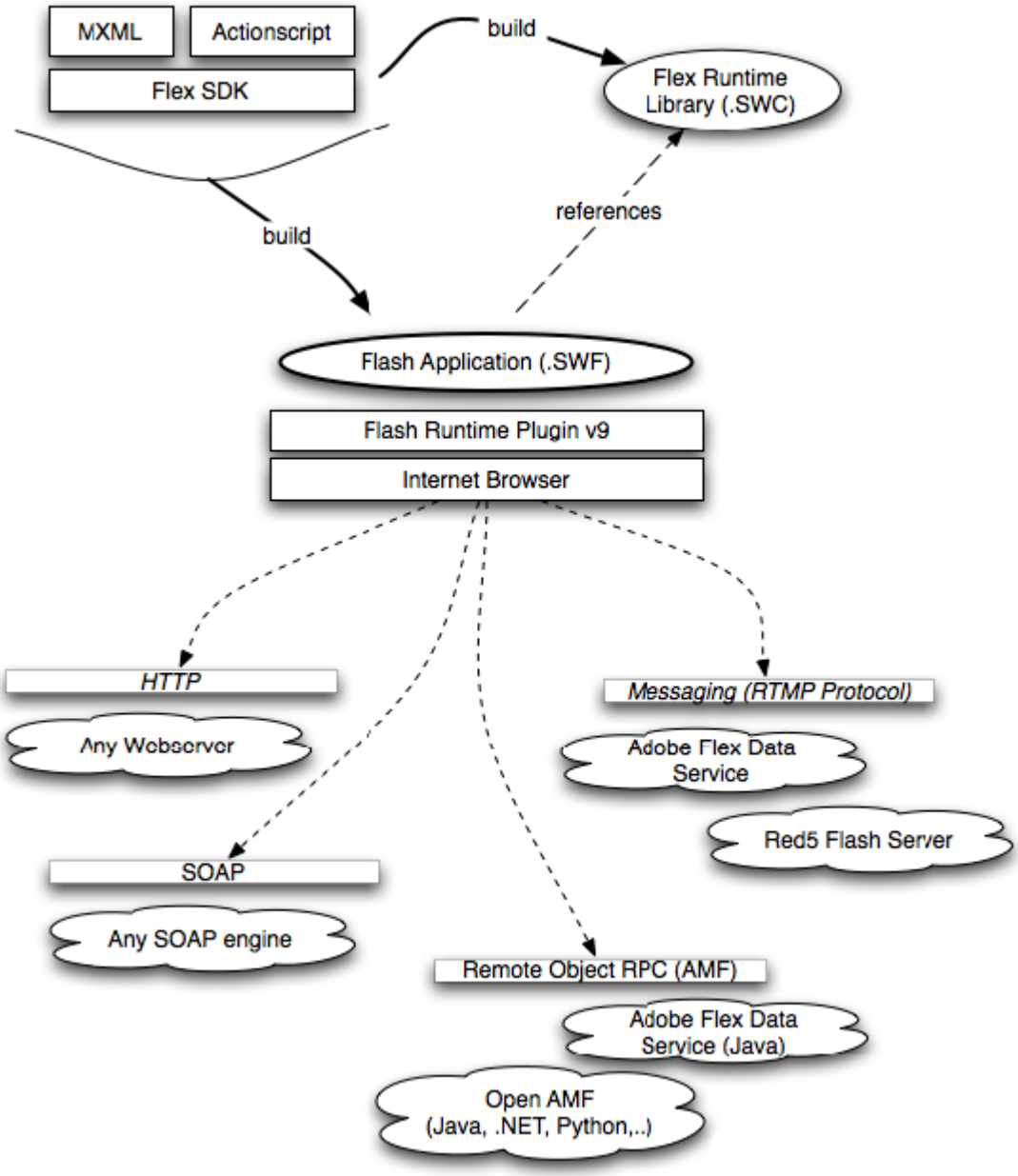


**Figura 22 - *Flex* – Visão Geral da Arquitectura (33)**

A observação da arquitectura acima descrita permite ter uma visão geral de como este funciona. De uma forma geral e através de uma perspectiva *Top-Down*, verificamos que o *Flex* consiste em um conjunto de ficheiros *MXML* e *ActionScript* que são compilados de forma a gerar os ficheiro *SWF's* que por sua vez consistem na aplicação *Flash* e que, via um *plugin* do *Flash*, é executado sobre qualquer *browser*. A comunicação com o servidor pode ser feito utilizando qualquer uma das técnicas de comunicação já aqui referidas.

### 2.4.3 Avaliação as Plataformas

Depois de uma visão geral sobre as plataformas a avaliar e proposta a framework de avaliação, de seguida serão avaliadas cada uma das plataformas, avaliando cada um dos atributos considerados. Para o efeito e como já foi referido anteriormente, é implementado para cada uma delas um caso prático que consiste num editor de formas, sendo esta uma amostra muito simplificada daquilo que será a parte essencial da aplicação desejada, dado que esta irá consistir em um editor de modelos. Relembrando, o caso prático irá permitir criar dois rectângulos e interliga-los por meio de uma linha, simulando desta forma um simples editor.

#### *2.4.3.1 Google Web Toolkit - GWT*

A plataforma *GWT* é uma plataforma *open-source* criada e disponibilizada pela *Google*. Esta plataforma é suportada por uma comunidade activa, que ao longo do tempo tem trabalhado sempre na evolução da ferramenta, disponibilizando um conjunto de vários releases (sendo o último GWT 1.5), a medida que a plataforma vai sendo actualizada. Esta possui muita e boa documentação, suportada não só pela própria *Google* como também por pessoas que, por gostarem deste projecto, decidiram também participar na evolução e optimização da mesma.

A principal linguagem do *GWT* é o *Java*, sendo que esta é utilizada para o desenvolvimento completo das aplicações, podendo também se utilizar uma mistura de *Javascript* e *Java* para casos particulares. A utilização de *Java* para o desenvolvimento de aplicações é uma mais-valia na medida em que esta é uma linguagem madura, robusta, expressiva e muito utilizada, pelo que também torna-se fácil encontrar documentação sobre a linguagem. Como já foi referido anteriormente, uma das particularidades mais importantes desta plataforma é o facto de que, dado que a linguagem de desenvolvimento do *GWT* é o *Java*, este permite que um projecto seja importado para um *IDE Java* da preferência do desenvolvedor, como por exemplo o *Eclipse* ou o *NetBeans*. Desta forma, e tratando-se de um desenvolvedor *Java*, este pode criar os projectos *GWT* no seu *IDE* favorito sem a necessidade de aprender novas ferramentas.

Existem várias técnicas de comunicação entre cliente/servidor, mas o mais importante e igualmente utilizado é o *Remote Procedure Call* (*RPC*) que permite de uma forma simplificada a partilha de objectos entre o cliente e o servidor sobre a rede. O desenvolvedor apenas tem de implementar os serviços que necessita para a comunicação e a plataforma gera as restantes classes necessárias, facilitando deste modo a tarefa do desenvolvedor abstraindo-o de detalhes de implementação.

Outra das vantagens do *GWT* é o facto de poder ser executada nos vários Sistemas Operativos da *Microsoft*, *Mac* e *Linux*. Este facto mostra o cuidado e a importância que a *Google* teve em conta na flexibilidade da plataforma, mostrando que esta é uma plataforma consistente e segura independentemente do ambiente em que é executada. Uma vez que o *GWT* pode ser utilizado em qualquer *IDE Java*, é possível utilizar o *JUnit* para testar a integração das aplicações implementadas, pelo que este tipo de tarefas é sempre facilitado pelas ferramentas em questão.

Quanto ao suporte de componentes gráficos, o *GWT* disponibiliza uma extensão designada "*gwt-incubator*", que disponibiliza as funcionalidades necessárias para a criação de gráficos.

## GWT – Implementando o caso prático

Para a implementação do caso teste foi utilizado o *Eclipse*.

Criada a estrutura do projecto, basta importar a biblioteca "*gwt-incubator*" e, a partir da *API* fornecida por ela começar a implementação do editor.

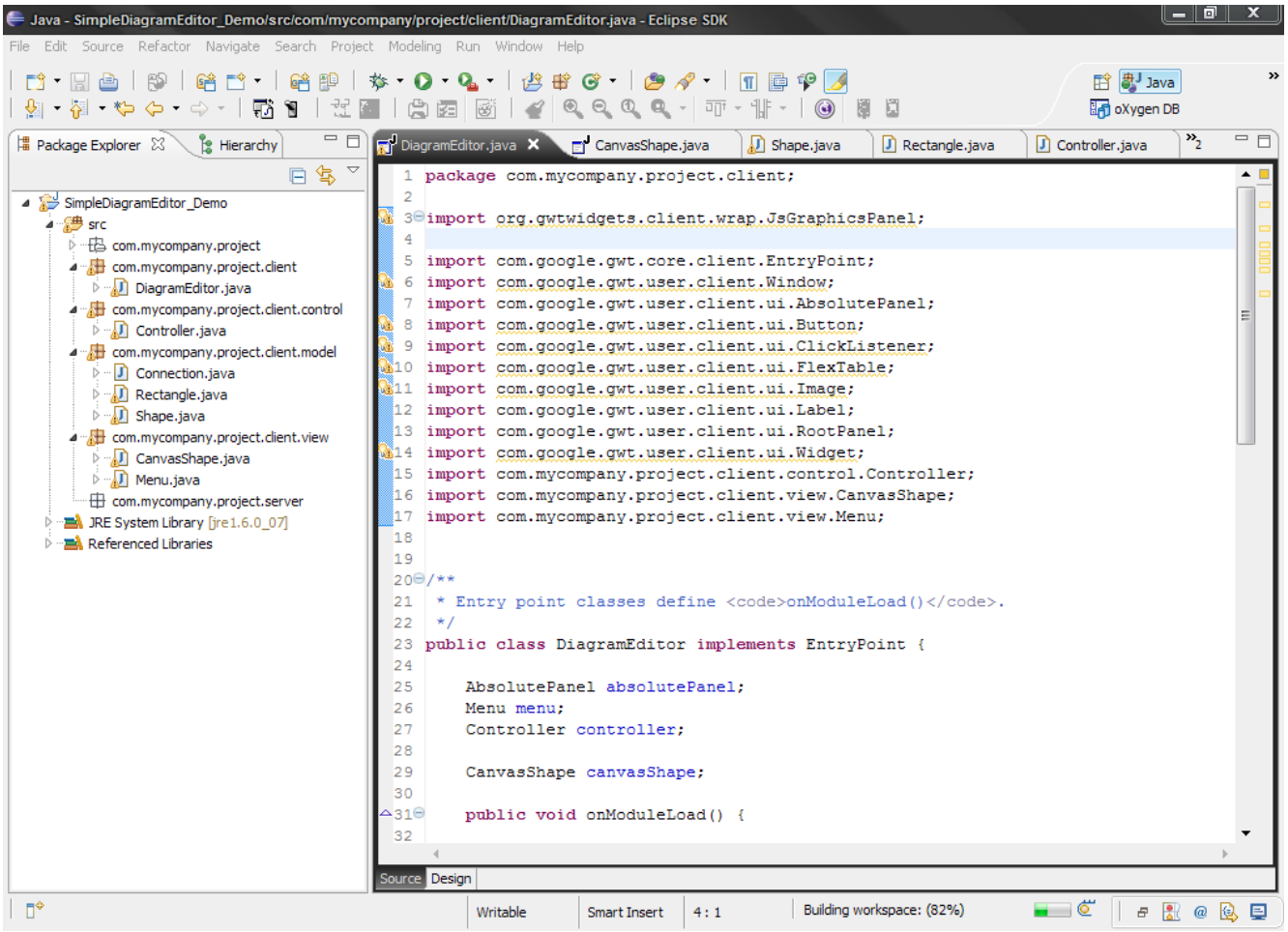


**Figura 23 - Estrutura do projecto *GWT* no *Eclipse***

## Google Web Toolkit – Testando o Editor de Formas

Depois de implementado, o resultado final foi o seguinte:

Em "*Hosted Mode*", utilizando o *browser* interno do *GWT*:

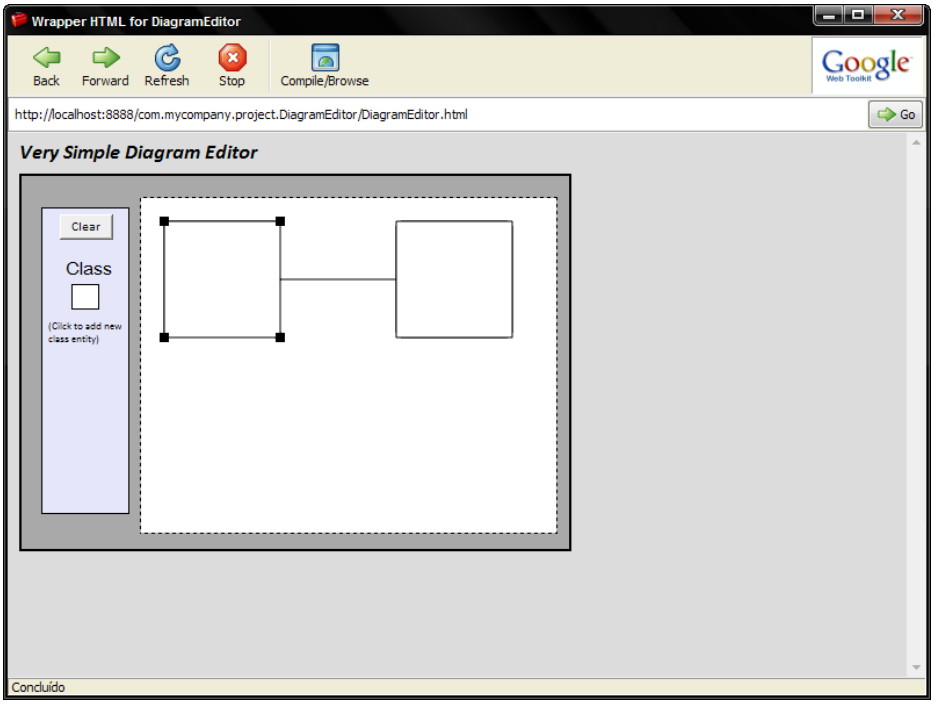


Figura 24 - Editor de Formas em Hosted Mode

Em "*Web Mode*":

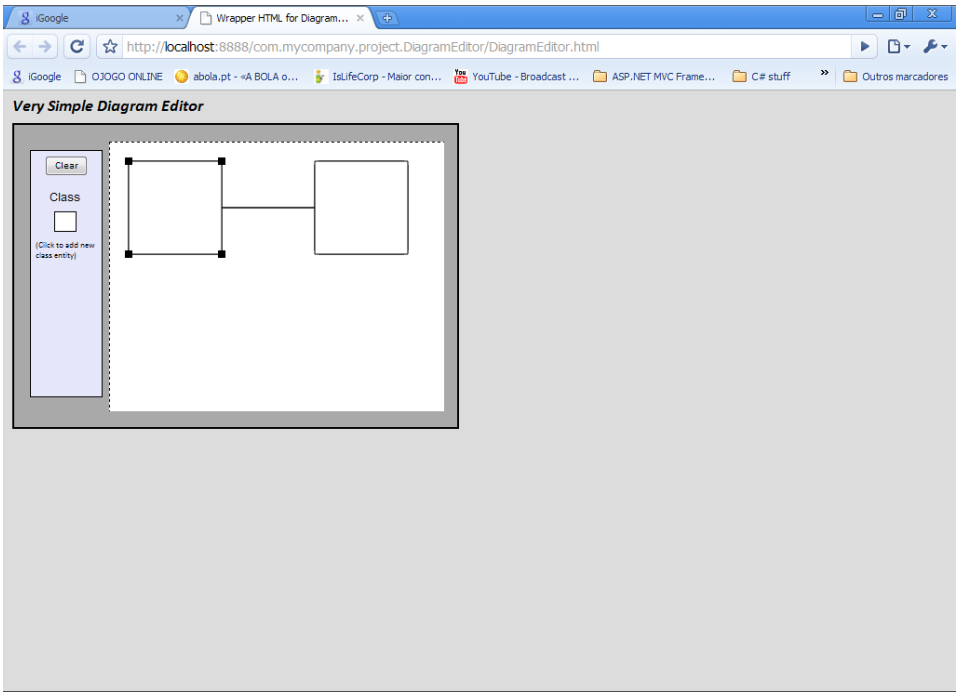


Figura 25 - Editor de Formas em Web Mode

### Google Web Toolkit – Conclusões sobre a plataforma

O *GWT* mostra-se assim uma plataforma muito interessante e muito útil para o desenvolvimento deste tipo de aplicações, fornecendo uma *API* para auxiliar o desenvolvimento de diagramas. A plataforma pelas potencialidades que já apresenta e pela sua capacidade de evolução, torna-a uma ferramenta muito atractiva para quem pretende desenvolver este tipo de aplicações para *Web,* principalmente para desenvolvedores JAVA.

#### *2.4.3.2 Silverlight*

O *Silverlight* criado e disponibilizado pela Microsoft pode ser integrada no *Visual Web Developer 2008 Express Edition*, ferramenta esta disponibilizada gratuitamente também pela *Microsoft*. Uma vez que o *Silverlight* é integrando no *.NET* podem ser utilizadas todas as linguagens suportadas por esta última para o desenvolvimento de aplicações *Web*, como por exemplo *C#*, *VB* e mesmo *ASP.NET*. Esta é uma excelente notícia para os desenvolvedores familiarizados com o *.NET*, no entanto para os restantes dos desenvolvedores, a tarefa já não é tão simples dado que têm de assimilar um conjunto de novas tecnologias. No entanto, a utilização integrada das várias tecnologias no desenvolvimento de aplicações tornam o *Silverlight* uma ferramenta poderosa.

Sendo a plataforma suportada pela *Microsoft*, esta possui uma grande e consistente comunidade a trabalhar na sua constante evolução. Possui vasta e boa documentação, pelo que se torna fácil encontrar material de estudo, tanto para um utilizador inicial como para um utilizador avançado.

Para a comunicação cliente/servidor pode-se utilizar diferentes técnicas, tais como:

- *ASP.NET Web Services (asmx)*
- *Windows Communication Foundation (WCF) Services (WCF Duplex Services)*
- *Sockets*

Sendo esta uma ferramenta *Microsoft*, só pode ser utilizada em máquinas *Microsoft* limitando assim a sua *portabilidade*. Isto claro para o desenvolvimento de aplicações *Web*, pois as aplicações *Web* baseadas no *Silverlight*, depois de concluídas e disponibilizadas na *Web*, podem ser utilizadas em qualquer computador através de um browser, bastando para isso instalar o *plugin* do *Silverlight* necessário para correr as aplicações *Silverlight*.

Para testar a integridade das aplicações desenvolvidas em *Silverlight*, pode ser utilizado o *NUnit*, um framework de automatização de testes semelhante ao *JUnit* no *Java* e que pode ser utilizado no *.NET* facilitando e promovendo a qualidade do trabalho realizado pelo desenvolvedor.

Para a elaboração de componentes gráficos, o *Silverlight* disponibiliza uma biblioteca designada *Shapes* que permite a criação das mais variadas formas, incluindo *rectângulos*, *círculos*, *elipses*, com o suporte de eventos sobre as mesmas.

## Silverlight - Implementando o caso prático

Depois de instalado o *Silverlight*, basta apenas executar o *Visual Web Developer* e, na criação de um novo projecto escolher a opção "*Silverlight Application*".

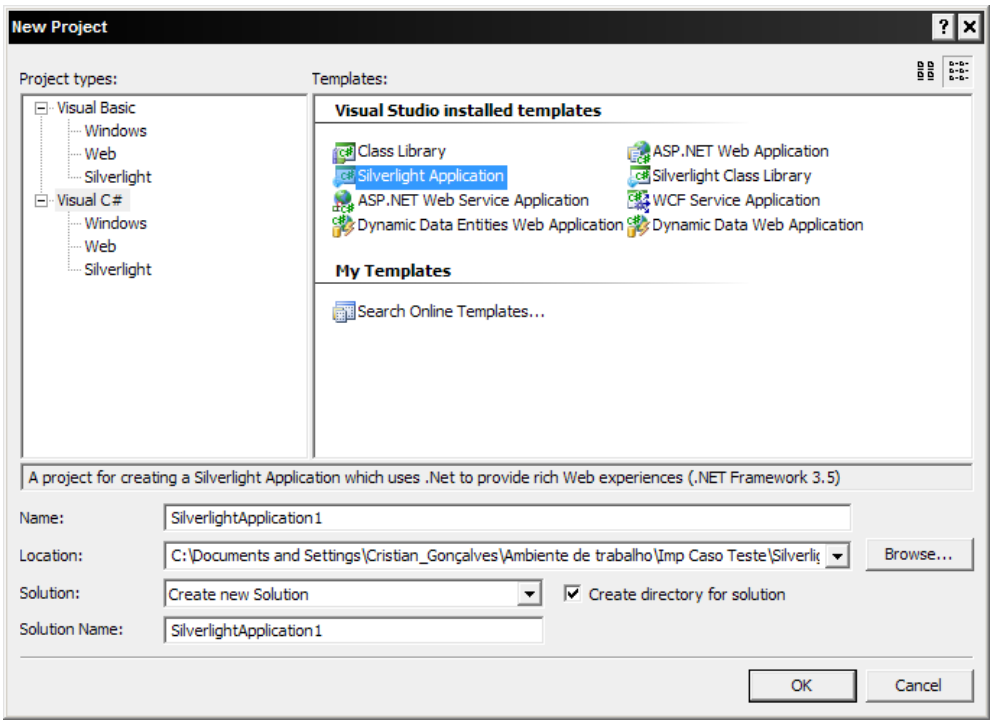


**Figura 26 - Criando uma aplicação *Silverlight***

Depois de criado e desenvolvido o projecto, o resultado é o seguinte:

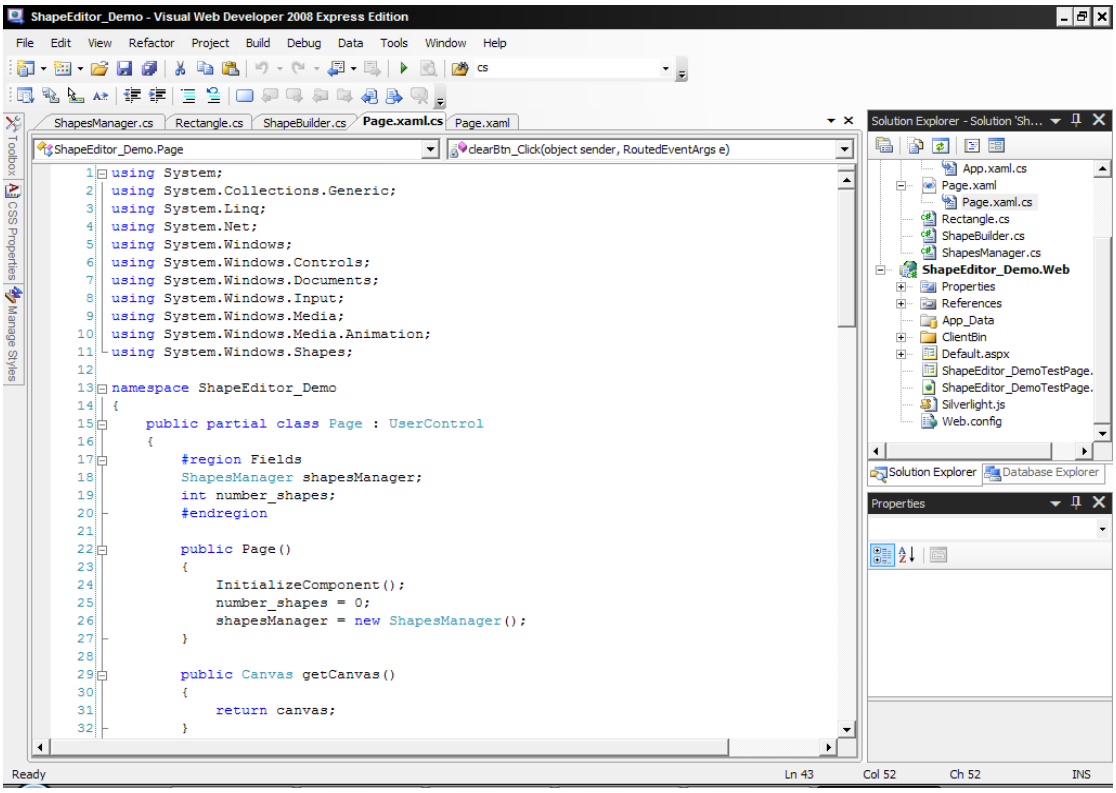


**Figura 27 - Estrutura de uma aplicação *Silverlight***

Depois de implementado, o projecto é testado sendo executado em um *browser* à escolha do desenvolvedor. Neste caso foi testado no *Google Chrome,* sendo o resultado o seguinte:

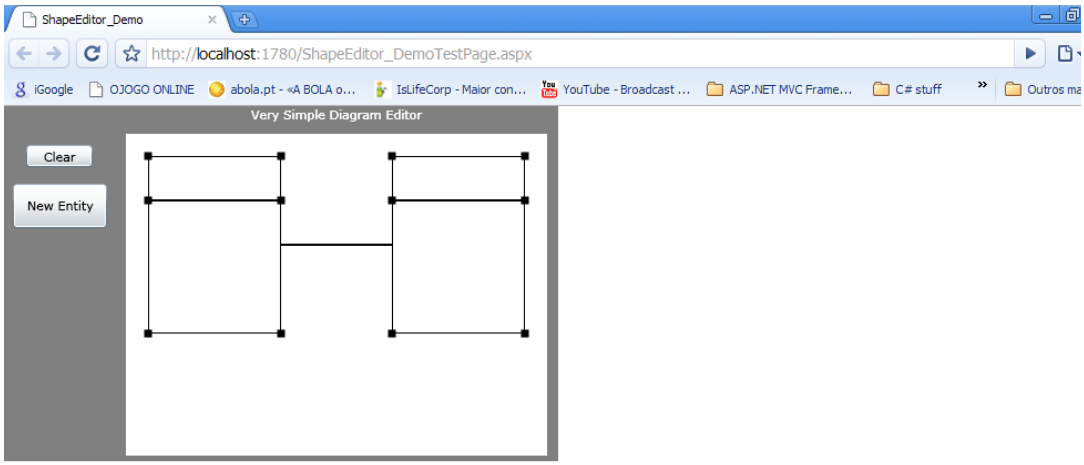


**Figura 28 - *Silverlight* - Testando Editor de Formas**

### Silverlight – Conclusões sobre a plataforma

O *Silverlight* mostra-se também um forte candidato para o desenvolvimento de aplicações Web, na medida em que estando inserido na plataforma *.NET* dispõe de todos os recursos disponibilizados por esta sendo indicado para aplicações que exijam um alto nível de interacção com o utilizador.

#### *2.4.3.3 Flex*

Sendo a *Adobe* a proprietária da ferramenta e sendo esta um dos gigantes no mundo das *Web* conclui-se desde já sobre a dimensão e a qualidade da entidade que dá suporte ao *Flex*. No entanto, e apesar da boa qualidade da documentação apresentada, por vezes torna-se difícil encontrar documentação específica para determinadas situações o que traz algumas dificuldades na aprendizagem da tecnologia.

A última versão do *Flex* já utiliza *ActionScript 3* que por sua vez já suporta o paradigma orientado à objectos, o que o torna uma linguagem madura e mais perceptível e de fácil aprendizagem para quem está familiarizado com o paradigma da programação orientada à objectos.

Outra das vantagens do *Flex*, mais concretamente da última versão disponibilizada (*Flex Builder 3*), é o facto de utilizar como editor de desenvolvimento o *Eclipse*, reaproveitando-se assim muitos dos seus recursos e aproveitando o facto de muitos desenvolvedores (especialmente desenvolvedores *Java*) já serem familiarizados com a ferramenta.

O *Flex Builder 3* é de fácil instalação, sendo que este é multi-plataforma e portanto pode ser instalado em *Windows*, *Mac* ou ainda *Linux*.

Quanto ao suporte à criação dinâmica de componentes gráficos, o *Flex* dispõe de um conjunto de bibliotecas que permitem a criação das mais variadas formas geométricas. Para tal, basta apenas importar as bibliotecas para se poder então utilizar as funções por elas disponibilizadas.

Esta plataforma, apesar de ser proprietária pelo que não pode ser utilizada livremente para fins empresariais, sendo um produto de uma das maiores empresas de software, mostra-se uma excelente plataforma para a criação de aplicações sobre a plataforma *Web*, disponibilizando um conjunto de recursos essenciais para o desenvolvimento de aplicações dinâmicas, complexas e que exigem elevados níveis de interacção com o utilizador.

## Flex – Implementação do caso Prático

Depois de criado o projecto no *eclipse*, podemos começar a desenvolver a aplicação. Tal como já foi dito anteriormente, o *Flex Builder* disponibiliza uma componente gráfica para o desenvolvimento das interfaces da aplicação, tal como mostra a figura seguinte:

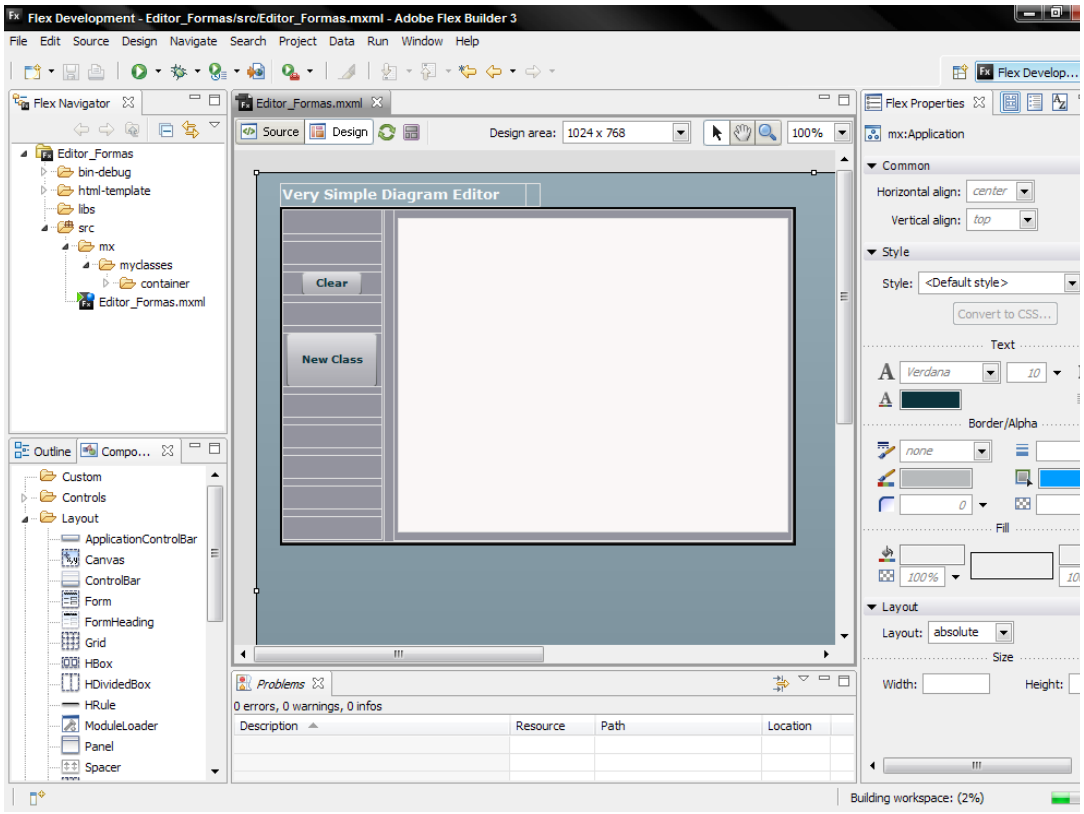


**Figura 29 - Desenvolvendo interfaces no Flex**

Depois de implementado e executado o caso teste o resultado foi o seguinte:

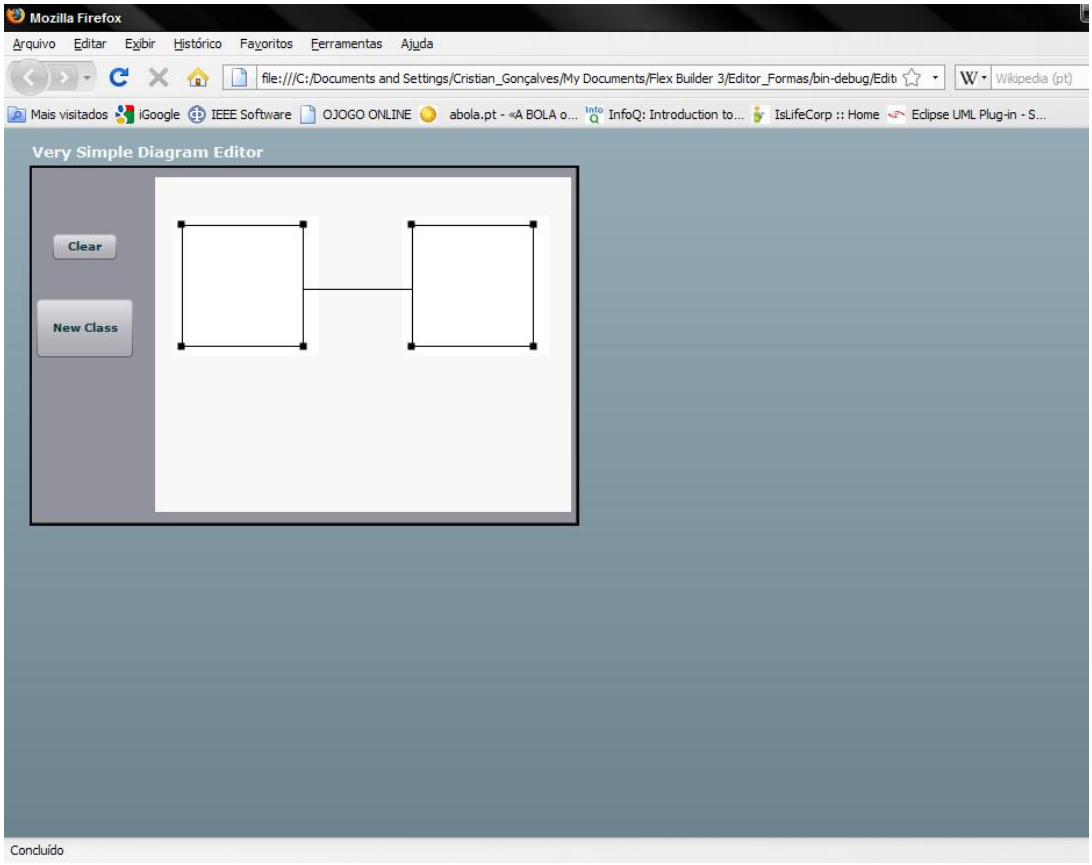


**Figura 30 - Flex – Testando o caso Prático**

## *Flex* - Conclusões sobre a Plataforma

O *Flex Builder 3* traz consigo um conjunto de bibliotecas para o suporte de componentes gráficos. Sendo uma ferramenta desenvolvida pela Adobe é com certeza uma plataforma muito completa e com grande suporte ao desenvolvimento de aplicações Web.

### 2.4.4 Resultados finais da Avaliação

Tabela 2 - Framework de Avaliação - Resultados

| | Google Web Toolkit | Silverlight | Flex |
|---|---|---|---|
| ***Dimensão da Comunidade de suporte*** | Grande | Grande | Grande |
| ***Quantidade Documentação disponível*** | Muita | Muita | Razoável |
| ***Qualidade Documentação disponível*** | Boa | Muito Boa | Boa |
| ***Linguagens suportadas*** | Java, JavaScript | Linguagens suportadas pelo .NET | MXML, ActionScript 3.0 |
| ***Comunicação cliente/servidor*** | RPC | ASP.NET Web Services<br>Windows Communication Foundation (WCF) Services (WCF Duplex Services)<br>Sockets | Web Services, HTTP Request, RPC |
| ***Portabilidade*** | 4 | 2 | 4 |
| ***Usabilidade*** | 4 | 4 | 4 |
| ***Facilidade de aprendizagem*** | 4 | 3 | 4 |
| ***Facilidade de instalação*** | 4 | 4 | 4 |
| ***Licenças*** | Software Livre | Software Proprietário | Software Proprietário |
| ***Suporte a Testes*** | 5 | 5 | 4 |
| ***API's para Componentes Gráficos*** | "gwt-incubator" | "System.Windows.Shapes" | "flash.display.Shape" |

### 2.4.5 Apreciação dos Resultados – Escolha da Plataforma

Avaliada cada uma das plataformas e apresentados os resultados finais da *framework* de avaliação é altura de fazer a escolha da plataforma que irá servir de suporte ao desenvolvimento da aplicação *Web* desejada. Note-se que algumas das dimensões avaliadas dependem dos critérios do desenvolvedor pelo que não pode existir uma concordância unânime sobre qual a melhor plataforma.

Os valores para cada uma das dimensões ficaram na maioria dos casos distribuídos, no entanto, existem alguns pontos importantes em que as plataformas são distintas. Pela observação dos resultados e tendo em conta a implementação do caso prático para cada uma das plataformas, foi decidido que o *Google Web Toolkit* é a plataforma a utilizar no desenvolvimento da aplicação *Web* desejada.

De seguida é descrito um conjunto de factores que justificam tal decisão:

1. O *GWT* é uma plataforma *open-source* pelo que pode ser utilizado livremente, seja para fins pessoais, educativos ou empresariais e esta é sempre uma mais-valia aquando da avaliação de uma plataforma. Sendo o código desta plataforma disponibilizada para a comunidade, beneficia de todas as vantagens de uma tecnologia *open-source*. As plataformas *Silverlight* e *Flex* são plataformas proprietárias e que poderão limitar a evolução futura da aplicação que se deseja desenvolver.
2. Embora exista muita e boa documentação sobre qualquer uma das plataformas (embora o *Flex* neste ponto perca pontos) as aplicações no *GWT* são totalmente desenvolvidas em *Java*, uma linguagem orientada à objectos, madura, robusta, muito utilizada no mundo do software e consequentemente muito bem documentada, sendo esta uma linguagem bastante acessível a qualquer desenvolvedor.
3. Por outro lado, existe a liberdade que o *GWT* oferece na importação dos seus projectos para qualquer *IDE Java*. Este é certamente um dos aspectos mais atraentes desta ferramenta. Deste modo, o desenvolvedor pode escolher um *IDE* da sua preferência, promovendo assim a sua experiência de utilização em ferramentas de desenvolvimento, beneficiando de aspectos importantes como usabilidade e tirando partido do conhecimento que já possui sobre as mesmas.
4. Outro dos aspectos importantes do *GWT*, também partilhado pelo *Flex*, é o facto de utilizar *RPC* na comunicação cliente/servidor, permitindo a partilha de objectos sobre a rede. Este aspecto facilita em muito a tarefa do desenvolvedor e contribui para a lógica das aplicações desenvolvidas.

O *GWT* foi desenvolvido pensando no desenvolvedor e na exigência das aplicações actualmente desenvolvidas para *Web*. Demonstra ser uma bela ferramenta de desenvolvimento e que deve ser adoptada ou testada por aqueles que precisem de desenvolver qualquer tipo de aplicação para a *Web*.

# 3 Proposta

Neste capítulo é feita a descrição de uma nova ferramenta baseada na Web. Esta ferramenta, designada *WebSketch*, consiste num editor de modelos, mais concretamente num editor de Protótipos Canónicos Abstractos (PAC's), modelos estes que servem de auxílio ao desenvolvimento de interfaces abstractas. Não existindo ainda nenhuma ferramenta do género sobre a plataforma Web, esta é uma oportunidade de, em contributo à Engenharia de Software, disponibilizar uma ferramenta que auxilie à elaboração deste tipo de modelos aproveitando todas as vantagens desta recente e poderosa plataforma.

Existindo já um conjunto alargado de plataformas que auxiliam o desenvolvimento deste tipo de aplicações, desenvolver uma aplicação Web exige um conjunto de esforços na sua modelação e implementação, uma vez que se pretende construir uma ferramenta tipicamente *Desktop* sobre uma plataforma completamente distinta, a Web. Desta forma, e tal como já foi evidenciado no Estado de Arte, dado que os ambientes de execução da aplicação são distintos, as características das aplicações em ambos os ambientes são também distintas.

## 3.1 Abordagem

No capítulo do Estado de Arte foi possível observar os principais desafios que se colocam ao desenvolvimento das aplicações Web bem como um conjunto de tecnologias relevantes e, foram avaliadas um conjunto de plataformas escolhidas para dar suporte ao desenvolvimento da ferramenta. Focados os principais desafios e escolhida a plataforma, de seguida será descrito a abordagem escolhida para descrição da ferramenta:

- Especificação das funcionalidades gerais da ferramenta;
- Desenho da Interface para a aplicação (inclui o estudo de aplicações do mesmo tipo, sejam aplicações Desktop ou aplicações já existentes sobre a Web, de forma a estudar a sua organização visual e estilos de interacção utilizados);
- Especificação a nível de implementação:
    - Descrição das bibliotecas usadas;
    - Descrição da arquitectura da aplicação:
        - Arquitectura vista camadas, de modo a observar de que forma está estruturada o projecto;
        - Arquitectura vista módulos, especificando a relação entre os vários módulos que compõem o projecto;
        - Vista Componente-Conector de forma a representar o comportamento da aplicação durante a sua execução;

É de acordo com esta abordagem que serão descritas e justificadas todas as decisões tomadas durante o processo de desenvolvimento da aplicação.

## 3.2 Requisitos e Funcionalidades

Como primeiro passo de qualquer processo de desenvolvimento de software, de seguida serão descritos os principais requisitos da aplicação bem como principais funcionalidades.

Sendo esta uma ferramenta baseada na Web, esta tem antes de tudo oferecer ao utilizador a oportunidade de se registar a fim de criar a sua conta, para deste modo ter capacidade de gerir as suas tarefas, pois este é considerado um requisito base de qualquer aplicação Web. Como ferramenta de modelação, a aplicação tem de fornecer ao utilizador suporte à gestão de modelos, que se pode traduzir nas seguintes tarefas: Criar, Editar e Eliminar Modelos. Estas são as principais tarefas de qualquer ferramenta de modelação onde, a partir destas, o utilizador já tem a capacidade de criar e gerir os seus próprios modelos utilizando o editor.

No entanto, sendo esta uma ferramenta de apoio ao processo de desenvolvimento de software e, sabendo que este é um processo onde a colaboração e comunicação são factores fundamentais no sucesso de qualquer produto de software, seria interessante proporcionar aos utilizadores mecanismos de colaboração. Note-se que esta dissertação não foca o estudo de editores colaborativos, no entanto, não deixa de ser importante fornecer as funcionalidades base para tal fim, evidenciando as potencialidades da aplicação da plataforma onde esta é desenvolvida. Assim sendo, o colaborador terá também a capacidade de adicionar colaboradores aos seus modelos.

Os casos de utilização essências da aplicação são apresentados na figura seguinte:

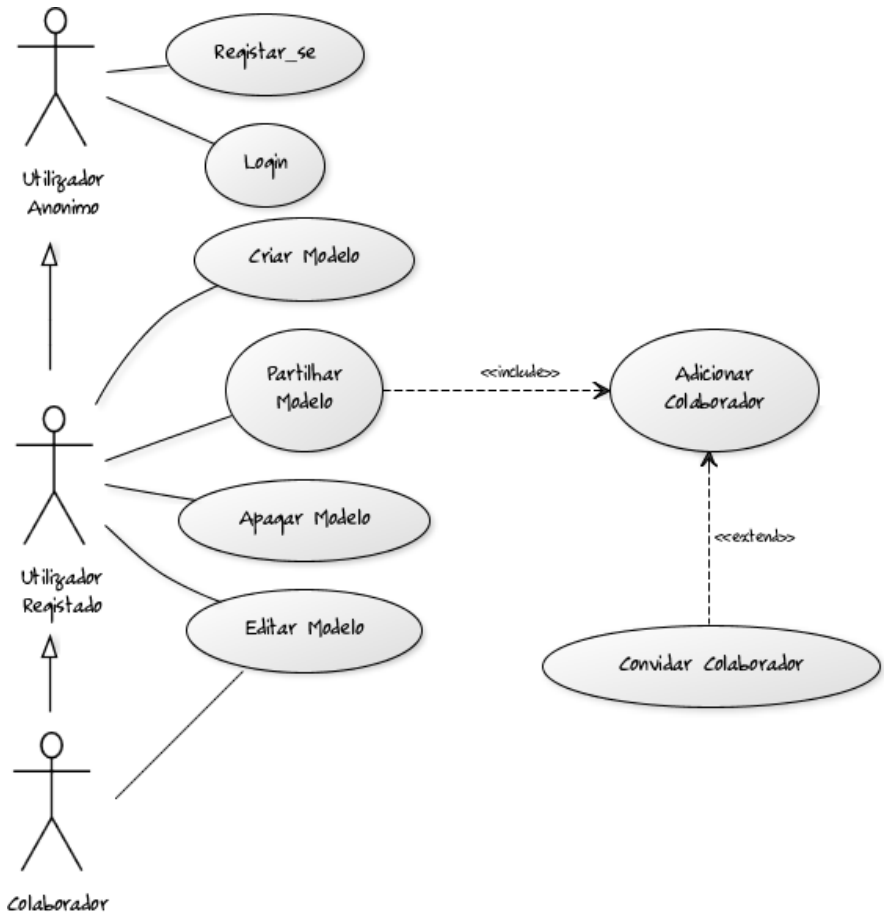


Figura 31 - WebSketch - Funcionalidades Principais

Estas foram as principais funcionalidades identificadas para a aplicação. Desta forma, temos uma aplicação onde o utilizador tem a capacidade de, sobre a plataforma Web editar modelos PAC com a capacidade de colaboração com outros utilizadores.

Na próxima secção serão apresentados os estudos realizados para a implementação das interfaces para a aplicação.

## 3.3 WebSketch – Desenvolvimento das Interfaces

Antes do desenho e implementação da interface da aplicação, foi realizada um estudo às interfaces de aplicações do mesmo tipo, ou seja, editores de modelos que se encontram sobre o ambiente Desktop ou Web, com o objectivo de concluir de que forma estão organizadas as interfaces e quais os estilos de interacção mais utilizados, sendo também possível observar as diversas semelhanças e respectivas diferenças na organização visual das aplicações Desktop vs Web. Dada a sua extensibilidade, o estudo realizado foi colocado em anexo, onde podem ser consultadas as interfaces estudadas e as conclusões tiradas.

### 3.3.1 Desenho das Interfaces

De acordo com as interfaces estudadas, é agora apresentada a principal interface pensada para o WebSkecth. A figura seguinte consiste na representação abstracta da interface:

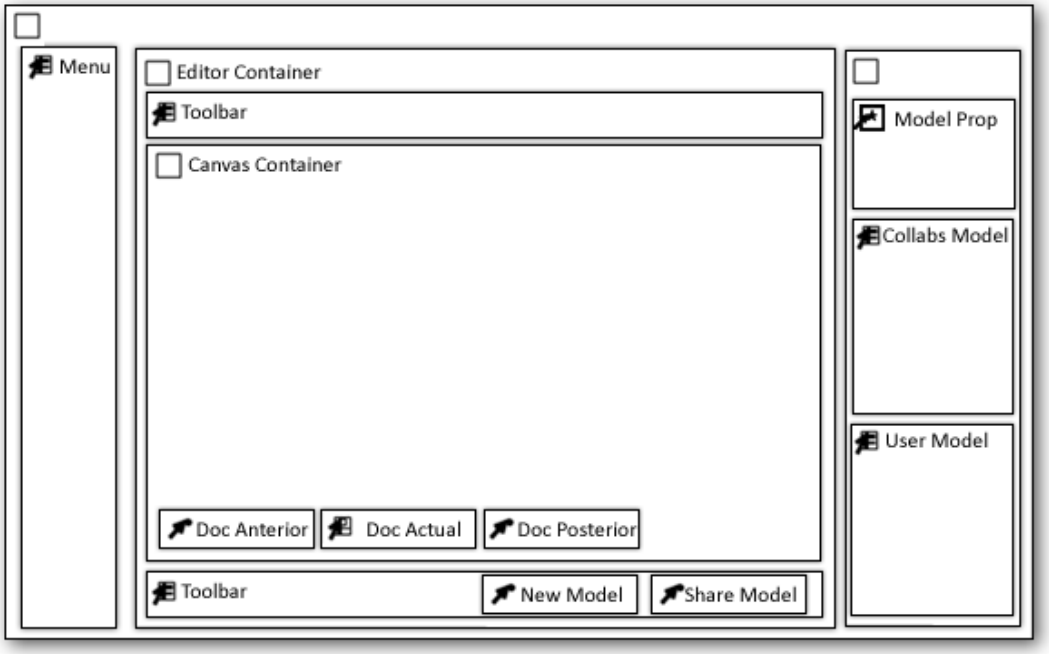


Figura 32 - WebSkecth PAC

De acordo com o PAC, a interface fica ordenada da seguinte forma:

Na zona lateral esquerda situa-se o menu com os símbolos PAC que o utilizador poderá arrastar para o canvas a fim de criar os modelos PAC.

Na zona central temos a área principal do editor. Na zona superior este contém uma *toolbar* que irá conter as funcionalidades básica de um editor, tal como funcionalidades de *undo/redo*, *zoom in/zoom out,* etc… A zona central consiste em um conjunto de vários documentos criados pelo utilizador relativos aos seus modelos e na zona inferior situam-se duas funcionalidades fundamentais, que consistem na criação de novos modelos e na partilha desses mesmos modelos.

Na zona lateral direita encontram-se três vistas distintas:

- A vista superior consiste nas propriedades do modelo que está sendo editado no momento.
- A vista central consiste na lista de colaboradores pertencentes ao modelo editado no momento. O utilizador pode adicionar colaboradores sempre que criar um novo modelo.
- A vista inferior consiste na lista de modelos do utilizador. Este pode através desta vista manipular os seus modelos.

## 3.4 Implementação

Nesta secção é descrito o processo de desenho e implementação da arquitectura da aplicação de acordo com os requisitos delineados anteriormente. Para uma boa compreensão da arquitectura implementada, serão apresentadas diversas vistas da mesma a fim de especificar de que modo foi esta estruturada, especificando a relação entre os módulos que a compõem. Antes de mais, serão descritas as bibliotecas adoptadas fazendo uma breve descrição de cada uma delas.

### 3.4.1 Bibliotecas usadas

Para além das bibliotecas disponibilizadas pela plataforma GWT, foram importadas algumas bibliotecas para dar auxílio, principalmente, à implementação da componente gráfica da aplicação. As respectivas bibliotecas são:

- GXT (23);
- GWTCanvas (34)(GWT-Incubator ) (posteriormente excluída)
- GWT Graphics (35)(biblioteca posteriormente escolhida ao invés da biblioteca GWTCanvas)

#### *3.4.1.1 Ext- GWT (GXT)*

O Ext-GWT, também designada GXT, trata-se de uma biblioteca Java desenvolvida com o intuito de facilitar o desenvolvimento de aplicações Web, utilizando como base a plataforma GWT. O GXT disponibiliza uma API para a construção de interfaces, dando poder ao programador de manipular sobre o browser um conjunto vasto de widgets e estilos de interacção que tradicionalmente se encontram nas aplicações Desktop.

A biblioteca GXT inclui: (23)

- boa performance e um largo conjunto de widgets personalizáveis;
- suporte à CSS;
- suporte ao RPC , JSON e XML;
- suporte a Java 1.5, incluindo enumerações, genéricos, entre outros…

O GXT é compatível com a maioria dos browsers, incluindo: (23)

- Internet Explorer 6+;
- FireFox 1.5+ (PC, Mac);
- Safari 3+;
- Opera 9+ (PC, Mac);

## Tipos de dados

Tratando-se neste caso de aplicações Web em que os dados situam-se tradicionalmente no lado servidor, qualquer técnica adoptada na componente cliente irá contribuir para a performance da própria aplicação.

Para este fim, o GXT disponibiliza uma API (36) que fornece técnicas avançadas para manipulação e gestão de dados. Fazendo o uso de classes como o ModelData, BaseModel ou ainda o BeanModel, bem como técnicas de manipulação local de dados (cache), a biblioteca oferece ao programador um conjunto de funcionalidades base para a manipulação de dados nos mais complexos widgets, tais como Tabelas, Grelhas ou ainda Listas, garantindo uma melhor performance à aplicação.

| Interface Summary | |
|---|---|
| BeanModelMarker | Marker interface that identifies bean model classes indirectly without having to modify the bean. |
| BeanModelTag | Tag interface used to identify Java Beans that may be used to generate `BeanModel` instances. |
| ChangeEventSource | Interface for object that notify listeners when changed. |
| ChangeListener | Interface for objects that listen for model change events. |
| DataProxy<D> | Defines the interface for objects that can retrieve data. |
| DataReader<D> | Interface for objects that translate raw data into a given type. |
| GroupingLoadConfig | List load config with support for grouping. |
| ListLoadConfig | Load config interface for list based data. |
| ListLoader<D> | Interface for list based loaders. |
| ListLoadResult<Data> | Load result interface for list based load results. |
| LoadConfig | Base interface for load config objects. |
| Loader<D> | Interface for objects that can load remote data. |
| Model | Primary interface for GXT model objects with event support. |
| ModelComparer<M extends ModelData> | Compares the model instances for equality. |
| ModelData | Primary interface for GXT model objects without support events. |
| ModelIconProvider<M extends ModelData> | |
| ModelKeyProvider<M extends ModelData> | Instances of this class provide unique keys for models. |
| ModelStringProvider<M extends ModelData> | Interface for objects that can translate a model's typed values to strings. |
| PagingLoadConfig | A `ListLoadConfig` with support for limit and offset values. |
| PagingLoader<D extends PagingLoadResult<?>> | A loader for a pageable set of data. |
| PagingLoadResult<Data> | A `LoadResult` for paging loaders. |
| TreeLoader<M extends ModelData> | A `Loader` for trees. |
| TreeModel | A Model that supports parent and children. |

Ilustração 1 – GXT Modelo de Dados (53)

## Arquitectura MVC

Sendo o GXT uma biblioteca desenvolvida para dar suporte à uma plataforma dedicada ao desenvolvimento de aplicações Web, esta já oferece uma aproximação à arquitectura MVC.

O seguinte esquema consiste na representação do modelo MVC no GXT:

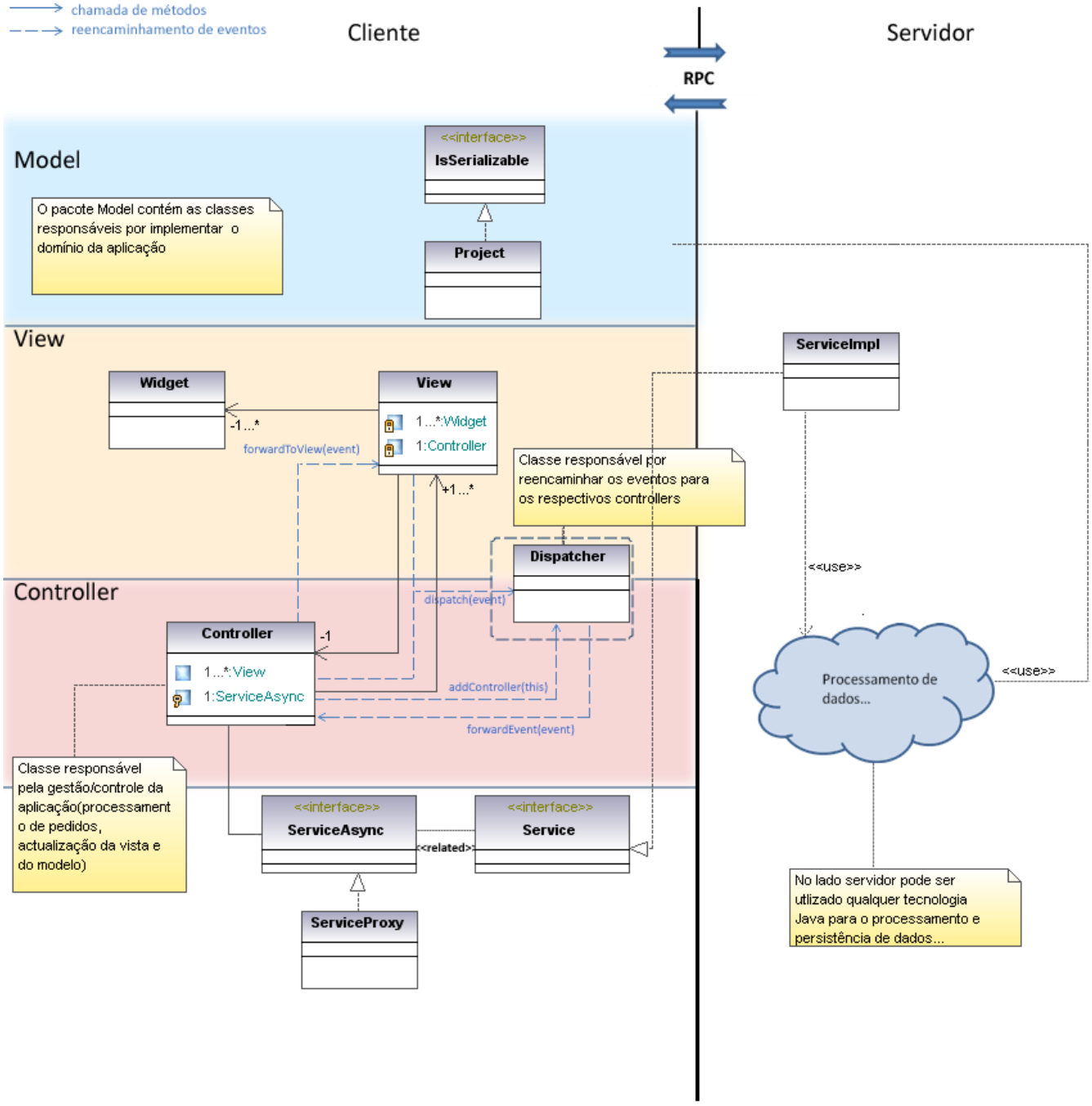


Figura 33 – Arquitectura MVC no GXT

Sendo o GWT uma plataforma baseada em eventos, verifica-se que toda a implementação do MVC é feita do lado cliente, sendo possível aceder aos serviços implementados no servidor através de comunicação RPC.

#### 3.4.1.2 Dispatcher

Enquanto a arquitectura MVC tradicional é representada pelas três componentes Model, View, Controller e suas respectivas relações, a aproximação do GXT à arquitectura MVC inclui uma classe adicional, a classe Dispatcher.

Esta classe vem alterar a relação e o comportamento das componentes no MVC, na medida em que esta é a responsável por reencaminhar todos os eventos produzidos na aplicação para os respectivos Controllers (37). Esta funciona basicamente como um *observable*, sendo os Controllers os seus *observers*, pelo que no início da execução da aplicação, cada vez que um controlador é criado, este adiciona-se ao Dispatcher. Durante a execução da aplicação, quando um evento é produzido, o Dispatcher, contendo todos os controllers, verifica qual ou quais destes suportam o evento, reencaminhando o respectivo evento.

Esta técnica permite manter uma maior independência entre as diferentes camadas, libertando a View da responsabilidade de realizar o correcto reencaminhamento dos eventos para os respectivos controllers.

#### 3.4.1.3 View

A View é responsável por inicializar e refrescar a interface da aplicação sempre que necessário. Aquando da ocorrência de um evento por parte do utilizador, a classe gera um evento e reencaminha-o para o respectivo controller através do Dispatcher.

#### 3.4.1.4 Model

Embora o modelo de domínio da aplicação seja implementada no lado cliente, uma vez que a comunicação entre o cliente/servidor é baseado no RPC, os objectos são facilmente partilhados entre ambas as partes. Deste modo, toda a informação gerada pela aplicação é mantida no lado servidor podendo ser acedida pelo cliente através da requisição de serviços.

#### 3.4.1.5 Controller

O Controller assume o papel principal na medida em que é o responsável pela transição e processamento de dados. Toda a informação da aplicação passa pelo Controller, sendo este o responsável por processar e reencaminhar a informação através da requisição de serviços ao lado servidor. Esta classe tem também a capacidade de gerar eventos. Por exemplo, quando existe uma modificação no modelo da aplicação, o controller notificado pode gerar um evento com os dados actualizados. Este evento irá ser reencaminhado pelo Dispatcher para os controllers que suportem o evento, para que estes possam notificar as respectivas Views e, desta forma, fazer o correcto “refrescamento” da aplicação. A utilização da comunicação RPC dá a sensação de que os dados estão sendo tratados localmente, no entanto, tal como pode ser observado na figura anterior, é feita a utilização de um proxy remoto. O proxy, contendo o pedido do controlador, reencaminha-o para o objecto do lado servidor que realmente contém o serviço desejado, reencaminhando posteriormente o resultado do serviço requisitado para o respectivo controlador.

A completude e eficiência desta biblioteca são razões para a sua adopção e utilização. Sendo a mais popular biblioteca para o desenvolvimento de interfaces sobre o GWT, esta biblioteca Java está em constante actualização e evolução, trazendo a cada nova versão melhorias e novas funcionalidades, tornando-a cada vez mais atractiva. Por estas razões e, pela elegância das interfaces e a vasta gama de estilos de interacções que esta permite, o GXT foi a biblioteca escolhida para o desenvolvimento deste projecto, ao invés da utilização da biblioteca disponibilizada por defeito pelo GWT.

#### *3.4.1.6 GWTCanvas*

Sendo uma biblioteca da Google, incluída no gwt-incubator (38), com suporte ao desenho de formas e imagens, esta foi a biblioteca inicialmente escolhida para dar suporte ao desenho de modelos na aplicação. O GWTCanvas fornece suporte:

- Métodos de rotação, translação e escala;
- Formas particulares, tais como arcos, linhas e curvas;
- Desenho e transformação de imagens;

No entanto, e apesar de esta ser uma biblioteca suportada pela Google, esta acabou por não ser alvo de grandes evoluções, apresentando principalmente deficiências nas rotinas de escrita de texto. Dadas estas desvantagens, foi decidido optar por outra biblioteca por sua vez mais completa e mais madura designada GWT Graphics.

#### *3.4.1.7 GWT Graphics*

Sendo uma biblioteca desenvolvida especialmente para o GWT, esta permite o desenho de gráficos, tendo sido testada sobre:

- Internet Explorer 6+;
- Firefox 3+;
- Safari 3.2+;
- Opera 9+;
- Chrome 4+;

Comparativamente à biblioteca GWTCanvas esta mostrou ser uma biblioteca mais robusta, permitindo um maior conjunto de funcionalidades sobre os gráficos e principalmente, maior eficácia nas rotinas de escrita de texto.

### 3.4.2 Serviços

Neste subcapítulo serão descritos os serviços implementados para a aplicação. Antes de mais, são apresentadas as classes responsáveis por declarar e implementar esses serviços. A figura seguinte, traduzida da Figura 14 apresentada no capítulo do estudo das plataformas, nomeadamente no estudo do mecanismo de comunicação baseado em RPC no GWT, descreve as respectivas classes:

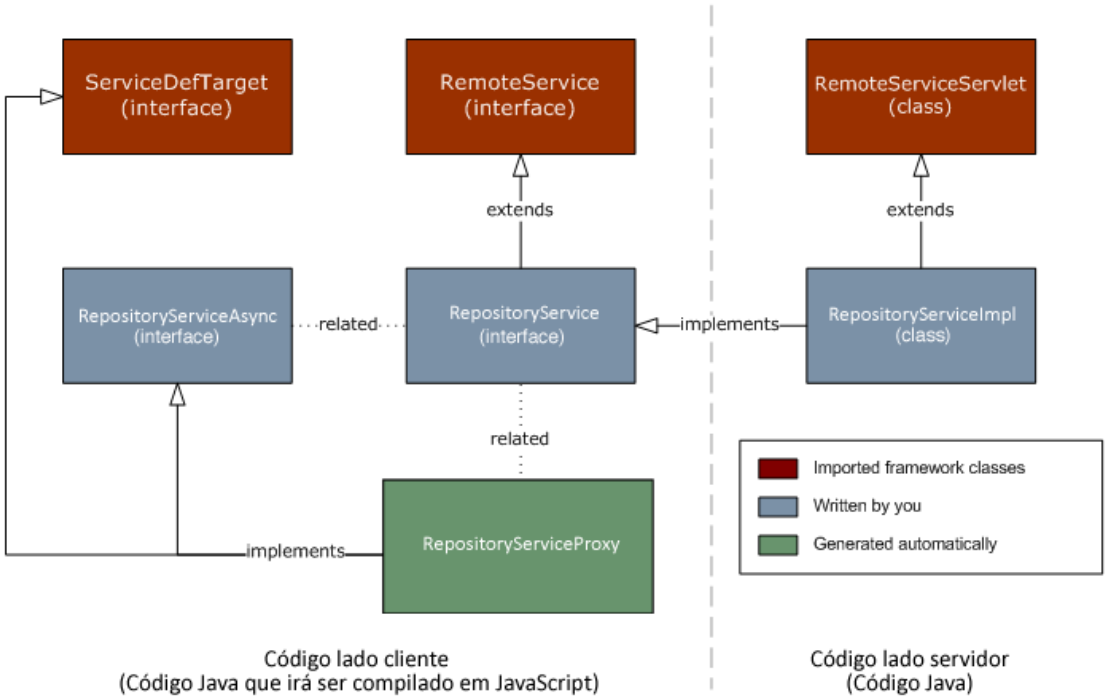


Figura 34 – Classes responsáveis por implementar os serviços e estabelecer a comunicação

Verifica-se que no lado cliente é declarada a interface *RepositoryService* responsável por declarar todos os serviços necessários à aplicação. Ligada à esta temos a interface *RepositoryServiceAsync,* interface esta que permite criar a versão assíncrona do serviço, implementada pela classe *RepositoryServiceProxy,* que consiste no proxy gerado pelo GWT e que será responsável por requisitar os serviços ao lado servidor. Do lado servidor são então implementados os serviços na classe *RepositoryService*.

Identificadas as classes responsáveis por declarar e implementar os serviços, de seguida é ilustrado os serviços implementados:

```
//services about models
void saveUpdatedModel(String modelName, ModelElement elem);
void saveModel(Model model, String username);
Model getModelByName(String modelName);
void removeModel(String modelName);
List<Model> getModels(String userName);
List<String> getModelsName(String username);
List<Model> getOpenModels(String username);


//services about  users
void saveUser(User user);
void removeUser(User user);
User getUser(String userName);
List<User> getAllUsers();
List<String> getAllUsersByName();
List<String> getAllUsersByEmail();
List<User> getUsersByModel(String modelName);
void addParticipantToModel(User user, String modelName);
void removeParticipantFromModel();
```

Figura 35 – WebSketch – Serviços implementados

Estes foram os serviços identificados para a gestão de utilizadores e respectivos modelos na aplicação.

### 3.4.3 Arquitectura

As decisões arquitecturais são um aspecto fundamental para o sucesso de qualquer aplicação. Enquanto numa aplicação Desktop tradicional a componente visual da aplicação e os seus respectivos dados encontram-se no mesmo local físico, numa aplicação Web estas mesmas componentes encontram-se em espaços físicos diferentes (cliente/servidor). Deste modo e, tratando-se neste caso de uma aplicação Web, as decisões arquitecturais revelam-se fundamentais para a boa performance da aplicação.

Ao longo deste capítulo serão descritas e justificadas as decisões arquitecturais do projecto desenvolvido, partindo-se de uma visão geral da arquitectura, especificando-se de seguida algumas das componentes mais importantes, mostrando a dinâmica/relação entre os diferentes elementos.

Sendo a arquitectura cliente/servidor a arquitectura base de qualquer aplicação Web e, sendo este projecto implementado sobre o GWT, plataforma baseada em eventos, tanto a implementação da componente visual como do próprio modelo da aplicação é feita do lado cliente, pelo que a comunicação com o servidor é feita utilizando comunicação RPC, através da requisição de serviços.

#### *3.4.3.1 Arquitectura da aplicação: Hosted Mode vs Web Mode*

Já referido no capítulo do Estado de Arte aquando do estudo do GWT, esta plataforma permite dois modos de execução, Hosted Mode e Web Mode. Sendo a primeira utilizada durante o processo de desenvolvimento da aplicação, esta permite a execução da aplicação em Java *bytecode* (linguagem na qual a aplicação é implementada), permitindo que o *debugging* da aplicação seja feito sem a necessidade de a executar na Web. Deste modo a "arquitectura Hosted Mode" da aplicação implementada pode ser representada tal como mostra a figura seguinte:

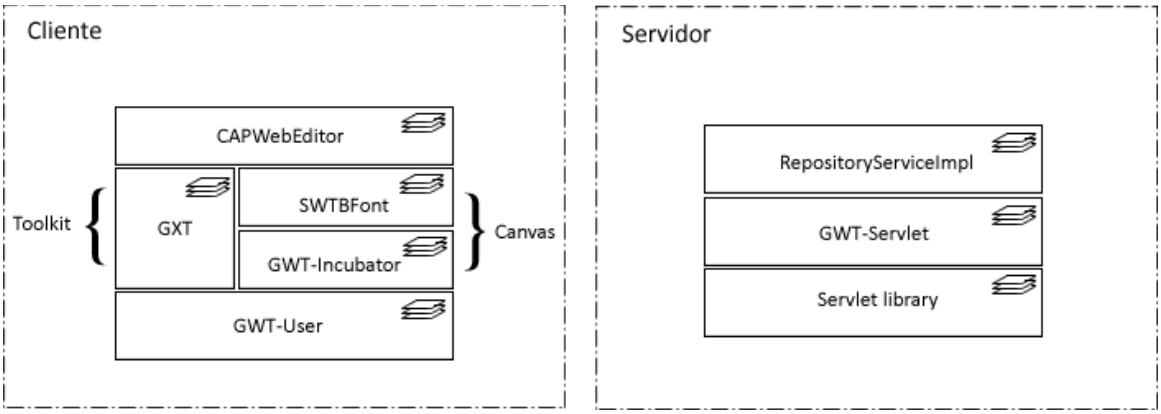


Figura 36 – Arquitectura da aplicação WebSketch em "Hosted Mode" – Vista Camadas

Pela observação da figura anterior, verifica-se que no lado cliente temos como camada base a biblioteca gwt-user. Esta biblioteca suporta o desenvolvimento de todas as componentes do lado cliente desenvolvidas sobre a plataforma GWT. Além desta, foram adicionadas outras bibliotecas como é o caso do GXT, *toolkit* escolhido para a implementação das interfaces da aplicação e ainda a biblioteca SWTBFont que, tal como já foi evidenciado anteriormente, consiste numa extensão à biblioteca GWTCanvas pertencente à biblioteca gwt-incubator, utilizada para implementar o *canvas* do editor. Assim sendo, a aplicação WebSketch foi desenvolvida sobre todas estas camadas, consistindo na camada inicial da aplicação.

Do lado servidor temos a implementação dos serviços declarados no lado cliente, suportada pela biblioteca *gwt-servlet* do GWT. Esta biblioteca baseia-se em Java Servlets para o processamento das requisições e respectivas respostas entre o cliente e o servidor.

O "modo de execução Web" (Web Mode) consiste, tal como o nome sugere, na execução da aplicação sobre a plataforma Web, para que possa finalmente ser testada e disponibilizada aos utilizadores finais. Para tal, o GWT utilizando o *Java-to-JavaScript compiler* converteu todo o código implementado, ficando este armazenado em um ficheiro JavaScript, neste caso de nome "*capwebeditor.nocache.js".* A arquitectura da aplicação em Web mode é ilustrada na figura seguinte:

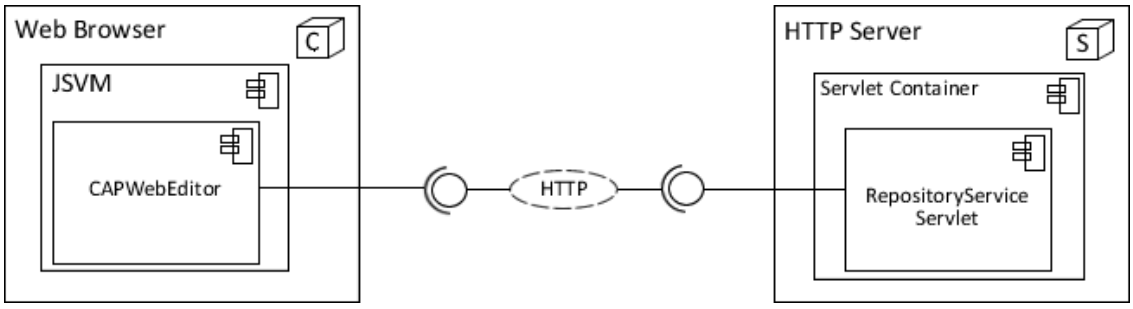


**Figura 37 – Arquitectura da aplicação WebSketch em "Web Mode" – Vista Componente-Conector**

Aqui a aplicação já está sendo executada sobre o ambiente para o qual foi originalmente implementada. No lado cliente:

- O browser carrega e processa a página "host" HTML gerada pelo GWT.
- Ao encontrar a tag "<script src="<Module Name>.nocache.js">" o código JavaScript do respectivo ficheiro é carregado e executado, sendo executada toda a lógica implementada na aplicação GWT.

O lado servidor que consiste em um servidor HTTP, possuindo a implementação de todos os serviços necessários, possui um servlet container responsável por executar os servlets, baseando-se estes no protocolo http para comunicação com o lado cliente.

### 3.4.3.2 Dependência entre as componentes

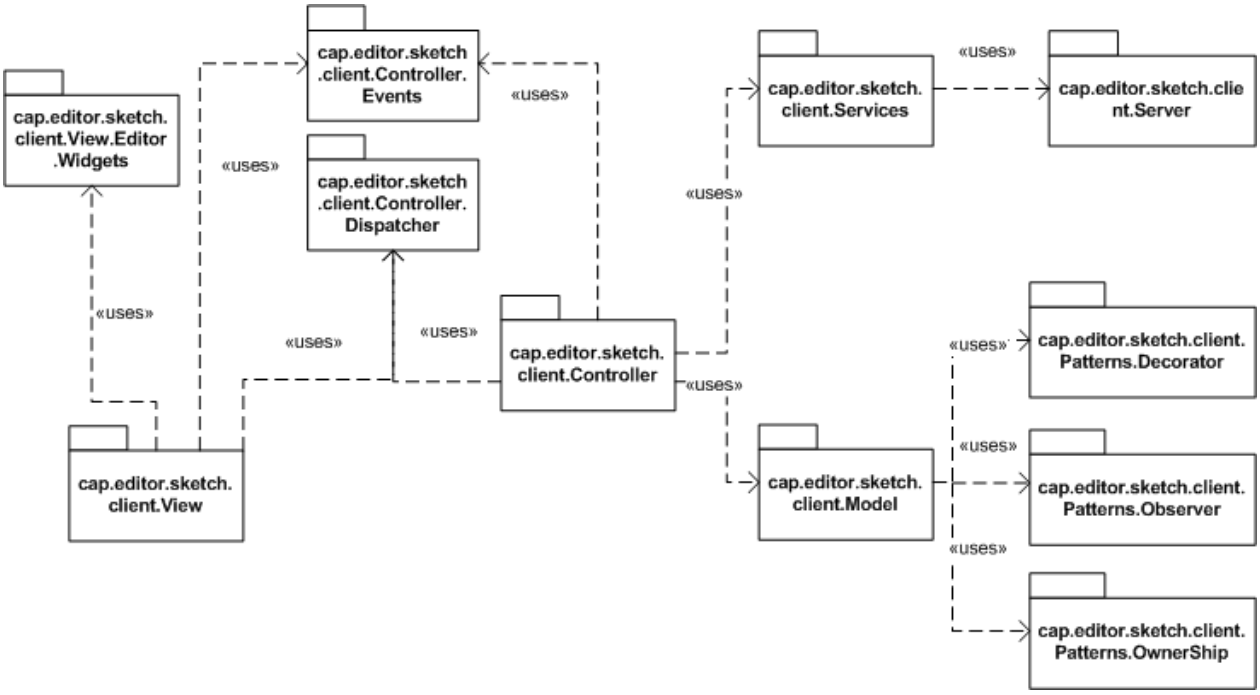


Figura 38 – Vista Módulos - Dependência entre os diferentes módulos

Depois da representação da arquitectura geral da aplicação, é apresentado o esquema que representa a dependência entre os diversos módulos. Através da observação do esquema é fácil perceber-se que o controller consiste no "núcleo" da aplicação. Fazendo uso dos principais módulos, controla o comportamento da aplicação mantendo canais de comunicação entre o cliente e o servidor. Respeitando a arquitectura MVC, os eventos produzidos nas vistas ou nos controles são reencaminhados pelo Dispatcher onde, tanto o controle como a vista fazem uso do pacote "Events" para a geração dos respectivos eventos. Sendo o modelo da aplicação representado pelo Model, o módulo ainda não referenciado, este faz uso do módulo "Patterns" a fim de modelar o comportamento dos seus elementos.

### 3.4.3.3 Edição de Modelos

Consistindo esta aplicação num editor de modelos, mais concretamente na edição de Protótipos Canónicos Abstractos e, tendo o utilizador a capacidade de criar e editar modelos e os seus respectivos elementos, surge um novo desafio de como poderão estas funcionalidades ser implementadas, tendo como base uma arquitectura cliente/servidor.

Perante este desafio surgem questões tais como:

1. Como modelar o comportamento entre os diferentes elementos?
2. Como e quando guardar esses elementos?

A solução para este tipo de perguntas irá ditar a performance e a facilidade de utilização da aplicação, pelo que se não for encontrada uma boa decisão arquitectural a aplicação não irá ter o comportamento desejado.

Tendo em conta as questões acima referidas, a solução encontrada foi a seguinte:

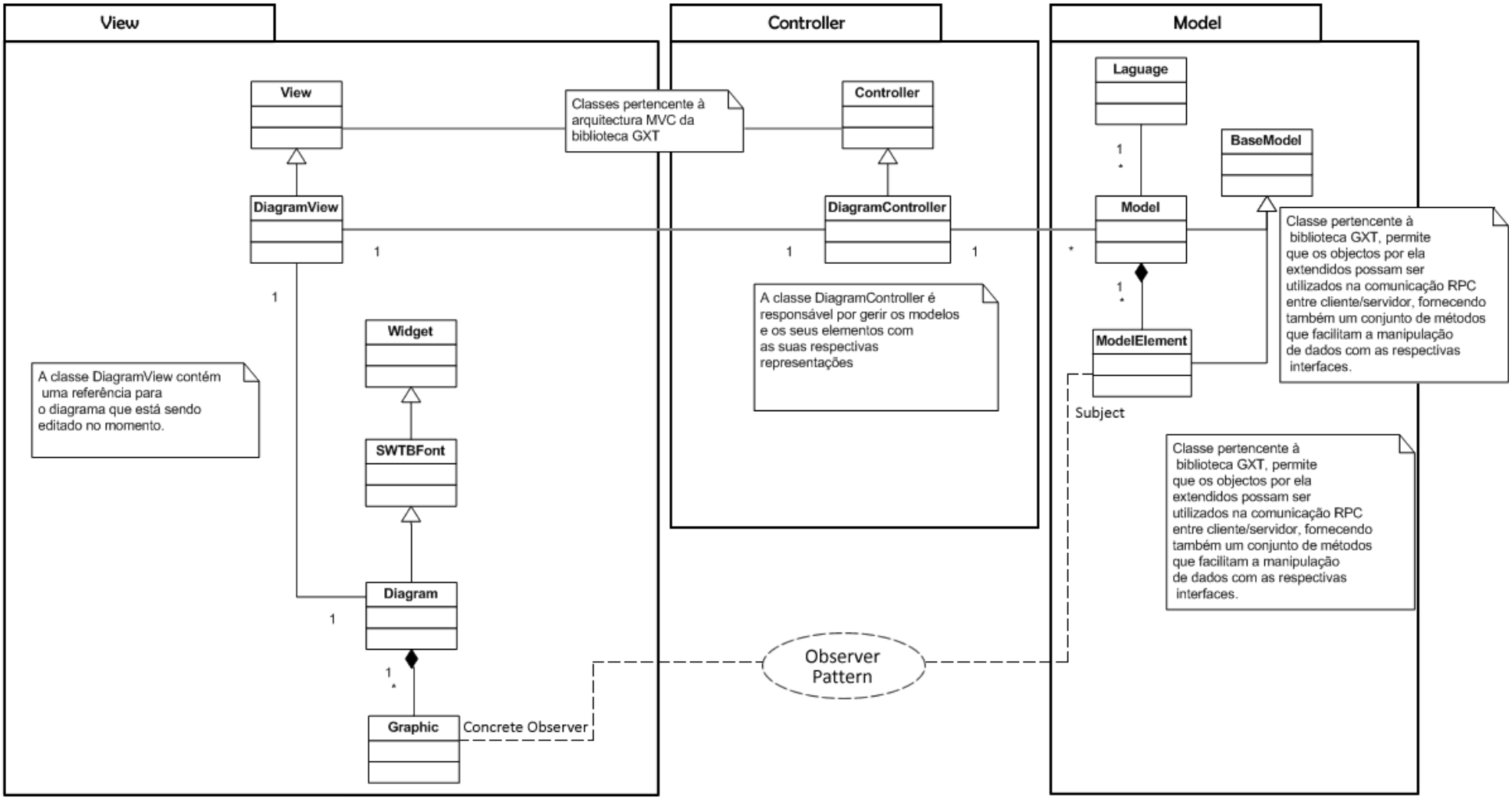


Figura 39 – Relação entre os modelos, os seus elementos e as suas representações

O esquema permite observar que para cada modelo criado a classe DiagramController cria um Diagram (canvas) que consiste na representação gráfica do modelo. O modelo por sua vez possui um conjunto de elementos cuja representação é da responsabilidade da classe Graphic.

Explicando mais detalhadamente o diagrama acima descrito, no pacote Modelo encontram-se as classes representantes do modelo de domínio da aplicação, onde basicamente temos a classes principais "Model" e "ModelElement". Associado à classe "Model" está a classe "DiagramController" do pacote Controller. Cada vez que o utilizador deseje criar um novo diagrama ou um novo "canvas" na aplicação, é criado antes de mais um novo "Model", sendo da responsabilidade da classe "DiagramController" delegar à classe "DiagramView" a criação de um novo "Diagram" (canvas) que irá consistir na representação gráfica do novo "Model" criado. Criado o novo modelo, o utilizador pode então adicionar novos elementos a fim de construir o seu modelo. Desta forma, temos que para cada "Model" está asssociado um conjunto de "ModelElement's" cuja representação gráfica é da responsabilidade da classe "Graphic". Desta forma, cada vez que é criado um novo item do modelo, é criado um novo "ModelElement" com a informação do item criado. Criado um novo "ModelElement", a classe "DiagramController" delega para a classe "DiagramView" a criação de um novo "Graphic" que irá ser criado de acordo com a informação do "ModelElement" que representa. Temos assim uma relação de "Observer/Subject", onde a vista (neste caso as classes "Diagram" e "Graphic") actualiza-se sempre que o Modelo sofre alterações (classes "Model" e "ModelElement").

Para uma melhor compreensão de como os pedidos se processam de acordo com esta arquitectura, de seguida serão apresentados dois diagramas de sequência que irão representar respectivamente o processo de criação de um modelo e o processo de criação de elementos dentro de um modelo. Os diagramas irão permitir compreender de que forma foi implementada a lógica do editor, seguindo a arquitectura MVC.

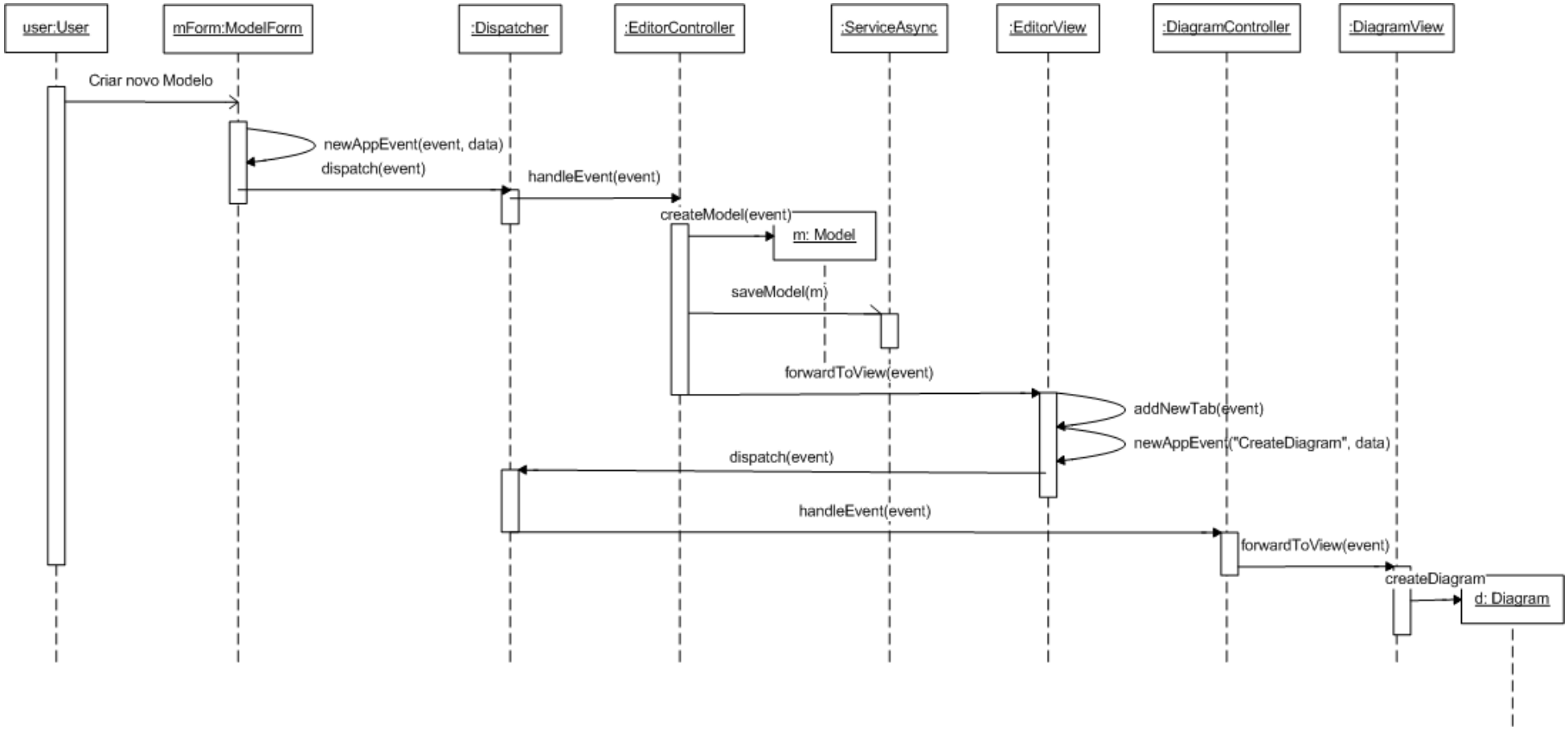


Figura 40 – [Diagrama de Sequência] Processo de criação de um novo Modelo

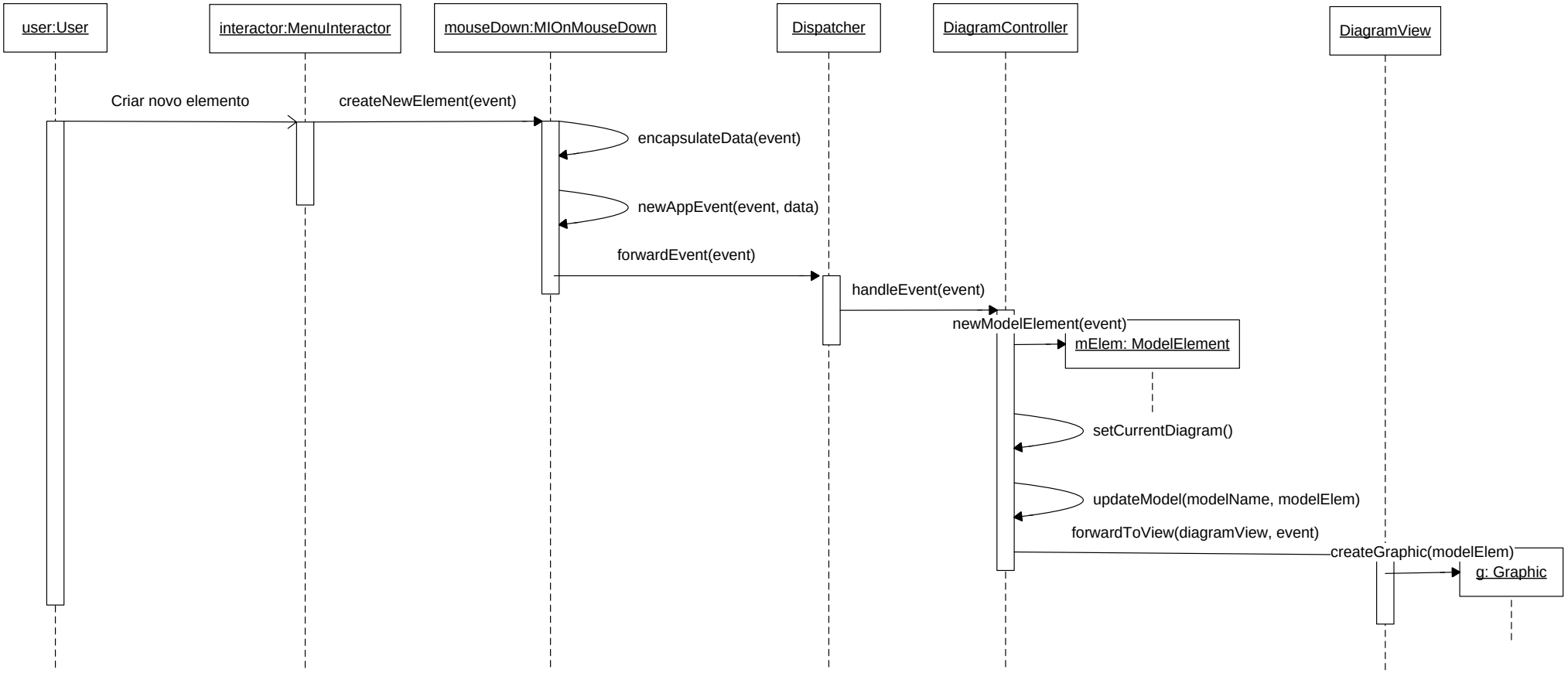


Figura 41 - Figura 42 – [Diagrama de Sequência] Processo de criação de um novo elemento no Modelo

Foi pretendido através dos diagramas anteriormente apresentados, mostrar de que forma foi organizada e implementada a arquitectura da aplicação. Respondendo a todos os requisitos estabelecidos esta apresenta uma solução flexível, baseada na arquitectura MVC. A independência estabelecida entre as diversas componentes permite que a aplicação seja modificada ou estendida sem muita dificuldade, fornecendo ao programador facilidades de modificação e implementação, aumentando as potencialidades da ferramenta.

# 4 Resultados

Faladas nas questões de desenho e implementação da ferramenta, de seguida serão apresentados os resultados finais obtidos.

## 4.1 Interface Geral da Aplicação

A interface geral da aplicação é dada pela seguinte figura:

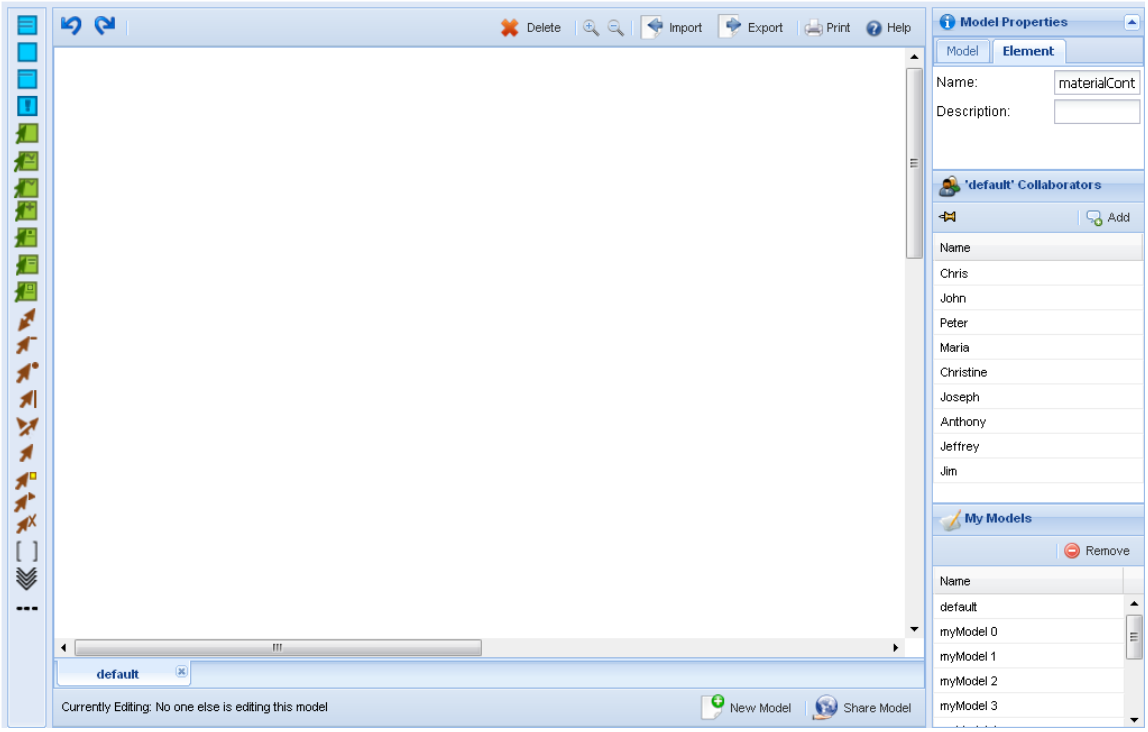


Figura 43 – WebSkecth – Interface principal

Como pode ser observado pela Figura 83, a interface da aplicação é composta por quatro áreas distintas. À esquerda encontra-se o menu composto pelos elementos do Protótipos Canónicos Abstractos, que são usados pelo utilizador para a edição dos modelos PAC. A zona superior é composta por uma *toolbar* que contém as funções principais que podem ser aplicados aos elementos de um modelo ou ao próprio modelo. Entre essas funcionalidades, temos as opções de *undo/redo*, eliminar elementos de um modelo, exportar ou importar um modelo, entre outros. À direita encontram-se três subáreas distintas, relativas às propriedades dos modelos e seus elementos, aos colaboradores de um respectivo modelo e a galeria de modelos do utilizador. Na zona inferior da aplicação existe mais uma *toolbar*, esta com duas funcionalidades distintas, sendo uma delas para a criação de novos modelos pelo utilizador e a outra que dá a possibilidade de adicionar colaboradores à um determinado modelo (colaboradores esses que ao serem adicionados, aparecem no painel à direita já mencionado). Na zona central, sendo esta a zona mais relevante ou de maior actividade na aplicação encontra-se a área de edição de modelos. É aqui que o utilizador irá modelar os seus modelos PAC. Uma vez que a aplicação é executada dentro de um browser, a interface da aplicação foi desenvolvida tendo em mente a maximização da área de edição.

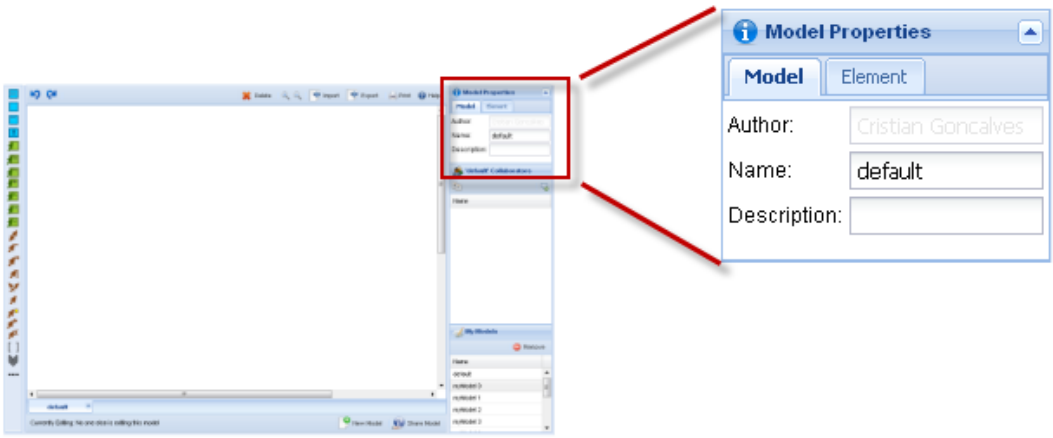


Figura 44 - WebSkecth (Janela de propriedades dos modelos e elementos)

A janela de propriedades contém duas abas respectivas às propriedades dos modelos e dos seus elementos. Aqui o utilizador pode consultar ou modificar directamente as propriedades de cada um deles, pelo que as modificações são actualizadas directamente na aplicação.

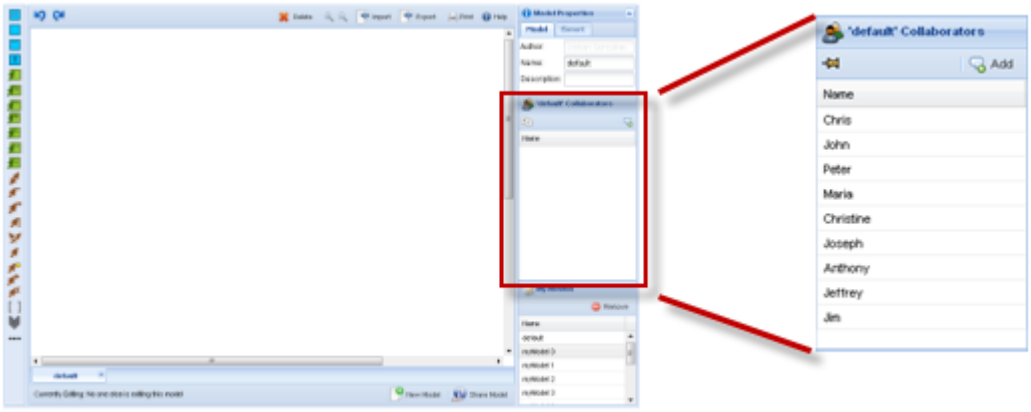


Figura 45 – Painel de Colaboradores

Apesar de esta não ter sido uma funcionalidade desenvolvida na sua íntegra, tratando-se esta de uma aplicação Web e de uma ferramenta a ser integrada no processo de desenvolvimento de software onde a comunicação é fundamental, a adição desta funcionalidade serve para demonstrar a potencialidade deste tipo de ferramentas, mostrando que é possível implementar mecanismos de suporte e comunicação que ajudem aos diferentes colaboradores de um projecto a trabalharem em conjunto em um ambiente partilhado. Neste caso, são apresentados os nomes dos colaboradores que se encontram a trabalhar em conjunto em um determinado modelo.

Para criar um novo modelo, o utilizador pode clicar no botão “New Model”, disponível na zona inferior ou utilizar a tecla de atalho “Alt + T”. Surge então uma nova janela, onde o utilizador pode nomear o novo modelo e possivelmente atribuir-lhe uma breve descrição.

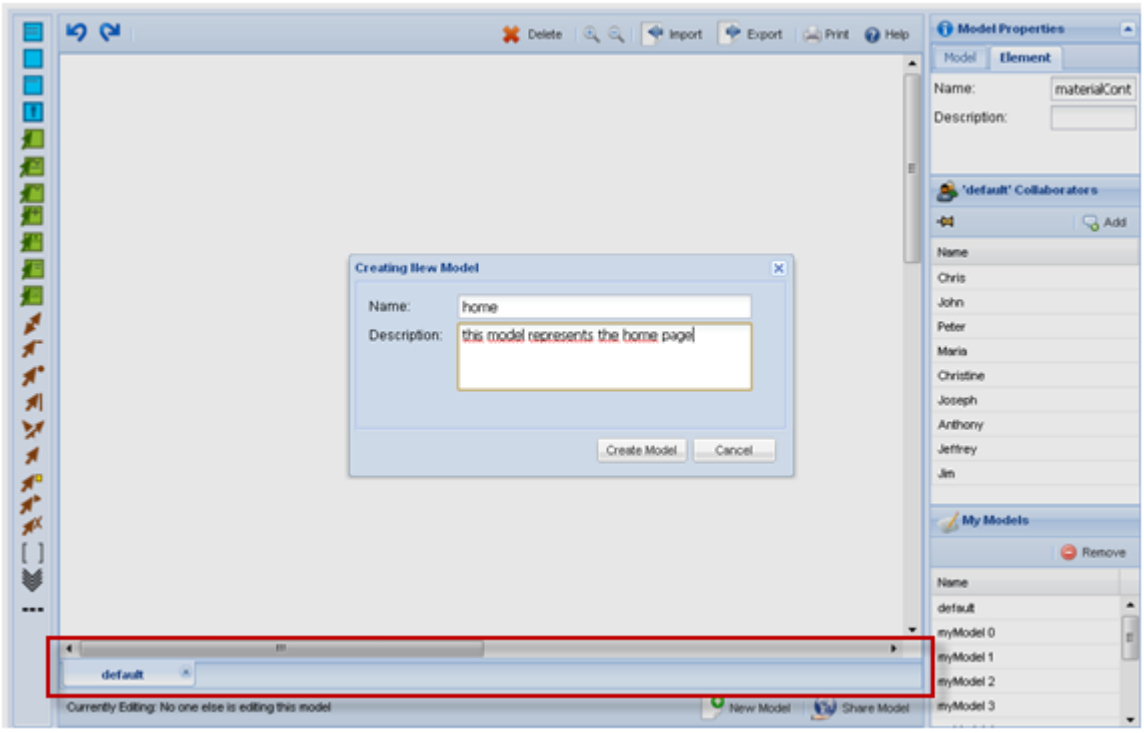


Figura 46 – Criação de um novo Modelo

Pela Figura 46, é possível observar na área salientada a vermelho a criação de abas para cada modelo criado. Esta técnica de interacção foi escolhida, pois permite ao utilizador a edição de diferentes modelos numa mesma área da aplicação. Desta forma, e fácil ao utilizador interagir com os diferentes modelos, sendo possível na interface da aplicação a edição de diferentes modelos numa mesma área de interacção.

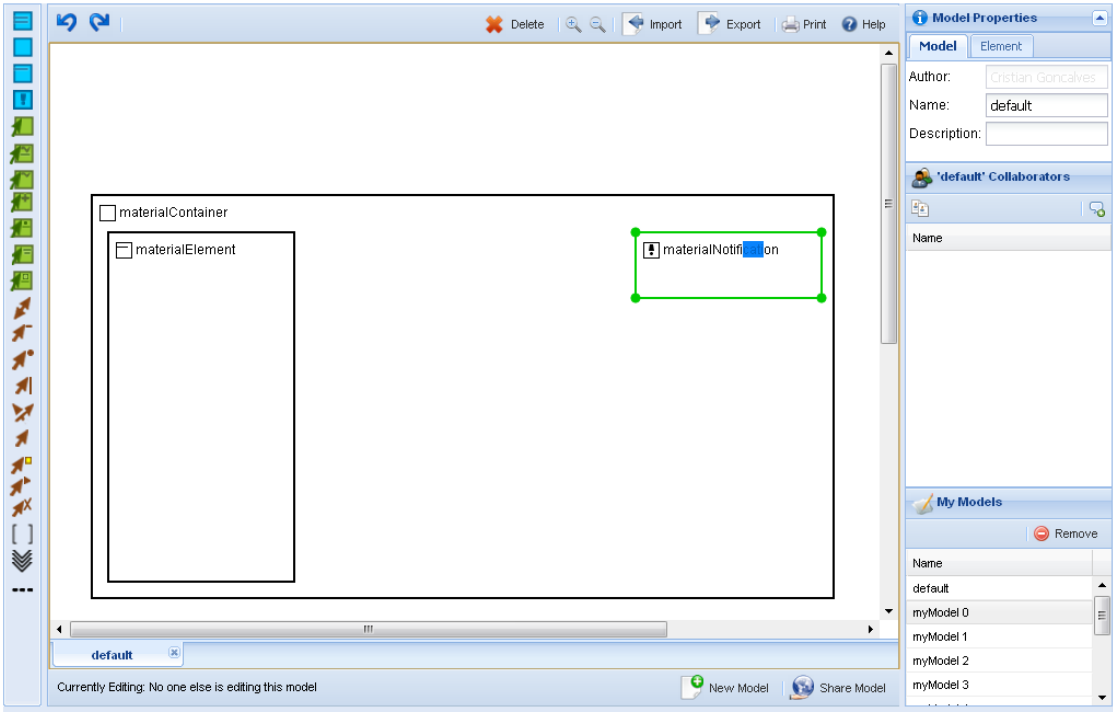


Figura 47 – Editando um modelo (interagindo com os elementos no canvas)

Criado um novo modelo, o utilizador pode então começar a editar os seus elementos. É possível adicionar, arrastar, redimensionar os elementos no canvas. Cada elemento tem na sua definição o conceito do *Ownership*, para que quando um elemento for arrastado ou apagado, todos os seus "filhos" (figuras dentro da figura em causa) sejam também movidos ou apagados.

Para editar um elemento do Modelo, o utilizador pode utilizar a janela na zona superior direita, correspondente às propriedades do modelo ou ainda utilizar a tecla de atalho "Ctrl + Shift" e clique com o botão esquerdo do rato, de forma a abrir uma janela popup com as propriedades do elemento.

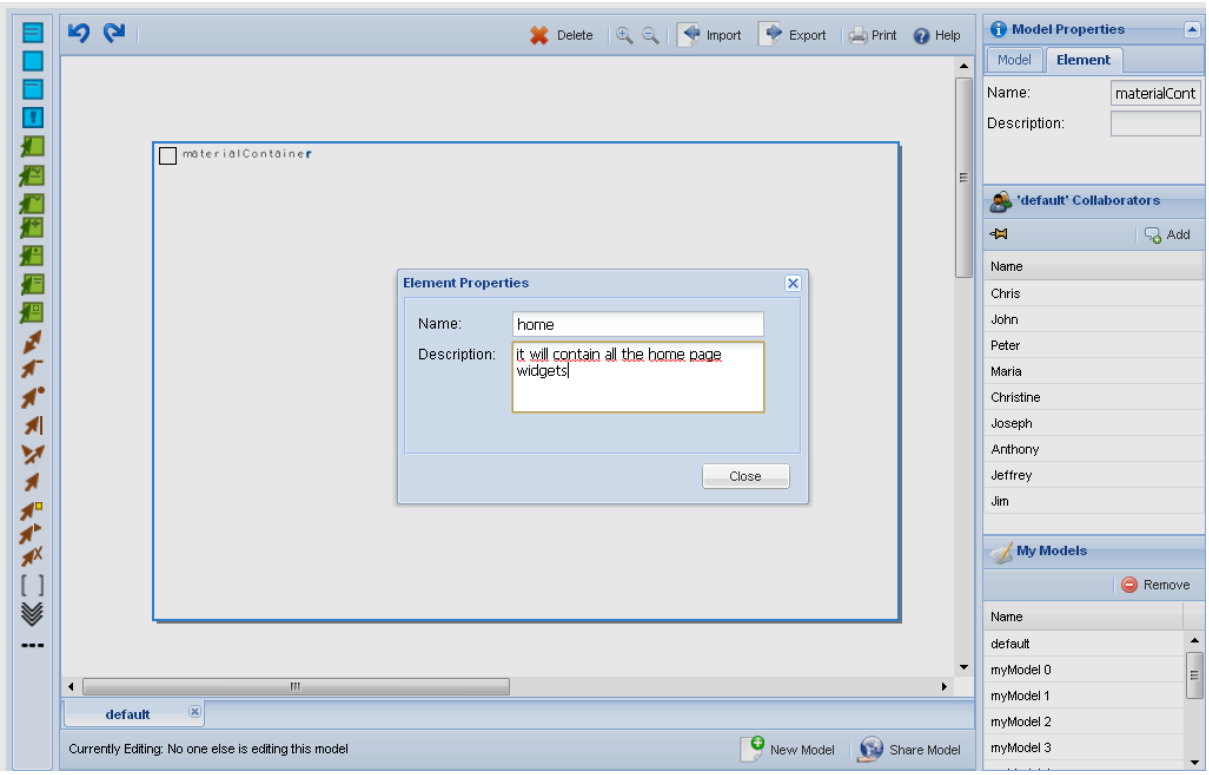


Figura 48 - WebSketch - Editando elementos

Assemelhando-se aos editores de modelos estudados durante a respectiva Tese, a aplicação foi desenvolvida com o intuito de se tornar uma ferramenta fácil de utilizar, de fácil compreensão, para que possa ser facilmente interiorizada e utilizada por qualquer utilizador na Web.

# 5 Conclusões e Perspectivas Futuras

## 5.1 Conclusões

As tecnologias actualmente disponíveis permitem o desenvolvimento de aplicações cada vez mais ricas, transportando para a Web características que normalmente se encontravam apenas em ambiente Desktop. Este tipo de aplicações encontra-se em verdadeira ascensão sendo alvo de estudo por parte de grandes entidades como a Google ou a Microsoft, reforçando ainda mais a expectativa de evolução das mesmas.

Nesta dissertação foi feita uma comparação entre as aplicações Desktop e as aplicações Web de forma a se identificar diferenças e semelhanças analisando-se em alguns casos de que forma as aplicações Desktop são migradas para o ambiente Web. Foi evidenciada a arquitectura MVC como elemento fundamental das aplicações Web actualmente desenvolvidas e analisadas algumas técnicas ou tecnologias mais usadas no seu desenvolvimento. Conclui-se que as tecnologias e técnicas disponíveis permitem desenvolver aplicações Web cada vez mais robustas e mais produtivas, onde as suas facilidades de acessibilidade e manutenção chamaram a atenção de muitas entidades que viram aqui a oportunidade de distribuir o seu software de uma forma simples e barata.

Realizado o estudo, foi proposto a implementação de uma ferramenta baseada na Web, designada *WebSketch* que consiste num editor para a modelação de Protótipos Canónicos Abstractos. Para tal foi utilizada o GWT, plataforma recentemente disponibilizada pela Google, que permite o desenvolvimento de aplicações Web em linguagem Java. Esta plataforma revela-se poderosa na medida em que permite o desenvolvimento das aplicações numa linguagem madura e robusta. No entanto, por ser uma plataforma recente e em constante evolução possui ainda algumas imperfeições com as quais o programador tem de saber lidar.

A ferramenta implementada demonstra ainda a possibilidade de colaboração que, embora não implementado, serve como referência da potencialidade não só da ferramenta como de qualquer aplicação desenvolvida sobre a Web.

Ao longo do desenvolvimento da aplicação foram enfrentadas algumas dificuldades tanto a nível de desenho como a nível de implementação da ferramenta. Um dos principais desafios foi a implementação do canvas, pois sendo a primeira vez a trabalhar com este tipo de elementos, de início a inexperiência acabou por se revelar, pelo que a lógica de implementação do canvas foi repensada algumas vezes. Por outro lado, a biblioteca *GWTCanvas* por ser uma biblioteca recente e precisando de ganhar ainda alguma maturidade apresenta alguns problemas. Um destes problemas deve-se ao facto de em alguns casos o *canvas* não possuir um comportamento completamente idêntico em *Hosted Mode* e *Web Mode*, obrigando a tomar em atenção alguns detalhes de implementação na tentativa de corrigir tais inconsistências. Outro problema consistiu na falta de documentação para esta biblioteca, sendo por vezes difícil encontrar soluções para determinados problemas.

Por outro lado o desenho da arquitectura do editor, nomeadamente na edição dos modelos e seus elementos, revelou-se um grande desafio na medida em que a representação e os dados dos elementos encontram-se no lado cliente e lado servidor respectivamente, pelo tiveram de ser testadas e avaliadas diversas soluções procurando sempre não afectar a performance da aplicação.

## 5.2 Perspectivas Futuras

O concluir de qualquer trabalho de investigação deixa sempre uma porta aberta a futuros estudos, pelo que esta dissertação não é excepção. Desta forma, no âmbito desta investigação poderão existir esforços futuros:

- No lançamento de uma versão mais estável da ferramenta proposta;
- Na investigação mais detalhada na área dos editores colaborativos sobre a Web, explorando a tecnologia *Operational Tranformation*;
- No suporte a geração de documentos XMI permitindo a integração dos modelos desenvolvidos em outras ferramentas de modelação;
- Na extensão da ferramenta à outro tipo de modelos.

# Bibliografia

1. **Constantine, Larry.** Canonical Abstract Prototype. *Constantine & Lockwood, Ltd.* [Online]

2. Ajax (programming) . *Wikipédia .* [Online]

3. Adobe. *Adobe.* [Online] 2009. http://www.adobe.com/.

4. JavaFX | RIA's JavaFX. *JavaFX.* [Online]

5. http://silverlight.net/. *Micorsoft Silverlight.* [Online] http://silverlight.net/.

6. Java - Sun MicroSystems. *Java.* [Online] http://www.java.com/pt_BR/download/index.jsp.

7. Hiperligação. *Wikipédia - a enciclopédia livre.* [Online] 2009. http://pt.wikipedia.org/wiki/Link.

8. Collaborative real-time editor. *Wikipédia - a enciclopédia livre.* [Online] 2009. http://en.wikipedia.org/wiki/Collaborative_real-time_editor.

9. Model View Controller. *ootips.* [Online] http://ootips.org/mvc-pattern.html.

10. Ruby On Rails. *Ruby On Rails.* [Online] http://rubyonrails.org/.

11. What is ASP.NET. [Online] 2009. http://www.ondotnet.com/pub/a/dotnet/2005/09/19/what-is-asp-et.html?page=7#new_features.

12. CakePHP. *CakePHP.* [Online] http://cakephp.org/.

13. World Wide Web. *Wikipédia.* [Online] http://pt.wikipedia.org/wiki/World_Wide_Web.

14. What is Cloud Computing. *searchCloudComputing.com.* [Online] http://searchcloudcomputing.techtarget.com/sDefinition/0,,sid201_gci1287881,00.html.

15. Google Application Engine - Google Code. *Google Code.* [Online] http://code.google.com/intl/pt-PT/appengine/.

16. Windows Azure. *Windows Azure Platform.* [Online] http://www.microsoft.com/windowsazure/windowsazure/.

17. Google Web Toolkit. *Google.* [Online] 2009. http://code.google.com/intl/pt-PT/webtoolkit/.

18. Google. *Google.* [Online] http://www.google.pt/.

19. GMail. *GMail - Email from Google.* [Online] http://mail.google.com/.

20. Google Calendar. *Google Calendar.* [Online] http://www.google.com/calendar/render.

21. Google Maps. *Google Maps.* [Online] http://maps.google.pt/.

22. What is Remote Procedure Call. *searchsoa.com.* [Online] http://searchsoa.techtarget.com/sDefinition/0,,sid26_gci214272,00.html.

23. Ext-GWT - Java Component Library. *Ext JS.* [Online] 2009. http://www.extjs.com/products/gxt/.

24. Internet Explorer 6. *Microsoft Internet Explorer.* [Online] http://www.microsoft.com/windows/ie/ie6/downloads/default.mspx.

25. Firefox. *Mozilla Europe.* [Online] http://www.mozilla-europe.org/pt/firefox/.

26. Safari 4.0.4. *baixaki.* [Online] http://www.baixaki.com.br/download/safari.htm.

27. Opera. *Opera Software.* [Online] http://www.opera.com/.

28. Google Chrome. *Google.* [Online] http://www.google.com/chrome.

29. Overview Using GWT. *Google Code.* [Online] 2009. http://code.google.com/p/google-web-toolkit-doc-1-5/wiki/OverviewUsingGWT.

30. ASP.NET Architecture (Illustration). *Directions on Microsoft.* [Online] 2009. http://www.directionsonmicrosoft.com/sample/DOMIS/update/2002/01jan/0102aidowa_illo1.htm.

31. Differences between ASP and ASP.NET. [Online] 2009. http://www.w3schools.com/aspnet/aspnet_vsasp.asp.

32. Flex for the Enterprise. [Online] 2009. http://www.authorstream.com/Presentation/Yuan-30944-ajaxworld-enterpriseflex-Flex-Architecture-Remoting-Data-Management-Messaging-Demo-Code-Advantages-Enterprise-Software-Requir-as-Entertainment-ppt-powerpoint/.

33. A sketch of Adobe Flex Architecture capabilities. [Online] 2009. http://www.pierocampanelli.info/technology/2008/01/20/a-sketch-of-adobe-flex-architecture-capabilities/.

34. GWTCanvas. *GWTCanvas.* [Online] http://code.google.com/p/google-web-toolkit-incubator/wiki/GWTCanvas.

35. GWT Graphics. *Vaadin.* [Online] http://vaadin.com/web/hene/wiki/-/wiki/Main/GWT+Graphics.

36. Ext GWT Help Center. *Ext GWT Help Center.* [Online] 2009. http://www.extjs.com/helpcenter/index.jsp?topic=/com.extjs.gxt.help/html/reference/dataloading.html.

37. Class Dispatcher (GXT API 2.0.1). *Ext JS.* [Online] 2009. http://www.extjs.com/deploy/gxtdocs/com/extjs/gxt/ui/client/mvc/Dispatcher.html.

38. google-web-toolkit-incubator. *Google code.* [Online] http://code.google.com/p/google-web-toolkit-incubator/.

39. **Júnior, Gilberto.** WebInsider. *WebInsider.* [Online] 2009. http://webinsider.uol.com.br/index.php/2006/05/20/google-web-toolkit-simplifica-desenvolvimento-ajax/.

40. Visão Geral do Produto. *Google Code.* [Online] 2009. http://code.google.com/intl/pt-BR/webtoolkit/overview.html.

41. Introdução ao GWT. *GWT.* [Online] 2009. http://www.gwt.com.br/2008/09/11/introducao-ao-gwt/#more-66.

42. Google Web Toolkit. *Google.* [Online] 2009. http://code.google.com/intl/pt-PT/webtoolkit/.

43. Visual Web Developer. [Online] 2009. http://www.microsoft.com/express/vwd/.

44. Tips and Tricks for ASP.NET. [Online] 2009. http://blogs.msdn.com/webdevelopertips/default.aspx .

45. **Busoli, Simone.** ASP.NET Internals – IIS and the Process Model. [Online] 2009. http://dotnetslackers.com/articles/iis/ASPNETInternalsIISAndTheProcessModel.aspx .

46. **Santos, Pedro.** Software de Qualidade. [Online] 2009. http://psantos.zi-yu.com/wiki.aspx?topic=PRE.SoftwareDeQualidade .

47. Silverlight Architecture. [Online] 2009. http://www.vectorlight.net/about/architecture.aspx .

48. **Moroney, Laurence.** Introdução ao Silverlight. [Online] 2009. http://msdn.microsoft.com/pt-br/library/cc580591.aspx.

49. **Uchôa, Juliana Prado.** Tutorial - Introdução ao Silverlight 2. [Online] 2009. http://infowd.blogspot.com/2008/09/tutorial-introduo-ao-silverlight-2-ol.html.

50. **Santos, Lauro.** Flex Builder 3 Beta disponível. [Online] 2009. http://www.laurosantos.com.br/blog/flex-builder-3-beta-publciodisponivel-para-download/.

51. Beginners Adobe Flex Tutorial. [Online] 2009. http://learnola.com/2008/12/06/beginners-adobe-flex-tutorial/.

52. Ext-GWT 2.0.1 Explorer. *EXT JS.* [Online] 2009. http://www.extjs.com/examples/explorer.html#overview.

53. com.extjs.gxt.ui.client.data (GXT API 2.0.1). *Ext JS.* [Online] 2009. http://www.extjs.com/deploy/gxtdocs/com/extjs/gxt/ui/client/data/package-summary.html.

54. Evolução Arquitectura Cliente/Servidor. *Knoll.* [Online] http://knol.google.com/k/danilo-rodrigues-da-silva/evolu%C3%A7%C3%A3o-da-arquitetura-cliente-servidor/anrd8mpq3bji/3#.

55. Making Remote Procedure Calls. *Google code.* [Online] Google , 2009. http://code.google.com/intl/pt-PT/webtoolkit/tutorials/1.6/RPC.html.

56. Interaction Styles. *Interaction Design.* [Online] 2009. http://www.interaction-design.org/encyclopedia/interaction_styles.html.

57. LetterPop. *LetterPop.* [Online] 2009. http://letterpop.com/.

58. Link Consolidation. *SEOmozBlog.* [Online] 2009. http://www.seomoz.org/blog/link-consolidation-the-new-pagerank-sculpting.

59. Microsoft PressPass. [Online] 2009. http://www.microsoft.com/presspass/press/2007/apr07/04-15WPFEPR.mspx.

60. MVC. *Wikipédia - a enciclopédia livre.* [Online] http://pt.wikipedia.org/wiki/MVC.

61. HowStuffWroks - "How Cloud Computing Works". *HowStuffWroks.* [Online] 2009. http://communication.howstuffworks.com/cloud-computing.htm.

62. Animation, Multimédia | Adobe Flash CS4. *Adobe.* [Online]

63. Altova UModel - UML Tool. *Altova.* [Online] http://www.altova.com/umodel.html.

64. Adobe Business Catalyst. *Adobe.* [Online] http://businesscatalyst.com/.

65. Online MindMeister and Brainstorming. *MindMeister.* [Online] http://www.mindmeister.com/.

66. Model View Controller Diagram. *Wikipédia.* [Online] http://pt.wikipedia.org/wiki/Ficheiro:ModelViewControllerDiagram.svg.

# 6 Anexo 1 – Estudo das Interfaces e Estilos de Interacção em Editores de Modelos

## Microsoft Visio 2007 - aplicação Desktop destinada à edição de modelos UML.

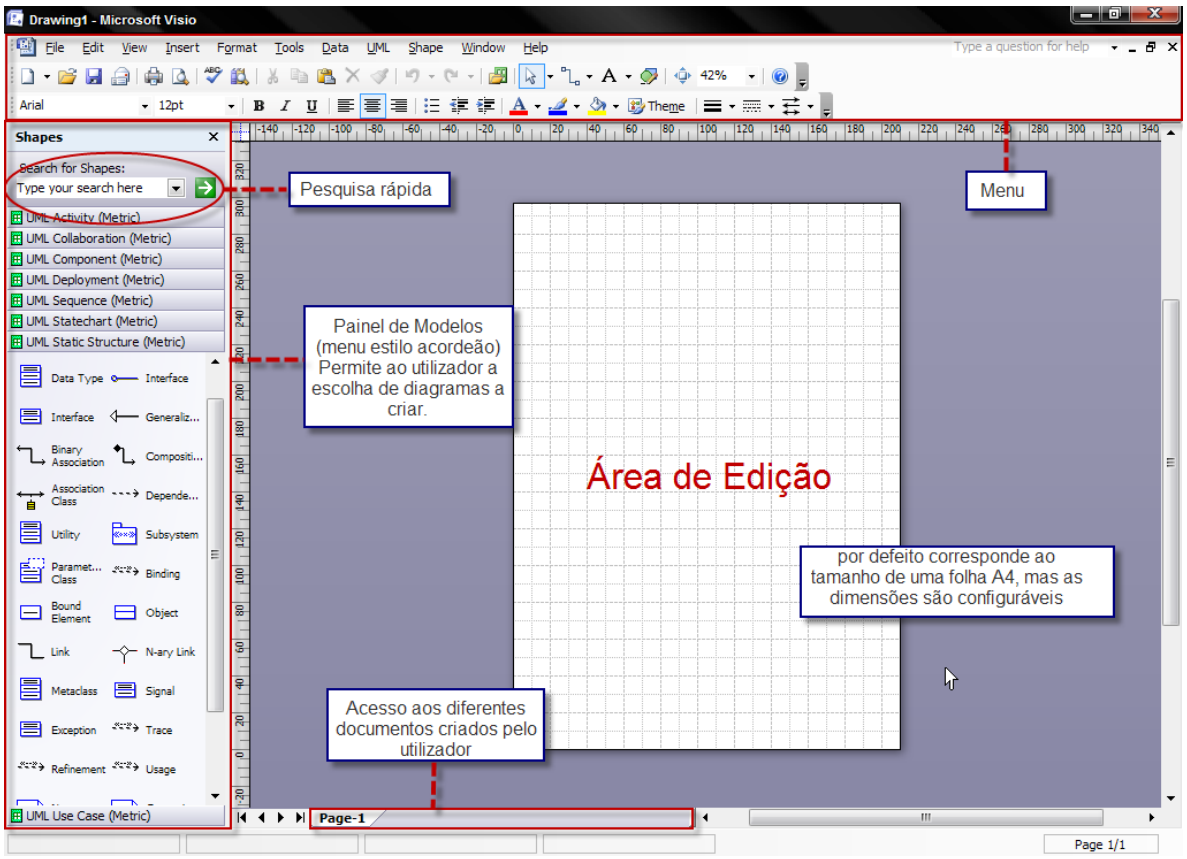


Figura 49 – Microsoft Visio, Estilos de Interacção

Sendo o Microsotf Visio uma aplicação Desktop, possui na zona superior um conjunto de menus que permitem executar variado número de tarefas. Para a edição dos modelos, possui à esquerda o menu que dá acesso ao elementos a serem editados e, ocupando o maior espaço na interface da aplicação, encontra-se a zona de edição, também desigando de canvas, onde os elementos são manipulados.

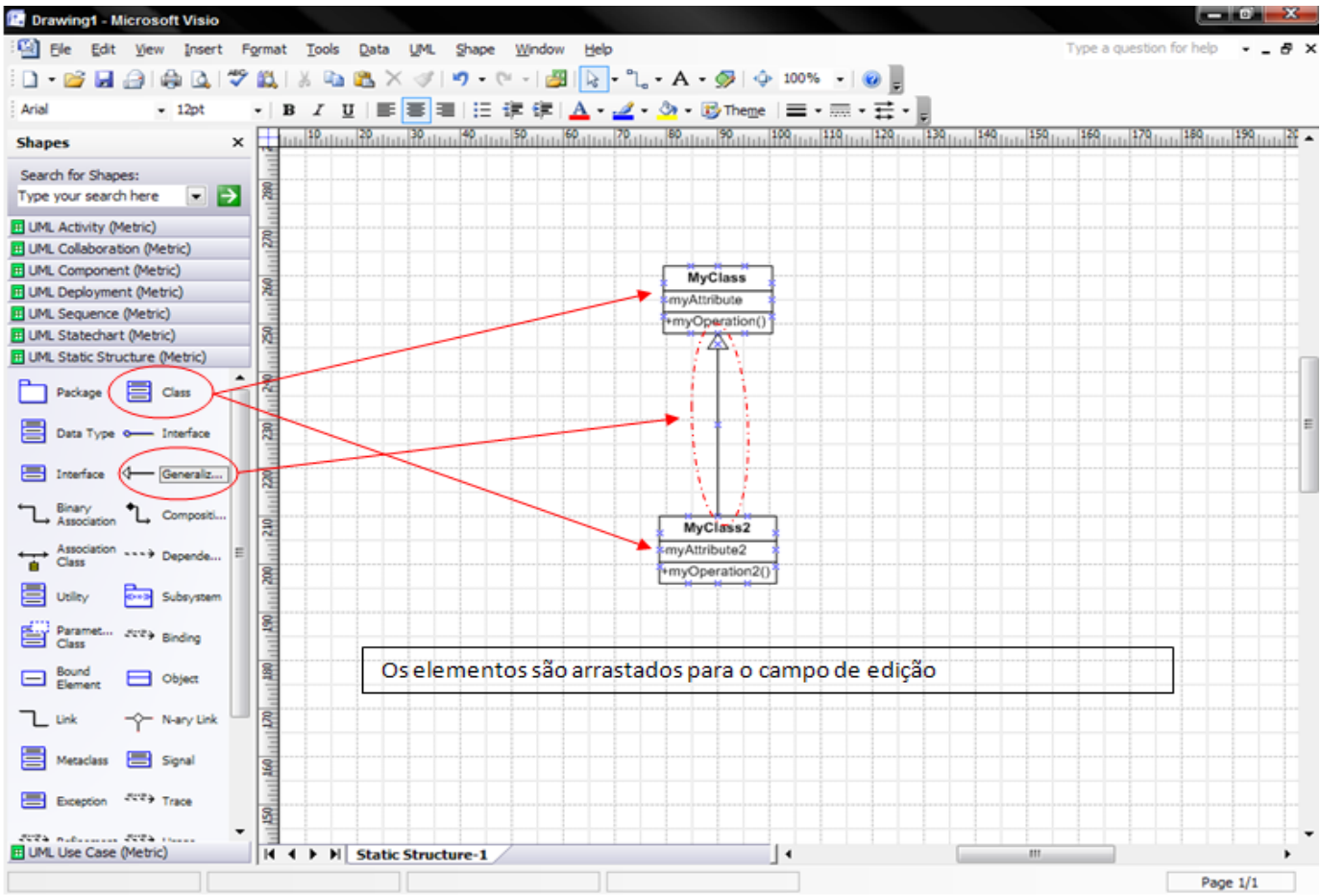


Figura 50 – Microsoft Visio – arrastando elementos para o Canvas

Para a manipulação dos elementos no canvas, o utilizador pode clicar e arrastar os elementos. A utilização de um "slide menu" permite que sejam apresentados um grande conjunto de modelos em um reduzido espaço na interface da aplicação.

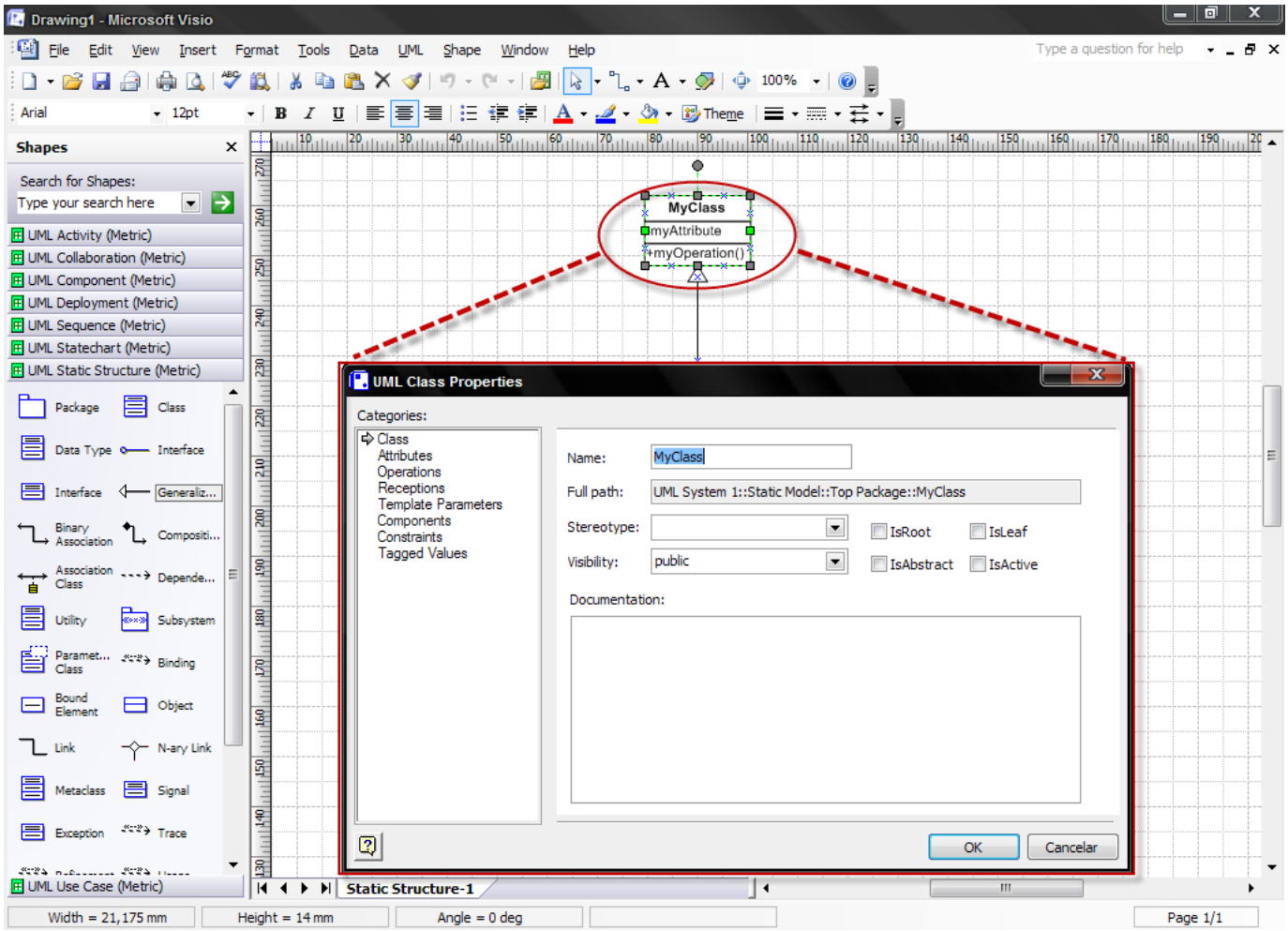


Figura 51 – Microsoft Visio – editando elementos no canvas

A edição das propriedades dos elementos no canvas é feita em janelas independentes (janelas popup) não permitindo que tais dados sejam directamente editados sobre o elemento no canvas.

## Gliffy – Aplicação Web para a edição de Modelos

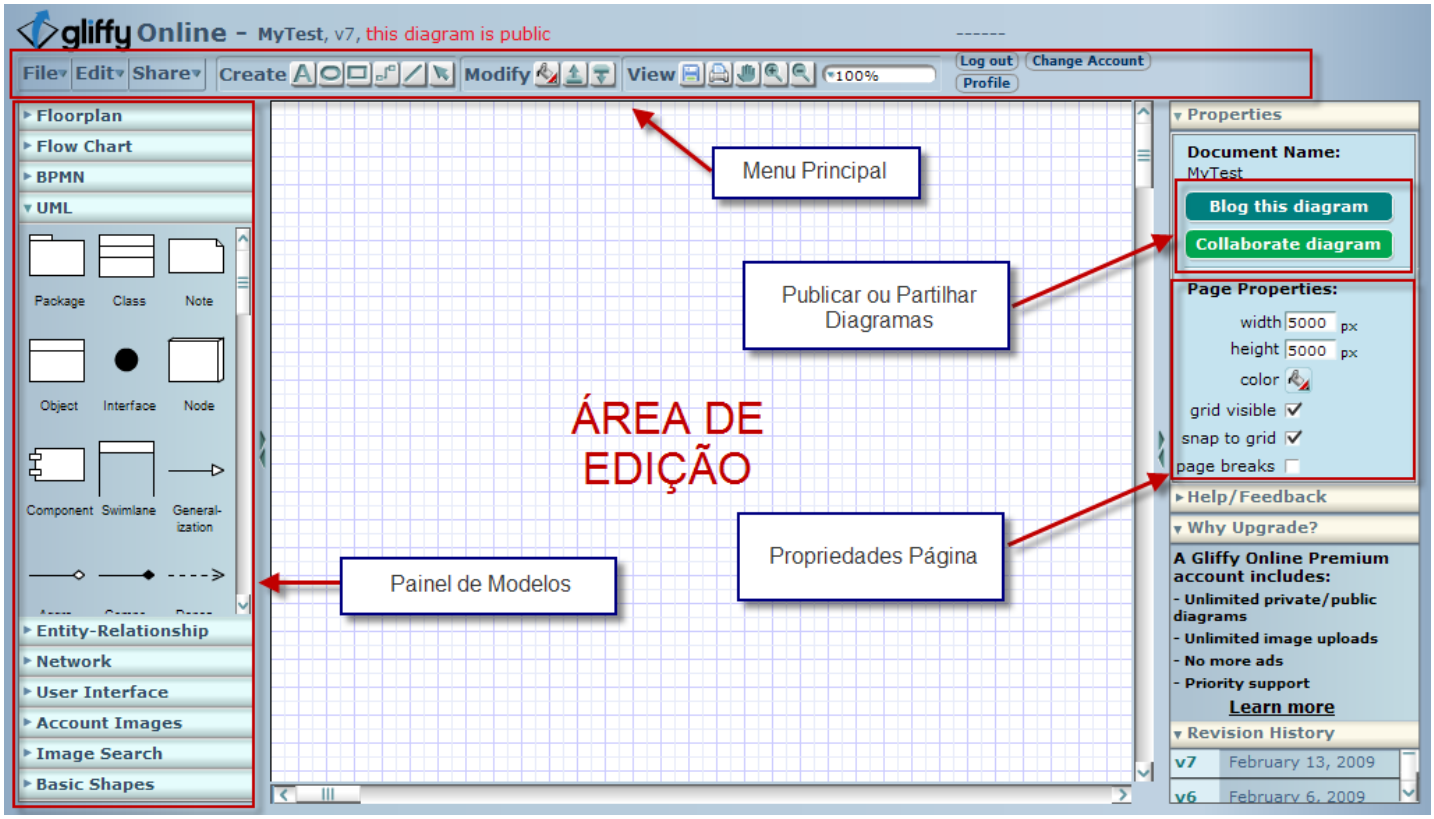


Figura 52 – Gliffy – Área de edição

O Gliffy é uma aplicação muito semelhante ao já apresentado Microsoft Visio, no entanto esta é uma aplicação disponível na Web. Estando sobre a Web, é uma aplicação mais simplificada comparativamente aos seus semelhantes sobre o ambiente Desktop, mas que permite basicamente os mesmos estilos de interacção.

Os elementos são arrastados e manipulados sobre o canvas, estando as suas propriedades disponíveis numa pequena janela à direita da aplicação, permitindo ao utilizador o fácil acesso e visualização das mesmas.

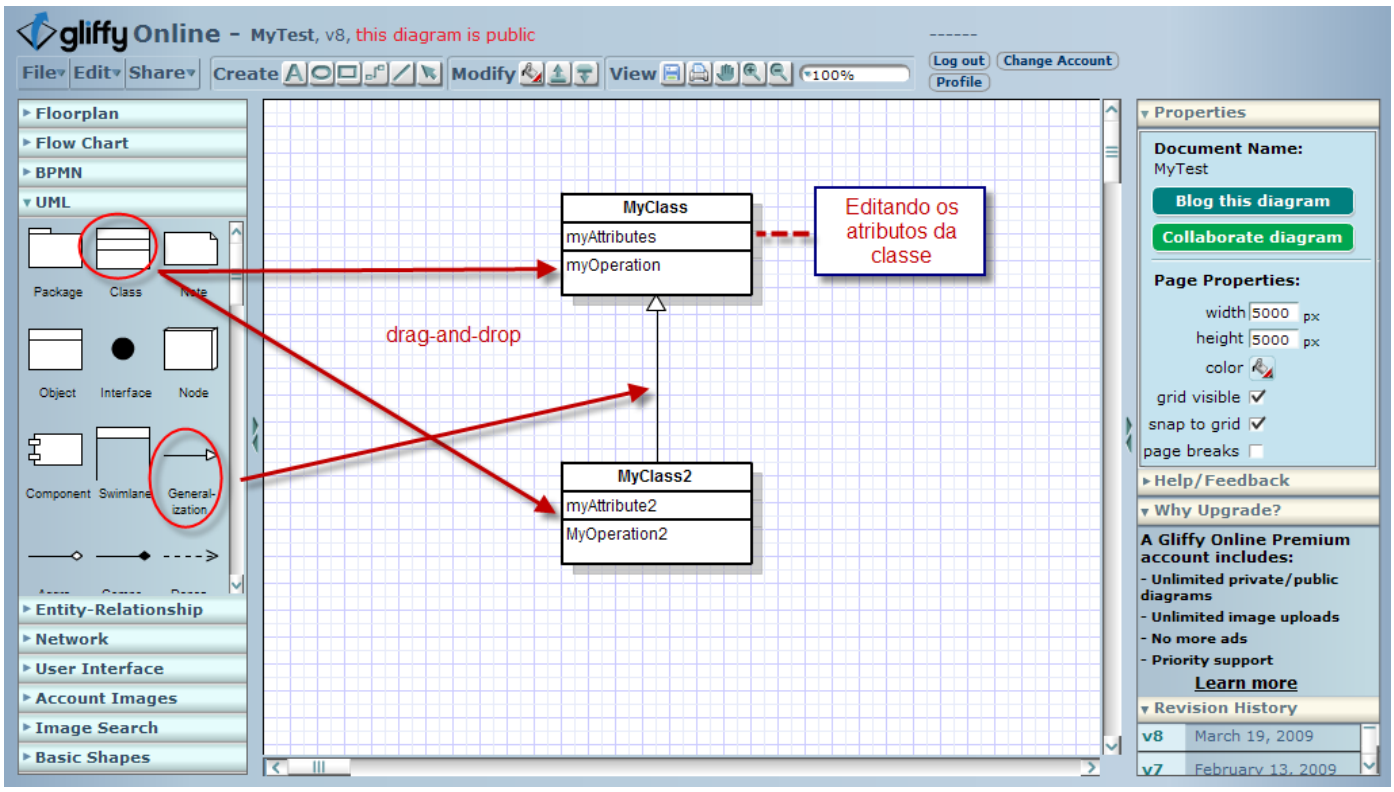


Figura 53 – Gliffy – Arrastando elementos para o canvas

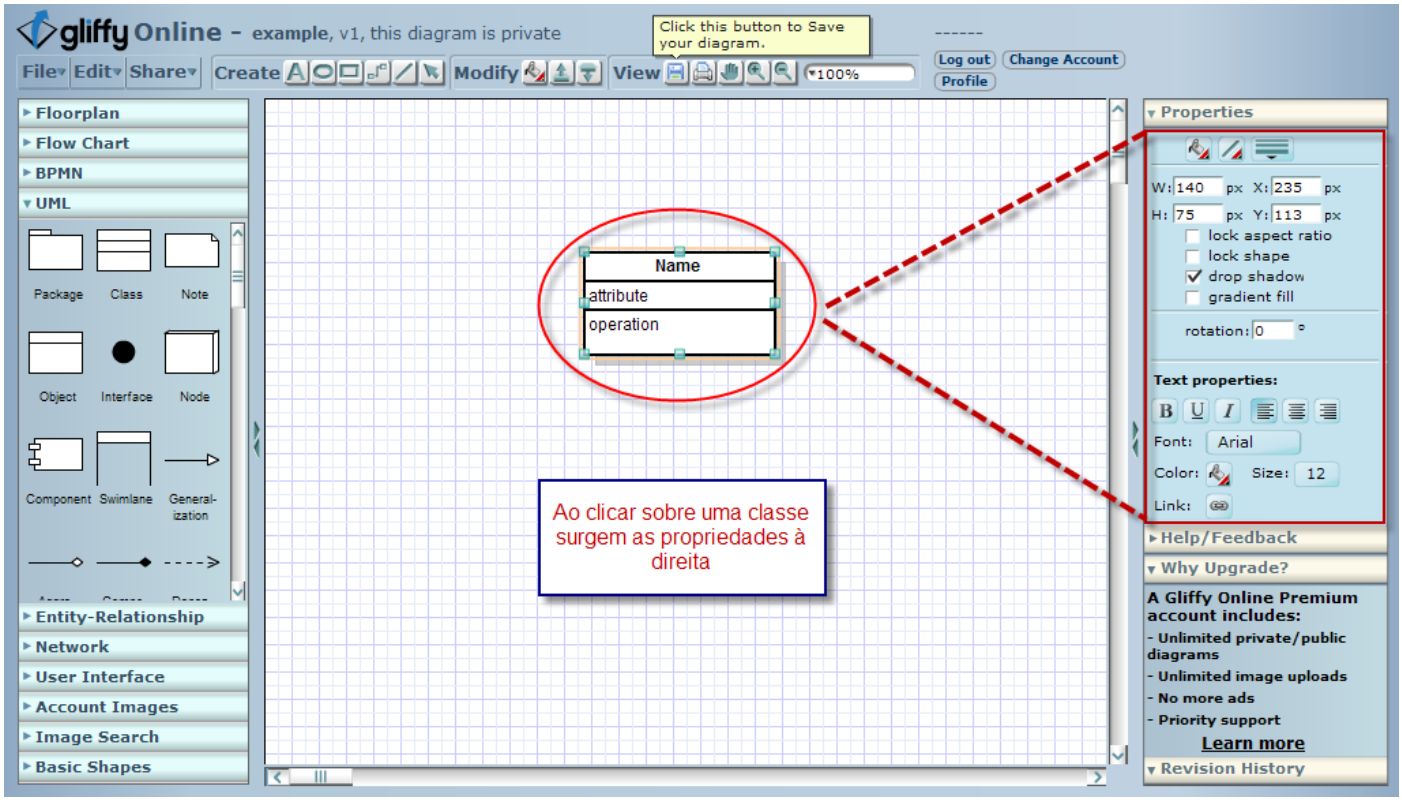


Figura 55 – Gliffy – Editando elementos

## GModeler – aplicação web que permite a criação de diagramas UML.

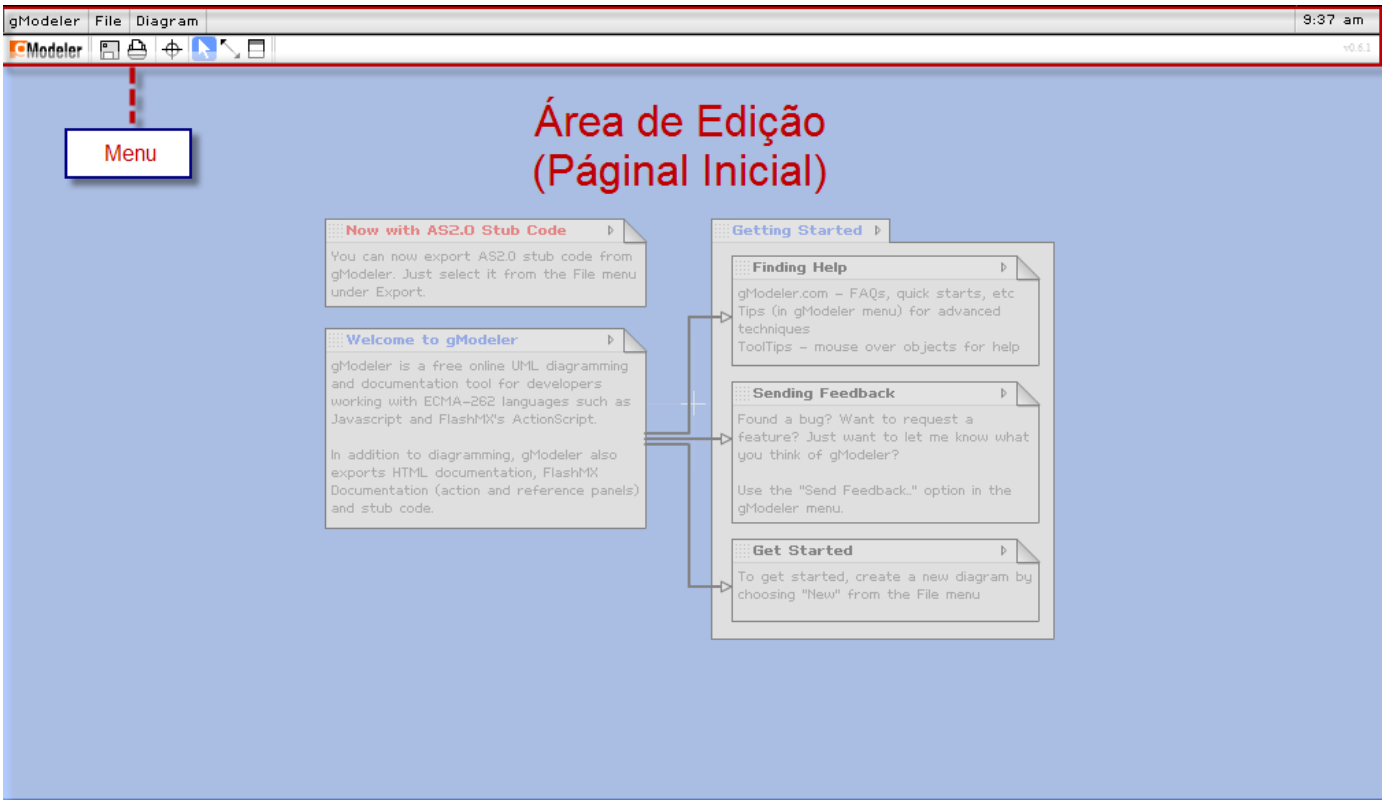


Figura 56 – Gmodeler - Menu

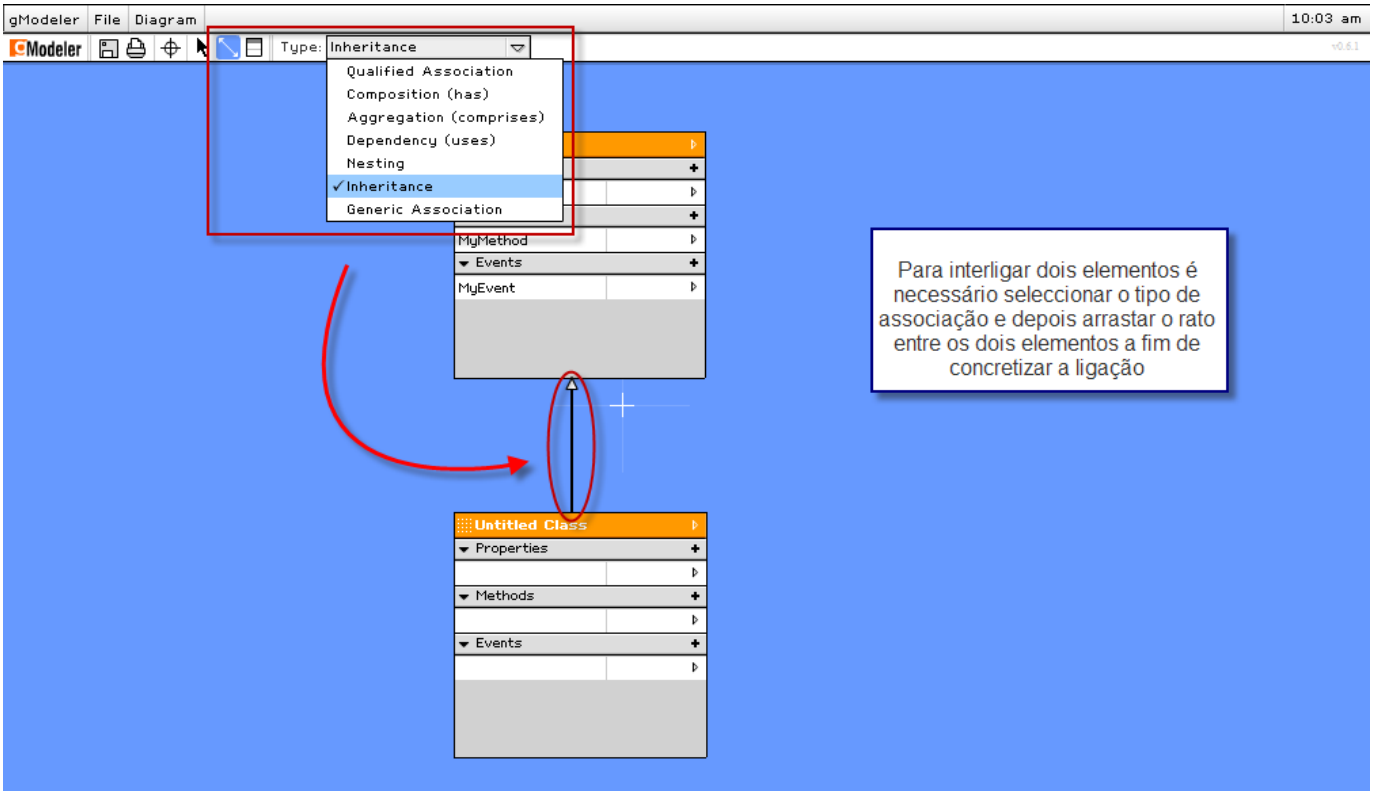


Figura 57 – Gmodeler – Arratando objectos para o canvas

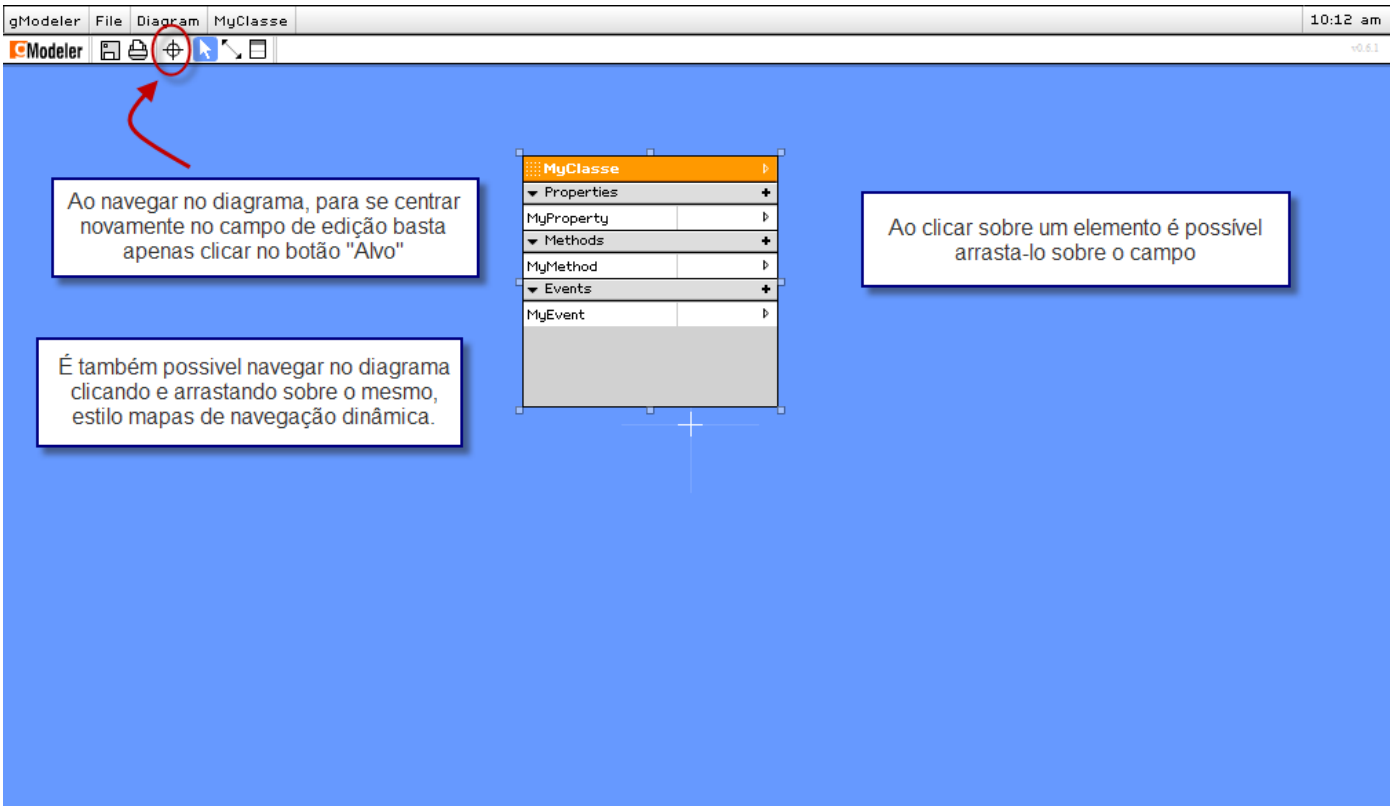


Figura 58 – Gmodeler – manipulação de objectos no canvas

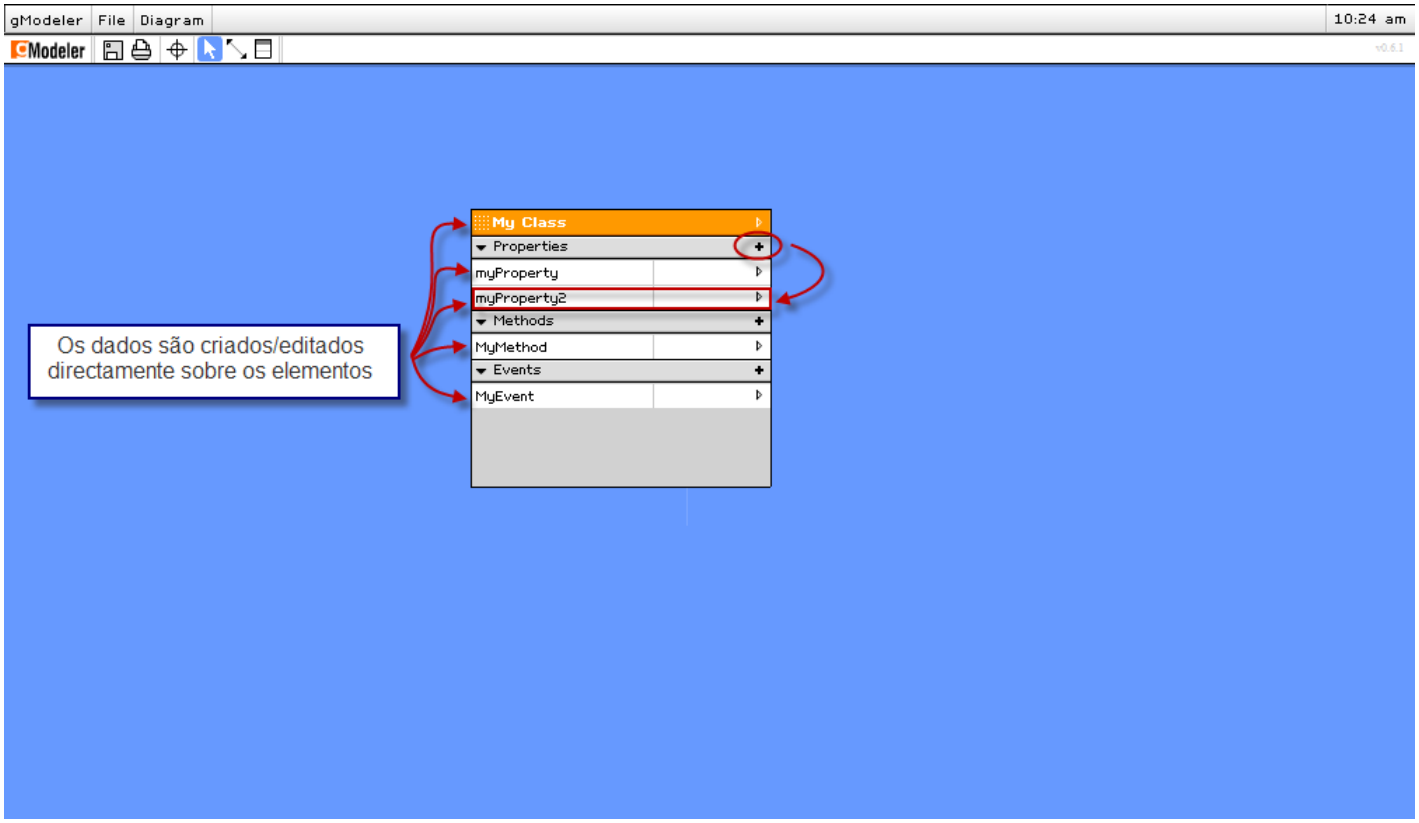


Figura 59 – Gmodeler – Edição de elementos no canvas

## MindMeister

Aplicação Web que permite a  criação de mapas mentais (brainstorming) permitindo a gestão e a partilha dos mesmos.

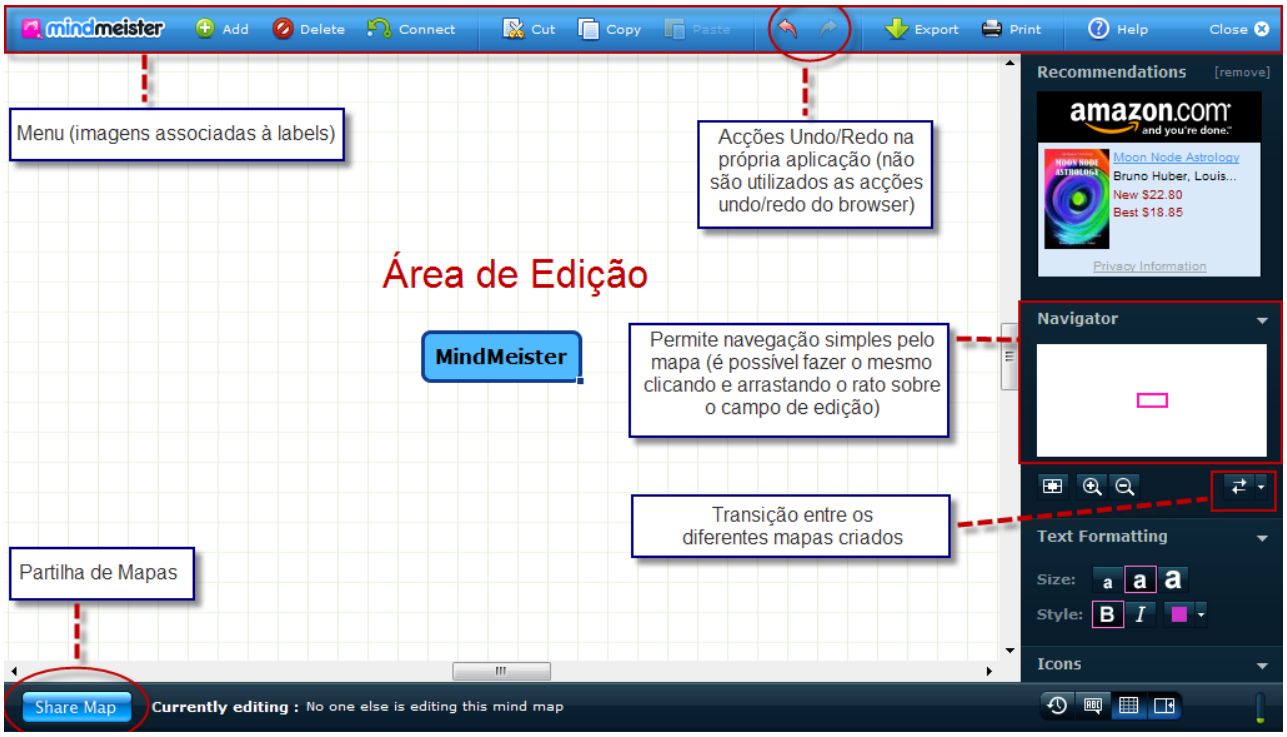


Figura 60 – MinMeister – Área de Edição

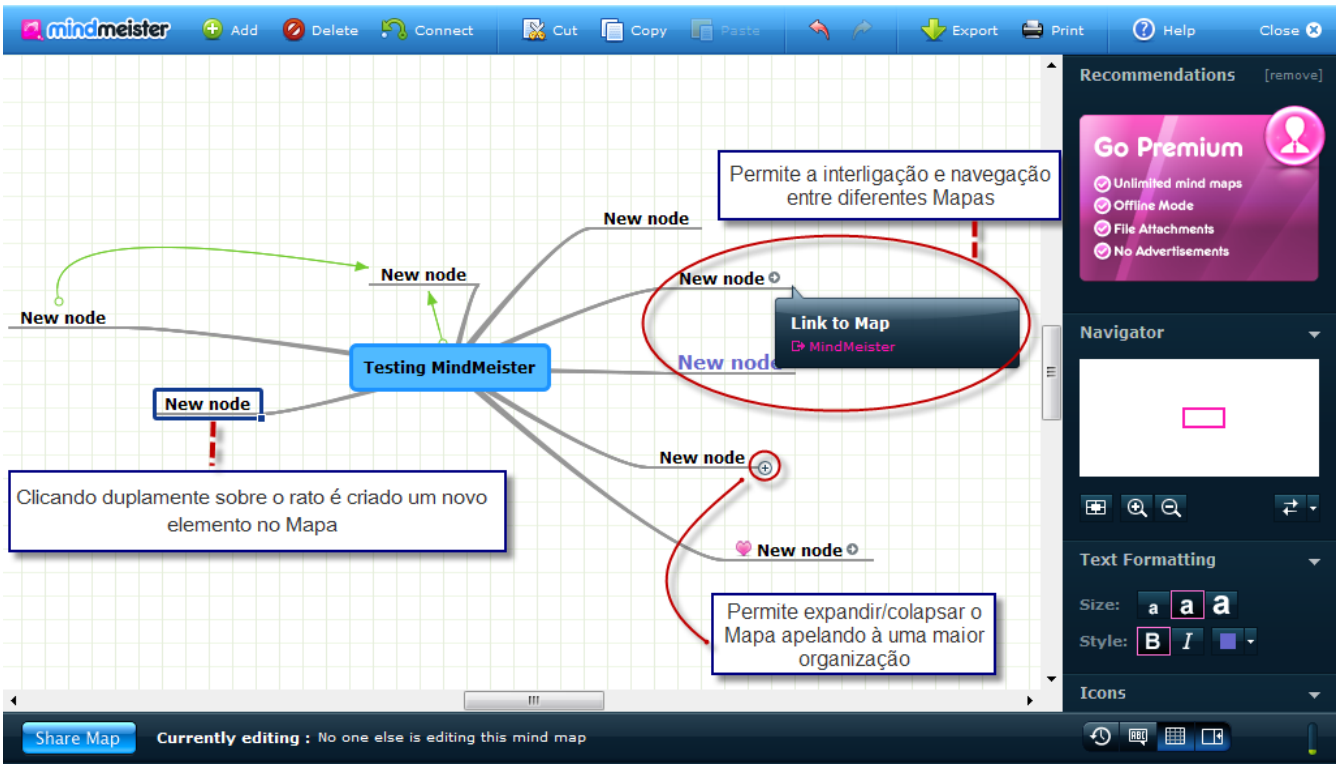


Figura 61 – MindMeister – Manipulação de objectos

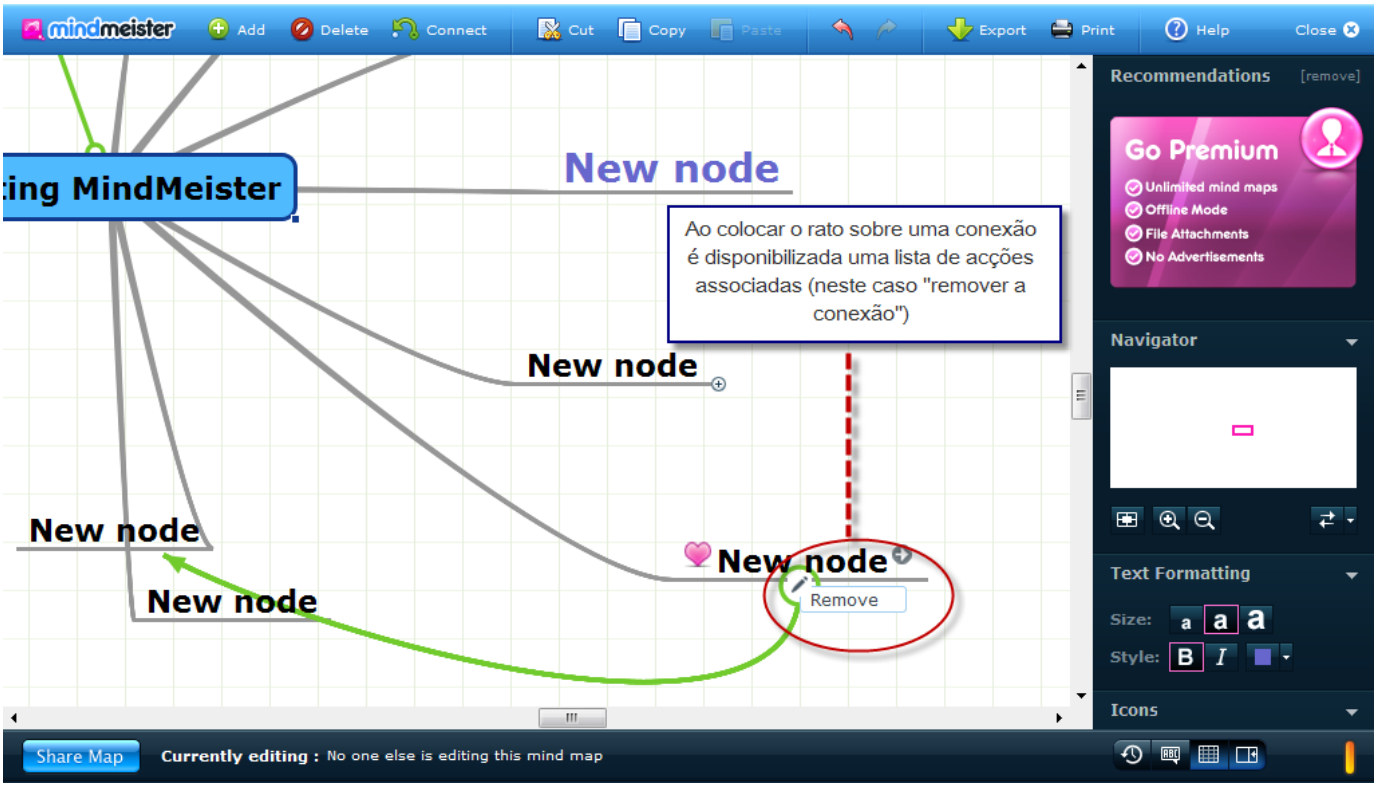


Figura 62 – MindMeister – Manipulação de objectos

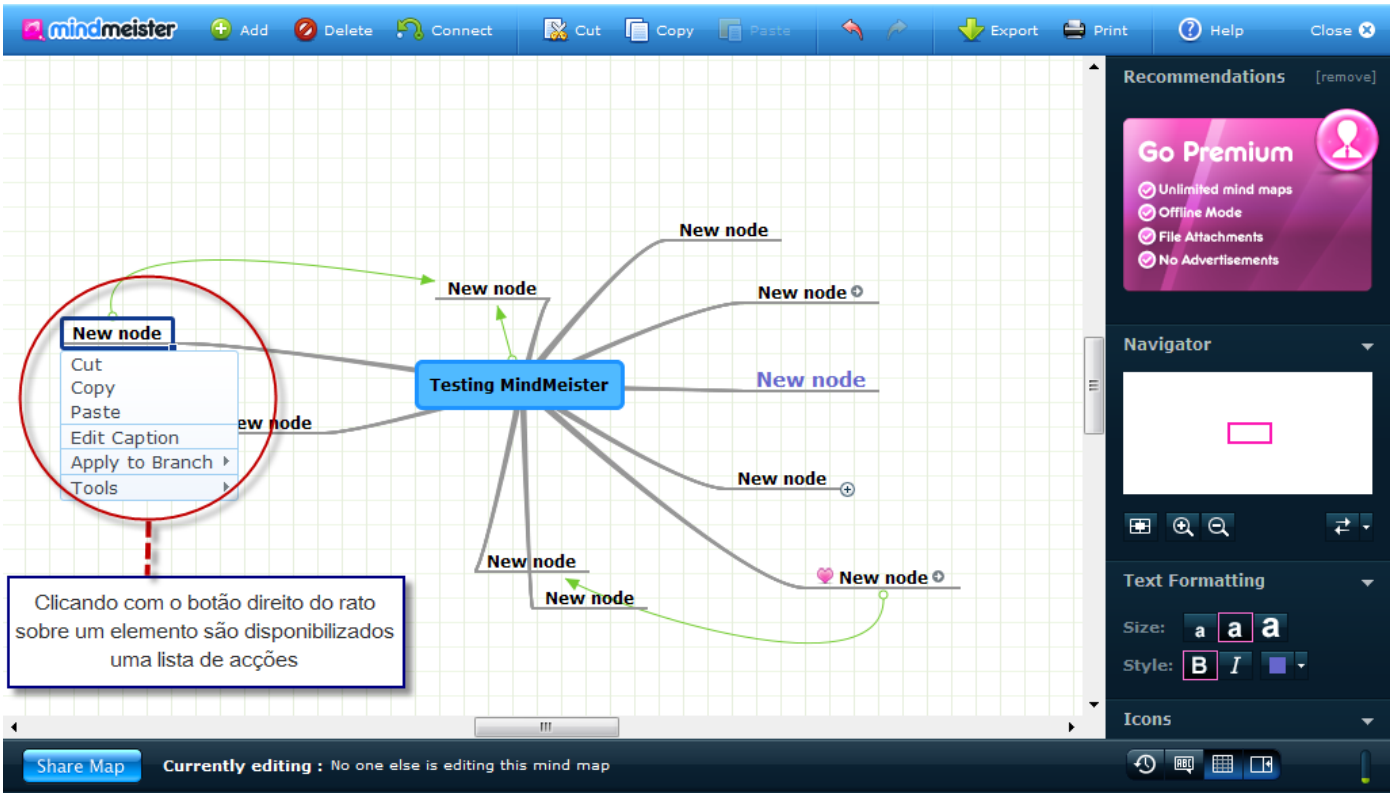


Figura 63 – MindMeister – Edição de elementos no canvas

O MindMeister é uma aplicação bastante intuitiva, permitindo que muitas das operações sobre os elementos sejam feitos no próprio canvas.

## Google Docs

Aplicação Web que permite a criação e partilha de documentos


Figura 64 – GoogleDocs

Editando um documento:

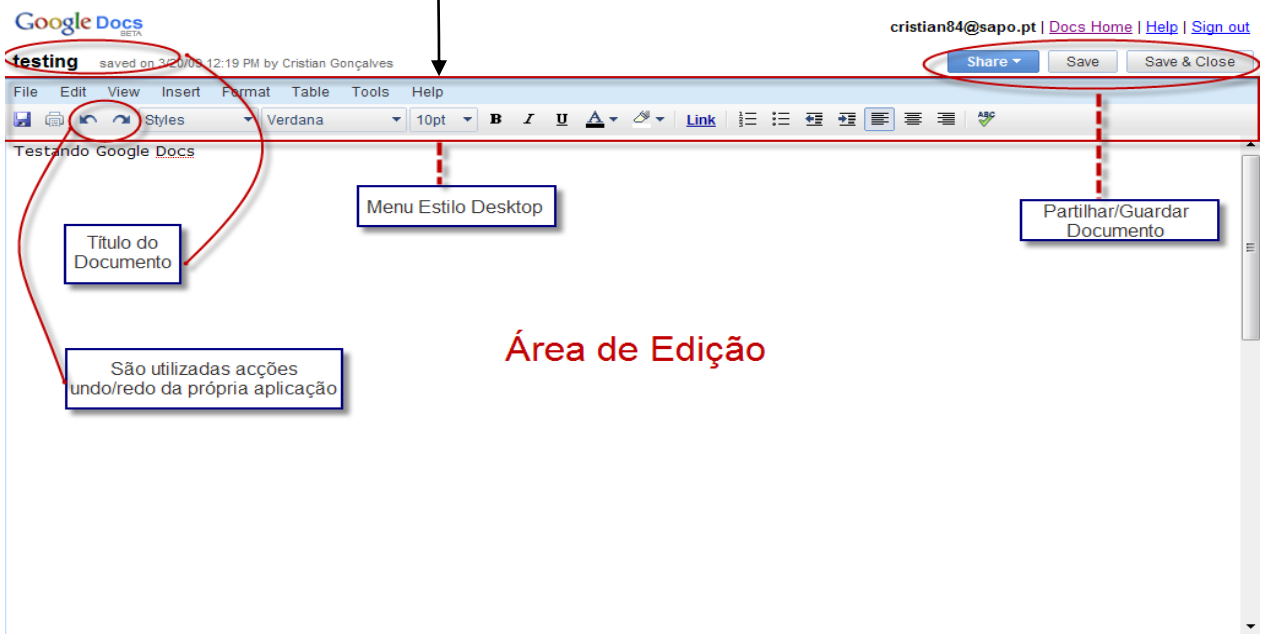


Figura 65 – GoogleDocs – Área de Edição

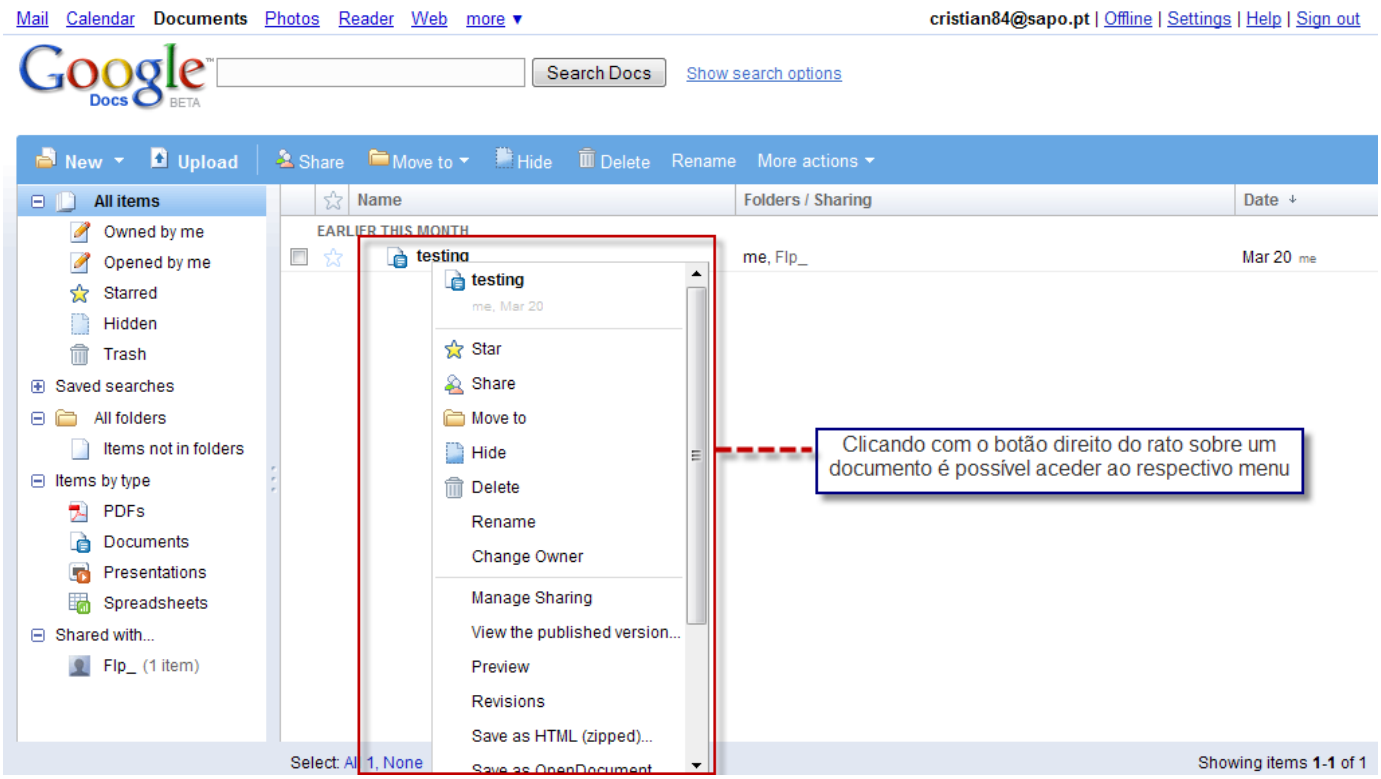


Figura 66 – GoogleDocs - Menu

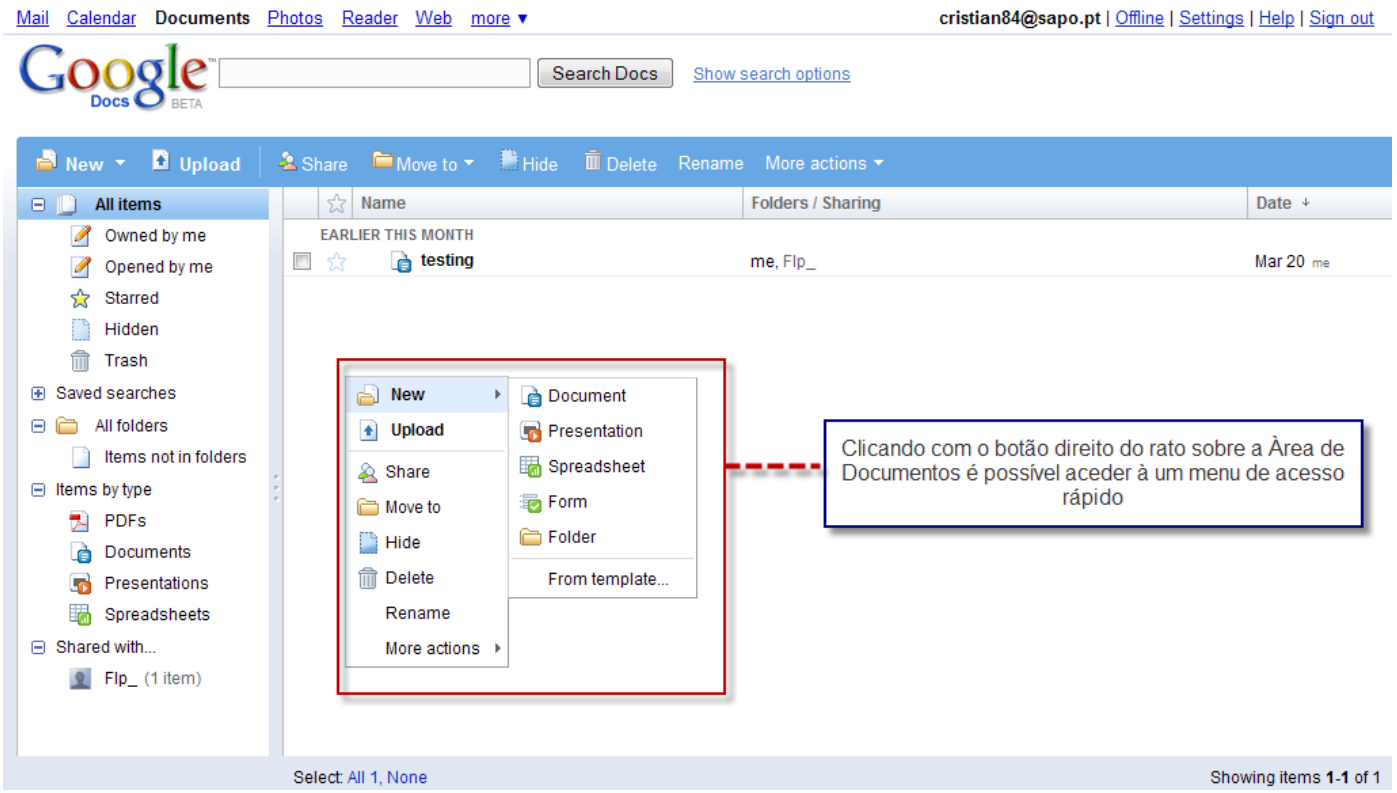


Figura 67 – Google Docs – Estilos de interacção

## SumoPaint - Editor de Imagem sobre a Web

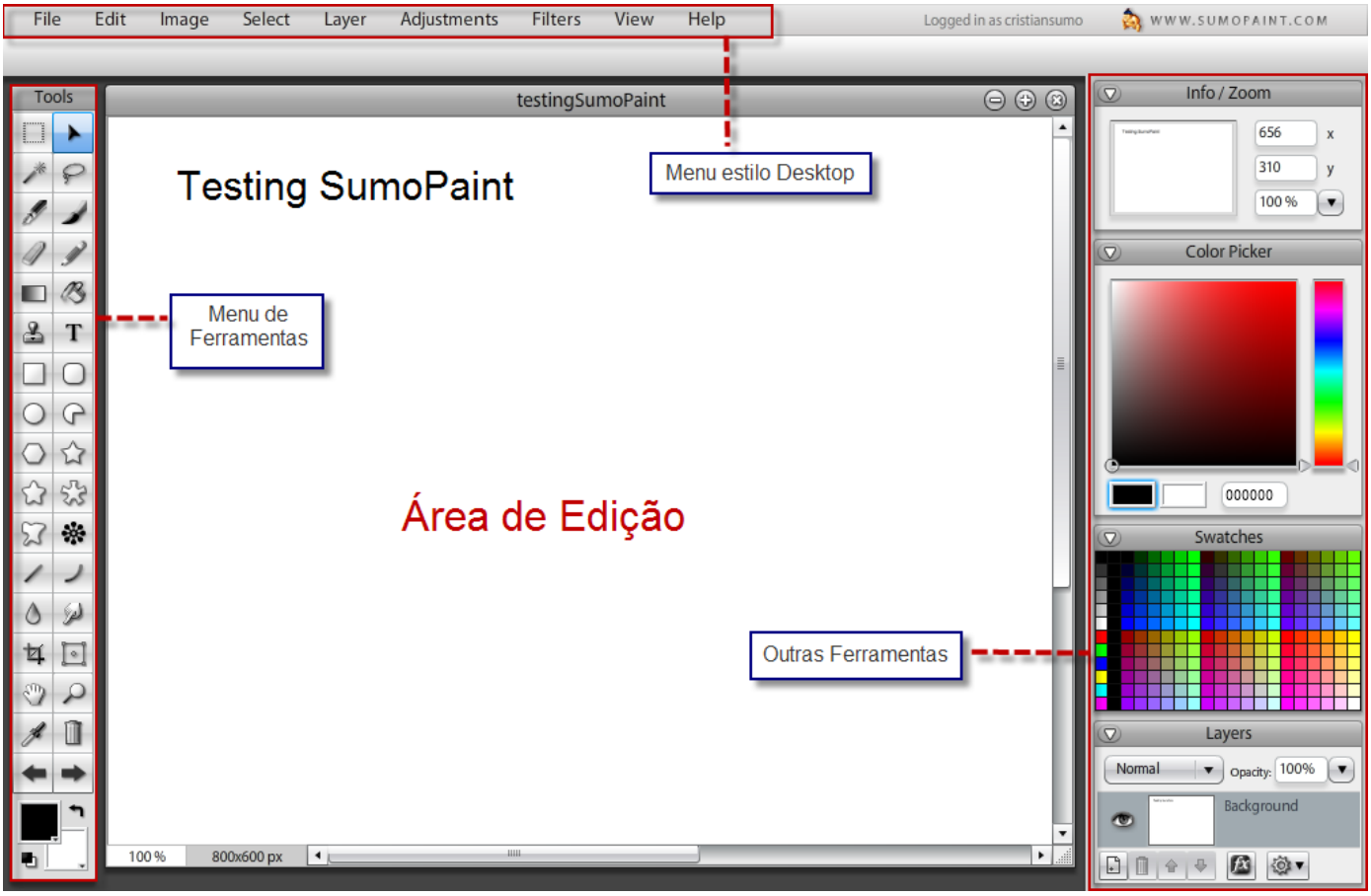


Figura 68 – SumoPaint – Área de Edição de elementos

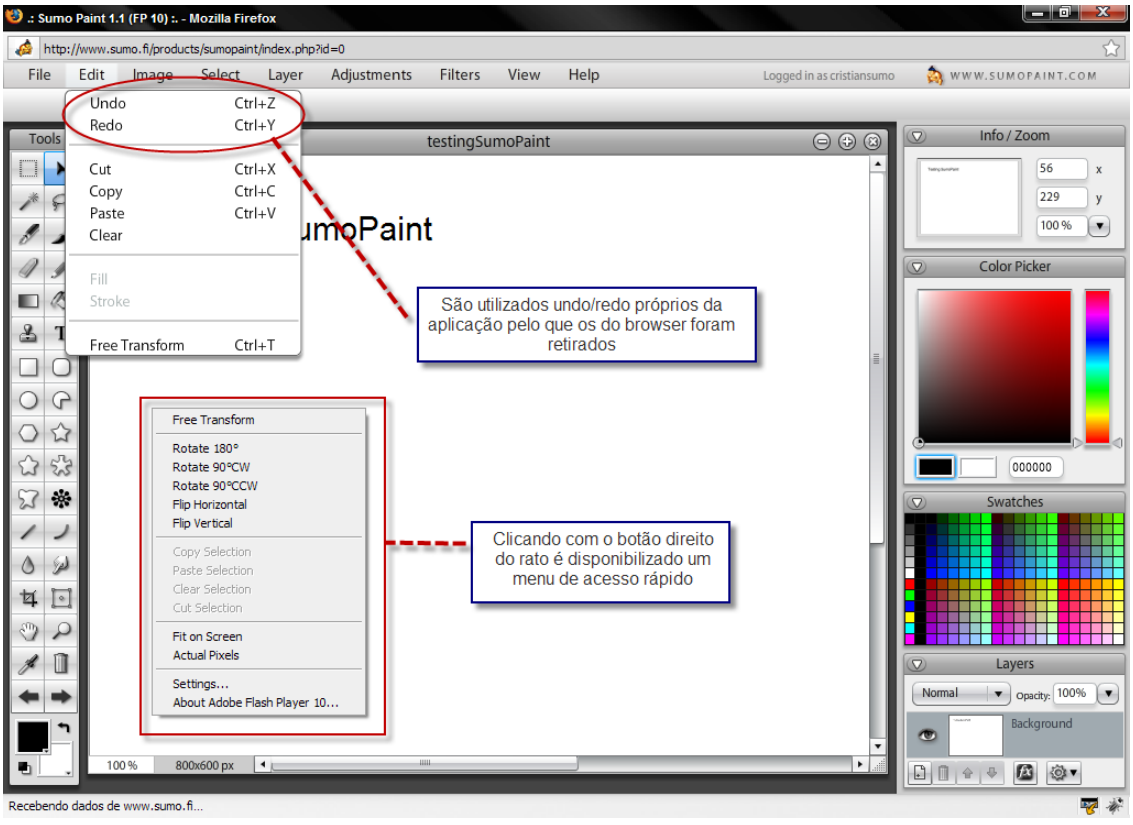


Figura 69 – SumoPaint – Estilos de interacção

## Tutor.com ClassRoom


Figura 70 - TutorClassRoom – Estilos de Interacção

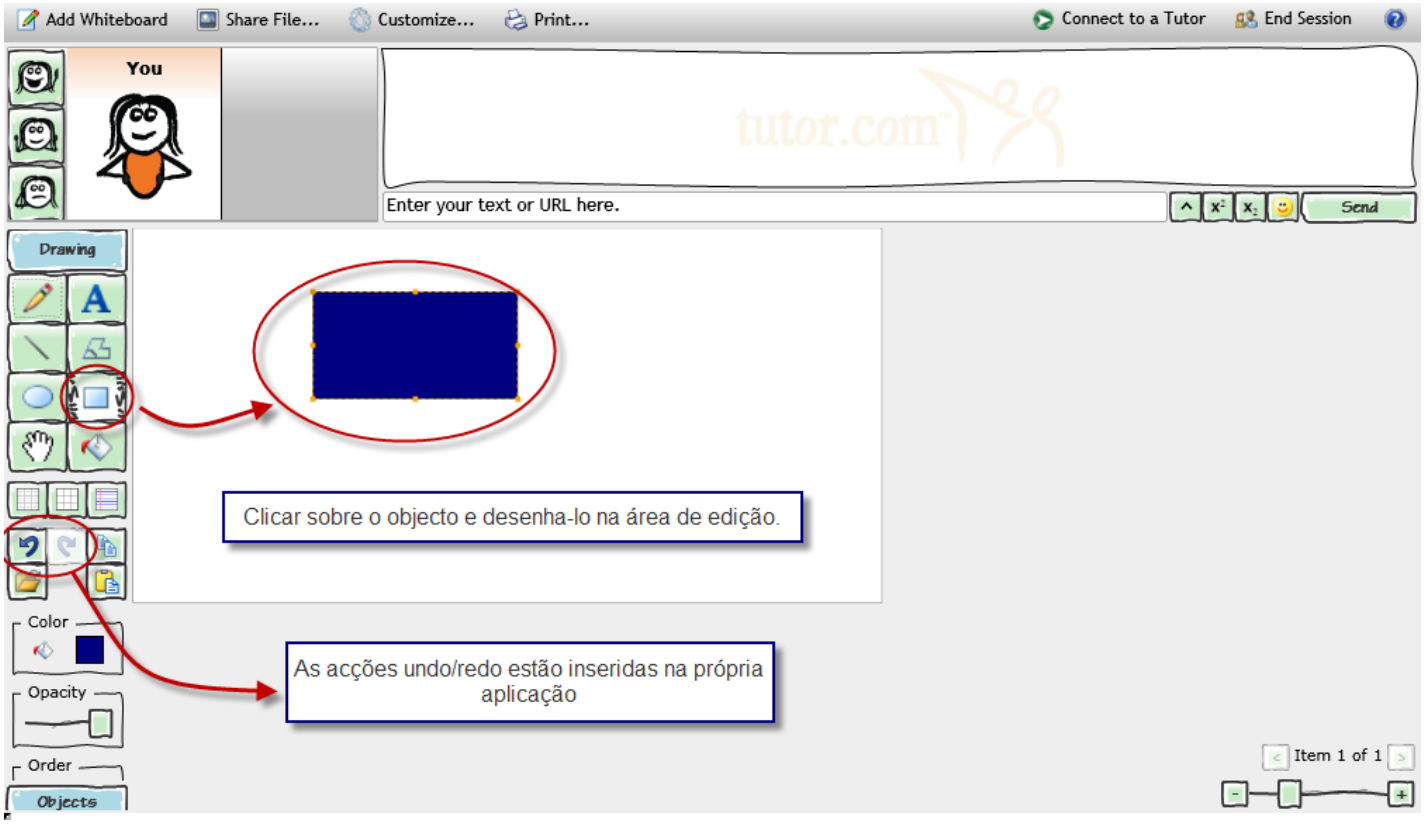


Figura 71 - TutorClassRoom – Estilos de Interacção

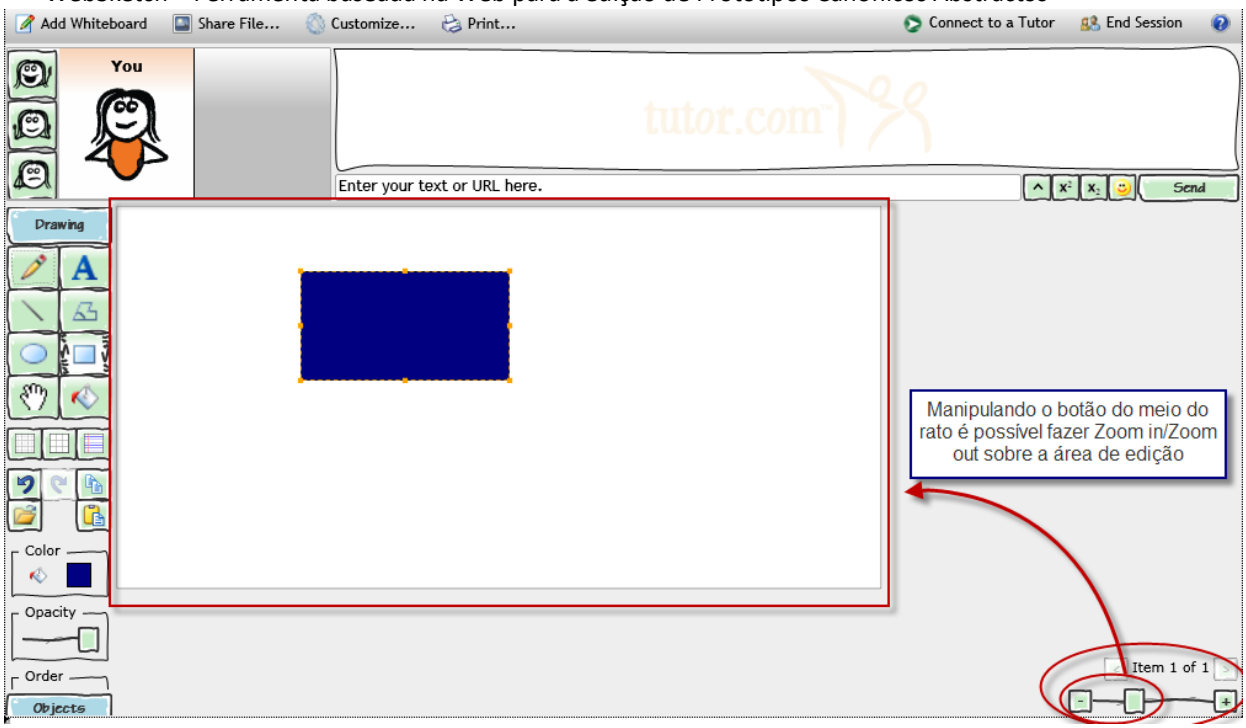


Figura 72 – TutorClassRoom – Zoom in

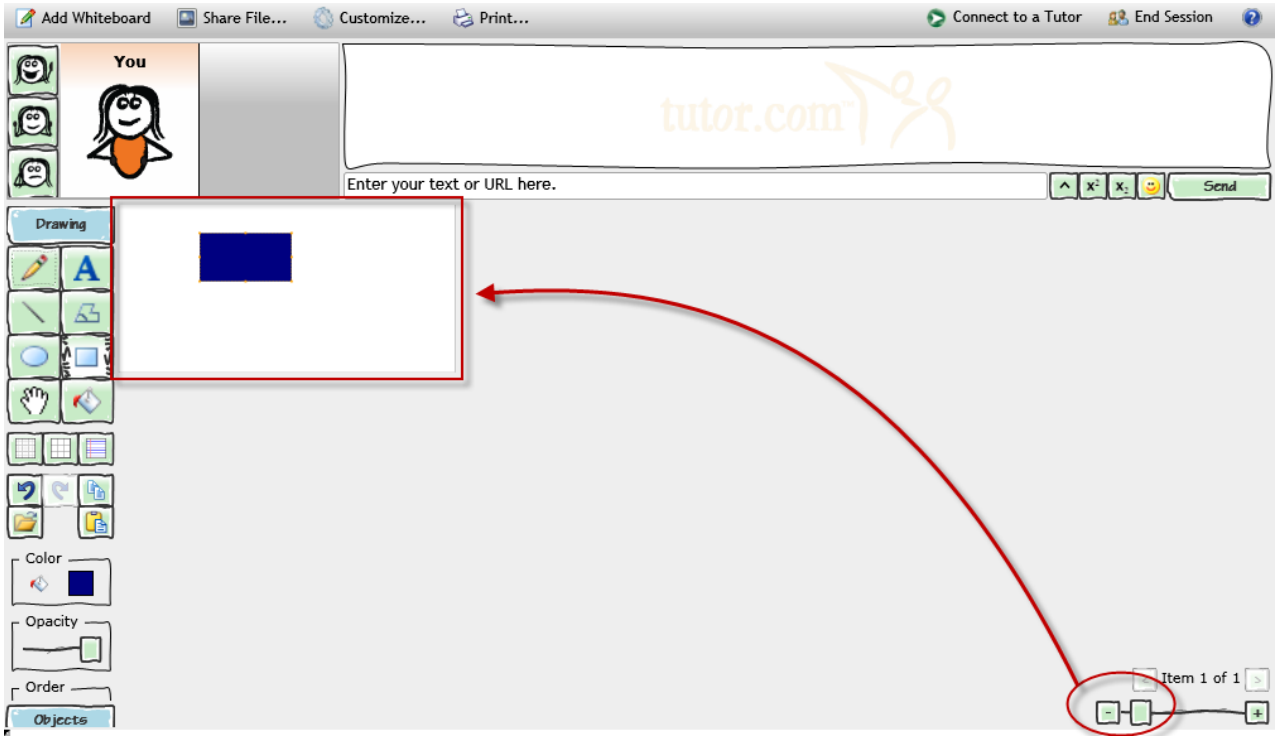


Figura 73 – TutorClassRoom – Zoom in (contt.)

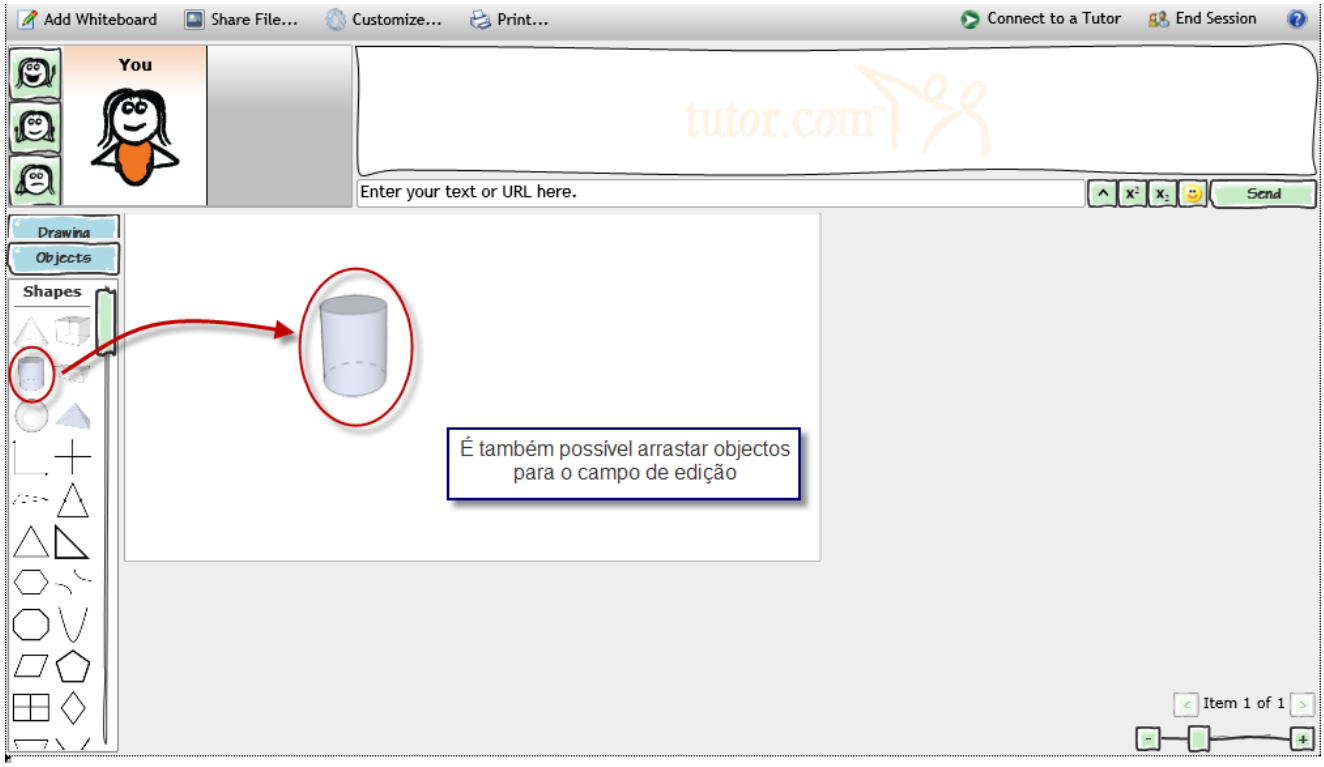


Figura 74 – TutorClass – Arrastando objectos para o canvas

## SmartDraw – Aplicação Desktop para a Edição de Modelos

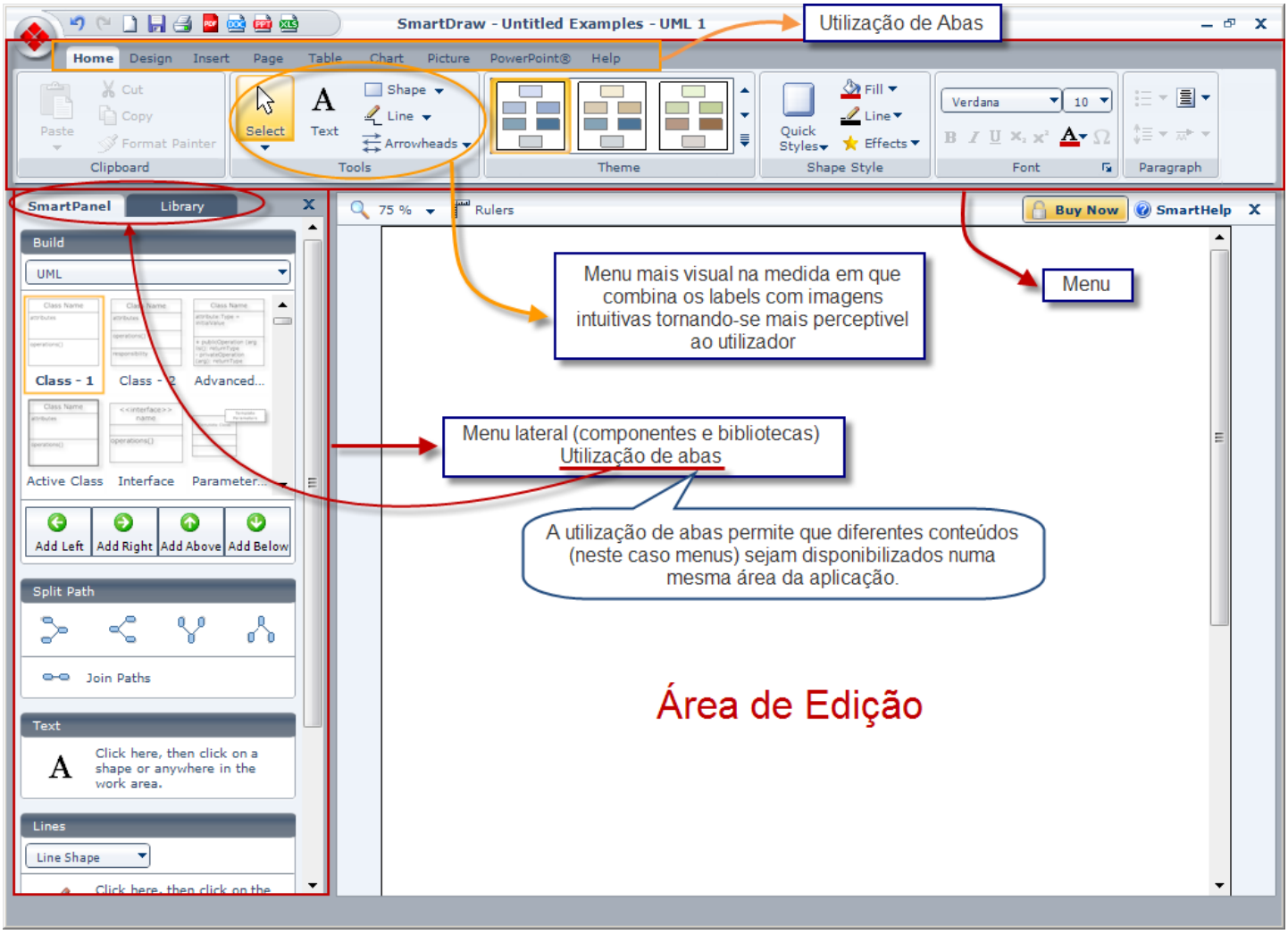


Figura 75 – SmartDraw – Layout e estilos de interacção

O SmartDraw diferencia-se dos outros editores, na medida em que o menu superior apresenta uma nova imagem, onde as funcionalidades são representadas por ícones, tornando-se assim mais intuitivo ao utilizador, e as funcionalidades são separadas por abas, permitindo que muitas funcionalidades sejam apresentadas em um mesmo espaço na interface da aplicação.

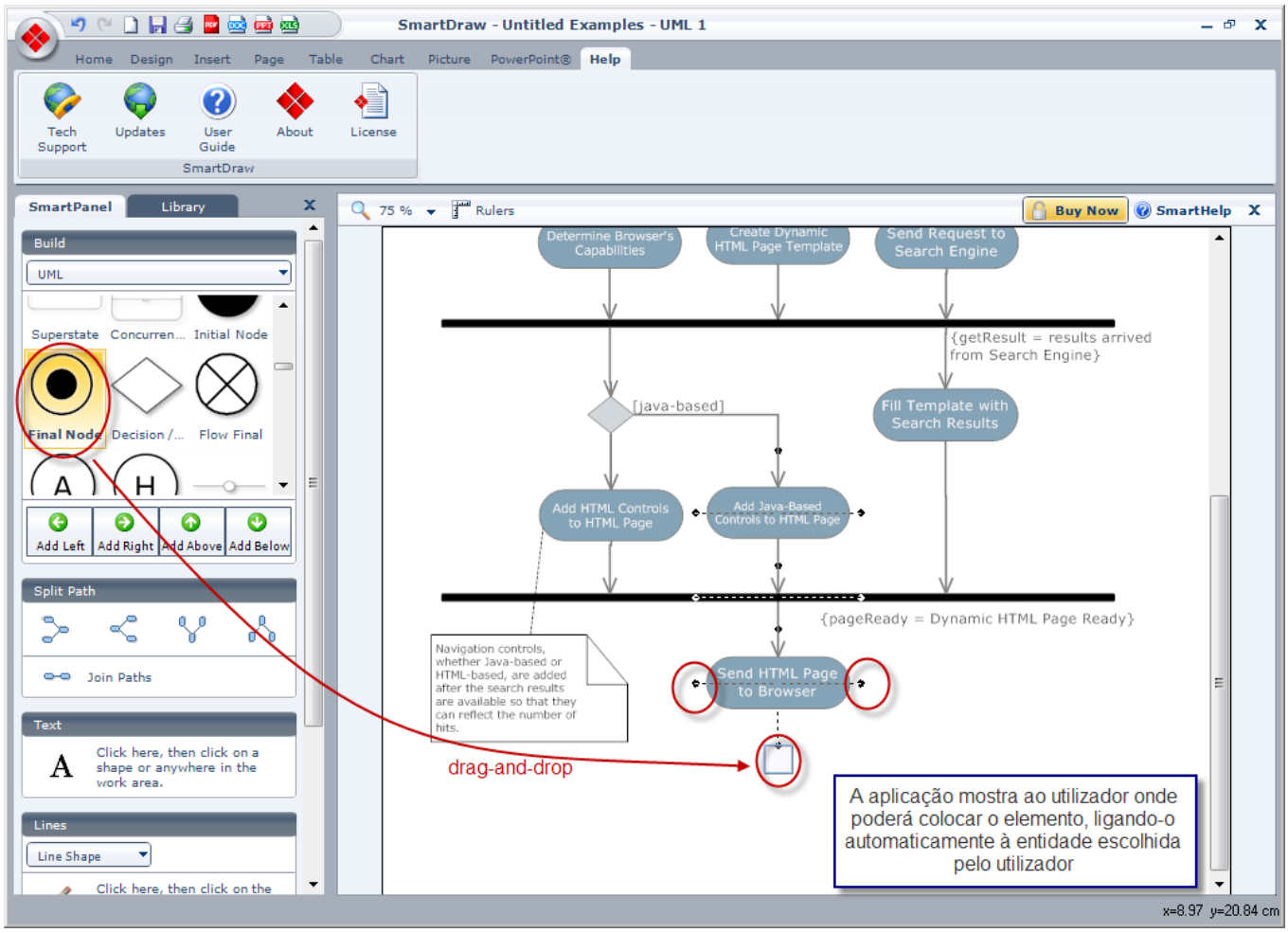


Figura 76 – SmartDraw – Arrastando objectos para o canvas

## Snagit Editor

A aplicacação Desktop Snagit permite capturar imagens do ambiente de trabalho. Esta aplicação possui um editor interno que pode ser utilizado para a edição das imagens capturadas.

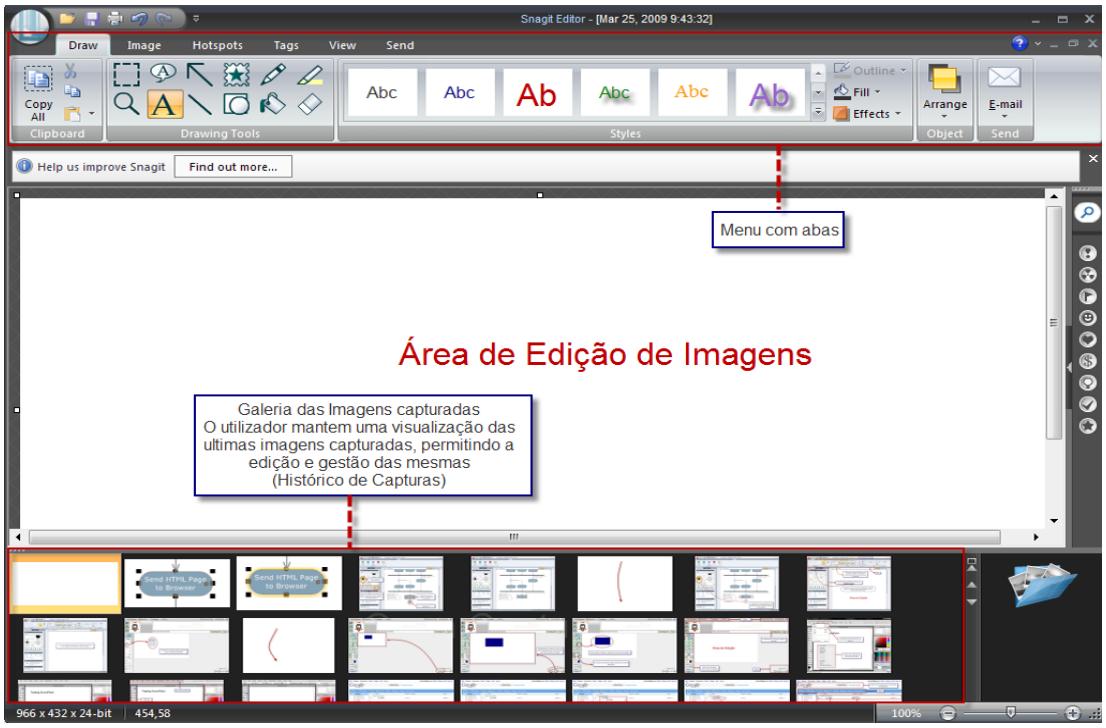


Figura 77 - SnagEditor ScreenShot

O editor de imagens do Snagit é muito simples, mas ao mesmo tempo eficaz, uma vez em que em apenas 3 áreas distintas consegue fazer sobressair as principais funcionalidades de qualquer editor.:

- Na zona superior apresenta as ferramentas habituais de edição
- Na zona central encontra-se a zona de edição das imagens
- Na zona inferior é disponibilizada ao utilizador a galeria das últimas imagens capturadas pelo utilizador.

Depois de estudadas as interfaces, de seguida serão elaboradas representações abstractas de cada uma das aplicações, utilizando para tal efeito os PAC's (Protótipos Canónicos Abstractos), abstraindo desta forma os detalhes e dando foco as principais componentes que constituem as interfaces. Abstraindo os detalhes das interfaces, é possível verificar como estas estão organizadas, dando foco às suas principais áreas.

## Microsoft Visio:

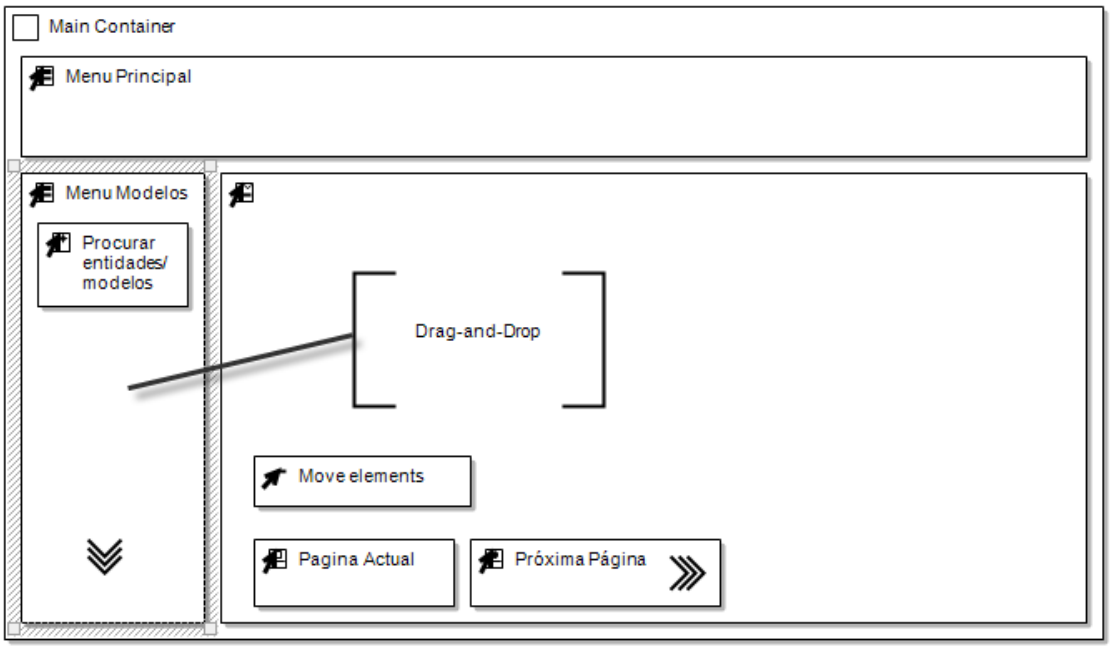


Figura 78 – Microsoft Visio - PAC

## Gliffy:

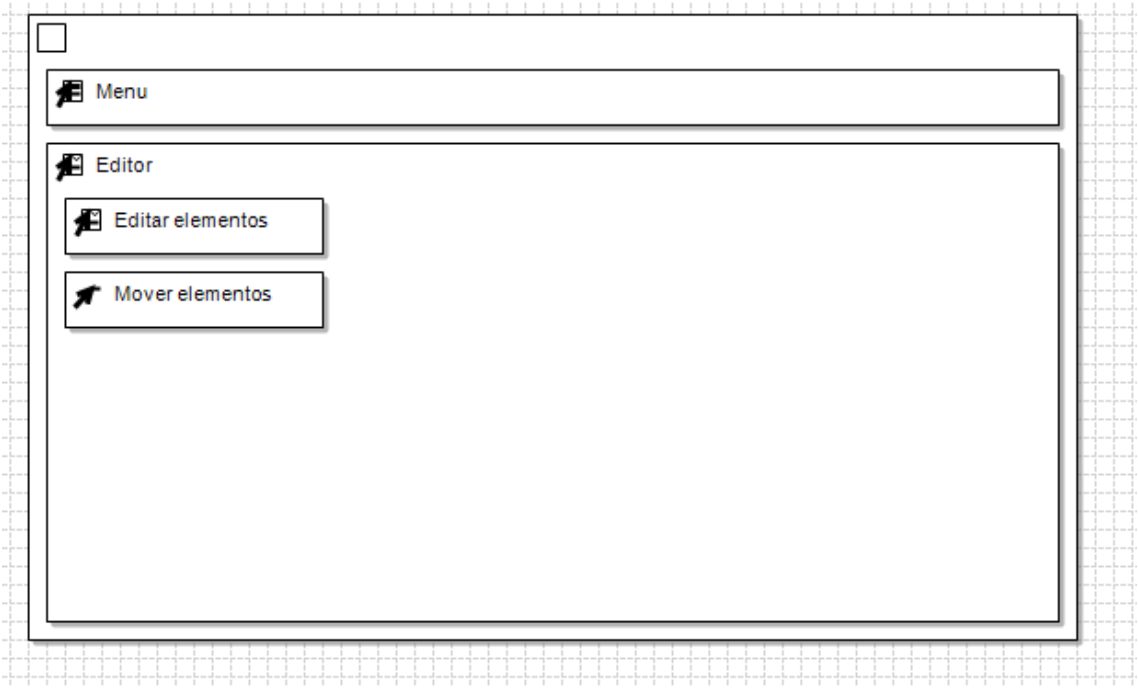


Figura 79 – Gliffy - PAC

## GModeler:

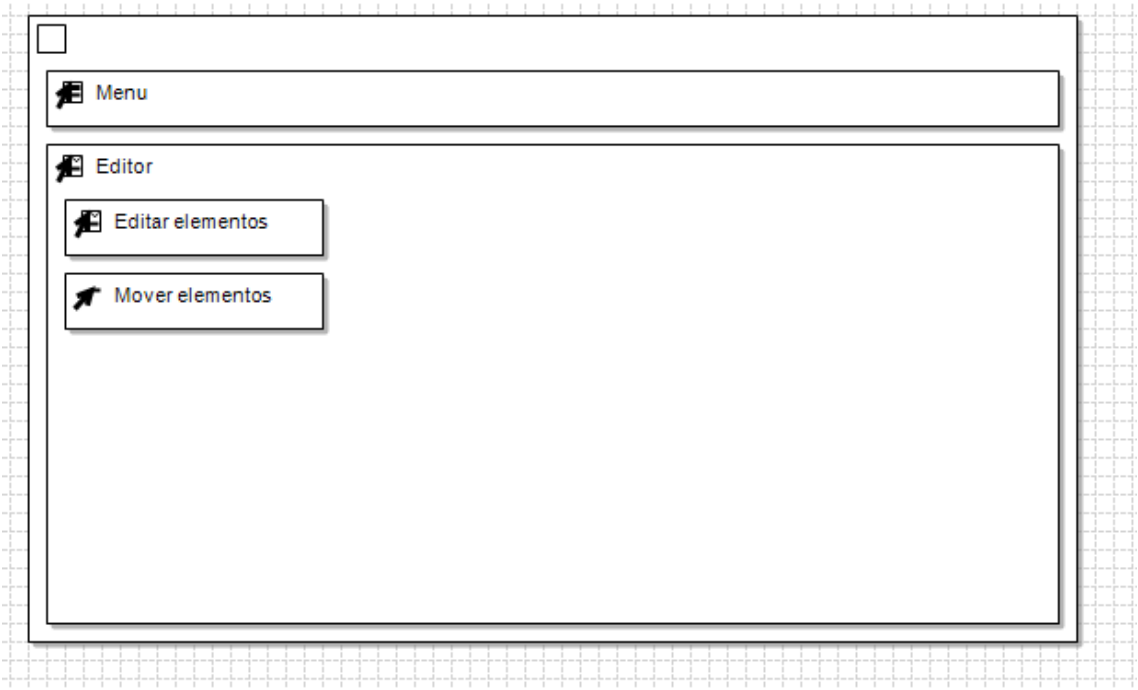


Figura 80 – Gmodeler - PAC

## MindMeister:

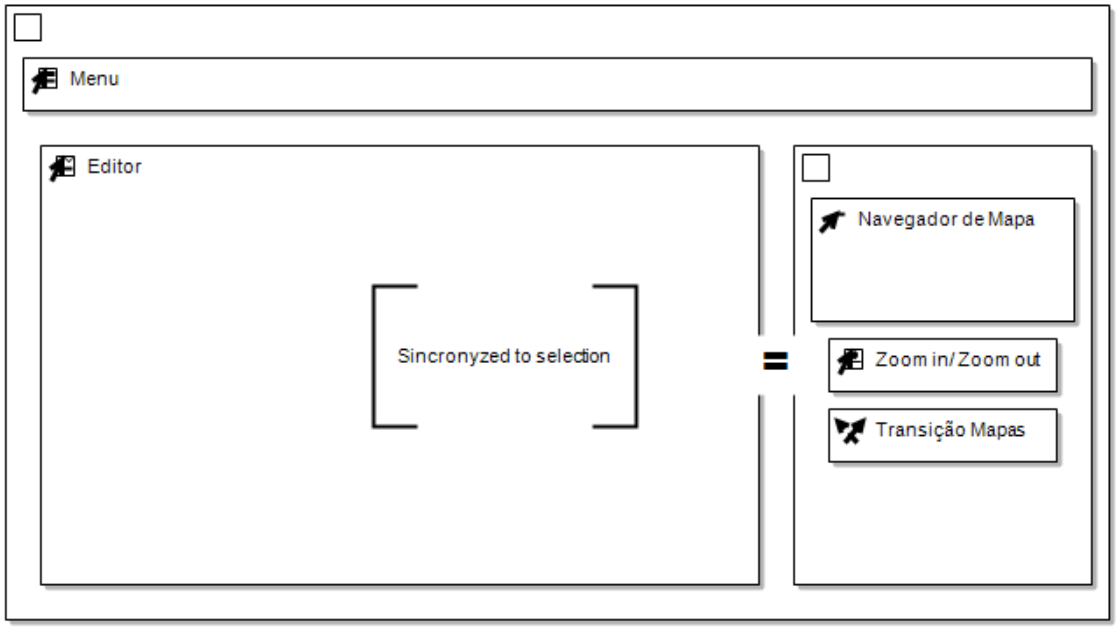


Figura 81 - MindMeister - PAC

## GoogleDocs Página Inicial:

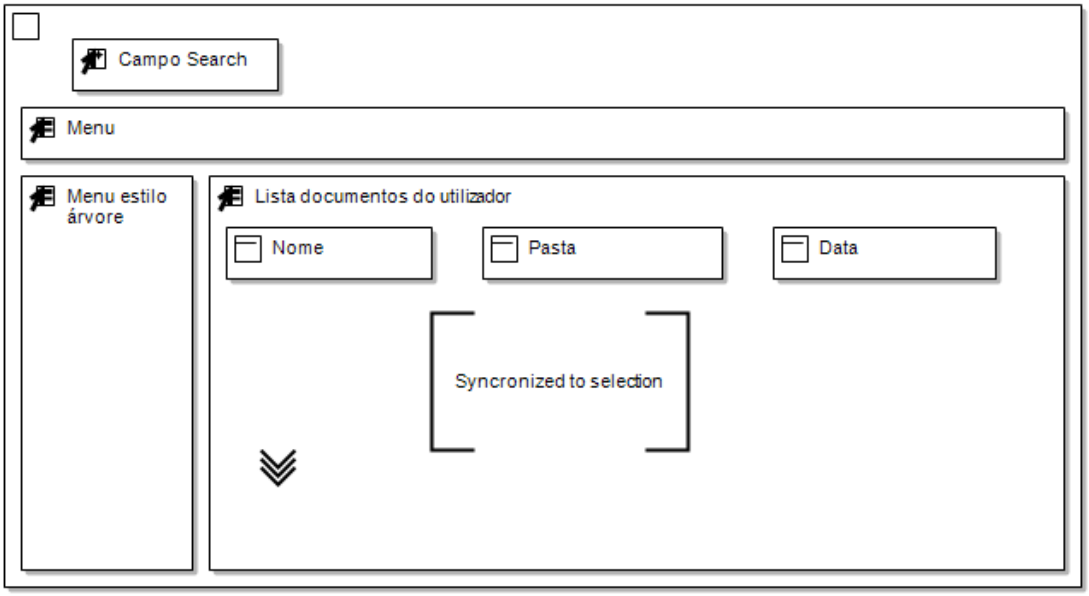


Figura 82 – GoogleDocs - PAC

## Google Editor:

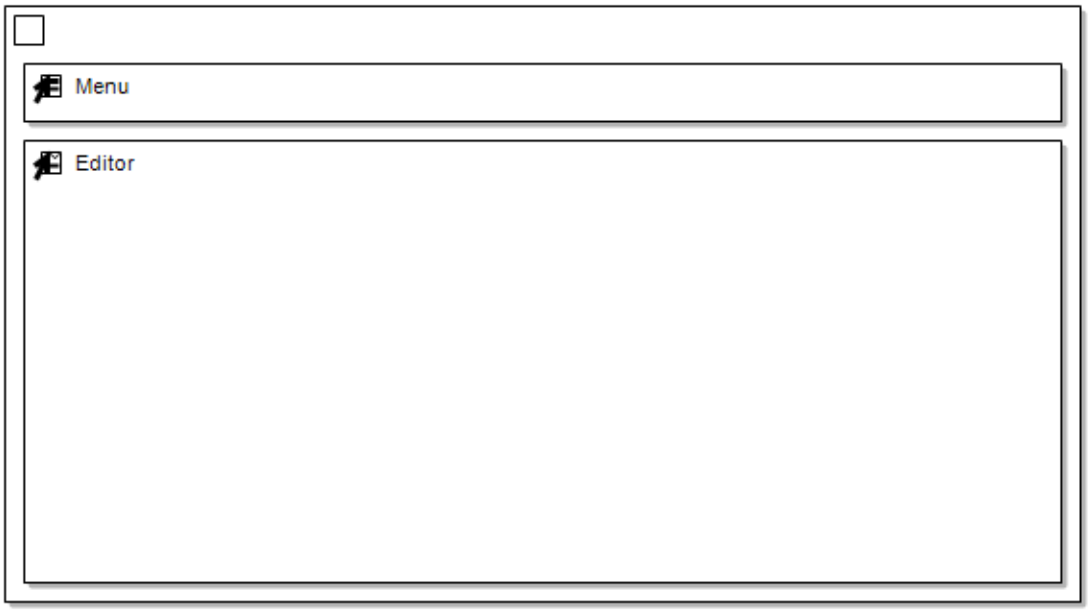


Figura 83 – GoogleDocs (página de edição de documentos) - PAC

## SumoPaint:


Figura 84 – SumoPaint - PAC

## TutorClass:

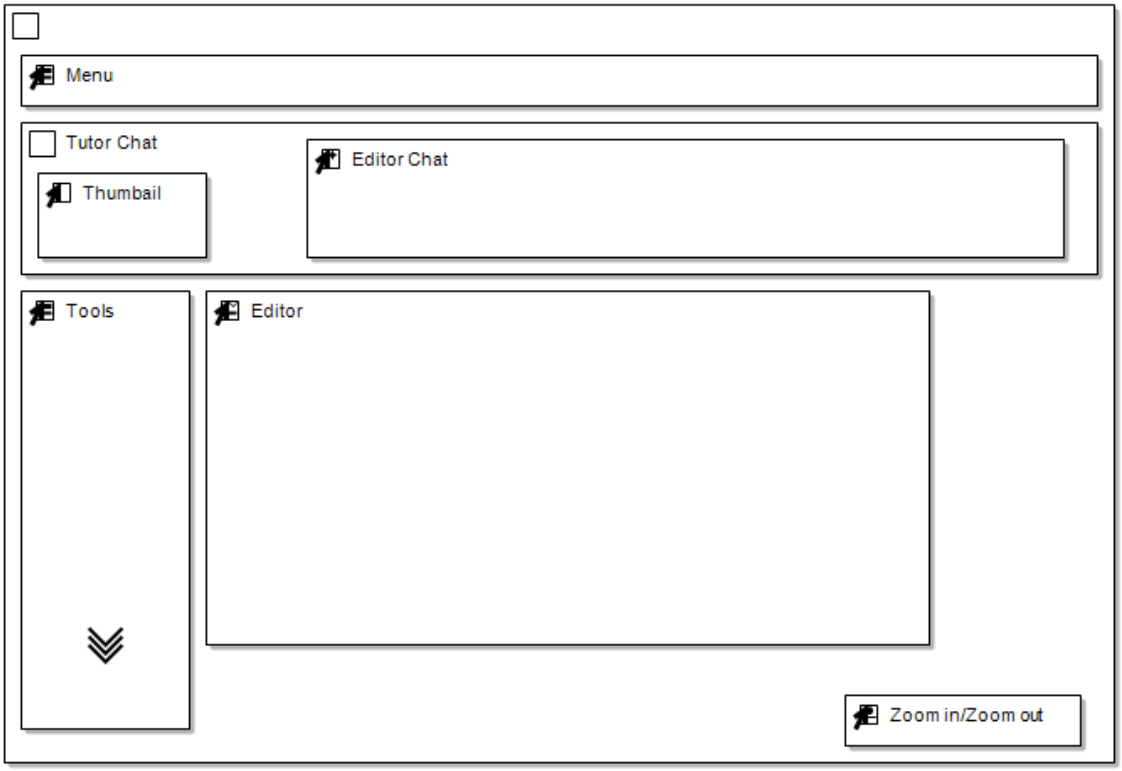


Figura 85 – TutorClass - PAC

## SmartDraw:

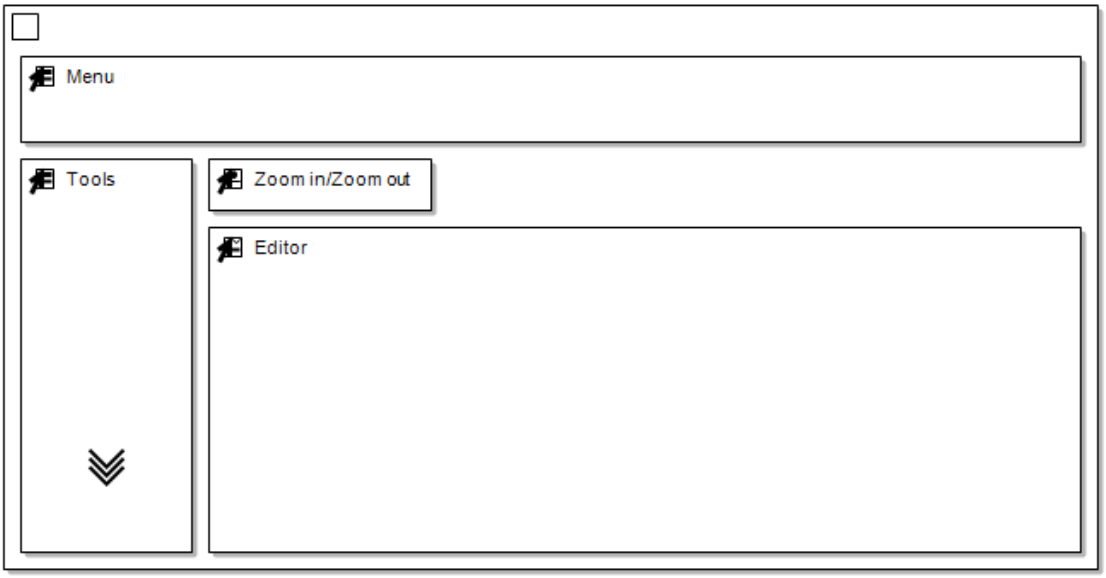


Figura 86 – SmartDraw -PAC

## Snagit:

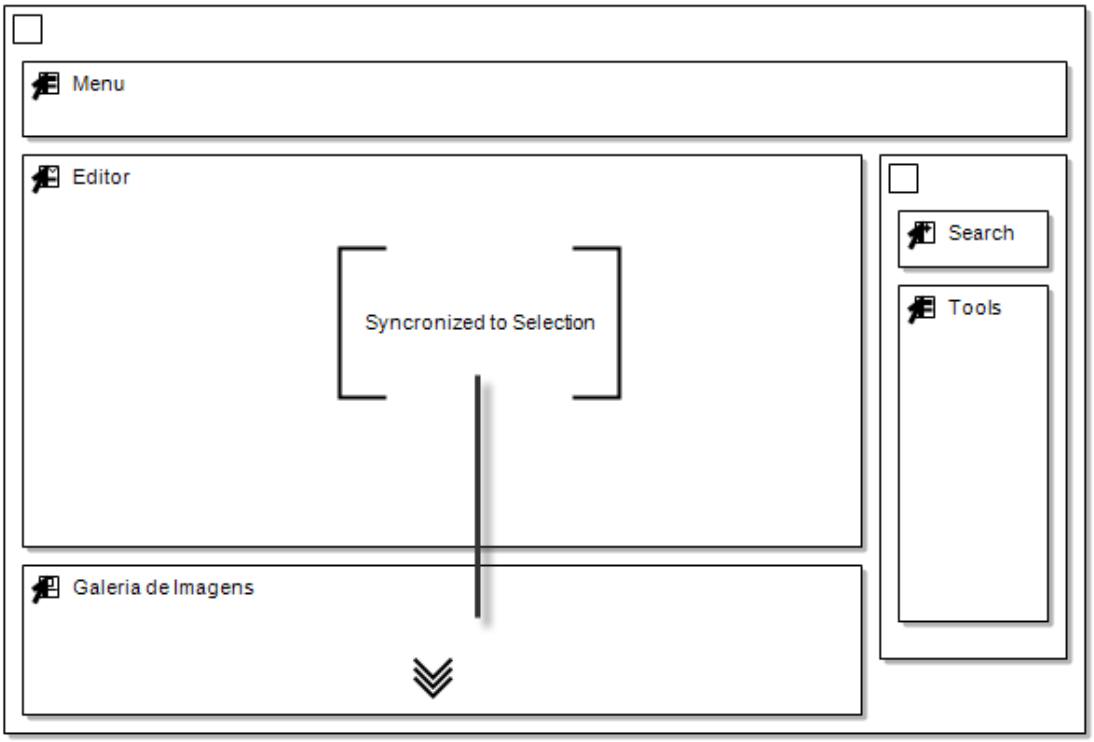


Figura 87 - Snagit

A representação abstracta da organização visual de cada uma das aplicações permite observar a existência de áreas comuns entre as diversas aplicações, nomeadamente:

- Na zona superior existe um menu onde o utilizador pode executar as mais diversas funções permitidas pela aplicação;
- À esquerda situa-se, na maioria dos casos outro menu contendo as ferramentas ou elementos que interagem directamente ou que estão directamente relacionados com o editor.
- Na zona central situa-se o editor, área principal de interacção com o utilizador.
- Algumas aplicações apresentam ainda na zona lateral direita ferramentas de interacção com o editor, informação relativa ao documento que está sendo editado ou ainda informação relativa aos elementos que estão sendo editados (propriedades dos elementos).

Depois das observações referidas é possível fazer uma representação do que é comum entre as interfaces das aplicações Desktop vs Web:

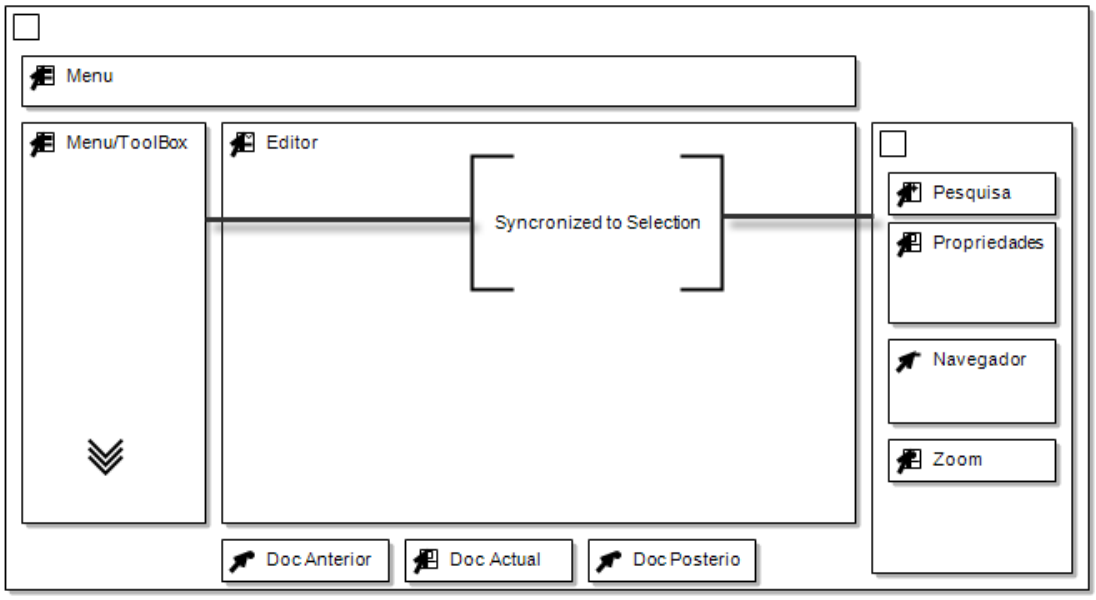


Figura 88 – Organização geral dos Editores